Stoner Pipeline Simulator (SPS) 9.8.0
Help and Reference

GL Noble Denton

Edition

This edition applies to the Stoner Pipeline Simulator (SPS) 9.8.0 software release version and to subsequent releases and modifications until otherwise indicated in new editions.

Trademarks

SynerGEE is a registered trademark and Stoner Software is a trademark of GL Industrial Services USA, Inc. All brands and product names are trademarks of the respective owner.

Restricted rights, warranties, and liabilities

The software discussed in this document is provided under a Software License Agreement and may be used or copied only in accordance with the terms of that license.

All warranties given by GL Industrial Services USA, Inc. concerning Germanischer Lloyd (GL) asset software are set forth in the Software License Agreement between GL Industrial Services USA, Inc. and the licensee.

GL assumes no responsibility for any errors that may appear in this document. We reserve the right to change our software and documentation without notice.

Use, duplication, or disclosure by the U.S. Government is subject to the restrictions defined as “Rights specified in the license” as set forth in subdivisions (a) and (b) of the DFARS clause 227.7202-3 entitled *Rights in Commercial Computer Software and Commercial Computer Software Documentation*.

Copyright notice

600 Bent Creek Blvd., Suite 100
Mechanicsburg, PA 17050 USA
+1 717 724 1900

www.gl-group.com

Table of Contents

Introduction to Stoner Pipeline Simulator

SPS Programs and Files

The SPS Environment

SPS Model Services Management

Configuring the OPC Client Service

Modeling the Physical System

Controlling the Simulation

Displaying and Printing Data

Validating, Troubleshooting, and Tuning

The INPREP File

The INTRAN File

Interactive Commands

User-defined Variables, Macros, and Include Files

Expressions, Operators, and Functions

Peek and Poke Keyletters and Attributes

Model Builder

Statefinder and Leakfinder

Trainer

Interfacing SPS to External Applications

Utilities

Quick Links

Glossary

CHAPTER

1

Introduction to Stoner Pipeline Simulator

The Stoner Pipeline Simulator (SPS) is an advanced transient hydraulic modeling service that simulates dynamic flow of a single fluid, batched fluids, or mixed fluids of a single phase through a pipeline. SPS reads text files containing detailed data that represents the pipeline (including pipes, pumps, compressors, valves, controllers, etc.). Using these text files, SPS constructs a mathematically sophisticated model. SPS can model operating characteristics of almost any proposed pipeline configuration and predict the outcome of various control strategies for operating scenarios such as pipe rupture, equipment failure, or other upset conditions.

SPS performs its simulations by calculating flow, pressure, density, temperature, and other variables at numerous locations along a pipeline model over time, and reports these values in the form of printed reports and graphs.

SPS simulates liquid (including column separation and slack line flow) or gas pipelines. It does *not* simulate two-phase flow systems. The simulation can be run either in batch mode (runs in the background) or in an interactive mode (responds to commands while the simulation runs). The initial state of the simulation can be a zero-flow condition, a user-defined steady-state condition, or a state saved during a prior simulation.

What can you do with SPS?

SPS can be used to solve almost any design or operating problem involving the analysis of the transient interactions of fluids and controls in liquid or gas pipelines. Some of the most common uses of SPS include the following:

- Analyze startup and shutdown procedures
- Analyze operational stability
- Analyze pump/compressor operating schedules
- Study economics of various designs and operational strategies
- Design sequencing systems
- Design surge relief systems
- Design cascade control systems
- Study survival time for gas delivery systems
- Analyze system response to potential upset conditions and evaluating corrective strategies
- Predict environmental impact of potential pipeline leakage
- Study the effects of batching, side stream delivery, or supply blending

- Study temperature rise in re-circulation systems and cooling or heating of products due to a transient interchange of heat with the pipeline surroundings
- Study thermal effects in gases, especially non ideal gases, such as Joule-Thompson cooling, decompression cooling, and inter-stage cooling required with polytropic compressors
- Design minimum flow bypass controls to prevent surge conditions in polytropic compressors
- Study the effects of pipe rupture, blow down cooling in gas pipelines to assess pipe steel brittleness, effect of pipe wrap and thermal environment in pipe burial on blow down rate
- Stop and restart models and check on model health, as well as ability to set flags for automatic restart of TRANS or shut down of TRANS to rerun PREPR for model updates.

In addition, module functions facilitate:

- Managing real-time predictive systems
- Training pipeline operators

SPS documentation

The following documentation is provided with SPS.

Installation Guide	Hardcopy information about system requirements, licensing, and installation of SPS.
SPS Help and Reference	Complete detailed information on model data preparation, methods and theory, and running SPS.
SPS Help and Reference, PDF	Fully linked and searchable PDF version of the help and reference that is suitable for printing.
SPS Tutorials (for Gas and Liquid)	Step-by-step tutorial for building and analyzing a gas or liquid model using SPS for Windows.

What is in this Help and Reference

The *Help and Reference* document, which you are currently reading, is divided into the following major sections:

- "Introduction to Stoner Pipeline Simulator" on page 39 provides an overview of the capabilities of SPS, the documentation set and conventions, and how to contact technical support.
- "SPS Programs and Files" on page 45 describes the individual programs (PREPR, TRANS, TPORT, and) that comprise SPS and also provides basic procedures for running these programs.
- "The SPS Environment" on page 83 describes the SPS windows, menus, and dialog boxes and provides an explanation of SPS settings.
- "Modeling the Physical System" on page 119 helps you to understand the options and types of modeling elements available for use in SPS and provides information on hydraulic modeling concepts.
- "Controlling the Simulation" on page 153 explains how to initialize, run, and control the model through batch commands or interactive commands.

- "Displaying and Printing Data" on page 163 describes options for retrieving data from the model and expressing it in useful displays, reports, and charts.
- "Validating, Troubleshooting, and Tuning" on page 193 helps you identify ways to improve model performance and accuracy.
- "The INPREP File" on page 205 provides a complete listing of the data entered in the INPREP file, which contains the definition of the physical model.
- "The INTRAN File" on page 425 provides a complete listing of all commands that may be entered in the INTRAN file, which allows you to control the model as a batch process.
- "Interactive Commands" on page 545 provides a complete listing of commands that may be entered interactively to control the simulation and the output of data.
- "Expressions, Operators, and Functions" on page 587 provides a listing of expressions, operators, and functions that you use within commands to control the simulation, process data, and control output.
- "User-defined Variables, Macros, and Include Files" on page 567 provides a listing of data processing commands that may be used in any of the input files.
- "Peek and Poke Keyletters and Attributes" on page 639 includes tables of keyletters and attributes that correspond to various devices or property sets. These keyletter and attribute combinations are used for a variety of purposes within SPS.
- "Model Builder" on page 697 describes the Windows-based graphical software product that aids in the development and maintenance of INPREP files.
- "Statefinder and Leakfinder" on page 741 provides information that helps you turn your off-line model into an online model.
- "Trainer" on page 877 describes how to set up and use a Trainer model.
- "Interfacing SPS to External Applications" on page 925 provides an overview of the various interfaces and servers available for SPS. Also provides a complete function reference and examples.
- "Utilities" on page 1043 provides information on the DEMAC and DRTU utilities.
- "Glossary" on page 1053 identifies and defines SPS-related terms.

Documentation conventions

In order to edit input files, you must thoroughly understand the documentation conventions. In addition, you should also become familiar with the syntax rules that govern the input files. For more information, see "Input syntax" on page 47.

- Entries within each line of data must be entered in the exact sequence as shown in the documentation.
- Input formats with no special formatting are variables for which you should substitute an appropriate entry.
- Bold characters must be entered exactly as shown in the documentation.
- Optional data is enclosed in square brackets [ ].
- Options in which you are given a choice are separated by | . For example, `ECHO=YES|NO`.
- In-text references to input entries may be surrounded by angle brackets < >.
- File names and files extensions use the following conventions:
    - File extensions are represented in all capital letters. For example, the INPREP file.

- File paths and names have an initial capital letter only.

Example Documentation Conventions

As an example, the following input format describes the input for the pipe parameters.

```
PIPEPARMS
[+ FRICTION [COLE | NIKUR | MOODY] [RUF | FF]]
[+ INITIAL PINIT]
[+ KNOT SPAC]
[+ THRM.COEFF n]
```

This example shows actual pipe parameters input:

```
PIPEPARMS
+ FRICTION MOODY 0.015
+ INITIAL 700
+ KNOT 2
```

In this example, the bold entries PIPEPARMS, + FRICTION, + INITIAL, and + KNOT have been entered exactly as shown. Plain text means a value needs to be substituted for the field; therefore, the values for FF, PINIT, and SPAC have been entered. Entries enclosed in square brackets [ ] are optional and may not have been included, such as +THERM.COEFF. Options are separated by | . In this example, MOODY was chosen from among COLE, NIKUR, and MOODY.

Displaying the SPS version number

To display the SPS version number

From the SPS startup window, select **Help > About SPS**.

Tip: The About Trans/Tport window also displays the current SPS version. You can access this information from the Trans/Tport Window by selecting Help > About Trans/Tport.

Copyright information

Copyright notice

600 Bent Creek Blvd., Suite 100
Mechanicsburg, PA 17050 USA
+1 717 724 1900

www.gl-group.com

Edition

This edition applies to the Stoner Pipeline Simulator (SPS) 9.8 software release version and to subsequent releases and modifications until otherwise indicated in new editions.

Restricted rights, warranties, and liabilities

The software discussed in this document is provided under a Software License Agreement and may be used or copied only in accordance with the terms of that license.

All warranties given by GL Industrial Services USA Inc. concerning GL asset software are set forth in the Software License Agreement between GL Industrial Services USA, Inc. and the licensee.

GL Industrial Services USA, Inc. assumes no responsibility for any errors that may appear in this document. We reserve the right to change our software and documentation without notice.

Use, duplication, or disclosure by the U.S. Government is subject to the restrictions defined as “Rights specified in the license” as set forth in subdivisions (a) and (b) of the DFARS clause 227.7202-3 entitled *Rights in Commercial Computer Software and Commercial Computer Software Documentation*.

Trademarks

StonerSoftware is a trademark of GL Industrial Services USA, Inc.

Microsoft, Access, Excel, Internet Explorer, Visual Basic, Windows, and XP are registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Other brands and product names are trademarks of their respective owners.

Contacting GL

If you need to contact the GL Stoner Software Technical Support staff about a software-related question or problem, you will want to have up-to-date GL contact information and be prepared to provide the version of the software you are running.

To display GL contact information

1. From the main menu, select **Help > Technical Support**.
2. The Technical Support dialog box shows supporting offices’ telephone numbers, fax numbers, and office addresses.

To email GL Technical Support

Note: In order to access this Technical Support feature, you need to have a MAPI server running. Typically, the MAPI server is set as the default for many email applications. Different email programs have different means for setting the MAPI server. Consult your system administrator for more detailed information on setting up your specific email system as the default MAPI server or client.

From the **SPS startup window**, click the SPS Email button.

—or—

Select **Tools > Mail**.

A new mail message to GL Technical Support (software.support@gl-group.com) opens in your email program.

CHAPTER

2

SPS Programs and Files

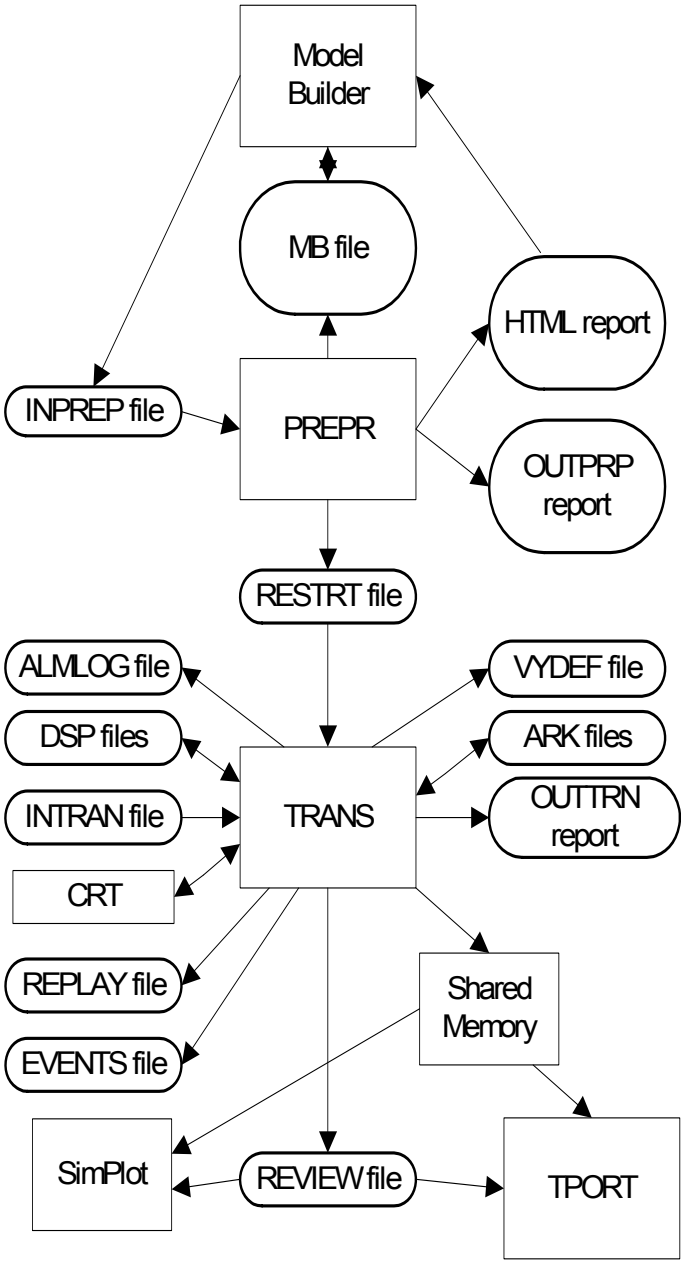
SPS is a collection of several different programs, as well as additional modules, utilities, and servers that supplement the capabilities of SPS. Each program requires input files with specific requirements and formats. Each program also generates output files used for diagnostics or as input for other SPS programs.

Overviews of the main SPS programs — PREPR, TRANS, TPORT, and SimPlot— are provided in the topics listed below. More detailed information on the use of these programs are provided throughout this *SPS Help and Reference* document. Modules and utilities are explained in the documentation provided with each module or utility.

The main SPS programs are:

- *PREPR.* The pre-processing program. See "Overview of PREPR" on page 53.
- *TRANS.* The transient simulation program. See "Overview of TRANS" on page 55.
- *TPORT.* The transport program. See "Overview of TPORT" on page 57.
- *SimPlot.* The Windows-based post-processing program. See "Overview of SimPlot" on page 191.

The following "System Flow Diagram" shows the process for importing files to the various SPS programs and the resulting output. On Windows systems, you may access these programs from a Model Builder window or from the SPS startup window.



System Flow Diagram

SPS files

SPS uses and generates numerous files. Input files are required for processing by the various SPS programs, while output files include information that is generated by running these programs.

Input files

A complete SPS model requires several types of files, which are processed by the various SPS programs. You create these input files differently, depending on your environment and the type of data you need to enter. For more information on specific input files, see:

- "The INPREP File" on page 205
- "The INTRAN File" on page 425
- "Overview of SimPlot" on page 191

Input files may also "include" one or more additional files to manage file size and create scenarios. For more information on include files, see "INCLUDE" on page 573.

Output files

When you run some SPS programs, SPS produces output files or reports that help you identify input errors or unacceptable initial operating values. Information in the output files may amplify error messages. Each time you run a program, you should review the error messages and the output files and then edit the input files and run the program again. Repeat this process as often as necessary to produce an acceptable working model.

See the discussion on each program for a description of the output files.

In addition to diagnostic output files, SPS also generates files that may be required or used in other programs. For more information, see "Relationship of TRANS and related files" on page 55.

File naming conventions

SPS uses and generates many files. Generally, you should use the same root name for all input and output files to avoid confusion. For example, if your model is named "MyModel," you would use files such as MyModel.inprep, MyModel.restrt, and MyModel.intran.

By default, SPS modules give all output files the same root name as the input file(s). You can create a model case, which allows you to designate the names and locations of all files involved with a particular model. For more information on case files, see "Cases" on page 99.

Input syntax

If you are using a text editor to enter model data, you must follow a required syntax. Input files use a free-field format that allows specially sequenced data to be inserted within each line, without having to conform to rigid spacing guidelines. This section describes the syntax guidelines for INPREP and INTRAN.

Device keyletters

To simplify command syntax, SPS uses keyletters to represent supported device types. For example, the keyletter "T" represents a transfer line. When a command parameter requires a device type, generally the keyletter is the required value.

A full list of device keyletters and attributes is available in "Peek and Poke Keyletters and Attributes" on page 639.

Spacing

The following spacing conventions apply to all input files.

- At least one blank space must be inserted between entries on the data line.
- You may indent input for clarity.

Continuation

SPS requires specific syntax for continuing input from one line to the next in input files. The syntax varies depending on which input file you are working on and whether the input line is a command or part of an expression.

General rules for continuation

Each line may contain up to 256 characters (columns). Usually, you may enter multiple lines of input to prevent long strings. In general, the rules for continuations are as follows:

- The line must be split between fields.
- In INPREP and INTRAN, a plus sign (+) in the first column followed by at least one space denotes a continuation.
- The continuation line may follow any number of comment lines.

Continuation lines are not supported for interactive commands. Enter all data related to the command on a single line.

Continuation of expressions

- Those commands that have command lists (e.g., IFELSE, DEFINE.SEQUENCE, WHENEVERs) do not use the plus sign convention for continuation. Each command is given on a separate line.
- If a MACRO argument list must be split, the end argument must be followed by a comma and the first argument on the succeeding line must be preceded by a comma. Any input in the MACRO list is read as an argument including line feeds. Therefore, you should specify an additional argument to hold the line feed and white space up to the comma on the next line. Care should be taken that the continuation breaks are consistent between the MACRO definition and the short form MACRO call. For more information, see “MACRO” on page 580.
- In the argument list for a macro definition, line breaks are ignored and no special continuation characters should be used.

```
MACRO(MY_MACRO(A,
               B), A B)
```

- In the argument list when a macro is used (also called the “short-form” macro call), line breaks are preserved as-is so that:

```
MY_MACRO(HELLO,
         WORLD)
```

expands to

```
HELLO
WORLD
```

Comments

Comment lines may be included anywhere in an input file after the title. Comments are used to insert explanations, provide line spaces for readability, insert labels, etc. A /* may appear anywhere on a line, signifying the beginning of a comment. Any text appearing after the /* is a comment and is ignored. The comment ends with the line; it does not continue to the next line. For example:

```
/* A comment can contain any combination of blanks and characters
ANY INPREP COMMAND /* Comments go to the end of the line
```

Default values

SPS uses the default value for numeric entries if you enter an asterisk (*) in a data input field. For backward-compatibility, SPS also interprets a zero as a request for the default value; however, you should avoid using this convention. SPS now issues a warning message before changing a zero to a default value other than zero.

Default values are usually documented, where applicable, in the description of a field and are enclosed in curly brackets { }. If you want to change the system defaults, use the SET.LIMIT command. See "SET.LIMIT" on page 219.

Numerical fields

SPS interprets every numeric entry as either an integer or a real number. If no decimal point appears, it is assumed to be to the right of the last digit in the entry. Scientific notation (e.g., 10E-8) and hexadecimal notation are also supported.

Element and node names

- Names may begin with either alphabetic or numeric characters; however, it is strongly recommended to begin with an alphabetic character.
- Numerals, alpha characters, and special characters (. _ $ #) may be used.
- Names cannot contain embedded blanks.
- Upper and lower case characters are treated as identical (a = A). The following are examples of acceptable names:

  PIPE1234

  P1234

  P_1234

  PIPEPIPEPIPE2

  PIPE.1234
- Names do not have a limit to the number of characters in them.

Case-sensitivity

INPREP data can be either upper case or lower case. INTRAN data must be upper case.

Abbreviations

Keywords may be abbreviated by not including the last letters of the keyword or the last letters before a period in a keyword. A keyword may be abbreviated to any length as long as the keyword is still unique. For example, DENSITY may be abbreviated as DENS or DE. Furthermore, HEAT.CAPACITY may be abbreviated as HEAT.CAPA, HEAT.CA, or HE.CA.

Wildcards

Many commands support a wildcard syntax that allows you to include multiple devices, attributes, or other items with a single statement. A wildcard character can act as a substitute for zero, one, or more characters of unspecified values. The wildcard characters supported by SPS are as follows:

Character	Equivalent value
Asterisk (*)	Any size string of characters, including zero characters.
Question mark (?)	Any single character.

For example, consider the following set of attributes associated with nodes:

- SLHV
- NLHV
- HHV
- SHHV
- NHHV

Assume that you are using a command that changes attribute values, such as POKEALL, and you want to include several attributes without having to type each one. The following table demonstrates some ways that wildcards would affect which attributes from this list would be included:

Parameter	Attributes that would be included
S*V	SLHV, SHHV
?LHV	SLHV, NLHV
NL?V	NLHV
N?HV	NLHV, NHHV
*HHV	HHV, SHHV, NHHV
S*V	SLHV, SHHV
*V	(all)
*	(all)

Using wildcards for peek names (PEEK.MATCH fields)

Certain commands, such as POKEALL and TRENDLIST, allow you to narrow down the list of specific device attributes (or *peek names*) that are affected by the command. Because a peek name includes both a device name and an attribute

separated by a colon, any logical wildcard expression for a peek name should also contain a colon. Consider the following examples, assuming that they are arguments in a POKEALL command:

POKEALL argument	Effect
`+ PEEK.MATCH NODE1:*`	All attributes of NODE1 are poked.
`+ PEEK.MATCH *:P?`	All two-character attributes that begin with “P”, for all the device names previously included in the command, are poked.
`+ PEEK.MATCH *N*:*P*`	All attributes that contain at least one “P”, for all device names that contain at least one “N”, are poked.

You should note the following wildcards and peek names:

- Due to the power and flexibility of wildcards, you should use special caution when designing those expressions. Improper use could quickly cause widespread, unintended alteration of data.
- Failure to use a colon in a peek name wildcard expression could cause unexpected results. Because you can achieve any pattern of matching using expressions with colons, it is recommended that you include a colon in all your peek name wildcard expressions.

Wildcard example

The following POKEALL command would poke the PMAX attribute of all externals in the model:

```
POKEALL *,
+ KEY.LETTER = E,
+ PEEK.MATCH = *:PMAX,
+ TO = 1200
```

CSV input syntax

The CSV format was introduced in SPS 9.6 for select INPREP and INTRAN commands, as listed below. The new format is being offered as an alternative to the “traditional” input format. Either format can be used, which ensures that you do not need to convert old data into the new format. However, data in the new style can be easily saved in separate files as CSV files and then edited with Excel. Excel will automatically divide the data into rows and columns, based on the location of the comma separators.

Note: Excel will fill out each row with trailing commas when saving back to CSV format. These trailing commas will appear in rows such as the “TABLE” row and the “ { ” and “ } ” rows. You should note that SPS will ignore any trailing commas, so you do not need to worry deleting them from your input files.

The CSV input format is used with the following commands. Examples of the usage of this input format can be found in each of the sections referenced below.

- INPREP
  - “VISCOSITY (non-Newtonian)” on page 239
  - “STATE TABLE” on page 245
  - “Compressor TABLE” on page 335
  - “Data curve (D)” on page 418
- INTRAN:
  - “DEFINE.FUNCTION” on page 444
  - “LINE.FILL” on page 465
  - “RAMP” on page 504
  - “SAVE.LINE.FILL” on page 512

Input format

The input format for the CSV input format is as follows:

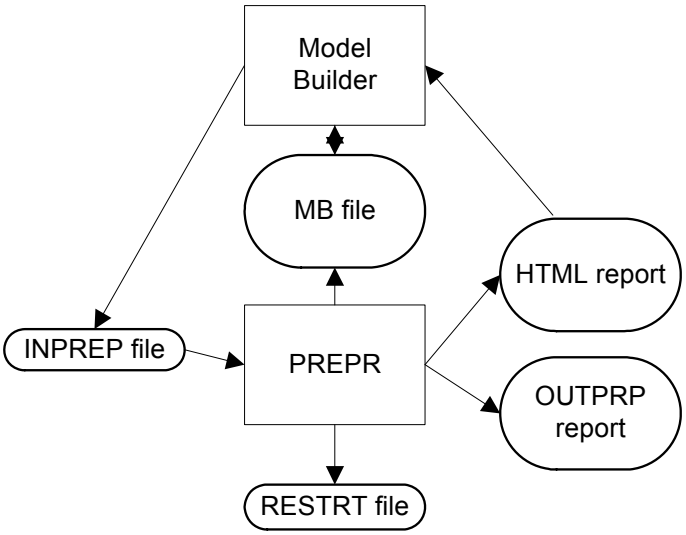
```
... TABLE
{
header-row
data-row-1
data-row-2
.
.
.
data-row-n
}
```

...	Context in which the table is being used.
TABLE	Required input; must be entered exactly as shown above.
{	Required input; must be entered exactly as shown above.

header-row	List of attribute names, separated by commas and/or spaces. The appropriate attribute names depend on the context. Additional data can be entered using a column label of “IGNORE”; the entire column is ignored (each cell must be alphanumeric, with no special characters or intermediate spaces).
data-row	List of numeric or text values, separated by commas and/or spaces, that correspond to the attributes in the header row. Fields that are entirely blank (adjacent commas) default to the corresponding value from the row above.
}	Required input; must be entered exactly as shown above.

Overview of PREPR

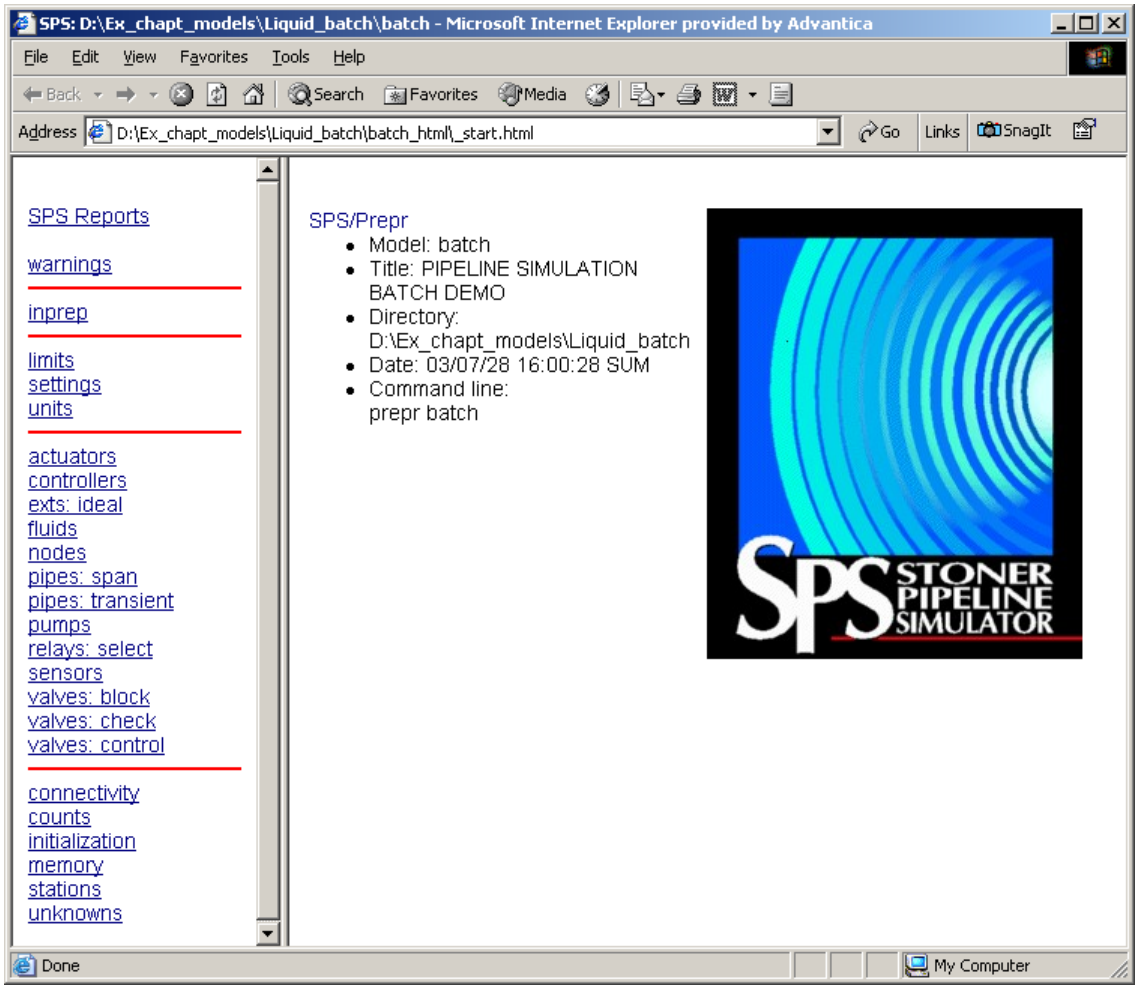
PREPR processes all data needed to define the configuration of the pipeline and calculates the initial state that can be used as the starting point of a transient simulation. PREPR requires an input file called the INPREP file. When PREPR is run, it produces an “OUTPRP report”, which provides diagnostics to help you resolve input errors. PREPR also produces a binary file called the RESTRT file, which is used for input to the transient simulation. For more information, see “HTML OUTPRP reports” on page 53.



PREPR Flow Diagram

HTML OUTPRP reports

When you run PREPR, SPS creates an HTML OUTPRP report that contains summary information on the INPREP file, error messages or warning messages, and the ability to browse the report through hyperlinks. An OUTPRP file is also generated, but most of the reports have been converted to HTML.



OUTPRP reports are stored in a folder in the model directory called *modelname*_html and are overwritten each time your run PREPR on the same model. In the folder, open the file _start.html to open the report on the starting page.

These HTML files will display in your default browser.

You should keep the following tips in mind when working with OUTPRP reports:

- If you are not using Model Builder, for best results, you should be sure your browser options are set to refresh every time you visit the page. For example, in Internet Explorer, select **Tools > Internet Options**. In the Internet Options dialog box, click the **General** tab. On the General tab, click **Settings** under Temporary Internet Files. In the Settings dialog box, under Check for newer versions of stored pages, choose **Every visit to the page**.

  To be sure you have the latest report, you may need to refresh the page each time you run PREPR.

- In you want to print an OUTPRP report that you are viewing in Internet Explorer, you may want to consider selecting File > Print Preview first, and then clicking the Print button that appears with the Preview window. Because OUTPRP reports use an HTML feature called frames, the File > Print option will only print the active frame. It will not print both frames in the report—the content frame on the right *plus* the links frame on the left. The easiest method to print the entire report is to select a Print Preview first, and then print the report.

Overview of TRANS

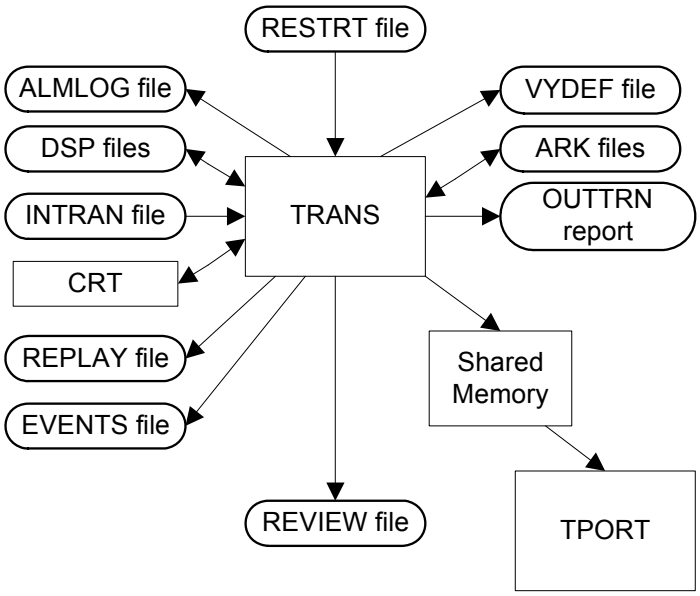
TRANS uses an initial state and the simulation control data to perform the transient simulation. The transient simulation is a complex series of calculations that predict the dynamic response of numerous variables all along the model. You can run the simulation in batch mode by specifying commands in the INTRAN file or you can enter commands interactively, changing variables and statuses at any time during the simulation. Additionally, you can build and review time and distance plots of any variable. TRANS produces output in several forms that show the values of important variables for each model component at requested intervals during the transient simulation. For more information, see:

- "Relationship of TRANS and related files" on page 55
- "Modes of running TRANS" on page 55
- "The INTRAN File" on page 425

For more information on creating an initial state for a transient simulation, see "Preparing to run a simulation" on page 153.

Relationship of TRANS and related files

The diagram illustrates a system flow diagram of TRANS and its associated files. A description of each of the files that TRANS uses can be found in the "Glossary" on page 1053.



TRANS System Flow Diagram

Modes of running TRANS

A TRANS simulation can be run in any of the following ways:

- Batch job
- Interactively

- Combination

In *batch mode*, all controls are specified in the INTRAN file and the simulation runs to completion. The results are interpreted; and if necessary, some simulation parameters are changed and another run is made. All simulation control data is entered into the INTRAN file. There is no user interaction or control during the simulation. The simulation runs for the entire length specified in the INTRAN file. Once the simulation is finished, you can run SimPlot to view and print the results of the simulation, or alternately, you can run TPORT to review the results interactively. After examining these results, you may decide to change equipment design or operating strategies and re-run the simulation, eventually working out a design or strategy that gives the desired results.

In *interactive mode*, much like operating an actual pipeline, you enter commands to make operational changes as the simulation runs. You can watch as the simulation proceeds, changing appropriate variables (e.g., set points), starting/stopping pumps/compressors, changing the status of valves, etc. at any time. Any computed variable may be displayed at each simulated time step at a location you have designated on a display.

When running interactively, you have total control over the simulation from start to finish. Using TRANS in interactive mode, you can:

- Start or stop the simulation at any time
- Archive a snapshot of the model state at any time
- Initialize the simulation from a previously archived state
- Observe how the model responds during a simulation
- Interact with the simulation (change pokable values)
- Create custom plots and reports, and save for later use
- Obtain output results at any time

When you use a combination of batch and interactive, you may define simulation control commands in the INTRAN file or you may enter instructions interactively, as needed.

For more information on the commands available in the INTRAN file and interactively, see “INTRAN file input” on page 426 and “Interactive Commands” on page 545.

Preparing to run TRANS interactively

To run TRANS interactively, the INTERACTIVE command must be present in the INTRAN file.

In addition, you can make the most of your interactive simulation by:

- Creating custom text displays to view the output in a desired format. See “Text displays” on page 186.
- Specifying reports, time plots, distance plots, and Show windows. See “Displaying and Printing Data” on page 163.
- Creating MACROs in the INTRAN file to abbreviate frequently-used interactive commands. See “Expressions, Operators, and Functions” on page 587.

Shared memory

Shared memory allows you to access TRANS data from a TPORT window or other module. Shared memory is created and managed based on the required settings that you specified to run TPORT or another module. In addition, you may request data used for distance plots, Show windows, and reports. Specifically, shared memory contains:

- Data that you indicate you want to share with TPORT or another module through the SHARE command. For more information, see “SHARE” on page 524.
- Data you want to make available for time plots through the TRENDLIST command. For more information, see “TRENDLIST” on page 537.
- Distance plot data
- Data used for Show windows and reports

While TRANS allows immediate access to all variables, you have limited access the data through shared memory. Access is limited both by the speed with which information may be accessed and the amount of data that can be accessed.

- It takes three time steps for requested data to be placed in shared memory and to be read by TPORT (or another module). **Not Available** displays in place of the variable values until the data becomes available.

  Tip: For small models, TRANS may be running too fast for TPORT to get information. If this is the case, you can enter a POKE command that pauses SPS in between time steps. Enter `POKE DELAY = .01`. See “POKE” on page 493 for more information.
- There is a specific amount of memory available for additional show data and for distance plot data. Once data is accessed, it remains available in these areas of shared memory for 20 time steps after it has no longer been viewed. If it has not been accessed again after twenty time steps, it is removed from shared memory and may no longer be accessed. These areas of memory may fill up, at which time, a message informs you to wait 20 time steps for additional memory.
- Model variables not specifically listed in the SHARE command may not be available, depending on the amount of shared memory available.

You can view statistics on shared memory with SHOW SHARE. For more information, see “Shared memory (SH) attributes” on page 685. If you have insufficient shared memory for your needs, you can override the values determined by TRANS using the environment variable SAISHAREDMEMEORYSIZE. For more information, see “SPS environment settings” on page 87.

For more information on streamlining CPU processing, see “TPORT CPU use” on page 196.

Overview of TPORT

TPORT allows you to work with multiple representations of a running model either locally or over a network. You may use TPORT to provide additional interactive sessions for a running TRANS model, or you may connect to a “cold” REVIEW file to review the results of a previous simulation. Unlike other SPS programs, TPORT does not have an input or an output file, but TPORT uses a cache of memory called *shared memory* to make the TRANS data accessible in the TPORT window. For more information, see “Shared memory” on page 57.

All standard interactive commands may be issued from TPORT. For a list of interactive commands, see “Interactive Commands” on page 545.

For more information on running TPORT, see “Running SPS programs from command line” on page 62 and “TRANS/TPORT window” on page 85. Also see “TRANS System Flow Diagram” on page 55 to understand the architecture.

Preparing to use TPORT

In order to run TPORT, either locally or over a network, you need to specify the following settings in your INTRAN file.

- “TRENDLIST” on page 537
- “SHARE” on page 524

If you want to use network TPORT, you will need to specify additional settings. For more information, see “Network TPORT” on page 58.

Network TPORT

Available on Windows only, network TPORT allows you to view and work with additional simulation windows that are available over a network. This feature allows you to use a remote computer to access an active TRANS simulation, running on a computer elsewhere on a local area network.

Network TPORT provides identical features to the traditional “local” TPORT functionality, which allows you to open an additional simulation window on the computer that is running SPS. The difference with Network TPORT is that you may now use another computer on your network to open additional simulation windows via TPORT.

Network TPORT requires some additional setup beyond the instructions provided in the *Installation Guide*. It also requires a special license, which you can obtain from Technical Support for no additional charge. For more information on TPORT, see “Preparing to use TPORT” on page 58.

Network TPORT uses at least two computers:

- A *Server* machine where the main model is stored and the main license of SPS is located. This can be any computer with SPS installed and not necessarily a network server.
- One or more *Client* machines through which the simulation is viewed and controlled remotely.

In order to use network TPORT, you need to:

- Install the appropriate versions of SPS on the Client and Server machines.
- Set up the Server and Client to use the same Lexicon Service.
- Modify the INTRAN file to run network TPORT.

Setting up the Server and Client machines for network TPORT

On the Server machine, you should install SPS and related software products as you would for typical, standalone use, as described in the *Installation Guide*. You do not need to install any additional software.

For the Client machine, you should follow the basic instructions provided in the *Installation Guide*, except substitute the special license you need to request from Technical Support. Do not use the normal license that you received with SPS.

To set up the Server and the Client to use the Lexicon Service

The following procedure must be performed on both the Client and Server machines.

1. From the **Start** bar, select **Start > Settings > Control Panel**.
2. On the Client machine, click on **My Computer** and select **Properties**.
3. On the **Advanced** tab, click **Environment Variables**.
4. In the Environment Variables dialog box, under **System variables**, double-click **SPS_LX_HOST**.

   —or—

   Click **New** to add a new SPS_LX_HOST.
5. The default vale for SPS_LX_HOST is `lx:localhost:10403`. Change `localhost` to the name of the Server PC. For example, to have both the Server machine and the Client machine use a Lexicon Service on a machine named MyPC, you would set SPS_LX_HOST on both PCs (the Server and the Client) to `lx:MyPC:10403`.

Preparing an INTRAN file to run network TPORT

You must perform the following procedure to modify any INTRAN files you intend to use with network TPORT.

To prepare an INTRAN file to run network TPORT

Network TPORT requires the same model settings used for local TPORT as well as some additional settings.

For local TPORT, you need to add a SHARE command to your INTRAN file. For network TPORT, you must configure the model running on the Server machine to automatically launch the TRANSSHARE server for network TPORT to work correctly. You can do this through your INTRAN file by using one of two methods: identifying a specific server name, or using a default server name that is placed in the Schematic control manager.

- To automatically launch the TRANSSHARE server using a specified server name, add the following statement to your model's INTRAN file, anywhere after the BEGIN statement.

  ```
  DO.INTERACTIVE "BACKGROUND transshare -c modelname -z RegisterName"
  ```

  where:

  - `modelname` is the name of the model file that you are running
  - `RegisterName` is a unique name that you create to identify this TRANSSHARE server on the network.

  For example, if the name of the model file you are running is “Gasmodel.inprep” and the unique name you have chosen to identify the TRANSSHARE server is “test”, the line would appear as follows:

  ```
  DO.INTERACTIVE "BACKGROUND transshare -c Gasmodel -z test"
  ```

  For convenience and consistency, you may want to use the name of the Server computer for the RegisterName. If you do not know the name of your Server computer, ask your system administrator.
- As an alternative, you can use the following variation of the previous statement to use the default server name that is placed in the Schematic control manager:

  ```
  DO.INTERACTIVE "BACKGROUND transshare -c modelname -x"
  ```

where:

- `modelname` is the name of the model file that you are running
- `-x` indicates that you want to use the default server name, as specified in the Schematic control manager.

Running network TPORT

After installing SPS on the Client and Server machines and modifying your INTRAN file, you are ready to run network TPORT.

To run network TPORT

1 On the Server machine, run PREPR and TRANS.

If the TRANSSHARE server has set up correctly on the Server machine, a DOS window displays the RegisterName that you specified in the DO.INTERACTIVE statement.

2 On the Client machine, click on the **Start** bar and select **Run**.

3 In the Run dialog box, type the following:

```
cmd
```

4 In the command window, type the following:

```
TPORT lx::RegisterName
```

where `RegisterName` is the name specified in the DO.INTERACTIVE statement in the model INTRAN file.

5 If you did not specify a RegisterName, then you need to do the following:

a Copy the file named `xxx.share` from the Server machine to the Client machine. The file `xxx.share` is located in the same folder as the running model's INTRAN file.

Note: TPORT gets its connection information from the file xxx.share, which is created when transshare is run. It will be easiest to put the file in a location with a small path to the file.

b Open a command window and type the following:

```
cd c:\path to share file\
```

c Type the following:

```
TPORT share::modelname
```

TPORT gets its connection information from the SPS Network Lexicon. For more information on the SPS Network Lexicon, see "SPS Network Lexicon" on page 61.

Note: You must enter this command from the DOS command prompt window and not the Run dialog box.

6 After a brief delay, the TPORT window displays, and you may begin interacting with the simulation as you would in a local TPORT window.

Notes: Network TPORT has all of the capabilities of local TPORT. However, some displays (most notably distance plots) will not be completely drawn until TRANS has completed one or two time steps after the display is opened.

Model size may restrict use of network TPORT.

SPS Network Lexicon

The SPS Network Lexicon consists of three programs: lxServer, lxUtil, and lxService.

lxServer

lxServer is a console application that maintains a mapping between keys and values. Programs access that mapping through a TCP/IP port. Command-line arguments to lxServer may be as follows:

```
-p <portnumber>
-t <tracelevel>
```

In a typical installation, lxServer is run by lxService and does not need to be interacted with by the user.

lxUtil

lxUtil provides access to lxServer from a command line. For example:

```
lxUtil set string1 string2
```

causes lxServer to map the string `string1` to the string `string2`.

lxUtil commands are as follows:

```
set <key> <value>
get <key>
list
del <key>
```

Additional options are listed below. These options must appear before the command.

`-h <connection>`

where <connection> has the form "lx:<hostname>:<portnumber>.

`-v`

which tells lxUtil to print out assorted debug information.

SPS users will typically not use lxUtil, although it could be used to diagnose problems with lxServer.

lxService

lxService is a Windows Service that runs lxServer (starting or stopping lxServer when the service is started or stopped). lxService is installed/uninstalled with the following commands:

```
lxService -Install
```

```
lxService -Install -u
```

lxService is installed and run automatically by the SPS installation.

Connections

The port number used by lxServer is the first of the following:

1. Port number specified on the command line (`-p <portnumber>`)
2. Port number from SPS_LX_HOST environment variable, which should have a value like `x:<hostname>:<portnumber>`
3. Port number 10403

Note that for the second option, lxServer ignores `<hostname>`. It can only open a port on the local host.

The other programs will connect to an lxServer using the following:

1. Program-specific command line argument
2. SPS_LX_HOST environment variable in the form lx:<hostname>:<portnumber>
3. Port number 10403 on the local machine.

For lxUtil the command-line argument is

```
-h lx:<hostname>:<portnumber>
```

For transshare the command-line argument is

```
-y lx:<hostname>:<portnumber>
```

TPORT does not have a command-line argument to change the connection. It will use SPS_LX_HOST or lx:localhost:10403.

Network TPORT users will want to set SPS_LX_HOST to be `lx:<server>:10403`, where `<server>` is a stable network machine (usually the machine used to run SPS).

Running SPS programs from command line

You may run SPS programs from the command line. The command line may be either the operating system's command line (system prompt) or the SPS command line. For more information, see "The command line in the TRANS/TPORT window" on page 87.

If you are on Windows, you may want to run SPS programs through the user interface. For more information, see "Running SPS programs from the SPS startup window" on page 84.

This section documents the commands you may use to run SPS programs from the command line and the optional command line arguments that control SPS environment settings. If you want to set default SPS settings, see "SPS environment settings" on page 87. For more information, see the documentation of these programs.

- PREPR. Processes all data needed to define the configuration of the pipeline and calculates the initial state that can be used as the starting point of a transient simulation
- *TRANS.* Uses an initial state and the simulation control data to perform the transient simulation.

- *TPORT.* Allows you to work with multiple representations of a running model either locally or over a network.
- *DRTU.* Allows you to view the contents of a binary RTU data file produced by the SCADA interface or the COMPRESS_RTU utility. This utility may be run through the command line or through a Windows user interface. Only valid for Statefinder/Leakfinder.
- *COMPRESS_RTU.* Reduces the storage required for initialization by continuously reading the SCADA system-generated RTU data file, time-averaging the data, and then writing a compressed RTU data file. This compressed file is then used as the dynamic data to drive the Statefinder or Leakfinder analysis. COMPRESS_RTU is also useful in an off-line test environment for Statefinder or Leakfinder models.Only valid for Statefinder/Leakfinder.
- *CTOREVIEW.* Converts an RTU data file to a REVIEW file, which can be accessed by TPORT or SimPlot to view time plots of the SCADA data. Only valid for Statefinder/Leakfinder.
- RTUMERGE. a utility that combines the contents of two or more RTU data files into a single RTU data file.
- PREPR. Processes all data needed to define the configuration of the pipeline and calculates the initial state that can be used as the starting point of a transient simulation. This utility is used with either a Statefinder or Leakfinder model, both of which may require input data from two or more sources. Only valid for Statefinder/Leakfinder.
- *RTUFILT.* Extracts the values of telemetered points from an input RTU data file, performs user-defined calculations with the input points, and writes calculated results to an output RTU data file.

Running PREPR from the command line

```
prepr model
[-inprep= inprep-file]
[-outprp= outprp-file]
[-restrt= restrt-file]
[-h]
[-v]
[-HideWindow]
[-StartMinimized]
[-StayMinimized]
[-noabortwindow]
```

Field	Units Key	Description
model	n/a	File name of the INPREP file, excluding the file extension. SPS looks for the model file specified in the PATH environment variable. For more information on the PATH environment variable, see "List of SPS settings" on page 88. For more information on setting environment variables in general, see "Environment variables" on page 98.
inprep-file	n/a	Name of the INPREP file, if you want to specify different names for model-related files.
outprp-file	n/a	Name of the OUTPRP file, if you want to specify different names for model-related files.
restrt-file	n/a	Name of the RESTRT file, if you want to specify different names for model-related files.
h	n/a	Displays help for these commands.
v	n/a	Displays the version of PREPR you are running.
HideWindow	n/a	Hides the PREPR window. **Note:** Use with caution because it can make the window difficult to shut down and may require manual ending the process through the Windows Control Panel.
StartMinimized	n/a	Suppresses PREPR window from displaying at startup.
StayMinimized	n/a	Suppresses the PREPR window from displaying.
noabortwindow	n/a	Prevents the OUTPRP window from appearing if the software product aborts.

Description

For more information, see "Overview of PREPR" on page 53.

Example input

To run PREPR on a model named CASE1 with the restart data going to a RESTRT file named CASRES.RESTRT, type the following at the command line.

```
prepr case1 -restrt=casres.restrt
```

Running TRANS from the command line

```
trans model
[-intran=intran-file]
[-outtrn=outtrn-file]
[-review=review-file]
[-crt=crt-type]
[-rows=crt-rows]
[-cols=crt-columns]
[-font=crt-font]
[-logtrn=terminal-output-file]
[-restrt=restrt-file]
[-replay=replay-file]
[-nolock]
[-fg=fgcolor]
[-bg=bgcolor]
[-ffg=ffgcolor]
[-fbg=fbgcolor]
[-fcmd="argument"]
[-cmd="argument"]
[-h]
[-v]
[-HideWindow]
[-StartMinimized]
[-StayMinimized]
[-noabortwindow]
```

Field	Units Key	Description
model	n/a	File name of the INTRAN file, excluding the file extension. SPS looks for the model file specified in the PATH environment variable. For more information on the PATH environment variable, see "List of SPS settings" on page 88. For more information on setting environment variables in general, see "Environment variables" on page 98.
intran-file	n/a	Name of the INTRAN file, if you want to specify different names for model-related files.
outtrn-file	n/a	Name of the OUTTRN file, if you want to specify different names for model-related files.
review-file	n/a	Name of the REVIEW file, if you want to specify different names for model-related files.
crt-type	n/a	Specified the CRT terminal type. For more information see the available options in "INTERACTIVE" on page 463. The CRT terminal display rows and columns will be rest to the default for the CRT terminal type you specify.

Field	Units Key	Description
crt-rows	n/a	Number of rows in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
crt-columns	n/a	Number of columns in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
crt-font	n/a	Font or font size to be used for displays. **Note:** Entering new rows and columns values requires the CRT terminal type to be entered also.
terminal-output-file	n/a	Copies the terminal display input/output to the log file only when using a vt100 terminal and only if running the simulation in the background.
restrt-file	n/a	Name of the RESTRT file to use for initialization, if different from the model name.
replay-file	n/a	Name of the REPLAY file to use for initialization, if different from the model name.
nolock	n/a	Disables the locks that TRANS uses to prevent multiple writers to files such as the REVIEW file. **Note:** If two or more users write to a file at the same time, it may be destroyed. Therefore, use this option only if you are the only user of the file, the file is on the network, and you are experiencing network-related performance issues.
fgcolor	n/a	Text foreground color for X-windows only. Foreground and background colors may be specified by one of two methods: • Specifying the English word for the color, i.e. red. Any color that requires spaces must be enclosed in two sets of quotation marks, i.e., ""medium violet red"". • Specifying the color bits, i.e., #RGB or #RRRGGGBBB.

Field	Units Key	Description
bgcolor	n/a	Text background color for X-windows only. See the description for “fgcolor” above for more information.
ffgcolor	n/a	Figure foreground color for X-windows only. See the description for “fgcolor” above for more information.
fbgcolor	n/a	Figure background color for X-windows only. See the description for “fgcolor” above for more information.
fcmd argument	n/a	Places its corresponding argument at the beginning of the INTRAN file. This command is executed before any command in the INTRAN file. -FCMD may be used when a specific INTRAN command has not been defined in the INTRAN file, but is desired at the time the model is started.
cmd argument	n/a	Submits a command to TRANS after the INTRAN file commands have been executed, which is after the first time step. The behavior is similar to starting the model and then issuing the command from the TRANS window. If you use this command, the initial display does not appear.
h	n/a	Displays help for these commands.
v	n/a	Displays the version of TRANS you are running.
HideWindow	n/a	Hides the TRANS window. **Note:** Use with caution because it can make the window difficult to shut down and may require manual ending the process through the Windows Control Panel.
StartMinimized	n/a	Suppresses TRANS window from displaying at startup.
StayMinimized	n/a	Suppresses the TRANS window from displaying.
noabortwindow	n/a	Prevents the OUTTRN window from appearing if the software product aborts.

Description

For more information, see “Overview of TRANS” on page 55.

Example input

Windows example

To run TRANS on a model named LOOP2; along with using an archive file named LOOP2SS.ARK as the starting point for the TRANS run, type the following at the command line:

```
trans loop2 -restrt=loop2ss.ark
```

Running TPORT from the command line

Notes: Before running TPORT on a model, start TRANS and pause the simulation before starting TPORT. For other prerequisites, see "Preparing to use TPORT" on page 58.

```
tport model
[-crt=crt-type]
[-rows=crt-rows]
[-cols=crt-columns]
[-font=crt-font]
[-fg=fgcolor]
[-bg=bgcolor]
[-ffg=ffgcolor]
[-fbg=fbgcolor]
[-cmd="argument"]
[-readonly]
[-h]
[-v]
[-HideWindow]
[-StartMinimized]
[-StayMinimized]
[-noabortwindow]
```

Field	Units Key	Description
model	n/a	File name of the INTRAN or REVIEW file, excluding the file extension. SPS looks for the model file specified in the PATH environment variable. For more information on the PATH environment variable, see "List of SPS settings" on page 88. For more information on setting environment variables in general, see "Environment variables" on page 98.
crt-type	n/a	Specifies the CRT terminal type. For more information see the available options in "INTERACTIVE" on page 463. The CRT terminal display rows and columns will be reset to the default for the CRT terminal type you specify.
crt-rows	n/a	Number of rows in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.

Field	Units Key	Description
crt-columns	n/a	Number of columns in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
crt-font	n/a	Font or font size to be used for displays. **Note:** Entering new rows and columns values requires the CRT terminal type to be entered also.
fgcolor	n/a	Text foreground color for X-windows only. Foreground and background colors may be specified by one of two methods: • Specifying the English word for the color, i.e. red. Any color that requires spaces must be enclosed in two sets of quotation marks, i.e., ""medium violet red"". • Specifying the color bits, i.e., #RGB or #RRRGGGBBB.
bgcolor	n/a	Text background color for X-windows only. See the description for “fgcolor” above for more information.
ffgcolor	n/a	Figure foreground color for X-windows only. See the description for “fgcolor” above for more information.
fbgcolor	n/a	Figure background color for X-windows only. See the description for “fgcolor” above for more information.
readonly	n/a	Indicates the TPORT session is read only.
cmd argument	n/a	Submits a command to TPORT after the INTRAN file commands have been executed, which is after the first time step. The behavior is similar to starting the model and then issuing the command from the TRANS window. If you use this command, the initial display does not appear.
h	n/a	Displays help for these commands.
v	n/a	Displays the version of TPORT you are running.
HideWindow	n/a	Hides the TRANS window. **Note:** Use with caution because it can make the window difficult to shut down and may require manual ending the process through the Windows Control Panel.
StartMinimized	n/a	Suppresses TRANS window from displaying at startup.

Field	Units Key	Description
StayMinimized	n/a	Suppresses the TRANS window from displaying.
noabortwindow	n/a	Prevents the OUTTRN window from appearing if the software product aborts.

Description

For more information, see "Overview of TPORT" on page 57.

Take note

Halting TPORT

Halting a TPORT window will not halt the TRANS run.

Example input

```
tport case1
tport peak_day -crt=xwindows -fg=white
tport /disk/e/southern/50hdd -crt=vt340 -cmd="show valve_136"
tport east -fg=cyan -bg=black -crt=xw
```

Running DRTU from the command line

```
DRTU [input-file-name]
[-MATCH=match-list]   /* (s1,s2,...,sn)
[-TBEGIN=begin-time]   /* "YY/MM/DD_hh:mm:ss"
[-TEND=end-time]   /* "YY/MM/DD_hh:mm:ss"
[-IDWIDTH=width]
[-MAXTRYS=number-of-retries]   /* integer
[-PBEGIN=begin-record-number]   /* integer
[-PEND=end-record-number]   /* integer
[-WAIT=milliseconds-between-retries]   /* integer
[-QUALITY=quality-flag]   /* (s1,s2,...sn)
[-CUSPS=<cusp-seconds>]   /* integer
[-GAPS=<gap-seconds>]   /* integer
[-GAPANY]
[-PROGRESS=records]   /* integer
```

Field	Description
input-file-name	The name of the RTU data file to read. {rtudata.dt}
match-list	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be displayed. Wild cards "*" and "?" are allowed. Match arguments are converted to uppercase; therefore, names in the RTU data file with lower case characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 50.
begin-time	This option is used to specify the beginning time of data to be viewed. The utility will look for a data point with a time stamp close to the TBEGIN date/time. Data will be displayed from that point on until program termination is caused by TEND, REND, or the end of the data. DRTU will skip data with bad time tags in exactly the same manner as the TRANS program.
end-time	This is the end time for data to be viewed. When DRTU reads a point ID with a time stamp after the TEND date/time, DRTU terminates.
width	This specifies the width of the SCADA point name to be listed. The default width is 12 characters.
number-of-retries	Number of times DRTU will retry reading each data point. DRTU will exit if new data is not available after this many retries of any point. {100} See the description of WAIT *milliseconds-between-retries*.
begin-record-number	This is the first point number to be viewed. The utility will display data beginning with this point number until terminated by a TEND, REND, or end of data. Use RBEGIN to view points suspected of having bad time tags. When RBEGIN is specified, DRTU prints out all records, including those with bad time tags. Also, DRTU will read past the cusp in this case.
end-record-number	This is the last point number to be viewed. The utility will terminate when this record is encountered.
milliseconds-between-retries	Wait time between retries, in milliseconds. {1000} See the description of MAXTRYS *number-of-retries*.

Field	Description
quality-flag	Allows you to specify that only points for which quality matches the specified quality flags be displayed.
cusp-seconds	Ignores MATCH. Will print only the points adjacent to a cusp of the specified size or larger. If used without RBEGIN, it implies RBEGIN = 1.
gap-seconds	Based on the current MATCH settings, prints only lines that would have been printed, with adjacent lines having a time-stamp separated by at least the specified amount.
GAPANY	Modifies GAPS, so that any time any single ID has the specified time gap, the points bounding the gap are printed.
records	Outputs a record, regardless of MATCH, CUSP, etc., after every specified number of records. This option makes it easier to see that DRTU is functioning when the output would otherwise be very sparse.

Description

DRTU is a utility designed to allow you to view the contents of a binary RTU data file produced by the SCADA interface or the COMPRESS_RTU utility. The RTU data file is a circular data file that contains SCADA point data for the input of dynamic data to the Statefinder or Leakfinder models. The file has a finite length; new data points are written over the oldest points once the size of the file reaches its defined size limit. The boundary between the newest point and the oldest point is referred to as the *cusp*.

Take note

Output

DRTU will display the contents of the RTU data file in the following manner:

```
Point No.    Date     Time       SCADA Point Name     Value       Quality

125000        91/07/21 00:42:44  STA535.PRES.DISC     757.89       good
125001        91/07/21 00:42:44  STA535.PRES.SUCT     590.45       good
125002        91/07/21 00:43:04  STA520.FLOW.STD      1987.12      good
```

Example input

The following examples should help illustrate the DRTU utility:

```
drtu rtudata.dt
Point No.    Date     Time       SCADA Point Name     Value     Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC      757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT      590.45     good
125002     91/07/21 00:43:04     STA520.FLOW.STD      1987.12     good
125003     91/07/21 00:42:44     STA060.PRES.DISC      799.03     good
125004     91/07/21 00:42:44     STA060.PRES.SUCT      653.87     good
125005     91/07/21 00:43:04     STA520.PRES.DISC      749.33     good
125006     91/07/21 00:43:04     STA520.PRES.SUCT      624.29     good
125007     91/07/21 00:43:04     STA520.FLOW.ACT      1103.65     good
```

Match is not case sensitive; however, use of match excludes any names with lowercase characters:

```
drtu rtu.dt -match=sta060.pres.disc     <---WINDOWS/DOS
Point No.     Date     Time      SCADA Point Name     Value     Quality
125003     91/07/21 00:42:44     STA060.PRES.DISC     799.03     good
125354     91/07/21 00:43:26     STA060.PRES.DISC     799.67     good
125702     91/07/21 00:43:59     STA060.PRES.DISC     799.92     good
drtu rtu.dt -match=sta520.flow.act    <---WINDOWS
Point No.     Date     Time     SCADA Point Name       Value    Quality
 < no points match - point name is lowercase in RTUDATA.DT>
```

Wildcards may be used. See "Wildcards" on page 50.

```
drtu rtu.dt "-match=sta5??.pres.*"     <---WINDOW
Point No.    Date     Time       SCADA Point Name    Value  Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC     757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT     590.45     good
125005     91/07/21 00:43:04     STA520.PRES.DISC     749.33     good
125006     91/07/21 00:43:04     STA520.PRES.SUCT     624.29     good
```

Time specification:

```
drtu rtu.dt -tbegin=91/07/21 -tend=91/07/21_00:43:00   <---WINDOWS
Point No.    Date     Time       SCADA Point Name     Value    Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC     757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT     590.45     good
125003     91/07/21 00:42:44     STA060.PRES.DISC     799.03     good
125004     91/07/21 00:42:44     STA060.PRES.SUCT     653.87     good
```

Running COMPRESS_RTU from the command line

```
COMPRESS_RTU [input-file-name] [output-file-name]
[-MATCH=match-list]     /* (s1,s2,...,sn)
[-INTERVAL=averaging-interval]     /* real
[-MAXRECS=max-points-in-output-file]     /* integer
[-START.RTUDATA=first-rec-to-read]     /* integer
[-START.COMPRESS=first-rec-to-write]     /* integer
[-TIMEOUT=time-out-period(seconds)]     /* integer
[-ECHO]     /* echo-reads-and-writes
[-SPEED=times-real-time]     /* real
[-h]
```

Field	Description
input-file-name	Name of RTU data file to be read for input. {rtudata.dt}
output-file-name	Name of new RTU data file to be opened for output. {compress.dt}
match	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be passed through to the output file. Wild cards * and ? are permitted. Match arguments are converted to uppercase; therefore, names in the RTU data file with lowercase characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 50. The first record in the file is always displayed first, whether or not it meets the match criteria {*}.
interval	For each SCADA point ID, this specifies the interval of time in minutes over which data from the RTU data file will be averaged.
maxrecs	Maximum number of points to allocate space for in the output file. After writing this many points, the output will wrap around and begin again, overwriting point number 1. Applicable only when the file is created. {100000}
start.rtudata	The point number where COMPRESS.RTU starts reading the input data file. The default is to start at the time corresponding to the output data file's newest points. (This default means that COMPRESS.RTU can be restarted after any amount of down-time, provided no input data has been lost.) The SCADA point numbers can be determined using the DRTU utility.
start.compress	Represents the starting point number to start writing to the output data file. The default is to start at the point just after the newest point in the output data file. The SCADA record numbers can be determined using the DRTU utility.
timeout	Period of time (seconds) the utility will wait for new data when the end of the RTU data file is encountered. After this amount of time, with no new data, SPS will exit. {172800 seconds}
echo	Echoes input points and output points to the display. Default is no echo. This option produces a lot of output and is not typically used.

Field	Description
speed	This option is used to set up a test environment in which the output file is being written as if the RTU data file is being written by a real SCADA system. SPEED=1 (and INTERVAL=0.001) will delay its output so that it copies records from its input file to its output file in real time. In an online environment, COMPRESS_RTU automatically waits for its input file to be updated, so the SPEED option is not necessary. The default is to write the data as quickly as possible.
h	Print the options this utility supports.

Description

COMPRESS_RTU is a utility used to average data contained in an RTUDATA file. It creates a compressed RTUDATA file.

In operational environments, such as Statefinder models or Leakfinder models of batched pipeline systems, the amount of historical recorded RTU data required to initialize a simulation is substantial. To reduce the storage required by this “cold start” process, COMPRESS_RTU is run on a continuous basis, reading the SCADA system generated RTU data file, time averaging the data, and writing a compressed RTU data file. This compressed file is then used as the dynamic data to drive the Statefinder or Leakfinder.

COMPRESS_RTU is also useful in an off-line test environment for Statefinder or Leakfinder models. With the various options, you can

- Average an RTU data file
- Create another RTUDATA file with only selected data points
- Start at a specified input file record
- Specify the time averaging period *and/or* specify the speed at which the output data is written

Take note

Output

Typically no output is displayed. The ECHO option enables various analytical output messages. Also, diagnostic messages are displayed.

INTERPOLATION.RULE

COMPRESS_RTU is incompatible with the INTERPOLATION.RULE option on the SCADA device. COMPRESS_RTU always uses simple averaging.

Quitting

.COMPRESS_RTU will run until no data remains for the timeout period. You can also stop the process using a control C.

Example input

Using a compression interval of 10 minutes, COMPRESS_RTU would write the following to its output file:

```
Example 1
REC#   DATE/TIME STAMP     SCADA ID   VALUE
```

```
100     94/05/01 00:00:05   Q1          250
101     94/05/01 00:00:05   Q2          250
102     94/05/01 00:00:15   Q1          350
103     94/05/01 00:00:15   Q2          350
Examples
compress_rtu rtu.dt -match="(s1,..,sn)"
Examples
compress_rtu rtu.dt -interval=10
Examples
compress_rtu rtu.dt -maxrecs=2500000
Examples
compress_rtu rtu.dt -st.rtu=125350
Examples
compress_rtu rtu.dt -st.comp=125350
Examples
compress_rtu rtu.dt -timeout=600
Examples
compress_rtu rtu.dt compress.dt -echo
Examples
compress_rtu rtu.dt compr.dt -speed=2
```

Running CTOREVIEW from the command line

```
CTOREVIEW [input-file-name] [output-file-name]
[-TBEGIN= begin-time]   /*"92/01/01 11:57:00"
[-TEND= end-time]   /*"91/01/01 11:57:00"
[-MATCH= match-list]   /*(s1,s2,...,sn)
[-MAX.NAMES= number-of-names]    /*integer
```

Field	Description
input-file-name	Name of RTU data file to convert. {rtudata.dt}
output-file-name	Name of REVIEW file to create must start with an alpha character. {bc.review}
begin-time	The beginning time in the input file to be converted. The utility will look for a data point with a time stamp close to the TBEGIN date/time. Data will be converted and written to the output file from that point until program termination is caused by TEND (or the end of the data).
end-time	This is the end time for data to be converted. CTOREVIEW will quit when it finds a point with time greater than TEND (or the end of the data).
match-list	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be displayed. Wild Cards * and ? are allowed, where * matches a string and ? matches a character. Match arguments are converted to uppercase; therefore, names in the RTU data file with lower case characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 50.
number-of-names	A guess at the number of identifiers that will be in the REVIEW file. The REVIEW file has a dictionary at the front; space for this dictionary is pre-allocated (the dictionary automatically grows, if required, overwriting early data records). The CTOREVIEW program allocates space for MAX.NAMES, assuming the names average 12 characters each {1000}. Do not use a very large number for MAX.NAMES, because disk space will be allocated to hold that many names.

Description

CTOREVIEW is a utility designed to convert an RTU data file to a REVIEW file. The REVIEW file can be accessed by TPORT or SimPlot to view time plots of the SCADA data.

Take note

Output

Progress is displayed in the following format:

```
Outputting a record for <SCADA name>, time <time>
Outputting a record for SS.102, time 92/10/25 22:00:00
```

Example input

Examples

ctoreview rtu.dt -tbegin=92/01/01_11:57:00

Examples

ctoreview rtu.dt -tend=92/01/01_11:57:00

Examples

ctoreview rtu.dt -match="(s1,s2,s3,...,sn)"

Examples

ctoreview rtu.dt -max.names=1000

Running RTUMERGE from the command line

```
RTUMERGE
-READ=input-file-list    /* (s1,s2,...,sn)
-WRITE=output-file       /* s1
[-MAXRECS=max-points-in-output-file]     /* integer
[-TIMEOUT=time-out-period-list (minutes)]    /* (r1,r2,...rn)
[-ECHO]
[-MEMSUM=memsum-after-these-points]    /* integer
[-QUIT]     /* quit when all timed out
[-H]
```

Field	Description
input-file-list	The list of the RTU data files to be read.
output-file	The name of the output file.
max-points-in-output-file	Maximum number of data points stored in the output file. RTUMERGE will write MAXRECS points to the output file and will then begin to write over the oldest existing data {1000000}. Once the number of data points has been selected for a given output file, it cannot be changed.
time-out-period-list	Defines the amount of time RTUMERGE will wait for additional data before it goes on without the data from the file that has timed out. There may be a different timeout period associated with each input file. {300 seconds}
ECHO	Print out each point as it is read and/or written. The name of the input file is written in the last column as the point is read. A W is written in the last column if the point is being written to the output file.
memsum-after-these-points	Every memsum number of points give a summary. {no summary}
QUIT	Quit when all the read files have timed out.

Description

RTUMERGE is a utility that combines the contents of two or more RTU data files into a single RTU data file. This utility is used with either a Statefinder or Leakfinder model, both of which may require input data from two or more sources. For example, this utility can be used with a single Statefinder model that requires dynamic online data, which is maintained by two separate SCADA systems. Each SCADA system is configured to generate an RTU data file recording the data from its database, which is required for the model. RTUMERGE runs continuously, reading the two SCADA generated RTU data files, merging the contents of the files into single file. RTUMERGE works by having a table of “current” points from each input file. RTUMERGE copies the point with the smallest time tag to the output file, and then gets a new point from the corresponding input file.

Note: With the RPC version of the API, RTUMERGE is no longer needed.

Take note

Restarts

RTUMERGE does a good job of restart if it is shutdown. RTUMERGE opens the output and finds the newest time. It then finds the time in the input and starts reading from there.

Output

RTUMERGE displays the contents of the RTU data file in the following format when the ECHO option is specified:

```
(Point No.   Date     Time     SCADA Point Name   Value  )

1231        91/07/21  00:42:44  STA535.PRES.DISC  757.89  W
125000      91/07/21  00:42:44  STA535.PRES.DISC  757.89  RTUDATA1.DAT
125001      91/07/21  00:42:44  STA535.PRES.SUCT  590.45  RTUDATA1.DAT
1232        91/07/21  00:42:44  STA535.PRES.SUCT  590.45  W
125002      91/07/21  00:43:04  STA520.FLOW.STD  1987.12  RTUDATA2.DAT
1233        91/07/21  00:43:04  STA520.FLOW.STD  1987.12  W
```

Example input

Running RTUFILT from the command line

```
RTUFILT MODEL
[-INFILT = infilt-file]    /* string
[-OUTFLT = outfilt-file]  /* string
[-RTUIN = rtuin-file]  /* string
[-RTUOUT = rtuout-file]  /* string
[-TBEGIN = begin-time]  /* "YY/MM/DD-hh:mm:ss"
[-TEND = end-time]  /* "YY/MM/DD_hh:mm:ss"
[-MAXPTS = maximum-points-in-rtuout]  /* integer
[-DUMP]   /* dEBUG PRINT
```

Description

The RTUFILT command line utility used for data filtering. It extracts the values of telemetered points from an input RTU data file, performs user-defined calculations with the input points, and writes calculated results to an output RTU data file. For more information, see “RTUFILT utility” on page 822.

User interface and utility debugging

RTUFILT is not an interactive utility and is typically run as a background process. While running, you cannot directly interact with the utility other than to stop it.

A dump flag, DUMP, is provided to allow debugging of output. When DUMP is specified, all CALC points are written to the RTUOUT file. Caution should be exercised when setting DUMP because RTUFILT output can be quite voluminous.

CHAPTER

3

The SPS Environment

SPS may be run on many systems and in a variety of configurations. The typical “look and feel” of the product environment varies, depending on your operating system and additional components you are licensed to use. This section also describes customizing environment settings to enhance your modeling experience and increase performance.

Using the SPS for Windows environment

The SPS for Windows environment includes a model builder environment from which you can graphically build a model and populate model data through dialog boxes. From the model builder, you can also verify the model and launch the simulation. Alternately, you may start the simulation through the SPS startup window. Model Builder largely replaces the functionality of the SPS startup window, but you still may need or prefer to use the SPS startup window.

Once the simulation is launched, you interact with the simulation through the TRANS/TPORT windows.

For more information on the main SPS windows, see:

- “Model Builder” on page 697
- “SPS startup window” on page 83
- “TRANS/TPORT window” on page 85

For more information on the Model Builder environment, see “Model Builder environment” on page 698.

SPS startup window

The SPS startup window allows you to open and manipulate files, set display and memory options, and run the simulation. For the most part you may perform the same tasks through the Model Builder interface.

Note: The SPS startup window is more flexible when launching a model in TPORT. A model must be loaded in Model Builder to launch a corresponding TPORT session. However, you may launch a TPORT session through the SPS startup window with only a REVIEW file (and no INPREP file).

For more information on opening and running model files, see “Running SPS programs from the SPS startup window” on page 84. For more information on setting options for SPS, see “Setting preferences in the SPS startup window” on page 100. For more information on running the simulation, see “Running the simulation” on page 157.

Running SPS programs from the SPS startup window

You can run SPS programs from the SPS startup window. Before running SPS programs, you must open the startup window and select a case. For more information on selecting a case, see "To select a case in the SPS startup window" on page 84.

To start SPS on Windows

Select the Stoner Pipeline Simulator program icon from the Programs listing in the Start Bar.

The SPS startup window displays.

To select a case in the SPS startup window

Note: See "Cases" on page 99 if you want to customize the name and location of the files associated with the case.

1. From the **SPS startup window**, select a **Case** for a simulation by one of the following methods:
   - Click **Browse** to browse your system's file structure to locate a **Case** or model files.
   - Select **File > Open** from the SPS startup window.
   - Click on the **Case** options menu to view and select from the Case history.
2. Type the Case name into the **Case** field.

To run PREPR from the SPS startup window

1. In the **SPS startup window**, specify the name of the **Case** that you want to run. For more information, see "To select a case in the SPS startup window" on page 84.
2. Select **Tools > PREPR**.

   —or—

   Click  .

When you run PREPR from the SPS dialog box, an MS-DOS command prompt window opens. PREPR runs in the background and any warning/error messages display in the system window. These messages write to the OUTPRP report.

3 After PREPR completes and you have examined all of the messages, click the Close symbol in the upper-right corner to close the window.

Tip: Windows users can also type **EXIT**.

To run TRANS from the SPS startup window

1 In the **SPS startup window**, specify the name of the **Case** that you want to run.

2 Select **Tools > TRANS**.

—or—

Click  .

Tip: To run the optimized version of TRANS, hold down the Shift key while clicking the TRANS button.

SPS opens a TRANS window.

To run TPORT from the SPS startup window

1 In the **SPS startup window**, specify the name of the **Case** that you want to run.

2 Select **Tools > TPORT**.

—or—

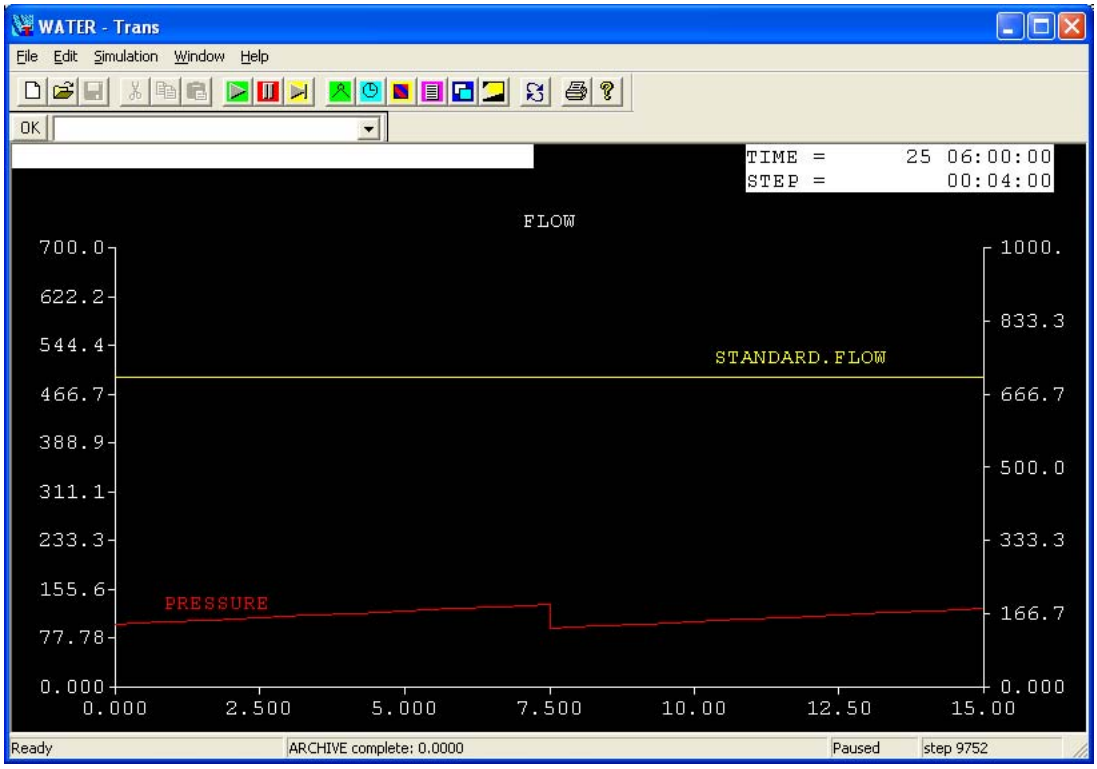
Click  .

SPS opens a TPORT window.

Tip: You can run multiple TPORT sessions on a single Case during a simulation. To run multiple TPORT sessions, repeat step 2 for each additional session.

TRANS/TPORT window

From the TRANS and TPORT windows, you can interactively control the simulation. You can perform actions such as opening and closing valves, starting and stopping rotating equipment, and changing set points. You also have multiple means of reviewing the results. You can look at plots of variables versus time, and variables versus distance. You can view tabular Reports or show information for a given device. You can also use a text editor to build custom Reports for displaying simulation results. For more information on running the simulation and entering interactive commands, see "Controlling the Simulation" on page 153. For more information on working with plots, reports, and Show windows, see "Displaying and Printing Data" on page 163.



TRANS is the main simulation window, and you can run only one TRANS window at a time for any given model/case. *TPORT* allows you to open as many additional simulation windows for the same running model as you would like. You can run separate models/cases of TRANS simultaneously, but this will impact the speed of the TRANS sessions.

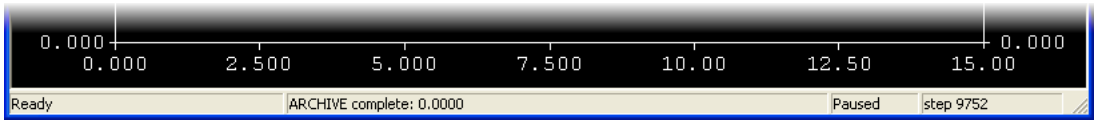
Because the windows and menus for TRANS and TPORT are identical, they are documented together. For more information on TRANS and TPORT, see "Overview of TRANS" on page 55 and "Overview of TPORT" on page 57. If you want to open a TPORT window, see "To run TPORT from the SPS startup window" on page 85.

The TRANS/TPORT toolbar

The TRANS/TPORT toolbar provides shortcuts for common functions, and many of the buttons are also available in the various SPS displays. Toolbars in the TRANS/TPORT window are movable and dockable. You can hide or show the TRANS/TPORT toolbar by selecting Window > Toolbar. Hover your cursor over each button to see a description.

The TRANS/TPORT status bar

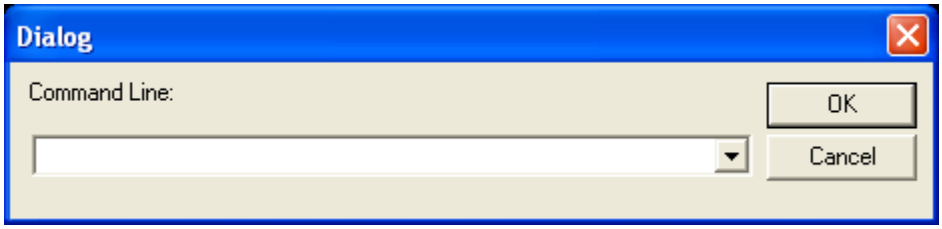
The status bar provides the status of the current simulation (such as Running or Paused) and the current time step. The status bar also provides a description of the toolbar button that the mouse is hovering over, as well as other notes the current simulation.



The command line in the TRANS/TPORT window

You can type commands into the command line or into the display of the main window (Typing a command into the TRANS/TPORT window invokes the popup command line). Click OK or press Enter to accept these commands. The command line is particularly useful for entering seldom-used commands that are not available through the menus.

The popup command line opens when you type any character in the TRANS/TPORT window.



The location of the standard command line is in the TRANS/TPORT window below the toolbar.

You can move the command line in the TRANS/TPORT window to any location in the TRANS/TPORT window. To move the command line, click anywhere on a gray area of command line, hold mouse button down and drag the command line to the desired location.

You can access a history of the commands already entered by clicking the options menu arrow. You can click on a previous command and then click OK to repeat that command.

You can hide or show the standard command line by deactivating and activating the command line option under the Window menu in the TRANS/TPORT window.

For more information, see "Entering commands from the SPS command line" on page 158.

SPS environment settings

SPS uses a collection of environment settings, which specify information such as display characteristics, time zone settings, and file locations. These settings can be managed in several different ways:

- At the system prompt or command line, when you launch a module, if you launch modules by this method. For more information on supported command line arguments, see "Running SPS programs from command line" on page 62.
- Within an sps.settings file(s). For more information on this file, see "sps.settings files" on page 91.
- Using a case, stored in a case file (Windows only). For more information on cases, see "Cases" on page 99.
- Using system environment variables. For more information on environment variables, see "Environment variables" on page 98.

Each of these tools has a specific group of settings it can manage, and none can handle all settings on its own. Therefore, you need to use a combination of these tools to manage your SPS settings. For example, you must use an environment variable to set the DREMTIMEZONE flag, which controls how the time zone TZ is handled during simulations. However, you can set the TZ time zone with either another environment variable, or you can use an

sps.settings file. For a listing of all SPS settings and the locations where they may be set, see "List of SPS settings" on page 88.

List of SPS settings

The following table lists SPS settings and where they may be managed. The keywords listed indicate the syntax of the environment variable, sps.settings file command, or system prompt argument, as applicable. The "Where Set" column indicates where you may set these settings, using the following abbreviations:

- C—SPS command line option
- S—sps.settings file
- E—System environment variable or system prompt

Note: You may edit environment variables using different methods, depending on your operating system. In addition, environment variables may be edited through the SPS command line using special syntax. For more information and examples, see "Environment variables" on page 98.

Keyword(s)	Description	Where set			Notes
		C	S	E	
COLS	Specifies an alternate terminal size for large displays. (Each installation has a default terminal display size, typically 24 rows x 80 columns).	X	X		TRANS, TPORT only
DREMCASENAME	Specifies a case name that represents the current model being run. This setting is only used when interfacing with a Trainer session.			X	Trainer only
DREMFILES	The directory where the SPS program files are installed. This setting is required.			X	
DREMLICENSE	Alternate directory containing the license file.			X	
DREMPATH_DSP	Specifies the directory (or directories) to search when looking for a display (DSP) file. For more information, see "Display directory (DREMPATH_DSP)" on page 94.		X	X	
DREMTIMEZONE	Controls the usage of the time zone specified by the TZ setting. For more information see "Simulation time" on page 199.			X	
FG, BG, FFG, FBG	Colors for simulation windows.	X	X		TRANS, TPORT only
FONT	Font size for simulation windows.	X	X		TRANS, TPORT only

Keyword(s)	Description	Where set			Notes
		C	S	E	
Path	Specifies the location of the SPS executable, used when SPS is launched from the operating system prompt. This environment variable may contain multiple paths for multiple applications, including those other than Stoner Software prodcuts. Therefore, when SPS is launched from the system prompt, it uses the first valid SPS path it finds in this variable. When a newer version SPS is installed, the path of the new version is automatically placed first.			X	Not SPS-exclusive
PRINTLICENSE	Specifies whether to print license information to a standard error, typically the window from which the process was launched. This is generally done for debugging work. (0 = False, anything else = True).			X	
ROWS	Specifies an alternate terminal size for large displays. (Each installation has a default terminal display size, typically 24 rows x 80 columns).	X	X		TRANS, TPORT only
SAIxxxSIZE	Specifies the memory allocated for use by SPS. Insufficient memory allocation may cause SPS to malfunction. For more information on setting memory allocation, search the knowledge base articles on GL's website or contact Technical Support.		X	X	
SAISHAREDMEMORYSIZE	Allocates additional shared memory size. Standard allocation is shared memory size + overhead + max (user size, 100000). For models with complex distance plots, extra memory is required. For more information, see "Shared memory" on page 57.			X	
SAIREADONLY	Specifies whether TPORT is read-only, or whether it can send commands to TRANS. If this variable is set to anything, TPORT is read-only.	X		X	Called READONLY when used from the TPORT command line.
SPS_FONT_FACE_NAME	Specifies the initial font for the Trans/Tport windows. The default is Courier New.		X	X	

Keyword(s)	Description	Where set			Notes
		C	S	E	
SPSGRID	Specifies whether symbols are shown in the TRANS and TPORT window to indicate the intersecting points of grid lines. You can enter the following values for this variable: • 0—No grid lines (default setting) • 1—period • 2—plus sign • 3—asterisk • 4—circle • 5—X • 6—combined plus sign/circle			X	
TZ	Defines the time zone for the simulation.		X	X	Not specific to SPS
{case}	Specifies the locations for all files associated with a particular model/ simulation, such as the INPREP, OUTPRP, RESTRT, INTRAN, and other files as applicable. This feature allows you to assemble the files for a model into a particular “case,” and then run simulations more quickly by specifying just the case name.				Stored in a case file, written and managed by SPS.

Priority of settings

Because certain settings can be specified in either an sps.settings file, environment variable, command line argument, or a combination of the three, SPS uses a standard process to determine which settings to use for any given simulation. When any module such as PREPR or TRANS is launched, SPS looks for settings in the following locations, in the following order:

1 In the arguments specified at the command line, if the module was launched in this manner.

2 In the sps.settings file within the model files directory, if it exists.

3 In the sps.settings file within the SPS install directory, if it exists.

4 In the system environment variables, as applicable.

For any given setting, SPS uses the first setting it finds, and any others are ignored. For example, if it finds a SAIPREPRSIZE memory setting in an sps.settings file, it will use that memory setting and ignore any memory setting it finds in your system environment variables.

If a setting is not found in any of these locations, SPS uses a default value as needed

sps.settings files

An sps.settings file contains certain settings information and can be customized for a particular model or models. An sps.settings file applies to the model(s) in the same directory as itself, allowing you to customize them as needed without affecting other models or simulations. An sps.settings file can contain information on the following:

- Memory settings
- Display settings
- Time zone settings

In previous versions of SPS, some of this information could only be specified by an environment variable, which affected all models and simulations. For example, memory settings were originally set with environment variables that you had to change manually if you wanted to change them between different simulations. With the sps.settings file, however, you can specify memory settings and place that file in a particular model directory, which will be read any time a module such as PREPR or TRANS is launched. In addition, settings within an sps.settings file are portable between computers, because you can simply move the settings file with the remainder of the model files. For these reasons, GL recommends that you use an sps.settings file(s) for any setting it supports, rather than environment variables.

To associate an sps.settings file with a particular model, you should store it in the same directory as the model files. When a module is launched for a particular model, SPS will look for sps.settings in the current active directory first, normally the model directory. If the settings are not found in that file, or if that file does not exist at all, SPS will then look in the SPS install directory for an SPS settings file. If no file is found, or the settings information is still incomplete, SPS will look for environment variables, if they exist, or use default values if not.

The sps.settings file is generated, read, and modified through the preference selections in Model Builder or the SPS startup window. If you are using SPS on another platform, however, there is no sample file available in the installation and you must write it from scratch. To work with the sps.settings file through model builder, see “Setting miscellaneous Model Builder preferences” on page 703. For more on working with sps.settings through the SPS startup window, see “Setting preferences in the SPS startup window” on page 100.

General syntax rules for an sps.settings file

For those settings supported by both sps.settings files and environment variables, the keywords are identical. For example, a TZ command in an sps.settings file is functionally comparable to a TZ environment variable. For these types of settings, GL recommends that you use an sps.settings file(s) rather than environment variables.

The following rules apply to the syntax of an sps.settings file.

- Comments are permitted in the file and must begin with double forward slash (//) delimiter. For any line that begins with this delimiter, the entire line is considered a comment and is ignored by SPS.
- If a variable is set more than once, the last one in the file takes precedence.
- Misspelled variables are ignored, and other syntax problems could cause other errors. As such, you should ensure that the sps.settings syntax is correct before launching a simulation.
- Case is not considered. For example, “SAIDEFAULTSIZE” is equivalent to “SaiDefaultSize.”

Specifying settings for an individual module

Unlike environment variables, in an sps.settings file you can optionally apply settings on a module-by-module basis. For example, you could assign different coloring styles to a TPORT and a TRANS window, even within the same simulation. To apply a setting to a particular module, you should generally use the following syntax:

```
setting.[module] = value
```

For example, the following command applies a foreground color to TPORT only:

```
fg.tport = white
```

However, the following setting would apply to all modules, because no specific module is specified:

```
fg = black
```

This syntax works with most settings and can be applied to the following primary modules.

- PREPR
- TRANS
- TPORT
-
- DEMAC

In addition, you can apply settings to other modules, using the following syntax:

- CMPF
- COMPRESS_RTU
- CTOREVIEW
- DRTU
- RTUFILT
- RTUGEN
- RTUMERGE
- STATDIFF
- SPSEXPORT
- TELLTRANS
- TRANSHEARTBEAT

  Tip: Keep in mind that case is not considered.

If a setting is not relevant to a particular module, it is ignored. For example, because the time zone is not relevant to PREPR, the setting tz.prepr would be ignored. If a setting requires a different syntax for module-specific application, such as memory allocation settings, this syntax is noted elsewhere in this document and within the factory comments contained in an sps.settings file.

Memory allocation

The syntax for resetting SPS memory allocation in an sps.settings file is as follows:

```
SAImoduleSIZE = "d:x[k|m] idh:x[k|m] imem:x[k|m]"
```

where:

module	=	The program or utility to be resized. See "Specifying settings for an individual module" on page 92 for a list of module keywords. Use of "default" (SAIDEFAULTSIZE) indicates all modules.
x	=	The value for d, idh, or imem. Numbers may be suffixed with k for kilo or m for mega. For more information on determining these values, search the Stoner Software knowledge base articles or contact Technical Support.

For example, consider the following setting:

```
SaiDefaultSize = "d=2m idh=1m imem=10m"
```

This setting specifies that for all modules:

- d-array is sized at 2,000,000 words
- idh-array is sized at 1,000,000 words
- imem-array is sized at 10,00,0000 words

Fonts

The syntax for setting fonts in an sps.settings file is as follows:

```
font.[module] = size
```

where:

module	=	Module to which to apply the font. See "Specifying settings for an individual module" on page 92 for a list of module keywords. To apply the color to all modules, omit this item.
size	=	Font size. Valid values include small, medium, large, and huge.

See "Specifying settings for an individual module" on page 92 for a list of module keywords.

As an example, the following line would set the font for the TRANS module to "medium":

```
font.trans = medium
```

Display colors

The syntax for setting colors in an sps.settings file for TRANS and TPORT is as follows:

```
item.[module] = color
```

where:

item	=	One of the following: • fg (foreground color) • bg (background color) • ffg (figure foreground color) • fbg (figure background color
module	=	Module to which to apply the color (trans and tport only). To apply the color to all available modules, omit this item.
color	=	Color to be applied. Possible colors include blue, red, green, cyan, magenta, yellow, orange, dkgreen, brown, dkblue, purple, pink, dkgray, ltgray, black, white.

For example, the following entries would set foreground to white and the background to black, for all modules:

```
fg = white
bg = black
```

Display directory (DREMPATH_DSP)

If you want to display an existing time plot, distance plot, report, or custom text display through the TRANS or TPORT interface, SPS needs to know where to locate the associated display (DSP) file to create the display. The display directory (DREMPATH_DSP) setting contains the directories in which to automatically search for DSP files. For more information on opening a display from the command line, see "To open a previously saved display from the command line" on page 164.

DREMPATH_DSP is optional. If you do not use DREMPATH_DSP, SPS looks for DSP files in the directory of the currently active model only. In either case, SPS produces an error if a DSP file cannot be found.

You can specify more than one display directory. In this case, SPS searches the directories in the order that they are listed, using the first DSP file that it finds with the specified name.

The syntax for setting the directory where SPS may access DSP files in an sps.settings file is as follows:

```
drempath_dsp= "path1 [path2] [pathn]"
```

where:

pathx	=	To specify the currently active model directory, enter a period (.) in the path. You may enter a specific path or a relative to the currently active model directory. Multiple directories are separated by a space. Additionally, you may use semi-colons for Windows. **Notes:** SPS cannot support directories in this variable that contain a space anywhere in the path. If a path contains a space, SPS assumes that a new directory is starting, which will likely produce an error. When saving or printing a DSP file, SPS uses the first directory listed.

As an example, the following syntax will cause SPS to look in the active model directory first, the “MyFiles” directory next, and then the networked location.

```
drempath_dsp = ". C:\MyFiles U:\Common_Displays"
```

Notice that the networked folder name contains an underscore rather than a space in the name to avoid errors.

Because SPS uses the first directory listed when saving or printing a DSP file, in this example all DSP files would be saved to the directory of the currently active model.

Time zone

The syntax for setting the time zone in an sps.settings file is as follows:

```
tz.[module] = standard_offset[dst_offset]
```

where:

module = Module to which to apply the zone. See "Specifying settings for an individual module" on page 92 for a list of module keywords. To apply the zone to all modules, do not specify any module.

standard_offset = Indicates the offset of the zone from Coordinated Universal Time (formerly known as Greenwich Mean Time, GMT), for standard time (not daylight savings). The offset uses the following syntax:

```
abbreviation[-]hours[:minutes[:secs]]
```

where:

- abbreviation—Any three-character or longer phrase, such as EST. This abbreviation is for visual convenience only, and does not affect the time zone specification. Only the numerical offset is actually used.
- dash (-)—Indicates that the zone you are specifying is east of the Prime Meridian. Absence of the dash (or an optional plus (+) sign) indicates west. The specification of east or west determines whether the offset is added or subtracted from UTC.
- hours:minutes:seconds—The numerical time offset from UTC. Minutes and seconds are optional.

For example, the following offset indicates Eastern Standard Time, which is 5 hours earlier than UTC:

```
EST5
```

Because the abbreviation is largely irrelevant, the following offset will indicate the same functional time zone:

```
EastStandTime5
```

SPS uses the Posix standard TZ processing, but other capabilities are available as well. For more information, see the TZ documentation for your operating system.

daylight_offset = Indicates the offset of the zone from Coordinated Universal Time, for daylight savings time. If this field is omitted, SPS assumes a daylight savings time of one hour ahead. The syntax of this field is identical to that of standard_offset.

For example, the following entries are all functionally identical, each specifying Eastern Time in the United States:

```
tz = EST5
tz = EST5EDT4
tz = EST5EDT
tz = Charlie5Brown4
```

The following excerpt from a sample sps.settings file shows some commonly used time zones:

```
TZ = UTC            // Coordinated Universal Time

TZ = NST3:30NDT1:30
```

```
//   Newfoundland Standard Time (NST) is 3.5 hours earlier than
//   Coordinated Universal Time (UTC). Standard time and daylight saving
//   time both apply to this locale. Newfoundland Daylight Time is 1.5
//   hours earlier than Coordinated Universal Time (UTC).
TZ = EST5EDT
//   Eastern Standard Time (EST) is 5 hours earlier than Coordinated
//   Universal Time (UTC). Standard time and daylight saving time
//   both apply to this locale. By default, Eastern Daylight Time
//   (EDT) is one hour ahead of standard time (that is, EDT4).

TZ = CST6CDT
//   Central Standard Time (CST)(U.S.)

TZ = MST7MDT
//   Mountain Time (MST)(U.S.)

TZ = PST8PDT
//   Pacific Standard Time (PST) is 8 hours earlier than Coordinated
//   Universal Time (UTC). Standard time and daylight saving time
//   both apply to this locale. By default, Pacific Daylight Time
//   is one hour ahead of standard time (that is, PDT7).

TZ = JST-9
//   Japanese Standard Time is (JST) 9 hours earlier than Coordinated
//   Universal Time (UTC). Daylight saving time doesn't apply in
//   this locale.

TZ = GMT0BST        // England, Wales, Scotland, Northern Ireland
TZ = CET-1CEST      // European Central Time
TZ = EET-2EEST      // European Eastern Time

TZ = AWST-8AWDT     // Australian Western Standard Time
TZ = AEST-10AEDT    // Australian Eastern Standard Time
```

For more information on simulation time, see "Simulation time" on page 199.

Sample sps.settings file

The following sample file shows some options specified through the sps.settings file. Other options are commented out, indicated by //.

```
// sps.settings
//
// memory settings
SaiDefaultSize = "d=2m idh=1m imem=4m"
SaiPreprSize = "d=1m idh=1m imem=1m"
SaiTransSize = "d=2m idh=2m imem=4m"

SaiGrafrSize = "imem=2m"
//
//
// Any of these settings can be entered with a ".module"
// suffix, to restrict the setting to that module.
```

```
//
// example:
// font = medium
// font.trans = small
// font.tport = large

// fonts: small, medium, large, huge
font = large
font.trans = medium
//
// colors:
// blue, red, green, cyan, magenta, yellow, orange, dkgreen,
// brown, dkblue, purple, pink, dkgray, ltgray, black, white
fg = black

bg = white
ffg = black
fbg = white
ffg.trans = blue

//
// timezone
tz = PST8PDT // west coast U.S., with DST enabled
tz = CST6CDT // central U.S., with DST enabled
```

Environment variables

Certain SPS settings can only be set using system environment variables, such as the location of SPS program files and other interfacing parameters. Other settings can be set either with an environment variable, an sps.settings file, or perhaps a command line argument.

For any setting that can be managed in an sps.settings file, GL recommends that you use this method, rather than a system environment variable. Unlike an environment variable, you can associate a specific sps.settings file with specific model, effectively customizing the settings of each model as needed without any further maintenance. An environment variable, however, applies globally to all simulations, and must be changed manually each time you want to change simulation settings. For more information on sps.settings files, see "sps.settings files" on page 91. For more information on supported command line arguments, see "Running SPS programs from command line" on page 62.

To set environment variables in Windows

Note: This procedure may vary depending on your opearting system.

1. From the Windows desktop, select **Start > Settings > Control Panel**.
2. In the **Control Panel** window, double-click **System**.
3. Select the **Advanced** tab.
4. Click **Environment Variables**.
5. Edit, add, or delete environment variables as desired.

   Notes: You must restart SPS before these environment variable changes will take effect.

"User" environment variables are accessible only under the current login profile. If you want your environment variable settings to be available to all users on a certain PC, you must maintain them in the "system" variable list.

To set environment variables from the command line (system prompt)

You may set environment variables from the command line (system prompt) by using the `setenv` command. This method sets the environment variables temporarily only while the corresponding process is running. Note that the syntax varies, depending on which operating system and shell you may be using.

On a DOS prompt, enter:

```
setenv VARIABLE VALUE
```

Under c shell enter:

```
setenv VARIABLE "VALUE"
```

Under Bourne shell enter:

```
VARIABLE "VALUE"
```

For example, if you want to change the folder where SPS looks for the license file, at the system prompt, enter the following:

```
setenv DREMLICENSE C:\NewLicenseFolder
```

To set environment variables from the SPS command line

In the SPS command line, enter the program name, followed by the command `-env` and the setting you want to change:

```
-env('setting1=value','setting2=value'
```

For example, if you want to change the directory to search when looking for a display (DSP) file, you may do so through the DREMPATH_DSP environment variable. From the SPS command line, enter:

```
-env('DREMPATH_DSP=C:\MyNewDisplayFolder'
```

Cases

Each SPS model uses a variety of input and output files, such as the INPREP, RESTRT, and INTRAN files. By default, SPS looks in the main model directory for input files, and places output files in the same directory. For example, if you open an INPREP file and run PREPR, by default the RESTRT file will be placed in the same directory as the INPREP file.

If you are a Windows user and you want to specify different directories and/or names for input and output files, you can use a case. For example, you might want to use the same source files for different simulations, but produce different output files based on simulation events. A case, stored as a case file, allows you to specify a specific path for each file type associated with an SPS model. If you have a case associated with a model, SPS will load it when you run a module

and handle files as specified. Note that cases are available on Windows only and are used by the SPS startup window and/or Model Builder.

To have a case automatically associated with a model upon a model load, its associated case file must have the same root name as the model. For example, if you open the MyModel.inprep file and SPS finds a MyModel.cas file in the same directory, that case will automatically be applied. Otherwise, you must load the case after the model is loaded.

If no case file is associated with a model, SPS assigns a default case to that model, in which all input and output files are given the same root name and stored in the same original directory.

You may also specify different names for model-related files when you run an SPS program from the command line. For more information on command line arguments, see "Running SPS programs from command line" on page 62.

For more information on SPS-related files, see "SPS Programs and Files" on page 45.

To manage a case on Windows

1 Load the model to which the case is or will be associated.

Note: A case is generally specific to a model. Therefore, you must have a model loaded before you can work on its case.

2 In the **SPS startup window**, click **Options**.

—or—

In **Model Builder**, select **Tools > Preferences**.

3 In the **Files** tab, you can:

- Load a case
- Create a new case
- Edit an existing case

Tip: If a file path is unspecified in the case, SPS uses default behavior for that particular file. Therefore, if you do not require a special name or location for a file, you can leave its location blank.

Setting preferences in the SPS startup window

You can set display preferences and allocate memory in SPS through the SPS startup window. For more information on using other methods to set environment settings, see "SPS environment settings" on page 87.

Setting preferences for the display through the SPS startup window

You can set default display information for TRANS and TPORT such as font size, window size, text and chart (plots) foreground color, and report and plot background colors through the user interface. Save the display options to use your selections whenever you run SPS. You may alternately choose to set display preferences in an sps.settings file. See "sps.settings files" on page 91.

To change the simulation displays settings

1 In the **SPS startup window**, click **Options**.

2 In the **Simulator Settings** dialog box, click the **Display** tab.

3 Under **Display**, choose a **Font size** from Small to Huge.

4 Next to **Columns** and **Rows**, type the desired number of characters in a column and the number of rows (depth) to be displayed for the main window.

5 Under **Colors**, choose a color for the foreground and background in a text display and for the foreground and background in a chart.

6 Click **Save**.

Allocating memory through the SPS startup window

From the Memory tab, you can adjust memory parameters to fit the size of the model being run for more efficient runs of small models. The three different types of memory used by SPS are imem, d, and idh. The actual amount of imem, d, and idh memory used displays in a memory summary in OUTPRP and OUTTRN. You can take the actual values, multiply by 1.25 and round to the nearest 1000.

Use the PREPR, TRANS and TPORT settings to set the memory parameters for each program, respectively. Use the Default settings to adjust memory parameters for other programs started from the SPS startup window Tools menu, such as Demac.

If the memory allocations are insufficient, the model will either not run or may run extremely slowly.

Note: You may alternately choose to allocate memory in an sps.settings file. See "sps.settings files" on page 91.

To allocate memory

1 In the **SPS startup window**, click **Options**.

2 In the **Simulator Settings** dialog box, click the **Memory** tab.

3 Type the changes to imem, d, and idh for PREPR, TRANS, and TPORT.

4 Click **Save**.

Choosing the font for displays

From the TRANS/TPORT window, you can change the current font, size, and style of text on a display. If you want to define persistent display settings in a file with other environment settings, see "Fonts" on page 93.

Note: You may alternately choose to set font characteristics in an sps.settings file. See "sps.settings files" on page 91.

To set the font in TRANS/TPORT

1 Select **Windows > Set Font**.

2 In the **Font** dialog box, select a font, font style, and font size.

3 Click **OK**.

Note: All changes made to the font through the Font dialog box appear in the current TRANS/TPORT session. Redo the changes if you quit and restart.

CHAPTER

4

SPS Model Services Management

SPS Model Services Management is a service and components used by SPS that allow you to manage all SPS components (TRANS, TRANSSHARE, etc.) necessary for running an online model service.

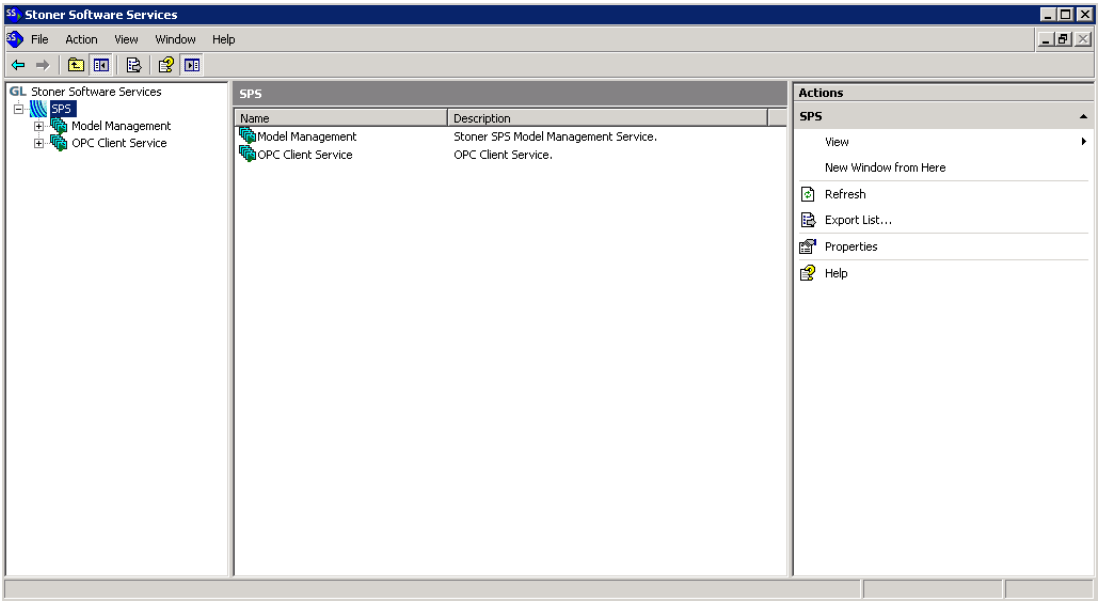
Managing online model services in SPS

Use SPS Model Services Management to manage model services and determine the status of online running model services. You can also view and add model services, and configure the settings used by model services and their components. SPS Model Services Management is accessed through the Stoner Software Services window.

To open the Stoner Software Services window

From the Windows Start menu, select **All Programs > Stoner Software > Stoner Pipeline Simulator 9.8 > Online Model Management.** The Stoner Software Services window opens and displays the SPS service node.

Note: If Stoner Services for SPS is not installed, the SPS service node does not appear.



Stoner Software Services window

Adding an online model service

The following procedure explains how to add a model service and define the Start Model action (configuring the INTRAN and Restart files, and setting the working folder). Once defined, the Start Model action is enabled and can be used to start the model service you created.

To add an online model service

1. Select **All Programs > Stoner Software > Stoner Pipeline Simulator 9.8 > Online Model Management.** The Stoner Software Services window opens and displays the SPS service node.
2. Expand the **SPS** service node. The Model Services Management node appears.
3. Right-click **Model Services Management**, and then click **Add Model**.

   —or—

   Click **Model Services Management**, and then in the **Actions Pane**, click **Add Model**. A New Model node appears in the Model Services Management tree.
4. In the tree view, click the new model. Properties for the new model appear in the **Properties** view.
5. In the **Properties** view, click the **Intran File** button. The Intran File dialog box appears.
6. Browse to the Intran file, and then click **Open**. The path to the Intran file appears in the Intran File box.
7. Click the **Restart File** button. The Restart File dialog box appears.
8. Browse to the Restart file, and then click **Open**. The path to the Restart file appears in the Restart File box.
9. Click the **Working Folder** button. The Browse for Folder dialog box appears.

   Note: The working folder is the folder from which TRANS runs when Action-Start Model is performed from the Actions Pane.
10. Browse to the Working folder, and then click **OK**. The path to the working folder appears in the Working Folder box.
11. Click **Apply**.
12. Right-click the model, and then click **Properties**. The Model Properties dialog box opens.
13. On the **General** tab, set the **Start Mode** property to **Automatic** to automatically start TRANS and TRANSSHARE.
14. Click **OK**.
15. In the tree view, right-click the model, and then click **Start Model**.

    Note: A green arrow indicates the model is started; a red dot indicates the model is stopped.

Managing model services

You can add and manage model services by right-clicking a model service in the Stoner Software Services window tree view and selecting an option from the shortcut menu. These tasks are also available from the Actions Pane. For more information, see “Using the Actions Panel for model service management” on page 109.

To add a model

1 Open the Stoner Software Services window.

2 In the Stoner Software Services window, right-click **Model Services Management**, and then click **Add Model**. A New Model node appears in the Model Services Management tree view.

3 Click the new model. Properties for the new model appear in the Properties view.

To start a service or model

1 Open the Stoner Software Services window.

2 In the Stoner Software Services window, right-click **Model Services Management**, and then click **Start**.

—or—

Expand **Model Services Management**, right-click a model, and then click **Start Model**.

Note: A green arrow indicates the model or service is started; a red dot indicates the model or service is stopped.

To stop a service or model

1 Open the Stoner Software Services window.

2 In the Stoner Software Services window, right-click **Model Services Management**, and then click **Stop**.

—or—

Expand **Model Services Management**, right-click a model, and then click **Stop Model**.

Note: A green arrow indicates the model or service is started; a red dot indicates the model or service is stopped.

To restart a service or model

1 Open the Stoner Software Services window.

2 In the Stoner Software Services window, right-click **Model Services Management**, and then click **Restart**.

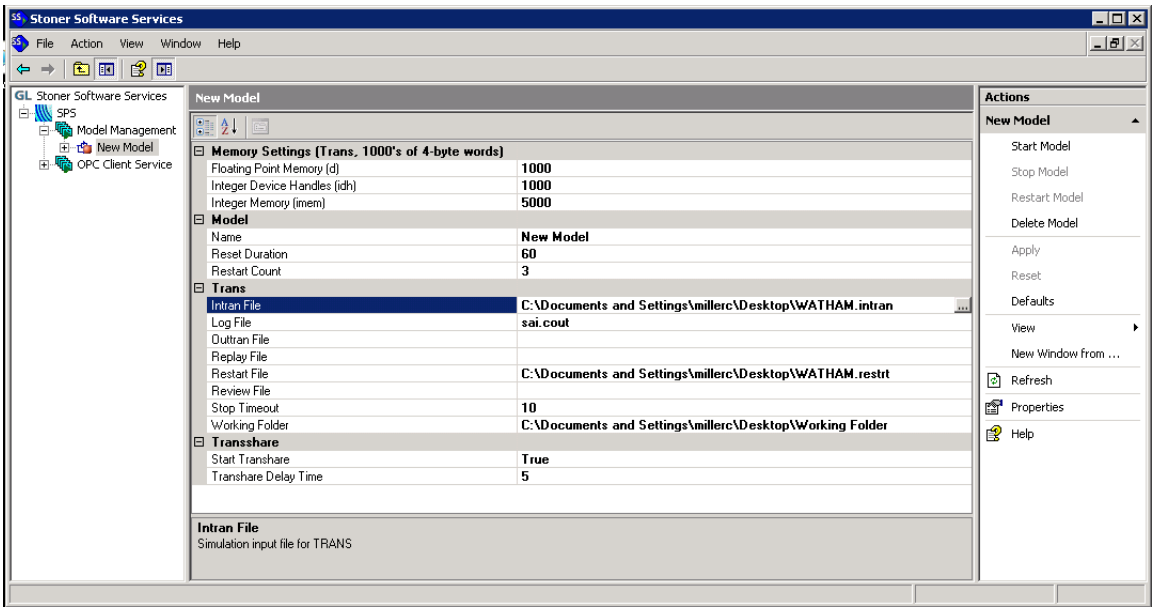
—or—

Expand **Model Services Management**, right-click a model, and then click **Restart Model**.

Note: A green arrow indicates the model or service is started; a red dot indicates the model or service is stopped.

Setting model service properties

The Properties View appears when you select a model from the Stoner Software Services Console Tree. Use the Properties view to set the INTRAN and Restart file, and the Working Folder locations. You can also set other properties if necessary. See the following table to view all available properties.



Stoner Software Services Model Management with Properties view

Use the properties in the following table to configure a model.

Note: The INTRAN and Restart files and the Working Folder settings are required to create and save a model.

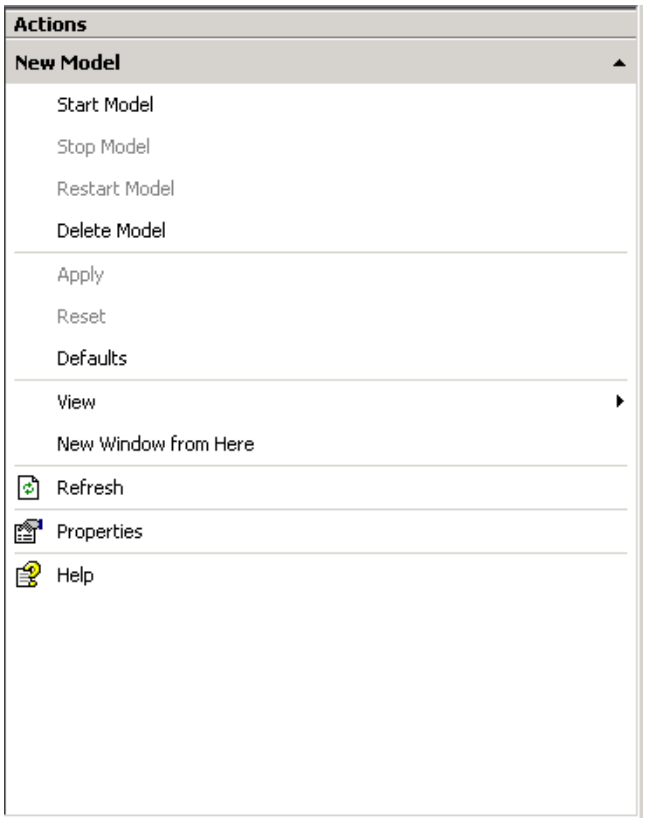
Memory Settings (1000's of 4-byte words)	• Floating Point Memory (d) • Integer Device Handles (idh) • Integer Memory (imem)
Model	• Name - Name given to the current model that helps to identify the model. The default value is "New Model". • Reset Duration - The amount of time (in minutes) that elapses before resetting the Restart Count. • Restart Count - The number of times the service will restart TRANS if it fails to run. The default value is "3". **Note:** For example, a value of -1 means the service will restart TRANS an infinite number of times if it fails to run.

Trans	• Intran File - (Required) Specifies the INTRAN file for the SPS model. Click the ellipsis to select an INTRAN file located in a user-specified directory. **Note:** The Action Start Model will be disabled from the Actions view if the INTRAN file is not specified. • Log File - Log file that is created in the specified working folder when TRANS runs. • Outtran File - (Optional) If an Outtran file is specified, it is created in the user-defined directory; otherwise, it is created in the working folder. • Replay File - (Optional) If a Replay file is specified, it is created in the user-defined directory; otherwise, it is created in the working folder. • Restart File - (Required) Specifies the Restart file required by TRANS to run. This file is created by running PREPR, and must be done prior to using the Model Service Management to run TRANS for the model. **Note:** The Action Start Model will be disabled from the Actions view if the Restart file is not specified. • Review File - (Optional) If a Review file is specified, it is created in the user-defined directory; otherwise, it is created in the working folder. • Stop Timeout - Specifies the time interval that is used to kill the TRANS and TRANSSHARE processes if they do not stop after Action - Stop Model is performed from the Stoner Software Services window. • Working Folder - (Required) Specifies a folder from which TRANS runs when Action - Start Model is performed from the Stoner Software Services window. **Note:** The Action Start Model will be disabled from the Actions view if the Restart file is not specified.
Transshare	• Start Transhare - If set to “True”, specifies whether Model Services Management should start the Transshare process. The service manages the Transshare process that it starts. If the Transshare process dies unexpectedly, the service restarts it as long as the associated TRANS is running. If set to “False”, the service will not start or manage the Transshare process. **Note:** You can start the TRANSSHARE process with a background command from the INTRAN file. • Transhare Delay Time - Used by the service as a delay interval to start Transshare after TRANS starts.

Using the Actions Panel for model service management

The Actions pane is accessed from the Stoner Software Services window, and contains the actions that are available to you based on the currently selected item. You can start, stop, restart, and delete a model; apply, reset, and set defaults; and view properties for a model. These tasks are also available from the Console Tree by right-clicking the Model Services Management node or model node and selecting an option from the shortcut menu.

Note: If the Actions pane is not shown on the Stoner Software Services window, from the toolbar click **View > Customize**. In the **Customize View** dialog box, select the **Action Pane** option, and then click **OK**.



Actions Pane - Model Services Management

To start a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Start model**. This action starts TRANS and TRANSSHARE (if it is configured) for the current model. This action requires that properties be set for the INTRAN and Restart files, and the Working Folder.

To stop a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Stop Model**. This action stops the running TRANS and TRANSSHARE if it was started by the current model service.

To restart a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Restart Model**. This action stops the currently running model (TRANS and TRANSSHARE), and then starts it again.

To delete a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Delete Model**. This action deletes the current online model configuration.

To apply changes to a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Apply** to apply the changes made to the current model properties in the Properties grid.

To reset the last edit action for a model service from the Actions Pane

1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Reset**. This action resets the last edit action for the previous configuration setting.

To restore model service defaults from the Actions Pane

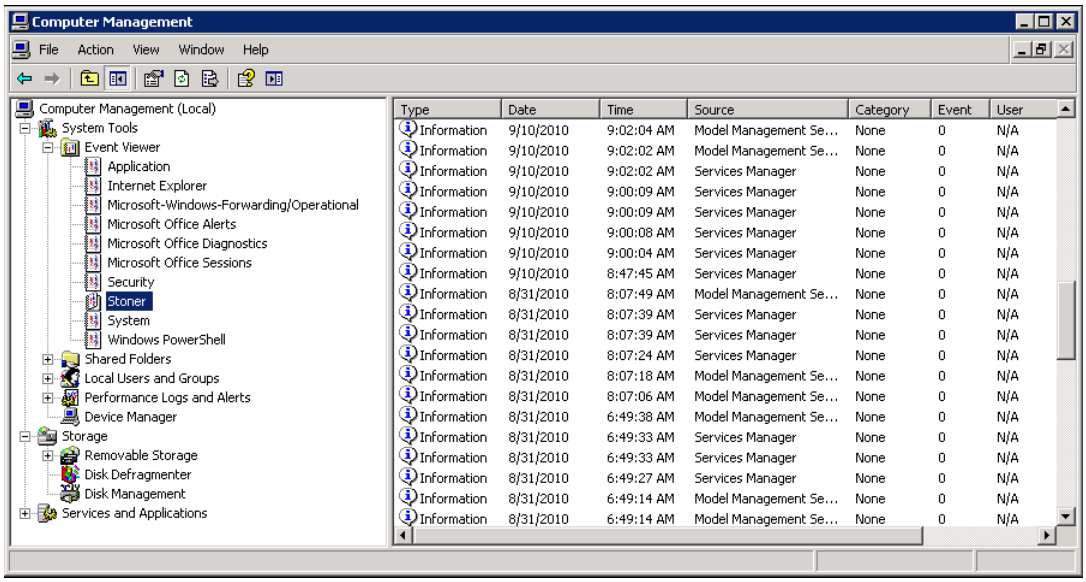
1. Expand the **Model Services Management** node, and then click a model.
2. In the **Actions Pane**, click **Defaults**. This action resets the default settings for the selected model.

Viewing the Stoner Services Event Viewer Log

All events generated by Model Services Management and OPC Client Service are logged under the Stoner node in the Microsoft Windows Event Viewer.

To view the Stoner Event Viewer log

1. On the desktop, right-click **My Computer**, and then click **Manage**.
2. Expand **Event Viewer**, and then click **Stoner**. The Stoner Event Viewer log appears in the right pane.



Stoner Services Event Viewer Log

CHAPTER

5

Configuring the OPC Client Service

The OPC Client Service is used to configure OPC connections, and allows you to specify the configuration file required to run an instance of the OPC connection. The OPC connection application (OPCtoSPS) is a proprietary OPC client that collects live pipeline data from a SCADA system for use by various Stoner Software products, including SPS.

The following procedure explains how to add a connection and define the Start Connection action (setting the configuration file, and the user ID and password). Once defined, the Start Connection action is enabled and can be used to start the connection you created.

To add an OPC Client Service connection

1. From the Windows Start menu, select **All Programs > Stoner Software > Stoner Pipeline Simulator 9.8 > Online Model Management.** The Stoner Software Services window opens and displays the SPS service node.
2. Expand the **SPS** service node, and then expand the **OPC Client Service** node. All connections managed by the service are displayed.

   Note: A green arrow indicates the connection is started; a red dot indicates the connection is stopped. If the service is not running, then the connection icon does not indicate a start or stop status.
3. Right-click the **OPC Client Service** node, and then click **Add Connection**.

   —or—

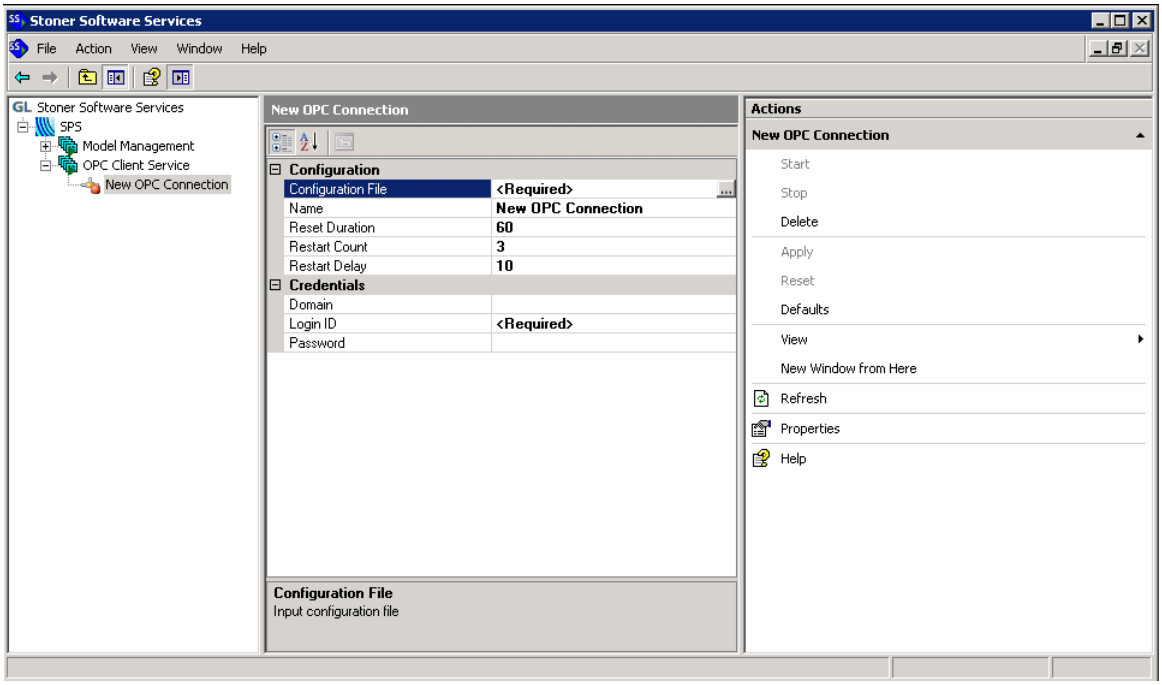
   Click **OPC Client Service**, and then in the **Actions Pane**, click **Add Connection**. A new Connection1 node appears in the OPC Client Service tree.
4. Click the new Connection1. Properties for the new Connection1 appear in the **Properties** view.
5. In the **Properties** view, click the **Configuration File** button. The Configuration File dialog box appears.
6. Browse to the Configuration file, and then click **Open**. The path to the Configuration file appears in the Configuration File box.
7. In the **Login ID** box, type the user name used by the service to start the OPCToSPS application.
8. In the **Password** box, type the password for the user name account.
9. Click **Apply**.
10. In the tree console, right-click the connection, and then click **Properties**. The OPC Connection Properties dialog box opens.
11. On the **General** tab, set the **Start Mode** property to **Automatic** to automatically start the connection.
12. Click **OK**.

13 In the tree view, right-click the connection, and then click **Start**.

Note: A green arrow indicates the connection is started; a red dot indicates the connection is stopped.

Setting connection properties

The Properties View appears when you select a connection in the Stoner Software Services Console Tree. Use the Properties view to set the Configuration file, and user ID and password. You can also set other properties if necessary. See the following table to view all available properties.



Stoner Software Services OPC Client Service Connection with Connection Properties view

Use the properties in the following table to configure a connection.

Note: The Configuration file, and User ID and password settings are required to create and save a connection.

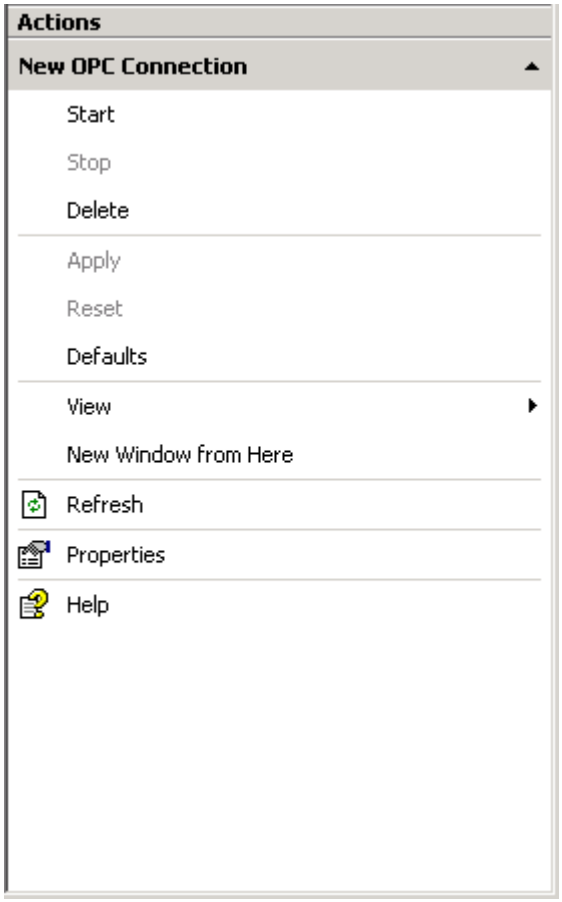
Configuration File	The Configuration file (*.cfg) that defines the configuration for the OPCToSPS application.
Name	Name that is used to help identify the connection. The default value is “New OPC Connection.
Reset Duration	The amount of time (in minutes) that elapses before resetting the Restart Count. The default value is 60 minutes.
Restart Count	Number of times the service will restart OPCToSPS if it fails to run. If a value of -1 is specified, the service will restart OPCToSPS an infinite number of times.
Restart Delay	Time the service will take to restart OPCToSPS if it fails to run. The default value is 10 seconds.

Domain	Used to specify the domain of the user account whose credentials are used to run the OPCToSPS application.
Login ID	User name or username@domainname used by the service to start the OPCToSPS application.
Password	Valid password used to log on the user.

Using the Actions Pane for connection management

The Actions pane is accessed from the Stoner Software Services window, and contains the actions that are available to you based on the currently selected item. You can start, stop, restart, and delete a connection; apply, reset, and set defaults; and view properties for a connection. These tasks are also available from the Console Tree by right-clicking the connection node and selecting an option from the shortcut menu.

Note: If the Actions pane is not shown on the Stoner Software Services window, from the toolbar click **View > Customize**. In the **Customize View** dialog box, select the **Action Pane** option, and then click **OK**.



Actions Pane - Connections Management

To start a connection from the Actions Pane

1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Start**. This action starts OPCToSPS for the current connection. This action requires that the Configuration File, Login ID, and password be set.

To stop a connection from the Actions Pane

1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Stop**. This action stops the running OPCToSPS application.

To delete a connection from the Actions Pane

1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Delete**. This action deletes the current connection.

To apply changes to a connection from the Actions Pane

1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Apply** to apply the changes made to the current connection properties in the Properties grid.

To reset a connection from the Actions Pane

1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Reset**. This action resets the last edit operation for the selected connection.

To restore connection defaults from the Actions Pane

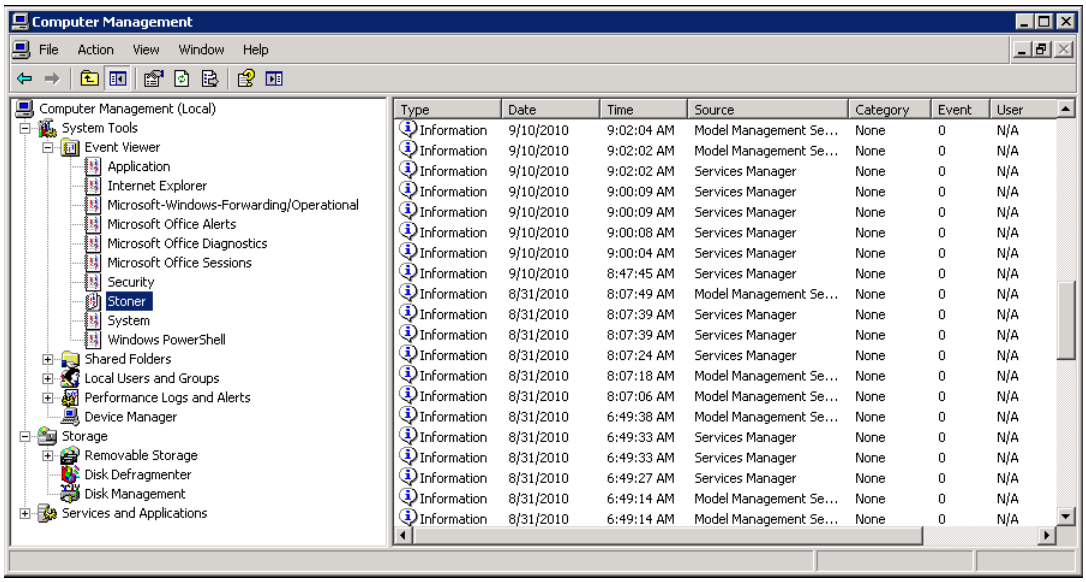
1. Expand the **OPC Client Service** node, and then click a connection.
2. In the **Actions Pane**, click **Defaults**. This action resets the default settings for the selected connection.

Viewing the Stoner Services Event Viewer Log

All events generated by Model Services Management and OPC Client Service are logged under the Stoner node in the Microsoft Windows Event Viewer.

To view the Stoner Event Viewer log

1. On the desktop, right-click **My Computer**, and then click **Manage**.
2. Expand **Event Viewer**, and then click **Stoner**. The Stoner Event Viewer log appears in the right pane.



Stoner Services Event Viewer Log

CHAPTER

6

Modeling the Physical System

You may approach building a model in several ways. You can manually enter data in an INPREP file or you can use an SPS model builder. This section focuses on model building concepts that apply to all methods of model building. If you are using SynerGEE as a model builder, see the SynerGEE documentation. If you are using SPS Model Builder, see "Model Builder" on page 697. If you are building a model using text files, see "The INPREP File" on page 205 for detailed data entry descriptions.

Before you build a model

Before you build a model you need to gather the appropriate data. You also need to understand what is relevant to the model and what is not. This section summarizes basic model data requirements as well as components to ignore when building a model.

Model data requirements

Pipe Information

- Diameter
- Length (specify whether units are actual pipe length or horizontal distance)
- Wall thickness
- Elevation profile (especially at locations where other equipment exists, such as valves, pumps or compressors, and pressure or flow instrumentation)
- MAOP and/or LAOP
- Piping material
- Young's modulus of elasticity
- Pipe friction factors (Darcy-Weisbach or Fanning friction factor, or internal roughness)

Valve Data

- Size
- Type (ball, gate, etc.)
- Frequency of operation (operated versus never operated; do not model valves that are never operated)

- Flow characteristics at various valve positions (Cv versus percent open)
- Valve operator speed and characteristics (time to open, time to close, and percent open with respect to time during valve open or close operations, if available)
- Check valve characteristics (include “damped” or “undamped”)
- Description of station discharge (or station suction) control valves, including types of set points (suction pressure set point, discharge pressure set point, flow rate set point)

Pump data (if liquid)

- Pump performance data (head, efficiency, horsepower, or torque versus flow rate)
- Rated conditions (conditions at the best efficiency point for head, flow, speed, and torque)
- Rotor polar moment of inertia *r2 or equivalent *r2, as viewed from the pump end for the driver, coupling, gearbox, pump and enclosed fluid, as applicable
- Pump characteristics diagram/synoptic chart (if not available, use curves from a similar speed pump)
- Driver type (induction motor, synchronous motor, turbine, etc. This is optional.)
- Driver torque versus speed (optional, useful for studying pump transients, such as those related to pump startup)
- Station controls description (minimum flow rate shutdown, low discharge pressure shutdown, etc.)

Centrifugal Compressor data (if gas)

- Centrifugal compressor performance data (head, efficiency, flow rate versus speed)
- Turbine performance characteristics (maximum turbine power, ambient temperature versus speed, minimum speed, maximum speed)
- Driver type (induction motor, synchronous motor, turbine, etc. Optional.)
- Station control methods (minimum flow rate shutdown, low discharge pressure shutdown, exhaust temperature shutdown, etc.)

Reciprocating Compressor data (if gas)

- Compressor cylinder data (diameter, stroke, speed range, etc.)
- Unloading equipment descriptions
- Driver type (integrated driver or separable)
- Station control methods (minimum flow rate shutdown, low discharge pressure shutdown, exhaust temperature shutdown, and description of control scheme, etc.)

Fluid Properties

- Basic description of each fluid (including a name for each fluid, liquid only)
- Density or API gravity (liquid)
- Viscosity or viscosity profile (liquid)
- Pressure bulk modulus (liquid)
- Temperature modulus or temperature modulus profile (liquid)

- Vapor pressure (liquid)
- Gas composition or specific gravity (gas)
- Custody transfer conditions

Boundary conditions

- Pressure and/or flow set points at sources
- Constant flow outlets or inlets
- Constant pressure outlets or inlets
- Types or control at all supplies and deliveries

Operational Data

- Normal startup and shutdown procedures
- Emergency operational procedures
- Constraints on pipeline and equipment operation

Schematics

- Detailed station drawings
- Piping network drawings, alignment sheets, etc.

Units

Units of measurement for all data

Components to ignore when building a model

Many components in a gas or liquid pipeline are required to operate the system in a safe and economical manner. However, some of these components are not required in the numerical model for the pipeline hydraulics. To determine which physical components and hardware on a pipeline can be ignored, base your decision on whether the equipment changes any of the following:

- pressure
- flow rate
- temperature (for thermal analysis)

If the equipment restricts the flow, generates a pressure loss, or changes the fluid properties, the effect on hydraulic performance of the element or hardware probably should be included in the model. The evaluation of the equipment is

based on the impact on the hydraulics and not the size or function of the equipment. The following table summarizes some of the equipment that may be ignored.

Component	Explanation
Thermal relief valve	These relief valves are provided on above ground piping for thermal expansion due to solar heating of the pipe. They are usually very small in size and capacity, say 1/2" to 3/4" with low flow rates (1-3 barrels per minute). They will not affect the transients even if they were to open because of their small size and capacity. They can be omitted from the model.
Low point drain valves or blow-off valves	These types of valves are manually operated and very small (2"-4"). They are used when the pipeline is being de-pressurized. They have no effect on the hydraulics.
Manual block valves	In-line, manual block valves can be omitted from the model if the pressure loss is small. Usually most in-line block valves on the main pipeline are “full port” gate valves or ball valves with very little pressure loss. These full port type valves are used in order to pass a scraper, instrument “pig,” or other type of element. In tank farm areas or gas/liquid process areas, the block valves can be reduced port or other types of valves (globe, butterfly, plug), which will have larger pressure losses because scrapers and pigs do not traverse these lines. You should be careful in considering the impact of the valves in these areas. **Note:** If a block valve is partially closed, the pressure loss will increase. If this is a normal condition (partial closure), the block valve or at least the pressure loss effects should be included in the model.
Liquid prover loops	In liquid pipeline systems, the “prover loop” is installed at or near the meters. The prover loops are used to calibrate or “prove” the flow registered by the meter. Prover loops may be in service continuously or intermittently. Even in continuous service, the flow rate does not change and the pressure loss through the loop is very small when compared to the main pipeline. Prover loops do not need to be included in the model. If the pressure loss is of concern, the losses can be included with the meter elements if the prover is in operation during the conditions being simulated.
Scraper traps (pig launcher/receiver)	Pipeline (gas and liquid) may have pig or scraper launching and receiving facilities at several locations. Normally the flow under operating conditions bypasses the launcher/receiver. They do not need to be included in the numerical model unless the simulation is for a trainer application. In some Trainer applications, the movement of the pig is part of the training simulation. In these cases, the operation of the valves for the launcher/receiver may be necessary for the simulation.

Component	Explanation
Bends, tees, elbows, eccentric/concentric reducers	Minor losses (such as bends, tees, and elbow) can be combined with other elements such as operating valves or an equivalent friction factor for the headers. Individual bends, tees, and elbows do not need to be modeled. If there are a large number of these types of fittings, they can all be combined as a single pressure loss element (header or valve) on the suction or discharge header at the pump station / compressor station. Reducers (eccentric or concentric) can be treated in the same manner.
Changes in pipe internal diameter	Small changes in pipe internal diameter can be ignored. For example, in a 24-inch nominal pipe, there are several wall thickness for various pressure rating. These different wall thicknesses for the 24-inch pipe will generate slightly different internal diameters. For example, per API Spec. 5L -: 24" 0.312" 23.376" 24" 0.375" 23.250" In this case, the same internal diameter (say 23.25") could be used. You should consider using the smaller diameter because it will generate slightly higher velocities (a more conservative surge response). Another consideration would be pipes of a different diameter but very short in length. For example, in long pipelines, the pipe may cross several rivers, roads, railroads, or other special conditions and have a thicker pipe wall or be one to two pipe sizes larger or smaller at the crossing. The lengths of the crossing depend on the physical layout; however, these special crossings are usually 200-1000 feet (0.04 to 0.20 miles). In long cross-country pipelines of 200-600 miles, these short lengths of slightly different pipes can be ignored.

There are several types of equipment that can either be included directly in the model or their effects combined with other elements.

Component	Explanation
Meter skids (pads)	Flow measuring devices come in a variety of types and functions. The traditional types for gas flow are orifice and/or venturi type. These devices create a pressure drop in the pipe. From this pressure loss and characteristics of the device, a flow rate can be computed. You should consider the magnitude of the pressure loss in evaluating whether to include the meter in the model. In liquid systems, flow is usually measured with a mechanical device like a turbine or positive displacement meter per ANSI/API MPMS 6.6 Section 6. The meter skid contains several block valves, check valves, optional straightening vanes and fittings. The whole meter skid can be treated as a unit with a pressure drop per flow path. In addition to the meter skid, a backpressure control valve may be provided downstream of the meter. The meter skid and backpressure valve can be combined into one element or regulator in the model.
Filters/strainers	Compressor stations and pump stations in a pipeline system may be equipped with filters or a strainer (usually on the suction piping). If these devices are properly maintained, the pressure loss is small (2-6 psi) for clean devices. As the filters/strainers become dirty, the pressure loss increases. You should consider the magnitude of the pressure loss for both clean and dirty conditions when determining if the devices should be included in the model.

Model options

Model options control basic settings that apply to the entire model, including phase selection, custody conditions, units selection, the equation of state, and the thermal mode. Some of these global settings are required. For more information on which settings are required, see “Required input for the INPREP file” on page 205. Global settings include the following:

- "STATE TABLE" on page 245
- "ISOTHERMAL" on page 247
- "THERMAL" on page 248
- "TRANSTHERMAL" on page 249
- "ENGLISH" on page 254
- "METRIC" on page 255
- "DEFUNITS" on page 256
- "USEUNITS" on page 258

Phase selection

SPS has been designed to handle both liquid and gas fluids flowing through pipeline systems, though not at the same time. SPS does not support generalized two-phase flow. See "Column separation (liquid only)" on page 148 and "Slack line flow" on page 148 for more information on what types of flow SPS does support.

Equations of state

You can choose from a variety of mathematical expressions, or *equations of state*, that describe the physical state of the fluid(s) in the pipeline. SPS supports the following equations of state, each with unique strengths that help you model specific problems.

Gas equations of state

Equation of State	Strengths
"STATE BWRS" on page 224	• Multi fluid simulations. • Gases at high pressure or mixtures of dissimilar fluids, including mixtures of hydrocarbons and acid gases. • Liquid systems such as LPG, or dense phase fluids like ethylene or carbon dioxide. • Not suitable for simulation near the two-phase region.
"STATE AGA" on page 221	• Multi fluid simulations. • Track user-defined fluid properties, as well as specific gravity, heating value, and/or percent compositions. • Works well for the range of pressures and temperatures typical for natural gas transmission systems.
"STATE CNGA" on page 228	• Natural gas. • Tracking of specific gravity. • Works well for the range of pressures and temperatures typical for natural gas transmission systems. • Only the gas specific gravity is needed to use this equation.

Liquid equations of state

Equation of State	Strengths
"STATE SCL" on page 230	• Multi-fluid liquid simulations. • Batched and/or product liquid pipelines. • Non-Newtonian Bingham Plastic flow (see "VISCOSITY (non-Newtonian)" on page 239). • Wax deposition (see "WAX" on page 242).
"STATE BWRS" on page 224	• Multi fluid simulations. • Gases at high pressure or mixtures of dissimilar fluids, including mixtures of hydrocarbons and acid gases. • Liquid systems such as LPG, or dense phase fluids like ethylene or carbon dioxide. • Not suitable for simulation near the two-phase region.
"STATE TABLE" on page 245	• Single-fluid liquid. • Temperature effects. • Can simulate almost any fluid, even if the fluid does not fit traditional correlations.

Thermal modes

SPS offers several thermal modes of simulation.

- Isothermal (see "ISOTHERMAL" on page 247)
- Thermal (see "THERMAL" on page 248)
- Transient-thermal (see "TRANSTHERMAL" on page 249)

For networks where thermal effects are negligible, use the *isothermal* mode, which assumes constant temperature throughout the network. Pipe end temperatures still may be entered. Where thermal gradients are strong enough to affect pressure, flow, and other system variables, SPS provides *thermal* and *transient-thermal* modes. Both of these allow input of temperature profiles for each pipe segment in the model, but the transient-thermal mode also tracks the effects of temperature changes over time.

The transient-thermal mode can be used to compute thermal transients not only in the fluid itself, but also in the surrounding environment, a feature that is useful for pipeline systems operating in temperature-sensitive environments. It also takes into account the effect of thermal interchange with the surroundings and the thermal changes caused by compression and decompression of a fluid, which may have significant effects on the fluid.

Units

As part of the general data input, SPS allows you to choose your preferred input and output units for pressure, flow, temperature, and other variables. SPS has built-in sets of units for both English and metric units systems, as well as appropriate units for gas or liquid simulations.

This section presents discussions on the following sets of units:

Internal units	Units that SPS uses for its internal calculations. In most circumstances, the SPS internal units are not relevant to the user.
User units	Units that you use to input data (like pressure, flow, length, diameter, etc.) in the INPREP file, INTRAN file, or interactively. These units are used by SPS for both input and output.
Built-in units	All of the various units that are pre-programmed into SPS, which can be selected individually using the USEUNITS command. For more information, see "USEUNITS" on page 258.
User-defined units	Any additional units that you define using the DEFUNITS command. In order to use a unit that is not built-in to SPS, the unit must be defined using DEFUNITS and then selected using USEUNITS. For more information, see "DEFUNITS" on page 256.

Values input in user units are converted to internal units for internal calculations. When TRANS sends output to the display, or to the outtran file, TRANS converts the output from internal units to user units. The result of this two-way conversion is that for both input and output, you are usually aware of the user units only.

There are, however, some cases where this conversion and units handling method does not apply. These cases are related to units handling in the REVIEW file and in the post-processor program.

Units derived from user units

In some cases, the SPS expects input (or generates output) in a unit that is derived from one or more user units. In most cases, the inputs are physical properties of a fluid. In other cases, the unit of the quantity is encouraged by industry convention.

As an example, consider the unit ΔPRESSURE; this unit is used on some of the fields of a valve and other places. You do not enter the units for ΔPRESSURE; instead, the units for ΔPRESSURE are derived from the PRESSURE unit. For the PRESSURE unit, you may select any of PSIA, PSIG, KPAA, KPAG, BARA, BARG, etc. If you select either PSIA or PSIG for PRESSURE, the PRESSURE unit is automatically PSI. Similarly, the ΔTEMPERATURE is derived from the user-selected TEMPERATURE unit.

Other more complex cases, such as HEAT.COND/TEMPERATURE, are similarly derived from the units for HEAT.COND and TEMPERATURE.

The absolute pressure unit is worth special mention. The absolute pressure unit, written $\text{PRESSURE}_{\text{abs}}$ in this document, is derived from the user-selected PRESSURE unit. The $\text{PRESSURE}_{\text{abs}}$ unit is used in various input fields where there is a strong industry convention to use absolute units, as opposed to gauge units. For example, almost universally, custody transfer pressure is expressed in absolute units. Additionally, some physical parameters such as vapor pressure are commonly expressed in absolute units. If your PRESSURE unit is either PSIA or PSIG, then your $\text{PRESSURE}_{\text{abs}}$ unit is PSIA. Similarly, if your PRESSURE unit is either KPAA or KPAG, then your $\text{PRESSURE}_{\text{abs}}$ unit is KPAA.

Gauge pressure and elevation

When using gauge pressure units, such as PSIG or KPAG, the applicable pressures are input and/or output relative to atmospheric pressure at the local elevation. The atmospheric pressure correction is calculated using the equation:

$$P_{ATM} = 14.696 \times (1 - 0.0000361 \times elev)$$

where:

P_{ATM}	=	atmospheric pressure, psia
elev	=	local elevation above sea level, feet

Units handling in the REVIEW file

The REVIEW file is where TRANS stores time plot and distance plot information for use interactively. Most types of data are stored in the REVIEW file in built-in units. For example, if you have specified that the simulation is to use metric units for input and output (i.e., metric units are user units), then pressures are typically entered in kilopascals (kpag). The built-in unit for pressure, however, is psig. When a pressure is entered in kpag, SPS converts that pressure to psig before any internal calculations are done. Pressure data stored in the REVIEW file is stored in built-in units, or psig.

When an interactive time plot is requested, the REVIEW file is read and the values in built-in units are converted to user units for display. For this example, with pressures specified in metric units, you never see any pressures expressed in psig while running TRANS.

This discussion of units handling covers most situations with the REVIEW file. For more information on two notable exceptions, see:

- "Units used for user-defined variables" on page 128
- "Control system input and output units" on page 128

Units used for user-defined variables

User-defined variables (DEFINEs) are not subject to any units conversion from user units to internal units. Consequently, if a variable is defined, then the value of that variable is stored in the REVIEW file in user units.

For example, suppose a user-defined variable IN_FLOW is defined that is equal to the flow rate into the model through some external. The value of IN_FLOW is stored in the REVIEW file in user units no matter what these units are. However, the flow rate through the external itself (not the defined variable) is stored in the REVIEW file using internal units. The significance of this is that the REVIEW file may contain values that represent the same physical quantity (in our example, flow into the model), but are numerically not the same in the REVIEW file. This is transparent when running TRANS because the external flow rate is converted back to user units before it is displayed.

Control system input and output units

Inputs to a control system are in terms of user units and are stored that way in the REVIEW file. For example, a sensor that measures the pressure at a certain point in the model has the measurement in user units. The peek value <sensor>:IN is stored in the REVIEW file in user units. The output from the sensor is strictly based on the input-to-output map that you specified for the sensor, using the sensed value in user units as the input to the sensor map.

Once a physical quantity (like pressure, flow, density, etc.) has been introduced into a control system, the concept of units becomes less important, perhaps even misleading. A better way to think of control system behavior is to use the idea of control signals (inputs and outputs) being manipulated by the control devices present. A sensor may measure pressure (a physical quantity) and output a signal that is not really the pressure but only based on the numerical value of the pressure that was the input to the sensor.

Default units

	ENGLISH		METRIC	
Keyword	**LIQUID**	**GAS**	**LIQUID**	**GAS**
AFLOW	MAB/D	AFT3/MIN	AM3/HR	AM3/HR
ANGLE	DEGREE	DEGREE	DEGREE	DEGREE
AREA	IN2	IN2	CM2	CM2
BULK.MOD	PSI	PSI	KPA	KG/M2
COMPOSITION[a]	FR	FR	FR	FR
COMP.AFLOW	n/a	AFT3/MIN	n/a	AM3/HR
DENSITY	LBM/FT3	LBM/FT3	KG/M3	KG/M3
DEPOSITION	LBM/BTU	LBM/BTU	KG/KJ	KG/KJ
DIAMETER	IN	IN	MM	MM
DUTY	MMBTU/HR	MMBTU/HR	GJ/HR	GJ/HR
EFFICIENCY	FR	FR	FR	FR
ELEVATION	FT	FT	M	M
FLOW	MB/D	MMFT3/D	M3/HR	M3/HR
FRACTION	FR	FR	FR	FR
FUEL	MB/D	MFT3/D	M3/HR	M3/HR
HEAT.RATE	BTU/HP-HR	BTU/HP-HR	KJ/KW-HR	KJ/KW-HR
HEAD	FT	FT	M	M
HEAT.CAPACITY	BTU/LBM-DF	BTU/LBM-DF	KJ/KG-DC	KJ/KG-DC
HEAT.COND	BTU/HR-FT-DF	BTU/HR-FT-DF	KJ/HR-M-DC	KJ/HR-M-DC
HEAT.TRANSFER	BTU/HR-FT2-DF	BTU/HR-FT2-DF	KJ/HR-M2-DC	KJ/HR-M2-DC
HEAT.VALUE	BTU/FT3	BTU/FT3	KJ/M3	KJ/M3
INERTIA	MLBM-FT2	MLBM-FT2	KG-M2	KG-M2
LENGTH.HEADER	MI	MI	KM	KM
LENGTH.PIPE	MI	MI	KM	KM
LINEPACK	MB	MMFT3	M3	M3
MASS	LBM	LBM	KG	KG
MFLOW	LBM/S	LBM/S	KG/S	KG/S
POWER	HP	HP	KW	KW
PRESSURE	PSIG	PSIG	KPAG	BARG
PUMP.AFLOW	MAB/D	n/a	AM3/HR	n/a
ROUGHNESS	IN	IN	MM	MM
SHEAR.RATE	1/SEC	1/SEC	1/SEC	1/SEC

	ENGLISH		METRIC	
Keyword	**LIQUID**	**GAS**	**LIQUID**	**GAS**
SHEAR.STRESS	PA	PA	PA	PA
SIG.TIME	MIN	MIN	MIN	MIN
SPEED	RPM	RPM	RPM	RPM
SWEPT.VOLUME	IN3	IN3	CM3	CM3
TEMPERATURE	DF	DF	DC	DC
TENSILE.STRESS	MMPSI	MMPSI	GPA	GPA
TIME	MIN	MIN	MIN	MIN
TFLOW	MMBTU/D	MMBTU/D	GJOULE/D	GJOULE/D
TORQUE	FT-LB	FT-LB	N-M	N-M
VALVE.CG	MB/D-PSI	FT3/HR-PSI	M3/HR-KPA	M3/HR-KPA
VALVE.COEFF	MB/D-PSI.5	FT3/HR-PSI.5	M3/HR-KPA.5	M3/HR-KPA.5
VALVE.FLOW	MB/D	MB/D	M3/HR	M3/HR
VELOCITY	FT/S	FT/S	M/S	M/S
VISCOSITY	CP	CP	CP	CP
VOLUME	MB	MMFT3	M3	M3
WALL	IN	IN	MM	MM

a. See "Units for composition" on page 130.

Units for composition

SPS supports batch and composition tracking. The units of each fluid or component can be specified either all at once or individually. Use the COMPOSITION keyword to specify the units of all of the fluids or components together. Use the COMPOSITION.<name> keyword to specify the units of an individual component, where <name> is the fluid or component name. For more information on available units for composition, see "Built-in units" on page 131.

Example 1

Using BWRS, composition is to be in percent, except for water content, which is to be in ppm.

```
USEUNITS COMPOSITION PERCENT
USEUNITS COMPOSITION.H2O PPM
```

Example 2

Using SCL, composition is to be in fraction, except for fluid DRA, which is to be in ppm (mass).

```
USEUNITS COMPOSITION FRACTION
USEUNITS COMPOSITION.DRA PPM
```

Built-in units

Class	Applicable Keywords	Built-in Name	Meaning
Area	AREA	IN2	in^2
		CM2	cm^2
Angle	ANGLE	DEGREE	degree (0-360)
Composition	COMPOSITION COMPOSITION.component	FR	fraction (0 - 1)
		PERCENT	percent (0 - 100)
		PPM	parts per million (0 - 10^6)
Density	DENSITY	LBM/FT3	lbm/ft^3
		KG/M3	$kilograms/m^3$
Distance	ELEVATION LENGTH.HEADER LENGTH.PIPE DIAMETER ROUGHNESS WALL HEAD	IN	inches
		FT	feet
		MI	miles
		MM	millimeters
		CM	centimeters
		M	meters
		KM	kilometers
		MICROIN	10^{-6}inches
		MICRON	10^{-3}mm
Duty	DUTY	MMBTU/HR	10^6btu/hr
		GJ/HR	10^9joules/hr
Efficiency	EFFICIENCY	FR	fraction (0 - 1)
		PERCENT	percent (0 - 100)

Class	Applicable Keywords	Built-in Name	Meaning
Flow (Actual)	AFLOW PUMP.AFLOW COMP.AFLOW	MAB/D	10^3bbls/day
		AB/D	bbls/day
		AB/HR	bbls/hour
		AB/MIN	bbls/min
		AGAL/MIN	gal/min
		ACFM	ft^3/min-ft^3/min
		AFT3/MIN	10^3ft^3/min
		AMCFM	10^3ft^3/min
		MAFT3/MIN	10^6ft^3/min
		MMAFT3/MIN	ft^3/hr
		ACFH	ft^3/hr
		AFT3/HR	10^3ft^3/hr
		AMCFH	10^3ft^3/hr
		MAFT3/HR	10^6ft^3/hr
		MMAFT3/HR	ft^3/day
		AFT3/D	10^3ft^3/day
		AMCFD	10^3ft^3/day
		MAFT3/D	10^6ft^3/day
		AMMCFD	10^6ft^3/day
		MMAFT3/D	10^6ft^3/day
		AM3/H	m^3/hr
		AM3/HR	m^3/hr
		AFT3/SEC	ft^3/sec
		AMEGAM3/D	10^6m^3/day
		AKILOM3/D	10^3m^3/day
		AKILOM3/H	10^3m^3/hr
		AM3/D	m^3/day
		AM3/S	m^3/sec
Flow (Mass units)	MFLOW	LBM/S	lbm/sec
		KG/S	kg/sec

Class	Applicable Keywords	Built-in Name	Meaning
Flow (Thermal)	TFLOW	MMBTU/D	10^6 BTU/day
		GJOULE/D	gigajoules/day
		KDTH/D	kilodekatherms/D
		DTH/HR	dekatherms/hour

Class	Applicable Keywords	Built-in Name	Meaning
Flow (Standard)	FLOW, FUEL	MB/D	10^3bbls/day
		B/D	bbls/day
		B/HR	bbls/hour
		B/MIN	bbls/min
		GAL/MIN	gal/min
		GPM	Gal/min
		FT3/MIN	ft^3/min
		MFT3/MIN	10^3ft^3/min
		MMFT3/MIN	10^6ft^3/min
		FT3/HR	ft^3/hr
		MFT3/HR	10^3ft^3/hr
		MMFT3/HR	10^6ft^3/hr
		FT3/D	ft^3/day
		MFT3/D	10^3ft^3/day
		MMFT3/D	10^6ft^3/day
		M3/HR	m^3/hr
		M3/D	m^3/day
		MMCFD	10^6ft^3/day
		MCFD	10^3ft^3/day
		MCFH	10^3ft^3/hr
		CFH	ft^3/hr
		CFM	ft^3/min
		MCFM	10^3ft^3/min
		MEGAM3/D	10^6m^3/day
		KILOM3/D	10^3m^3/day
		KILOM3/H	10^3m^3/hr
		M3/H	m^3/hr
		M3/S	m^3/sec

Class	Applicable Keywords	Built-in Name	Meaning
Fraction	FRACTION	FR	fraction (0 - 1)
		PERCENT	percent (0 - 100)
		PPM	parts per million (0 - 10^6)
Heat Rate	HEAT.RATE	BTU/HP-HR	btu/(hp-hr)
		KJ/KW-HR	Kj/kw-hr
Heat Capacity	HEAT.CAPACITY	BTU/LBM-DF	btu/(lbm-°F)
		KJ/KG-DC	kilojoules/(kg-°C)
Heat Conductivity	HEAT.COND	BTU/HR-FT-DF	btu/(hr-ft-°F)
		KJ/HR-M-DC	kilojoules/(hr-m-°C)
		W/M-DK	watts/m-°K
Heat Transfer	HEAT.TRANSFER	BTU/HR-FT2-DF	btu/(hr-ft^2-°F)
		KJ/HR-M2-DC	kilojoules/(hr-m^2-°C)
Heating Value	HEAT.VALUE	BTU/FT3	btu/ft^3
		KJ/M3	kj/m^3
Moment of Inertia	INERTIA	MLBM-FT2	10^3lbm-ft^2
		KG-M2	kilograms-m^2
Mass	MASS	LBM	pounds (mass)
		KG	kilograms
Power	POWER	HP	horsepower
		KW	kilowatts
Pressure	PRESSURE	PSIA	lb/in^2 (abs)
		ATMA	atmospheres (abs)
		BARA	bars
		KPAA	kilopascals (abs)
		KG/CM2A	kg/cm^2 (abs)
		PSIG	lb/in^2 (gauge)
		ATMG	atmospheres (gauge)
		BARG	bars (gauge)
		KPAG	kilopascals (gauge)
		KG/CM2G	kg/cm^2 (gauge)

Class	Applicable Keywords	Built-in Name	Meaning
Pressure Difference	BULK.MOD SHEAR.STRESS TENSILE.STRESS	PSI	lb/in^2
		ATM	atmospheres
		BAR	bars
		KPA	kilopascals
		KG/CM2	$kilograms/cm^2$
		KG/M2	$kilograms/m^2$
		MMPSI	$10^6 lb/in^2$
		GPA	$10^9 pascals$
		PA	pascals
Rotational Speed	SPEED	RPM	rpm
Temperature	TEMPERATURE	DF	°Fahrenheit
		DR	°Rankin
		DC	°Celsius
		DK	°Kelvin
Time	TIME SIG.TIME	MIN	minutes
Torque	TORQUE	FT-LB	foot-pound (force)
		N-M	newton-meter
Valve Coefficient	VALVE.COEFF	MB/D-PSI.5	$10^3 bbl/(day\text{-}psi^{.5})$
		M3/HR-KPA.5	$m^3/(hour\text{-}kpa^{.5})$
		FT3/HR-PSI.5	$ft^3/(hour\text{-}psi^{.5})$
		GAL/MIN-PSI.5	USGal/min $psi^{.5}$
Valve Coefficient Gas	VALVE.CG	CFH/PSI	$ft^3/hr\text{-}psi$
		M3/HR-KP	$m^3/hr\text{-}kpa$
Velocity	VELOCITY	FT/S	ft/s
		M/S	m/s
Viscosity	VISCOSITY	CP	centipoise
		PE	poise
		LBM/FT-S	lbm/(ft-s)
		LBM/FT-HR	lbm/(ft-hr)

Class	Applicable Keywords	Built-in Name	Meaning
Volume (Standard)	VOLUME SWEPT.VOLUME LINEPACK	MB	10^3bbls
		FT3	ft^3
		MMFT3	10^6ft^3
		IN3	in^3
		M3	m^3
		CM3	cm^3
		BBL	barrels
		KM3	10^3m^3
		MM3	10^6m^3

Modeling equipment, supplies, and deliveries

Planning the simulation

Before constructing a model, you need to collect necessary data, including any paper maps, performance curves, etc. Drawing a schematic of the network helps you better visualize the mechanics of the simulation and you can clearly define relationships between the various components. Although you may build models by entering data in an INPREP file, using a model builder provides you with an interactive visual representation of your model. For more information on using a model builder, see “Model Builder” on page 697 or the SynerGEE Gas documentation.

Literal or idealized simulation

SPS allows for detailed modeling of a system with almost one-to-one correspondence with facilities in the field. However, you might choose to use idealized elements for simulations that do not require detailed modeling.

Built-in idealized elements allow for simple ways to model real-world elements such as compressors, control valves, and supplies and loads. You may augment built-in idealized elements using a MACRO. See the discussion on each element type to determine which element best fits your modeling needs. Also see “MACRO” on page 580 to determine whether you should incorporate macros in your model.

Supplies and deliveries

Use either the supply/delivery external (E) or nodes (NODE) to model flow into or flow out of the network (such as at a sale or delivery point, reservoir, injection well, pipe rupture, flow to/from another network, etc.). A tank element (TK) may be used to model a fixed pressure tank.

Nodes versus externals

When defining elements, you must specify tie points, or *nodes*, that connect the various elements. You can use the nodes already established from the element definition to model flow into or out of the network. However, externals offer some advantages.

- Multiple externals can be connected to a node, which can make accounting simpler in situations where multiple physical supplies/deliveries occur at a single node. The system may have one flow that is constant at 200, and another flow that varies between 10 and 20. Instead of having one node flow that varies from 210 to 220, you can have one external at 200 and another external that varies between 10 and 20.
- Two or more inlet streams of different composition that vary in flow rate may be connected to a single node. This situation is more complex to handle using just a node flow, because the node flow composition would need to vary continuously.
- Externals have more flexibility in flow sign convention. Nodes and TAKE externals use the convention that flow (both Q: and :NQ) is positive if it is into the model. SALE externals use the convention that nominal flow rate (:NQ) is positive if it is out of the model. See "Fluid Flow" on page 145.
- By default, all nodes have a flow set point SQ of 0 and the node pressure P is constrained to be within a range, PMIN≤ P≤ PMAX. If you keep this default and use externals to model all of the flow into and out of the network, then the node pressure constraints will function as a kind of pressure relief valve, and the nodes will automatically start flowing to relieve any pressure violation. In this situation, any node that is flowing can be easily identified as a location where the pressure would otherwise be out of range. Note that a node can switch to pressure control regardless of whether the flow set point is 0; it is just more obvious to see the flow change from 0 to some non-zero value, compared with, for example, changing from 200 to some non-200 value.
- In large models, you may want to group flows together and take their sum by defining a user-defined variable. Externals with unique names may facilitate this as opposed to using flow at nodes, which may be restricted through naming conventions designed for another convenience, such as a geographic system.

Externals and nodes can be used together to model flow into and out of the network. Furthermore, any node can model flow into or out of the network and any externals that are connected to the node can be flowing simultaneously.

Regulating pressure or flow at nodes/externals

Pressure or flow at a node or SALE/TAKE external can be regulated:

- You can poke/ramp to the desired pressure or flow set point.
- For a pressure controlled node or SALE/TAKE, the flow limits automatically constrain the flow rate.
- For a flow controlled node or SALE/TAKE, the pressure limits automatically constrain the pressure.

Pressure or flow at a P-CONTROL, Q-CONTROL, or Q(P) external can be regulated by:

- Using a control system
- Specifying a signal generator (input reference)
- Assigning a characteristic flow/pressure data curve
- Poking/ramping the set point of the control system to the desired pressure or flow set point

Setting composition at inflows

This discussion is applicable for those equations of state that support composition tracking, i.e.:

- AGA
- BWRS
- BWRS.MOLE
- CNGA
- SCL
- SCLPROP

Additionally, the NOTRACK option must not have been selected for this discussion to be applicable. You can specify the composition of any flow that is into the model at any node, SALE, or TAKE. At any such flow, three distinct blends are applicable:

- *The composition set points for the inflowing stream.* You have direct control over the composition set points. The peek attributes for the composition set points are of the form :S<name>, where <name> is the component/fluid/property name. If flow is into the model, the composition of the flowing stream will be the same as the composition set points. If flow is out of the model, the composition of the flowing stream will be the same as the composition at the node.
- *The composition at the node after any mixing.* Any number of flows may be towards or away from a node. The node completely mixes all of the flows that are towards the node. All of the flows that are away from the node are this mixed composition. The composition at the node, after mixing, is identified by the peek attributes :N<name>, where <name> is the component/fluid/property name.
- *The flowing composition.* The flowing composition into or out of the model is identified by the peek attributes :<name>, where <name> is the component/fluid/property name. The flowing composition will be either the set point composition or the node composition, depending on whether flow is into or out of the model, respectively.

All of the composition-related attributes :S<name>, :N<name>, and :<name> are available at every node, SALE, and TAKE. In each case, :S<name> is the composition set point, :N<name> is the composition at the node (the mixed composition), and :<name> is the composition of the stream that is flowing either into or out of the model.

Additionally, for the various gas equations of state, the following attributes are also available as :S<name>, :N<name>, and :<NAME> at every node, SALE, and TAKE:

	SG	**LHV**	**HHV**	**WOB**
AGA	yes*	yes	yes*	yes
BWRS	yes	yes	yes	yes
BWRS.MOLE	yes	yes	yes	yes
CNGA	yes*	yes	yes*	no

Note: In the previous table, an asterisk (*) denotes that the corresponding set point, :S<name>, is pokable. For those that are pokable, the set point is used to set the corresponding property of the inflow. For those that are not pokable, the :S<name> attribute is still available as a peek, which is calculated from the set point composition.

For the BWRS, BWRS.MOLE, and SCL equations of state, the composition set points correspond to component fractions. These fractions should sum to one; if they do not sum to one, the software automatically normalizes them so that they do sum to one.

Setting temperature at inflows

This discussion is applicable for THERMAL and TRANSTHERMAL models. Please note that for THERMAL models, the temperature tracking is done for stations only, and stops at pipes (either T or GP).

You can specify the temperature of any flow that is into the model at any node, SALE, or TAKE. At any such flow, three distinct temperatures are applicable:

- *The temperature set point for the inflowing stream.* You have direct control over the temperature set point. The peek attribute for the temperature set points are of the form :ST. If flow is into the model, the temperature of the flowing stream will be the same as the temperature set point. If flow is out of the model, the temperature of the flowing stream will be the same as the temperature at the node.
- *The temperature at the node after any mixing.* Any number of flows may be towards or away from a node. The node completely mixes all of the flows that are towards the node. All of the flows that are away from the node have this mixed temperature. The temperature at the node, after mixing, is identified by the peek attribute :NT.
- *The flowing temperature.* The flowing temperature into or out of the model is identified by the peek attribute :T. The flowing temperature will be either the set point temperature or the node temperature, depending on whether flow is into or out of the model, respectively.

Controlling pressure, flow, or heat rate at nodes, SALES, and TAKES

At any node, SALE, or TAKE, you may specify the pressure set point SP or flow set point SQ, but not both simultaneously. Additionally, for the BWRS, BWRS.MOLE, and AGA equations of state, you may alternatively specify the heat rate set point SH.

You can change a node, SALE, or TAKE from one method of control to another simply by poking the set point that you want to be active. When you do so, any other set point automatically goes inactive and the corresponding now-inactive set points are set to 0.

For example, if the pressure set point SP at a TAKE is 100, the flow set point SQ (and heat rate set point SH, if applicable) will be 0. If you change the flow set point SQ to 50, the pressure set point SP will automatically be set to 0.

In addition to the pressure, flow, and heat rate set points, each of these items has upper and lower limits as well. The three methods of control are handled in a similar way. First, the actual control value is taken to be the set point, but bounded by the corresponding limits:

$$\mathrm{SX(actual)} = \begin{cases} \mathrm{XMIN} & \text{if } \mathrm{SX} < \mathrm{XMIN} \\ \mathrm{XP} & \text{if } \mathrm{XMIN} \le \mathrm{SX} \le \mathrm{XMAX} \\ \mathrm{XMAX} & \text{if } \mathrm{XMAX} < \mathrm{SX} \end{cases}$$

where X is either P, Q, or H, whichever is active.

If this control, XP (actual), does not cause any of the other limits to be violated,

$$QMIN \le Q \le QMAX \text{ and } HMIN \le H \le HMAX \text{ ... when SP active}$$
$$PMIN \le P \le PMAX \text{ and } HMIN \le H \le HMAX \text{ ... when SQ active}$$
$$PMIN \le P \le PMAX \text{ and } QMIN \le Q \le QMAX \text{ ... when SH active}$$

then that is what the control is and the controlled value is equal to its set point.

Otherwise, the control method automatically switches to the most constraining limit in an attempt to satisfy all of the constraints, and the set point is not met. In this case, the most constraining limit takes control of the node/SALE/TAKE, and the value that is in control will be equal to its limit.

If the limiting value later lifts off of its limit (due to changing conditions in the model), the node/SALE/TAKE will automatically switch back to the requested control method, and the set point will be met once again.

Multiple pressure, flow, and/or heat rate control at a node

A node may be set up to have multiple pressure controls, flow controls, and/or heat rate controls through externals that are connected to the node, with or without the additional control through the node itself. In this situation, there may be substantial “circulation” flow from one external to another at the node. This circulation flow occurs because the flow that other elements take from the node (or deliver to the node) is determined only by the node pressure. Because of this fact, simply adding a flow-controlled external to a node that is already pressure-controlled will have no effect on the other elements that are connected to the node.

Pipes

SPS offers the following pipe types.

Type	Strengths
“General pipe - transient (GP)” on page 261	Transient equations; GP syntax can be exported by SynerGEE.
“Transfer line - transient (T)” on page 263	Transient equations.
“Header (H)” on page 276	Steady state equations (typically used to model station piping). Idealized fixed outlet temperature or fixed ΔT heat exchangers.
“Heat exchanger (HE)” on page 281	Tube and shell heat exchanger.

Physical pipes may be modeled using any combination of the elements in the previous table. Deciding which modeling elements to use and where to use them is an important decision. It can have a dramatic impact on the validity of the model, how easy the model is to tune, and on how fast the model runs.

Use either the transfer line (T) or the general pipe (GP) to model each pipe of significant length compared to the time scale of interest, usually several hundred feet or more. Each T and GP element has a uniform diameter, wall thickness, and roughness (or friction) along the entire length. In typical use, the piping between stations is modeled with one or perhaps a few individual T or GP elements. Minor changes in diameter, wall thickness, etc. typically are not modeled. Minor supplies and deliveries are typically modeled at a nearby node instead of creating a node just for the purpose of modeling the minor supply or delivery.

Use a header (H) to model a pipe segment that is relatively short compared to the time scale of interest. Typically, headers are appropriate for modeling piping within stations while the GP and T are appropriate for modeling the piping between stations. However, if you are interested in a detailed simulation of a control system at a station, for example, and if the time scale is relatively fast, then you may want to model the various pipe segments at the station with transient pipe elements.

A heat exchanger (H or HE) is similar to a header but also includes the ability to cool and/or heat the fluid.

Header calculations are based on steady state equations. Pressure drop is a function of the flow rate and friction, and does not include time. For each pipe segment which has different pressure-drop characteristics, specify a separate header pipe. A node may have any number of header pipes connected to it.

SPS can also be used to model in detail the fast transients associated with, for example, surge control systems. In this situation, in which the timing of the surge and how the surge interacts with the control system is being studied, it is generally appropriate to model even relatively short lengths of station piping using transfer lines.

Typically, if Δt is the time interval corresponding to the fastest (shortest duration) event you wish to model, and λ is the wave propagation velocity in the pipe, and if $\lambda\Delta t$ exceeds the pipe length, then model the pipe as a header. Otherwise, model the pipe as a T or GP element.

For some typical fluids, the propagation velocity λ is approximately:

- 5000 ft/s (water)
- 2500 ft/s (many oils)
- 300-1100 ft/s (many gases)

The propagation velocity, λ, in feet/sec is computed by the equation:

$$\lambda = \frac{\left(\sqrt{144*32.2}\right)\sqrt{\frac{K}{\rho}}}{\sqrt{1+c\left(\frac{K}{Young}\right)\left(\frac{d}{Thick}\right)}}$$

where:

K	=	Bulk modulus of fluid (psi)
ρ	=	Density of fluid (lbm/ft^3)
d	=	Pipe diameter
THICK	=	Pipe wall thickness in the same units as d
YOUNG	=	Modulus of elasticity of the pipe wall (psi)
C	=	Pipe anchor constraint factor (dimensionless)

The pipe anchor constraint factor cannot be entered directly. Instead, adjust the modulus of elasticity.

Valves and regulators

Element type	Description
"Block valve (BV)" on page 306	• Gas only valve. • Stop and start flow. • May be modeled with a series header and/or a check valve. • Requires manual operation through an input command. • Uses a gas equation and gas valve coefficient.
"Block valve (B)" on page 308	• Stop and start flow. • May be modeled with a series header and/or a check valve. • Requires manual operation through an input command.
"Check valve (CV)" on page 311	• Gas only valve. • Avoid flow reversals. • May be modeled with a series header. • Uses a gas valve coefficient.
"Check valve (BC)" on page 313	• Avoid flow reversals. • May be modeled with a series header.
"Control valve (V)" on page 316	• Control or throttle flow at any given point. • May be modeled with a series header and/or a check valve. • Requires an actuator to position the valve stem according to signals from a controller or relay.
"Relief valve (RV)" on page 322	• Idealized relief valve. • Opens to keep the upstream pressure below a limit.
"Grove G887 relief valve (V G887)" on page 325	• Liquid only valve. • Models a Grove Flex-Flo Model 887 surge relief valve. • Requires a control system.
"General regulator (RG)" on page 329	• Gas only valve. • Acts like a control valve that does not require a user-defined control system. • Models a downstream pressure regulator, an upstream pressure regulator, or a flow regulator.
"Idealized regulator - control valve (RE)" on page 330	• Acts like a control valve that does not require a user-defined control system. • Models a downstream pressure regulator, an upstream pressure regulator, or a flow regulator.

Compressors

SPS has several compressors.

Compressor	Description
"Centrifugal compressor (CC)" on page 339	• Similar to KC but with SynerGEE style attribute names. • Can be exported by SynerGEE.
"General compressor (GC)" on page 343	• Similar to KP but with SynerGEE style attribute names. • Can be exported by SynerGEE.
"Idealized controllable centrifugal compressor (KC)" on page 346	• General-purpose centrifugal compressor. • Can use actuated or idealized control.
"Theoretical horsepower-flow compressor (KP)" on page 359	• General-purpose theoretical polytropic compressor. • Idealized control only.
Reciprocating compressor (KR)	Obsolete. Use "Reciprocating compressor (RC)" on page 371.
"Variable guide vane compressor (KV)" on page 364	• Similar to KC but guide vane angle can be controlled instead of speed. • Can use actuated or idealized control.
"Reciprocating compressor (RC)" on page 371	• General-purpose reciprocating compressor. • Can use actuated or idealized control. • SynerGEE style attribute names. • Can be exported by SynerGEE.

Fuel use data for each compressor is in "Compressor fuel" on page 337.

Pumps

SPS offers a pump element, for more information, see "Pump (P)" on page 382.

Control elements

SPS allows you to control process variables (pressure, flow, etc.) at specific locations in the model during the simulation. Simpler models may use built-in set points of various idealized elements; however, for high-fidelity detailed modeling of control systems, you will want to combine instruments such as controllers, actuators, sensors, and relays into a *control loop*.

Control loops are modeled as electrical circuits that carry voltage signals from one control element to the next. The control signal usually originates at a sensor, which senses a fluid property or a fluid state property somewhere in the model. The voltage signal usually passes from the sensor to a controller, which processes and transmits the signal in much the same way as a standard P-I-D controller works on real pipelines. SPS allows you to control the set point of a controller either by specifying a constant set point or by defining the signal from one element as the set point of another (as in a cascade control loop).

Several sensors and controllers may be installed in parallel, series, or both to monitor and process various sensed data. Any of several types of relays may be used to convert or select among the various signals before routing the voltage to an actuator, which actually controls such process equipment functions as pump/compressor speed, pump/compressor horsepower, stem position on a control valve, etc.

For example, to model the operation of a pump/compressor that automatically slows down at low suction or high discharge pressures, you could install:

- Two sensors to monitor the pressure at the suction and discharge ends
- Two controllers that receive and transmit the sensor signals to a relay
- A single low-select relay, which selects and transmits the lower of the two signals to an actuator
- An actuator, which controls the driver speed

Arbitrary time-varying signals can be generated using an input reference. This allows you to control instrumentation, such as manual ramping of driver speed on pump startup. The input reference can be programmed to provide an input value to a sensor at some pre-determined time or event during the simulation.

Spans

A span is a series of elements, including pipes, headers, block valves, check valves, control valves, and externals. Each element connects to only two elements, one on each end, except for an external connection. Spans are automatically generated and named. The name is the first element name, an underscore (_), and the last element name. Each span has a Show window.

In many systems, spans allow you to view summary information for the piping between stations with a single element.

For more information on disabling spans, see “SELECT (INPREP)” on page 211. For more information on span peek and poke attributes, see “Span (SP) attributes” on page 859.

Fluid Flow

This section describes:

- Flow direction and connections
- Acoustic velocity
- Node capacitance
- Column separation
- Slack line flow
- Batch and composition tracking

See “Supplies and deliveries” on page 137 to better understand methods of flow allocation.

Flow direction and connections

Most hydraulic elements have a from-end and a to-end (except externals, which have only one end, and tube/shell heat exchangers, which have four ends). Flow typically progresses from the from-end to the to-end, but most hydraulic elements can flow in either direction. You may define the element connections in terms of named nodes or in terms of the connection to an adjoining element. Either of these methods may be used or they may be combined.

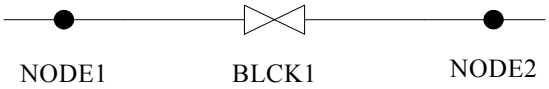
On output reports, flow response values are reported in positive and negative terms: positive values indicate flow from the from-end to the to-end; negative values indicate flow in reverse.

Control elements (controllers, relays, actuators, etc.) handle signals, not flows, so their connections are in terms of where the input signals originate.

Named nodes (the recommended way)

Each element, except externals and tube-shell heaters, has a from-node and a to-node. These nodes may be named and used to define the element connections.

For example, consider a block valve named BLCK1. This valve has a named from-node, NODE1, and a named to-node, NODE2.

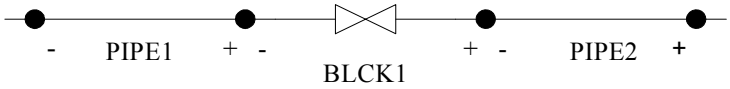


In this example, the valve BLCK1 has NODE1 as its from-node, and NODE2 as its to-node.

+/- Connections

Connections also may be made by referencing the from-end (-) and to-end (+) of an element. In this case, the nodes do not have a specific name but rather take the name of the connected element.

For example, consider the same block valve described in “Named nodes (the recommended way)” above, but this time the valve connects to a pipe segment at each end.



The valve BLCK1 is connected to the to-end of PIPE1 (entered as +PIPE1) and the-from end of PIPE2 (entered as -PIPE2).

Note: If using this method for externals, SPS considers the connection point to be the + side.

Acoustic velocity

Transients, under some conditions, move at the acoustic velocity of the fluid in the pipe environment. This velocity may be slower or faster than the velocity one finds in a handbook.

First, two acoustic velocities may be found in a handbook: adiabatic acoustic velocity and isothermal acoustic velocity. The real acoustic velocity in a pipeline is usually somewhere in between.

Acoustic velocity is defined (by pedants) as the eigenvalue of the differential operator describing the flow process. Specifically, this is related to the system of differential equations described in "Pipes" on page 141. This value depends upon the equation of state, which relates the density to pressure and temperature and the pipe expansibility, i.e., how the cross section varies with pressure.

In general terms, acoustic velocity is the rate a disturbance moves in a fluid at rest in the pipe. Acoustic velocity is determined by:

- fluid density (affects how much force is required to accelerate the fluid
- fluid compressibility (affects the impulse from a given force
- pipe expansibility

Generally, the acoustic velocity, c, is written in consistent units as:

$$c = \sqrt{K/\rho} \Big/ \sqrt{1 + \left(K/E\right)\left(d/t\right)}$$

where

K	=	bulk modulus, which is the derivative of the pressure with respect to density divided by the density
r	=	density
d	=	pipe diameter
t	=	wall thickness
E	=	Young's Modulus of the pipe material

If, instead of steel, the pipe was made of rubber, E is much smaller, and the acoustic velocity is lower. If the bulk modulus is high (such as heavy crudes), the acoustic velocity is higher. Also, if the bulk modulus is small, such as for gases, where it is numerically equal to approximately the pressure, the acoustic velocity is quite small. Also, the effective bulk modulus is dependent upon whether the fluid is kept at constant temperature, or whether the temperature is allowed to rise in response to the pressure changes caused by a disturbance. If the temperature is kept constant, the bulk modulus, and hence the acoustic velocity is lower.

The latter effect is significant, although the change in pressure in a fluid results in only a small temperature rise, the change in density caused by the increase in temperature is noticeable when compared with the change in density caused by the pressure wave. This is the reason that isothermal and adiabatic acoustic velocities differ, by as much as 15%. In the real world, the process of wave passage is somewhere between adiabatic and isothermal. When the fluid pressure increases and the corresponding change in temperature occurs, heat starts to interchange with the pipe surroundings. If the transient is slow, the process tends to be more nearly isothermal. If it is fast, it is closer to adiabatic.

The disturbances move at acoustic velocity only in an ideal sense. The acoustic velocity only represents the speed of the disturbance without any underlying flow. Any flow would significantly slow the effective velocity of a disturbance and smear out the wave fronts. The higher the friction drop, the more slowing and smearing.

SPS in the transient thermal mode solves the aforementioned equations taking into account the interrelations between all these factors.

Node capacitance

Nodes have a small capacitance (volume) associated with them to minimize solution problems when multiple station elements are connected in parallel. If the flows are not exactly the same through the connected station elements (headers, valves, etc.), the intervening nodes must have capacitance to absorb the mismatched flows until constant flow is achieved. Without a small capacitance, pressures would become unstable and either go to a very low number (out flow higher than in flow) or go to a large pressure (out flow less than in flow). This capacitance does not affect the overall fluid volume in the system because it is limited to a small value. This is helpful in cases where closed valves are in series or connected to flow controlled externals.

Column separation (liquid only)

Column separation occurs when pressure in a system drops below vapor pressure from, for example, pump failures or sudden valve closures. A vapor cavity forms in a small fraction of the pipe while the liquid columns on either side of the pipe accelerate. Sudden collapse of the bubble causes a spike in pressure.

To model column separation, use the COLSEP command in the INTRAN file. See "COLSEP" on page 441. Consider the following when modeling column separation:

- The compressibility of the vapor bubbles is taken into account in calculating the wave speed. In the column separation regime, the wave speed is often quite low because it is related to the bulk modulus of the bubble-containing liquid, which may be only a few hundred psi.
- An assumption is made that the fluid is free of entrained gases and that any vapor that forms will go back into solution. If there are entrained gases that dissolve slowly and/or incompletely, SPS calculates a larger surge due to collapse than is experienced in the field.
- The change in temperature from the collapse of the vapor cavity is not calculated. Therefore, depending upon the tuning parameters a different collapse may be calculated than really occurs in the field.

To accurately model column separation and collapse, tune the three COLSEP input data items, COLAPS, BAND and RATE. The purpose of the dead band is physically justifiable. The simulation assumes that a certain pressure reduction below the bubble point is necessary to initiate nucleation of the bubbles. However, column separation is performed isothermally even if transient thermal mode has been specified. See "TRANSTHERMAL" on page 249. As a bubble forms and begins to be filled with vapor and expands, heat is required to form the vapor and support the expansion. This means that the bubble point is lowered (because the temperature is lowered locally). The dead band and rate input parameters provide an attempt to approximate this thermal effect. The collapse of the bubble is similarly associated with thermal effects. As the bubble is compressed, heat is released. This must be conducted into the liquid, raising its temperature and hence its bubble point. Both of these thermal processes have the effect of delaying the phase transfer. The dead band and the collapse rate constants are attempts to approximate the thermal effects of collapse. The default constants are for water at 60°F.

Slack line flow

Slack line flow occurs when the hydraulic grade line and the elevation intersect, typically in mountainous regions. In slack line flow, the liquid flows along the bottom of the pipe while a static vapor cavity occupies the upper portion of the pipe. If the pressure increases above the vapor pressure of the fluid, the vapor cavity is absorbed back into the fluid. If the pressure increases rapidly, the collapse of the vapor cavity can be accompanied by a large spike in pressure.

To model slack line flow, use the COLSEP command in the INTRAN file. See "COLSEP" on page 441. The following are general comments on the simulations:

- The (infinite) compressibility of the vapor and the channel flow in the liquid cavity produce a wave speed that is quite low.
- The liquid is assumed to contain no entrained gases that will be difficult to redissolve. If there are such gases in the field fluids, when the vapor cavity collapses, the simulated surge may differ from that in the field.
- The same thermal effects are operative in the slack line case as in the column separation. The data entry parameters COLAPS, BAND and RATE play similar roles in controlling the formation and collapse of the vapor cavity. These may be tuned to match available data.
- Holdup will occur in the simulation because pipe elevation information may introduce changes in size of slope. The data parameters :HMUL and :HMIN allow fine tuning such effects as described in "Tuning holdup" below.

Tuning holdup

The physical concept of minimum holdup is the amount of fluid that the pipe will contain after draining. Holdup results when fluid is trapped in parts of the pipeline that have a slightly lower elevation.

In SPS, holdup is calculated and maintained by the user-defined elevation profile even if it is in much greater detail than the pipe knot spacing. Sufficient detail to determine the holdup in a slack system is difficult to obtain, but holdup in a pipe may be tuned. You may tune holdup with the Transfer line (T) attributes poke attributes:

- :HMUL is a multiplier to be applied to the holdup implied by elevation profile. If :HMUL is greater than one, holdup increases.
- :HMIN tuning is by pipe and is the minimum fraction holdup that is permitted in any interval. Care should be taken when adjusting :HMIN. If :HMIN is increased, fluid is instantaneously created in each interval that had a holdup fraction less than the new value. If :HMIN is decreased, any extra fluid that can flow starts to move.

The effective holdup in any interval is the maximum of :HMIN and the product of the actual holdup multiplied by :HMUL.

Batch and composition tracking

Any number of batches of any number of different fluids may be defined to coexist in one simulation, and these batches move in accordance with the flow rate in the equipment. They may blend where two different batches come together, the mixture resulting in a possible curved concentration profile in pipes fed by a point of blending.

Batches and compositions are tracked using *fluid mixture vectors* (FMVs). An FMV is the composition at a point; the composition between two FMVs is the linear interpolation of the composition at the FMVs. This means that SPS represents continuously changing fluid properties with a very small number of FMVs, provided the fluid properties change in a basically linear fashion.

The TRIVC (minimum change in concentration) and TRIVB (minimum batch size), set through the equation of state, determines whether a new FMV is created. TRIVC is used as a relative tolerance—if the concentration of a single component or fluid changes by a half percent out of linear, then a new FMV is created to preserve the non-linearity. In addition, two FMVs should not be closer than the TRIVB volume.

The number of FMVs in a system affects the time step. The time step is adjusted so that a FMV reaches the pipe end exactly at the end of a time step. Therefore, equipment at the pipe end uses the appropriate composition.

For liquid systems, each FMV does not necessarily represent a new batch. New batches are launched when the controlling fluid changes based on the mass fraction or the mixing weight parameters.

Also see "Setting composition at inflows" on page 139.

Displaying batches on a distance plot

You can track batches on a distance plot by choosing the BATCHTRACK option. By default, each fluid displays inside the batch bar, and vertical white bars appear between the fluids. If you want to display vertical white bars at the end of each pipe inside the batch bar, you can poke the global variable SHOW.PIPEENDS. For more information, see "GLOBALS (GB) attributes" on page 665 and "POKE" on page 493.

You have the option of entering a number in the BATCHTRACK field (in place of the Y/N). This number is the maximum height (in characters) of the part of the batch-tracking rectangle above the color bar. This area is referred to as the *vertical space* in which fluid names for the batches are plotted vertically along the batch bar until the specific batch is long enough to fit the fluid name to be plotted horizontally within the specific batch. This number input corresponds to the maximum number of characters of a fluid name that can be plotted vertically. Entering a number between 1 and 19 displays the batch bar. The minimum value is one. The maximum value is 19. If no number is entered and the batch bar is specified (with a Y), the default is 3.

Displaying batches on a Show window

You can view batch information on a pipe on a Show window. See "Batch tracking (BT) attributes" on page 644 for a list of keywords and descriptions. The batch peeks provide information a specific batch within a pipe. The integer batch in parentheses specifies which batch is desired.

- 1: The batch nearest (at) the to-end of the pipe
- -1: The batch nearest (at) the from-end of the pipe
- 2: The batch second nearest the to-end of the pipe

Batch boundaries occur at FMVs for SCL and between FMVs for SCLPROP, but not all FMVs are batch boundaries. The BATCHBOUNDARY attribute (see "GLOBALS (GB) attributes" on page 665) may be used to fine-tune batch displays. For the SCLPROP equation of state, batches may be subdivided into additional "political" batches (change in label but the same fluid) using ID1 and ID2:

LAB	Show boundaries where label changes.
ID1	Show boundaries where ID1 changes.
ID1LAB	Show boundaries where ID1 or label changes.
ID2	Show boundaries where ID2 changes.
ID2LAB	Show boundaries where ID2 or label changes.
ID1ID2	Show boundaries where ID1 or ID2 changes.
ID1ID2LAB	Show boundaries where ID1, ID2 or label changes.
FMV	Show boundaries where FMV changes.

The following is a discussion of the individual peeks.

:FLU(*)	Name of the dominant fluid component in a specified batch. The dominant fluid is an SCL or SCLPROP FLUID name, or a BWRS component.
:ID1(*)	Used for SCLPROP only. Batches have a pokable ID1, which has the form NNNNL, where each N is a digit, and L is a letter. It may be an integer with four or less digits. ID1 must be enclosed in quotation marks (" ").
:ID2(*)	Used for SCLPROP only. Batches have a pokable ID2, which has a maximum size of a four-digit integer.
:POS(*)	Location of the nearest edge of the specified batch in survey distance.
:BPP(*)	Location of the nearest edge of the specified batch. The distance is in PIPE.LENGTH units from the end of the pipe.
:ETA(*)	Predicted remaining time (in minutes) until the near end of the batch reaches a pipe end. The predicted time is based on the instantaneous fluid velocities. :ETA(1) is always 0, because the end of the batch nearest the to-end of the pipe is already at the pipe end. A large number (one month) is given if the current fluid velocities do not move the batch edge to a pipe end.

Additionally, each batch using SCLPROP can be displayed with a Show window. Each batch has an associated name. The name will be BATCHBOUNDARY-ID3 where ID3 is the batch number. Each label designator is separated by a hyphen (-) in developing the batch name.

When displaying a batch, the total volume of the batch is displayed along with average fluid properties for the batch. You can adjust fluid properties, batch IDs, and batch volume by adjusting head or tail location or volume. The following restrictions apply to altering batch properties:

- Changing the fluid name (LAB) does not change fluid properties.
- Changes to batch size may only increase the size. To decrease a batch size, increase an adjacent batch. Also, changes to batch size must occur in the pipe that the appropriate batch interface is in. If the head location is adjusted, sufficient volume must be in the pipe for the head position to move forward.
- Changes in head and tail locations and changes to fluid properties may be confusing if adjacent batches have significant diffusion between them. A change in volume or fluid property only affects the fluid between FMVs that have the same fluid batch and properties. Batches change between two adjacent FMVs and a change to volume and/or fluid property does not affect the interface fluid properties and volume. Therefore, the resulting change in fluid property or batch volume may not be as expected. The largest this additional volume can be is a full knot spacing of fluid.
- Changing a fluid name, therefore renaming the batch, does not remove the old batch from the model. The old batch has a volume of 0.
- Updating the batch information can be compute intensive. The variable BT.DT on the SHOW GLOBALS display can be used to set the time between updates of batch information. The default is every time step. In most cases, batch information need not be updated in SHAREDMEMORY this frequently; BT.DT could be set to update every two minutes or longer and still have sufficient information the batch.

Displaying batches in a text display

A character batch bar for pipes can be displayed on a text display by entering the :BAT attribute for a pipe inside the square brackets. For example: [PIPE1:BAT] displays the first letter of each batch fluid in PIPE1 within the length of the square brackets in the text display.

CHAPTER

7

Controlling the Simulation

Building and running a model is an iterative process. Once you have built the physical model and developed the base control logic in the INTRAN file, you will likely want to run the simulation and interact with the model. You may initialize the model with different starting conditions, change variables as the simulation progresses, and even submit a sequence of events. The SPS for Windows user interface offers menus for issuing some of the most common commands and display of data; however, you also may enter any interactive command from the command line. For more information on entering interactive commands from the command line, see "Interactive Commands" on page 545.

Writing control logic

When building an SPS model, you can use logic to control certain operations and to perform calculations. The logic used in SPS is comprised of expressions, operators, functions, user-defined variables, and commands. You may solve unique problems by applying the logic in creative ways.

Control logic is typically written in the INTRAN file, or batch mode, but may also be issued interactively or in some combination of batch and interactive. For example, you may want to define a sequence of events in the INTRAN file (see "DEFINE.SEQUENCE" on page 448). During the simulation you can activate the sequence of events (see "SUBMIT.SEQUENCE" on page 534). For more information on modes of running TRANS and syntax used in the INTRAN file, see "Overview of TRANS" on page 55. To reference a listing and description of expressions, operators, functions, and commands that may be used to construct logic, see:

- "Expressions" on page 587
- "Functions" on page 592
- "INTRAN file input" on page 426
- "Interactive Commands" on page 545
- "User-defined Variables, Macros, and Include Files" on page 567

Preparing to run a simulation

Before running a simulation, you need to have defined the physical properties for the model as well as any control logic or processing commands you need. In addition, before running the model, you need to decide how to *initialize* the model.

You may want to begin the simulation with specific starting conditions. For example, once you run a simulation, you can then save the simulation conditions and begin another simulation with those same starting conditions. In some cases, you may generate a report that shows the differences between one state and another. You may initialize a model in one of several ways.

Zero flow initialization

Zero flow initialization is the default type of initialization. It causes the simulation to begin with a hydrostatic equilibrium throughout the model (all the fluid and equipment at rest). This option is used to simulate pipeline system startup.

This is the option most often used when a system is modeled for the first time. The system is initialized with zero flow and hydrostatic pressure based on the elevation, fluid density, and the specified initial pressure. The initial pressure is specified at the highest elevation point. The specified initial pressure should be greater than the vapor pressure for liquid systems. Also, the initial pressure should not cause the static pressure at low points to exceed MAOP, alarm pressure set points, or shut down control pressure set points.

Once TRANS is started, external flows and pressures are ramped to the user-defined set points. Any pumps/compressors that were specified as STARTING are started. Also, the control system becomes active with actuator input signals initially at zero. Set up boundary conditions carefully so that during startup, the model does not exhibit extremely high or negative pressures, which may prevent a reasonable state from being obtained. Such a situation may occur if a flow-controlled external is connected to the piping system through a closed valve. For more information, see "Boundary conditions" on page 202.

Steady-state initialization (from SynerGEE or SPS)

If you have built a model in SynerGEE, you may wish to balance the model to achieve a steady state solution. After this steady state solution has been obtained in SynerGEE, you can save a steady state file (STS) that contains the steady state data from which you may begin a transient simulation in SPS. You may also save a steady state file from an SPS run. See "LOAD.STEADY" on page 476, "SHOW.STEADY" on page 527, and "SAVE.STEADY" on page 516 for more information.

See the SynerGEE documentation for more information on building and balancing a model in SynerGEE. See *SynerGEE Model Builder* for additional information, tips, and cautions.

Loading a steady state file

You can load steady state data. The data may be saved from SynerGEE or from a previous SPS run. You can specify settings to have SPS balance the model and generate reports of discrepancies in the OUTTRN file.

To load a steady state file

1 From the **Trans/Tport window**, select **Simulation > Steady State > Load Steady**.

2 In the **Load Steady** dialog box, click **Browse** and navigate to the steady state file you want to import. You may choose either a steady state (STS) file or a comma separated value (CSV) file.

Tip: You may select more than one file by pressing Ctrl and clicking on each file.

3 Choose options and set tolerances.

Echo Input	Echoes the content of the steady state file to the OUTTRN file.
Warn Extra	Warns when extra devices or peeks are in the steady state file.
Warn Missing	Warns when values are missing.
Warn Tolerance	Warns when data is not within the specified tolerances.
Balance	Has SPS balance the model from this steady state.
Relative Tolerance Pressure	Sets the maximum allowable percent difference in pressure between the pressure value SynerGEE supplied and the value SPS accepts.
Absolute Tolerance Pressure	Sets the maximum allowable pressure difference between the value SynerGEE supplied and the value SPS accepts.
Relative Tolerance Flow	Sets the maximum allowable percent difference in flow between the value SynerGEE supplied and the value SPS accepts.
Absolute Tolerance Flow	Sets the maximum allowable flow difference between the value SynerGEE supplied and the value SPS accepts.
Other Relative Tolerance	Sets the relative tolerance used for all values, but may be overridden for pressures and/or flows.
Other Absolute Tolerance	Sets the absolute tolerance used for all values, but may be overridden for pressures and/or flows. This value is in the user units for whatever value is being tested.
Log File Name	Type a name for a new log file or use the name of an existing log file. The file will be stored in the model directory.
Log File Append	Check to add data to an existing log file.

4 Click **OK** to load the steady state file.

Showing steady state data

You can generate a report that shows how closely the current model values are to the values last imported from a steady state file. You may want to generate this report before a balance to show how far the model is from the desired state, immediately after a balance to show how well the balance worked, or a few steps after a balance to show how well the model is holding the steady state.

To show steady state data

1 From the **Trans/Tport window**, select **Simulation > Steady State > Show Steady**.

2 In the **Show Steady** dialog box, click **Browse Device** to choose a device for which you want to show data.

Note: Only choose a device if you want to limit the report to a single device.

3 Click **Browse File** and navigate to the report file to which you want to overwrite or append steady state data.

4 Check **Append to File** to append the data. If you clear **Append to File**, the report file will be overwritten with the status from the current SPS simulation.

5 Under **Sort By**, choose a method by which the data will be sorted and shown.

- **Absolute** displays descending order by absolute value of difference.
- **Relative** displays descending order by absolute value of relative difference.
- **Device** displays ascending order by peek name.

6 Click **OK**.

Saving steady-state data

You can generate a steady state file using the model's current values. The model may or may not be in steady state.

To save a steady state file

1 From the **Trans/Tport window**, select **Simulation > Steady State > Save Steady**.

2 In the **Save Steady** dialog box, click **Browse**.

3 In the **Save As** dialog box, navigate to the location where you want to save the steady state file.

4 Select a file to which you want to overwrite or append data or type a new name for the file, and then click **Save**.

5 In the **Save Steady** dialog box, check **Append to File** to append the data. If you clear **Append to File**, the data will be overwritten with the status from the most current SPS simulation.

6 Click **OK**.

SPS saves the default information for each device in the model.

LOAD.STATUS initialization

LOAD.STATUS is used to initialize a simulation from a previously run simulation or to return an ongoing simulation to a former state, allowing for several what-if cases during a single TRANS session. The command changes the model state to the way it was when the simulation was archived and the time is reset to that given with BEGIN.

The data that is loaded may be from an archive file (ARK), created by the ARCHIVE command or from a status file (STA), created by the SAVE.STATUS command. Both file types contain a snapshot of the model parameters at the requested time, but the status file contains less information than the archive file and takes less time to write. The archive file, however, contains sufficient data to be used as a RESTRT file, if needed. To use the archive file as the Restart file, you can manually change the file type or you can define the archive file as the Restart file. For more information, see "Cases" on page 99. To do this from the command line, see "Running SPS programs from command line" on page 62.

The LOAD.STATUS command may be used either interactively or in INTRAN. The model being run does not need to have the same physical configuration as the previous run. Elements may be added or deleted between models. Typically, the simulation will steady out faster than when starting from a zero flow state. See "LOAD.STATUS" on page 471 in INTRAN for a description of loaded information that overrides information specified in the INPREP file.

Note: Make sure the commands in INTRAN do not inadvertently reset model conditions.

To archive the simulation

1 Select **Simulation > Archive**.

2 In the **Save As** dialog box, select an existing archive file to overwrite or type a new file name for the archive.

3 Click **Save**.

To load an archive

1 Select **Simulation > Load Status**.

A Load Status dialog box opens.

2 Type an archive file name (for example, save.ark) and go to step 5.

—or—

Click **Browse** to browse for an archive on your file system.

Another Load Status dialog box opens.

3 Select an archive from your file system.

4 Click **Open**.

The first **Load Status** dialog box returns.

5 Enter a time into the **Set Time** text box if you want the simulation to reset to another time; otherwise, the time will be set to the saved state time.

6 Click **OK**.

The archive loads into the simulation.

Running the simulation

When you run the simulation, you are running TRANS. After you initially start the simulation, you have several options for interactively running and stopping the simulation. You can run the simulation to the specified end time, run for a single time step, or restrict the time or conditions under which the simulation runs. You may also pause the model and then resume the simulation. If the simulation stops prematurely, you should review the OUTTRN file. For more information, see "RUN, RUN UNTIL, RUN WHILE, RUN FOR" on page 556.

To run the simulation

From any SPS window, select **Simulation > Run** (Ctrl+R).

The simulation begins taking time steps.

To run for one time step

From any SPS windows, select **Simulation > Step**.

The simulation runs for one time step.

To perform a restricted run

1 From any SPS window, select **Simulation > Run For**.

2 In the **Restricted Run** dialog box, choose **Run For**, **Run While**, or **Run Until**.

- If you chose **Run For**, choose either hours, minutes, seconds, or steps, then type a value in the **Count** box to specify how many hours, minutes, seconds, or steps to run.
- If you chose **Run While** or **Run Until**, click **...** to browse variable names.

3 In the **Browse Variable Names** dialog box, select a **Variable Name** as part of your conditional.

4 Click **OK**.

5 In the **Restricted Run** dialog box, select an operator for the conditional (e.g. =, >, <).

6 Type a number in the **Value** text box to complete the conditional. The time value expression may be the number of minutes desired, such as 240, or a date and time of day (99/02/01 16:30:00), depending on whether your model uses elapsed time or clock format. For more information, see "Elapsed time versus clock format" on page 199.

7 Click **OK** to execute the restricted run.

Once a Restricted Run condition is entered and accepted, the status displays in the upper-left portion of the Trans window.

To pause the simulation

From any SPS window, select **Simulation > Pause**.

The simulation stops taking time steps.

Entering interactive commands

In addition to commands you have already defined in the INTRAN file, you may also issue commands interactively during the simulation. Commands may be entered through:

- The command line
- Menus, buttons, and dialog boxes

This section describes how to issue some of the most commonly-entered commands. For a list of interactive commands available, see "Interactive Commands" on page 545.

Entering commands from the SPS command line

The SPS command line is where the interactive commands are entered and where SPS sends messages the simulation. For more information on using the command line in the TRANS/TPORT window, see "The command line in the TRANS/TPORT window" on page 87.

All interactive displays reserve the top line for the command line. The command line limit is 255 characters. Because the > prompt takes up two spaces, the input is limited to 253 characters. If the simulation is paused, then the paused prompt > is displayed in the upper-left corner. If the simulation is running, then the running message <Running> is shown.

Editing data and variables through SPS for Windows

Through the SPS for Windows interface, you can select menu items to edit any simulation variable, open or close valves, start or stop a compressor or pump unit, and submit a defined sequence of events. If you prefer, you can enter

commands to accomplish these tasks, such as POKE, OPEN, CLOSE, SUBMIT.SEQUENCE, etc., from the command line. For more information, see "Interactive Commands" on page 545.

To edit a variable

1 Select **Simulation > Edit Variable**.

2 In the **Edit Variable** dialog box, browse for the name of the variable that you want to edit.

3 Type a **New Value** for the variable.

Tip: You can type a time delay in minutes if you want the edit to occur later in the simulation.

Clear **Go to minimum time step** if you do not want the simulation to go to the minimum time step upon editing the variable.

4 Click **OK**.

Note: When you edit a variable through SPS, you are essentially entering a POKE command. In a POKE command, SPS sets the time step to the minimum value (DTMIN). A similar command is available that changes the variable without causing the time step to go to DTMIN if the change has no hydraulic effect. For more information, see "POKE" on page 493 and "SET" on page 518.

To edit variables from a Show

1 Double-click the variable to access the **Edit Variable** dialog box.

—or—

From the **Show** window, right-click on the data variable that you want to edit and select **Edit Variable** from the popup menu.

2 In the **Edit Variable** dialog box, type a **New Value** for the variable.

Tips: You can type a time delay in minutes if you want the edit to occur later in the simulation.

Clear **Go to minimum time step** if you do not want the simulation to go to the minimum time step upon editing the variable.

3 Click **OK**.

Note: When you edit a variable through SPS, you are essentially entering a POKE command. In a POKE command, SPS sets the time step to the minimum value (DTMIN). A similar command is available that changes the variable without causing the time step to go to DTMIN if the change has no hydraulic effect. For more information, see "POKE" on page 493 and "SET" on page 518.

To edit variables from a Report

1 Double-click the variable to access the **Edit Variable** dialog box.

—or—

From the **Report**, right-click the data variable that you want to edit and select **Edit Variable** from the popup menu.

2 In the **Edit Variable** dialog box, type a **New Value** for the variable.

Tips: You can type a time delay in minutes if you want the edit to occur later in the simulation.

Clear **Go to minimum time step** if you do not want the simulation to go to the minimum time step upon editing the variable.

3 Click **OK**.

Note: When you edit a variable through SPS, you are essentially entering a POKE command. In a POKE command, SPS sets the time step to the minimum value (DTMIN). A similar command is available that changes the variable without causing the time step to go to DTMIN if the change has no hydraulic effect. For more information, see "POKE" on page 493 and "SET" on page 518.

To open and close block valves from a Trans/Tport window

1 From the Trans/Tport window, select **Simulation > Open/Close/Stop Valve**.

2 In the **Valve Control** dialog box, click the block valve that you want to open, close, or stop.

3 Click **Open Valve**, **Close Valve**, or **Stop Valve**, depending on the action you want to perform.

4 Click **OK** to close the dialog box.

To open and close block valves from a Show window

From the Show window for a block valve, right-click and select **Open**, **Close**, or **Stop**, depending on the action you want to perform.

To start/stop Compressor and Pump units from a Trans/Tport window

1 Select **Simulation > Start/Stop Unit**.

2 In the **Unit Control** dialog box, select the name of the Pump or Compressor that you want to start or stop.

3 Click **Start Unit** or **Stop Unit**, depending on the action you want to perform.

4 Click **OK** to close the dialog box.

To start/stop Compressor and Pump units from a Show window

From the Show window for a Compressor or Pump, right-click and select **Start** or **Stop**, depending on the action you want to perform.

To submit a sequence

1 Select **Simulation > Submit Sequence**.

2 In the **Submit Sequence** dialog box, select a sequence name.

3 Under **Arguments**, type the required input(s) in the appropriate text boxes if needed.

4 Click **OK**.

Define sequences in the INTRAN file using the DEFINE.SEQUENCE command.

Ending a simulation

You specify an end time for the simulation using the BEGIN command. If you want to end a simulation before it times out, you may manually end the simulation. For more information, see "HALT/QUIT" on page 550. If you choose to restart the simulation, you should run PREPR before running TRANS again.

To quit a simulation

Select **File > Exit**.

—or—

Select **Simulation > Halt**.

—or—

Click the close symbol in the upper-left corner of the Trans/Tport window.

Note: If you are running both a Trans and a Tport session, selecting **Simulation > Halt** in the Tport session will exit the Trans session, leaving the Tport session running.

To exit SPS

In the **SPS startup window**, click **Exit**.

CHAPTER

8

Displaying and Printing Data

SPS provides several ways to display and print data, including:

- "Compressor/pump map plots" on page 166
- "Distance plots" on page 169
- "Time plots" on page 179
- "Reports" on page 182
- "Text displays" on page 186
- "Show windows" on page 188

Any of the these displays may be printed. For more information, see "Printing" on page 191.

In addition, you may manipulate and output data for display in other software products through . For more information, see "Overview of SimPlot" on page 191.

This section describes procedures common to all display types, the different display types, and how to set up reports, plots, text displays, and Show windows.

Working with displays

Displays include reports, plots, and custom text displays. These displays share many common procedures for creating, opening, viewing, and saving, and are stored in the same file format.

Reports and plots are created interactively while custom text displays must be created through a text editor and then saved in a text file. Plots and reports as well as text displays should be saved with the DSP file extension. See "display (DSP) file" on page 1054. For more information on creating custom text displays, see "Text displays" on page 186.

During a simulation, you may create and work with new plots or reports, or you may open a previously saved display. SPS keeps working versions of reports and plots in memory. Once you open a saved display, you may interactively edit it, view different displays, and then return to the plot or report you modified, and SPS will remember the changes you made. However, if you leave the simulation without saving, your changes to the plot or report will be lost. Therefore, you need to know whether you are opening a display from a file or from memory. For more information on setting a default directory for opening and saving display files, see "Display directory (DREMPATH_DSP)" on page 94.

This section describes common tasks for creating, viewing, and saving displays. For more information on each individual display type, see "Distance plots" on page 169, "Time plots" on page 179, "Reports" on page 182, and "Text displays" on page 186.

To open a previously saved display in the Trans/Tport window

Note: You can select text displays, reports, distance plots, and time plots by this method.

1. Select **File > Open Display** or click the Open toolbar button.
2. In the Open dialog box, select a display from the list and click **Open**.

   The text display, report, or plot displays in the Trans/Tport window.

To open a previously saved display from the command line

Note: In order to open a previously saved display, plot, or report from the command line, you need to know the name of the display. If the file is not in the model directory, the directory must be defined in the DREMPATH_DSP environment variable. For more information, see "Display directory (DREMPATH_DSP)" on page 94.

Enter the REREAD command followed by the name of the display, plot, or report that you want to view.

For example, to display a plot named FIGP.DSP, enter:

```
REREAD FIGP
```

To open a display that is currently in memory from the command line

Interactively enter the name of the plot or report.

For example, to display a plot named FIGP.DSP, enter:

```
FIGP
```

To automatically open a display when you run TRANS

When an interactive run begins, TRANS looks for a display called INIT.DSP. If TRANS finds it, then it displays initially when the model begins running. Instead of having a display called INIT, you can define an existing display, report, or figure as the INIT display as in the following.

```
MACRO(INIT, OVERVIEW)
```

TRANS displays OVERVIEW.DSP as the initial display instead of INIT.DSP.

To launch a Show window, use this MACRO syntax:

```
MACRO(INIT, SHOW devicename)
```

For more information, see "MACRO" on page 580.

To save a plot or report from the command line

From the display where the plot or report is displayed, interactively enter:

```
save
```

The plot data is saved to a DSP file that uses the name you specified in the menu. The file is stored in the model directory, or whichever directory you have specified in the DREMPATH_DSP environment variable. For more information on setting this variable, see "Display directory (DREMPATH_DSP)" on page 94.

To edit the current plot or report from the command line

From the current plot or report, interactively enter:

```
flip
```

To edit axes settings from the command line

You can interactively edit axes settings for time plots or distance plots without going into the plot editormenu. Interactively enter XMIN, XMAX, YMIN, YMAX, ZMIN, or ZMAX followed by a number to change the specified axis minimum/maximum to the value input. Interactively enter XINTS, YINTS, or ZINTS followed by a number to change the number of intervals for the specified axis.

For example, to change the maximum value of the Y axis to 500, interactively enter the following while the plot is displayed:

```
YMAX 500
```

If you do not see the change immediately, you may need to refresh the display. See “To refresh the display” on page 165.

For more information on editing axes settings through the editor, see “To edit the current distance plot” on page 171 or “To edit the current time plot” on page 182.

To refresh the display

From the Trans/Tport window, select **Edit > Refresh**.

—or—

Click the Refresh toolbar button  .

Plot overrides

You can show and/or edit time plot and distance plot overrides for axis settings from a Show window. The overrides are useful to view a series of plots over the same time scale.

For example, if the simulation has progressed to the next day, and you want to concentrate on noon to 1:00 pm of the previous day, you would set the overrides accordingly. Then, as long as the overrides are active, any plot you bring up would display noon to 1:00 pm. These settings override individual plot settings applied in the plot editor menu.

To change plot override settings, interactively enter `show tplot` or `show dplot`.

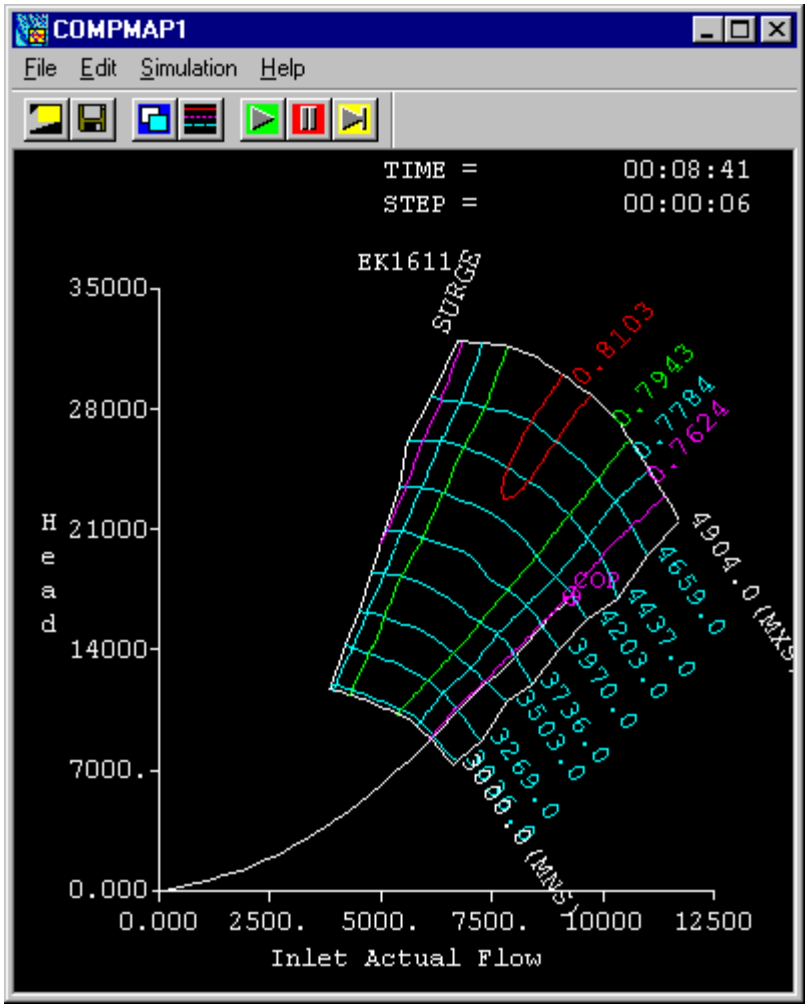
On Windows you can select **Window > Show**. In the Browse Device Names dialog box, select **plot limits** and **time plot** or **distance plot**.

You can edit pokable attributes from the Show window. For more information, see “To edit variables from a Show” on page 159 or “Show windows” on page 188.

Compressor/pump map plots

A compressor/pump map plot graphically displays the performance information specified in the INPREP file for a compressor or pump. The map plot also displays the current operating point of the selected unit and the operating point's history. Using the buttons, the map plot window toolbar can run or pause the simulation, save the current map plot, display the mapped unit in a Show window, and edit the current plot's attributes.

The compressor/pump map plot plots the performance maps of centrifugal compressors (KC and CC), variable guide vane compressors (KV), and pumps (P).



To create a compressor/pump map plot

1 From the Trans/Tport window, select **Window > Compressor/Pump Map Plot** or click on the Compressor/Pump Map Plot toolbar button .

2 In the Compressor/Pump Map Plot dialog box, click **New**.

3 In the Compressor/Pump Map Information tab of the Compressor/Pump Map Editor, select a compressor or pump to plot.

4 Click the **Figure Attributes** tab.

5 Use the Figure Attributes tab of the Compressor/Pump Map Editor to specify the following:

Name	Name of the compressor/pump map plot. The maximum length is 80 characters.	REQUIRED

Title	Title that will display on the compressor/pump map plot.
	• The title may consist of up to 3 lines, with a combined total of 250 characters.
	• Enter `\n` to specify line breaks in the title.
	• Enter `\\` to specify a single backslash character.
	For example, if you type the following in the Distance Plot Title text box:
	The title would appear as follows:
	`C:\model\new`
	`Second Title Line`
	`Third Title Line`
		Optional

6 Click **OK** to view the plot in the Compressor/Pump Map Plot window.

To open a compressor/pump map plot

1 From the Trans/Tport window, select **Window > Compressor/Pump Map Plot** or click on the Compressor/Pump Map Plot toolbar button  .

2 In the Compressor/Pump Map Plot dialog box, select a plot from the **Existing Compressor/Pump Map Plots** list, then click **OK**.

The selected compressor/pump map displays in the Compressor/Pump Map Plot window.

3 Select **File > Open Display** to browse to a display file to view or click **Open**.

To edit a compressor/pump map plot

1 Open the map plot that you want to edit. For more information on opening a compressor/pump map plot, see "To open a compressor/pump map plot" on page 167.

2 From the Compressor/Pump Map Plot window, select **Edit > Edit Figure** or click the Edit Figure toolbar button  .

3 Use the Compressor/Pump Map Editor to edit the plot settings, as necessary. For more information on the available settings, see "To create a compressor/pump map plot" on page 166.

4 Click **OK** to close the Compressor/Pump Map Editor.

To change the history duration for a compressor/pump map plot

1 From the Compressor/Pump Map Plot window, select **Edit > Edit Figure** or click the Edit Figure toolbar button  .

2 In the Compressor/Pump Map Editor, click the **Compressor/Pump Information** tab.

3 In the Compressor/Pump Information tab, click **More Parameter Settings**.

4 In the Compressor/Pump Map Settings dialog box, type a history duration.

5 Click **OK** to close the Compressor/Pump Map Settings dialog box.

6 Click **OK** to close the Compressor/Pump Map Editor.

To add or remove curves in a compressor/pump map plot

1 From the Compressor/Pump Map Plot window, select **Edit > Edit Figure** or click the Edit Figure toolbar button.

2 In the Compressor/Pump Map Editor, click the **Compressor/Pump Information** tab.

3 In the Compressor/Pump Information tab, click **Add or Delete Curves**.

4 Perform the following steps to add or remove curves.

To:	Do This:
Add a curve	a In the Compressor/Pump Map Curves dialog box, specify the curve type by selecting either **Constant Speed Curve** or **Constant Efficiency Curve**. b n the **Value** text box, type a value for the curve. c Click **Add**. The curve appears in the Plot Item list. d Click **OK** to close the Compressor/Pump Map Curves dialog box. **Note:** For information on editing the curve line color, style, and label, see "To edit curve line styles, colors, and labels in a compressor/pump map plot" on page 168.
Remove a curve	a In the Map Curves dialog box, select a curve under **Plot Items**. b Click **Delete**. The curve is removed from the Plot Item list. c Click **OK** to close the Compressor/Pump Map Curves dialog box.

5 Click **OK** to close the Compressor/Pump Map Editor.

To edit curve line styles, colors, and labels in a compressor/pump map plot

1 From the Compressor/Pump Map Plot window, select **Edit > Edit Curves**.

2 In the Curve Attributes dialog box, under Preview, select the curve that you want to edit.

3 Under **Color**, select a line color for the curve.

4 Under **Type**, select a line style for the curve.

5 Under **Label**, type the label that you want to be displayed with the curve.

6 Click **OK** to close the Curve Attributes dialog box.

To edit the axis settings for a compressor/pump map plot

1 From the Compressor/Pump Map Plot window, select **Edit > Edit Figure** or click the Edit Figure toolbar button.

2 In the Compressor/Pump Map Editor, click the **Figure Attributes** tab.

3 In the Figure Attributes tab, type a new title, axes labels, minimums, maximums, and intervals as needed.

4 Click **OK**.

To display input points on a compressor/pump map plot

Note: Input points are data values defined in the INPREP file and are "shown" in the plot with circles drawn around them.

From the Compressor/Pump Map Plot window, select **Edit > Show Input Points**.

To save a compressor/pump map plot

1 Click on the **Compressor/Pump Map Plot** that you want to save.

2 Click **Save** in the Compressor/Pump Map Plot window toolbar to save the plot under the default name.

—or—

From the Compressor/Pump Map Plot window, select **File > Save As**. In the Save As dialog box, enter a name and path for the plot to be saved. Click **Save**.

Distance plots

A distance plot graphically displays system information, such as pressure or standard flow, as a function of distance. Distance plots may have either two or three axes, depending on the specified variables and format. In addition to the horizontal axis, you may define up to two vertical axes (one on either side of the distance plot).

- The horizontal (X) axis is the dependent variable and represents distance.
- The left vertical axis is the Y axis, and the right vertical axis is the Z axis. The use of both Y and Z does not imply any three-dimensional representation, but rather is used to name the two vertical axes. Dissimilar variables (like pressure and flow) may be plotted on the same vertical axis, but it is recommended to plot each on its own axis so that independent scaling may be used.

You have control over the number of historical curves shown on a distance plot. You may poke the GLOBALS attribute DISTPLOT.HIST to set the number of curves. For more information, see "GLOBALS (GB) attributes" on page 665. You may also clear the distance plot with the CLEARHIST command.

If there are multiple paths between the chosen from-element and to-element, TRANS/TPORT finds and uses the shortest path from the end of the from-element name to the beginning of the to-element name. Alternatively, you can use a "define path" as the path between the two nodes. Define paths, as described in "DEFINE.PATH" on page 571, are specific paths that are defined through the model using the names of pipes, headers, externals, valves, pumps, nodes, and/or defined path names.

You can display only one distance plot at a time in a TRANS/TPORT window. However, you can have an unlimited number of TPORT windows open, each with its own distance plot.

You can interactively override plot settings. You can also view a listing of the plot overrides. For more information, see "Plot overrides" on page 165 and "SHOW" on page 559.

For more information on common tasks associated with displays, see "Working with displays" on page 163.

To create a new distance plot from the command line

Interactively enter the DISTPLOT command.

```
DISTPLOT
```

To create a distance plot using the Trans/Tport window

Note: The minimum amount of information needed to create a distance plot is the distance plot name, at least one variable, and the from and to elements.

1 From the Trans/Tport window, select **Window > Distance Plot** or click the Distance Plot toolbar button  .

2 In the Distance Plot dialog box, click **New**.

3 Use the Distance Plot Editor to specify the following:

Name	Name of the distance plot. The maximum length is 80 characters.	REQUIRED

Title	Title that will display on the distance plot.
	• The title may consist of up to 3 lines, with a combined total of 250 characters.
	• Enter `\n` to specify line breaks in the title.
	• Enter `\\` to specify a single backslash character.
	For example, if you type the following in the Distance Plot Title text box:
	The title would appear as follows:
	`C:\model\new`
	`Second Title Line`
	`Third Title Line`
		Optional

From and **To**	From and To devices that bound the distance plot. To specify the From and To devices: • Type the device name in the appropriate text box. —or— • Click  and then browse to the location. You can also type the name of a DEFINE.PATH in both the From and To box. For more information on define paths, see "DEFINE.PATH" on page 571.	REQUIRED
DT	Specifies how often the plot updates its curves.	Optional
Stations	Specifies whether each station's name plots on the distance plot.	Optional
Batch Track	For liquid systems, use the Batch Track field to plot the batch bars on the distance plot. A number value in this field corresponds to the minimum number of characters used to describe the batch.	Optional

History	Specify the number of history curves plotted. Setting this field to 0 plots all of the curve's history.	Optional
Plot Items	Perform the following steps to add plot items to your distance plot: a In the Plot Items area, click **Browse**. b In the Distance Plot Items dialog box, select a plot item. (See "Distance plot items" on page 172 for descriptions of the available distance plot items.) c Click **OK**. d Back in the Distance Plot Editor, select the **Axis** on which to plot the item and the **Line** color and **Line Type** for the plotted curve. **Tip:** A line type of 0 corresponds to a solid line, while line types 1 through 4 correspond to different dashed line formats. When you add the item to the Plot Items list (as described in the next step), the selected line type and color is shown in the list. e Click **Add** to add the item to the Plot Items list.	REQUIRED
Axis Settings	Label, minimum value, maximum value, and interval for each axis.	Optional

4 Click **Save** or **Save As** to save changes to the display file, or click **OK** if you want to view the changes but not save them permanently.

Tip: From the Trans window, click the Refresh toolbar button to clear the distance plot history and refresh the Distance Plot display.

To open a distance plot

1 From the Trans/Tport window, select **Window > Distance Plot** or click the Distance Plot toolbar button  .

2 In the Distance Plot dialog box, under **Existing Plots**, select the name of the distance plot that you want to open.

3 Click **OK**.

The distance plot displays in the Trans/Tport window.

4 Select **File > Open Display** to browse your file system for a display file to view or click **Open**.

Tip: From the Trans window, click the Refresh toolbar button to clear the distance plot history and refresh the Distance Plot display.

To edit the current distance plot

1 With the distance plot that you want to edit displayed in the Trans/Tport window, select **Window > Flip** or click the Flip toolbar button  .

2 Use the Distance Plot Editor to perform any of the following tasks:

- To *edit* an existing plot item, select the desired item and then change the options for Axis, Line, and Line Type. Click **Change** to commit the changes.
- To *remove* a plot item, select the desired item and click **Remove**.
- To *add* a plot item, click **Browse** to select the plot item that you want to add, and then set the options for Axis, Line, and Line Type. Click Add to add the new plot item to the plot item list.

For detailed information the options in the Distance Plot Editor, see "To create a distance plot using the Trans/Tport window" on page 170.

3 Click **Save** to save changes to the display file, and then click **OK** to view the changes.

—or—

Click **OK** if you want to view the changes but not save them permanently.

Tip: From the Trans window, click the Refresh toolbar button to clear the distance plot history and refresh the Distance Plot display.

Distance plot items

The following table lists common distance plot items. In addition to these items, any item that is peekable on a pipe can be plotted. In this case, the corresponding plot is a constant along the pipe because peeks are not a function of distance. This capability is useful for creating distance plots of items such as OD, WT, and RUF.

Note: Distance plots default to displaying PIPE.DIST on the X axis, regardless of whether PIPE.DIST or HORIZ.DIST was specified on the pipe (T or GP). To display HORIZ.DIST on the X axis, choose HORIZ.DIST as one of the distance plot items.

Distance plot item	Description
API.GRAVITY	API gravity of the fluid at 0 psig and 60°F.
BASE.DENSITY	Computed Density at Base Pressure and Temperature (SCLPROP).
BASE.VISCOSITY	Computed Viscosity at Base Pressure and Temperature (SCLPROP).
BASE.BULK.MODULUS	Bulk Modulus at Base Pressure and Temperature (SCLPROP).
BULK.MODULUS	Bulk Modulus of the fluid (SCLPROP).
COMPONENTS	Individual components or user-defined property (BWRS, SCL and AGA).
DRA	Amount of DRA in COMPOSITION.DRA units. (See "Units for composition" on page 130.) Requires fluid entries in STATE SCL. For more information, see "STATE SCL" on page 230.
DRAG.REDUCTION	Drag reduction agent effectiveness (0-1). Requires fluid entries in STATE SCL. For more information, see "STATE SCL" on page 230.
DENSITY	Calculated density.
ELEVATION	User-defined elevation.
ELEVATION.SLOPE	Slope of the elevation changes.
FRIC.FACTOR	Friction factor.
FRIC.DERIV	Derivative of friction factor.

Distance plot item	Description
FRIC.PRES.DROP	Frictional pressure drop.
GRASHOF	Grashof number.
GRAV.PRES.DROP	Gravitational pressure drop.
GROUND.TEMP	Ground temperature.
HEAD	Calculated total head for liquid systems.
HEAT.CAPA	Heat capacity.
HEAT.COND	Heat conductivity.
HEAT.FLUX	Heat flux for the laminar film layer only.
HEAT.TRANS.COEF	Heat transfer coefficient for laminar film layer only.
HIGH.HEAT.VALUE	Calculated gas high heating value (BWRS).
HORIZONTAL.DIST	Calculates the survey distance as a function of pipe distance. Used to establish mileposts. (Plot on the X axis.)
KNOT.NUMBER	Knot number.
LAOP	Lowest allowable operating pressure.
LASP	Lowest allowable surge pressure
LIQ.FRAC	Liquid fraction (from 0 to 1). Liquid fraction < 1 indicates slack line flow is occurring.
LOW.HEAT.VALUE	Calculated gas low heating value (BWRS).
MAOH	Maximum allowable operating head. Calculated based on user-defined MAOP.
MAOP	Maximum allowable operating pressure.
MASH	Maximum allowable surge head.
MASP	Maximum allowable surge pressure.
MASS.FLOW	Calculated mass flow rate.
MAX.PRESSURE	Maximum pressure observed since the last PROFILE time.
MIN.PRESSURE	Minimum pressure observed since the last PROFILE time.
NOMINAL.VOLUME	Nominal pipe volume.
NUSSELT	Nusselt number.
PIPE.FLOW	Calculated flow rate at pipeline conditions (calculated using flowing pressure and temperature).
PRANDTL	Prandtl number.
PRESSURE	Calculated pressure.
REYNOLDS	Reynolds number calculated.
S.VOL	Derivative of volume with respect to distance. (standard volume/foot).
SONIC.VELOCITY	Sonic velocity.

Distance plot item	Description
SPEC.GRAV	Calculated gas specific gravity (BWRS).
STANDARD.FLOW	Calculated flow rate at standard conditions. Custody transfer conditions are given on the CUSTODY entry in the INPREP file.
TEMPERATURE	Temperature. This is user-defined or calculated depending on the thermal mode used.
TEMP.MODULUS	Temperature modulus.
THERMAL.FLOW	Thermal flow rate.
VAPOR.PRESSURE	Vapor pressure of the liquid.
VELOCITY	Calculated velocity.
VISCOSITY	Viscosity of the fluid.
VP.HEAD	Head based on vapor pressure (liquid).
WALL.TEMP	Temperature of the pipe wall.
WAX.BUILDUP	Wax buildup along pipe.
WAX.RATE	Wax deposition rate.
WAX.TEMP	Wax temperature.
WOBBE.INDEX	Calculated Wobbe index (BWRS).

Time plots

A time plot graphically displays device information (i.e., P+, P-, Q+, etc.) as a function of time. Time plots may have either two or three axes depending on the variables and format that are specified. In addition to the horizontal axis, you may define one or two vertical axes, one on either side of the plot.

- The horizontal (X) axis is the dependent variable and represents time.
- The left vertical axis is the Y axis, and the right vertical axis is the Z axis. The use of both Y and Z does not imply any three-dimensional representation, but rather is used to name the two vertical axes. Dissimilar variables (like pressure and flow) may be plotted on the same vertical axis, but it is recommended to plot each on its own axis so that independent scaling may be used.

You can display only one time plot at a time in a TRANS/TPORT window. However, you can have any number of TPORT sessions running, each with its own time plot.

Note: Time plots can handle a maximum of 35,000 points for a single pen on a plot, and 100,000 points total for a single plot wth multiple lines. The plot freezes if the count is exceeded. This issue can be avoided if a sliding time window is used in the plot set.

Only those variables included in the TRENDLIST can be plotted in a time plot. For more information, see “TRENDLIST” on page 537.

You can interactively override plot settings. You can also view a listing of the plot overrides. For more information, see “Plot overrides” on page 165 and “SHOW” on page 559.

For more information on common tasks associated with displays, see “Working with displays” on page 163.

To create a time plot from the Trans/Tport window

1 From the Trans/Tport window, select **Window > Time Plot** or click the Time Plot toolbar button  .

2 In the Time Plot dialog box, click **New**.

3 Use the Time Plot Editor to specify the following:

Name	Name of the time plot. The maximum length is 80 characters.	REQUIRED

Title	Title that will display on the time plot.
	• The title may consist of up to 3 lines, with a combined total of 250 characters.
	• Enter `\n` to specify line breaks in the title.
	• Enter `\\` to specify a single backslash character.
	For example, if you type the following in the Time Plot Title text box:
	The title would appear as follows:
	`C:\model\new`
	`Second Title Line`
	`Third Title Line`
		Optional

Time format	Specifies the time format for the plot. Available choices are **Clock**, **Day**, **Hour**, **Minute**, or **Second**.	REQUIRED

Plot Items	Perform the following steps to add plot items to your time plot: a In the Plot Items area, click **Browse**. **Note:** A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. For more information, see "TRENDLIST" on page 537. b In the Browse Variables dialog box, select from the available **Device Types**, **Device Name**, and **Attributes**. c Click **OK**. d Back in the Time Plot Editor, select the **Axis** on which to plot the item and the **Line** color and **Line Type** for the plotted curve. **Tip:** A line type of 0 corresponds to a solid line, while line types 1 through 4 correspond to different dashed line formats. When you add the item to the Plot Items list (as described in the next step), the selected line type and color is shown in the list. e Click **Add** to add the item to the Plot Items list.	REQUIRED
Axis Settings	Label, minimum value, maximum value, and interval for each axis. **Note:** For information on using the axis settings to show sliding trends, see "Sliding trends" on page 182.	Optional

4 Click **Save** or **Save As** to save changes to the display file, or click **OK** if you want to view the changes but not save them permanently.

Tip: From the Trans window, click the Refresh toolbar button to clear the time plot history and refresh the time plot display.

To create a time plot from a Report

1 Right-click on the data variable in the Report that you want to plot.

2 Select **Time Plot** from the popup menu.

SPS opens a time plot of the selected variable in the Trans/Tport window.

Note: A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. For more information, see "TRENDLIST" on page 537.

To create a time plot from a Show

1 Right-click on the data variable (preceded by a box or circle with a curve inside or) in the Show that you want to plot.

2 Select **Time Plot** from the popup menu.

SPS opens a time plot of the selected variable in the Trans/Tport window.

Note: A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. See "TRENDLIST" on page 537.

To create a time plot from the Edit Variable dialog box

1. From the Edit Variable dialog box, click **Browse** to select a variable to view in a time plot (if necessary).
2. In the Browse Variable Names dialog box, select a variable name.
3. Click **OK**.
4. In the Edit Variable dialog box, click **Plot**.

   SPS opens a time plot of the selected variable in the Trans/Tport window.

   Note: A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. For more information, see "TRENDLIST" on page 537.

To open a time plot

1. From the Trans/Tport window, select **Window > Time Plot** or click the Time Plot toolbar button  .
2. In the Time Plot dialog box, select a time plot from the **Existing Time Plot** list, then click **OK**.

   The time plot is displayed in the Trans/Tport window.

   Note: A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. See "TRENDLIST" on page 537.
3. Select **File > Open Display** to browse your file system for a display file to view or click **Open**.

To edit the current time plot

1. With the time plot that you want to edit displayed in the Trans/Tport window, select **Window > Flip** or click the Flip toolbar button  .
2. Use the Time Plot Editor to perform any of the following tasks:
   - To *edit* an existing plot item, select the desired item and then change the options for Axis, Line, and Line Type. Click **Change** to commit the changes.

     Note: A variable must first be specified in the TRENDLIST in the INTRAN file to be plotted in a time plot. For more information, see "TRENDLIST" on page 537.
   - To *remove* a plot item, select the desired item and click **Remove**.
   - To *add* a plot item, click **Browse** to select the plot item that you want to add, and then set the options for Axis, Line, and Line Type. Click Add to add the new plot item to the plot item list.

   For detailed information the options in the Time Plot Editor, see "To create a time plot from the Trans/Tport window" on page 179.
3. Click **Save** to save changes to the display file, and then click **OK** to view the changes.

   —or—

   Click **OK** if you want to view the changes but not save them permanently.

Sliding trends

A sliding trend allows you to view data over a specific range of time as the simulation progresses. Instead of specifying a concrete axis minimum for a time plot, you specify a negative value that represents the time interval relative to the current simulation time. As time increases, this trend window "slides" to show a different time frame. In this way, you could, for example, always view the last 2 hours of data, as the simulation progresses.

You can specify this time frame by entering a negative axis minimum, either interactively from the time plot or when editing the time plot. For more information on editing time plots, see "To edit axes settings from the command line" on page 165 or "To edit the current time plot" on page 182.

Notes: When you enter the axis minimum, be sure to use the same time format and units as you specified in the TIME field.

For example, if you want to see the last 2 hours from the current simulation time:

- If the time units are in hours, enter -2 for the x-axis minimum
- If the time units are in minutes, enter -120 for the x-axis minimum
- If the units are in clock format, enter -02:00 for the x-axis minimum

Reports

A Report displays dynamic, user-requested information in tabular form. Once a Report is defined, you can display it in a window at any time during the simulation or save it for later use. For more information on common tasks associated with displays, see "Working with displays" on page 163.

The Report window displays user-requested information in tabular form. From the Report window you can control the simulation (including pausing, running, and stepping through the simulation), edit the Report items, and view other displays. You can also edit the Report contents and determine when to refresh the Report display.

Reports feature the ability to edit variables and request time plots of variables for a single Report item. To do this, right-click and select Edit Variable or Time Plot from the popup menu. Selecting Edit Variable opens a dialog box for changing a variable's value. Selecting Time Plot opens a time plot of the variable into the Trans/Tport window.

The Report display area also features the ability to open a Show window on a device. (See "To open a Show window" on page 189.)

The column bar displays the label for each column. You can resize each column by dragging the column bar divider to a new location. This will automatically update the column width(s) in the Report editor. You can save changes to the Report by clicking Edit Report, then saving the report. The current window position and the sort order of the report will also be saved.

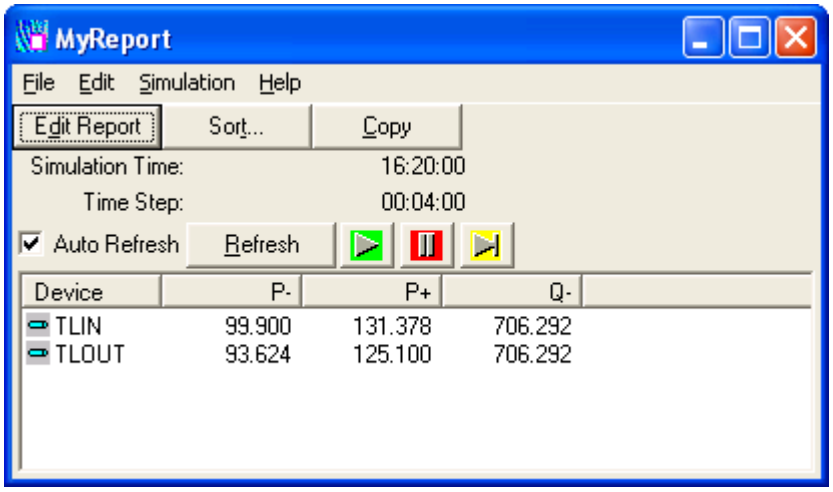


Figure 8-1 Report window

The Report button bar accesses the most commonly used menu features. Also, it displays the current simulation time and time step.

To create or edit a Report

1 From the Trans/Tport window, select **Window > Report** or click the Report toolbar button. 

2 In the Report dialog box, make sure that **Report** is selected and click **New**.

3 In the Report Editor, next to **Report Name**, type a name for the Report. A maximum of 80 characters is allowed.

4 Select the devices you want to display in the report.

To display	Do this
All devices of a particular type(s)	Click the **Types** browse button. In the Select Device Types dialog box, select the type(s) of equipment to display in the Report. **Note:** You can select multiple device types by pressing Ctrl while selecting items in the Select Device Types dialog box.
A particular device by name	Under **Name**, type the name of the device you want to display.
All devices of a particular subtype, such as SALE or TAKE	Under **Subtypes**, type the device subtype you want to display.
A range of device along a connected path	After selecting the device type(s), subtypes, or names, as described above, click the browse buttons next to **From** and **To** to select the boundaries for the devices that you want to select.

5 Under **Name Width**, type a value for the initial width of the Name column in the report.

Note: The column width may be changed later in the report by dragging the column to the desired width.

6 Perform the following steps to add, remove, and change items that will be included in the report:

a Under **Available Items**, click the drop-down arrow to choose an item for the report, or type the report item directly in the text field.

Notes: Only the peekable items associated with the equipment type(s) you selected are listed in Available Items. To select an item that is in array form, such as the coordinates of a data curve, you can type the array name and the index number(s) desired (for example, X (1) or X (2)).

When selecting attributes for any given device, a second attribute may be inserted after a colon to specify the query. For more information, see "Multiple colon syntax" on page 588.

b Under **Label**, type the column label text that will be displayed for this item in the report.

c Under **Label Width**, type a value for the initial width of the data column in the report.

Note: The column width may be changed later in the report by dragging the column to the desired width.

d Click **Add** to add the data item to the report.

e To *change* a data item in the report, select the data item in the **Selected Items** list and then edit the **Available Items**, **Label**, or **Label Width** fields, as appropriate. Click **Change** to accept the new values.

f To *remove* a data item from the report, select the data item in the **Selected Items** list and then click **Remove**.

7 If desired, click **Save** or **Save As** to permanently save the report as a display file.

8 Click **OK** to accept the changes and close the dialog box.

To open an existing Report

1 From the Trans/Tport window, select **Window > Report** or click the Report toolbar button. 

2 In the Report dialog box, make sure that **Report** is selected. Under **Existing Report**, select the report you want to view and then click **OK**.

Note: To display another report while you are already viewing a report in the in the Report window, select **File > View Different Report**.

To edit the current Report

1 While viewing the report that you want to edit, select **Edit > Edit Report** or click the **Edit Report** button.

2 In the Report Editor, change a Report item by clicking on the variable to which you want to make changes from the list of selected plot items.

Note: When selecting attributes for any given device, a second attribute may be inserted after a colon to specify the query. For more information, see "Multiple colon syntax" on page 588.

3 Edit the devices, variables, and other settings for your report according to the instructions provided in "To create or edit a Report" on page 183.

4 If desired, click **Save** or **Save As** to permanently save the report changes to a display file.

Tip: If you adjusted the report column widths in the Report window, the new column width(s) will be shown in the report editor. You can save the current column widths by saving the report.

5 Click **OK** to accept the changes and close the dialog box.

To sort data in a Report

1 Open the Report that contains the data that you wish to sort.

2 In the Report window, click **Sort**.

3 In the Sort Options dialog box, under **Sort By**, select the first variable that you want to use to sort the report.

4 Select the **Ascending** and/or **Magnitude** check boxes, if desired, to further define the sort order.

- Selecting *Ascending* to sort the data from least to greatest.
- Selecting *Magnitude* to sort the data using the absolute value of each item.
- Select neither option to sort the values in descending order.

5 Repeat step 3 and step 4 for the **Then By** and **Finally By** options, if desired, to further sort the report based by additional variable choices.

6 Select the **Re-sort every step** check box if you want to re-sort the data in the report based on the defined criteria with each time step.

7 Click **OK** to accept the changes and return to the report.

Tips: If defining more than one criterion for sorting, sorting uses the top criterion first, then the middle criteria, and lastly the bottom criterion.

To copy data in a Report to the Windows clipboard

1 Open the Report that contains the data that you want to copy.

2 In the Report window, click **Copy**.

The data copies to the Windows clipboard. You may paste to the software product of your choice.

To display a Report in a text-based format

1 From the Trans main menu, select **Window > Compatibility > Text Report Screen**.

2 Create or open a report following the instructions in "To create or edit a Report" on page 183 or "To open an existing Report" on page 185.

To update the display in a Report window

In the Report window, check **Auto Refresh** to automatically update the display with each time step.

—or—

Click **Refresh** to update to the current time step.

To print a report

1 From the Report window menu, select **File > Print**.

2 Use the Print dialog box to choose a printer and set other options for the print job.

3 Click **OK**.

Text displays

Text displays provide a way to display output in a customized format or to issue often used commands from a convenient menu. Text displays may contain both static information and dynamic information. Static data includes non-changing text used for labeling device and variable names next to their values, headings, and drawing simple schematics. Dynamic data is information that updates at each time step, such as pressure and flow data. You may also specify an option for the dynamic information to be changed during the simulation.

Text displays are created by entering text in a text editor and saving the file with the DSP file extension. Leave the first line in the text file blank. The first line is reserved for the status message. Static text or dynamic fields may be entered on the following lines.

While text displays may offer a visual representation of the model, they have no effect on the connectivity of the model. The schematic view in Model Builder allows you to build the connectivity of the model. For more information, see "Model Builder" on page 697. In addition, Schematic controls allows you to create a visual representation of the model and integrate it with other applications. For more information, contact GL.

Dynamic fields

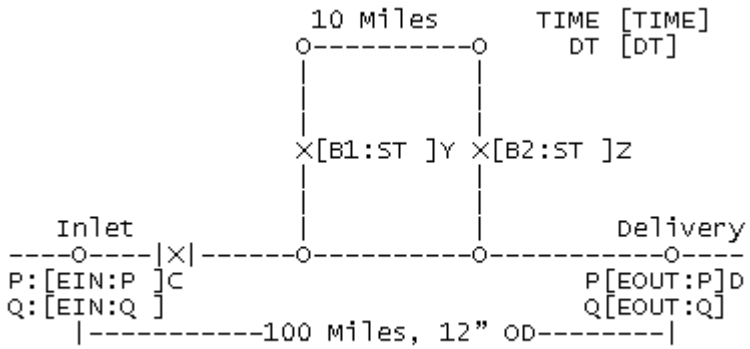
Dynamic information is defined by enclosing a peek attribute in square brackets ([ ]). The peek name can be followed by a comma and a format specification for the output:

```
device_name:attribute,nnn.dd
```

where `nnn` is the field width and `dd` is the number of decimal places.

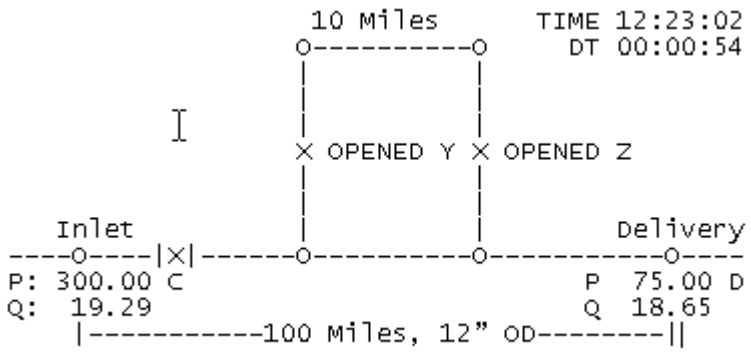
You may enter a character immediately following the closing bracket of a dynamic field. This character is called a poke character and provides a method for you to change the value of the simulation variable being displayed in the field.

The following text display shows a schematic view of the system. It contains both static information, such as the distance of 10 miles as well as dynamic information, such as the simulation time.



Sample text display (unrefreshed)

In the refreshed sample text display, the variable peek names previously shown in the square brackets [ ] are replaced by their value when TRANS is running. The rest of the text remained static. Numbers that refresh are right-justified in the field while character strings that refresh, such as component status, are left-justified in the field. Every peek attribute listed in "Peek and Poke Keyletters and Attributes" on page 639 can be displayed on a text display inside the square brackets.



Sample text display (refreshed)

Sample text display menu

You can create a custom text display that provides a convenient portal to various other displays. In a text editor, you can enter something like the following:

```
INDIRECT COMMANDS
COMMAND         POKE
[HELP]          A
[OVERVIEW]      B
[SHOW PIPE1]    C
[PUMPS]         D
[EXTERNALS]     E
```

Each of the entries under COMMAND represents a DSP file. To select a command, enter the POKE letter for the command and press Enter twice to execute the command. The poke letters are only valid from this display.

Aliasing expressions in text displays ($ALIAS)

In a text display, you can use the $ALIAS feature to replace a lengthy expression (device name:variable, defined variable, etc.) with a shorter expression. This feature is useful to display a lot of information on a small display.

The following example demonstrates the $ALIAS feature being used on a text display:

```
$ALIAS aaa=BLOCKVALVE1:P-, aab=LONGPIPE1:P-, aac=RELIEFVALVE_1:P-,
    aad=EXTERNAL12:SP
$ALIAS bba=BLOCKVALVE1:ST, bbb=LONGPIPE1:Q-, bbc=RELIEFVALVE_1:ST
                           TIME [TIME  ]
                           DT   [DT    ]
BLOCKVALVE1:P-    [aaa ]   LONGPIPE1:P- [aab  ]   RELIEFVALVE_1:P- [aac]
BLOCKVALVE1:ST    [bba ]   LONGPIPE1:Q- [bbb  ]   RELIEFVALVE_1:ST [bbc]
EXTERNAL12:SP     [aad ]
```

Some comments on the $ALIAS use in this text display:

- A comma must be between successive fields on the $ALIAS line. No comma is after the last field on the $ALIAS line. Each field consists of a unique name, followed by the equals sign (=), followed by a peekable device parameter or defined variable.

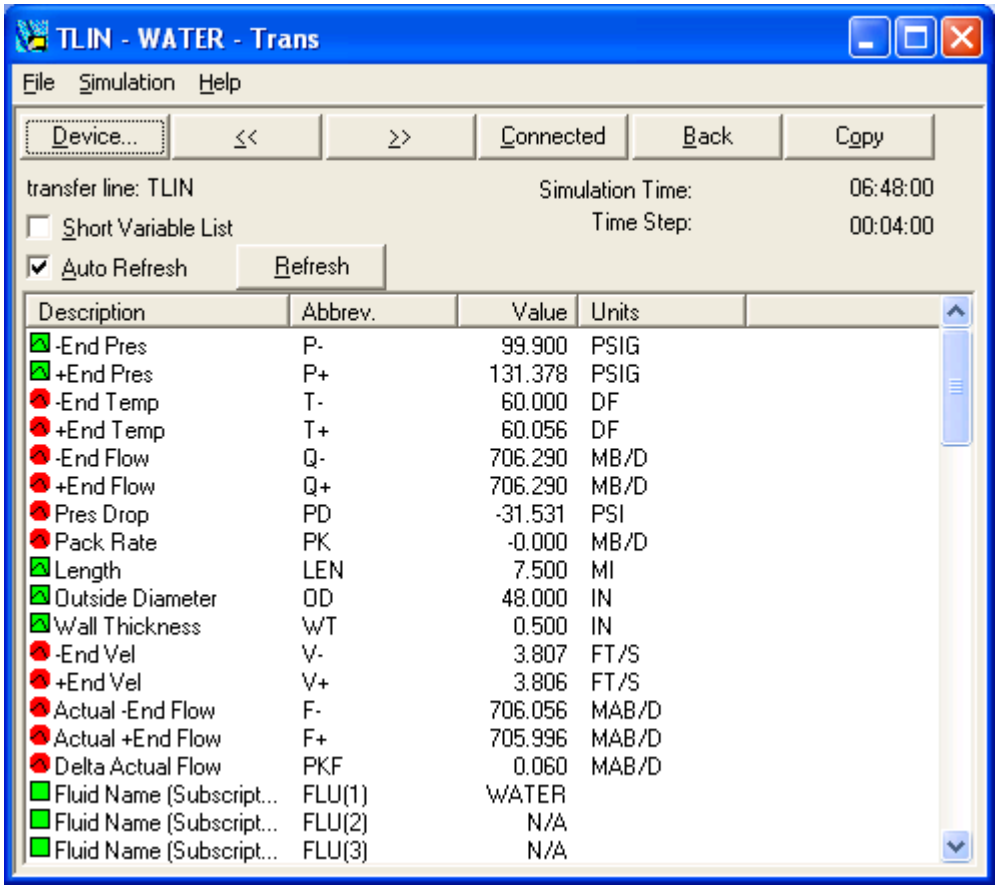
- If a text display is using the $ALIAS feature, only one $ALIAS command is needed for a text display. To have multiple field lines for one $ALIAS, put a comma after the last field on each line. In this example, multiple $ALIAS commands appear on the text display to group certain fields for organizational purposes.
- An $ALIAS line is limited to 80 characters. Anything longer than 80 characters is truncated. The display maybe greater than 80 characters.
- The unique name for the $ALIAS (e.g., aaa) may be in upper case or lower case and the number of characters in the name has no limit.
- $ALIAS names are alphanumeric but must start with an alpha character.

Show windows

All modeling elements and named nodes in a model have certain data associated with them, such as pressures, flows, temperatures, and statuses. Any of the data that SPS tracks for any modeling element, named node, defined variable, etc. is available for viewing in a Show window.

In addition to viewing data in a Show window, you can edit variables and request time plots from a Show window. The symbols next to each device attribute indicate whether you may edit the data and/or create a time plot.

	A variable with a green box preceding it is editable (may be “poked”). See “To edit variables from a Show” on page 159.
	A variable with a red circle preceding it is not editable (may only be viewed or “peeked”).
 	If the green icon box or red icon circle preceding the variable contains a curve through it, the variable is trendable (may create a time plot). Only variables specified in the TRENDLIST command in the INTRAN file may be displayed on a time plot. For more information on time plots, see “Time plots” on page 179 and “To create a time plot from a Show” on page 181.



To open a Show window

1 Open a Show window in one of the following ways:

- From the Trans/Tport window, select **Window > Show** or click the Show toolbar button  .
- From the Report window, select **File > Show** or double-click on the item you want to show.
- From the Report or Show window, right-click on the item you want to show.

  Note: The Show window is only available if the item you click on is a device. For example, if you click on the CB attribute for a compressor or pump, the value is the name of the controlling actuator. When you right-click and select Show, the Show window for the actuator appears.
- From the Compressor/Pump Map Plot window, select **File > Show Device Properties**.

2 In the Select Device dialog box, select a **Device Type** and **Device Name** to Show.

3 Click **OK**.

SPS opens the Show window of the selected device. To change the device displayed or select options in the Show window, see "To change the display in a Show window" on page 190.

To open a Show window from the command line

To display a Show window, enter the SHOW command followed by the name of the device. For example, display the information available for an element named BLOK1 on the command line:

```
SHOW BLOK1
```

A Show window displays data on BLOK1.

For more information, see "SHOW" on page 559.

To change the display in a Show window

1. Click items in the button bar to select a specific device, go to the previous or next device in the browse list, show connected devices, or display a history of recent Show selections.
2. Check **Short Variable List** to limit the list to the most frequently referenced attributes.
3. Check **Auto Refresh** to automatically update the display each time step.

   —or—

   Clear **Auto Refresh** and click **Refresh** to update to the current time step.

To show a device connected to the current device in the Show window

1. In the Show window, click **Connected**.

   The <Device> Connections dialog box displays the devices that are hydraulically connected to the device in the current Show window.
2. Select a device, and then click **OK** to display information for the new device in the Show window.

To display a Show window in a text-based format

1. From the Trans main menu, select **Window > Compatibility > Text Show Screen**.
2. Open the Show window following the instructions in "To open a Show window" on page 189.

To show a device previously displayed in the Show window

1. In the Show window, click **Back**.

   The Show History dialog box displays all of the devices previously displayed in the current Show window.
2. Select a device, and then click **OK** to display information for the new device in the Show window.

Overview of SimPlot

SimPlot allows you to generate graphs from the data computed by TRANS and written to the REVIEW file. You can use SimPlot to generate both Time charts and Distance charts for a wide range of simulation purposes, including surge studies, control studies, and inventory. The charts that you create in SimPlot can be printed, saved in several common graphic file formats (including BMP, JPG, and PNG), and exported to Microsoft Excel.

You can combine data from several different REVIEW files into a single chart. You can also view a time chart dynamically—as the REVIEW file is updated from a live simulation, the SimPlot chart will update to reflect the changes.

For more information on using SimPlot, see the SimPlot Online Help.

Printing

In Windows, you can print and in some cases print preview distance plots, time plots, reports, Show windows, and compressor/pump maps using standard Windows controls. In addition to printing to a printer or plotter, you may also print Show windows, reports and custom text displays to a file.

To print to a file

1 Display the Show window, report, or custom text display you want to print to a file. For more information, see "Working with displays" on page 163.

From	Do this
Command Line	The Report must be viewed in compatibility mode (Window > Compatibility). From a text display or Report, type **`print`** `new_filename` to print the text to a new file. The entire contents of the text display are printed to file. For entire contents of a Report only, type **`printall`** `filename`.
Menu	Select **File > Print** from the Trans/Tport window, the Report window, or the Show window.

2 In the Print dialog box, under Name, select the **Generic/Text Only** printer.

Note: You must set up a Generic/Text Only Printer. See the Windows help for more information)

3 Check the **Print to File** option.

4 In the Print To File dialog box, select the file name and directory.

5 Click **OK**.

CHAPTER

9

Validating, Troubleshooting, and Tuning

This section is intended to provide guidance in troubleshooting the model build process as well as finding ways to improve performance and accuracy of results during the simulation.

Troubleshooting during the model build process

Building a model is an iterative process. You should do the work in stages, using PREPR and TRANS throughout to ensure the integrity as you go. For example, after you complete a simple section of the model, you should run PREPR and then check the OUTPRP report. If input errors or unacceptable initial operating values in the INPREP file are detected by PREPR, an error message is given at the system prompt and the OUTPRP report shows where the errors occurred in the INPREP file and a possible reason for the errors. Edit the INPREP file, fix the input data and then re-run PREPR. Repeat this step as often as necessary until an acceptable working model is produced. Review the results of the OUTPRP report to see if the model processed without errors. After PREPR runs successfully on a relatively simple section of the model, gradually add more complexity to the model and re-run PREPR periodically until you have completed the physical portion of the model.

Now that you have all of the physical elements of the system in place, you need to create an INTRAN file, which describes how to control the simulation. This may involve writing control logic to schedule events and operations or issuing a command to manipulate data and results. Again, begin by adding simple commands to the INTRAN file and then run TRANS and check the OUTTRN report. If input errors in the INTRAN file are detected by TRANS, an error message is given at the system prompt and the OUTTRN file shows where the error occurred in the INTRAN file and a possible reason for the error. Edit the INTRAN file, correct the input data and then re-run TRANS. Repeat this step as often as necessary until TRANS starts running. Once you have reviewed and evaluated the output from a TRANS run, re-run the simulation as often as necessary to test variations in the operating scenario. Iteratively add more complex logic to the INTRAN file.

To test changes in equipment design or general data, make changes to the INPREP file, then re-run PREPR successfully before running TRANS again.

Tips for troubleshooting an input file

In addition to reviewing the "Input syntax" on page 47, check the following:

- Does the syntax require a comma at the end of the line or not? Omitting a required comma often results in a message EXPECTING CLOSING RIGHT PARENTHESIS from _____.
- Have you included all necessary closing brackets in logical statements and expressions?
- Have you accidentally commented out part of your data?

Performance tuning

The speed at which an SPS model runs has a direct impact on the amount of time it takes to tune the model, and an indirect impact on your patience and interest in tuning the model.

Speed performance is best tracked by viewing DCPU. DCPU is the number of CPU seconds required by the model to take a time step and it is also the sum of all the $AUDITS. Each of these $AUDITS indicates where the CPU is being spent. In other words, they show the time expense associated with each major component of SPS software. The term expense is used figuratively in this context and that is why the speed audits have a $ symbol in front and that is also why elements and configurations that use a lot of memory are often referred to as costly. The goal of speed tuning is to reduce the cost of these audits, thereby decreasing DCPU.

This section contains areas that you should concentrate on tuning. Within each topic is a listing of which audits will be affected and also the probable amount of effort required to make the necessary analysis and changes. For more information on the audits, see "AUDITS (AU) attributes" on page 643.

If you are working in a liquid system with slack line flow, you may want to tune the holdup. See "Slack line flow" on page 148 for more information.

OUTTRN error messages

Audits:	N/A
Effort:	Medium
Hydraulic impact:	Generally results in a faster, more accurate, and more robust model.

The first step in improving model speed performance is to reduce the number of error messages in the OUTTRN file. The sources of errors causing compressor trips, check valve chatter, or out of range pressures and velocities should be determined and corrected.

The next step is to find what is limiting the time step. Poke DEBUG to -4200 and view the OUTTRN file to find the location of pressure transients or errors that are limiting the time step.

Often a smaller error tolerance value (1 or 2 psi) reduces the out of tolerance messages. The model typically recovers from numerical error with a small tolerance but has more difficulty recovering from larger numerical errors. Also, a minimum time step that is too large can cause significant errors.

For example, equation of state errors can be caused by a number of problems. Some typical problems are the fluid is not in the same phase at custody conditions that it is at pipeline conditions. For liquid systems running SCL, if failures occur after pump startup, the bulk modulus definition probably has a problem.

TRENDLISTs

Audits:	$REVIEW
Effort:	Low
Hydraulic impact:	None

The TRENDLIST statement in the INTRAN file specifies the peeks that will be written to the REVIEW file. By reducing the TRENDLIST, $REVIEW will be reduced. Many of the elements have over 20 peeks that can be written to the REVIEW file, while you will usually view only five or six of these peeks in time plots. Using a detailed TRENDLIST that limits the peeks will reduce the amount written to the REVIEW file, which will reduce processing.

$REVIEW can also be affected by the amount of IMEM used with TRANS. For more information on memory settings, see "Memory allocation" on page 92.

SHARELISTs

Audits:	$REVIEW
Effort:	Low
Hydraulic impact:	None

You can specify which peeks and pokes are written to shared memory every time step by TRANS for display purposes in TPORT. TRANS converts the specified peeks and pokes from internal units to user units and then writes them to shared memory. This requires a certain amount of CPU every time step, which you can reduce by decreasing the number of points in the SHARELIST. For more information, see "Shared memory" on page 57.

Element post-mortems

Audits:	$ISOLVE, $NISOLVE
Effort:	Medium
Hydraulic impact:	High

Element post-mortems occur when the software could not converge to an answer within the appropriate tolerances. They usually indicate that a piece of equipment is modeled incorrectly or something is highly unstable. To find the number of NISOLVEs, view NISOLVE on a time plot. This line should be at one almost all the time (NISOLVEs greater than one during startup of the model are typical). You can determine which elements are in post-mortem by poking DEBUG = 890123 and then reviewing the OUTTRN file.

Knot spacing and pipes

Audits:	$MAIN
Effort:	Medium
Hydraulic impact:	High

Hydraulic calculations are performed at all knots in pipes. The more knots in a model, the slower it runs. SPS automatically determines a time step and knot spacing that works well in most situation. However, if you are an experienced user, you can override the minimum knot spacing by:

- Setting the knot spacing with PIPEPARMS
- Using the AMULT for looped lines
- Installing HEADERs in places of short pipes

When over-riding the knot spacing, be sure that the resulting minimum time step does not permit full travel of a valve in a single time step or spin-up or spin-down of pumps and compressors in a single time step. Knot spacing that is too large will both slow down the model and reduce the computational accuracy, and significant OUTTRN errors may be generated. If PREPR detects that more than 25% of the pipes in your model are less than the user-specified knot spacing, PREPR will report an error. You must correct this error before the simulation can proceed.

For more information on how SPS determines time steps and knot spacing, see "Calculation intervals" on page 197.

Batch size

Audits:	$BATCH, NBATCH, $MAIN
Effort:	Medium
Hydraulic impact:	High (Liquid systems) Medium (Gas systems)

If compositional tracking is being used, fluid property information will be stored within FMVs that are *launched* at all injections and *caught* at all deliveries. SPS will adjust DT so that FMV arrival at all complex connections (connection points with more than two pipes) falls precisely on a time step. Consequently, the more FMVs in a model, the more often SPS will reduce the time step.

To reduce NBATCH (the number of FMVs in a model) the TRIVB and TRIVC parameters should be increased. Similarly, to increase NBATCH and improve the fidelity of fluid property tracking, TRIVB and TRIVC should be decreased. As NBATCH is reduced, $MBATCH and $MAIN will decrease as well.

TPORT CPU use

Effort:	Low
Hydraulic impact:	None

TPORT can have excessive CPU use as it polls frequently for new data in shared memory. If TRANS takes several seconds to take a time step, TPORT CPU use can be reduced by defining in INTRAN the variable $TPORT_SLEEP_FACTOR = time, where time is the sleep time in seconds. For more information, see "Shared memory" on page 57.

Calculation intervals

SPS calculates variables, such as pressure, flow, etc. at standard distance increments called knots and time intervals called time steps. Both time steps and the knot spacing are determined automatically by SPS, but you may override these settings. Before choosing to override these settings, you should understand:

- How SPS calculates time steps and knot spacing
- The interrelationship of time steps and knot spacing
- Reasons for adjusting the time step and/or knot spacing
- Guidelines when overriding time steps and knot spacing

Time steps

The time step (DT) at each iteration must fall between the bounds of a maximum (DTMAX) and minimum (DTMIN) time step. The default maximum time step is 60 minutes. The minimum time step (DTMIN) is determined based on the minimum of:

- Splitting the shortest pipe (T or GP) into two segments and then dividing by the wave speed
- Dividing the shortest opening/closing time of actuators or valves by five
- Dividing the shortest spin-up/spin-down time of rotating equipment by five

Other factors may affect the time step:

- The time step varies dynamically depending on hydraulic conditions. Smaller time steps are required to model transients, but larger time steps are used when the model is relatively steady.
- Any time-based action specified in the INTRAN file causes the time step to land as close as possible to the specified time. Examples are POKE, OPEN, CLOSE, START, STOP, ARCHIVE, PROFILE, etc.
- The POKE command causes the time step to go to the minimum, DTMIN, immediately after the specified time, and should be used for hydraulic related settings. The SET command does not cause the time step to go to the minimum, and can be used for non-hydraulic related settings.
- Because batches must arrive at connection points at a time step, systems using equations of state that allow batch or composition tracking can limit the step. Conversely, disabling batch and composition tracking for single fluid systems using the NOTRACK option will increase the time step.

The time step may be displayed through Show windows, displays, time or distance plots, and reports through the GLOBALS: DT attribute. For more information, see GLOBALS (GB) attributes.

Knot spacing

SPS calculates the knot spacing based on the calculated minimum time step and the sonic velocity. The calculated velocity can be overridden using the PIPEPARMS command. See “PIPEPARMS” on page 216. Every pipe (T or GP) contains at least 3 knots, regardless of the knot spacing. The knot spacing remains consistent throughout the simulation.

To better understand knot spacing, consider the following example. Assume that you have a 20 mile transfer line. With a knot spacing of 2 miles, the transfer line will be broken into 10 calculation intervals. With a knot spacing of 4 miles, it

would be broken into 5 calculation intervals. If the knot spacing is 10 miles, the transfer line would be broken into 2 segments.

Overriding time steps and knot spacing

Although a relationship exists between time steps and knot spacing, they are not directly dependent during the simulation. Time steps may vary throughout a simulation while knot spacing does not. In general, larger knot spacing coincides with a longer time step.

While smaller knot spacing allows tighter tracking of volatile transients and surges, a time step that is too small will slow down the calculation. Knot spacing that is too large will both slow down the model and reduce the computational accuracy, and significant OUTTRN errors may be generated.

You should change these settings only if you are an experienced user, trying to meet specific goals to tune your model. For example, you may want to decrease knot spacing to track volatile transients and surges or you may need to reduce DTMAX because you have a TRANSTHERMAL model. For other specific instances, you may want to discuss the changes with Technical Support. If you do choose to alter these settings you must proceed carefully.

You may control the time step (DT) by specifying the maximum (DTMAX).

- DTMIN may be changed by adjusting the knot spacing in the INPREP file. For more information, see “PIPEPARMS” on page 216. When controlling the minimum time step, you may need to experiment with the highest value that provides good results.
- DTMAX may also be changed through the INTRAN file or interactively using the GLOBALS keyword. If you want to limit time step growth, you can set DTMAX to DTMIN. See “GLOBALS (GB) attributes” on page 665.

You can override the minimum knot spacing by:

- Setting the knot spacing with PIPEPARMS. See “PIPEPARMS” on page 216.
- Installing headers (HEADER) in place of short pipes (T or GP). See “Header (H)” on page 276.

When overriding the knot spacing, be sure that the resulting minimum time step does not permit full travel of a valve in a single time step or spin-up or spin-down of pumps and compressors in a single time step. In addition, the changes in flow for an external must come from the pack in the connected pipe(s) for the first time step.

If PREPR detects that more than 25% of the pipes in your model are less than the user-specified knot spacing, PREPR will report an error. You must correct this error before the simulation can proceed.

Tuning knot spacing and time steps

To test the knot spacing, choose a “worst” case scenario for the type of system to be simulated (i.e. the same fluid, the same pipe size, the fastest opening valve, or the most rapidly changing flow rate possible from an external). Then run with your selected knot spacing. Include a +PROFILE record in the INTRAN file to produce a distance plot at a time when the front should be the steepest you think it could be. Save the resulting REVIEW file. Then run with half the knot spacing. Then use to produce plots of all three runs. If the original knot spacing is satisfactory, then all the curves should lie on top of each other. If you see a gradual progression toward the solution with the smallest knot spacing, then repeat the process with still smaller knot spacing until you find one that gives two curves that agree. The largest knot spacing that produces agreement is then a safe knot spacing to use. The same procedure can be used to determine the largest time step that can be used.

Simulation time

When you set the simulation time in SPS, you need to consider the following modeling objectives:

- Will your model be an online model or an off-line model only?
- Does your model need to run from a specified date and time?
- If you are running from a specific point in time, does your model need to account for daylight savings time?

Once you have determined the answers to these questions, you need to specify what time format you will use and then use this format consistently to specify the starting conditions, operational changes, and end times.

Elapsed time versus clock format

You can indicate times in SPS as:

- Minutes since the start of the simulation (elapsed time)
- A specific point in time (clock format)

Typically, you would use elapsed time for an off-line model and clock format for an online model. If you want to extend an offline simulation to several weeks or reference particular days, then you should use clock format. For more information on time settings for an online model, see "Differences between off-line and online models" on page 744.

Regardless of whether you use elapsed time or clock format, the entry method should be used consistently and then the output will be reported in the same format.

Elapsed time

Elapsed time is typically the format used for offline modeling. To specify elapsed time, you must enter the begin time in the INTRAN file in minutes since the start of the simulation.

The following example from an INTRAN file specifies that the simulation start from an initial record from the RESTRT file and time zero. For more information on initial settings, see "Preparing to run a simulation" on page 153.

```
BEGIN 0,
+ BEGIN.TIME = 0
```

If you are using elapsed time, all time entries in your model should be entered in minutes since the start of the simulation.

Clock format

Clock format is the format you need to use for online modeling. You may use it under specific circumstance for offline modeling. For more information, see "Elapsed time versus clock format" on page 199.

To specify clock format, you must enter the begin time as an expression using the TIMEVALUE function.

The following example from an INTRAN file specifies that the simulation starts from an initial record from the RESTRT file and 1:40 p.m. on November 6, 2002:

```
BEGIN 0,
```

```
+ BEGIN.TIME = TIMEVALUE("02/11/06 13:40:00")
```

The TIMEVALUE function converts the clock format into SPS internal time for processing but then outputs times for reporting in clock format. For more information on SPS internal time, see "Display of simulation time on displays and reports" on page 200.

Once clock format is selected, all commands to TRANS that are related to time should be provided in clock format. Some commands expect an expression for the entry of clock format and therefore must use the TIMEVALUE function. Other commands expect the entry of clock format to be specified as a quoted string.

Note: For more information on how you should enter the clock format for each command, see the syntax and example for each command you want to use.

Display of simulation time on displays and reports

SPS converts both elapsed time and clock format to an internal time for processing and then reports the time in the same format you used to input.

Display of elapsed time

If you enter all times in elapsed time, SPS displays time on reports, time plots, distance plots, and other displays in elapsed time. The elapsed time is not reported as minutes since the beginning of the simulation but rather the time is converted to seconds, minutes, hours, and days, as necessary. For example, 1439 minutes since the start of the simulation will be displayed as 23:59:00. At 1441 minutes, the time will be displayed as 01 00:01:00. At 31 days and 1 minutes, the time will be displayed as 02/01 00:01:00. The second month has 29 days. After the time reaches 12/31 23:59:59, the next time that displays will be 69/01/01 00:00:00.

Display of clock format

If you are using clock format, SPS displays time on reports, time plots, distance plots, and other displays in clock format. However, if you are using clock format, you need to be aware of some environment variable settings that ensure the proper output of the simulation time.

All time values that are converted to or from internal time are converted based on the user-defined environment variable DREMTIMEZONE. Additionally, the various time function operators, such as HOUR(X) and DAY(X) respect the DREMTIMEZONE environment variable.

GL recommends use of the DREMTIMEZONE = LOCAL setting. Before choosing a setting you should consider the following issues.

The setting DREMTIMEZONE = LOCAL:

- Is recommended and default setting for Windows.
- Results in time conversions based on the operating system environment variable TZ, which takes into account the time zone and daylight savings time conversions.
- Requires all times that are passed to the model via API calls to be GMT.
- Has no time discontinuity due to daylight savings time conversions.

- Is more compatible with INTRAN commands that use a repeat period because the period is relative to GMT, not local time. For example, an item with a repeat period of 1440 will trigger at the start of the day GMT, not start of the day local time. Use the keyword DAILY on the TIMETABLE command to trigger events at the start of the day using local time.

The setting DREMTIMEZONE = STANDARD:

- Is not recommended.
- Does not perform time zone or daylight savings time conversions.
- Requires all times that are passed to the model via API calls to be the system time.
- Results in a time discontinuity any time the system time is changed, such as for daylight savings time.
- Requires the following steps if the computer system time must be changed for daylight savings time:
  - Completely shut down the model and interfaces before the time change
  - Re-initialize the RTUDATA file after the time change
  - Restart the model using a LOAD.STATUS with the SET.TIME option

For more information on how to set the DREMTIMEZONE environment variable, see "SPS environment settings" on page 87.

You will also need to set the local time zone. GL recommends setting the time zone setting TZ through the sps.settings file. See "sps.settings files" on page 91.

Error tolerance

SPS solves non-linear systems of equations at each time step. Transient analysis consists of a linearized solution of partial differential equations. The pressure error (in psi) in the simulation is estimated by evaluating the maximum of the absolute value of half the second difference of pressure using the last three time levels. In other words, it takes the second derivative of pressure with respect to time over the last two values. This is explained as follows: Suppose you have three values of pressure; P_1, P_2 and P_3, corresponding to time t_1, t_2 and t_3, respectively, and this tolerance calculation is to decide whether to accept P_3.

There are three different time step points (a calculated pressure at a specific time). The current first derivative of pressure is $(P_3-P_2)/(t_3-t_2)$ and is valid at time $(t_3+t_2)/2$.

The old first derivative of pressure was $(P_2-P_1)/(t_2-t_1)$ and was valid at time $(t_2+t_1)/2$.

This gives a second derivative approximated by $((P_3-P_2)/(t_3-t_2) - (P_2-P_1)/(t_2-t_1)) / ((t_3+t_2)/2 - (t_2+t_1)/2)$.

If t_3-t_2 is called DT, and t_2-t_1 is called DTOLD, the approximate second derivative in time can be rewritten as:

$$\frac{d^2P}{dt^2} \cong 2\frac{\left(\frac{(P_3 - P_2)}{DT} - \frac{(P_2 - P_1)}{DTOLD}\right)}{(DT + DTOLD)} \equiv \overline{P}_{tt}(DT)$$

Because most approximations are linearized to be second-order correct in time, a very conservative assumption is that the maximum local error in pressure can be bounded by:

$$\frac{1}{2} \left|\bar{P}_{tt}\right| \overline{DT^2}$$

The latter term approximates the quadratic term in Taylor's Theorem, and is the maximum deviation from a straight-line extrapolation. Thus for this step, the value of DT must be chosen so that the maximum local error due to curvature in pressure as a function of time should not exceed the user-defined bound TOLERANCE:

$$\frac{1}{2} \left|\bar{P}_{tt}\right| \overline{DT^2} < \langle \text{TOLERANCE} \rangle$$

Moreover, again to be conservative, an additional heuristic test is included which involves the value t by which the pressure changes during the time step. The measure of local error for determining DT for this step is to choose DT so that:

$$\text{MAX}_{\text{nodes.knots}} \left\{ \text{Min} \left(\left| (DT)^2 \frac{\frac{(P_3 - P_2)}{DT} - \frac{(P_2 - P_1)}{DTOLD}}{(DT + DTOLD)}, \frac{20|P_1 - P_3|}{(DT)^2} \right| \right) \right\} < \langle \text{TOLERANCE} \rangle$$

where:

$$\bar{P}_t = \frac{P_3 - P_2}{DT}$$

If, at minimum time step,

$$\bar{P}_t$$

exceeds the value of TOLERANCE at any knot or node in the system, an error message is written to the OUTTRN and EVENTS files; otherwise, the time step is automatically reduced when this estimate exceeds TOLERANCE. In some situations, e.g., in column separation, this second difference is not a good measure of the local error, and the error may appear larger than the tolerance even at minimum time step. The TOLERANCE is specified on the BEGIN line. See "BEGIN" on page 434.

Boundary conditions

Reasonable boundary conditions must be established in the model data. If the boundary conditions are based on actual pipeline operation, the model should run.

The following issues cause problems in transient modeling:

- You cannot control both pressure and flow from an external at the same time; this is a steady-state concept. It simply cannot be done in the field, and it does not work in the model. Instead, control either pressure or flow at the external with limits set for the other variable. The model calculates the uncontrolled variable. If the model is set up accurately, the uncontrolled variable should track actual conditions.
- A flow-controlled external should not be the only element attached to a valve. If the valve closes or if sonic flow is achieved, unreasonable pressures may result. If the external needs to be flow controlled, PMIN and PMAX limits should be set to prevent unreasonable pressures.

- A flow-controlled external should not be used to supply flow to pumps or compressors. The pumps or compressors are not necessary and may automatically shut down because the external allows the pressure to rise to meet the required pressure downstream of the pump or compressor. Externals supplying pumps and compressors should be on pressure control.
- Two pressure-controlled externals should not be connected to the same node.

CHAPTER

10

The INPREP File

The INPREP file is a required text input file that includes the following data:

- All reference information needed to build the model. This includes standard units, standard reference values, temperature mode setting, equation of state to be used, and friction factor that is to be used for the pipe parameters.
- All information needed to describe the physical components of the model. This includes equipment and control instrument specifications, connections, dimensions, reference settings, data curves, etc.).

This chapter of the *SPS Help and Reference* is divided into the following topics:

- "Required input for the INPREP file" on page 205 lists the required input sections for the INPREP file.
- "INPREP file input" on page 206 lists all input sections that can be used in the INPREP file.
- "Sample INPREP file" on page 209 provides a short sample of an INPREP file. Other sample INPREP files are provided with your SPS installation.
- Finally, starting with "TITLE" on page 210, this section provides a detailed description of each input section that you can use in the INPREP file. Each section includes the input format, input examples, and additional input examples and notes where necessary.

Required input for the INPREP file

The following input is required in every INPREP file:

- TITLE line (must be on the first line)
- Phase (LIQUID or GAS)
- Units (ENGLISH or METRIC)
- Reference values for standard conditions (CUSTODY)
- Global pipe parameters (PIPEPARMS)
- Thermal mode (ISOTHERMAL, THERMAL or TRANSTHERMAL)
- Equation of state (STATE AGA, STATE BWRS, STATE CNGA, STATE SCL, VISCOSITY (non-Newtonian), WAX, STATE TABLE)
- =EQUIPMENT line to separate general data from element data
- Elements, nodes, user-defined variables, defined paths, data curves, etc. in any order

For basic descriptions of the various settings and element types, see “Modeling the Physical System” on page 119. For a full list of input you may enter in the INPREP file, see “INPREP file input” on page 206. For information on syntax requirements, see “Input syntax” on page 47.

INPREP file input

Options

Equipment

Nodes and Externals

Pipes

Valves and regulators

Compressors

Pumps

Control elements

Other

Common input

Common commands

"TESTMACRO" on page 584

For more information on using logic with commands, see "Expressions, Operators, and Functions" on page 587.

Sample INPREP file

```
ISOTHERMAL AGA, WITH SALE AND TAKE EXTERNALS
GAS
ENGLISH
ISOTHERMAL 60

STATE AGA
 + GAS 0.5896 0.001 1061.59
 + COST 1.75
 + H2O .08
+ HS .0001
 + C1 .95
 + C2 .03
 + C3 .01
+ IC4 .004
 + NC4 .003

CUSTODY PRESSURE = 14.969, TEMPERATURE = 60

 /* PIPELINE CONFIGURATION DATA
 =EQUIPMENT

E   IN1   NODE1  TAKE   500    0   60
E   IN2   NODE2  TAKE   500    0   60
GP  PIPE1 NODE1  NODE3   10   30  0.375  60  60
GP  PIPE2 NODE2  NODE3   10   30  0.375  60  60
GP  PIPE3 NODE3  NODE4   25   30  0.375  60  60
E   OUT   NODE4  SALE     0  200   60
E   OUT   NODE4  SALE     0  200   60
```

TITLE

INPREP

```
TITLE
```

Field	Units Key	Description
TITLE	n/a	Title or name of the model.

Description

The TITLE line is the first non-commented line in the INPREP file. PREPR terminates with an error if the first non-commented line is a valid command. Use one line, with any combination of blanks and spaces, to express the title of the simulation. SPS removes leading and trailing blanks.

Example input

```
THIS IS AN EXAMPLE TITLE
```

SELECT (INPREP)

INPREP

```
SELECT PRODUCT
[+ PERIODS = P1 P2 P3 ...]
[+ PRESSURE = P]
[+ TEMPERATURE = T]
[+ PDF = YES | NO]
[+ BMC = YES | NO]
[+ SPANS = YES | NO]
```

Field	Units Key	Description
PRODUCT	n/a	Name of the product to be run. One of: SIMULATOR, TRAINER, STATEFINDER, LEAKFINDER, or PREDICTOR. {SIMULATOR} **Note:** Must be the same in both INPREP and INTRAN command.
P_i	TIME	Leakfinder averaging periods. (LEAKFINDER only)
P	PRESSURE	Pressure used for reporting and balancing diagnostic flows. This value should be approximately the system wide average operating pressure. {100 psig} (STATEFINDER and LEAKFINDER only)
T	TEMP	Temperature used for reporting and balancing diagnostic flows. This value should be approximately the system wide average operating temperature. {60°F} (STATEFINDER and LEAKFINDER only)
PDF	n/a	Flag that indicates whether to include PDF in state estimation. {YES for liquid, NO for gas} (STATEFINDER and LEAKFINDER only)
BMC	n/a	Flag that indicates whether to include BMC in state estimation. {YES for liquid, NO for gas} (STATEFINDER and LEAKFINDER only)
SPANS	n/a	Enable SPANS for autocalibration and generates a case-spans.dsp custom display file. This file may help when studying and tuning spans autocalibration. {YES} For more information on spans, see “Spans” on page 145.

Description

SELECT specifies a product and is validated against the license file. The SELECT statement in the INPREP file must match the SELECT statement in the INTRAN file. For more information, see “SELECT (INTRAN)” on page 517.

Example input

Use the Statefinder component in the INPREP file:

```
SELECT STATEFINDER
+PRESSURE = 500
+TEMPERATURE = 65
```

GAS

INPREP

GAS

Description

GAS specifies that the fluid(s) in the model is gas.

Example input

Specify simulation of a gas system.

```
GAS
```

LIQUID

INPREP

```
LIQUID
```

Description

LIQUID specifies that the fluid(s) in the model is liquid.

Example input

Specify simulation of a liquid system.

```
LIQUID
```

CUSTODY

INPREP

```
CUSTODY
[+PRESSURE = PREF,]
[+TEMPERATURE = TREF]
```

Field	Units Key	Description
PREF	$PRESSURE_{abs}$	Reference pressure for custody transfer {14.696 psia}
TREF	TEMPERATURE	Reference temperature for custody transfer {60 °F}

Description

CUSTODY is required for all models. You can specify reference values for pressure and temperature (standard conditions) used for custody transfer and in the pipe expansibility equation. These values are used to convert between standard and actual volumes. Make sure you have selected values that do not cause the fluid to be in a different phase than in the model.

Example input

Establish custody conditions of 14.73 psia and 60°F.

```
CUSTODY
+ PRESSURE = 14.73,
+ TEMPERATURE = 60
```

PIPEPARMS

INPREP

```
PIPEPARMS
[+ FRICTION [COLE | NIKUR | MOODY [RUF | FF]]
[+ INITIAL PINIT]
[+ KNOT SPAC]
[+ THRM.COEFF n]
```

Field	Units Key	Description
RUF	ROUGHNESS	Colebrook or Nikuradse roughness. See "Entering default friction factor and roughness in Model Builder" on page 216. {0.0009}
FF	n/a	Moody friction factor. See "Entering default friction factor and roughness in Model Builder" on page 216. {0.013}
PINIT	PRESSURE	Initial pressure for the model at the highest elevation entered at a pipe-end. Used only to initialize the system at zero flow (hydrostatic equilibrium). {300 psig} Also see "Preparing to run a simulation" on page 153.
SPAC	LENGTH.PIPE	User-defined knot spacing. Enter the knot spacing only if you want to override the default knot spacing determined by SPS. See "Calculation intervals" on page 197 for detail on how SPS determines knot spacing.
n	1/ΔTEMPERATURE	Thermal expansion coefficient. {1.33E-6}

Description

PIPEPARMS specifies various global parameters applicable to pipes (T or GP).

Take note

Entering default friction factor and roughness in Model Builder

Default roughness and friction are set through the Limits editor or by right-clicking on the field in an editor and selecting Properties. For more information, see "Limits editor" on page 709 and "Setting defaults, limits, and other attribute properties" on page 724.

Example input

Example 1

Define pipe (T or GP) parameters such that the initial pressure is 500 psig and the Colebrook correlation is used for friction factor.

```
PIPEPARMS
+ FRICTION COLE
+ INITIAL 500
```

Example 2

Define pipe (T or GP) parameters such that the initial pressure is 700 psig, a Moody friction factor of 0.015 is used, and the nominal knot spacing is 2 miles.

```
PIPEPARMS
+ FRICTION MOODY 0.015
+ INITIAL 700
+ KNOT 2
```

NOTRACK

INPREP

```
NOTRACK
```

Description

NOTRACK “turns off” the tracking of batches or fluid blending for simulations using the AGA, BWRS, CNGA or SCL[PROP] equations of state. These equations of state automatically initialize batch tracking and/or blending simulations. When batch tracking or blending, SPS tries to control the time step so that the fluid mixture vectors reach the pipe end exactly at the end of a step so that any equipment at the pipe end (density sensor, for example) uses the appropriate composition. If neither tracking nor blending is needed, these features can be turned off by entering NOTRACK.

NOTRACK must be input after the equation of state and before EQUIPMENT.

Take note

Cannot set fluid properties

If NOTRACK is selected, you cannot set fluid properties at nodes or externals.

Example input

Disable tracking of batches or fluid blending:

```
NOTRACK
```

SET.LIMIT

INPREP

```
SET.LIMIT
+ KEY SUB ATT LL LOW HIGH HH DEF UNIT
+ KEY SUB ATT LL LOW HIGH HH DEF UNIT
+ ...
```

Field	Units Key	Description
KEY	n/a	Keyletter corresponding to the device for which the limits or default value should be changed.
SUB	n/a	Subtype associated with the device. If the device has no subtype, use an asterisk (*) as a placeholder. For example, P-CONTROL is a subtype for an external (E).
ATT	n/a	Attribute for which new limits or a default value is being entered. Review the OUTPRP report to see the attributes.
LL	user units	The Low-Low limit will cause a fatal error if an input value is less than this value. An asterisk (*) is a placeholder and the existing value is kept.
LOW	user units	Any value less than this Low limit, but greater than the low-low limit will generate a warning. An asterisk (*) is a placeholder and the existing value is kept.
HIGH	user units	Any value higher than this High limit, but less than the high-high limit will generate a warning. An asterisk (*) is a placeholder and the existing value is kept.
HH	user units	Any input value greater than the High-High limit will generate a fatal error. An asterisk (*) is a placeholder and the existing value is kept.
DEF	user units	Default input value for this variable. If no input is given this value will be used. An asterisk (*) is a placeholder and the existing value is kept.
UNIT	n/a	Unit in which the input values are given. The unit only needs to be specified if the input is in units other than the user units.

Description

SET.LIMIT serves two purposes:

- Helps you detect input errors.
- Allows you to redefine global default values for various data attributes.

A summary displays at the completion of PREPR in the OUTPRP report. A list of valid keyletters, subtypes, and attributes displays as well as default limit values. Check for messages and errors in OUTPRP, which describe any exceptions or input out of range.

The asterisk (*) can be used as a placeholder on the SET.LIMIT command so that the corresponding value will be left intact.

Input limits

Input limits help detect input errors that are easy to make, yet difficult to find, such as entering header length in feet when it should be in miles.

Warning messages are generated for values that are less than the Low limit and greater than or equal to the Low-Low limit and greater than the High limit and less than or equal to the High-High limit. Fatal errors are generated for input less than the Low-Low limit or greater than the High-High limit.

The default limits were chosen to allow most models to run. For specific models, you may want to tighten limits to perform more stringent data checks.

Default values

You can redefine default values for each instance of a particular device type. However, if you supply a specific value for a particular device, the specific value will be used. For example, SET.LIMIT can be used to set the default CVO for all block valves (BV). This value is the default for all BV elements unless you supply a different value when defining a particular BV element in the =EQUIPMENT section. If you choose to use the default value when defining a particular device, use an asterisk (*) as a placeholder.

Example input

For the pipe diameter attribute, OD, specify a low limit of 20 inches and a high limit of 48 inches so that the software will give warnings when OD is outside this range. No information is given after the 48 because the defaults are acceptable.

```
SET.LIMIT
+ T * OD * 20 48
```

STATE AGA

INPREP

```
STATE AGA [TRIVB] [TRIVC]
+ GAS SG CO2 HHV
+ label1 val1
...
+ labeln valn
```

Field	Units Key	Description
TRIVB	VOLUME	Batch size below which the batch should be dropped from consideration
TRIVC	n/a	Mass concentration below which the fluid in question should be dropped from consideration in a blended mixture {0.005}
SG	n/a	Specific gravity of the gas at custody conditions
CO2	COMPOSITION.CO2	Mole fraction of carbon dioxide
HHV	HEATING.VALUE	Volumetric Higher (Gross) Heating Value of the gas at custody conditions.
$label_n$	n/a	Label for a user-defined gas property to be tracked. Maximum of a four-character name.
val_n	n/a	Numerical value corresponding to the label.

Description

The AGA equation of state may be used for multi-fluid simulations and is an implementation of the Gross Characterization Method of AGA Report No. 8, “Compressibility Factors of Natural Gas and Other Related Hydrocarbon Gases.” It may also be used to track user-defined fluid properties.

Take note

Usable ranges

“Compressibility Factors of Natural Gas and Other Related Hydrocarbon Gases,” Second Edition, November 1992, published by AGA, outlines guidelines for using the Gross Characterization Method. According to this document, STATE AGA is valid for temperatures from 32°F to 130°F (0°C to 55°C) and pressures to 1200 psia, provided the natural gas characteristics fall within the nominal ranges defined in the following table.

Quantity	Range
Relative density[a]	0.554 to 0.87
Gross heating value[b]	477 to 1150 btu/scf
Gross heating value[c]	18.7 to 45.1 MJ/m^3
Mole percent methane	45.0 to 100

Quantity	Range
Mole percent nitrogen	0 to 50.0
Mole percent carbon dioxide	0 to 30.0
Mole percent ethane	0 to 10
Mole percent propane	0 to 4.0
Mole percent total butanes	0 to 1.0
Mole percent total pentanes	0 to 0.3
Mole percent hexanes plus	0 to 0.2
Mole percent helium	0 to 0.2
Mole percent hydrogen	0 to 10.0
Mole percent carbon monoxide	0 to 3.0
Mole percent argon[d]	--
Mole percent oxygen[e]	--
Mole percent water	0 to 0.05
Mole percent hydrogen sulfide	0 to 0.02

a. Reference condition: Relative density at 60°F, 14.73 psia

b. Reference conditions: Combustion at 60°F, 14.73 psia; density at 60°F, 14.73 psia

c. Reference conditions: Combustion at 25°C, 0.101325 MPa; density at 0°C, 0.101325 MPa

d. Reference conditions: Combustion at 25°C, 0.101325 MPa; density at 0°C, 0.101325 MPa

e. Reference conditions: Combustion at 25°C, 0.101325 MPa; density at 0°C, 0.101325 MPa

The AGA equation of state will fail if you use values outside of the prescribed range, and you will likely receive “Equation of state failed” error messages in the OUTTRN file. If your values are out of range, you should use the BWRS equation of state instead of AGA. For more information, see “STATE BWRS” on page 224.

User-defined qualities

The user-defined qualities permit definition of any number of additional qualities to be tracked through the system, but each user-defined quality has storage overhead associated with it. Each quality is given a numerical value. The user-defined qualities are mixed on a volumetric basis. All nodes and externals will have the same list of user-defined properties.

Mixing rule

The fluid properties are mixed based on mass. The user-defined properties are mixed based on volume.

Use of NOTRACK

Using the AGA equation of state automatically initializes batch tracking and/or blending simulations. If neither tracking nor blending is needed, these features can be turned off by entering NOTRACK after the AGA input. See “NOTRACK” on page 218. Tracking and blending slow the simulation and should be disabled if not needed.

Units for composition

You may choose units for composition, including fraction, percent, or parts per million. For more information, see “Units for composition” on page 130.

Example input

The gas has a specific gravity of 0.6, a higher heating value of 1000 BTU/FT^3, and a carbon dioxide mole percent of 1%. Two user-defined properties are to be tracked: cost and water content of 7 (in arbitrary units, such as LB/MMCF).

```
STATE AGA
+ GAS .6 0.01 1000
+ COST 2.25
+ H2O 7
```

STATE BWRS

INPREP

```
STATE BWRS[.MOLE] [TRIVC] [TRIVB]
+ NAMES NAME1 NAME2 NAME3 ...
+ INITIAL FR1 FR2 FR3 ...
[+ VISC V01 V02 V03 ...]
[+ VPMI VPMI1 VPMI2 VPMI3 ...]
[+ VTMI VTMI1 VTMI2 VTMI3 ...]
```

Field	Units Key	Description
.MOLE	n/a	Use mole fractions instead of mass fractions. If BWRS.MOLE is specified in the INPREP file, then the E command in INTRAN is in mole fractions. Also, any reported compositions are in mole fractions.
TRIVC	n/a	Mass concentration below which fluid components should be dropped from consideration. {0.005}
TRIVB	VOLUME	Smallest batch volume to consider in tracking fluid composition variations. {0.5 MBBLS}
$NAME_j$	n/a	Name of a recognized component from the "Component names" table, presented later in this section.
FR_j	COMPOSITION	The mass or mole fractions corresponding to the fluids that were entered on the NAMES command. The total number of mass fractions entered must be equal to the number of components entered and the sum of the mass fractions must equal one. For more information on units, see "Units for composition" on page 130.
$V0_j$	VISCOSITY	Viscosity for each of the named components {0.01 cp}
$VPMI_j$	$1/\Delta P$	Pressure coefficient of viscosity {0/psi}
$VTMI_j$	$1/\Delta T$	Temperature coefficient of viscosity {0°F}

Description

The Benedict-Webb-Rubin-Starling (BWRS) equation of state may be used for multi-fluid simulations and is based on the Starling modification of the Benedict-Webb-Rubin formulation. It permits accessing a set of pre-defined components by name with corresponding BWRS coefficients. The BWRS equation of state may be used for gases at high pressure or mixtures of dissimilar fluids. This includes mixtures of hydrocarbons and acid gases (CO_2, H_2S, N_2, etc.), especially in situations where known compositions of fluid sources may mix in varying ratios in the network.

BWRS may be used for liquid systems such as LPG, or dense phase fluids like ethylene or carbon dioxide. BWRS is not suitable for simulation near the two-phase region.

The BWRS equation of state is only valid for a single phase during one simulation. Conditions should not exist in the simulation that would cause more than one phase to exist. This includes the phases at the custody pressure and temperature. The lower temperature limit is -59.69°F.

Using the BWRS equation of state automatically initializes batch tracking and/or blending simulations. If you are modeling a constant composition, you will likely want to turn this off. For more information, see "NOTRACK" on page 218.

Component names

Key name	Actual name
C1	Methane
C2	Ethane
C3	Propane
IC4	Isobutane
NC4	n-Butane
IC5	Isopentane
NC5	n-Pentane
NC6	n-Hexane
NC7	n-Heptane
NC8	n-Octane
NC9	n-Nonane
NC10	n-Decane
NC11	n-Undecane
ETYL	Ethylene
PPYL	Propylene
H2	Hydrogen
HE	Helium
O2	Oxygen
N2	Nitrogen
CO2	Carbon Dioxide
H2S	Hydrogen Sulfide
BENZ	Benzene
TOLU	Toluene
NC12	n-Dodecane
NC13	n-Tridecane
NC14	n-Tetradecane
NC15	n-Pentadecane
NC16	n-Hexadecane
NC17	n-Heptadecane
NC18	n-Octadecane

Key name	Actual name
HVY1	Composite of heavy ends
HVY2	Composite of heavy ends
H2O	Water

Properties of the heavy ends

	HVY1	HVY2
Critical temperatures	1002.4	1279.4
Critical pressures	143.0	108.0
Molecular weights	352.7	478.9
Acentric factors	0.8	0.825
Critical densities	0.0431	0.0318
Heating values	0.0	0.0
Interaction coefficients	0.0	0.0

Take note

Viscosity equation

The viscosity, μ, of the fluid (at P,T) is given by:

$$\mu = \mu_0 \exp (\text{VPMI} \bullet \Delta P + \text{VTMI} \bullet \Delta T)$$

where:

μ_0 is the base viscosity, i.e., μ at custody temperature and pressure.

Viscosity mixing rule

The viscosity of a fluid mixture is calculated by the foregoing formula with $\log(\mu_0)$ evaluated as the weight-fraction weighted average of the logarithms of the values of μ_0 for each component, and with the VPMI and VTMI coefficients taken as the weight-fraction weighted average of the respective coefficients for each component.

Units for composition

You may choose units for composition, including fraction, percent, or parts per million. For more information, see "Units for composition" on page 130.

Example input

Example 1

Specify BWRS input for a gas with the following composition:

Gas	Composition
Methane	85% by mass
Ethane	7% by mass
Propane	5% by mass
Nitrogen	3% by mass

```
STATE BWRS
+ NAMES    C1     C2     C3     N2
+ INITIAL  0.85    0.07   0.05   0.03
```

If the composition was given in mole fraction instead of mass fraction the input is:

```
STATE BWRS.MOLE
+ NAMES    C1     C2     C3     N2
+ INITIAL  0.85    0.07   0.05   0.03
```

Example 2

Specify BWRS input for dense-phase carbon dioxide with custody conditions of 60°F and 2000 psia. These coefficients work well for temperature range of 40°F to 100°F, and a pressure range of 1500 to 3000 psig.

```
CUSTODY PRESSURE 2000, TEMPERATURE 60
STATE BWRS
+ NAMES       CO2
+ INITIAL     1
+ VISCOSITY   0.0966354
+ VPMI        1.46992E-4
+ VTMI       -1.07794E-2
```

STATE CNGA

INPREP

```
STATE CNGA SG [HHV]
```

Field	Units Key	Description
SG	n/a	Gas specific gravity (air = 1.0)
HHV	HEAT.VALUE	Higher heating value {1100 btu/ft^3}.

Description

The CNGA equation of state is a gas equation of state developed for natural gas and may be used to track specific gravity and heating values.

When composition tracking is enabled, you can set the inflowing SG and/or HHV at externals and/or nodes using the SSG and SHHV attributes, respectively.

This equation of state works well for the range of pressures and temperatures typical for gas transmission systems. To use this equation of state, only the gas specific gravity is needed.

Take note

Equations

$$Z = 1 / Y$$

$$Y = 1 + (344000 \text{ P } 10^{(1.785 \text{ x SG})}) / T^{3.825}$$

where:

Z	=	Gas compressibility
P	=	Pressure (psia)
T	=	Temperature (°R)
SG	=	Gas specific gravity (Air = 1.0)

The previous equation is dimensional. Use of different user units for pressure and temperature does not affect this equation of state.

Use of NOTRACK

Using the CNGA equation of state automatically initializes batch tracking and/or blending simulations. If neither tracking nor blending is needed, these features can be turned off by entering NOTRACK after the CNGA input. See “NOTRACK” on page 218. Tracking and blending slow the simulation and should be disabled if not needed.

Specifying SG and HHV

SG and HHV can be specified at externals and nodes (via SSG and SHHV) and on load.input. See “Setting composition at inflows” on page 139 for more information. The lower heating value, LHV, is calculated and is available for viewing, but cannot be adjusted directly. Adjust HHV instead.

Example input

Specify the CNGA equation of state for a gas with a specific gravity of 0.60:

```
STATE CNGA 0.60
```

STATE SCL

INPREP

```
STATE SCL[PROP] [TRIVC] [TRIVB] [ASTM]
+ FLUID FNAME
+ DENSITY P0 T0 ρ0 [PM0] [TM0] [PTMULT] [PPMULT] [TTMULT]
+ VISC (μ0 [VPMI] [VTMI]) | (A  B)
[+ HCAP Cv0 [CvT]]
[+ HCOND K0 [KT]]
[+ BING S0 SPC STC]
[+ VAPOR.PRESSURE VP1 [T1 VP2 T2]]
[+ DRA.DR dr1 ... drn]
[+ DRA.CON con1 ... conn]
[+ DRA.DROP pd]
[+ COLOR color]
[+ MIXWT mxwt]
```

Field	Units Key	Description
PROP	n/a	See "Batch tracking" under "Take note" for the differences between SCL and SCLPROP
TRIVC	n/a	Mass concentration below which you wish the fluid in question to be dropped from consideration in a blended mixture {0.005}
TRIVB	VOLUME	Batch size below which you wish the batch to be dropped from consideration {.5 MBBLS}
ASTM	n/a	ASTM or none. If present, SPS will use the ASTM form of viscosity {not active}
FNAME	n/a	Name for this fluid
P_0	PRESSUREabs	Base pressure of this fluid at T_0 {custody conditions}
T_0	TEMPERATURE	Base temperature of this fluid {custody conditions}
ρ_0	DENSITY	Base density of this fluid, i.e., density at P_0 and T_0
PM_0	BULK.MOD	Bulk modulus at P_0 and T_0, which is the density multiplied by the derivative of pressure with respect to density at constant temperature {270,000 psi}
TM_0	ΔTEMPERATURE	Temperature modulus at P_0 and T_0, which is the density multiplied by the derivative of temperature with respect to density at constant pressure {-2900°F}
PTMULT	n/a	Multiplier for the ΔPΔT term. This coefficient can be computed from the derivative of the bulk modulus with respect to temperature {-9.7}

Field	Units Key	Description
PPMULT	n/a	Multiplier for the $(\Delta P)^2$ term. This coefficient is half of the sum of one plus the derivative of the bulk modulus with respect to pressure at constant temperature and at base conditions {9.5}
TTMULT	n/a	Multiplier for the $(\Delta T)^2$ term. This term is half the sum of one plus the derivative of the temperature modulus with respect to temperature at constant pressure and at base conditions (P_0,T_0) {1.15}
μ_0	VISCOSITY	Viscosity at P_0 and T_0
VPMI	$1/\Delta P$	Pressure coefficient of viscosity {0/psi}
VTMI	$1/\Delta T$	Temperature coefficient of viscosity {-0.015/°F}
A	n/a	Constant for ASTM formula of viscosity {18}
B	n/a	Constant for ASTM formula of viscosity {7}
C_{v0}	HEAT.CAPACITY	Heat capacity at constant density {1 btu/(lbm°F)} This value overrides the SPH value entered with TRANSTHERMAL when calculating the temperature difference across a pump.
C_{vT}	HEAT.CAPA/ΔT	Temperature coefficient of heat capacity {0 btu/(lbm°F^2)}
K_0	HEAT.COND	Heat conductivity at T_0 {0.343 btu/(hr-ft°F)}
KT	HEAT.COND/ΔT	Temperature coefficient of heat conductivity {0 btu/(hrft°F^2)}
S_0	SHEAR.STRESS	Limiting shear stress at P_0 and T_0 {0}
SPC	SHEAR.STRESS/ΔP	Pressure coefficient of the limiting shear stress {0}
STC	SHEAR.STRESS/ΔT	Temperature coefficient of the limiting shear stress {0}
VP_1	$PRESSURE_{abs}$	Vapor pressure of the fluid at temperature T_1{0.2563 psia}
T_1	TEMPERATURE	Temperature corresponding to the vapor pressure VP_1 {60°F}
VP_2	$PRESSURE_{abs}$	Vapor pressure of the fluid at temperature T_2 {0.9492 psia}
T_2	TEMPERATURE	Temperature corresponding to the vapor pressure VP_2 {100°F}
dr	n/a	Drag reduction corresponding to the specified DRA concentration (values 0-1 are accepted). Required for DRA modeling.
con	COMPOSITION.DRA	Concentration of the drag reduction agent corresponding to the specified drag reduction. Values must be monotonically increasing. Required for DRA modeling.
pd	ΔPRESSURE	Pressure drop through valves at which the DRA concentration is reduced to zero. {20}

Field	Units Key	Description
color	n/a	Batch bar color to be used on distance plots. Color choices are: blue, red, green, cyan, magenta, yellow, orange, dkgreen, brown, dkblue, pink, dkgray, and ltgray.
mxwt	n/a	Mixing weight. During mixing the weight multiplied by the mass fraction of the fluids are compared; the highest value determines the batch label. {1}

Description

The slightly compressible liquid (SCL) equation of state may be used for multi-fluid liquid simulations for which one or more fluids and their corresponding properties are defined. The STATE SCL is suitable for batched and/or product liquid pipelines. The SCL equation of state also supports non-Newtonian Bingham Plastic flow. For more information, see "VISCOSITY (non-Newtonian)" on page 239.

STATE SCL defines one or more liquids and their corresponding properties. STATE SCL assumes that each fluid may be described by a simple quadratic equation of state with simple blending rules. If STATE SCL is present, one or more FLUID commands must follow to describe the fluid properties of all the fluids of interest.

Take note

Initialization

The model is initialized with the fluid defined on the first FLUID command. See "LINE.FILL" on page 465 or "LOAD.INPUT" on page 469 to override with an initial batch alignment. .

Use of NOTRACK

Using the SCL equation of state automatically initializes batch tracking and/or blending simulations. If neither tracking nor blending is needed, these features can be turned off by entering NOTRACK after the SCL input. See "NOTRACK" on page 218. Tracking and blending slow the simulation and should be disabled if not needed. NOTRACK may not be used with SCLPROP.

Batch tracking

As fluids enter the system they form batches. The batches move continuously at the pipeline flow velocity. The SCL equation of state does not model diffusion; the SCLPROP equation of state does model diffusion.

The SCLPROP diffusion is based on the second difference in density. Therefore, the majority of the diffusion occurs at the start of the batch where the density difference is sharpest, but some diffusion occurs along the entire length of the piping system. The DIFFUSE variable on the SHOW GLOBALS display may be tuned to match actual pipeline diffusion by increasing or decreasing the amount of diffusion.

Diffusion

This parameter is a multiplier on the rate of diffusion. (Diffusion is modeled only for SCLPROP).

Diffusion is calculated as follows:

$$dv / dt = coef \bullet d^2v / dx^2$$

where:

ν	=	Specific volume of mixture (based on base densities)
x	=	Distance
coef	=	DIFFUSE * 0.13 * diameter * MAX(|V-|, |V+|)
t	=	Time

This equation is satisfied for each fluid mixture vector (FMV). The effect of this equation is that the rate of diffusion decays exponentially with distance (or time at constant velocity).

The initial value of 1 produces some diffusion. To increase the diffusion rate, set the GLOBAL peek DIFFUSE to a value greater than 1. To decrease the diffusion rate, set DIFFUSE to a value less than 1. DIFFUSE = 0 implies no diffusion.

Compare the actual pipeline diffusion (by looking at densitometer data) with the modeled diffusion, and tune DIFFUSE so that the modeled diffusion rate agrees with the actual rate.

Also consider the TRIVC value from the SCLPROP command when tuning DIFFUSE. Decreasing TRIVC reduces diffusion and increasing TRIVC increases diffusion.

Blending

During the simulation, streams of fluids may blend (for example, at a pipe Y-junction). The specific volume of the mixed fluids is computed as the weighted average of the specific volumes of the fluids present. The logarithm of the viscosity of a fluid mixture is computed as the weighted average of the logarithms of the viscosities of the fluids present.

Output

The batch profile can be displayed using interactive TRANS. The concentration of a particular fluid may be sensed using a named-fluid sensor, NFLD. The density of the mixture may be determined using a density sensor, DENS. Density is also reported at nodes.

Drag reduction agent (DRA)

A special fluid called DRA is defined if one or more fluids have the DRA.DR and DRA.CON input. The fluid name DRA is reserved for this purpose. This special DRA fluid has the bulk properties (P_0, T_0, ρ_0, etc.) of the first fluid in your fluid table. The DRA fluid can be injected either with another fluid or into a flowing stream. The effective friction factor is reduced by the DRA.DR table value corresponding to the DRA concentration. See the “Drag reduction” under “Take note” for details. DRA can be injected at any point along the model.

DRA is typically injected into the model using a TAKE external that is configured to inject 100% DRA (external:SDRA=1) at the desired flow rate. The DRA is typically injected in trace amounts, at 20 to 40 PPM. The flow set point (external:SQ) can be set up to use a relatively small flow unit, such as GPM. The flow rate for the DRA external can be calculated by summing the non-DRA flows into the node, appropriately converting units, and then taking the desired fraction.

The DRA effect is reduced to 0 whenever the fluid flows through a running pump or a valve with a pressure drop greater than pd. To model partial DRA degradation, reinject DRA downstream of these elements. Because the effect of shear, time, and the distance on DRA degradation is not well known, these effects are ignored.

FLUID

FLUID is used to specify the names and properties of the various fluids. Any number of fluids may be defined. If a default fluid property is to be used, enter a zero. If the default values for all items in the second line of the FLUID command are to be used, the second line of the FLUID command may be omitted. The + sign is a continuation symbol and must be the first character on the line if input for the line is to be entered.

Specific volume equation

The specific volume, V = 1/ρ, of the fluid at (P,T), i.e., at pressure P and temperature T, is given by:

$$V = V_0(1 - \Delta P / PM_0 - \Delta T / TM_0 - \alpha\Delta P\Delta T / (PM_0TM_0) + \eta(\Delta P)^2 / (PM_0)^2 + \nu(\Delta T^2) / (TM)^2)$$

where:

V_0	=	The base specific volume of the fluid, i.e., at (P_0,T_0)
ΔP	=	P - P_o
ΔT	=	T - T_o
α	=	PTMULT = -1 - (PM_T* TM_0/PM_0)
η	=	PPMULT = (1 + PM_p)/2
ν	=	TTMULT = (1 + TM_T)/2
P_0	=	Base pressure
T_0	=	Base temperature
ρ_0	=	Density at P_0 and T_0
PM_0	=	Bulk modulus at P_0 and T_0
TM_0	=	Temperature modulus at P_0 and T_0
PM_T	=	The partial derivative of bulk modulus with respect to temperature at (P_0, T_0)
PM_P	=	The partial derivative of bulk modulus with respect to pressure at (P_0, T_0)
TM_T	=	The partial derivative of temperature modulus with respect to temperature at (P_0,T_0)

Note: The temperature modulus TM is -1/β, where β is the coefficient of volumetric expansion that appears explicitly in the definition of the GRASHOF number. (See "GRASHOF" under "Take note" in "TRANSTHERMAL" on page 249). The term TMT is thus βT/β02, and ν (TTMULT) is then (β02 + βT)/(2β02), where βT is the partial derivative of β with respect to temperature at constant pressure at (P_0,T_0).

Specific volume mixing rule

The specific volume of a fluid mixture is calculated by the foregoing formula for V with the coefficients V_0, -1/PM_0, -1/TM_0, $\alpha/(PM_0TM_0)$, $\eta/(PM_0)^2$ and $\nu/(TM_0)^2$ computed as the weight-fraction average of the respective coefficients for each component of the mixture.

Density equation

The density, ρ, of the fluid at pressure P and temperature T is calculated from the specific volume V, by ρ = 1/V.

Viscosity equation

The viscosity, μ, of the fluid (at P,T) is given by:

$$\mu = \mu_0 \exp (VPMI \bullet \Delta P + VTMI \bullet \Delta T)$$

where:

μ_0 is the base viscosity, i.e., μ at (P_0,T_0).

When ASTM is entered on the SCL line, the API formula is used to calculate viscosity:

$$\log_{10} \log_{10} (\nu + 0.7) = A - B \log_{10} T$$

$$\nu = \mu / \rho$$

where:

ν	=	Kinematic viscosity (in cSt or mm^2/s)
μ	=	Absolute viscosity (in cp)
ρ	=	Fluid density (in g/cm^3)
T	=	The temperature (in °R with ENGLISH, in °K with METRIC)
ΔP	=	$P - P_o$
ΔT	=	$T - T_o$

*** Reference: ASTM D341-87

Viscosity mixing rule

The viscosity of a fluid mixture is calculated by the foregoing formula with $\log(\mu_0)$ evaluated as the weight-fraction weighted average of the logarithms of the values of μ_0 for each component, and with the VPMI and VTMI coefficients taken as the weight-fraction weighted average of the respective coefficients for each component.

When ASTM is entered on the SCL line, viscosity of the mixture is calculated by A and B taken as the weight-fraction weighted average of the respective A and B for each component.

Heat capacity equation

The heat capacity of the fluid at constant volume, C_v, and at temperature, T, is given by:

$$C_v = C_{vo} + C_{vT} \bullet \Delta T$$

In consistent units, you can compute C_v from C_p with:

$$C_v = C_p - \left(\frac{T_{abs} PM}{TM^2 \rho} \right)$$

where:

C_P is the heat capacity at constant pressure

T_{abs} is the absolute temperature

ΔT is $T-T_o$

Heat conductivity equation

The heat conductivity, K, of the fluid (at T) is given by:

$$K = K_0 + KT \bullet \Delta T$$

where:

K_0 is the base heat conductivity, i.e., K at (P_0, T_0)

$\Delta T = (T - T_0)$

Bingham Plastics definition

An extension of the SCL viscosity definition allows the use of one or more Bingham Plastics for the flowing fluid. To model the performance of Bingham plastics in pipelines, define the limiting shear stress, τ_0, that the plastic can support at zero shear rate as an affine function of pressure and temperature.

The shear stress, τ, is given by:

$$\tau = \tau_0 + u * \mu \infty$$

where:

τ_0	=	Limiting shear stress (0 for Newtonian fluids)
u	=	Shear rate
$\mu \infty$	=	Limiting viscosity for the plastic (this is the same as the viscosity μ defined by the coefficients μ_0, VPMI, and VTMI)

The function τ_0 is given by:

$$\tau_0 = \text{Max}\ (0, S_0 + SPC \bullet \Delta P + STC \bullet \Delta T)$$

where:

ΔP	=	$P - P_o$
ΔT	=	$T - T_o$

If the polynomial evaluates τ_0 to be negative, then τ_0 is taken to be zero. The $\mu \infty$ for a mixture of Bingham plastics is approximated by using the mass-weighted log average of the corresponding values, and τ_0 is taken to be the mass average.

The friction factor approximation used is the one presented by Darby and Melson in *Chemical Engineering*, December 29, 1981. However, a small creep speed is defined in the program, and below the creep speed the friction factor is held constant. The friction factor used is independent of the pipe roughness and if τ_0 is zero the friction corresponds to a smooth pipe case.

Also, if the calculated Bingham Plastic friction is lower than the Newtonian friction, the software uses the Newtonian friction.

Non-Newtonian viscosity

If you are using STATE SCL, you can model non-Newtonian rheology using a viscosity table. For more information, see "VISCOSITY (non-Newtonian)" on page 239.

Wax deposition

If you are using STATE SCL, you can model wax deposition. For more information, see "WAX" on page 242.

Vapor pressure equation

The vapor pressure of a fluid is calculated at varying temperatures, using the latent heat of vaporization:

$$\Delta H \Big/ (R_{const} / M) = \left(\log \frac{P_1}{P_2} \right) \Big/ \left((T_1 - T_2) \Big/ (T_1 \times T_2) \right)$$

where:

P_1	=	Vapor pressure at T_1 (absolute units)
P_2	=	Vapor pressure at T_2 (absolute units)
ΔH	=	Latent heat of vaporization
R_{const}	=	Gas constant
T_1	=	Temperature at which P_1 is measured (absolute units)
T_2	=	Temperature at which P_2 is measured (absolute units)

For SCLPROP, the vapor pressure corresponding to the flowing temperature is calculated. The defaults are based on water. Vapor pressures are mixed volumetrically.

Drag reduction

Drag reduction is calculated by pipe interval. The effective friction factor for a pipe interval, after drag reduction, is calculated as

$$FFE = FF \bullet (1 - DR) \bullet (1 + FC)$$

where:

FF	=	Friction factor
DR	=	Drag reduction
FC	=	Independent friction correction tuning parameter

The :FFE peek is the length-average of the internally calculated FFEs.

DR is determined from the user-defined table for the fluid and the specified concentration of DRA. Because the amount of drag reduction varies by individual fluid, each fluid should have its own unique DRA.DR and DRA.CON entries.

The DRA.DR and DRA.CON table values are used to compute A and B coefficients that provide a least-squares fit to:

$$1/DR = 1 / (A \bullet CON) + B$$

If mixing occurs, the drag reduction coefficients A, B, and PD, as well as the DRA concentration itself, are mixed linearly in proportion to the flow rate. Subsequent DRA calculations are made using the mixed values of the coefficients and DRA concentration.

Tracking fluid properties differences in SCL and SCLPROP

You can track fluid properties of fluid names by specifying STATE SCLPROP. While SCL tracks fluid batches through the pipeline according to the fluid name, SCLPROP generally tracks fluid batches through the pipeline via fluid properties. Therefore, SCL is better for batch tracking operations when using assumed fluid properties, while SCLPROP is more useful when knowledge of fluid properties is most important.

Both forms of the equation of state track fluid information using FMVs (fluid mixture vectors). The SCL FMV has an entry for every user-defined fluid and tracks the mass fraction of each fluid. Fluid properties are determined by multiplying each user-defined fluid property by the mass fraction of each fluid and summing the total. The SCLPROP FMV has an entry for each individual fluid property.

Example input

Specify the SCL equation of state input for two different crude oils:

```
STATE SCL
+ FLUID CRUDE1
+ DENSITY 14.696  60  58.2  270000
+ VISC 3.78
+ FLUID CRUDE2
+ DENSITY 14.696  59  59.1 290200
+ VISC 4.01
+ HCAP 0.996
+ DRA.DR 0 .12 .22 .25 .32 .38
+ DRA.CON 0 20 40 60 80 100
```

Because some of the fluid properties (such as the temperature coefficient of viscosity) are not entered, the default values are used.

The names entered for the fluids (CRUDE1, CRUDE2, etc.) are used to identify fluid batches when the batch tracking profile option is activated.

Note: To have the effects of DRA modeled, a fluid named DRA must be injected at an external. DRA will mix with both fluids but only CRUDE2 will be affected by the DRA.

VISCOSITY (non-Newtonian)

INPREP

Viscosity may be expressed using SPS's CSV input format, which was introduced in SPS version 9.6. For more information on this format, see "CSV input syntax" on page 52.

```
VISCOSITY
+ FLUID <name1>
+ TABLE
{
VISCOSITY, SHEAR.RATE, TEMPERATURE
v1, s1, t1
v2, s2, t2
...
vn, sn, tn
}

...

+ FLUID <namei>
+ TABLE
{
VISCOSITY, SHEAR.RATE, TEMPERATURE
v1, s1, t1
v2, s2, t2
...
vn, sn, tn
}
```

Viscosity may also be expressed in the following format:

```
VISCOSITY
+ FLUID name1
+ V(T,S)    s1      s2   ...   sm
+    t1      v11     v12        v1m
+    t2      v21     v22        v2m
+    ...
+    tn      vn1     vn2        vnm

...

+ FLUID namei
+ V(T,S)    s1      s2   ...   sm
+    t1      v11     v12        v1m
+    t2      v21     v22        v2m
+    ...
+    tn      vn1     vn2        vnm
+ ...
```

Viscosity may alternately be expressed as follows:

```
VISCOSITY
+ FLUID name1
+ TABLE   VISCOSITY    SHEAR.RATE   TEMPERATURE
+            v1            s1           t1
+            v2            s2           t2
+           ...
+            vn            sn           tn

...

+ FLUID namei
+ TABLE   VISCOSITY    SHEAR.RATE   TEMPERATURE
+            v1            s1           t1
+            v2            s2           t2
+           ...
+            vn            sn           tn
+ ...
```

Field	Units Key	Description
name1...namei	n/a	Name of the fluid
v1...vn	VISCOSITY	Viscosity when t1...tn and s1...sn
s1...sn	SHEAR.RATE	Shear rate when t1...tn and v1...vn
t1...tn	TEMPERATURE	Temperature when s1...sn and v1...vn
t1...tn	TEMPERATURE	Temperature
s1...sm	SHEAR.RATE	Shear rate
v11...vnm	VISCOSITY	Viscosity at t1...tn and s1...sm

Description

The viscosity table is used to model non-Newtonian rheology, specifically, when the viscosity is a function of shear rate and temperature. The slightly compressible liquid equation of state is required in order to use the viscosity table. For more information, see “STATE SCL” on page 230.

Three input styles are available. You may use a different input format for each fluid that you define. When using the first style, the V(T,S) can be entered as V(S,T), in which case each data column is for a particular temperature and each data row is for a particular shear rate. When using the second and third styles, the VISCOSITY, SHEAR.RATE, and TEMPERATURE columns can be in any order.

Take note

Viscosity data for SCL still required

Even if you enter data in the VISCOSITY table, you still must provide viscosity data in the STATE SCL. The STATE SCL has other uses, such as pump viscosity correction, viscosity sensor, and the reporting of viscosity at an external. For more information, see “STATE SCL” on page 230.

How to enter in Model Builder

If you want to enter viscosity data in Model Builder, on the model explorer, open the **Options** folder and then the **Fluids** folder. Double-click on a fluid name. In the fluid editor, click the **Edit** button next to **VISC TBL**. For more information on editing the table, see "Fluids editor" on page 709 and "Viscosity table editor" on page 709.

Example input

Example 1

Enter a table of viscosity as a function of temperature and shear rate for a fluid named AFLU. The temperature ranges from 5 °C to 35 °C and the shear rate is 2, 6, 10, 20, 60, and 120 reciprocal seconds.

```
VISCOSITY
+ FLUID AFLU /* shear rate -->
+ V(T,S)   2      6      10     20     60    120
/*         ----   ----   ----   ----   ----   ----
+ 5       85.6   63.7   53.8   46.8   44.1   41.1
+ 10      55.5   39.7   34.8   31.2   29.7   28.1
+ 15      34.7   26.7   24.2   22.2   21.4   20.5
+ 20      21.4   19     17.7   16.6   16.1   15.6
+ 25      13.9   14.1   13.4   12.9   12.6   12.3
+ 30      10.3   10.8   10.5   10.3   10.1    9.9
+ 35       8.9    8.5    8.5    8.4    8.3    8.2
```

WAX

INPREP

```
WAX MXR HCON [DENS] [MXT]
+ TABLE DEPOSITION WALLTEMP VELOCITY
```

Field	Units Key	Description
MXR	ROUGHNESS	The maximum pipe wall roughness of pipe coated with deposited wax.
HCON	HEAT.COND	Heat conductivity of the wax.
DENS	DENSITY	Effective density of the deposited wax {56 lbm/ft^3}
MXT	ROUGHNESS	Maximum wall thickness. {0.25 inches} The specified value will be flagged if it is greater than 10 percent of the pipe diameter.
DEPOSITION	DEPOSITION	The wax deposition tendency at the corresponding wall temperature and velocity. (Must be positive.)
WALLTEMP	TEMPERATURE	The wall temperature at which the wax deposition was measured. Temperature should be below cloud point.
VELOCITY	VELOCITY	The fluid velocity at which the wax deposition was measured.

Description

Wax deposition simulates the formation of wax on the pipe wall due to heat transfer. The heat conductivity of the wax layer will be included in the heat transfer calculations for transient thermal calculations. The heat flow through the wax layer is assumed to be steady-state.

All fluids are assumed to deposit wax in a given system, if conditions permit. The same wax deposition table is used for an entire piping system. If the fluid velocity or temperature is out of range, a constant extrapolation is used.

The wax deposition will reduce the cross-sectional area of the pipe and therefore reduce the inventory of any pipes with WAX deposits. The fluid velocity will be calculated on the reduced pipe cross-sectional area.

The slightly compressible liquid equation of state and the transient thermal mode are required for wax deposition to occur. For more information, see “STATE SCL” on page 230 and “TRANSTHERMAL” on page 249.

Wax deposition calculations

The table indicates the wax deposition tendency at a given wall temperature and wall shear rate. The wax deposited is then a function of the wax deposition tendency and the heat flux at the pipe wall.

The adjustment to the heat transfer coefficient for wax buildup is:

$$\hat{h} = \frac{hk_{wax}}{k_{wax} + h\tau_n\alpha}$$

The wax temperature is calculated as:

$$T_{wax} = \frac{\tau_n \hat{h} \ \Delta \ T \ \alpha}{k_{wax}} + T_{wall}$$

The following formula is used to calculate wax deposit:

$$\tau_{n+1} = \tau_n + \frac{\hat{h}\Delta T \ f(T_{wax}, \bar{v})\Delta t}{\alpha\beta\rho_{wax}}$$

where:

τ_n	=	Wax buildup at knot at time step n (in)
h	=	Heat transfer coefficient at knot (btu/hr-ft^{2}•°F)
$\hat{h}$	=	Heat transfer coefficient with wax effects (btu/hr-ft-°F)
ΔT	=	Fluid - pipe wall temperature (°F)
T_{wax}	=	Wax temperature at knot (°F)
$\bar{v}$	=	Average cross-sectional velocity of the fluid at knot (ft/sec)
ρ	=	Density of the wax (lbm/ft^3)
f	=	User-defined wax deposition function (lbm/btu)
k_{wax}	=	Heat conductivity of the wax (btu/hr-ft-°F)
Δt	=	Time step (min)
T_{wall}	=	Temperature of the pipe wall at knot (°F)
α	=	Conversion factor 1/12 ft/in
β	=	Conversion factor 60 min/hr

The wax is distributed evenly around the pipe wall until thickness meets MXT.

If the heat flux becomes negative (pipe wall temperature is greater than the flowing fluid temperature), wax will be removed from the pipe wall based on the deposition table.

Because the shear stress in flowing pipelines is usually well below the shear stress required to move wax back into solution, this possible effect is not modeled. You may remove the WAX in a pipe by poking the WAX thickness to 0.

Frictional effects of wax buildup

As wax builds up on the pipe wall in accordance with the deposition rate formula, the diameter decreases and results in a loss of flow area. Based on the wax density, the current deposition, and the pipe diameter without wax buildup, SPS determines the fraction of flow area lost at each location in the pipe. The friction factor used locally will be determined by first calculating the effective friction factor from the Reynolds Number (computed using the wax-reduced diameter) and calculated pipe roughness (pipe roughness plus the wax deposition thickness bounded by the user-defined maximum roughness.)

Example input

Wax deposition occurs in a crude oil pipeline. The wax deposition parameters are:

```
/*   rrough hcond   den  mxt
WAX   1.0    0.1   57.0  0.5
```

```
+TABLE    WALLTEMP      DEPOSITION    VELOCITY
+          20              0.9           1
+          30              0.5           1
+          40              0.2           1
+          50              0.0           1
+          20              0.8           3
+          30              0.45          3
+          40              0.15          3
+          48              0.0           3
+          20              0.7           5
+          30              0.4           5
+          40              0.15          5
+          45              0.0           5
+          20              0.6           8
+          30              0.35          8
+          40              0.05          8
+          42              0.0           8
```

STATE TABLE

INPREP

The STATE TABLE input may be expressed in one of two different formats. The first format uses SPS's CSV input format, which was introduced in SPS version 9.6. For more information on this format, see "CSV input syntax" on page 52.

```
STATE TABLE
+ VAPOR.PRESSURE VP
+ SPECIFIC.HEAT SPH
{
PRES, TEMP, DENS, VISC
PRES1 TEMP1 DENS1 VISC1
PRES2 TEMP2 DENS2 VISC2
...
PRESj TEMPj DENSj VISCj
}
```

Alternately, the STATE.TABLE input may be expressed as follows:

```
STATE TABLE
+ VAPOR.PRESSURE VP
+ SPECIFIC.HEAT SPH
+ TABLE PRES TEMP DENS VISC
+       PRES1 TEMP1 DENS1 VISC1
+       PRES2 TEMP2 DENS2 VISC2
+       ...
+       PRESj TEMPj DENSj VISCj
```

Field	Units Key	Description
VP	$PRESSURE_{abs}$	Vapor pressure {1 psia}
SPH	n/a	Specific heat (heat capacity/heat capacity of water) {1}
$PRES_j$	PRESSURE	Table pressure value
$TEMP_j$	TEMPERATURE	Table temperature value
$DENS_j$	DENSITY	Table density value
$VISC_j$	VISCOSITY	Table viscosity value

Description

The thermal equation of state by regression (STATE TABLE) may be used for single-fluid liquid simulations in which temperature effects are significant. This equation of state allows you to enter a table of density and viscosity as a function of pressure and temperature.

The STATE TABLE line must be followed by a table of density (ρ) and viscosity (μ) corresponding to pressure (P) and temperature (T) expected during the simulation. In addition, the table must include:

- At least four distinct pressures per temperature value
- At least five distinct temperatures

Within each temperature value, as the pressure increases the density and viscosity also increase. As the temperature value increases, the highest values of the density and viscosity should be lower than the previous temperature's lowest density and viscosity values, respectively. The maximum number of table entries is 400.

Take note

Density equation

The density (ρ) is approximated by:

$$\rho = a_0 + a_1T + a_2T^2 + a_3P + a_4T\,P + a_5P^2 + a_6T^3 + a_7T^2P + a_8T\,P^2$$

where the coefficients a_0, a_1, ... , a_8 are determined to produce the best "least squares" curve fit for the table data.

Viscosity equation

The viscosity (μ) is approximated by:

$$\log \mu = b_0 + b_1T + b_2P + b_3T^2 + b_4T\,P + b_5P^2 + b_6T^3 + b_7T^2P + b_8T\,P^2$$

where the coefficients b_0, b_1, ... , b_8 are determined to produce the best "least-squares" curve fit for the table data.

Example input

Enter state table data input for an oil:

```
STATE TABLE
+ VAPOR.PRESSURE  1
+ SPECIFIC.HEAT  1
+ TABLE PRES TEMP DENS VISC
+ 500   40  48.918  11.206
+ 900   40  49.035  11.742
+ 1300  40  49.150  12.215
+ 1700  40  49.341  12.865
+ 500   50  47.021  9.954
+ 900   50  47.444  10.111
+ 1300  50  47.856  10.342
+ 1700  50  48.012  10.956
    .
    .
    .
+ 500   70  44.326  7.876
+ 900   70  44.768  8.012
+ 1300  70  44.998  8.231
+ 1700  70  45.109  8.509
```

ISOTHERMAL

INPREP

```
ISOTHERMAL [TEMP]
```

Field	Units Key	Description
TEMP	TEMPERATURE	Default temperature {70°F}

Description

In isothermal mode, temperatures are specified by you and not calculated by SPS. The pipe-end temperatures that you enter are interpolated and used to calculate density, velocity, and other parameters. Temperature mixing is not done, and fluid heating/cooling due to compression/expansion is not modeled. Isothermal mode is generally sufficient for many simulations.

Many systems can be modeled accurately using the isothermal mode. Using the isothermal mode simplifies input and decreases running time.

Example input

Model a network in a temperature-independent way with a fluid temperature of 60°F:

```
ISOTHERMAL 60
```

THERMAL

INPREP

```
THERMAL [TEMP]
```

Field	Units Key	Description
TEMP	TEMPERATURE	Default temperature {70°F}

Description

In thermal mode, you enter some temperatures, while others are calculated by SPS. For example, you enter pipe-end temperatures and inlets. On all elements except transfer lines, heating/cooling due to compression/expansion is modeled. Transfer lines use a linear interpolation of user-supplied pipe-end temperatures. Temperature mixing at stations is done, stopping at transfer lines. A fluid gets the specified pipe temperature when it enters the transfer line

Thermal mode is typically used when thermal effects at the stations are important and the stations are far enough apart that the temperature stabilizes between stations, such that you can input a value.

With the thermal mode the temperatures at the ends of all hydraulic elements are displayed as part of its normal output. The thermal mode may be used to study the temperature dynamics of pump or compressor stations.

Example input

Model a network with temperature data calculated as variable in equipment other than pipes:

```
THERMAL
```

TRANSTHERMAL

Peek and Poke Attributes

INPREP

```
TRANSTHERMAL [TEMP]
[+ SP.HEAT SPH]
[+ HEATCOND HCOND]
[+ COLBURN Ac Bc Cc]
[+ GRASHOF Ag Bg1 Bg2 Cg Dg Eg Fg Gg Hg]
[+ CG.TRANS Re1 Re2]
[+ MIN.FILM.COEF MFC]
[+ HEAT.FRIC.EXP HFELH HFELC HFETH HFETC]
[+ FORCED.CONVECTION FORCED]
[+ FREE.CONVECTION FREE]
```

Field	Units Key	Description
TEMP	TEMPERATURE	Default temperature {70°F}
SPH	n/a	Ratio of the heat capacity at constant volume of the fluid to the heat capacity at constant pressure of water {1}. This entry applies to the CNGA and TABLE equations of state. This line should not be entered if the AGA, BWRS, SCL, or SCLPROP equation of state is used, because these equations of state calculate SP.HEAT.
HCOND	HEAT.COND	Thermal conductivity of fluid. This value may be superseded by values entered for an equation of state. {0.343 btu/(hr-ft-°F). This entry applies to the AGA, BWRS, CNGA, and TABLE equations of state. This line should not be entered if the SCL or SCLPROP equation of state is used, because these equations of state calculate HCOND.
Ac	n/a	Scale coefficient in Colburn term. {0.023}
Bc	n/a	Reynolds number exponent in Colburn term. {0.8}
Cc	n/a	Prandtl number exponent in Colburn term. {0.3333}
Ag	n/a	Scale coefficient in laminar term. {1.0}
Bg_1	n/a	Viscosity ratio exponent for increasing temperature (heating). {.14—Sutherland's constant}
Bg_2	n/a	Viscosity ratio exponent for decreasing temperature (cooling) {.14—Sutherland's constant}
Cg	n/a	Scale coefficient representing effective d/L value {0} (Recall $0^0 = 1$)
Dg	n/a	Reynolds x Prandtl number exponent in Grashof term {1}

Field	Units Key	Description
Eg	n/a	Coefficient in laminar term {.01}
Fg	n/a	Exponent of Grashof x Prandtl number {.75}
Gg	n/a	Exponent of d/L {0}
Hg	n/a	Exponent in laminar term {.33}
Re_1	n/a	The Reynolds number separating laminar-dominated (Grashof) from the transition region for heat transfer {2100}
Re_2	n/a	The Reynolds number separating the transition region from the turbulent (Colburn) region for heat transfer {10000}
MFC	HEAT.TRANSFER	Minimum film coefficient, h {0}
HFE	n/a	Heat friction exponent used for friction correction ratio {0}
FORCED	n/a	WALL.TEMP or MEAN.TEMP in forced convection term {MEAN.TEMP}
FREE	n/a	WALL.TEMP or MEAN.TEMP in free convection term {MEAN.TEMP}

Description

In transient thermal mode, most temperatures are calculated by SPS. You enter the ground temperature, as a function of location, and the inlet fluid temperature, as a function of location and time. Afterwards, SPS models heating/cooling due to compression/expansion for all elements. In addition, the heat interchange between fluids, pipe walls, insulation, and the ground is modeled.

Transient thermal mode is typically used in liquid systems when there is a strong temperature dependency on viscosity, such as with a batched line that transports heavy crudes. Also, it is used for gas systems when temperature effects are important for station operation and the stations are too close together for the temperature changes to settle out between stations.

Take note

Steady-state temperature profile

Use care in relying on TRANSTHERMAL simulation results before the piping system has reached a radial steady-state temperature with the ground. The process of reaching a radial steady state can be expedited by poking TT.RESET to a non-zero value (see “TRANSTHERMAL (TR) attributes” on page 694. This forces a radial steady-state temperature solution.

When TT.RESET is non-zero, an overall heat transfer coefficient between the fluid and the outer ground is calculated at every pipe knot assuming steady-state radial heat flux. In effect, the layers between the fluid and outer ground are treated as though they have no heat capacity. The radial temperature profile between the fluid and the outer ground at each pipe knot is updated each time step with the steady-state solution of the radial heat equation in the region surrounding the fluid.

Poking TT.RESET to a positive value produces a one time step radial steady-state solution. TT.RESET is then automatically reset to zero. Poking TT.RESET to a negative value forces a continuous radial steady-state solution for each time step until you poke TT.RESET to a positive value or zero.

Knot spacing

TRANSTHERMAL studies require smaller knot spacing to be accurate. For more information, see "Calculation intervals" on page 197.

COLBURN

COLBURN calculates the inside heat transfer coefficient between the fluid and the pipe wall in the turbulent-dominated flow region.

GRASHOF

GRASHOF calculates the inside heat transfer coefficient between the fluid and the pipe wall in the laminar-dominated flow region.

The heat transfer coefficient is deduced from the Nusselt number, which is calculated as follows:

$$Nu = \alpha \left(Ac \ Re^{Bc} \ Pr^{Cc} \right) +$$

$$(1- \alpha) Ag\left((\mu_m/\mu_w)(\rho_w/\rho_m)\right)^{Bgi} \left[\left((\pi/4)(Cg\ ) Re\ Pr\right)^{Dg} + Eg\left(Gr\ Pr\ \right)^{Fg} \left(Cg\ \right)^{Gg} \right]^{Hg}$$

where:

Nu	=	Nusselt number
α	=	Transition factor (1 = fully turbulent, 0 = fully laminar)
Re	=	Reynolds number
Pr	=	Prandtl number
Gr	=	Grashof number
A...H	=	Parameters from the COLBURN and GRASHOF lines
ρ	=	Fluid density
m	=	Fluid viscosity

The subscript w refers to values in the wall film. The subscript m refers to average values in the main stream.

The transition coefficient, α, is computed as follows from Re and C_G^{GC}.TRANS line entries Re_1 and Re_2:

$$\alpha = \begin{cases} 0 \quad for \quad Re \le Re_1 \\ \gamma(2-\gamma) \quad for \quad Re_1 < Re \le Re_2 \\ 1 \quad for \quad Re_2 < Re \end{cases}$$

where:

$$\gamma = (\ln Re - \ln Re_1) / (\ln Re_2 - \ln Re_1)$$

The quadratic interpolation with γ in the transition zone approximates the shape of the transitions described in the standard literature.

The Nusselt, Reynolds, Prandtl, and Grashof numbers are dimensionless, and in any system of units are defined as:

$$Nu = h\ d\ /\ k$$

$$Re = dv\ \rho\ /\ \mu$$

$$Pr = C_p \mu / k$$

$$Gr = d^3 \rho^2 g \beta \mid \Delta T \mid / \mu^2$$

where:

h	=	Pipe inner wall film coefficient (heat transfer coefficient between the pipe wall and fluid)
d	=	Pipe inside diameter
k	=	Heat conductivity of the fluid
v	=	Average velocity of the fluid over cross section
C_p	=	Heat capacity of fluid at constant pressure
g	=	Acceleration due to gravity
β	=	Coefficient of volumetric expansion (negative inverse of temperature modulus). See "STATE SCL" on page 230
ΔT	=	Temperature differential between fluid and pipe wall

The Reynolds and Prandtl numbers in the term with the Dg exponent may be evaluated at either the mean flowing temperature or the wall temperature. If the line +FORCED.CONVECTION WALL.TEMP appears, the temperature used is the wall temperature. Otherwise, by default, or if +FORCED.CONVECTION MEAN.TEMP appears, the mean temperature across the flowing section is used.

The Grashof and Prandtl numbers in the term with the Fg exponent may be evaluated at either the mean flowing temperature or the wall temperature. If the line +FREE.CONVECTION WALL.TEMP appears, the temperature used is the wall temperature. Otherwise, by default, or if +FREE.CONVECTION MEAN.TEMP appears, the mean temperature across the flowing section is used.

In TRANSTHERMAL simulations a friction correction for heating or cooling is applied as:

$$f / f_{iso} = \left(\left(\mu_w / \mu_m \right) \left(\rho_m / \rho_w \right) \right)^{HFE}$$

where:

f	=	Friction factor used in the transient thermal case
f_{iso}	=	Friction factor calculated for isothermal conditions
HFE_{LH}	=	Value of HFE for heating in laminar flow region {0}
HFE_{LC}	=	Value of HFE for cooling in laminar flow region {0}
HFE_{TH}	=	Value of HFE for heating in turbulent flow region {0}
HFE_{TC}	=	Value of HFE for cooling in turbulent flow region {0}

Laminar flow is flow below Re_1; turbulent flow is flow above Re_2. For $Re_1 < Re < Re_2$, the value of HFE is interpolated with log Re between values for Re_1 and Re_2.

Heating is defined to occur if the wall temperature is higher than the average temperature of the main stream. Otherwise the cooling case applies.

Example input

Model a network tracking temperature response over time in pipes, other equipment, and the surrounding environment with the following input data:

- The BWRS or SCL equation of state is used.
- The thermal conductivity of the fluid is 0.03 btu/(hr-ft-F).
- The turbulence-dominated scale coefficient is 0.023.
- The Reynolds number exponent is 0.8.
- The Turbulence-dominated Prandtl number exponent is 0.3333.
- The laminar-flow-dominated scale coefficient is 0.5 (Ag).
- The viscosity ratio exponent is 0.15 for both heating and cooling (Bg1 and Bg2).
- The effective d/L ratio is .001 (Cg).
- The RePr exponent is 1.0 (Dg).
- The GrPr scale coefficient is 0.012 (Eg).
- The GrPr exponent is 0.8 (Fg).
- The d/L exponent in the free convection term is 0 (Gg).
- The exponent of the laminar term is .35 (Hg).
- The laminar-to-turbulent transition zone is 2100 < Re < 10000.
- The minimum film coefficient is 0 btu/(hr-ft^2-F).
- The friction adjustment exponent is .329 for heating and 0.231 for cooling in the laminar region. There is no adjustment in the turbulent region.
- Free convection based on wall temperature.

```
TRANSTHERMAL
+ HEATCOND  0.03
+ COLBURN   0.023  0.8  0.3333  0
+ GRASHOF   0.5  0.15  0.15  .001  1.0  0.012  0.8  0.  0.35
+ MIN.FILM.COEF  0.
+ HEAT.FRIC.EXP  .329  .231  0.  0.
+ FREE.CONVECTION WALL.TEMP
```

Using TRANSTHERMAL mode requires some specific input in the INPREP file, including pipes transfer lines with transient thermal entries and a data curve relating temperature as a function of distance for the ground. The following samples show these data requirements.

The following data curve illustrates the temperature of the ground being 80°F at the beginning of the pipe, 70°F 1 mile along the pipe and 60°F 2 miles along the pipe.

```
D GDTMP TEMP
80. 70. 60.
 0. 1 2
```

The following pipe shows transient thermal temperature mode values for the pipe wall and the ground material.

```
T ...
.
.
.
+ WALL DEN 490. CP .11 K 26  THIK .375 KNOT 2
+ GRND DEN 103. CP .19 K .15 THIK 60. KNOT 5 TEMP GDTMP
```

ENGLISH

INPREP

```
ENGLISH
```

Description

ENGLISH selects the English (or American engineering) system of units for input and output.

A list of all the default units is shown in the table of "Default units" on page 129. A list of all the built-in units in SPS is shown in the table "Built-in units" on page 131. To define a new unit to replace one of the built-in units, use DEFUNITS and USEUNITS.

Example input

Select the English system of units.

```
ENGLISH
```

METRIC

INPREP

```
METRIC
```

Description

METRIC selects the metric system of units for input and output.

A list of all the default units is shown in the table of "Default units" on page 129. A list of all the built-in units in SPS is shown in the table "Built-in units" on page 131. To define a new unit to replace one of the built-in units, use DEFUNITS and USEUNITS.

Example input

Select the metric system of units.

```
METRIC
```

DEFUNITS

INPREP

```
DEFUNITS NAME = EXPRESSION
```

Field	Units Key	Description
NAME	n/a	Alphanumeric name for units (for example, YD).
EXPRESSION	n/a	Expression defining a new unit name in terms of a built-in or previously defined unit name (for example, FT/3.0).

Description

DEFUNITS defines a new unit. The unit is defined in terms of a built-in unit or previous user-defined units.

If the quantity of the unit is expressed in the units on the right, it has the value shown when expressed in units on the left. For example, the following command could be included to define yards in terms of feet, where FT is the built-in name for feet:

```
DEFUNITS YD = FT / 3.0
```

This statement indicates that yards (YD) should equal the total number of feet divided by three.

For pressure and temperature units, the expression on the right of the equals sign must be reducible to the form a*X + b, where a and b are constants and X is a units name. For all other units, the expression must be reducible to the form a*X. It is not necessary that the expression be written in this form, only that the expression is reducible to this form. It may be written using the four arithmetic operators (+, -, *, and /) and parentheses ( ).

Take note

Using a new unit

Once a new unit has been defined with the DEFUNITS command, it must be linked to the types of units with the USEUNITS command. You should not redefine a built-in unit.

Use of [ ] not allowed

Do not use square brackets to enclose the expression. For example, the following command will produce an error:

```
  /*Incorrect entry
DEFUNITS YD = [FT / 3.0]
```

Example input

Example 1

Define a new length unit named Furlong that is equal to a length value in feet divided by 660.

```
DEFUNITS Furlong = FT / 660
```

Example 2

Define a new power unit named WTTS that is equal to a power value in kilowatts multiplied by 1000.

```
DEFUNITS WTTS = KW * 1000
```

USEUNITS

INPREP

```
USEUNITS KEYWORD NAME
```

Field	Units Key	Description
KEYWORD	n/a	Units keyword from “Default units” on page 129.
NAME	n/a	Name for unit (for example, YD).

Description

USEUNITS overrides units selected with either ENGLISH or METRIC on an item by item basis. USEUNITS indicates to use either a built-in unit or a defined unit (DEFUNITS). In the following example, YD must be defined using the DEFUNITS command because it is not a distance unit listed in the “Built-in units” on page 131.

```
USEUNITS LENGTH.HEADER YD
```

Take note

Abbreviation of units keywords and unit names

Units keywords may be abbreviated as documented in “Documentation conventions” on page 41.

Units names, either built-in or user-defined, may not be abbreviated. The names must be entered exactly as shown in either “Built-in units” on page 131 or as they were defined in the DEFUNITS, respectively.

BWRS composition user-definitions

You may specify a single component to have different units than the other components by specifying USEUNITS COMPOSITION.XX UNIT. The XX is the component name. The selection of mole or mass input is done with “STATE BWRS” on page 224.

SCL composition user-definitions

You may specify a single fluid to have different units than the other fluids by specifying USEUNITS COMPOSITION.XX UNIT. The XX is the fluid name.

Example input

Example 1

Use the units defined in the DEFUNITS example. See “DEFUNITS” on page 256.

```
USEUNITS LENGTH.PIPE Furlong
USEUNITS POWER WTTS
```

Example 2

When using ENGLISH, the default value for pressure is psig. Specify psia units for pressure; pipe length in miles and header length in feet.

```
USEUNITS PRESSURE PSIA
USEUNITS LENGTH.PIPE MI
USEUNITS LENGTH.HEADER FT
```

Note that the default units for LENGTH.PIPE is miles, so you need not enter it.

Example 3

Using BWRS, composition is to be in percent, except for water content, which is to be in ppm.

```
USEUNITS COMPOSITION PERCENT
USEUNITS COMPOSITION.H2O PPM
```

Example 4

Using SCL, composition is to be in fraction, except for fluid DRA, which is to be in ppm.

```
USEUNITS COMPOSITION FRACTION
USEUNITS COMPOSITION.DRA PPM
```

=EQUIPMENT

INPREP

```
=EQUIPMENT
```

Description

=EQUIPMENT signals the end of the general data section and the beginning of the definition of individual equipment and control system items. =EQUIPMENT is required for every INPREP file and the equals sign (=) is required at the beginning of this line.

Example input

Specify that the general data has ended and that the model structure and equipment data is beginning.

```
=EQUIPMENT
```

General pipe - transient (GP)

Peek and Poke Attributes

INPREP

```
GP FID FROM TO LEN OD WT
[+ COLEBROOK ROUGH]
[+ TEMPERATURE TEMP]
```

Field	Units Key	Description
FID	n/a	Unique name or facility identifier of the pipe.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
LEN	LENGTH.PIPE	Actual length of the pipe.
OD	DIAMETER	Outside diameter.
WT	DIAMETER	Wall thickness. {0.25 in.}
TEMP	DEGREES	Initial temperature of the fluid flowing through pipe. {temperature from thermal mode line}
ROUGH	ROUGH	Pipe roughness.

Description

The general pipe (GP) models a section of pipe of significant length with uniform diameter and wall thickness. The general pipe (GP) may be entered directly in SPS or exported by SynerGEE. GP is similar to the SPS Transfer Line (T) and the SynerGEE Fundamental Pipe Equation (FM). It differs from the FM in that you enter outside diameter and wall thickness instead of internal diameter only.

The GP pipe supports all the options that the transfer line (T) supports; only the options that are exported by SynerGEE are documented here. See the "Transfer line - transient (T)" on page 263 for the other options.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Pipe temperatures

The TEMP field is optional. If this field is not included, the pipe fluid temperature defaults to the global default temperature specified on the thermal mode line. If the temperature is entered, it overrides the global default temperature.

Frictional pressure loss calculations

For friction factor, f, the frictional pressure gradient is taken as:

$$\frac{dp}{dx} = \frac{-f\ G\ |G|}{2g_c\ D\ \rho}$$

where:

p	=	Pressure
x	=	Length along the pipe in the direction of positive flow
G	=	Mass flow per unit area
D	=	Internal pipe diameter
ρ	=	Density
g_c	=	Dimensional constant with units of (mass x length)/(force x time^2) that is used to relate force and mass units. In a consistent metric system, g_c is equal to unity.

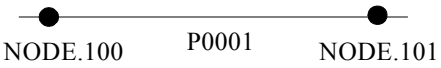
The friction factor for the general pipe is calculated through the Colebrook method.

$$\frac{1}{\sqrt{f}} = -0.86\ln\left(\frac{\varepsilon/D}{3.7} + \frac{2.51}{Re\sqrt{f}}\right)$$

where:

e/D is the dimensionless ratio of pipe roughness to inside diameter, and Re is the Reynolds number.

Example input



A pipe named P0001 is between a node named NODE.100 and a node named NODE.101. The line is 11 miles long and has a 24" outside diameter with a 0.281" wall thickness.

```
GP P0001 NODE.100 NODE.101 11 24 0.281
```

Transfer line - transient (T)

Peek and Poke Attributes

INPREP

```
T NAME FROM TO LEN OD WT
[+ TEMPERATURES T- T+]

/* The following four options are mutually exclusive; use at most one.

[+ COLEBROOK RUF]
[+ MOODY FF]
[+ NIKURADSE RUF]
[+ PPQ P- P+ Q]
[+ PIPE.DIST POST1 POST2 ...]
[+ HORIZ.DIST POST1 POST2 ...]
[+ ELEVATION ELEV1 [ELEV2] ...]
[+ MAOP MAOP1 [MAOP2] ...]
[+ LAOP LAOP1 [LAOP2] ...]
[+ MASP MASP1 [MASP2] ...]
[+ LASP LASP1 [LASP2] ...]
[+ AMULT AMP]
[+ YOUNG MODULUS]

/* The entries below are optional and are only input when the
/* TRANSTHERMAL temperature mode is used. You can input
/* 0, 1, 2, 3, or all 4 of the entries as desired.

[+ WALL] [DEN DENS] [CP HCAPA] [K HCOND] [KNOT N]
[+ WRAP] [DEN DENS] [CP HCAPA] [K HCOND] [THIK THICK] [KNOT N]
[+ FILL] [DEN DENS] [CP HCAPA] [K HCOND] [THIK THICK] [KNOT N]
[+ GRND] [DEN DENS] [CP HCAPA] [K HCOND] [BDEP BDEP] [THIK THICK] [KNOT N]
[+ TEMP TCURV]
```

Field	Units Key	Description
NAME	n/a	Unique name of the pipe.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
LEN	LENGTH.PIPE	Actual length of the pipe.
OD	DIAMETER	Outside diameter.
WT	WALL	Wall thickness.
T±	TEMPERATURE	Initial temperatures of the fluid at from-end (-) and to-end (+) ends. {the thermal mode temperature value}
Friction	n/a	Friction loss method

Field	Units Key	Description
RUF	ROUGHNESS	Colebrook or Nikuradse roughness.
AMP	n/a	Area multiplier (number of identical parallel lines). {1.0}
FF	n/a	Moody friction factor.
P±	PRESSURE	Pressures at from-end (-) and to-end (+) used to back-calculate roughness or friction factor.
Q	FLOW	Flow used to back-calculate roughness or friction factor (negative if flowing toward the from-end).
$POST_k$	LENGTH.PIPE	List of distance positions along the pipe.
$ELEV_k$	ELEVATION	List of elevations at each distance position.
$MAOP_k$	PRESSURE	List of maximum allowable operating pressures at each distance position.
$LAOP_k$	PRESSURE	List of lowest allowable operating pressures at each distance position.
$MASP_k$	PRESSURE	List of maximum allowable surge pressure at each distance position. {110% of corresponding MAOP value}
$LASP_k$	PRESSURE	List of lowest allowable surge pressure at each distance position. LASP does not have a physical meaning but could be used to set limits to prevent low pressure surge from causing vapor formation or a check that pipe collapse would not occur.
TABLE	n/a	Can list PIPE.DIST, HORIZ.DIST, ELEV, MAOP, LAOP, MASP, and/or LASP values along the pipeline at specific points in a table form (need at least 2 of the 4 variables in the table). Can also input IGNORE (the values, which must be numeric, are ignored). You can input values in two different formats: • With the table format, input the + TABLE with the + in column 1. On the next line, input a + in column 1 and then the keywords (ELEV, MAOP, etc.). Then input the numbers beginning on the next line. An example of this format can be seen in "Example 4" at the end of this section. • With the CSV format, enter + TABLE in the first line and then the keywords, separated by either commas or spaces, in the second line. enter a { bracket in the third line, and then start entering data in the fourth line, again separated by either commas or spaces. After all data is entered, enter a } bracket on the last line. An example of this can be seen in "Example 4" at the end of this section. For more information on the CSV input format, see "CSV input syntax" on page 52.
DRAD	1/length	DRA Degradation Coefficient

Field	Units Key	Description
MODULUS	TENS.STRESS	Young's modulus. {29 MMPSI—Specified through SET.LIMIT}
DENS	DENSITY	Density of: • pipe {490 lbm/ft^3} • wrap {36 lbm/ft^3} • fill {103 lbm/ft^3} • ground {103 lbm/ft^3} See "Pipe environment calculations" under "Take note".
HCAPA	HEAT.CAPACITY	Heat capacity of: • pipe {0.11 btu/lbm -°F} • wrap {0.25 btu/lbm -°F} • fill {0.19 btu/lbm -°F} • ground {0.19 btu/lbm -°F} See "Pipe environment calculations" under "Take note".
HCOND	HEAT.COND	Thermal conductivity of: • pipe {26 btu/hr - ft -°F} • wrap {0.087 btu/hr - ft - °F} • fill {0.15 btu/hr - ft - °F} • ground {0.15 btu/hr - ft - °F} See "Pipe environment calculations" under "Take note".
BDEP	WALL	Pipe burial depth, existing natural ground grade contour or natural water bottom grade contour to "top-of-pipe" and coating. The measurement is to the top (or outside of) coating, if present. {48 inches}
THICK	WALL	Thickness of: • pipe {0.336 inches} • wrap {0 inches} • fill {0 inches} • ground {48 inches} See "Pipe environment calculations" under "Take note".

Field	Units Key	Description
N	n/a	Number of thermal knots per hydraulic knot in the layer • pipe {2} • wrap {0} • fill {0} • ground {5} See "Pipe environment calculations" under "Take note".
TCURV	n/a	Data curve temperature profile of the outer boundary. {The default is the temperature specified in "TRANSTHERMAL" on page 249.

Description

The transfer line (T) models a section of pipe of significant length with uniform diameter and wall thickness. Shorter sections of pipe should be modeled using the header elements. See "Header (H)" on page 276.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Pipe end temperatures

Entries in the T- and T+ fields are optional. If these fields are not entered, the pipe fluid temperature defaults to the global default temperature specified on the thermal mode indicator (ISOTHERMAL, etc.). If these fields are entered, the entered values override the global default temperature.

Overall heat transfer coefficient peek

The overall heat transfer coefficient, OHTC, is calculated for each pipe by the software as a way to help you double-check your data entry of the various heat transfer coefficients. The calculated OHTC includes the ground, fill, wrap, and pipe wall layers. It does not include the film coefficient.

Note that SPS performs a detailed transient simulation using each layer separately. The OHTC is calculated for comparison purposes only.

Distance plots

Any transfer line distance plot item (vector) or any peek (scalar) can be plotted as a function of distance using the distance plot capabilities of TRANS. Note that distance plot items are usually functions of distance so they vary with distance along each pipe. However, peeks are scalars and are constant along each pipe, but may vary pipe to pipe.

Distance entries for plots

You may specify ELEV, MAOP, LAOP, etc. data as a function of distance along each pipe. If you specified TABLE, these values are entered in table format. In defining these values, you may choose one of three methods for describing the distance along the pipe:

Method 1 - No HORIZ.DIST or PIPE.DIST

If you specify ELEV, MAOP, etc. without any PIPE.DIST or HORIZ.DIST values, then the ELEV, MAOP, and/or LAOP values are taken to be equally spaced along the length of the pipe.

Method 2 - PIPE.DIST

The PIPE.DIST entry is used to input actual pipe lengths. Defining ELEV, MAOP, etc. as a function of PIPE.DIST implies that the distances entered are relative to the length of the pipe. (5000 feet of pipe tilted on a 30 degree slope is still defined as 0-5000 on the PIPE.DIST column of data).

Method 3 - HORIZ.DIST

The HORIZ.DIST entry is used to input horizontal distances of the pipe (such as survey lengths) along the ground. Defining ELEV, MAOP, etc. as a function of HORIZ.DIST implies that the distances entered are relative to the horizontal distance from one end of the pipe to the other end. (5000 feet of pipe tilted on a 30 degree slope is defined as 0-4330 on the HORIZ.DIST column of data.)

In either case, the distances are scaled so that the total length of the pipe matches the LEN entry on the first line. You may specify any number of values, but the number of distance values entered must equal the number of ELEV, LAOP, and MAOP values entered.

Time plots of array peeks

To create a time plot for array peeks of transfer lines, you must create a user-defined variable and then plot the variable.

For example, you have a TRENDLIST * running a LIQUID batched model. If you show a transfer line, the BPP(?) peek is not being trended. To trend BPP, create the following user-defined variable:

```
DEFINE ZZZ = T1:BPP(1)
```

Now you can trend ZZZ on a time plot.

For more information on creating a user-defined variable, see “DEFINE” on page 568.

Elevation

The ELEVATION (or simply ELEV) entry is used to enter detailed elevation along the pipe. Elevation is entered as actual elevation above sea level. The ELEV values correspond to the distance or other table values on a point by point basis.

Area multiplier

The AMULT line is used to simplify input for identical parallel lines or for calculating annular flow or flow inside of a non-circular pipe. Only one set of pipe data needs to be input for identical parallel lines. Specify the total number of parallel lines with the AMP entry. To provide a means for calculating annular or non-circular cross-sectional flow, this entry may be made as any positive number (not necessarily an integer). The FLOW entry is the total flow for all AMP lines.

Friction loss methods

You may specify one of four available methods for calculation of frictional pressure drop:

- Colebrook

- Nikuradse
- Moody
- Back-calculate either pipe roughness or friction factor depending on the global friction key specified

You may override the default method or the default values on a pipe by pipe basis.

Frictional pressure loss calculations

For friction factor, f, the frictional pressure gradient is taken as:

$$\frac{dp}{dx} = \frac{-f\ G\ |G|}{2g_c\ D\ \rho}$$

where:

p	=	Pressure
x	=	Length along the pipe in the direction of positive flow
G	=	Mass flow per unit area
D	=	Internal pipe diameter
ρ	=	Density
g_c	=	Dimensional constant with units of (mass x length)/(force x time^2) that is used to relate force and mass units. In a consistent metric system, g_c is equal to unity.
f	=	Friction factor determined in one of three ways according to the friction key entered on the FRICTION line. These are:

Colebrook

$$\frac{1}{\sqrt{f}} = -0.86 \ln\left(\frac{\varepsilon/D}{3.7} + \frac{2.51}{Re\sqrt{f}}\right)$$

where:

e/D is the dimensionless ratio of pipe roughness to inside diameter, and Re is the Reynolds number.

Nikuradse

f is a constant given by the equation:

$$1/\sqrt{f} = 2\log_{10}(D/e + 1.14)$$

where:

D/e is the dimensionless ratio of inside pipe diameter to roughness.

Moody

f is a constant FF. This corresponds to fully turbulent flow.

f is used in the "Momentum Equation" under "Take note".

Friction factor regions

Friction factor calculation methods depend on the Reynolds Number (Re) of the fluid. The following describes the treatment and breakpoints used within SPS.

Sublaminar

Re < 100 uses a constant friction factor of .64 — Note that Sublaminar Reynolds Factor can be defined by POKEing $SUBLAMINAR.FRICTION.FACTOR = new value.

Laminar

100 < Re < 2100 — Friction factor is calculated as 64/Re.

Transition Region

2100 < Re < 3000 — Friction factor is the weighted average of the laminar and turbulent friction factors.

Turbulent Region

3000 < Re — Friction factor is calculated with the user-specified method.

PPQ

If pre-determined pressure and flow data is available, you can use the + PPQ form to allow PREPR to back-calculate either the pipe roughness or friction factor depending on the friction key specified in the PIPEPARMS command (Note that this option is only available and applicable to Colebrook roughness and Moody Friction Factor). The specified pressures must include elevation effects.

Fluid dynamics calculations

Internally, the model uses the following partial differential equations to model flow along a pipeline.

- Continuity Equation
- Momentum Equation
- Energy Equation
- Flow Area Equation
- Transient Thermal Equation in Pipe and Surroundings

Please see the nomenclature in this section for the meaning of the different variables.

Continuity Equation

$$\frac{\partial(v\,A\rho)}{\partial x}+\frac{\partial(A\rho)}{\partial t}=0$$

Momentum Equation

$$\frac{\partial(vA\rho v)}{\partial x}+\frac{\partial(A\rho v)}{\partial t}+g_c A\,\frac{\partial P}{\partial x}+\frac{A\rho f\,|v|v}{2d}+A\rho g\,\frac{\partial h}{\partial x}=0$$

Energy Equation

$$\frac{\partial\left(vA\rho\left(g_c U + v^2/2\right)\right)}{\partial x} + \frac{\partial\left(A\rho\left(g_c U + v^2/2\right)\right)}{\partial t} +$$

$$g_c\left(\frac{\partial(vPA)}{\partial x} + P\,\frac{\partial A}{\partial t}\right) + A\rho v g\,\frac{\partial h}{\partial x} + g_c \pi d h_1\,\left(T - T_0\right) = 0$$

Note: If the internal energy density, U, and pressure, P, are taken as a functions of ρ and T, then by definition:

$$c_v = \frac{\partial U}{\partial T}$$

and by a simple thermodynamic identity:

$$\frac{\partial U}{\partial \rho} = -\frac{\left(T_{abs}\,\frac{\partial P}{\partial T_{abs}} - P\right)}{\rho^2}$$

Flow Area Equation

$$A = A_o \bullet A_{mult} \bullet [\ 1 + p_{mult} \bullet (p - p_o) + t_{mult}\,(t - T_o)\ ]$$

where:

A_o	=	Area calculated from user-entered diameter
A_{mult}	=	Area multiplier from user input (usually 1)
p_{mult}	=	Diameter divided by Young's modulus multiplied by the wall thickness
t_{mult}	=	Thermal expansion coefficient (user input, PIPEPARMS line)
p_o	=	Custody transfer pressure
t_o	=	Custody transfer temperature
p	=	Internal pipeline pressure (calculated at every knot)
t	=	Internal pipeline temperature (calculated at every knot)

Pipe environment calculations

For TRANSTHERMAL simulations, you need to specify thermal properties for the total environment surrounding the pipe using the transient-thermal entries. One line of data is entered for each of the radially-symmetric concentric layers surrounding the fluid: the pipe-wall, each pipe wrapping material, the fill material, and the ground material. For each, you may enter density, heat capacity, thermal conductivity, and thickness. Each thickness is defined from the outer-diameter of the previous layer.

The ground layer thickness needs special consideration. SPS uses a radially symmetric formulation for heat transfer, but the ground layer is not radially symmetric. To take into account the actual heat transfer paths, the effective ground thickness, grnd.thik, is calculated from the burial depth using the following equation:

$$\text{grnd.thik} = R1\ \left(\frac{2h}{D} + \left[\ \left(\frac{2h}{D}\right)^2 - 1\ \right]^{\frac{1}{2}} - 1\right)$$

where:

grnd.thik	=	Ground thickness for the radially symmetric model
R1	=	The radius from pipe center to the ground layer (includes wrap and fill)
h	=	Actual burial depth to center line of pipe
D	=	Pipe outside diameter

If you enter a value for GRND BDEP instead of GRND THIK, the software calculates the grnd.thik value automatically, using:

$$h = D / 2 + WRAP.THIK. + BDEP$$

BDEP is measured from the existing natural ground grade contour or natural water bottom grade contour to “top-of-pipe” and coating. The measurement is to the top (or outside of) coating, if present.

This concept of compensating for the actual heat transfer paths is often called the S-factor. The effective ground thickness is used regardless of whether the simulation is steady-state or transient.

In addition, you may specify a curve with temperature as a function of pipe length (from the from-end) for the outer-boundary using a data curve (type TEMP), with the actual curve name entered in the TCURV field. The default is the temperature specified on the TRANSTHERMAL command.

In addition, the ground temperature at the pipe ends may be adjusted using the :TG- and :TG+ attributes. The ground temperature along the pipe is linearly interpolated between :TG- and :TG+. To be physically realistic, adjustments to the ground temperature must occur slowly, for example, with a RAMP command set up to ramp the temperatures over a period of days or even weeks.

$$k\left(rT_r\right)_r / r = C_p \rho T_t$$

Transient Thermal Equation in Pipe and Surroundings

In each radial zone with continuity in temperature and heat flux rate at each zone boundary, $T(R_1) = T_0$ and $T(R_2) =$ TCURV value, where R_1 is the pipe wall radius and R_2 is the radius of the outer edge of the ground thickness.

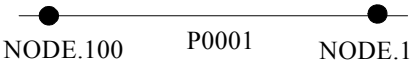
In consistent units the variables above are as shown in the nomenclature below. The reference conditions are those entered on the CUSTODY line. For more information, see “CUSTODY” on page 215.

Symbol	Description
A	Cross section area for flow in pipe
A_{ref}	Cross section area at P_{ref} and T_{ref}
C_p	Heat capacity at constant pressure
C_v	Heat capacity at constant density
D	Inside pipe diameter
E	Young’s Modulus
f	Friction factor
g	Acceleration due to gravity
g_c	Dimensional constant, mass x length/(force x time^2)

Symbol	Description
h_1	Film coefficient for heat transfer, fluid-to-pipe wall
k	Heat transfer coefficient for pipe surroundings
P	Pressure of fluid
P_{ref}	Reference pressure
R1	The radius from pipe center to the ground layer (includes wrap and fill)
R_2	Outer radius for thermal calculation
r	Radial distance from center line of pipe
T	Temperature of fluid
T_{abs}	Absolute fluid temperature
T_{ref}	Reference temperature
T_0	Fluid-side pipe wall temperature
U	Internal energy density (force * length/mass)
t	Time
v	Fluid velocity in the direction of positive x
x	Distance from nominal from-end of pipe
ρ	Fluid density
η	Coefficient of thermal expansion
τ	Pipe wall thickness

Example input

The first two examples reference this diagram.



Example 1

A pipe named T0001 is between a node named NODE.100 and a node named NODE.101. The line is 11 miles long and has a 24" outside diameter with a 0.281" wall thickness. The elevation is 1200 ft at the first node and 1500 ft at the second node.

```
T T0001 NODE.100 NODE.101 11 24 0.281
+ ELEVATION 1200 1500
```

Example 2

Same as in Example 1, except the 11 mile pipe has seven different elevation points.

```
T T0001 NODE.100 NODE.101 11 24 0.281
+ PIPE.DIST 0   2.2  3.5  5.9 7.1  8.8  11
+ ELEVATION 1200 1400 1700 1100 1300 1600 1500
```

Example 3

A pipe named TL4 is installed between nodes ND30 and ND31. The line is 2 miles long and has a 20" outside diameter. with a 0.25" wall thickness. The temperature mode in the model is TRANSTHERMAL. This example inputs values for the pipe wall and the ground. (The wrap and fill values are the default values.) A data curve named GDTEMP for the ground temperature as a function of distance has been previously defined in the INPREP file:

```
T TL4 ND30 ND31 2 20 .25
+ WALL DEN 411 CP .14 K 24  THIK .25 KNOT 3
+ GRND DEN 111 CP .23 K .19 THIK 55  KNOT 6
+ TEMP GDTEMP
```

Example 4

A pipe named TL3 is between a node named N1 and a node named N2. The line is 30 miles long and has a 20" outside diameter with a 0.25" wall thickness. The first example below shows the pipe distance and elevation data for the pipe in table form:

```
T TL3 N1 N2 30 20 0.25
+ TABLE
+ PIPE.DIST   ELEV
+  0          200
+  5          250
+ 10          220
+ 15          245
+ 20          265
+ 25          220
+ 30          250
```

The following example shows the same pipe distance and elevation data for the pipe in CSV form:

```
T TL3 N1 N2 30 20 0.25
+ TABLE
{
PIPE.DIST, ELEV
0, 200
5, 250
10, 220
15, 245
20, 265
25, 220
30, 250
}
```

Calculating DRAD values

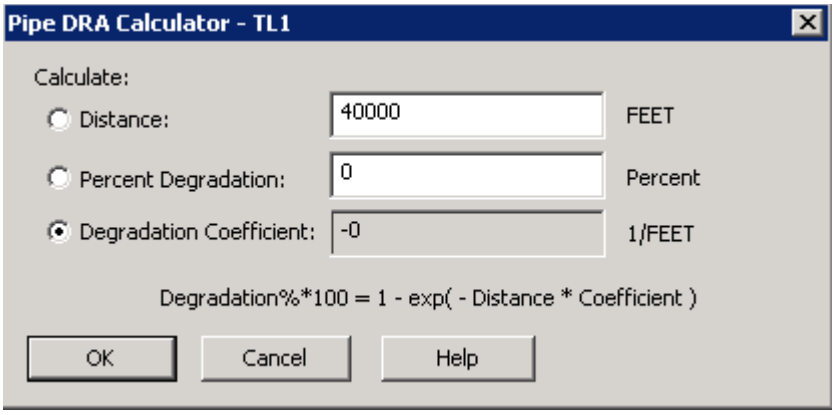
The DRA Calculator is accessed through the Transfer Lines node in the Model Explorer. The DRA Calculator allows you to calculate Distance, Percent Degradation, or Degradation Coefficient values.

To calculate DRAD values

1 From the **Model Builder** window, in the **Model Explorer**, expand **Pipes**, and then expand **Transfer Lines**.

2 Double-click a transfer line. The Transfer Line dialog box appears.

3 Right-click the **DRAD** box, and then click **Calculator**. The Pipe DRA Calculator dialog box opens.



Pipe DRA Calculator in Model Builder

4 To calculate the Degradation Coefficient, select the **Degradation Coefficient** option, and then enter values in the **Distance** and **Degradation Coefficient** boxes.

—or—

To calculate the Distance, select the **Distance** option, and then enter values in the **Percent Degradation** and **Degradation Coefficient** boxes.

—or—

To calculate the Percent Degradation, select the **Percent Degradation** option, and then enter values in the **Distance** and **Degradation Coefficient** boxes.

5 Click **OK**. The Pipe DRA Calculator dialog box closes and the calculator's Degradation Coefficient appears in the DRAD box.

Header (H)

Peek and Poke Attributes

INPREP

```
H NAME FROM TO LEN OD WT [T-] [T+]

/* One of the following two lines must be entered

[+ PPQ P- P+ FLOW]
[+ MOODY FRICT]
[+ ELEVATION ELEV- ELEV+]

/* One of the following is used for a simple heat exchanger

[+ TEMP TMP ]
[+ DELT DLT ]
```

Field	Units Key	Description
NAME	n/a	Unique name of the header.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
LEN	LENGTH.HEADER	Length of the header.
OD	DIAMETER	Outside diameter.
WT	WALL	Wall thickness.
T±	TEMPERATURE	Initial temperatures at from-end (-) and to-end (+). Leave blank for ISOTHERMAL systems.
P±	PRESSURE	Pressure at each end. PPQ entry is required for PREPR to back-calculate friction.
FLOW	FLOW	Reference flow (negative if + to -). PPQ entry is required for PREPR to back-calculate friction.
FRICT	n/a	Moody friction factor.
ELEV±	ELEVATION	Elevation at each end. {ΔELEV = 0}

Field	Units Key	Description
TMP	TEMPERATURE	Used to model an idealized heat exchanger with a fixed outlet temperature. The outlet temperature is set to TMP. The outlet is defined to be the end of the exchanger that is delivering flow into a node. **Note:** Does not apply when using ISOTHERMAL mode.
DLT	ΔTEMPERATURE	Used to model an idealized heat exchanger with a constant temperature differential. The outlet temperature is changed from the inlet temperature by an amount equal to DLT. Positive DLT indicates a temperature rise and negative DLT indicates a temperature drop. If you want a temperature change of zero, you must enter a small non-zero number. Entering 0 disables the feature and makes it not pokable. **Notes:** Does not apply when using ISOTHERMAL mode.

Description

The header (H) models short sections of pipe such as at a station, yard piping, etc. Headers do not contain line pack and should not be used to model a transient pipe. Headers are not used to calculate heat transfer with the surrounding environment. Longer pipes requiring these types of calculations should be modeled using the transfer line or general pipe. For more information, see "General pipe - transient (GP)" on page 261 or "Transfer line - transient (T)" on page 263.

The header may also be used to model an idealized heat exchanger with either a fixed outlet temperature or a fixed temperature differential.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Friction factor

If P± and FLOW are entered, then don't enter FRICT. SPS automatically back-calculates the friction factor using the pressure/flow/temperature data. Otherwise, enter the Moody friction factor.

Header pressure drop calculations

Pressure drop for headers is computed in consistent units from the definition of fully turbulent friction:

$$\Delta P = -[f\rho v\,|v|\,L / 2D + g\rho\Delta h] / g_c$$

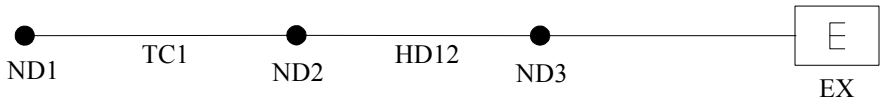
where:

D	=	Inside pipe diameter
f	=	Moody friction factor

g	=	Acceleration due to gravity
g_c	=	Dimensional constant (mass * length)/(force * time^2)
Δh	=	(h+) - (h-)
h+	=	Elevation at from-end of header
h-	=	Elevation at to-end of header
L	=	Header length
ΔP	=	(P+) - (P-)
P-	=	Pressure at from-end of header
P+	=	Pressure at to-end of header
v	=	Velocity in the direction towards to-end of header.
r	=	Average density of fluid at flowing conditions [(ρ+) + (ρ-)]/2

Example input

Example 1



A header named HD12 is installed between nodes ND2 and ND3. The line is 300 feet long and has a 18" outside diameter with a 0.375" wall thickness. The initial upstream and downstream pressures are not specified so they equal to the INIT field on the PIPEPARMS directive, corrected for elevation. The initial flow is 0 and the friction factor is 0.0125. The elevation is 900 ft at the inlet node and 904 ft at the outlet node: (LENGTH.HEADER is in FT):

```
H HD12 ND2 ND3 300 18 0.375
+ MOODY 0.0125
+ ELEV 900 904
```

Example 2

A heat exchanger modeled as a header is named BHAC1 and is installed between nodes N123 and N124. The heat exchanger is to produce a pressure drop due to friction and also maintain the discharged flow temperature at 130°F. The heat exchanger is 300 feet long and has a 18" outside diameter with a 0.375" wall thickness. (LENGTH.HEADER is in FT.) The initial upstream and downstream pressures are not specified so they equal to the INIT field on the PIPEPARMS directive, corrected for elevation. The initial flow is 0 and the friction factor is 0.0125. The elevation is 900 ft at the inlet node and 904 ft at the outlet node:

```
H BHAC1 N123 N124 300 18 0.375
+ MOODY 0.0125
+ ELEV  900 904
+ TEMP 130
```

FLOWMETER

Peek and Poke Attributes

INPREP

```
FLOWMETER NAME FROM TO
+  PPQ PIN POUT FLOW [T]
```

Field	Units Key	Description
NAME	n/a	Unique name of the flow meter.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
PIN	PRESSURE	Inlet pressure to the flow meter; used only for sizing the resistance.
POUT	PRESSURE	Outlet pressure from the flow meter; used only for sizing the resistance.
FLOW	FLOW	Representative positive flow through the flow meter; used only for sizing the resistance.
T	TEMPERATURE	Temperature corresponding to pressures and flows; used only for sizing the resistance. {thermal mode temperature}

Description

FLOWMETER models physical flow meters in the system. In off-line systems, the flow is calculated based on system hydraulics. In online systems, a SCADA element controls the flow through the element. For more information on using the flow meter in online products, see “FLOWMETER for online modeling” on page 787.

In off-line models, the FLOWMETER acts similarly to a header with a pressure drop that is calculated based on the resistance and the flow rate.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Pressure loss

Input is used to calculate a K factor for pressure drop.

$$K = \rho \Delta P / FLOW^2$$

where:

K	=	Resistance coefficient
ρ	=	Density at P_{in} and T
ΔP	=	P_{in} - P_{out} for the FLOWMETER
FLOW	=	Flow through the FLOWMETER

Example input

Flow meter has a normal flow of 250 MMCFD, an inlet pressure of 600 psig and an outlet pressure of 599 psig.

```
FLOWMETER M1234 N0123 N0124
+ PPQ  600 599 250
```

Heat exchanger (HE)

Peek and Poke Attributes

INPREP

```
HE NAME -TS +TS -SS +SS UAVE AREA TUBEQ TUBEΔP
[+ [SHELLQ] [SHELLΔP]]
```

Field	Units Key	Description
NAME	n/a	Unique name of the heat exchanger.
±TS	n/a	Tube-side from/to connection points or named nodes.
±SS	n/a	Shell-side from/to connection points or named nodes).
UAVE	HEAT.TRANSFER	Average overall heat transfer coefficient between the tube-side and shell-side fluids.
AREA	AREA	Total heat transfer area.
TUBEQ	FLOW	Expected operating flow rate in tube-side.
TUBEΔP	ΔPRESSURE	Pressure drop at TUBEQ in the tube-side.
SHELLQ	FLOW	Expected operating flow rate on shell side. {TUBEQ}
SHELLΔP	ΔPRESSURE	Pressure drop at SHELLQ on the shell side. {TUBEΔP}

Description

The heat exchanger (HE) models a tube and shell heat exchanger. For modeling idealized fixed outlet temperature or fixed ΔT heat exchangers, use the header (H) element. See "Header (H)" on page 276.

Note: HE should not be used with the ISOTHERMAL option.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Naming convention

The tube and shell sides are each modeled as a special header with the NAME attached to a .T or .S attribute. The tube-side header is named NAME.T, while the shell-side header is NAME.S. These names are generated automatically, and care should be taken to reference the extended name when peeking or defining variables, etc. The heat flux rate (or duty, :DTY) is available for each side of the heat exchanger.

Heat flux rate

The heat flux rate between the tube and shell fluids is:

$$Q_H = UAVE \bullet AREA \bullet \Delta T_{lm}$$

where:

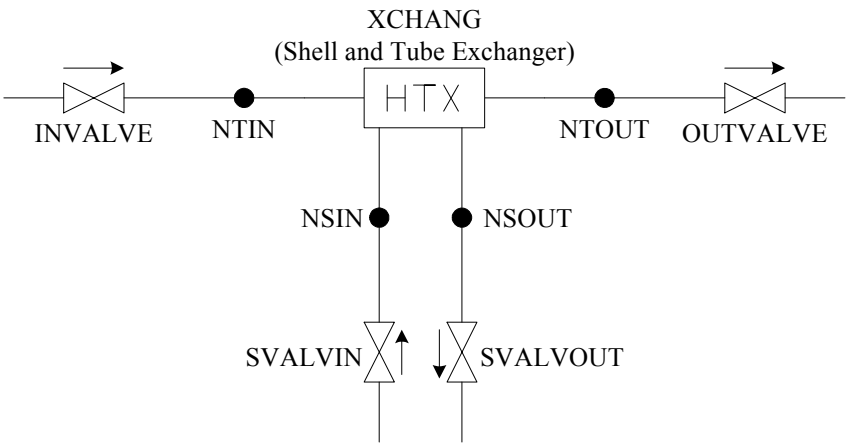
UAVE	=	the average overall heat transfer coefficient
AREA	=	the total heat transfer area
ΔT_{lm}	=	the logarithmic mean temperature difference between the fluids at the ends of the tube and shell sides.

The assumptions in this equation are that the heat capacity of the fluid on each side is constant along the length with a value equal to the average of the heat capacities of the fluid at the inlet and outlet, and that UAVE may be represented by a single constant. For these reasons, the simulation of boiling or condensation in a heat exchanger is not treated properly.

Distance plot for a header

If you want to create a distance plot that includes heat exchanger data, you should not specify a path that includes both the shell and tube.

Example input



The tube side of a shell/tube heat exchanger named XCHANG feeds from NTIN and delivers to NTOUT. The shell side feeds from NSIN and delivers to NSOUT. The area for heat transfer is 800 square inches and the overall heat transfer coefficient is 130. The tube-side flow rate is 650 MB/D and the tube-side pressure drop is 4.2 psig. The shell side flow rate is 1000 MB/D and the shell-side pressure drop is 2.9 psig:

```
HE XCHANG NTIN NTOUT NSIN NSOUT 130 800 650 4.2
+ 1000 2.9
```

NODE

Peek and Poke Attributes

Note: If you are modeling flows using externals, you need not specify NODEs.

INPREP

```
NODE NAME SP SNQ [T+] [PRATE] [QRATE]
[+ USEUNITS FLOW UNAME]
[+ FLUID FLU1  FLU2  ... FLUn
+ INITIAL FR1  FR2  ... FRn]
+ QMIN QMIN
+ QMAX QMAX
+ PMIN PMIN
+ PMAX PMAX
[+ ELEVATION ELEV]
```

Field	Units Key	Description
NAME	n/a	Node name.
SP	PRESSURE	Set point pressure. (Enter a zero if using flow control.).
SNQ	FLOW or UNAME	Set point flow rate. (Enter a zero if using pressure control.).
T+	TEMPERATURE	Temperature of the fluid if the flow is into the model. {temperature from thermal mode line}
PRATE	ΔPRESSURE/TIME	Rate of pressure change for a pressure ramp. {120 pressure units/minute}
QRATE	ΔFLOW/TIME	Rate of flow change for a flow ramp. {120 flow units/minute}
ATTRIBUTE	n/a	Pokable variable for the node (e.g., PMAX, QMIN, ...).
EXPRESSION	n/a	New value for the attribute.
UNAME	n/a	Flow unit other than the default unit to be used for this node.
FLU_i	n/a	Names of fluids or components entering the system at this node.
FR_i	FRACTION	Corresponding fraction of each fluid or component entering the system at this node. **Note:** You should enter this value, even if it is zero.
QMIN	FLOW	Minimum allowable flow at the node. {-10,000 MB/D}
QMAX	FLOW	Maximum allowable flow at the node. {10,000 MB/D}
PMIN	PRESSURE	Minimum allowable pressure at the node. {-14.6 PSIG}
PMAX	PRESSURE	Maximum allowable pressure at the node. {5000 PSIG}
ELEV	ELEVATION	Elevation at the node.

Description

A node is a connection point between two or more elements. At any node, you may model flow that is into or out of the model. Additionally, you may want to consider defining flow through externals. See “Nodes versus externals” on page 138.

The sign convention for node flow is exactly the same as that of a TAKE external: Flow entering the model is positive and flow leaving the model is negative.

Nodes can be configured to maintain either a pressure or flow set point (but not both), or to follow a pressure or flow schedule specified with the RAMP command either here or in the INTRAN file. For pressure-controlled nodes, enter the pressure set point in the PRES field and a 0 in the FLOW field. For flow-controlled nodes, enter a 0 in the PRES field and the flow set point in the FLOW field. Entering 0 for both PRES and FLOW sets the node on flow control at 0 flow. The PRATE and QRATE fields are used to limit the rate of change from one value to another.

There can be multiple + POKE and + RAMP lines for the node and they can be in any order. See “POKE and RAMP” on page 423. Each node has PMIN, PMAX, QMIN and QMAX peek/poke attributes that appear on the Show window. (See “Peek and Poke Keyletters and Attributes” on page 639). The defaults for these four pokable attributes are as follows:

```
QMIN: -10000 MBPD
QMAX:   10000 MBPD
PMIN:  -15 PSIG
PMAX:   5000 PSIG
```

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Pressure, flow, thermal flow control with constraints

See “Controlling pressure, flow, or heat rate at nodes, SALES, and TAKES” on page 140.

Composition control

See “Setting composition at inflows” on page 139.

Temperature control

See “Setting temperature at inflows” on page 140.

Flow units

The units of flow at a node may be specified on a node by node basis. This capability is most useful when the particular node has a significantly different flow rate than most other nodes or externals. For example, a model that has most node/external flows in MB/D may have some flows, such as for trace components and/or DRA, in GAL/MIN.

FLUID and INITIAL

The +FLUID and +INITIAL lines are used to set the composition that the node will use when the node flow is into the model. See “Setting composition at inflows” on page 139.

SET.LIMIT

Nodes use the same set limits as Externals. The following is an example of a set limit for a node:

```
SET.LIMIT
+ E SALE/TAKE T -10 0 120 700
```

For more information on SET.LIMIT, see “SET.LIMIT” on page 219.

Example input

Example 1

A pressure-controlled node with named EXT1 has an initial pressure of 30 psig and the set point temperature is 60°F (note that this temperature does not affect the flowing temperature unless the flow is into the model).

```
NODE EXT1 30. 0. 60.
```

Example 2

A flow-controlled node with negative flow is named EDEL. The initial flow rate is 50 MB/D and the set point temperature is 60°F (note that this temperature does not affect the flowing temperature unless the flow is into the model).

```
NODE EDEL 0. -50. 60.
```

Example 3

A flow-controlled node with positive flow named INLET has an initial flow of 1000 and the temperature of the fluid entering the system is 60°F. The rate of flow change for a flow ramp is to be a value of 800 flow units/minute:

```
NODE INLET 0. 1000. 60. 0 800.
```

Example 4

A pressure-controlled node named OUT1 has an initial pressure of 300 psig and the temperature of the fluid entering the system is 60°F. The pressure is set to 175 psig when the simulation starts to run:

```
NODE OUT1  300. 0. 60.
+ POKE SP = 175
```

Example 5

A pressure-controlled node named NOD1 has an initial pressure of 28 psig and the temperature of the fluid entering the system is 65°F. The elevation is 540 feet:

```
NODE EXT1 28. 0. 65.
+ ELEVATION 540
```

E SALE/TAKE

Peek and Poke Attributes

Note: Externals have only a + side. To reference an external as a node connection for some other element, precede the external name with the + symbol.

INPREP

```
E NAME CNC TYPE SP SNQ [T+] [PRATE] [QRATE]
[+ USEUNITS FLOW UNAME ]
[+ FLUID FLU1  FLU2  ... FLUn
 + INITIAL FR1  FR2  ... FRn ]
+ QMIN QMIN
+ QMAX QMIN
+ PMIN QMIN
+ PMAX QMIN
[+ ELEVATION ELEV]
```

Field	Units Key	Description
NAME	n/a	Unique name of the external.
CNC	n/a	Connection point or named node.
TYPE	n/a	One of the keywords SALE or TAKE.
SP	PRESSURE	Set point pressure. (Enter a zero if using flow controlled SALE or TAKE).
SNQ	FLOW or UNAME	Set point flow rate (enter a zero if using pressure controlled SALE or TAKE).
T+	TEMPERATURE	Temperature of the fluid if the flow is into the model. {temperature from thermal mode line}
PRATE	ΔPRESSURE/TIME	Rate of pressure change for a pressure ramp. {120 pressure units/minute}
QRATE	ΔFLOW/TIME	Rate of flow change for a flow ramp . {120 flow units/minute}
ATTRIBUTES	n/a	Pokable variable for the external (i.e., PMAX, QMIN, ...).
EXPRESSION	n/a	New value for the attributes.
UNAME	n/a	Flow unit if other than the default unit is to be used for this external.
FLU_i	n/a	Names of fluids or components entering the system at this external.
FR_i	FRACTION	Corresponding fraction of each fluid or component entering the system at this external. **Note:** You should enter this value, even if it is zero.
QMIN	FLOW	Minimum allowable flow at the node. {-10,000 MB/D}

Field	Units Key	Description
QMAX	FLOW	Maximum allowable flow at the node. {10,000 MB/D}
PMIN	PRESSURE	Minimum allowable pressure at the node. {-14.6 PSIG}
PMAX	PRESSURE	Maximum allowable pressure at the node. {5000 PSIG}
ELEV	ELEVATION	Elevation at the node.

Description

The SALE or TAKE (E) external type is used to model a regulated flow that enters or leaves the network. The external can be specified to be either a SALE (flow leaving the system), or a TAKE (flow entering the system). The only difference between SALE and TAKE type externals is the sign convention used for *nominal flow* (NQ); positive NQ at a TAKE point indicates flow entering the system, while positive NQ at a SALE point indicates flow leaving the system. For both SALE and TAKE type externals, positive Q indicates flow entering the system, and negative Q indicates flow leaving the system.

SALE and TAKE type externals can be configured to maintain either a pressure or flow set point (but not both simultaneously), or to follow a pressure or flow schedule specified either with the +RAMP line on the E command (see "POKE and RAMP" on page 423) or in the INTRAN file with the RAMP command (see "RAMP" on page 504). For pressure-controlled externals, enter the pressure set point in the PRES field and a 0 in the FLOW field. For flow-controlled externals, enter a 0 in the PRES field and the flow set point in the FLOW field. Entering 0 for both PRES and FLOW sets the external on flow control at 0 flow. The PRATE and QRATE fields are used to control the rate of change during a ramp from one value to another.

An external can have multiple + POKE and + RAMP lines, and they can be in any order. See "POKE and RAMP" on page 423. Each SALE and TAKE has PMIN, PMAX, QMIN and QMAX peek/poke attributes that appear on the Show window. (See "Peek and Poke Keyletters and Attributes" on page 639). The defaults for these four pokable attributes are as follows:

```
QMIN:   -10000 MBPD
QMAX:    10000 MBPD
PMAX:    5000 PSIG
PMIN:   -15 PSIG
```

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Pressure, flow, thermal flow control with constraints

See "Controlling pressure, flow, or heat rate at nodes, SALES, and TAKES" on page 140.

Subtype attribute

The :STYP attribute distinguishes the external type in a Show window, report, or text display. For a SALE external, the :STYP peek attribute is SALE. For a TAKE external, the :STYP peek attribute is TAKE. Also, this peek attribute provides

a way to use the TRENDLIST and PEEKLIST commands in TRANS to group externals for trending and/or summing up variables by subtype.

Composition control

See "Setting composition at inflows" on page 139.

Temperature control

See "Setting temperature at inflows" on page 140.

Flow units

You can specify the flow at an external in a secondary flow unit. This can be used when a fluid that is entering the system has a significantly different flow rate when compared to other fluids—for example, a fluid with a flow of .1 gal/min, while the other system flows are in MB/D. This feature can be used to enter trace components.

FLUID and INITIAL

The +FLUID and +INITIAL lines are used to set the composition that the node will use when the node flow is into the model. See "Setting composition at inflows" on page 139.

Example input

Example 1

A pressure-controlled TAKE external named EXT1 is connected to a node named NV1. The initial pressure is 30 psig and the set point temperature is 60°F (note that this temperature does not affect the flowing temperature unless the flow is into the model).

```
E EXT1 NV1 TAKE 30. 0. 60.
```

Example 2

A flow-controlled SALE external named EDEL is connected to a valve named CV240. The initial flow rate is 50 MB/D (out of the model) and the set point temperature is 60°F (note that this temperature does not affect the flowing temperature unless the flow is into the model).

Example 3

```
E EDEL +CV240 SALE 0. 50. 60.
```

A flow-controlled TAKE external named INLET is connected to a node named NODE1. The initial flow is 1000 and the temperature of the fluid entering the system is 60°F. The rate of flow change for a flow ramp is to be a value of 800 flow units/minute:

```
E INLET NODE1 TAKE 0. 1000. 60. 0 800.
```

Example 4

A pressure-controlled SALE external named OUT1 is connected to a node named ND11. The initial pressure is 300 psig and the set point temperature is 60°F (note that this temperature does not affect the flowing temperature unless the flow is into the model). The pressure is set to 175 psig when the simulation starts to run:

```
E OUT1 ND11 SALE 300. 0. 60.
+ POKE P+ = 175
```

E P-CONTROL

Peek and Poke Attributes

Note: Externals have only a + side. To reference an external as a node connection for some other element, precede the external name with the + symbol.

E NAME CNC **P-CONTROL** P_1 P_2 [T+]

Field	Units Key	Description
NAME	n/a	Unique name of the external.
CNC	n/a	Connection point or named node.
P_1	PRESSURE	Lower end of the pressure control range.
P_2	PRESSURE	Upper end of the pressure control range.
T+	TEMPERATURE	Temperature of the fluid if the flow is into the model. {temperature from thermal mode line}

Description

The P-CONTROL external type is used to model a supply or delivery point that is pressure-controlled. Flow enters or leaves the system to meet pressure requirements controlled via a controlling actuator. ("Actuator (A)" on page 400.)

Pressure is given by:

$$P = P_1 + X (P_2 - P_1)$$

where:

X is the actuator output (0 < X < 1).

Take note

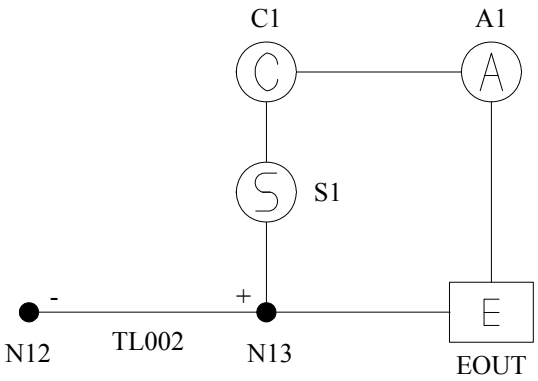
Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Subtype attribute

A peek attribute, :STYP, has been added to all externals. This attribute distinguishes the external type in a Show window, report, or text display. For a Pressure Control external, the :STYP peek attribute is P-CONTROL. Also, this peek attribute provides a way to use the TRENDLIST and PEEKLIST commands in TRANS to group externals for trending and/or summing up variables by subtype.

Example input



A pressure-controlled supply external named EIN is connected to a node named NOD3 that is the upstream connection for a pipe named TL001. The pressure range is from 100 to 500 psig. The temperature of the fluid entering the system through this external is 60°F. This external needs a control system defined with it. The control system consists of an actuator, a controller, and a sensor.

```
E EIN NOD3  P-CONTROL 100. 500. 60.
A A1 C C1 P EIN X(V)2 0. 1. V1
C C1 S1:OUT 250 1 0. 0. 0. -800.
S S1 -TL001 PRES V(Z)2 0. 500. 0. 500.
```

E Q-CONTROL

Peek and Poke Attributes

Note: Externals have only a + side. To reference an external as a node connection for some other element, precede the external name with the + symbol.

INPREP

```
E NAME CNC Q-CONTROL FLOW1 FLOW2 [T+]
```

Field	Units Key	Description
NAME	n/a	Unique name of the external.
CNC	n/a	Connection point or named node.
$FLOW_1$	FLOW	Lower end of the flow control range.
$FLOW_2$	FLOW	Upper end of the flow control range.
T+	TEMPERATURE	Temperature of the fluid if the flow is into the model. {temperature from thermal mode line}

Description

The Q-CONTROL external type is used to model a supply or delivery point that is flow-controlled. The flow for this external is controlled via a controlling actuator. (See "Actuator (A)" on page 400.)

Flow is given by:

$$FLOW = FLOW_1 + X\ (\ FLOW_2 - FLOW_1\ )$$

where:

X is the output of the controlling actuator.

Flow is positive if into the model and negative if out of the model.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Do not connect to closed valve

It is recommended that the flow controlled external not be tied to a system in such a way that it can ever be dead-ended into a closed valve. Such a structure can lead to unbounded pressures.

Initialization

Actuators are initialized at zero value, so that the initial flow from the flow controlled external is $FLOW_1$. If the system is to be initiated at zero flow, then $FLOW_1$ must be entered as zero.

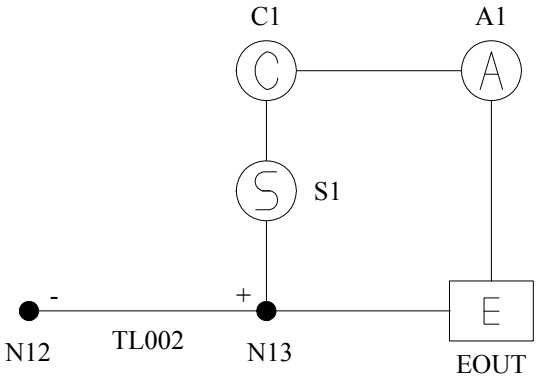
Subtype attribute

A peek attribute, :STYP, has been added to all externals. This attribute distinguishes the external type in a Show window, report, or text display. For a flow control external, the :STYP peek attribute is Q-CONTROL. Also, this peek attribute provides a way to use the TRENDLIST and PEEKLIST commands in TRANS to group externals for trending and/or summing up variables by subtype.

Isolated externals

If two or more flow controlled externals are isolated by valves, headers, and pumps/compressors, (i.e., without pipes or pressure controlled externals), so that their flows must algebraically sum to zero at every instant, the physical system defined generates a singular algebraic system. This leads to unexpected diagnostic error messages during simulation.

Example input



A flow-controlled delivery external named EOUT is connected to a node named N13 and to the downstream side of a pipe named TL002. The flow range is from 0 to 8000 (note that the delivery flow values are negative). The temperature of the fluid entering the system through this external is 60°F. Because this external is controlled by an actuator, the example needs a control system defined with it. The control system consists of an actuator, a controller, and a sensor.

```
E EOUT N13  Q-CONTROL -0. -8000. 60.
A A1 C C1 Q EOUT X(V)2 0. 1. V1
C C1 S1:OUT 250 1 0.3 1. 0. -800.
S S1 +TL002 FLOW V(Z)2  0. 10000. 0. 10000.
```

E Q(P)

Peek and Poke Attributes

Note: Externals have only a + side. To reference an external as a node connection for some other element, precede the external name with the + symbol.

INPREP

```
E NAME CNC CURV FLOW1 FLOW2 P1 P2 [T+]
```

Field	Units Key	Description
NAME	n/a	Unique name of the external.
CNC	n/a	Connection point or named node.
CURV	n/a	Name of a data curve defining flow as a function of pressure: Q = Q(P) (If function is a straight line, enter Q(P)2.)
$FLOW_n$, P_n	FLOW	Corresponding flow (Q) and pressure (P) values. Used to scale the curve.
T+	TEMPERATURE	Temperature of the fluid if the flow is into the model. {temperature from thermal mode line}

Description

The flow as a function of pressure external type is used to model external flow entering or leaving the system, such as a supply well, sale point, injection well, pipe rupture, etc., where flow is controlled by back-pressure instead of being regulated by control instrumentation. To define flow as a function of pressure, enter the name of a data curve (type Q(P)) in CURV). If the function is linear, simply enter Q(P)2 and then enter corresponding values of pressure and flow in P_n and $FLOW_n$.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Flow sign convention

Flow is positive if it is into the model. To be physically stable, a Q(P) function must have a strictly negative slope (i.e., $(FLOW_2 - FLOW_1)/(P2 - P1) < 0$). The function must pass through a 0 flow point. It should be defined over some positive pressure interval that does not include the bubble point, but that does include all of the pressures to be encountered in the simulation. Simulations that lead to pressures outside the range of definition of Q(P) produces indeterminate results, and may lead to singular systems of algebraic equations and abnormal termination of the simulation.

Subtype attribute

A peek attribute, :STYP, has been added to all externals. This attribute distinguishes the external type in a Show window, report, or text display. For a Flow as a Function of Pressure external, the :STYP peek attribute is QOFP. Also,

this peek attribute provides a way to use the TRENDLIST and PEEKLIST commands in TRANS to group externals for trending and/or summing up variables by subtype.

Example input

A flow as a function of pressure external named EXT0002 is connected to a pipe named TL002. The external has a linear curve defining flow as a function of pressure. The flow endpoints are 0 and 600 with corresponding pressure endpoints of 320 and 0. The temperature of the fluid is 60°F:

```
E EXT0002 +TL002 Q(P)2 600  0  0 320
```

E AIR-INLET Valve

E <NAME> <CNC> **AIR-INLET** [THROAT] [C_{in}] [C_{out}] [P_{atm}] [T_{atm}]
[+ [MW_{air}] [k] [T_{pipe}] [ΔP_{open}]

Field	Units Key	Description
<NAME>	n/a	Unique name of the air valve.
<CNC>	n/a	The single tie-point connection for the air valve.
THROAT	AREA	Valve throat area {0.1 in^2}.
C_{in}	n/a	Inlet throat convergence factor. {1}
C_{out}	n/a	Outlet throat convergence factor. {1}
P_{atm}	PRESSURE	Ambient Atmospheric Air Pressure. {14.7 psia}
T_{atm}	TEMPERATURE	Ambient Atmospheric Air Temperature. {60°F}
MW_{air}	NA	Molecular weight of atmospheric air. {29 lbm/lb-mol}
k	n/a	Heat capacity ratio, Cp/Cv. {1.4}
T_{pipe}	TEMPERATURE	Temperature of air in the pipe. {60°F}
ΔP_{open}	PRESSURE	Pressure differential to open valve for air intake, psi. {2 psi}

Description

The Air Valve (E) directive is used to model a special type of valve that admits air into a pipeline (normally a water line) to keep a pressure vacuum from collapsing the line.

Take note

Isentropic flow equations

Two equations describe flow through an air valve. One of these is for flow into the pipe from outside, q_1; the other is for flow out of the pipe to the outside, q_2.

When P/P_{atm} is less than 1 (i.e., the pressure inside the pipe is less than the ambient pressure), flow occurs from the outside into the pipe. Flow of air is initiated only when $P_{atm} - P > \Delta P_{open}$. In this case the flow q_1 is given by:

$$q_1 = 60 \ \text{THROAT} \ C_{in} \ P_{atm} \ \sqrt{2 g_c \ \gamma \ MW_{air} \ k \ / \ (\ (\ k-1 \ ) \ R \ (\ T_{atm} + 460 \ ) \ )}$$

where:

γ	=	$(P/P_{atm})^{2/k} - (P/P_{atm})^{(k+1)/k}$
q_1	=	rate of air flow into pipe, lbm/min
P	=	Pressure of air in the pipe, psia
gc	=	32.174 lbm x feet/(lbf x sec^2)
R	=	Gas constant, 1545 foot - lbf/lb - mol/°R

In these units, lbm is pounds mass, lbf is pounds force.

When P/P_{atm} approaches 1, γ and thus q_1 approach zero.

When P/P_{atm} exceeds 1 (i.e., the pressure inside the pipe exceeds the ambient pressure), flow then occurs from the pipe to the outside. In this case the flow q_2 is given by:

$$q_2 = 60 \ \text{THROAT} \ C_{out} \ P \sqrt{2g_c \ \xi \ MW_{air} \ k \ / \ \left(\left(k - 1 \right) \ R \ \left(T_{pipe} + 460 \right) \right)}$$

where:

$$\xi = \left(P / P_{atm} \right)^{-2/k} - \left(P / P_{atm} \right)^{-(1+k)/k}$$

Equations q_1 and q_2 are valid for pressures satisfying:

$$r_c \leq P/P_{atm} \leq 1/r_c$$

where:

$$r_c = \left(2/\left(1 + k \right) \right)^{k/(k-1)}$$

This limiting set of ratios corresponds to the ratios at which the air velocity through the valve becomes sonic. For pressures outside the range given in (1), the inflow takes on the value given by q_1 when $P/P_{atm} = r$, and the outflow takes on the value given by q_2 when $P/P_{atm} = 1/r_c$.

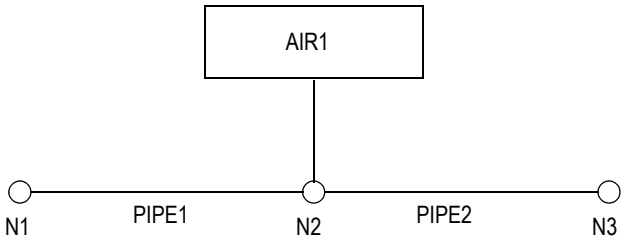
Sub type suffix

A peek suffix, :STYP, has been added to all Externals. This suffix lets the user recognize the type of the External directly when displaying the External on a show screen, a report or a text display. For an Air Valve External, the :STYP peek suffix is AIR-INLET. Also, this peek suffix provides a way to use the TRENDLIST and PEEKLIST directives in TRANS to group Externals for trending and/or summing up variables by sub type.

General considerations

- The AIR VALVE should be located at a high point in the model.
- The air pocket does not move along the pipe; it is simply let in then let out again when the pressure returns to normal.
- The effect of the air inside the pipe is modeled by adjusting the bulk modulus of the liquid.

Example input



An air valve named AIR1 is connected to a node named N2 that is connected to the downstream end of a Transfer Line named PIPE1. The valve throat area is 2.3 square inches. The inlet and outlet throat convergence factors are .92 and .89 respectively. All other values are the default values.

```
E AIR1 N2 AIR-INLET 2.3  0.92  0.89
```

E SURGETANK

Peek and Poke Attributes

Note: Externals have only a + side. To reference an external as a node connection for some other element, precede the external name with the + symbol.

INPREP

```
E NAME CNC SURGETANK TYPE ID LEVEL PINIT HEIGHT
+ CLEN CDIAM FF [K1] [K2] [C1] [C2] [NTANK]
```

Field	Units Key	Description
NAME	n/a	Unique name of the surge tank.
CNC	n/a	Connection point or named node.
TYPE	n/a	SPHR (spherical), VCYL (vertical cylinder), or HCYL (horizontal cylinder).
ID	DIAMETER	Inside diameter of the tank.
LEVEL	ELEVATION	Initial level of the liquid in the tank at simulation start. For HCYL and SPHR, LEVEL ≤ DIAM.
PINIT	PRESSURE	Initial pressure of gas in the tank.
HEIGHT	ELEVATION	Maximum height or length of the tank (omit only if SPHR is designated).
CLEN	LENGTH.HEADER	Length of the connecting pipe from CNC to the tank.
CDIAM	DIAMETER	Inside diameter of the connecting pipe.
FF	n/a	Moody friction factor.
K_1, K_2	n/a	Entrance and exit loss at the top of the connecting pipe, respectively {1}
C_1, C_2	n/a	Exit and entrance loss at the bottom of the connecting pipe, respectively. {1}
NTANK	dimensionless	Polytropic exponent for compression in the tank. {1.3}

Description

The pressurized surge tank external type models a trapped-gas surge tank for liquid pipeline systems. This pressurized surge tank takes into account the polytropic compression of the trapped gas blanket along with entrance and exit line losses and head.

Take note

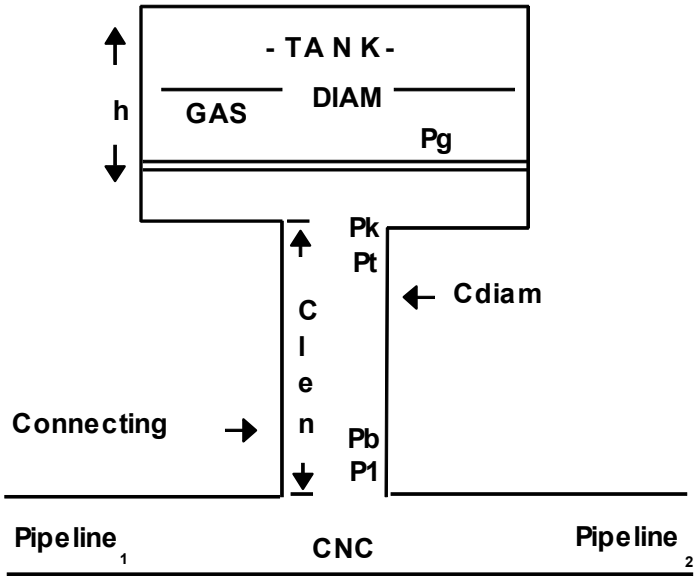
Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Subtype attribute

The :STYP attribute distinguishes the external type in a Show window, report, or text display. For a Surge Tank external, the :STYP peek attribute is SURGETANK. Also, this peek attribute provides a way to use the TRENDLIST and PEEKLIST commands in TRANS to group externals for time plots and/or summing up variables by subtype.

Tank Schematic for Definition of Symbols



Note: Units shown in the following table are for the calculations. Input values correspond to the units key.

Symbol	Description
Pg	Pressure of gas in the tank, psia.
Pg(0)	Initial value for Pg: PINIT expressed in psia.
Pk	Pressure in liquid at the bottom of the tank, psia.
Pt	Pressure at the top of the connecting pipe, psia.
Pb	Pressure at the bottom of the connecting pipe, psia.
Pl	Pressure at the equipment end tied to the connecting point, CNC, psia.
Pl(0)	Initialization value for Pl.
Clen	Length of the connecting pipe, assumed vertical, ft.
h	Height of the liquid column in tank, ft.
h(0)	LEVEL.
Cdiam	Diameter (inside) of the connecting pipe, ft.
g	Acceleration due to gravity, ft/sec^2.
g_c	32.174x144 $(lbm - in^2)/(lbf - sec^2 - ft)$.
χ	$[Pl(0) - g\rho LEVEL+Clen/g_c] [V(0)]NTANK$.
V	Volume of gas in the tank, ft^3.
V(0)	Initialization gas volume at liquid height h(0).

Symbol	Description
v	Superficial velocity of liquid in the connecting pipe, positive for flow into tank, ft/sec.
ρ	Liquid density, lbm/ft^3
t	Time, in seconds. When used as a subscript, implies derivative with respect to time.
FF	Friction.

The dynamics of the surge tank function are simulated by solving the following system of equations associated with the foregoing variables:

$$P\ V^{NTANK} = \xi$$

The tank is simulated as if there was a quick-acting vapor valve to prevent loss of the gas blanket at low pressure.

$$Pk = Pg + g\ \rho\ h / g_c$$

$$P_t - Pk = K\ \rho\ v\ |\ v\ |\ (\ 2g_c\ )$$

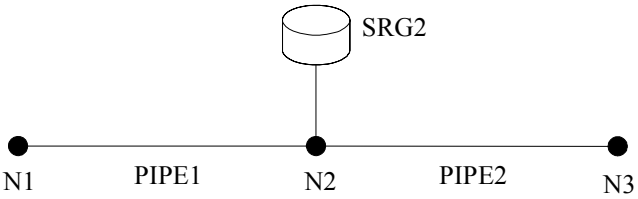
K = K1 if v > 0; otherwise K = K2

$$Pt = Pb - FF\ Clen\ \rho\ v\ |\ v\ |\ /\ (\ 2g_c\ Cdiam\ ) - g\ Clen\ \rho\ /\ g_c - \rho v1\ /\ g_c$$

$$P1 - Pb = C\ \rho\ v\ |v|\ /\ (2g_c)$$

C = C2 if v > 0; otherwise C = C1

Example input



A vertical cylinder surge tank named SRG2 is connected to a node named N2 that is connected to the downstream end of a pipe named PIPE1. The tank is a vertical cylinder with an inside diameter of 72 inches, a height of 10 feet, an initial liquid level of 2 feet, and an initial tank gas pressure of 100 psig. The connecting pipe length is 3 feet with an inside diameter of 6 inches and a friction factor of 0.014:

```
E SRG2 N2 SURGETANK VCYL 72 2 100 10
+ 3 6 .014
```

Tank (TK)

Peek and Poke Attributes

INPREP

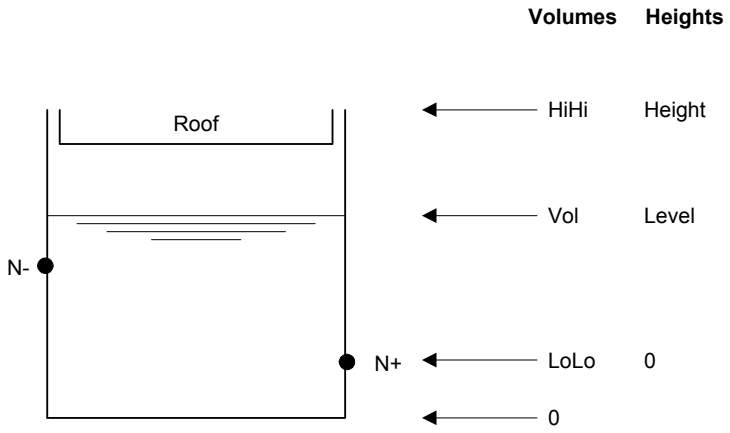
```
TK NAME FROM [TO]
[+ HEIGHT HT]
[+ ELEV EFROM [ETO]]
[+ FLUID FINIT]
[+ HIHI HIHI]
[+ HI HI]
[+ LO LO]
[+ LOLO LOLO]
[+ TEMPERATURE T]
[+ INITIAL.VOLUME VINIT]
[+ VOLUME.CURVE CURVE]
[+ CHECK]
[+ SPILLOVER]
[+ ROOF P-ROOF]
[+ IN.CV Cv_in]
[+ OUT.CV Cv_out]
```

Field	Units Key	Description
NAME	n/a	Unique name of the tank.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
HT	ELEVATION	Height of the tank in respect to the discharge node. {40 ft}
EFROM	ELEVATION	Elevation at the from-end.
ETO	ELEVATION	Elevation at the to-end.
FINIT	n/a	Named fluid initially in the tank.
HIHI	VOLUME	Capacity of the tank. {40 MB}
HI	VOLUME	Nominal full volume. {40 MB}
LO	VOLUME	Nominal empty value. {0 MB}
LOLO	VOLUME	Volume at outlet height. {0 MB}
T	TEMPERATURE	Temperature of fluid. {68 °F}
VINIT	VOLUME	Initial volume. {0 MB}
CURVE	n/a	Name of a data curve showing volume as a function of level. Curve type is TKVL. {Linear} See “Data curve (D)” on page 418.
CHECK	n/a	Indicates presence of check valves that restrict all flow to be in the nominal direction. A two-connection tank may have either zero or two check valves. {no} **Note:** The Check option is not available in Model Builder.

Field	Units Key	Description
SPILLOVER	n/a	Indicates the tank will spill excess in-flow and records the accumulating spill volume for reporting purposes. A non-spillover tank prevents spilling. {no}
P-ROOF	ΔPRESSURE	Fixed pressure (above gauge) on top of the fluid {0}
Cv_{in}	VALVE.COEFF	Valve coefficient controlling pressure loss associated with flow into the tank. {3800 GAL/MIN-PSI.5}
Cv_{out}	VALVE.COEFF	Valve coefficient controlling pressure loss associated with flow out of the tank. {5200 GAL/MIN-PSI.5}

Description

The tank (TK) can model a tank with fixed or variable area and assumes a fixed pressure exerted on top of the fluid. You can specify optional check valves and account for pressure losses through the valves. All tanks prevent air from flowing into the nodes. You may also specify whether the tank allows spillover or not. A tank has a fixed (but pokable) fluid temperature. The following diagram explains the nomenclature used for tanks.



Tank nomenclature

The tank geometry is specified by a curve of volume as a function of height (TKVOL). See "Data curve (D)" on page 418. If the curve is shorter than the tank, the extrapolation uses a constant area (constant dVolume/dLevel). A poke that changes HiHi does not change the shape of the curve, it just changes how much of the curve is used (or how much it is extrapolated). Alternatively, you may omit the volume curve and specify the capacity of the tank (HIHI) and the height of the tank (HT) to simulate a fixed-area tank.

Tanks may have a status of EMPTY, FULL, EMPTYING, FILLING, STATIC, OVER_EMPTY, OR SPILLING.

Take note

Common syntax for equipment

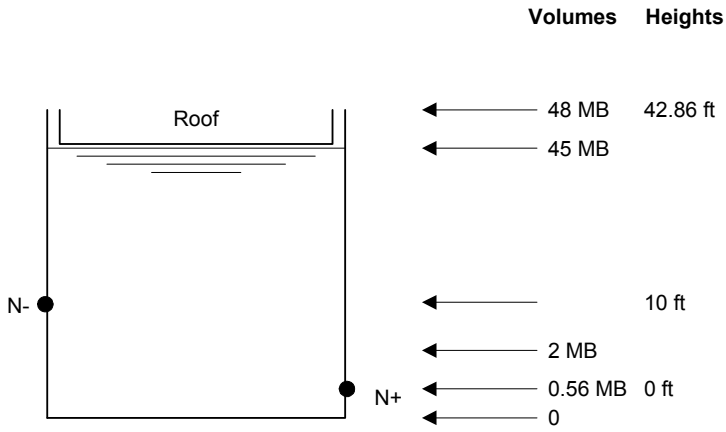
Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Curves

Tank curves should not start at 0, 0. Ending volume needs to be a small positive value, such as 0.001.

Example input

Model a water tank with the configuration shown in the following diagram.



The pressure loss at the inlet is 3000 GAL/MIN-PSI.5 and the volume of the tank is controlled by a data curve named TANK1.

```
TK TANK N8
+ HEIGHT 42.86
+ ELEVATION 610 600
+ HIHI 48
+ HI 45
+ LO 2
+ LOLO 0.56
+ ROOF 1.5
+ VOLUME.CURVE TANK1
+INITIAL.VOLUME  45
+IN.CV 3000
+FLUID WATER
```

Block valve (BV)

Peek and Poke Attributes

INPREP

```
BV FID FROM TO CGMIN CGMAX TT FR
```

Field	Units Key	Description
FID	n/a	Unique name of the block valve.
FROM	n/a	From-node or upstream connection point.
TO	n/a	To-node or downstream connection point.
CGMIN	VALVE.CG	Valve coefficient at full-close position. {0}
CGMAX	VALVE.CG	Valve coefficient at full-open position. {1,000,000 ft^3/hr-psi}
TT	MINUTES	Valve travel time from fully closed to fully open (or vice versa).
FR	n/a	Initial fraction open (stem position). {0}

Description

The block valve (BV) element models valves that are set to a given position.

Block valves are manually operated using OPEN, CLOSE, and STOP interactive commands and/or INTRAN commands.

The block valve is similar to the SynerGEE General Valve (GV). The BV is similar to the SPS B element, but it uses a gas equation and a gas valve coefficient.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Flow transmissibility

$$Q = 24Cg\sqrt{(P_1 - P_2)P_2} \quad \text{when } \frac{P_1}{P_2} \le 1.89 \quad \text{(subsonic)}$$

$$Q = 12Cg(P_1) \quad \text{when } \frac{P_1}{P_2} \rangle\ 1.89 \quad \text{(sonic)}$$

where:

Q	=	Flow rate (scfd)
C_g	=	Regulator constants (cfh/psi)
P_1	=	Upstream pressure (psia)
P_2	=	Downstream pressure (psia)

Valve Coefficient, CG

The valve coefficient, C_G, is calculated by interpolating the valve opening or closing curve, depending on whether the valve is opening or closing, respectively.

Initial fraction

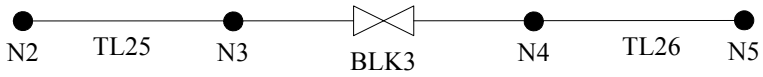
FR ranges from 0 to 1. FR of 0 means closed (using CGMIN) and 1 means open (using CGMAX). FR = (C_G – CGMIN) / (CGMAX – CGMIN).

Open and close time

The travel time, the time to open the valve, and the time to close the valve must be equal.

Example input

A block valve named BLK3 is installed between pipes named TL25 and TL26. The valve has a travel time of 3.5 min and valve coefficients of 32433 at full-open and 0 at full-close.



Example 1

Specify the input for valve BLK3 with a linear C_G curve and starting from a closed position.

```
BV BLK3 N3 N4 0 32433 3.5
```

Example 2

Specify the input for valve BLK3 with a linear C_G curve and starting from an open position.

```
BV BLK3 N3 N4 0 32433 3.5 1
```

Block valve (B)

Peek and Poke Attributes

```
B NAME FROM TO CRV-O CRV-C [Cvc] Cvo t [FR]
[+ HEADER ΔP Q]
[+ CHECK ΔP Q]
```

Field	Units Key	Description
NAME	n/a	Unique name of the block valve.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
CRV-O	n/a	Name of a data curve showing valve coefficient vs. time for valve opening, $C_v(t)$. Curve type is OPEN. For more information, see "Data curve (D)" on page 418. For a straight line enter OPEN2.
CRV-C	n/a	Name of a data curve showing valve coefficient vs. time for valve closing, $C_v(t)$. Curve type is CLSE. For more information, see "Data curve (D)" on page 418. For a straight line enter CLSE2.
C_{vc}	VALVE.COEFF	Valve coefficient at full-close position. {0}
C_{vo}	VALVE.COEFF	Valve coefficient at full-open position.
t	TIME	Valve travel time from fully closed to fully open (or vice versa).
FR	n/a	Initial fraction open (stem position). {0}
ΔP	ΔPRESSURE	Pressure drop used to size an optional series header or check valve
Q	FLOW	Flow used to size an optional series header or check valve.

Description

The block valve (B) is used to model a block valve. A header and/or a check valve may be modeled in series with the block valve.

Block valves are manually operated using OPEN, CLOSE, and STOP interactive commands and/or INTRAN commands.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Data curves

If you enter * for C_{vc}, C_{vo}, or t, the value from the data curve is used. If you enter a nonzero value, the data curve is adjusted to correspond to the value specified.

You must enter values when using the predefined curves.

Peekable and pokable

The open and close valve coefficients (C_{vo} and C_{vc} respectively) and the valve travel time are peekable and pokable. When the C_{vc} and/or the C_{vo} value(s) is poked, the corresponding close and/or open C_v curve(s) is adjusted to suit the new value(s).

Fraction is pokable.

Note: If the optional + HEADER and/or the + CHECK integrated elements are entered, the Cvo value that is shown during TRANS will be less than the value that is entered for the block valve. This is because the value that is shown during TRANS is the effective Cvo (which accounts for the block valve and the integrated elements; because this is “one” element). The value input for the block valve accounts only for the block valve. If the Cvo value is poked interactively, the new value is the effective C_{vo}.

Series header

The + HEADER line is optional. This means that a header is to be modeled in series with the block valve. The ΔP and Q entries on this line and the density at custody transfer conditions are used to size the header. Because custody conditions are used, the pressure loss will not correspond for high pressure gas.

Series check valve

The + CHECK line is optional. This means that a check valve is to be modeled in series with the block valve. The check valve modeled in this way opens and closes instantaneously to prevent back flow. The ΔP and Q entries on this line and the density at custody transfer conditions are used to size the open check valve. Because custody conditions are used, the pressure loss will not correspond for high-pressure gas.

Flow-transmissibility characteristics

Transmissibility for flow across a block valve is expressed in terms of valve coefficient (C_v) using the formulas:

$$\text{Liquid}: \quad Q = C_v\,(\rho/\rho_b)\sqrt{\Delta P\,\rho_w/\rho}$$

where:

Q	=	Flow rate at base (custody transfer) density, ρ_b
ρ	=	Density at flowing conditions
ΔP	=	Pressure drop
ρ_w	=	Density of water
ρ_b	=	Density at custody conditions

Note: The liquid equation is accurate only in the range: $0.25 < \Delta P < 10000$ psi.

$$\text{Gas}: \quad Q = C_v\,N_1\sqrt{(P_u^2 - P_d^2)/(Z\,G\,T)}$$

where:

Q	=	Flow
Z	=	Upstream compressibility factor
G	=	Specific gravity of gas relative to air (MWgas/MWair)

T	=	Absolute upstream temperature
P_u	=	Absolute upstream pressure
P_d	=	Absolute downstream pressure
N_1	=	Units conversion (default $120\sqrt{^\circ R / PSI}$)
		(the default flow units are those of VALVE.COEFF).

If the + HEADER is present, then the additional pressure drop across the header section, ΔP_h, is given by:

$$\Delta P_h = \frac{Q^2}{C_h}$$

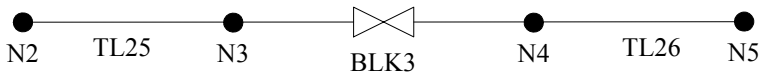
where:

C_h is a transmissibility chosen to match the data on the + HEADER line.

If the + CHECK is present, then the additional pressure drop across the open check valve section is given by a similar formula. The transmissibility factor is used to adjust the valve coefficient.

Example input

A block valve named BLK3 is installed between pipes named TL25 and TL26. The valve has a travel time of 3.5 min and valve coefficients of 32433 at full-open and 0 at full-close.



Example 1

Specify the input for valve BLK3 with a linear C_v curve and starting from a closed position:

```
B BLK3 N3 N4 OPEN2 CLSE2 0 32433 3.5
```

Example 2

Specify the input for valve BLK3 with a linear C_v curve and starting from an open position:

```
B BLK3 N3 N4 OPEN2 CLSE2 0 32433 3.5 1
```

Example 3

Specify the input for valve BLK3 with C_v curves specified by the data curves BVOPEN and BVCLOS, using default CVC and CVO. Travel time is 0 and starting from a closed position. This valve also has a check valve modeled in series with the block valve with a 10 psig pressure drop at a flow rate of 800:

```
B BLK3 N3 N4 BVOPEN BVCLOS 0 1e6
+ CHECK 10 800
```

Check valve (CV)

Peek and Poke Attributes

```
CV FID FROM TO CRV-O CRV-C [CGMIN] CGMAX TT FR
```

Field	Units Key	Description
FID	n/a	Unique name of the check valve.
FROM	n/a	From-node name or upstream connection point.
TO	n/a	To-node name or downstream connection point.
CRV-O	n/a	Name of a data curve showing valve coefficient vs. time for valve opening. Curve type is OPEN. For more information, see "Data curve (D)" on page 418. For a straight line enter OPEN2.
CRV-C	n/a	Name of a data curve showing valve coefficient vs. time for valve closing. Curve type is CLSE. For more information, see "Data curve (D)" on page 418. For a straight line enter CLSE2.}
CGMIN	VALVE.CG	Closed valve coefficient (minimum). {0}
CGMAX	VALVE.CG	Open valve coefficient (maximum).
TT	MINUTES	Valve travel time from fully closed to fully open (or vice versa).
FR	n/a	Initial fraction open (stem position). {0}

Description

The check valve (CV) is used to model a check valve. It is similar to the SynerGEE General Valve (GV) with the CHECK option and similar to the BC element in SPS, except it uses a gas valve coefficient.

Flow-transmissibility characteristics

Transmissibility for flow across a check valve is expressed in terms of a gas valve coefficient (C_v) using the following formulas:

$$Q = 24Cg\sqrt{(P_1 - P_2)P_2} \quad \text{when} \quad \frac{P_1}{P_2} \leq 1.89 \ \text{(subsonic)}$$

$$Q = 12Cg(P_1) \quad \text{when} \quad \frac{P_1}{P_2} \rangle\ 1.89 \quad \text{(sonic)}$$

where:

Q	=	Flow rate (scfd)
C_g	=	Regulator constants (cfh/psi)
P_1	=	Upstream pressure (psia)
P_2	=	Downstream pressure (psia)

Take note

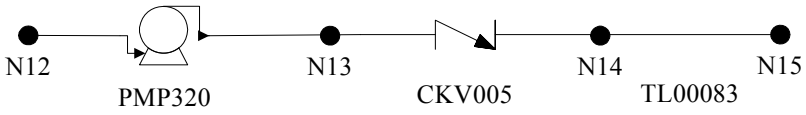
Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Open and close time

The travel time, the time to open the valve, and the time to close the valve must be equal.

Example input



Example 1

A check valve named CKV005 is connected to nodes named N13 and N14 and is installed between a pump named PMP320 and a pipe named TL00083. Opening and closing flow-resistance characteristics of the check valve are defined by data curves named CKOPN and CKCLS with a travel time of 0. The valve is initially closed.

```
CV CKV005 N13 N14 CKOPN CKCLS 0 1e6
```

Check valve (BC)

Peek and Poke Attributes

```
BC NAME FROM TO CRV-O CRV-C [Cvc] Cvo t FR
[+ HEADER ΔP Q ]
```

Field	Units Key	Description
NAME	n/a	Unique name of the check valve.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
CRV-O	n/a	Name of a data curve showing valve coefficient vs. time for valve opening. Curve type is OPEN. For more information, see "Data curve (D)" on page 418. For a straight line enter OPEN2. {OPEN2]
CRV-C	n/a	Name of a data curve showing valve coefficient vs. time for valve closing. Curve type is CLSE. For more information, see "Data curve (D)" on page 418. For a straight line enter CLSE2. {CLSE2}
C_{vc}	VALVE.COEFF	Valve coefficient at full-close position.
C_{vo}	VALVE.COEFF	Valve coefficient at full-open position.
t	TIME	Valve travel time from fully closed to fully open (or vice versa).
FR	n/a	Initial fraction open (stem position). {0}
ΔP	ΔPRESSURE	Pressure drop used to size an optional series header.
Q	FLOW	Flow used to size an optional series header.

Description

The check valve (BC) is used to model a check valve. A header may be modeled in series with the check valve. Check valves open with positive ΔP and close with negative Q.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Data curves

If you enter * for C_{vc}, C_{vo}, or t, the value from the data curve is used. If you enter a nonzero value, the data curve is adjusted to correspond to the value specified.

You must enter values when using the predefined curves.

Peekable and pokable

The open and close valve coefficients (C_{vo} and C_{vc} respectively) and the valve travel time are peekable and pokable. When the C_{vc} and/or the C_{vo} value(s) is poked, the corresponding close and/or open C_v curve(s) is adjusted to suit the new value(s).

Note: If the optional + HEADER integrated element is entered, the C_{vo} value that is shown during TRANS will be less than the value that is entered for the check valve. This is because the value that is shown during TRANS is the effective C_{vo} (which accounts for the check valve and the integrated header; because this is "one" element). The value entered for the check valve accounts only for the check valve. If the C_{vo} value is poked interactively, the new value is the effective C_{vo}.

Status

All check valves start in the closed position, unless specified as open. If a check valve is added to a model, it will start in the closed position even if an archived state indicates flow.

Series header

The + HEADER is optional. This means that a header is to be modeled in series with the check valve. The ΔP and Q entries on this line and the base density at custody transfer conditions are used to size the header. Because custody conditions are used, the pressure loss will not correspond for high pressure gas.

Flow-transmissibility characteristics

Transmissibility for flow across a check valve is expressed in terms of valve coefficient (C_v) using the following formulas for liquid or gas:

Liquid

$$Q = C_v \, (\rho / \rho_b) \sqrt{\Delta P \, \rho_w / \rho}$$

where:

Q	=	Flow rate at base (custody transfer) density, ρ_b
ρ	=	Density at flowing conditions
ΔP	=	Pressure drop
ρ_w	=	Density of water
ρ_b	=	Density at custody conditions

Note: The liquid equation is accurate only in the range $0.25 < \Delta P < 10000$ psi.

Gas

$$Q = C_v \, N_1 \sqrt{\left(P_u^2 - P_d^2\right) / (Z G T)}$$

where:

Q	=	Flow
Z	=	Upstream compressibility factor
G	=	Specific gravity of gas relative to air (MWgas/MWair)
T	=	Absolute upstream temperature
P_u	=	Absolute upstream pressure
P_d	=	Absolute downstream pressure
N_1	=	Units conversion (default $120\sqrt{^\circ R / PSI}$)

(the default flow units are those of VALVE.COEFF).

If the + HEADER line is present, then the additional pressure drop across the header section, DP_h, is given by:

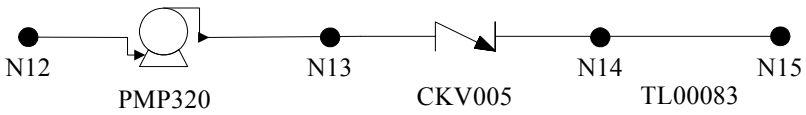
$$\Delta P_h = \frac{Q^2}{C_h}$$

where:

C_h is a transmissibility chosen to match the data on the + HEADER line.

The transmissibility factor is used to adjust the valve coefficient.

Example input



Example 1

A check valve named CKV005 is connected to nodes named N13 and N14 and is installed between a pump named PMP320 and a pipe named TL00083. Opening and closing flow-resistance characteristics of the check valve are defined by data curves named CKOPN and CKCLS with a travel time of 0. The valve is initially closed.

```
BC CKV005 N13 N14 CKOPN CKCLS 0 1e6
```

Example 2

In this example, the scenario is the same as in Example 1, except you want to specify linear opening and closing valve coefficient as a function of time, and a header is modeled in series with the check valve. In addition, the valve is initially open. The header has an initial pressure drop of 10 psig with a flow rate of 20. The check valve fully opened valve coefficient is 10000 and the fully closed coefficient is 0.15:

```
BC CKV005 N13 N14 OPEN2 CLSE2 0.15 10000. 0.05 1
+ HEADER 10 20
```

Control valve (V)

Peek and Poke Attributes

INPREP

```
V NAME FROM TO CRV-CX [Cvc] Cvo [CRV-FD ΔPX]
[+ HEADER ΔP Q]
[+ CHECK ΔP Q]
```

Field	Units Key	Description
NAME	n/a	Unique name of the control valve.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
CRV-CX	n/a	Name of a data curve defining C_v vs. stem position. Curve type is C(X). For more information, see "Data curve (D)" on page 418. If linear, enter C(X)2. For equal percentage valves, enter EQPC.
C_{vc}	VALVE.COEFF	Valve coefficient at full-close position. {0}
C_{vo}	VALVE.COEFF	Valve coefficients at full-open position.
CRV-FD	n/a	Name of a data curve defining valve correction coefficient vs. pressure drop, f(ΔP). Curve type is F(D). For more information, see "Data curve (D)" on page 418. {default correction coefficient =1} **Note:** To apply a multiplier for pressure drop values (DPX) in Model Builder, right-click on the CRV-FD text box and select Properties.
ΔPX	n/a	Multiplier for pressure drop values if a data curve was named in the CRV-FD. {1}
ΔP	ΔPRESSURE	Pressure drop used to size an optional series header or check valve.
Q	FLOW	Flow used to size an optional series header or check valve.

Description

The control valve (V) is used to model an actuator-controlled valve, which can be used to throttle flow for maintaining a set point. A header and/or a check valve may be modeled in series with the control valve.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Valve coefficient as a function of stem position

Control valves are modeled by varying the transmissibility coefficient as a function of the valve stem position, X. The stem position, X, is the output of an actuator and is between 0 and 1.

Equal-percentage valves

This valve approximates the equal percentage behavior that actual valves display. The equal percentage behavior is valid for stem positions greater than 0.2, and approximates C as the stem moves from 0.2 to 0 in a smooth equal percentage manner. C_{vc} for this valve must be positive, then $C_v(X)$ is given by:

$$C_v(X) = A \bullet \exp(kX) + B$$

where:

$$A = \frac{C_{vo} - C_{vc}}{\exp^k - 1}$$

$$B = C_{vc} - A$$

$k = 3.448$

Data curves

If you enter * for C_{vc}, C_{vo}, or t, the value from the data curve is used. If you enter a nonzero value, the data curve is adjusted to correspond to the value specified.

You must enter values when using the predefined curves.

Peekable and pokable

The open and close valve coefficients (C_{vo} and C_{vc} respectively) are peekable and pokable. When the C_{vc} and/or the C_{vo} value(s) is poked, the C_v vs. stem position curve is adjusted to suit the new value(s). The new C_v value(s) will be used in the computation.

Note: If the optional + HEADER and/or the + CHECK integrated elements are entered, the C_{vo} value that is shown during TRANS will be less than the value that is entered for the control valve. This is because the value that is shown during TRANS is the effective C_{vo} (which accounts for the control valve and the integrated elements; because this is “one” element). The value entered for the control valve accounts only for the control valve. If the C_{vo} value is poked interactively, the new value is the effective C_{vo} value (which accounts for the control valve and the integrated elements).

Series header

The + HEADER line is optional. This line means that a header is to be modeled in series with the control valve. The ΔP and Q entries on this line and the base density at custody transfer conditions are used to size the header. Because custody conditions are used, the pressure loss will not correspond for high pressure gas.

Series check valve

The + CHECK line is optional. This means that a check valve is to be modeled in series with the control valve. The check valve modeled in this way opens and closes instantaneously to prevent back flow. The ΔP and Q entries on this line and the base density at custody transfer conditions are used to size the open check valve. Because custody conditions are used, the pressure loss will not correspond for high pressure gas.

Valve correction coefficient

The f(ΔP) function is typically a slowly changing function of ΔP. This function, therefore, has not been linearized with ΔP in the simulation. If the function has a slope of large magnitude or if ΔP changes rapidly from step to step, a time step limitation (decreasing dtmax) should be invoked to assume stability. This generally is necessary only during periods of rapid change of ΔP across the valve.

Flow-transmissibility characteristics

Transmissibility for flow across a control valve are expressed using the following formulas:

Liquid

$$Q = C_v \left(\rho / \rho_b\right) f\left(\Delta P\right) \sqrt{\Delta P \rho_w / \rho}$$

where:

Q	=	Flow rate at base (custody transfer) density, ρ_b
ρ	=	Density at flowing conditions
ρ_b	=	Density at custody conditions
f(ΔP)	=	Valve correction coefficient from CRV-FD entry
ΔP	=	Pressure drop
ρ_w	=	Density of water

Note: This liquid equation is accurate only in the range eq 0.5<DP<10000psi.

The flow is equivalent to flow in consistent units at flowing conditions given by:

$$q = C_v \ f\left(\Delta P\right) \sqrt{\Delta P / SG}$$

where: SG = ρ/ρ_w

Liquid critical flow

Critical flow occurs when:

$$\Delta P \geq C_f^2 \left(P_u - P_o\right)$$

where:

C_f	=	The critical flow factor, 1
P_u	=	Absolute upstream pressure
P_o	=	Absolute bubble point pressure

When critical flow occurs, decreasing the downstream pressure does not increase the flow rate at a given valve position. The corresponding critical flow equation is:

$$Q = C_v (\rho / \rho_b) f (P_u - P_{ct}) \sqrt{(P_u - P_{ct}) \rho_w / \rho}$$

where:

P_u	=	Absolute upstream pressure
P_{ct}	=	Absolute cutoff pressure (when critical flow occurs)

Gas

$$Q = C_v N_1 f(\Delta P) \sqrt{(P_u^2 - P_d^2) / (ZGT)}$$

where:

Q	=	Flow, standard
N_1	=	Units conversion (default 120 / °R/PSI) (the default flow units are those of VALVE.COEFF)
f(ΔP)	=	Valve correction coefficient from CRV-FD entry
P_u	=	Absolute upstream pressure
P_d	=	Absolute downstream pressure
Z	=	Compressibility factor at upstream temperature and average pressure
G	=	Specific gravity of gas relative to air (MWgas/MWair)
T	=	Absolute upstream temperature

Gas critical flow

Critical flow (sonic flow) occurs when:

$$\Delta P \geq .5 \; C_f^2 \; P_u$$

where:

C_f	=	Critical flow factor, 1.0
P_u	=	Absolute upstream pressure

When critical flow occurs, decreasing the downstream pressure does not increase the flow through the valve at a given valve position. The corresponding critical flow equation is:

$$Q = C_v \; N_1 \; P_u \; f(\Delta P) \sqrt{\left(1 - .25 C_f^2 \right) / \; ZGT}$$

If the + HEADER line is present, then the additional pressure drop across the header, ΔP_h, is given by:

$$\Delta P_h = \frac{Q^2}{C_h}$$

where:

C_h is a transmissibility chosen to match the data on the + HEADER line.

This flow coefficient is combined with the user-defined C_{vc} or C_{vo} as:

$$c_v' = 1/\sqrt{(1/c_v^2)+(1/C_{vh}^2)}$$

where:

c_v'	=	Value used by model for C_{vc} or C_{vo}. This is also the value that may be peeked or poked in INTRAN
ρ_b	=	Fluid density at custody transfer conditions
c_v	=	INPREP value of C_{vc} or C_{vo}

If the + CHECK line is present, then the additional pressure drop across the open check valve section is given by a similar formula.

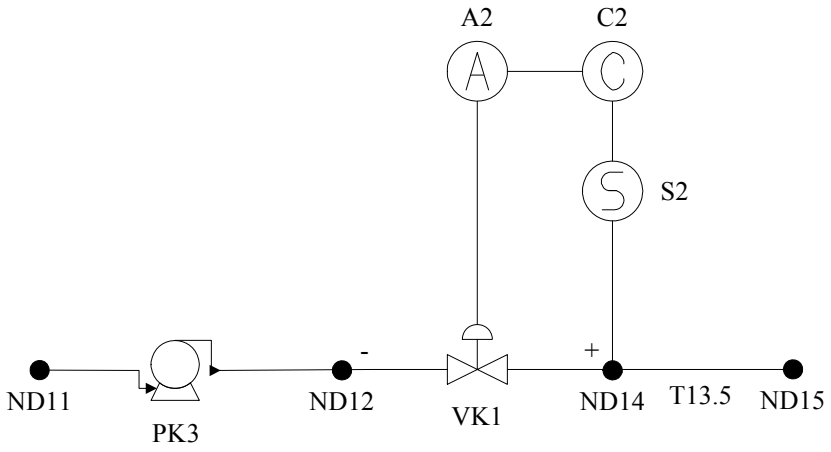
Control system

Because an actuator controls this valve, a control system must be defined with the control valve. Control systems typically consist of an actuator, a controller, and a sensor. The rate constant specified for the actuator is a limiting factor for the action time of the control valve.

Gas sonic flow

There may be instabilities in the solution if a valve in sonic flow isolates a single node or a small group of nodes from the system.

Example input



Example 1

A control valve named VK1 is connected to nodes named ND12 and ND14 and is installed between a pump named PK3 and a pipe named T13.5. Specify valve VK1 as an equal percentage valve with a closed valve coefficient of 0 and an open valve coefficient of 10000. A check valve is modeled in series with the control valve with an initial pressure drop of 0 and an initial flow rate of 2500:

```
V VK1 ND12 ND14 EQPC 0. 10000.
+ CHECK 0. 2500.
```

```
A A2 C C2 V VK1 X(V)2 0. 1. V15
C C2 S2:OUT 500 1. 0.3 1. 0. -1000
S S2 +VK1 PRES V(Z)2 0. 1000 0. 1000
```

Relief valve (RV)

Peek and Poke Attributes

INPREP

```
RV  NAME  FROM TO  CRV-CX  [Cvc]  Cvo
+ DIAMETER  DIAM
+ PRESSURE.SETPOINT  Po  [Pc]
[+ OPENING.CURVE  CRV-FTo  TTo  NCo]
[+ CLOSING.CURVE  CRV-FTc  TTc  NCc]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relief valve.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
CRV-CX	n/a	Name of a data curve defining C_v vs. stem position. Curve type is C(X). For more information, see "Data curve (D)" on page 418. If linear, enter C(X)2. For equal percentage valves, enter EQPC.
C_{vo}	VALVE.COEFF	Valve coefficient at full-open position.
C_{vc}	VALVE.COEFF	Valve coefficient at full-close position. {0}
DIAM	DIAMETER	Diameter at flange (used to calculate/display velocity).
P_o	PRESSURE	Opening pressure set point.
P_c	PRESSURE	Closing pressure set point. {P_o}
CRV-FT_o	n/a	Name of a data curve showing valve fraction vs. time for valve opening, TT_O. Curve type is F(T). For more information, see "Data curve (D)" on page 418. For a straight line, enter F(T)2. {F(T)2}
TT_o	TIME	Opening time. {.0083 min}
NC_o	ΔPRESSURE	Opening normalization constant. {0 = not used}
CRV-FT_c	n/a	Name of a data curve showing valve fraction vs. time for valve closing, TT_c. Curve type is F(T). For more information, see "Data curve (D)" on page 418. For a straight line, enter F(T)2. {F(T)2}
TT_c	TIME	Closing time. {.0083 min}
NC_c	ΔPRESSURE	Closing normalization constant. {0 = not used}

Description

The relief valve (RV) is an idealized element that controls the upstream pressure. The RV includes a valve only. The rest of the relief system (i.e., the equipment downstream of the valve, such as a pipe, tank, external, etc.) must be modeled independently.

The relief valve controls the upstream pressure by opening or closing based on the upstream pressure and its two set points. User-defined opening and closing curves control the rate at which the valves open and close. Valve opening and closing time is modeled such that the valve may experience over-pressure and/or under-pressure.

The following algorithm describes how the RV behaves. It does not describe details associated with switching an opening valve to closing, and vice-versa.

“Time spent opening” is a variable that holds the elapsed time that the RV has been opening. It is initially 0 and is reset to 0 whenever the valve starts closing. Similarly, “time spent closing” holds the elapsed time that the RV has been losing. It is initially 0 and is reset to 0 whenever the valve starts opening.

If inlet pressure > PO

1. Set “time spent closing” = 0
2. Add DT to “time spent opening”
3. Look up “time spent opening” in curve CRV-O to get FR. Look up new FR in curve CRV-CX to get Cv, but limit rate of change of FR to (PO - P)/NCO
4. ST will be either OPEN or OPENING
5. MODE will be PO or NCO

If inlet pressure < PC

1. Set “time spent opening” = 0
2. Add DT to “time spent closing”
3. Lookup “time spent closing” in curve CRV-C to get FR
4. Lookup new FR in curve CRV-CX to get Cv, but limit rate of change of FR to (P – PC)/NCC
5. ST will be either CLOSED or CLOSING
6. MODE will be PC, NCC, or CLOSED

If PC ≤ inlet pressure ≤ PO

1. Leave valve alone
2. ST will usually be STOPPED, but may be OPEN or CLOSED
3. MODE will be OKAY

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Example input

Add an equal percentage relief valve named BD that connects to nodes B and D. The valve coefficient at full open is 0.1 and the valve coefficient at full close is 1000. The diameter at the flange is 12 in., the opening pressure at the set point is 250 and the closing pressure at the set point is 225. The valve references an opening curve named RV_OPEN and a closing curve named RV_CLSE. The opening time is 1 second.

The data curve must be defined before use.

```
D RV_OPEN F(T)
+ 0.00 0.25 0.50 1.00
+    0    1    2    3

D RV_CLSE F(T)
+ 1.00 0.50 0.25 0.00
+    0    1    2    3
```

Then you may reference the data curve from the relief valve.

```
RV BD B D EQPC 0.1 1000
+ DIAMETER 12
+ PRESSURE.SETPOINT 250 225
+ OPENING.CURVE RV_OPEN 0.016667
+ CLOSING.CURVE RV_CLSE 0.016667
```

Grove G887 relief valve (V G887)

Peek and Poke Attributes

```
V NAME FROM TO G887 A1 b C1
+ [Pv] [R] [Ps0] [Ps1] * [ET] [A2] [C2]
```

Field	Units Key	Description
NAME	n/a	Unique name of the Grove G887 relief valve.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
A_1	VALVE.COEFF	Flow factor or constant used to calculate the valve coefficient for valve flow from the from-end to the to-end.
b	n/a	Exponent in the flow equation. (Minimum permissible value is 0).
C_1	VALVE.COEFF	Flow coefficient for valve flow from the from-end to the to-end when back pressure is limiting the flow.
Pv	$PRESSURE_{abs}$	Vapor pressure. {1 psia}
R	n/a	Critical ratio for cavitation. {1}
P_{s0}	PRESSURE	Lowest value of sleeve pressure. {100 psi}
Ps_1	PRESSURE	Highest value of sleeve pressure. {1000 psi}
*	n/a	Placeholder.
ET	n/a	A linearization parameter. {0.2}
A_2	VALVE.COEFF	Flow factor or constant used to calculate the valve coefficient for valve flow from the from-end to the to-end. {A_1}
C_2	VALVE.COEFF	Flow coefficient for valve flow from the from-end to the to-end when back pressure is limiting the flow. {C_1}

Description

The Grove G887 relief valve (V) is used to model a Grove Flex-Flo Model 887 surge relief valve. The Grove valve is modeled like a control valve and it must have a control system. The flow rate through the surge relief valve is initially a function of percent overpressure. As pressure increases, the flow rate may become limited by back pressure or by cavitation. The flow rate for each operating condition is calculated and the minimum value is used to determine the actual pressure - flow conditions in the valve. For this discussion, let:

- Q_1 represent the flow rate based on the general flow formula
- Q_2 represent the flow rate based on either back pressure limited flow or cavitation

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Recommend

$Ps_0 = Ps_1$ where the value is identical to the controller set point, SP.

Parameters

GL recommends that you contact the Grove Valve and Regulator Company for the appropriate input parameters.

This section contains equations for valve flow.

$$Q_1 = \pm\ A_X \left[100 \left(\frac{\max(P_X - P_S, 0)}{P_S}\right)\right]^b \sqrt{\frac{P_S}{G}}$$

where:

±	=	Use "+" if flow is in forward direction, "-" otherwise
P_L	=	Pressure at from-end
P_R	=	Pressure at the to-end
P_X	=	Upstream (actual, not nominal) pressure in absolute units
P_S	=	Pressure set point
A_X	=	Flow constant: A_1 (forward flow) or A_2 (reverse flow)
b	=	Exponent in flow equation
G	=	Specific gravity of the fluid relative to water at 60°F

The term $[100 \max(P_X - P_S,0)/P_S]$ represents the percent over pressure (% pressure rise over set point pressure). The term $A_X [100 \max(P_X - P_S,0)/P_S]^b$ represents the flow coefficient, based on the flow factor A_X.

$$Q_2 = \pm C_X \sqrt{\frac{|\Delta P|}{G}}$$

$$\Delta P = bound\ (0,\ 0.8(P_X - RP_V),\ |P_L - P_R|\ )$$

where:

±	=	Use "+" if flow is in forward direction, "-" otherwise
C_X	=	Flow coefficient: C_1 (forward flow) or C_2 (reverse flow)
bound(a,b,c)	=	max(a, min(b,c))
Pv	=	The bubble point pressure, in absolute units
R	=	The critical ratio for cavitation.

Then:

$$Q = \begin{cases} Q_1 & if\ |Q_1| < |Q_2| \\ Q_2 & otherwise \end{cases}$$

The positive flow (from P_L to P_R) is given as q, which solves the first order equation:

$$q' + r(q - Q) = 0$$

where:

q'	=	The derivative of q with respect to time
r	=	The rate constant for the actuator controlling the G887 valve

The rate constant r sets the rate with which the flow changes toward the value of Q. To make this more specific, if a step-change in Q of one unit is made at time zero (assuming q and Q have been equal for a long period), then the difference between q and Q is exp(-rt). This means that at a time such that rt = 3, then $|q-Q| = 1/e^3$, which is 0.05. So, if the valve can break 95% open in, for example, 100 milliseconds, r may be chosen as described in the next paragraph.

"Actuator (A)" on page 400 states that the rate constant is entered with units of reciprocal minutes. Therefore, an r is desired in reciprocal minutes such that rt = 3, if t is 100 milliseconds, expressed in minutes. Such a t is 0.1/60, so that $r*0.1/60=3$ or $r=1800$ reciprocal minutes.

Note: The global parameter DTMIN should be smaller than the actuation time for this valve to permit opening of the relief valve over multiple time steps.

Valve fraction

The fraction open, FR, of the G887 valve is based on the following:

$$FR = \begin{cases} Q_1 / Q_2 & \text{if } Q_1 < Q_2 \\ 1 & \text{otherwise} \end{cases}$$

Linearization parameter

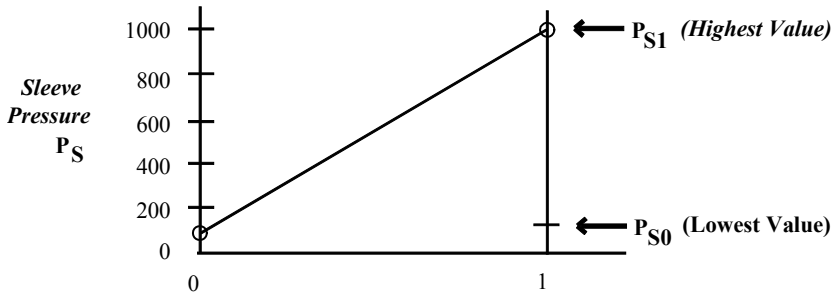
The ET is a linearization parameter that represents the interval on the argument axis zero psi in which the square root function is linearized as a chord instead of the inverse parabola. The default value should suffice in all cases that should arise in typical pipeline situations. If other than the default values for A_2 and C_2 are to be used, enter ET as 0.2.

Sleeve pressure

$$P_S = P_{S0} + (P_{S1} - P_{S0}) \bullet X$$

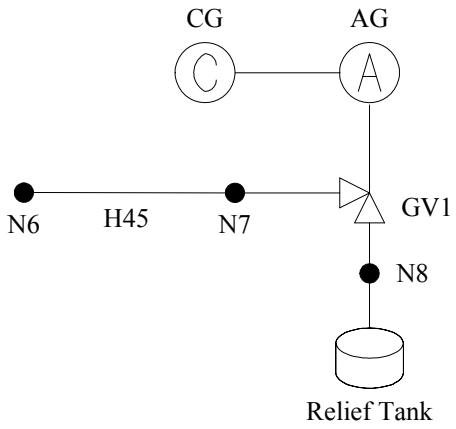
where:

X	=	The output of the actuator
P_{S0}	=	Lowest value of sleeve pressure, entered relative to local atmospheric pressure. Default is 100 psi.
P_{S1}	=	Highest value of sleeve pressure, entered relative to local atmospheric pressure. Default is 1000 psi.



Example input

Because an actuator controls this valve, the example needs to have a control system defined. The control system consists of an actuator, a controller, and a sensor. (See "Control elements" on page 144 for information on these control elements).



An 8" Grove G887 relief valve named GV1 (ANSI Class 300) is connected to nodes named N7 and N8 and is installed between a header named H45 and a relief tank named TANK whose pressure is held at 10 psig. The specific gravity of the fluid is 0.83. The surge relief valve has the following specifications:

A_1	=	8.57 gpm
b	=	1
C_1	=	850 gpm
G	=	0.83
Pv	=	1 psia
R	=	1
ET	=	.2

The initial sleeve pressure is 500 psig and is specified as the set point of controller CG. The rise time for flow is to be 0.033 seconds (33 milliseconds). The following INPREP input defines the valve system:

The following INPREP input defines the foregoing valve system:

```
V GV1 N7 N8 G887 8.57 1 850
+ 1 1 500 500 * 0.2
I DUMMY F(T)2 0 0 0 1
C CG DUMMY:OUT 500 1 0  0  0  1E12
A AG C CG V GV1 X(V)2 0 1 V1800
```

General regulator (RG)

Peek and Poke Attributes

```
RG FID FROM TO [CRV-CX] CGMIN CGMAX
[+ TRAVEL.TIME TT]
[+ FLOW.SETPOINT SQ]
[+ UPSTREAM.PRESSURE.SETPOINT SPU]
[+ DOWNSTREAM.PRESSURE.SETPOINT SPD]
```

Field	Units Key	Description
FID	n/a	Unique name of the regulator.
FROM	n/a	From-node name or upstream connection.
TO	n/a	To-node name or downstream connection point.
CRV-CX	n/a	Name of a data curve defining C_V vs. stem position. Curve type is C(X). For more information, see "Data curve (D)" on page 418. If linear, enter C(X)2. For equal percentage valves, enter EQPC.
CGMIN	VALVE.CG	Closed valve coefficient (minimum). {0}
CGMAX	VALVE.CG	Open valve coefficient (maximum).
TT	TIME	Full travel time for the regulator valve.
SPU	PRESSURE	Upstream pressure set point.
SPD	PRESSURE	Downstream pressure set point.
SQ	FLOW	Flow set point.

Description

The RG is an idealized regulator, which can be a downstream pressure regulator, an upstream pressure regulator, or a flow regulator. The equation used is the same as the SynerGEE BR or DR. The RG is similar to the RE in SPS, but it uses a gas valve coefficient.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Valve opening

SynerGEE uses a linear valve curve; SPS, by default, uses an equal percentage valve curve but can use any valid user-defined valve curve.

Open and close time

The travel time, the time to open the valve, and the time to close the valve must be equal.

Idealized regulator - control valve (RE)

Peek and Poke Attributes

```
RE NAME FROM TO [CRV-CX] [Cvc] Cvo
[+ TRAVEL.TIME TT]
[+ FLOW.SETPOINT SQ]
[+ UPSTREAM.PRESSURE.SETPOINT SP-]
[+ DOWNSTREAM.PRESSURE.SETPOINT SP+]
[+ FRACTION.OPEN.SETPOINT SFR]
[+ CHECK]
```

Field	Units Key	Description
NAME	n/a	Unique name of the regulator.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
CRV-CX	n/a	Name of a data curve defining C_v vs. stem position. Curve type is C(X). For more information, see "Data curve (D)" on page 418. If linear, enter C(X)2. For equal percentage valves, enter EQPC.
C_{vc}	VALVE.COEFF	Valve coefficient at full-close position. {0}
C_{vo}	VALVE.COEFF	Valve coefficient at full-open position.
TT	TIME	Full travel time for the regulator valve. {.5 minutes}
SQ	FLOW	Flow set point. {+ ∞ }
SP-	PRESSURE	Upstream pressure set point. {- ∞ }
SP+	PRESSURE	Downstream pressure set point. {+ ∞ }
SFR	n/a	Fraction open set point. {1}

Description

The idealized regulator (RE) models a regulator (a control valve with its associated control system) with up to four set points: flow, upstream pressure, downstream pressure, and fraction open. In addition the idealized regulator constrains the valve's fraction open rate of change to be consistent with specified full travel time TT and if the + CHECK option is selected the regulator only allows flow in the positive direction.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Idealized control

The RE element is idealized in the sense that it is modeled as though it had an ideal control system. If the regulator valve can be throttled to avoid violating a set point, then that set point value is maintained. In practice however, with a realistic value for TT, the full travel time constraint often dominates the regulator's control during hydraulic transient.

The set points applied to the regulator control the regulator valve's fraction open such that:

- Flow through the regulator is maintained at or below the flow set point
- Discharge pressure is maintained at or below the discharge pressure set point
- Upstream pressure is maintained at or above the upstream pressure set point
- Fraction open is maintained at or below the fraction open set point

Each of these limits is evaluated with the one resulting in the smallest fraction open governing. In effect, the idealized regulator control system is equipped with a low select relay.

The regulator set points are not violated unless one of the following occurs:

- The fraction open is restricted by the maximum rate of change defined by the full travel time TT
- The regulator is fully open
- The regulator fully closed
- The check valve, present by selecting the + CHECK option, is closed

Equal-percentage valve

This valve approximates the equal percentage behavior that actual valves display. The equal percentage behavior is valid for stem positions greater than 0.2, and approximates C as the stem moves from 0.2 to 0 in a smooth equal percentage manner. C_{vc} and C_{vo} for the regulator must not be less than zero. If a value of zero is entered for either C_{vc} or C_{vo} the zero value is interpreted as 0.0001 MB/D-PSI$^{0.5}$.

$C_v(X)$ is given by:

$$C_v(X) = A \bullet \exp(kX) + B$$

where:

$$A = \frac{C_{vo} - C_{vc}}{\exp^{k} - 1}$$

$$B = C_{vc} - A$$

$$k = 3.488$$

Check valve

If the optional +CHECK is specified, the regulator is modeled with an idealized series check valve. This idealized check valve's operation is simulated such that:

- Flow is only allowed in the positive direction (from-end to to-end)
- The valve closes instantaneously to prevent negative flow
- The valve opens instantaneously upon a positive pressure differential (P- > P+)

- When open, it exhibits no pressure drop.

Note that the idealized series check valve is modeled as a valve separate from the regulator valve. As such, the regulator valve's fraction open varies consistent with the regulator's idealized control system even while the check valve's status is CLOSED.

Flow-transmissibility characteristics

Transmissibility for flow across this element is expressed using the following formulas.

Liquid

$$Q = C_v \left(\frac{\rho}{\rho_b} \right) \sqrt{\Delta P \frac{\rho_w}{\rho}}$$

where:

Q	=	Flow rate at base (custody transfer) density, ρ_b
ρ	=	Density at flowing conditions
ΔP	=	Pressure drop
ρ_w	=	Density of water
ρ_b	=	Density at custody conditions

Note: The liquid equation is accurate only in the range $0.5 < \Delta P < 10000$ psi

The flow is equivalent to flow in consistent units at flowing conditions given by:

$$q = C_v \sqrt{\frac{\Delta P}{SG}}$$

where $SG = \rho/\rho_w$

Liquid critical flow

Critical flow occurs when:

$$\Delta P \geq C_f^2 \, (P_u - P_o)$$

where:

C_f	=	The critical flow factor, 1.0
P_u	=	Absolute upstream pressure
P_o	=	Absolute bubble point pressure

When critical flow occurs, decreasing the downstream pressure does not increase the flow rate through the element at a given fraction open. The corresponding critical flow equation is:

$$Q = C_v \left(\frac{\rho}{\rho_b} \right) \sqrt{(P_u - P_{ct}) \frac{\rho_w}{\rho}}$$

where:

P_u	=	Absolute upstream pressure
P_{ct}	=	Absolute cutoff pressure (when critical flow occurs)

$$P_{ct} = \left(P_u \ (1 - C_f^2) + \ P_o \ C_f^2\right)$$

Gas

$$Q = C_v \ N1 \sqrt{\frac{(P_u^2 - P_d^2)}{(Z \ G \ T)}}$$

where:

Q	=	Flow, standard
Z	=	Compressibility factor at upstream temperature and average pressure
G	=	Specific gravity of gas relative to air (MWgas/MWair)
T	=	Absolute upstream temperature
P_u	=	Absolute upstream pressure
P_d	=	Absolute downstream pressure
N_1	=	Units conversion

$$120\sqrt{°R/PSI}$$

The default flow units are those of VALVE.COEFF.

Gas critical flow

Critical flow (sonic flow) occurs when

$$\Delta P \geq .5 \ C_f^2 \ P_u$$

where:

C_f	=	Critical flow factor, 1.0
P_u	=	Absolute upstream pressure

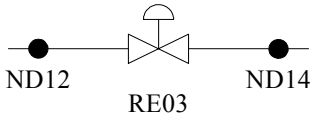
When critical flow occurs, decreasing the downstream pressure does not increase the flow rate through the element at a given fraction open. The corresponding critical flow equation is:

$$Q = C_v \ N_1 \ P_u \ C_f \sqrt{\frac{(1-.25C_f^2)}{ZGT}}$$

Actuator control

Alternately to the idealized control, you can put a full control system on the RE regulator.

Example input



An idealized control regulator, named RE03, is connected to nodes ND12 and ND14. Specify valve RE03 as an equal percentage valve with a closed valve coefficient of 0, an open valve coefficient of 80, a downstream set pressure of 500 and a travel time of .5 minutes.

```
RE RE03 ND12 ND14 0 80
+ TRAVEL.TIME .5, DOWN.PRES.SET 500
```

Now, instead of specifying an equal percentage valve, use a valve curve named V250VB.

```
D V250VB C(X)
 0 .01   .025  .055  .100  .165  .250  .370  .568   1
 0 .1111 .2222 .3333 .4444 .5555 .6666 .7777 .8888  1

RE RE03 ND12 ND14 V250VB 0 2000
+ TRAVEL.TIME .5
+ DOWN.PRES.SET 500
```

Compressor TABLE

The various compressor types use either the table syntax or the CSV syntax to input certain variables. This section explains the general rules for creating tables of compressor data. For general rules on SPS's input formats, see "Input syntax" on page 47 and "CSV input syntax" on page 52.

The following input format uses SPS's CSV input format for one dependent variable.

```
+ TABLE
{
KEY_f, KEY_x, KEY_y, [KEY_z]
f_1, x_1, y_1, [z_1]
f_2, x_2, y_2, [z_2]
...
f_n, x_n, y_n, [z_n]
}
```

The following input format uses SPS's CSV input format for two dependent variables.

```
+ TABLE
{
KEY_f, KEY_g, KEY_x, KEY_y
f_1, g_1, x_1, y_1
f_2, g_2, x_2, y_2
...
f_n, g_n, x_n, y_n
}
```

Alternately, the following input format uses the table syntax for one dependent variable.

```
+ TABLE  KEY_f  KEY_x  KEY_y  [KEY_z]
[        f_1  x_1  y_1  [z_1]]
[        f_2  x_2  y_2  [z_2]]
...
[        f_n  x_n  y_n  [z_n]]
```

Alternately, the following input format uses the table syntax for two dependent variables.

```
+ TABLE  KEY_f  KEY_g  KEY_x  KEY_y
[        f_1  g_1  x_1  [y_1]]
[        f_2  g_2  x_2  [y_2]]
...
```

```
[        f_n  g_n  x_n  [y_n]]
```

Field	Description
KEY_f, KEYg, KEYx, KEYy, and KEYz	Keyword parameters.
f	A value must be entered on each line of the table.
g	A value must be entered on each line of the table. Used only when table has two dependent variables, such as for compressor head and efficiency table, where both head and efficiency are dependent on flow and speed: head(flow,speed), efficiency(flow, speed)
x	A unique value must be entered on each line of the table. For each unique y, x values must be strictly increasing.
y	A value must be entered on the first line. Additional values may be entered on subsequent lines. If any y value is omitted on subsequent lines, the missing value is copied from the line above. For each unique z, y values must be increasing (but not strictly increasing).
z	A value must be entered on the first line. Additional values may be entered on subsequent lines. If any z value is omitted on subsequent lines, the missing value is copied from the line above. z values must be increasing (but not strictly increasing).

Description

This command declares a relationship f(x, y), f(x,y,z), or g(x, y) as it relates to various compressor tables. SPS can calculate the dependent variable by interpolating whenever necessary.

Each (x,y) pair must be unique. For each unique z, the y's must be in increasing (but not strictly increasing) order. For each unique y, the x's must be in strictly increasing order.

The case of f(x, y, z) is similar in that each (x, y, z) set must be unique. Furthermore, z's must be increasing (but not strictly increasing); for each unique z, y's must be increasing (but not strictly increasing); and for each unique y, x's must be strictly increasing.

Example input

See the relevant example for the compressor you are entering.

Compressor fuel

Note: The various compressor types calculate and/or extract fuel in a consistent way. This section documents the fuel use capability of all of the various compressor types.

```
... [FCNC]
[+ NO.FUEL]
[+ HEAT.RATE HR]
[+ TABLE HEAT.RATE POWER [SPEED]]
```

Field	Units Key	Description
...	n/a	Compressor key letters followed by other fields. See specific compressor documentation for details. "Compressors" on page 144 provides a list of compressors types.
FCNC	n/a	Name of node from which compressor draws fuel.
NO.FUEL	n/a	If this line is entered, the fuel is calculated but it is not taken from the model.
HR	HEAT.RATE	The heat rate per power output required to fuel the compressor, for a single compressor. {0}
HEAT.RATE column	HEAT.RATE	The heat rate per power output required to fuel the compressor, for a single compressor. See "Compressor TABLE" on page 335 for rules.
POWER column	POWER	The power level at which the heat rate applies. For the CC, KC, and KV compressors, the power column is ISO power. The ISO power is the compressor load power corrected to sea level and the reference temperature TR (from the REFERENCE.TEMPERATURE entry for the compressor), exclusive of inlet and exhaust duct losses. For other compressor types the power column is site power. See "Compressor TABLE" on page 335 for rules.
SPEED column	SPEED	The rotary speed at which the functional relationship between heat rate and power applies. The speed column is not supported by the GC, KP, or KV compressors. The speed column is optional for the other compressor types. See "Compressor TABLE" on page 335 for rules.

Description

The compressor fuel is taken from the model by default. To suppress taking fuel from the model, enter the NO.FUEL keyword.

The compressor fuel use may be entered as a constant heat rate per power or as a table of heat rate as a function of power and optionally speed, but not both.

Fuel use equation

The heat rate, HR, is taken from the HEAT.RATE constant entry or is interpolated from the HEAT.RATE table entry, whichever is applicable:

$$HR = \begin{cases} hr \\ HEAT.RATE\left(\frac{PWR}{NRN}\right) \\ HEAT.RATE\left(\frac{PWR}{NRN}, RPM\right) \end{cases}$$

If the compressor current operating point lies outside of the heat rate table data, then the heat rate table value closest to the operating point is used.

The thermal fuel flow HF and the fuel gas rate QF are:

$$HF = BPWR \bullet HR$$

$$QF = \frac{HF}{LHV}$$

where BPWR is the effective brake power computed from the compressor power and the entered duct losses DLOSS, if any:

$$BPWR = PWR / [1 - DLOSS]$$

The DLOSS entry is supported by only the CC and KC compressors. For other compressor types, use DLOSS=0 in this equation.

Centrifugal compressor (CC)

Peek and Poke Attributes

INPREP

```
CC FID FROM TO [FCNC]
± TABLE    HEAD  EFFICIENCY  FLOW   SPEED

/*ALL INPUT BELOW IS OPTIONAL

[+ DRIVER  GAS | ELECTRIC]
[+ CONTROL.SPEED  SS]
[+ MINIMUM.SPEED  SMIN]
[+ MAXIMUM.SPEED  SMAX]
[+ MAXIMUM.POWER  MXPW]
[+ START.TIME  START]
[+ STOP.TIME  STOP]
[+ DISCHARGE.PRESSURE.SETPOINT  PDMAX]
[+ SUCTION.PRESSURE.SETPOINT  PSMIN]
[+ FLOW.SETPOINT  SQ]
[+ NPOLY  NPOLY]
[+ TRR  TRR]
[+ SURGE.CONTROL.METHOD  RECYCLE | SPEED | TRIP ]
[+ TRIPS  YES | NO ]
[+ NO.FUEL]
[+ HEAT.RATE HR]
[+ TABLE HEAT.RATE POWER [SPEED]]
```

Mandatory entries

Field	Units Key	Description
FID	n/a	Name of the centrifugal compressor.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.

Table entries

Field	Units Key	Description
HEAD	HEAD	Table values of polytropic compressor head. See “Compressor TABLE” on page 335 for rules.
EFFICIENCY	EFFICIENCY	Table values of polytropic efficiency. See “Compressor TABLE” on page 335 for rules.

Field	Units Key	Description
FLOW	COMP.AFLOW	The flow at suction conditions to the compressor. See "Compressor TABLE" on page 335 for rules.
SPEED	SPEED	The speed for which the head and efficiency as a function of flow relationships apply. See "Compressor TABLE" on page 335 for rules.

Optional entries

Field	Units Key	Description
GAS	n/a	Indicates that the compressor is using a gas driver. This is the default value. Compressors with gas drivers have their power automatically derated for elevation, ambient temperature, and duct losses.
ELECTRIC	n/a	Indicates that the compressor is using an electric driver. Compressors with electric drivers do not have their power derated. When set to ELECTRIC, the following are not allowed by PREPR (and cannot be entered in Model Builder): • NO.FUEL (implied by ELECTRIC) • HEAT.RATE • TABLE HEAT.RATE
FCNC	n/a	Name of node from which compressor draws fuel. See "Compressor fuel" on page 337.
SS	SPEED	Controlled compressor speed. {Maximum speed in head-efficiency table}
SMIN	SPEED	Compressor minimum speed. {Minimum speed in head-efficiency table}
SMAX	SPEED	Compressor maximum speed. {Maximum speed in head-efficiency table}
MXPW	POWER	Maximum permissible turbine power under any conditions. {}
START	TIME	Spin-up time for starting compressor. {3 minutes}
STOP	TIME	Spin-down time for stopping compressor. {3 minutes}
PDMAX	PRESSURE	Discharge pressure set point. {0 = inactive}
PDMIN	PRESSURE	Suction pressure set point. {0 = inactive}
SQ	FLOW	Flow set point. {0 = inactive}
NPOLY	n/a	Polytropic exponent. {If you use an equation of state that permits evaluation of heating value (BWRS or AGA), the value is calculated as compositions change upstream of the compressor; otherwise, the default is 1.3. Note that a user-entered value always overrides a calculated value. If you want to get the default behavior, either do not enter NPOLY or set NPOLY to 0 in TRANS.}
TRR	n/a	Temperature rise ratio used to determine discharge temperature. Defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.

Field	Units Key	Description
RECYCLE	n/a	If the compressor surges, flow is recycled to correct the surge condition. This is the default.
SPEED	n/a	If the compressor surges, the speed is controlled so that no recycle flow occurs. This method may violate set points. If a physical limitation such as maximum speed or maximum horsepower is violated the unit trips off.
TRIP	n/a	If a surge condition exists, the compressor trips (shuts down).
YES	n/a	Violations of physical limitations cause the compressor to trip (shut down). This is the default.
NO	n/a	Violations of physical limitations are ignored.

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Description

The centrifugal compressor (CC) models a variable-speed, gas-turbine-driven centrifugal compressor operating in parallel and simulated as a single unit.

Under any speed condition, the unit is operated to avoid both the *surge* and *stonewall* flow rates. The surge line is defined by the minimum flow rates entered for the set of speed values in the table of head and efficiency. The stonewall line is defined by the maximum flow rates entered for the set of speed values in the table of head and efficiency.

In addition, the compressor adjusts to the maximum power constraints imposed by one of two relationships, whichever is the more restrictive: the data entered via the table with the maximum power, or the data entered via the exhaust temperature in conjunction with the maximum exhaust temperature. The current value for the maximum power constraint derived from either of the data entered may be viewed as a peek.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Automatic Power Deration

Compressors with gas drivers (the default) have their power automatically derated for elevation, ambient temperature, and duct losses. Compressors with electric drivers do not have their power derated.

SynerGEE considerations

- The CC compressor supports all of the input options that the KC compressor supports; only the options that are exported by SynerGEE are documented here.
- SynerGEE has fuel rate in units of standard volume over power, e.g., SCFD/HP. SPS has fuel rate in units of energy over energy, e.g., BTU/HP-HR.
- The SPS compressor names are generated from the SynerGEE station name, path number, and unit number, e.g. STA1_1_1 for STA1 unit 1 path 1.

- The SPS from-node and to-node are taken from the corresponding CS element in SynerGEE.
- The CC can be used only inside a station in SynerGEE. The CC compressor is converted to a stand-alone compressor for SPS.
- In SynerGEE, the pressure, flow, and ratio set points and limits are entered at the CS level not at the CC element. In SPS, these values are specified on the CC element. The SynerGEE export process copies the CS values to the CC in SPS.

General compressor (GC)

Peek and Poke Attributes

INPREP

```
GC FID from to [FCNC]
±RATED.POWER RP
+ PARAMETERS K1 K2 K3
[+ MAXIMUM.POWER PWMAX]
[+ POWER.SETPOINT SPPW]
[+ DISCHARGE.PRESSURE.SETPOINT PDMAX]
[+ SUCTION.PRESSURE.SETPOINT PDMIN]
[+ MAXIMUM.FLOW QMAX]
[+ MINIMUM.FLOW QMIN]
[+ BYPASSCHECK]
[+ START.TIME START]
[+ STOP.TIME STOP]
[+ NPOLY NPOLY]
[+ TRR TRR]
[+ NO.FUEL]
[+ HEAT.RATE HR]
[+ TABLE HEAT.RATE POWER]
```

Field	Units Key	Description
FID	n/a	Name of the compressor.
from	n/a	Name of the from connection point or named node.
FCNC	n/a	Name of node from which compressor draws fuel. See “Compressor fuel” on page 337.
to	n/a	Name of the to connection point or named node.
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.
RP	POWER	Rated power for spin-up and-spin down sequencing. If the rated power is not known, a reasonable approximation should be entered and a large number can be entered for the maximum power.
K1	POWER/FLOW	Compressor equation coefficient. {193 HP/(MMSCFD)}
K2	POWER/FLOW	Compressor equation coefficient. {192 HP/(MMSCFD)}
K3	dimensionless	Compressor equation exponent. {.231}
PWMAX	POWER	Maximum allowable compressor power constraint. Defaults to the RATED.POWER value.
SPPW	POWER	Compressor power set point. {2 * MXPW}
PDMAX	PRESSURE	Discharge pressure set point. {10,000 psig}
PDMIN	PRESSURE	Suction pressure set point. {0 psig}
QMAX	FLOW	Maximum flow constraint. {10,000 MMCF/D}
QMIN	FLOW	Minimum flow constraint. {0 MMCF/D}

Field	Units Key	Description
BYPASSCHECK	n/a	Check valve on parallel bypass line.
START	TIME	Spin-up time of the compressor. {3 minutes}
STOP	TIME	Spin-down time of the compressor. {3 minutes}
NPOLY	Dimensionless	Polytropic exponent. {1.3}
TRR	Dimensionless	Temperature rise ratio used to determine discharge temperature. Defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Description

The general compressor (GC) is an idealized compressor used to theoretically model any type of compressor. The GC is an idealized control element, which means that no control system is necessary for the compressor to hold set points. The compressor has logic which acts like a control system in maintaining set points and constraints. When a compressor cannot operate with all its constraints satisfied, it is automatically shut down.

This compressor may be exported from SynerGEE.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

SynerGEE considerations

- The GC compressor supports all of the input options that the KP compressor supports; only the options that are exported by SynerGEE are documented here.
- SynerGEE has fuel rate in units of standard volume over power, e.g., SCFD/HP. SPS has fuel rate in units of energy over energy, e.g., BTU/HP-HR.
- In SynerGEE, a bypassed compressor is saved as a stopped compressor.
- In the CONT options (POWER, DCONT, SCONT, and BYPASS) specified in SynerGEE convert to the SPS equivalents of POWER.SETPOINT, DISCHARGE.PRESSURE.SETPOINT, SUCTION.PRESSURE.SETPOINT, and BYPASSCHECK.

Example input

A compressor named COMP1 is connected to nodes STA1S and STA1D. The following parameters are used: rated power is 3100, K1 = 193, K2 = 192, and K3 = .231 and discharge pressure set point 985. The maximum power is 10,000 and the heat rate is 9167.

```
GC COMP1 STA1S STA1D
+ RATED.POWER 3100
```

```
+ PARAMETERS 193 192 .231
+ DISCH.PRESS.SETP 985
+ MAX.POWER 10000
+ HEAT.RATE 9167
```

Idealized controllable centrifugal compressor (KC)

Peek and Poke Attributes

INPREP

```
KC NAME FROM TO [FCNC]
±TABLE    HEAD  EFFICIENCY  FLOW   SPEED
[+ DRIVER  GAS | ELECTRIC]
[+ HYDRAULIC.EFFICIENCY.FACTOR  HEC]
[+ CONTROL.SPEED  SS]
[+ MAXIMUM.TURBINE.EXHAUST  MXTE]
[+ MINIMUM.SPEED  MNS]
[+ MAXIMUM.SPEED  MXS]
[+ NUMBER.RUNNING  NRN]
[+ MECHANICAL.EFFICIENCY  ME]
[+ MAXIMUM.POWER  MXPW]
[+ START.TIME  START]
[+ STOP.TIME  STOP]
[+ AMBIENT.TEMPERATURE  TA]
[+ MAXIMUM.DISCHARGE.TEMP  MXT+]
[+ DISCHARGE.PRESSURE.SETPOINT  SP+]
[+ SUCTION.PRESSURE.SETPOINT  SP-]
[+ FLOW.SETPOINT  SQ+]
[+ DUCT.POWER.LOSS  DLOSS]
[+ REFERENCE.TEMPERATURE  TR]
[+ NPOLY  NPOLY]
[+ TRR  TRR]
[+ ELEVATION  ELEV]
[+ SURGE.CONTROL.METHOD  RECYCLE | SPEED | TRIP ]
[+ TRIPS  YES | NO ]
[+ TABLE   MAXIMUM.POWER  AMBIENT.TEMP  SPEED]
[+ TABLE   EXHAUST.TEMP  POWER  AMBIENT.TEMP  SPEED]
[+  HEAT.RATE  HR]
[+  NO.FUEL]
[+ TABLE   HEAT.RATE  POWER  [SPEED]]
```

Mandatory entries

Field	Units Key	Description
NAME	n/a	Name of the centrifugal compressor.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.

Table entries

Field	Units Key	Description
HEAD	HEAD	Polytropic compressor head. See "Compressor TABLE" on page 335 for rules.
EFFICIENCY	EFFICIENCY	Polytropic efficiency (decimal values) See "Compressor TABLE" on page 335 for rules.
FLOW	COMP.AFLOW	The flow at suction conditions to the compressor. See "Compressor TABLE" on page 335 for rules.
SPEED	SPEED	The speed for which the head and efficiency as a function of flow relationships apply. See "Compressor TABLE" on page 335 for rules.

Optional entries

Field	Units Key	Description
GAS	n/a	Indicates that the compressor is using a gas driver. This is the default value. Compressors with gas drivers have their power automatically derated for elevation, ambient temperature, and duct losses.
ELECTRIC	n/a	Indicates that the compressor is using an electric driver. Compressors with electric drivers do not have their power derated. When set to ELECTRIC, the following are not allowed by PREPR (and cannot be entered in Model Builder): • + MAXIMUM.TURBINE.EXHAUST • + AMBIENT.TEMPERATURE • + DUCT.POWER.LOSS • + REFERENCE.TEMPERATURE • + ELEVATION • + TABLE MAXIMUM.POWER • + TABLE EXHAUST.TEMPERATURE • + HEAT.RATE • + NO.FUEL (implied by ELECTRIC) • + TABLE HEAT.RATE
FCNC	n/a	Name of node from which compressor draws fuel. See "Compressor fuel" on page 337.
HEC	EFFICIENCY	Multiplier of table hydraulic efficiency values. {1}
SS	SPEED	Controlled compressor speed. {Maximum speed in head-efficiency table}
MXTE	TEMPERATURE	Maximum Power Turbine Exhaust Temperature. { $^{\infty}$ }
MNS	SPEED	Compressor minimum speed. {Minimum speed in head-efficiency table}

Field	Units Key	Description
MXS	SPEED	Compressor maximum speed. {Maximum speed in head-efficiency table}
NRN	dimensionless	Number of identical running compressors. {1}
ME	EFFICIENCY	The mechanical efficiency. {0.97}
MXPW	POWER	Maximum permissible turbine power under any conditions. { ∞ }
START	TIME	Spin-up time for starting compressor. {3 minutes}
STOP	TIME	Spin-down time for stopping compressor. {3 minutes}
TA	TEMPERATURE	Ambient temperature. {70°F}
MXT+	TEMPERATURE	High discharge temperature trip. { ∞ }
SP+	PRESSURE	Discharge pressure set point. {0 = inactive}
SP-	PRESSURE	Suction pressure set point. {0. = inactive}
SQ+	FLOW	Flow Set point. {0 = inactive}
DLOSS	EFFICIENCY	The fraction of power at the generator terminal lost to inlet and outlet ducting. {Default = .025}
TR	TEMPERATURE	Reference temperature from manufacturer for converting hydraulic power to brake horsepower {59°F}
NPOLY	dimensionless	Polytropic exponent. {If you use an equation of state that permits evaluation of heating value (BWRS or AGA), the value is calculated as compositions change upstream of the compressor; otherwise, the default is 1.3. Note that a user-entered value always overrides a calculated value. If you want to get the default behavior, either do not enter NPOLY or set NPOLY to 0 in TRANS.}
TRR	n/a	Temperature rise ratio used to determine discharge temperature. Defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.
ELEV	ELEVATION	Elevation of the compressor. {Default is node elevation.}
RECYCLE	n/a	If the compressor surges, flow is recycled to correct the surge condition. This is the default.
SPEED	n/a	If the compressor surges, the speed is controlled so that no recycle flow occurs. This method may violate set points. If a physical limitation such as maximum speed or maximum horsepower is violated the unit trips off.
TRIP	n/a	If a surge condition exists, the compressor trips (shuts down).
YES	n/a	Violations of physical limitations cause the compressor to trip (shut down). This is the default.
NO	n/a	Violations of physical limitations are ignored.

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Turbine power as a function of ambient temperature

Field	Units Key	Description
MAX.POWER	POWER	The maximum turbine ISO power that can be developed. See "Compressor TABLE" on page 335 for rules.
AMB.TEMP	TEMPERATURE	The ambient temperature for turbine operation. See "Compressor TABLE" on page 335 for rules.
SPEED	SPEED	The speed at which the functional relationship between MAX.POWER and AMB.TEMP applies. See "Compressor TABLE" on page 335 for rules.

Relationship of exhaust temperature and power for the turbine

Field	Units Key	Description
EXH.TEMP	TEMPERATURE	The power turbine exhaust temperature. See "Compressor TABLE" on page 335 for rules. **Note:** If this table is entered, the relationships defined can be used along with the maximum exhaust temperature value to limit the available turbine power.
POWER	POWER	The ISO power level corresponding to the preceding exhaust temperature. See "Compressor TABLE" on page 335 for rules.
AMB.TEMP	TEMPERATURE	The ambient temperature. See "Compressor TABLE" on page 335 for rules.
SPEED	SPEED	The speed value for which the foregoing relationships are defined. See "Compressor TABLE" on page 335 for rules.

Description

The idealized controllable centrifugal compressor (KC) models single or multiple (NRN > 1) variable-speed, gas-turbine-driven centrifugal compressors operating in parallel and simulated as a single unit.

Under any speed condition, the unit is operated to avoid both the *surge* and *stonewall* flow rates. The surge line is defined by the minimum flow rates, in COMP.AFLOW units, entered for the set of speed values in the TABLE of head and efficiency. The stonewall line is defined by the maximum flow rates, again in COMP.AFLOW units, entered for the set of speed values in the TABLE of head and efficiency.

In addition, the compressor adjusts to the maximum power constraints imposed by one of two relationships, whichever is the more restrictive: the data entered via the TABLE with the MAXIMUM.POWER keyword, or the data entered via the EXHAUST.TEMP in conjunction with the maximum exhaust temperature entered using the MXTE keyword. The current value for the maximum power constraint derived from either of the data entered may be viewed as the peek :MXP.

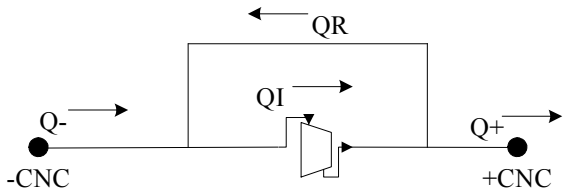
Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Flows

For a discussion on flows, consider the following schematic:



There are four definitions for flow (at standard conditions) for the KC compressor.

- Q- is used to define the flow into the compressor from the connecting inlet device.
- QI is used to define the total flow through the compressor.
- QR is used to define recycle flow.
- Q+ is the flow out of the compressor to the connecting outlet device.

The relationships of Q-, Q+, QI, and QR are defined by:

$$QI = Q- + QR$$
$$Q+ = QI - QR$$

The foregoing values of Q are all referenced to standard (custody transfer) conditions.

An additional flow variable, FI, is also defined. FI is QI converted to volumetric flow rate at inlet conditions. As such, FI is the flow on which surge and stonewall control is based. FI is in COMP.AFLOW units at inlet conditions.

Speed control

You can choose one of two modes for compressor speed control:

- Speed controlled by ideal set point (:SQ+) (:SP+) (:SS) or (:SP-)
- Speed controlled by actuator

Mode 1: Ideal set points

You may select the speed of the unit to be controlled to desired flow, speed, discharge pressure, or suction pressure set points. This mode is selected by entry of one or more positive values via keywords SQ+, SP+, SP-, or :SS in the INPREP file or by interactively poking :SQ+, :SP+, :SP-, or :SS to a positive value during the simulation. The unit controls the running speed to attempt to maintain whichever set point is limiting. In any case, the running speed is limited to the :MXS and :MNS. The unit also controls to limit the power developed to the maximum power available. This is accomplished by adjusting the running speed to the value needed to maintain the limiting set point. The unit exercises automatic internal controls as necessary to avoid the surge and stonewall lines.

Mode 1 is not available if an actuator is attached to the compressor.

The unit shuts down if the speed falls below minimum speed (MNS) while attempting to maintain a set point or meeting a constraint. In thermal simulations, the unit also shuts down if the discharge temperature exceeds the high temperature trip point, MXT+. The unit may be restarted after automatic shutdown by issuing a START command. Unwanted shutdowns of the latter sort can be avoided by using the default value for MXT+, or poking :MXT+ to a quite large value.

Mode 2: Actuator controlled

You may elect to control the compressor speed by attaching an actuator (and an appropriate user-defined control system) that ramps compressor speed in response to actuator input signals. The compressor speed is given by:

$$S = MXS \bullet X$$

where X is the output of the controlling actuator. The actuator's input may be controlled by signals from controllers or relays in response to sensed values. If this mode of operation is selected, Mode 1 is unavailable and the values entered for :SS, :SQ+, :SP+ or :SP- are ignored. The unit, however, controls to avoid the surge and stonewall lines.

The unit is also controlled to limit the power to :MXP. If the power at the actuator-controlled speed exceeds :MXP, the unit maintains the power at :MXP by reducing the speed. The control system's controller set point is ignored until the power requirement of the actuator-controlled speed falls below :MXP (the speed at :MXP falls below MXS * X), at which time, speed control is returned to the actuator.

The unit shuts down if the actuated speed falls below :MNS.

In thermal simulations, the unit also shuts down if the discharge temperature exceeds the user-defined :MXT+ value. The unit may be restarted after automatic shutdown by issuing a START command.

Surge and stonewall control

In all speed control modes, the KC unit functions to avoid both the surge and stonewall lines. As an aid to interactive control a variable FI/S is peekable using the attribute :FIS. In this variable S is the speed of the compressor. This peek is helpful because for most surge and stonewall lines, FI is approximately linear with S.

If the calculated value of :FIS falls below the surge line, the unit implicitly recycles sufficient discharge gas to maintain :FIS at the surge value. The rate of gas recycle at custody transfer conditions is peekable using the attribute :QR. The unit shuts down if the discharge temperature exceeds :MXT+.

If the calculated value of :FIS rises above the stonewall line, the unit limits :FIS so as not to exceed the stonewall value. This is accomplished by increasing the effective internal resistance to flow of the unit until the flow satisfies the stonewall constraint.

START/STOP control

A compressor that is STOPPED may be started by issuing a START command in the INTRAN file, or by issuing a START command interactively. The spin-up time on starting is given as START.TIME in TIME units. For actuator-controlled compressors, the controlling actuator must always have a positive output.

Similarly, a compressor that is RUNNING can be stopped by issuing a STOP command in the INTRAN file, or by issuing a STOP command interactively. The spin-down time on stopping is given as STOP.TIME in TIME units.

Switching from starting to running

All of the following criteria must be met for a KC compressor to switch from starting to running:

- Built-in check valve is open
- The compressor is not in surge
- Speed has stopped ramping up at the maximum rate, and
- Speed > min(MinSpeed, 0.90*MaxSpeed)

Equations

Head calculation equation

The compressor's developed head (in HEAD units) is evaluated from the TABLE of head and efficiency entries. If a HEAD as a function of actual FLOW relation is entered for only one speed, the head is computed by the fan laws as follows:

$$H = H_1(FI / S) (S / S_1)^2$$

where:

H	=	Head of the machine at current conditions
FI	=	Current inlet flow rate at actual conditions
H_1	=	Function representing the HEAD data expressed in terms of the argument FI_1/N_1
FI_1	=	Set of inlet-condition-flow values (FLOW) associated with HEAD values of the head-efficiency TABLE
S_1	=	Single speed (SPEED) input in the HEAD TABLE
S	=	Current speed of the compressor

In this case the surge line is taken to be a linear relation between flow at actual conditions and SPEED with slope defined by the smallest flow given in the TABLE. Similarly the stonewall line is taken to be a linear relation between flow at actual conditions and SPEED with slope defined by the largest flow given in the TABLE.

If head as a function of actual flow relations are entered for more than one SPEED in the head-efficiency TABLE, and the current operating point is between two input speeds, then the head is obtained by interpolation. If the operating point lies outside the range defined by the table data entered and the compressor speed lies between entries for MAXIMUM.SPEED and MINIMUM.SPEED, the head is then calculated using data from the speed closest to the operating point.

Efficiency calculation

The hydraulic efficiency, eff, is calculated from the EFFICIENCY data entered using the head-efficiency TABLE. As in the case for head above, if EFFICIENCY as a function of FLOW data are entered only for one speed and entries have been made for MINIMUM.SPEED and MAXIMUM.SPEED, the value of eff is calculated for speeds within the permissible range as:

$$eff = eff_1 \bullet (F1/S)$$

where eff_1 is a function representing the EFFICIENCY data of the TABLE expressed in terms of the argument FI_1/S_1. FI_1 is the set of FLOW rates with associated EFFICIENCY values, and S_1 is the single reference SPEED.

If more than one SPEED entry is made, and the operating point lies between two of the speed curves, the value of eff is determined by interpolation.

Brake power equation

The brake power (PWR) required to be furnished by the gas turbine to the compressor is given by:

$$PWR = \frac{W \cdot H}{C \cdot eff \cdot ME}$$

where:

W	=	The mass rate of flow of the gas, $FI \cdot \rho_s$
FI	=	The compressor's inlet flow rate at suction conditions
ρ_s	=	The gas density at suction conditions
C	=	A dimensional constant determined by the units chosen
eff	=	The compressor hydraulic efficiency, a fraction
ME	=	Mechanical efficiency

The value of PWR is not allowed to exceed BPM, the Gas Turbine power available as evaluated from the minimum of the MAXIMUM.POWER TABLE or EXHAUST.TEMP TABLE and MAXIMUM.TURBINE.EXHAUST entries corrected for ambient temperature, pressure and duct losses. The maximum brake power available is computed as:

$$BPMIN = Min(BPMT, BPMX) \bullet \delta\ (1\text{-}DLOSS)$$

where BPMT and BPMX are defined respectively below and where:

δ	=	The ratio of inlet atmospheric pressure to the pressure at sea level
DLOSS	=	The duct loss
BPMT	=	$BP_1\ (T_s)S/S_1$

If MAXIMUM.POWER from MAXIMUM.POWER TABLE is entered for only one SPEED, then

BP_1	=	The MAXIMUM.POWER function of AMBIENT.TEMP entered
T_a	=	The current ambient temperature,
S	=	The current turbine speed
S_1	=	The reference SPEED

If more than one value for SPEED is entered, and the current operating speed, S, lies intermediate to the entered speed data, the value of BPMT is calculated by interpolation of the MAXIMUM.POWER as a function of AMBIENT.TEMP and SPEED table entries. If the operating point lies outside the given curves, the value of BPMT is computed using the closest data curve for $BP_1(T_a)$.

BPMX is computed as follows. From EXHAUST.TEMP in the TABLE entries, the exhaust temperature, Peek :TE, is calculated by table lookup from the EXHAUST.TEMP entries using current values of ambient temperature, compressor power and speed. Then

$$BPMX = PEX(NX)$$

where PEX(NX) is the power corresponding to the speed NX for which, at current ambient temperature, the exhaust temperature satisfies:

:TE = EXHAUST.TEMP

Note: BPM will not exceed the value entered as MAXIMUM.POWER as the absolute maximum rated power of the Gas Turbine.

Power turbine exhaust temperature

If the EXHAUST.TEMP TABLE entry is made, the power turbine exhaust Temperature is computed both for limiting the power by maximum exhaust temperature and for display as peek value :TE. This calculation is performed by table lookup from the data. The power argument used, BPWR, is the compressor power, :PWR, augmented by the duct losses.

The exhaust temperature (in TEMPERATURE units) is then computed by interpolation from EXHAUST.TEMP using BPWR, the ambient temperature, T_a, specified for the machine and its current speed. If an entry is made for MXTE, the power of the turbine is restricted so that the exhaust temperature will not exceed MXTE.

Thermal simulation

If a thermal simulation is performed, the absolute discharge temperature (T_D) is evaluated in terms of the absolute suction temperature by:

$$T_D = T_S(P_D / P_S)^a$$

where:

a	=	(TRR - 1) / (TRR * eff)
eff	=	Compressor efficiency
T_S	=	Absolute suction temperature
P_D/P_S	=	Ratio of the absolute discharge pressure to absolute suction pressure
TRR	=	Input via the keyword TRR.

Note: Typically the TRR is not equal to NPOLY but is adjusted for proper temperature rise.

Stability

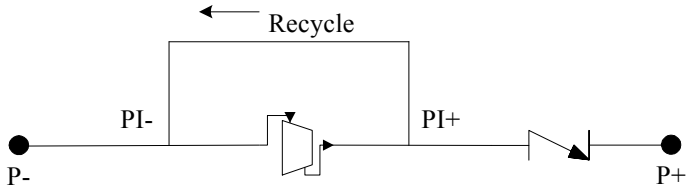
The ideal formulation of set points for the KC compressor may lead to unstable behavior in a transient environment. Specifically considerations of what happens with multiple compressors operating in parallel with the same type of set point (e.g. discharge pressure) become difficult. If the set points are the same and the compressors are in partial recycle, then the solution for flow through each compressor becomes indeterminate. If the set points are different, the solution can misbehave for other reasons. In a thermal simulation the problem of multiple compressors in parallel also becomes unmanageable if no steps are taken to stabilize the problem.

A similar problem occurs when a compressor starts against a pressure differential. Therefore, each compressor has implicit check valves. Thus as a compressor spins up, no flow can occur until the compressor's head exceeds the head imposed by the pressure differential. Because the head developed by the compressor increases with the square of the speed, the head rises very rapidly through the implicit check valve. On the time step at which the head first exceeds the line pressure, it may do so by a substantial amount. This is true even if the time step control is targeted to coincide with the time at which the head exceeds the pressure differential by a small amount. This is the result of head changing rapidly with speed and the pressure differential at the step end is not always well predicted by extrapolation. If the head exceeds the pressure differential by a small amount, the linearization of the compressor is seriously compromised. This

results in an enormous flow through the compressor on the first time step that the unit can overcome the pressure differential.

Implicit check valve

As a result of the stability problems, the implicit check valve is given a resistance to flow. Consider the following schematic:



If the pressure differential is negative, i.e., P+>PI+, then the check valve closes (valve resistance is set high). When the head generated by the compressor exceeds the pressure differential, the check valve opens and the resistance goes to zero. A small resistance is simulated when the compressor is actively recycling and the compressor is not flow controlled. In this case, the resistance is $.001*P_S/Q_{max}$.

Automatic power deration

Compressors with gas drivers (the default) have their power automatically derated for elevation, ambient temperature, and duct losses. Compressors with electric drivers do not have their power derated.

Example input

The following example defines a centrifugal compressor named COMP1:

```
KC COMP1 NODE1 NODE2
- CONTROL.SPEED 4659.
+ MAXIMUM.TURBINE.EXHAUST 972.
+ MINIMUM.SPEED 3036.
+ MAXIMUM.SPEED 4904.
+ NUMBER.RUNNING 1.
+ MECHANICAL.EFFICIENCY 0.97
+ MAXIMUM.POWER 45572.
+ START.TIME 10.
+ STOP.TIME 5.
+ AMBIENT.TEMPERATURE 59.
+ MAXIMUM.DISCHARGE.TEMP 150.
+ DISCHARGE.PRESSURE.SETPOINT 1850.
+ SUCTION.PRESSURE.SETPOINT 850.
+ FLOW.SETPOINT 1800.
+ TABLE   HEAD     EFFIC     FLOW     SPEED
         23500    .755       5400    4203
         23400    .782       6000
         23000    .797       6500
         22500    .807       7000
         21900    .81        7500
         21000    .807       8000
         19500    .795       8500
         18000    .775       9000
```

```
                17600    .767       9200
                26200    .75         5600    4437
                26180    .765       6000
                26100    .784       6500
                25600    .798       7000
                25100    .807       7500
                24300    .811       8000
                23500    .809       8500
                22400    .80        9000
                21000    .785       9500
                19200    .76       10000
                28700    .752        6150    4659
                28500    .768       6500
                28300    .785       7000
                28000    .798       7500
                27700    .807       8000
                27000    .811       8500
                26000    .81        9000
                25000    .805       9500
                23500    .79       10000
                22000    .771      10450
                32000    .757        6700    4904
                31900    .769       7000
                31800    .785       7500
                31500    .798       8000
                31000    .806       8500
                30100    .81        9000
                29200    .811       9500
                28200    .808      10000
                26700    .798      10500
                24200    .776      11100

+ TABLE   MAXIMUM.POWER   AMBIENT.TEMP   SPEED
                41867             0        4203
                39644            20
                37421            40
                35198            59
                32604            80
                30011           100
                27788           120
                44645             0        4659
                42237            20
                39644            40
                37050            59
                34271            80
                31493           100
                28899           120
                45942             0        4904
                43163            20
                40384            40
                37791            59
                34827            80
                31863           100
                29270           120
```

```
+ TABLE  HEAT.RATE     POWER        SPEED
          11449       18525        4203
          10295       22230
           9745       25935
           9443       29640
           9274       33345
           9230       35198
          11449       18525        4659
          10295       22230
           9656       25935
           9274       29640
           9035       33345
           8875       37050
          13046       14820        4904
          11536       18525
          10295       22230
           9674       25935
           9230       29640
           8964       33345
           8786       37050
           8698       38532
+ TABLE    EXHAUST.TEMP    POWER    AMBIENT.TEMP    SPEED
            730          25935              0        3269
            813         29640
            903         33345
           1001         37050
           1054         38902
            784          25935             20
            864         29640
            959         33345
           1082         37050
            831          25935             40
            917         29640
           1037         33345
            876          25935             59
            981         29640
            937          25935             80
           1000         27602
            690          25935              0        3736
            761         29640
            835         33345
            914         37050
            955         38902
            745          25935             20
            811         29640
            884         33345
            968         37050
           1016         38902
            790          25935             40
            857         29640
            937         33345
           1043         37050
            830          25935             59
            904         29640
```

```
1003       33345
 878        25935             80
 971       27602
1000       30455
 668        25935              0        4203
 732       29640
 797       33345
 864       37050
 899       38902
 942       41162
 723        25935             20
 781       29640
 843       33345
 911       37050
 960       39495
 766        25935             40
 824       29640
 888       33345
 964       37050
 804        25935             59
 865       29640
 937       33345
 972       35197
```

Theoretical horsepower-flow compressor (KP)

Peek and Poke Attributes

INPREP

```
KP NAME FROM TO [FCNC]
±RATED.POWER RP

/* Select the calculation method by entering either

/* PARAMETERS or EFFICIENCY and NPOLY
[+ PARAMETERS K1 K2 K3]
[+ EFFICIENCY EFF]

/* The following are optional

[+ MAXIMUM.POWER MXPW]
[+ MINIMUM.POWER MNPW]
[+ POWER.SETPOINT SPW]
[+ MAXIMUM.DISCHARGE.PRESSURE MXP+]
[+ DISCHARGE.PRESSURE.SETPOINT SP+]
[+ MINIMUM.SUCTION.PRESSURE MNP-]
[+ SUCTION.PRESSURE.SETPOINT SP-]
[+ MAXIMUM.FLOW MXQ]
[+ MINIMUM.FLOW MNQ]
[+ FLOW.SETPOINT SQ+]
[+ MAXIMUM.RATIO MXR]
[+ MINIMUM.RATIO MNR]
[+ RATIO.SETPOINT SR]
[+ BYPASSCHECK]
[+ TRIPS  YES | NO ]
[+ SDROP P- P+ Q]
[+ DDROP P- P+ Q]
[+ START.TIME START]
[+ STOP.TIME STOP]
[+ NUMBER.RUNNING NRN]
[+ NPOLY NPOLY]
[+ TRR TRR]
[+ NO.FUEL]
[+ HEAT.RATE HR]
[+ TABLE HEAT.RATE POWER]
```

Field	Units Key	Description
NAME	n/a	Name of the compressor.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.

Field	Units Key	Description
FCNC	n/a	Name of node from which compressor draws fuel. See "Compressor fuel" on page 337.
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.
RP	POWER	Rated power for spin up and spin down sequencing.
K1	POWER/FLOW	Compressor equation coefficient. {193 hp/(mmscf/d)}
K2	POWER/FLOW	Compressor equation coefficient. {192 hp/(mmscf/d)}
K3	dimensionless	Compressor equation exponent. {.231}
EFF	EFFICIENCY	Efficiency. {1.0}
MXPW	POWER	Maximum allowable compressor power constraint. Defaults to the RATED.POWER value.
MNPW	POWER	Minimum allowable compressor power constraint. {0}
SPW	POWER	Compressor power set point. {2 * MXPW}
MXP+	PRESSURE	Maximum discharge pressure constraint. {10,000 psig}
SP+	PRESSURE	Discharge pressure set point. {10,000 psig}
MNP-	PRESSURE	Minimum suction pressure constraint. {0 psig}
SP-	PRESSURE	Suction pressure set point. {0 psig}
MXQ	FLOW	Maximum flow constraint. {10,000 $MMFT^3/D$}
MNQ	FLOW	Minimum flow constraint. {0 $MMFT^3/D$}
SQ+	FLOW	Flow set point. {10,000 $MMFT^3/D$}
MXR	FRACTION	Maximum compressor ratio constraint. {10}
MNR	FRACTION	Minimum compressor ratio constraint. {1}
SR	FRACTION	Compression ratio set point. {10}
BYPASSCHECK	n/a	Check valve on parallel bypass line.
P-	PRESSURE	Pressure at the - connection. {0}
P+	PRESSURE	Pressure at the + connection. {0}
Q	FLOW	Flow associated with the pressure loss. {0}
START	TIME	Spin up time of the compressor. {3 minutes}
STOP	TIME	Spin down time of the compressor. {3 minutes}
NRN	dimensionless	Number of units running in parallel. {1}
NPOLY	dimensionless	Polytropic exponent. {1.3}

Field	Units Key	Description
TRR	n/a	Temperature rise ratio used to determine discharge temperature. Defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.
TRIPS	n/a	If YES, violations of physical limitations cause the compressor to trip (shut down). If NO, the compressor will not shut down. {YES}

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Description

The theoretical horsepower-flow compressor (KP) is an idealized compressor used to theoretically model any type of compressor. The KP is an idealized control element, which means that no control system is necessary for the compressor to hold set points. The compressor has logic which acts like a control system in maintaining set points and constraints. When a compressor cannot operate with all its constraints satisfied, it is automatically shut down.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

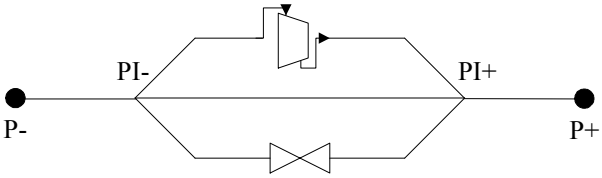
Horsepower calculation

If you enter EFF (and not PARAMETERS, or neither EFF nor PARAMETERS, in which case EFF defaults to 1.0), the hydraulic power required by the compressor is given by:

$$PWR = \frac{C\,Q\,(P_{ref}/\gamma)(R^{r}-1)T_S Z_S}{T_{ref}\cdot EFF}$$

where:

C	=	A dimensionless constant that relates power units to time rate of work {3.0303 if P_{REF} in psia, flow is in MMSCFD, and power is in horsepower}
Q	=	Flow rate at custody conditions
Z	=	Compressibility
T_S	=	Suction flowing temperature (absolute)
γ	=	(NPOL-1)/NPOL
P_{REF}	=	Custody transfer pressure (absolute)
R	=	Compression ratio PI+/PI- (both in absolute)
T_{REF}	=	Custody transfer temperature (absolute)



In the event that you enter a line with the keyword PARAMETERS, the power required from the driver is given, using the same notation by:

$$:\mathrm{PWR} = \mathrm{Q}\left[\mathrm{K}_1\left(\mathrm{R}^{\mathrm{K}3}-1\right)+\mathrm{K}_2\left(\mathrm{F}^{\mathrm{K}3}-1\right)\right]\mathrm{T}_S\,\mathrm{Z}_S\,/\,\mathrm{T}_{\mathrm{REF}}$$

where K_1, K_2, and K_3 are user-defined data, and F is $(K_1/K_2)^{1/K3}$. The term $K_1(R^{K3} - 1)$ plays the role of $C(P_{REF}/\gamma)(R\gamma - 1)$ with $K_3 = \gamma$. The term $K_2(F^{K3} - 1)$ represents an additional power loss term with effective ratio equal to F, now with K_2 playing the role of $C(P_{REF}/\gamma)$. Of course, to be physically meaningful, $K_1 > K_2 > 0$. Also, if $K_2 = K_1$, then the term multiplying K_2 is zero and the power equations are identical if $K_1 = C(P_{REF}/\gamma)$, $K_3 = \gamma$ and :EFF=1.

Suction and discharge piping losses

Suction and discharge drop pressures and flows are used to calculate a resistance. This resistance is used to calculate the suction and discharge piping pressure losses at varying pressures and flows.

Bypass

There is an implicit check valve in series with the compressor to prevent back flow through the compressor. There is an implicit bypass block valve around the compressor, which always allows positive bypass flow, and allows negative bypass flow unless the optional bypass check valve is specified. Parallel units should have the BYPASSCHECK specified.

Set points

The set point values are targets for compressor operation. The set point that uses the lowest power is met. However, set points may be violated if holding a set point causes a compressor constraint to be violated.

Constraints

The user-defined set points are used to control the compressor unless one of the user-specified constraints is violated. If the compressor operation gets to a point where it cannot be operated without violating a constraint, the compressor is shut down. (Such as operating at minimum ratio and violating the maximum discharge pressure constraint.) The basic constraint logic is as follows:

- Rate of Change of Power - power cannot be increased faster than (rated power)/(start time)
- Rate of Change of Power - power cannot be decreased faster than (rated power)/(stop time)
- Maximum Discharge Pressure - hold pressure at or below the specified pressure
- Minimum Suction Pressure - hold pressure at or above the specified pressure
- Maximum Flow - hold the flow rate at or below the specified flow
- Minimum Flow - hold the flow at or above the specified flow
- Maximum Ratio - hold the compression ratio at or below the specified ratio

- Minimum Ratio - hold the compression ratio at or above the specified ratio
- Maximum Power - hold the compressor power at or below the specified power
- Minimum Power - hold the compressor power at or above the specified power

Temperature rise

The temperature rise across the compressor is calculated in the following manner:

$$T_D = T_S \; R^{\frac{TRR-1}{TRR}}$$

Start and stop times

The start and stop behavior of the compressor is governed by the user-defined start and stop time and the rated power. On startup, the compressor power is linearly ramped at the rate of rated power divided by start time. During startup, the minimum constraints on power, flow and ratio are ignored. Once the compressor has reached another set point or constraint and the minimum constraints on power, flow and ratio are satisfied, the status goes from starting to running. On shut down, the compressor power is ramped down at the rate of rated power divided by the stop time. The rate of change of power also constrains the compressor behavior when the compressor power is being ramped due to changes in set point values or governing constraints.

Rated power

The user-defined value for rated power should be a reasonable value. If the rated power is not known, a reasonable approximation should be entered and a large number can be entered for the maximum power.

Example input

A compressor named COMP1 is connected to nodes STA1S and STA1D. Fuel is to be taken automatically from the suction. Rated power is 3100, K1 = 193, K2 = 192, and K3 = .231. Discharge pressure set point is 985, maximum discharge pressure is 1000. For suction drop, - 605 is the inlet pressure, 600 is the suction flange, and flow is 500. For discharge drop, 990 at the flange, 985 out of the station, and flow is 500. The maximum power is 10,000 and the heat rate is 9167.

```
KP COMP1 STA1S STA1D STA1S
+ RATED.POWER 3100
+ PARAMETERS 193 192 .231
+ MAX.DISC.PRESS 1000, DISCH.PRESS.SETP 985
+ MAX.POWER 10000
+ SDROP 605 600 500
+ DDROP 990 985 500
+ HEAT.RATE 9167
```

Variable guide vane compressor (KV)

Peek and Poke Attributes

```
KV NAME FROM TO [FCNC]
±TABLE    HEAD  EFFICIENCY  FLOW  ANGLE
+ SPEED SPD

/*ALL INPUT BELOW IS OPTIONAL

[+ USEUNIT.POWER UNAME]
[+ HYDRAULIC.EFFICIENCY.FACTOR HEC]
[+ NUMBER.RUNNING NRN]
[+ MINIMUM.ANGLE MNA]
[+ MAXIMUM.ANGLE MXA]
[+ SET.ANGLE SA]
[+ MECHANICAL.EFFICIENCY ME]
[+ MAXIMUM.POWER MXPW]
[+ MINIMUM.POWER MNPW]
[+ POWER.SETPOINT SPWR]
[+ START.TIME START]
[+ STOP.TIME STOP]
[+ MAXIMUM.DISCHARGE.TEMP MXT+]
[+ DISCHARGE.PRESSURE.SETPOINT SP+]
[+ MAXIMUM.DISCHARGE.PRESSURE MXP+]
[+ SUCTION.PRESSURE.SETPOINT SP-]
[+ MINIMUM.SUCTION.PRESSURE MNP-]
[+ FLOW.SETPOINT SQ+]
[+ NPOLY NPOLY]
[+ TRR TRR]
[+ ANGLE.MOVEMENT.TIME AMT]
[+ SURGE.CONTROL.METHOD  RECYCLE | TRIP ]
[+ TRIPS  YES | NO ]
[+ ANGLE.CONTROL.METHOD AUTO | MANUAL]
[+ NO.FUEL]
[+ HEAT.RATE HR]
[+ TABLE HEAT.RATE  POWER]
```

Mandatory entries

Field	Units Key	Description
NAME	n/a	Name of the variable guide vane compressor.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.
SPD	SPEED	Speed at which the compressor will operate.

Table entries

Field	Units Key	Description
HEAD	HEAD	Compressor head. See "Compressor TABLE" on page 335 for rules.
EFFICIENCY	EFFICIENCY	Polytropic efficiency (decimal values). See "Compressor TABLE" on page 335 for rules.
FLOW	COMP.AFLOW	The flow at suction conditions to the compressor. See "Compressor TABLE" on page 335 for rules.
ANGLE	ANGLE	The gas compressor vane angle for which the head and efficiency as a function of flow relationships apply. See "Compressor TABLE" on page 335 for rules.

Optional entries

Field	Units Key	Description
FCNC	n/a	Name of node from which compressor draws fuel. See "Compressor fuel" on page 337.
UNAME	n/a	Power unit other than the default to be used for this compressor.
HEC	EFFICIENCY	Multiplier of table hydraulic efficiency values. {1}
NRN	dimensionless	Number of identical compressors running in parallel. {1}
MNA	ANGLE	Minimum guide vane angle must be greater than or equal to the minimum angle in the table. {Minimum angle in the head-efficiency table.}
MXA	ANGLE	Maximum guide vane angle must be less than or equal to the maximum angle in the table. {Maximum angle in the head-efficiency table.}
SA	ANGLE	Guide vane angle set point. None denotes angle is dynamic. {none}
ME	EFFICIENCY	The mechanical efficiency. {0.97}
MXPW	POWER	Maximum permissible Turbine power under any conditions. {100,000 HPR}
MNPW	POWER	Minimum permissible power. {0}
SPWR	POWER	Power set point. {100,000 HP}
START	TIME	Spin-up time for starting compressor. {3 minutes}
STOP	TIME	Spin-down time for stopping compressor. {3 minutes}
MXT+	TEMPERATURE	High discharge temperature trip. {500· FR}
SP+	PRESSURE	Discharge pressure set point. {10,0000 psig}
MXP+	PRESSURE	Maximum discharge pressure constraint. {10,000 psig}
SP-	PRESSURE	Suction pressure set point. {0}
MNP-	PRESSURE	Minimum suction pressure constraint. {0}
SQ+	FLOW	Flow set point. {10,000 $MMFT^3/D$}

Field	Units Key	Description
NPOLY	dimensionless	Polytropic exponent. {If you use an equation of state that permits evaluation of heating value (BWRS or AGA), the value is calculated as compositions change upstream of the compressor; otherwise, the default is 1.3. Note that a user-entered value always overrides a calculated value. If you want to get the default behavior, either do not enter NPOLY or set NPOLY to 0 in TRANS.}
TRR	n/a	Temperature rise ratio defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.
AMT	TIME	Time to move from minimum angle to maximum angle. {1 minute}
RECYCLE	n/a	If the compressor surges, flow is recycled to correct the surge condition. This is the default.
TRIP	n/a	If a surge condition exists, the compressor trips (shuts down).
YES	n/a	Violations of physical limitations cause the compressor to trip (shut down). This is the default.
NO	n/a	Violations of physical limitations are ignored.
AUTO	n/a	Idealized control automatically adjusts guide vane angle to satisfy set points and constraints when no actuator is attached. This is the default.
MANUAL	n/a	After startup, the compressor operates at the guide vane angle set point and other set points are ignored.

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Description

The variable guide vane compressor (KV) is used to model single or multiple (NRN > 1) variable guide vane compressors operating in parallel and simulated as a single unit. This compressor may be driven by either a fixed speed gas turbine or by an electric driver.

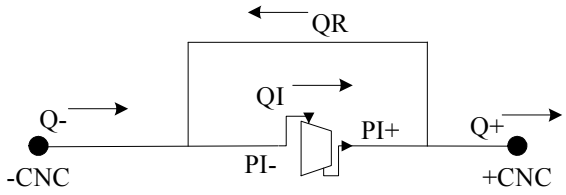
Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Flows

For a discussion of flows, consider the following schematic:



There are several definitions for flow (at standard conditions) for the KV compressor. Q- is used to denote the flow into the compressor from the connecting inlet device. QI is used to denote the total flow through the compressor. QR is used to denote recycle flow. Q+ is the flow out of the compressor to the connecting outlet device.

An additional flow variable, FI, is also defined. FI is QI converted to volumetric flow rate at inlet conditions. As such, FI is the flow on which surge and stonewall control is based. FI is in COMP.AFLOW units at inlet conditions.

Compressor control

You can choose one of three modes for compressor control that are available after the compressor reaches its fixed running speed:

- Constant speed. Changes in required head or flow are accommodated through changes in guide vane angle. [Default mode]
- Constant guide vane angle and constant speed. (Manual)
- Actuator control of the angle.

Manual set guide vane position

In this mode, set the guide vane angle, :SA. The value of the set guide vane angle, :SA, must be between :MXA and :MNA. Any other user-defined set points (suction pressure, discharge pressure, flow, or power) are ignored. Violation of any of the constraints: maximum power, minimum power, maximum discharge temperature, maximum discharge pressure, or minimum suction pressure, will shut down the compressor, if trips are enabled. The unit may be restarted after automatic shutdown by issuing a START command.

Automated control of guide vane angle

Using this mode, control of flow and head are accomplished through changes in inlet guide vane angle. Angle movement is restricted to :MXA and :MNA. The guide vane angle changes to meet the most controlling of the user-defined set points, :SQ+, :SP+, :SP-, and :SPWR.

The user-defined set points (suction pressure, discharge pressure, flow, or power) are used to control the compressor unless one of the user-specified constraints (maximum discharge pressure, minimum suction pressure, maximum discharge temperature, minimum power, or maximum power) is violated. If a constraint is violated and the violation can be cleared by ignoring a set point, the control will be based on the constraint; except for :MXT+, which causes a trip. If the compressor operation gets to a point where it cannot be operated without violating a constraint, the compressor is shut down, if trips are enabled. The unit may be restarted after automatic shutdown by issuing a START command.

Actuator control of guide vane position

In this mode, an actuator and accompanying control system are used to determine the guide vane angle. The angle movement is between :MXA and :MNA. Any user-defined set points are ignored. Violation of any of the constraints: maximum power, minimum power, maximum discharge temperature, maximum discharge pressure, or minimum suction

pressure, will shut down the compressor, if trips are enabled. The unit may be restarted after automatic shutdown by issuing a START command.

Surge and stonewall control

In all modes, the KV unit functions to avoid both the surge and stonewall lines. Surge is prevented by the method selected with SURGE.CONTROL.METHOD. If the method is recycle, the rate of gas recycle at custody transfer conditions is peekable using the attribute :QR.

Prevention of stonewall conditions is accomplished by increasing the effective internal resistance to flow of the unit until the resulting head-flow satisfies the stonewall constraint.

START/STOP control

A compressor that is STOPPED may be started by issuing a START command in the INTRAN file, or by issuing a START command interactively. The spin-up time on starting is given as START.TIME in TIME units.

Similarly, a compressor that is RUNNING can be stopped by issuing a STOP command in the INTRAN file, or by issuing a STOP command interactively. The spin-down time on stopping is given as STOP.TIME in TIME units.

When a KV compressor is started, the angle is set at the minimum angle. The compressor is brought up to the specified speed in the user-specified start time. Once the compressor is up to speed, the angle will go to the set angle or be adjusted to meet pressure and flow set points.

Equations

Head calculation equation

The compressor's developed head (in HEAD units) is evaluated from the TABLE of head and efficiency entries. At a minimum, data at two different angles is required.

The surge line is taken to be a linear relation between flow at actual conditions and ANGLE with slope defined by the smallest flow given in the TABLE. Similarly the stonewall line is taken to be a linear relation between flow at actual conditions and ANGLE with slope defined by the largest flow given in the TABLE.

The current operating point must be between two input angles, and the head is obtained by interpolation. The angles entered must cover the operating range of the unit. No extrapolation outside the given table values will be done.

Efficiency calculation

The hydraulic efficiency, eff, is calculated from the EFFICIENCY data entered using the head-efficiency TABLE. The efficiency value is determined by interpolation.

Brake power equation

The brake power, :PWR, required to be furnished by the driver to the compressor is given by:

$$:\mathrm{PWR} = \mathrm{W\,H} / \{\mathrm{HEC}\,(\mathrm{C\,eff})\,\mathrm{ME}\}$$

Where W, the mass rate of flow of the gas, is given by:

$$\mathrm{W} = \mathrm{FI}\rho_s$$

where:

C	=	a dimensional constant determined by the units chosen
ρ_s	=	the gas density at suction conditions
FI	=	the compressor's inlet flow rate at suction conditions
eff	=	the compressor polytropic efficiency, a fraction

The value of :PWR is not allowed to exceed MAXIMUM.POWER or to be less than MINIMUM.POWER.

Thermal simulation

If a thermal simulation is performed, the absolute discharge temperature (T_D) is evaluated in terms of the absolute suction temperature by:

$$T_D = T_S(P_D / P_S)^a$$

where:

a	=	(TRR - 1) / (TRR)(eff)
eff	=	Compressor polytropic efficiency
T_S	=	Absolute suction temperature
P_D/P_S	=	Ratio of the absolute discharge pressure to absolute suction pressure
TRR	=	Input via the keyword TRR.

Note: Typically the TRR is not equal to NPOLY but is adjusted for proper temperature rise.

Stability

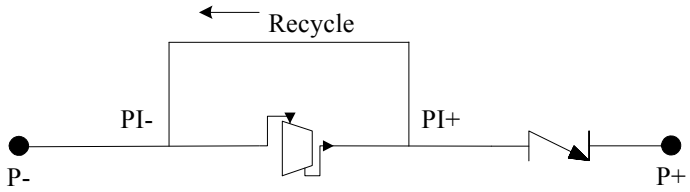
The ideal formulation of set points for the KV compressor may lead to unstable behavior in a transient environment. Specifically, considerations of what happens with multiple compressors operating in parallel with the same type of set point (e.g. discharge pressure) become difficult. If the set points are the same and the compressors are in partial recycle, then the solution for flow through each compressor becomes indeterminate. If the set points are different, the solution can misbehave for other reasons. In a thermal simulation the problem of multiple compressors in parallel also becomes unmanageable if no steps are taken to stabilize the problem.

A similar problem occurs when a compressor starts against a pressure differential. Therefore, each compressor has an implicit check valve. Thus as a compressor spins up, no flow can occur until the compressor's head exceeds the head imposed by the pressure differential. On the time step at which the head first exceeds the line pressure, it may do so by a substantial amount. This is true even if the time step control is targeted to coincide with the time at which the head exceeds the pressure differential by a small amount. This is the result of head changing rapidly with speed and angle and the pressure differential at the step end is not always well predicted by extrapolation. If the head exceeds the pressure differential by a small amount, the linearization of the compressor is seriously compromised. This results in an enormous flow through the compressor on the first time step that the unit can overcome the pressure differential.

Implicit check valve

As a result of the stability problems, the implicit check valve is given a resistance to flow.

Consider the following schematic:



If the pressure differential is negative, i.e., P+>P-, then the valve resistance is set high. When the head generated by the compressor exceeds the pressure differential, the check valve opens and the resistance goes to .001*P_S/Q_{max}.

Example input

The following example defines a variable guide vane compressor named COMP1:

```
KV COMP1 NODE1 NODE2
+ SPEED  5000
+ MECHANICAL.EFFICIENCY  0.97
+ MAXIMUM.POWER  45572.
+ START.TIME  10.
+ STOP.TIME  5.
+ MAXIMUM.DISCHARGE.TEMP  150.
+ DISCHARGE.PRESSURE.SETPOINT  1850.
+ SUCTION.PRESSURE.SETPOINT  850.
+ FLOW.SETPOINT  1800.
+ TABLE HEAD       EFFIC     FLOW       ANGLE
       16200       .76       5600       -50
       16000       .79       7000
       15500       .81       8200
       14800       .82       9400
       14400       .82       10100
       13400       .81       11200
       12400       .79       12200
       10600       .73       13800
       17000       .76       7200       0
       16600       .81       9800
       16200       .83       11000
       13800       .83       14000
       12800       .81       14800
       12000       .79       15400
       10800       .75       16800
```

Reciprocating compressor (RC)

Peek and Poke Attributes

INPREP

```
RC NAME FROM TO [FCNC]
±MAXIMUM.POWER PWMAX
+ MAXIMUM.SPEED SMAX
+ TABLE   CLEARANCE.RATIO   SWEPT.VOLUME

/* CONSTANT TABLE ITEMS MAY BE ENTERED
/* IN NON-TABLE FORMAT USING THE TABLE HEADING AS KEYWORD

[+TABLE  COMPRESSOR.EFFICIENCY  COMPRESSION.RATIO]
[+COMPRESSOR.EFFICIENCY CE]
[+TABLE   MECHANICAL.EFFICIENCY   SPEED]
[+MECHANICAL.EFFICIENCY ME]
[+VOLUMETRIC.EFFICIENCY.CORRECTION LC [LM]]
[+MINIMUM.SPEED SMIN]
[+MINIMUM.TORQUE TQMIN]
[+MAXIMUM.TORQUE TQMAX]
[+MAXIMUM.RATIO RMAX]
[+DISCHARGE.PRESSURE.SETPOINT PDMAX]
[+SUCTION.PRESSURE.SETPOINT PSMIN]
[+FLOW.SETPOINT SPQ]
[+SPEED.SETPOINT SPS]
[+SPEED.DEAD.BAND DBS]
[+TORQUE.SETPOINT SPT]
[+TORQUE.DEAD.BAND DBT]
[+UNLOADING.STEP.NUMBER STEP]
[+UNLOADING.GOAL UG]
[+NPOLY NPOLY]
[+TRR TRR]
[+START.TIME START]
[+STOP.TIME STOP]
[+DELAY DEL]
[+TRIPS TRP]
[+NO.FUEL]
[+TABLE   HEAT.RATE POWER [SPEED]]
[+HEAT.RATE HR]
```

Mandatory Keyword Entries

Field	Units Key	Meaning
NAME	n/a	Name of the reciprocating compressor.
FROM	n/a	Name of connecting inlet device or named node.
TO	n/a	Name of connecting outlet device or named node.

Field	Units Key	Meaning
±	n/a	Initial status of the compressor: + for STARTING, - for STOPPED.
PWMAX	POWER	The maximum permissible power at which the compressor may operate while running at maximum speed.
SMAX	SPEED	The maximum permissible speed at which the compressor may operate.

Note: Rated torque is determined by the ratio PWMAX / SMAX. In this document, Torque is expressed as a percentage of rated torque, i.e.,: Torque = 100*(POWER / SPEED) / (PWMAX / SMAX)

Mandatory Table Entries

Field	Units Key	Meaning
CLEARANCE.RATIO	Fraction	Clearance ratio, m, which is the ratio of clearance volume to swept volume, corresponding to values of the SWEPT.VOLUME on the same line. Each entry defines an unloading step. The unloading steps are numbered as they are defined in increasing sequence. See "Compressor TABLE" on page 335 for rules.
SWEPT.VOLUME	SWEPT.VOLUME	Swept or displacement volume corresponding to the clearance ratio. See "Compressor TABLE" on page 335 for rules.

Optional Table Entries

COMPRESSOR.EFFICIENCY	EFFICIENCY	Hydraulic compressor efficiency. See "Compressor TABLE" on page 335 for rules.
COMPRESSION.RATIO	Dimensionless	Compression ratio. See "Compressor TABLE" on page 335 for rules.
Note: COMPRESSOR.EFFICIENCY may optionally be entered using the keyword COMPRESSOR.EFFICIENCY. In this case the hydraulic compressor efficiency used for all compression ratios will be the user-defined value CE. {1}		
MECHANICAL.EFFICIENCY	EFFICIENCY	Mechanical efficiency. See "Compressor TABLE" on page 335 for rules.
SPEED	SPEED	Operating speed of the compressor. See "Compressor TABLE" on page 335 for rules.
Note: MECHANICAL.EFFICIENCY may optionally be entered using the keyword MECHANICAL.EFFICIENCY. In this case the mechanical efficiency used for all values of speed will be the user-defined value ME. {.95}		

Optional Keyword Entries

Field	Units Key	Meaning
FCNC	n/a	Name of node from which compressor draws fuel. See "Compressor fuel" on page 337.
LC	Dimensionless	Additive correction to volumetric efficiency. $$VE = 1 - LC - LM \bullet R - m(R^{1/n} - 1)$$ where R is compression ratio, m is clearance ratio and n is the polytropic exponent. {.04}
LM	Dimensionless	Multiplicative correction to volumetric efficiency. See the formula for *LC* above. {0}
SMIN	SPEED	Minimum speed at which the compressor is allowed to run. Once the compressor has ramped up to running status, speed is forced to be at or above minimum speed. {0 rpm}
TQMIN	FRACTION.TORQUE	Minimum torque, as a percentage of rated torque, at which the compressor is allowed to run. If conditions force the compressor to run at less than minimum torque for longer than the delay time, DEL, then the compressor is automatically shut down, if trips are enabled. {80 percent}
TQMAX	FRACTION.TORQUE	Maximum torque, as a percentage of rated torque, at which the compressor is allowed to run. If conditions force the compressor to run at greater than maximum torque for longer than the delay time, DEL, then the compressor is automatically shut down, if trips are enabled. {100 percent}
RMAX	Dimensionless	Maximum compression ratio at which the compressor is allowed to run. If conditions force the compressor to run at greater than maximum compression ratio, then the compressor is automatically shut down, if trips are enabled. {10}
PDMAX	PRESSURE	Discharge pressure set point. {10,000 psig}
PSMIN	PRESSURE	Suction pressure set point. {0 psig}
SPQ	FLOW	Flow set point. Pokable. {10,000 MMFT3/D}
SPS	SPEED	Speed set point. Used only as an unloading goal. If the unloading goal is to target the speed set point, then when operation has been outside of the speed dead band around the speed set point for longer than the delay time, DEL, the unloading step is automatically incremented or decremented to attempt to bring operation within the dead band. {SMAX}
DBS	SPEED	Speed dead band. See the description of the speed set point, *SPS*, earlier in this table. {5 rpm}

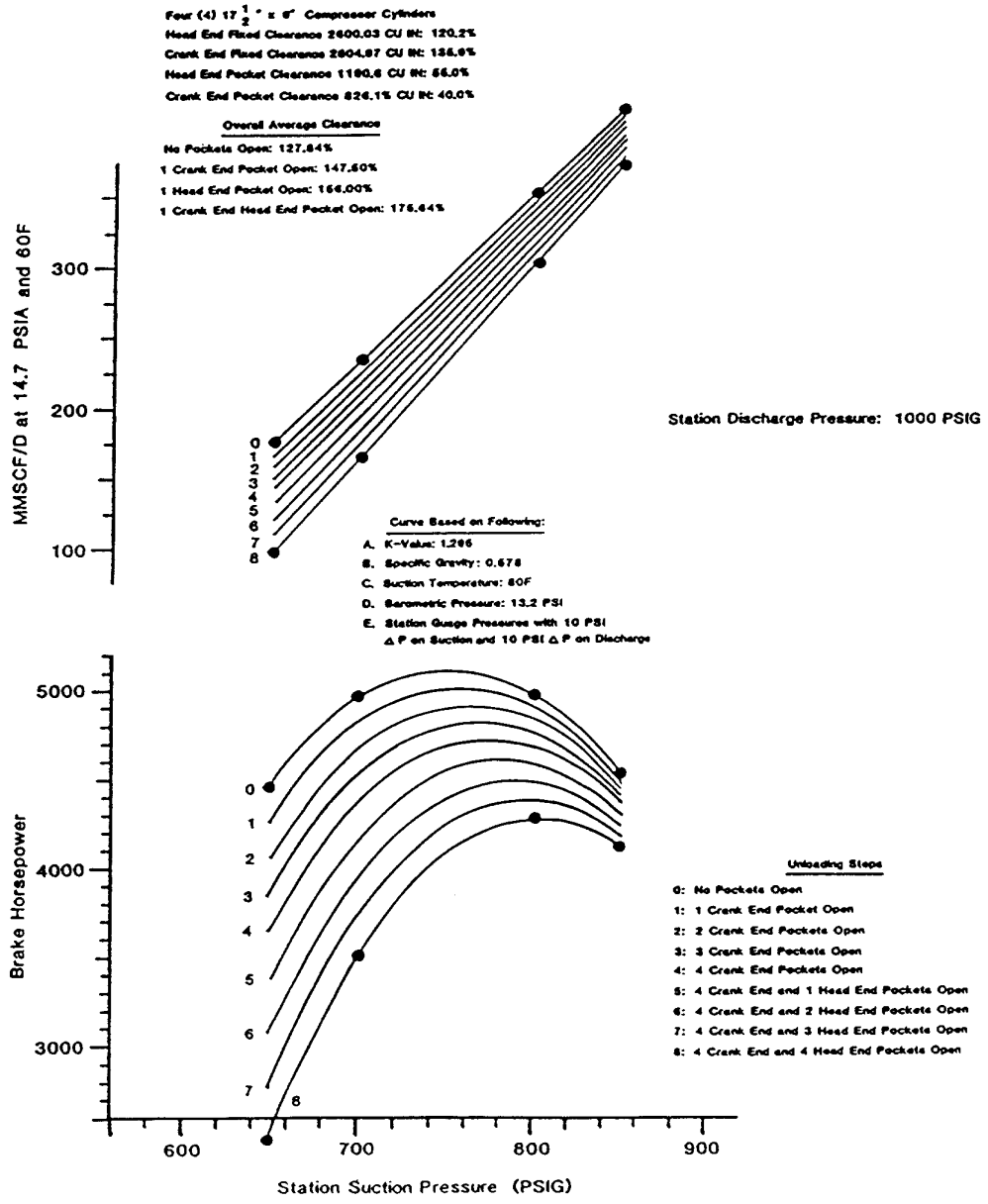
Field	Units Key	Meaning
SPT	FRACTION.TORQUE	Torque set point. Used only as an unloading goal. If the unloading goal is to target the torque set point, then when operation has been outside of the torque dead band around the torque set point for longer than the delay time, DEL, the unloading step is automatically incremented or decremented to attempt to bring operation within the dead band. {TQMAX}
DBT	FRACTION.TORQUE	Torque dead band. See the discussion of the torque set point, *SPT*, earlier in this table. Pokable. {1 percent}
STEP	Dimensionless	Unloading step set point. Used only as an unloading goal. If the unloading goal is to target the unloading step set point, then the unloading step remains fixed at the user-defined value. {1}
UG	n/a	Unloading goal. The choices for unloading goal are TORQUE.SETPOINT, SPEED.SETPOINT or FIXED.UNLOADING.STEP. (For pokes, the choices are denoted SPT, SPS and MANUAL.) {TORQUE.SETPOINT}
NPOLY	Dimensionless	Polytropic exponent. {1.3}
TRR	Dimensionless	Temperature rise ratio used to determine discharge temperature. Defaults to the initial value of NPOLY. If you want to track NPOLY, you need a RAMP statement to have TRR track NPOLY.
START	TIME	Spin-up time for starting the compressor. {.5 min}
STOP	TIME	Spin-down time for stopping the compressor. {.5 min}
DEL	TIME	Time delay for both automatic shut down, i.e., tripping, and automatic unloading step selection. {3 min}
TRP	n/a	Trips flag. Entered as a YES or NO. Determines whether or not automatic shut-down, tripping, in response constraint violations is enabled. {YES}

Note: The optional fuel entries—NO.FUEL, HEAT.RATE, and TABLE HEAT.RATE POWER [SPEED]—are common to multiple compressor types. These fuel entries are described in "Compressor fuel" on page 337.

Description

The reciprocating compressor (RC) is used to model a single reciprocating gas compressor. The RC compressor has two independent control parameters, speed and unloading step. Speed varies continuously, while the unloading step, i.e., a clearance ratio and a swept volume, comes from a finite list defined by table entry. The values of the control parameters, together with the compressor's operating environment, determine the state of the compressor. Compressor operation may be simulated with idealized control or with speed controlled by an actuator.

Typical operating curve



Equations

Polytropic Power Equation

The polytropic compressor cycle is used to model reciprocating compressors. The indicated power is determined by:

$$IHP=NRN\ DSV\ (RPM/\gamma)P_S[r^{\gamma}-1-m(1+r-r^{1/n}-r^{\gamma})]UM$$

where:

NRN	=	Number of identical units running
DSV	=	Current displacement

RPM	=	Actual running speed
γ	=	(n-1)/n
n	=	Polytropic exponent
r	=	Compression ratio P_D/P_s
P_D	=	Discharge pressure in absolute units
P_s	=	Suction pressure in absolute units
m	=	Current clearance ratio
um	=	Units multiplier

The value of DSV, which is a unit's total displacement (swept volume) in consistent units, depends on which unloading step selection mode is chosen. The value of m in any case is the ratio of the total clearance volume to the total swept or displacement volume.

Developed Power Equation

The developed power, HP, is given by:

$$HP = IHP\ (PCF(r) / ME)$$

where:

IHP	=	Indicated power
PCF(r)	=	PCF data curve evaluated at current value of r (compression ratio)
ME	=	Mechanical efficiency

Flow Equation

The compressor's capacity, Q, at custody transfer conditions (P_{ref},T_{ref}) is given by:

$$Q = NRN \cdot (DSV / Z_S) \cdot RPM \cdot (P_S / P_{ref}) \left(1 - CCF - m\left(r^{1/n} - 1\right)\right) (T_B / T_S) \cdot UM$$

where:

NRN	=	Number of identical units running
DSV	=	Current displacement
RPM	=	Current running speed
T_s	=	Suction temperature in absolute units
n	=	Polytropic exponent
r	=	Compression ratio
Z_s	=	Suction compressibility
T_B	=	Absolute temperature at which standard volume is referenced (from CUSTODY line, T_{ref} converted to absolute temperature.
P_{ref}	=	Reference pressure from CUSTODY directive
T_{ref}	=	Reference temperature from CUSTODY directive
CCF	=	Capacity correction factor
m	=	Clearance ratio
UM	=	Units multiplier

In the flow equation, the value of Z_s depends upon the equation of state used in the simulation.

Power and capacity correction methods

The reciprocating compressor is designed to model single stage reciprocating compressors driven by a gas engine. Compression control is obtained by compressed volume restraint using discrete, step type cylinder unloading as opposed to *infinite step*, or continuous unloading. This step type control is the most widely used for reciprocating compressors and allows these types of compressors to operate at constant speed over a wide range of suction and discharge conditions and volumes compressed. Thus, individual units can be sized and equipped to run efficiently under varying load conditions.

Field installed reciprocating compressors rarely attain the ideality of the classic thermodynamic isentropic compression cycle. Thus, empirical correction factors are needed to account for an actual compressor's deviations from that of the isentropic cycle representation. The isothermal compression cycle is even less attainable due to the impracticality of building sufficient cylinder cooling to carry away the heat developed during the compression process.

The need for empirical correction factors arises as the result of operational deviations caused by such realities as:

- Friction between moving parts as well as frictional resistance to flow
- Piston ring leakage, blow-by and seal leakage
- Dilution of suction gas charge by re-expanded gas contained in unavoidable (or purposeful) clearance spaces
- Gas turbulence within the compressor cylinders
- Heat transfer to and from contained gas via surrounding cylinder parts
- Suction and discharge valve leakage and frictional pressure losses
- Pulsations caused by intermittent flow

These physical realities collectively degrade an actual compressor cylinder's performance when compared to that of an ideal isentropic characterization. Of these losses, the most important are those that have a significant effect on the cylinder's volumetric efficiency and power requirements.

In terms of *English units*, the calculation of isentropic power required to compress one million standard cubic feet per day from a given suction pressure to a given discharge pressure is as follows:

$$\frac{IHP}{MMSCFD} = \frac{3.0303\, T_S\, Z_S\, P_B \left(r^{\gamma} - 1\right)}{T_B\, Z_B\, \gamma}$$

where:

γ	=	(k-1)/k
$\frac{IHP}{MMSCFD}$	=	Indicated horsepower per million standard feet per day at 100% mechanical efficiency.
k	=	Isentropic compression cycle exponent such that Pv^k = constant (where P = pressure, v = specific volume).
T_s	=	Absolute suction temperature
Z_s	=	Suction gas compressibility
P_B	=	Base pressure at which Std volume is referenced (psia)
T_B	=	Absolute temperature at which standard volume is referenced (from CUSTODY line, T_{ref} converted to absolute temperature.

Z_B	=	Gas compressibility at base conditions (very nearly, 1.0)
r	=	Compression ratio (absolute compressor discharge pressure/absolute compressor suction pressure.)

To convert this equation for use in practical compressor modeling applications, two correction factors are introduced as follows:

$$\frac{AIHP}{MMSCFD} = \frac{3.0303PCF}{ME} \frac{T_s \ Z_s \ P_B \ \left(r^{\gamma} - 1\right)}{T_B \ Z_B \ \gamma}$$

where:

$\frac{AIHP}{MMSCFD}$	=	Actual HP required per million standard cubic feet per day
ME	=	Unit mechanical efficiency (decimal)
PCF	=	Power correction factor
γ	=	(n-1)/n
n	=	Polytropic exponent such that Pv^n = constant

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Control Goals

You may set one of two independent goals for the compressor's operation:

- hydraulic goal
- unloading goal

Hydraulic Goal

(Idealized Control) Operate at the most limiting of the set points for DISCHARGE PRESSURE, SUCTION PRESSURE and FLOW. In other words, operate so that

$$PD \leq PDMAX$$
$$PU \geq PSMIN$$
$$Q \leq SPQ$$

and one of the conditions is an equality. The set point for which equality holds is called the active set point.

(Actuator Control) Operate at the speed determined by the output from an actuator:

$$S = SMAX \bullet X$$

where S is the compressor's speed, SMAX is maximum speed and X is the actuator output (a value between zero and one).

Note: Presence of an actuator for the compressor automatically turns on actuator Control and turns off Idealized Control, so that the discharge pressure, suction pressure and flow set points are ignored.

Unloading Goal

Operate inside the dead band around either the TORQUE SETPOINT or the SPEED SETPOINT, or operate at a FIXED UNLOADING STEP.

Accepting operation inside a dead band, rather than achieving the unloading goal exactly, is necessary because the unloading steps come in discrete jumps.

Constraints

The two control parameters, speed and unloading step, are adjusted automatically, subject to delays associated with dead band functionality, to move operation towards the control goals provided that this can be done without violating the following constraints:

- MINIMUM SPEED
- MAXIMUM SPEED
- MINIMUM TORQUE
- MAXIMUM TORQUE
- MAXIMUM COMPRESSION RATIO

Expressed more mathematically, the constraints are:

$$SMIN \leq S \leq SMAX$$
$$TQMIN \leq TQ \leq TQMAX$$
$$R \leq RMAX$$

SPS enforces the maximum compression ratio constraint by tripping the compressor (if tripping is enabled). Otherwise, the maximum compression ratio constraint has no effect.

Sometimes achieving both control goals would lead to violation of a speed or torque constraint. In this case, the hydraulic goal takes priority over the unloading goal, except in the case of a fixed unloading step. In other words, when necessary, an unloading goal targeting a torque or speed set point is overridden in favor of using the unloading step to satisfy the speed and torque constraints, while continuing with the hydraulic goal.

When there is no available unloading step for which the speed and torque constraints are satisfied and the hydraulic goal is met, a speed constraint will override the hydraulic goal and violation of a torque constraint for a period exceeding the delay time will cause the compressor to trip.

If there is no choice of the control parameters for which all the constraints are satisfied, including the bounds imposed by pressure and flow set points when idealized control is in effect, then the compressor is tripped, provided that tripping is enabled.

Note that maximum power is not listed as a constraint. PWMAX is only available while running at maximum speed. At lower speeds, the maximum available power drops in proportion to the reduction in speed, making maximum torque the effective maximum power constraint.

Starting and stopping

A compressor that is STOPPED may be started by issuing a START command in the INTRAN file or by poking a START command on a display. Similarly, a compressor that is RUNNING can be stopped by issuing a STOP command either from the INTRAN file or by poking a STOP command on a display.

An RC compressor starts by increasing speed linearly, at the rate going from zero to maximum speed in the start time. While starting the unloading step changes, with no delay, to keep operation as close as possible to the unloading goal. Speed continues to increase linearly until a hydraulic goal takes control. Tripping is disabled while the compressor's status is STARTING.

The point at which the compressor's status changes from STARTING to RUNNING is somewhat arbitrary. So that the compressor will not immediately trip unless the hydraulic goals force it, the status does not change until a hydraulic goal has taken control of speed and torque is above minimum torque.

An RC compressor stops by decreasing speed linearly, at the rate going from maximum speed to zero in the stop time. The unloading step is held fixed while the compressor is stopping.

Dead bands

When the status of a compressor is RUNNING, as long as compressor operation is inside the dead band around the unloading goal and satisfies the constraints on speed and torque, the unloading step is not changed. When compressor operation falls outside the dead band, accumulation of time outside the dead band, TODB, begins.

TODB is a signed number that is negative when compressor operation is below the dead band around the unloading goal and positive when above the dead band. It is reset to zero whenever operation moves back inside the dead band or crosses to the other side of the dead band. TODB accumulates at a rate for which time is penalized by distance outside the dead band, so TODB is likely to be different from change in simulation time.

When TODB exceeds the delay period, DEL, the unloading step is changed to the closest better one, unless that would cause cycling between two unloading steps on successive time steps. If no better unloading step is available, then TODB continues to accumulate up to a maximum of 5*DEL.

When no unloading step is between maximum and minimum torque, the unloading step closest to the acceptable torque range is selected, with no delay.

Example input

```
RC RECIP1 NODE1 NODE2 NODE1
+ MAXIMUM.POWER  4000.
+ MAXIMUM.SPEED  330.
+ MINIMUM.SPEED  200.
+ MAXIMUM.TORQUE  100.
+ MINIMUM.TORQUE  70.
+ UNLOADING.GOAL  TORQUE.SETPOINT
+ TORQUE.SETPOINT  95.
+ TORQUE.DEAD.BAND  2.
+ DISCHARGE.PRESSURE.SETPOINT  725.
+ SUCTION.PRESSURE.SETPOINT  550.
+ VOLUMETRIC.EFFICIENCY.CORRECTION  .042  0.0
+ START.TIME  2.
+ STOP.TIME  0.5
+ DELAY  3.
```

```
+ MECHANICAL.EFFICIENCY  0.930
+ HEAT.RATE  6794.
+ TABLE  CLEARANCE.RATIO       SWEPT.VOLUME
              0.321             15.263
              0.366
              0.409
              0.453
              0.504
              0.554
              0.604
              0.648
              0.688
              0.736
              0.786
              0.837
              0.886
              0.936
              0.987
              1.037
              1.081
              1.125
              1.169
+ TABLE    COMPRESSOR.EFFICIENCY    COMPRESSION.RATIO
                    0.908                   1.100
                    0.850                   1.150
                    0.828                   1.200
                    0.818                   1.250
                    0.815                   1.300
                    0.815                   1.350
                    0.819                   1.400
                    0.825                   1.450
                    0.832                   1.500
                    0.852                   1.600
                    0.879                   1.700
                    0.931                   1.800
```

Pump (P)

Peek and Poke Attributes

```
P NAME FROM TO SPEED HPWR [INERT] [*] [TORQ]
± RPM-BP HP-BP HD-BP Q-BP [TQ+BP] [TQ-BP] [DH+BP] [DH-BP]
[+ STAGES STG]
[+ STOPPING.FRICTION SFR]

/* Multiple head curves can be entered in increasing speed order.

[+ H-CURVE HD-MULT Q-MULT RPM]

/* Multiple power or efficiency curves can be entered in
/* increasing speed order. Only use power or efficiency, not both.

[+ P-CURVE HP-MULT Q-MULT RPM]
[+ E-CURVE E-MULT Q-MULT RPM]

/* Optional integrated devices

[+ SUCTION BLOCK ΔP Q BCRV TRT START STOP]
[+ SUCTION CHECK ΔP Q]
[+ DISCHARGE BLOCK ΔP Q BCRV TRT START STOP]
[+ DISCHARGE CHECK ΔP Q]
[+ BYPASS CHECK ΔP Q]
```

Field	Units Key	Description
NAME	n/a	Unique name of the pump.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
SPEED	SPEED	Maximum rated pump speed.
HPWR	POWER	Maximum rated driver power.
INERT	INERTIA	Effective moment of inertia (wet) for pump and driver combined. {1 mlbm-ft^2}
*	n/a	Placeholder for compatibility.
TORQ	n/a	Average starting torque as a percent of running torque for the motor. All preceding values are required. {160 percent}
±	n/a	ON/OFF key for the initial status of the pump: + for STARTING, - for STOPPED.
RPM-BP	SPEED	Pump speed at best efficiency point (BEP).
HP-BP	POWER	Pump brake horsepower based on water at BEP.
HD-BP	HEAD	Pump differential head at BEP.
Q-BP	PUMP.AFLOW	Pump flow rate at BEP.

Field	Units Key	Description
TQ+BP	TORQUE	Pump torque at BEP flow and 0 speed. {-0.4*Torque at BEP}
TQ-BP	TORQUE	Pump torque at BEP reverse flow and 0 speed. {0.85*Torque at BEP}
DH+BP	HEAD	Differential head at BEP flow and 0 speed. {-0.6*HD-BP}
DH-BP	HEAD	Differential head at BEP reverse flow and 0 speed. {-0.75*HD-BP}
STG	n/a	Number of stages for viscosity correction. {Smallest integer greater than HD-BP(ft)/600ft}
SFR	n/a	Friction term to reduce the pump spin-down time when the pump is stopping. See "Stopping friction" under "Take note" for more information. {0}
H-CURVE	n/a	Name of a data curve defining pump head as a function of flow.
HD-MULT	n/a	Multiplier for head data (usually 1). {1}
Q-MULT	n/a	Multiplier for flow data (usually 1). {1}
RPM	SPEED	Pump speed at which HEAD data applies.
P-CURVE	n/a	Name of a data curve defining power as a function of flow.
HP-MULT	n/a	Multiplier for power data (usually 1) {1}
Q-MULT	n/a	Multiplier for flow data (usually 1){1}
RPM	SPEED	Pump speed at which POWER data applies
E-CURVE	n/a	Name of a data curve defining efficiency as a function of flow. See "Description" under "Take note".
E-MULT	n/a	Multiplier for efficiency data (usually 1). {1}
Q-MULT	n/a	Multiplier for flow data (usually 1). {1}
RPM	SPEED	Pump speed at which EFFICIENCY data applies.
ΔP	ΔPRESSURE	Pressure drop across the fully open valve used to size the valve.
Q	FLOW	Flow through the fully open valve used to size the valve.
BCRV	n/a	Opening (and closing) curve to be used for the block valve. Use OPEN2 for a straight line open/close.
TRT	TIME	Travel time for the block valve.
START	TIME	Delay after the pump is started before the block valve starts to open.
STOP	TIME	Delay after the pump is stopped before the block valve starts to close.

Description

The Pump (P) models a fixed-speed or variable-speed centrifugal pump. A head as a function of flow data curve may be entered for each speed for which data is available. A power as a function of flow, or efficiency as a function of flow, data curve may also be entered for each speed for which data is available. Power and efficiency curves may not be mixed for the same pump. Power/efficiency curves determine hydraulic power. This does not include the inertial power expended to accelerate the pump. Driver power is the sum of hydraulic inertial power.

Note: Head curves may be entered without entering power/efficiency curves, and vice versa. Standard built-in curves are supplied automatically when no curves of a given type have been entered.

Optionally, the pump may have integrated devices (series block valves and check valves on the suction and discharge sides of the pump along with an optional bypass check valve).

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Speed/HP control

The pump may be modeled as a fixed speed pump without a control system or as a variable speed pump with a control system. An actuator may be used to control either speed or driver power. The choice of which is controlled is made in the definition of the controlling actuator. If speed is to be controlled, the speed at which the pump runs is SPEED multiplied by X, where X is the actuator output. (Actuator output is always a fraction between zero and one. See "Actuator (A)" on page 400.) If driver power is controlled, the power is HPWR multiplied by X, where again X is the output of the controlling actuator. In all cases, pump operation is limited by the maximum driver power constraint, HPWR, and the maximum speed constraint, SPEED. These constraints override the normal controls if they would be violated by some other means.

There are three modes for running a pump:

- The pump can be a fixed speed pump without a control system.
- The pump can have an actuator that drives the speed. In this case multiple pump curves may be entered. The pump speed is determined by multiplying the actuator output, X, by SPEED.
- The pump can have an actuator that drives the power. The pump power is determined by multiplying the actuator output, X, by HPWR.

START/STOP control

A pump that is STOPPED may be started by issuing a START command either in the INTRAN file or interactively. Similarly, a pump that is RUNNING can be stopped by issuing a STOP command either in the INTRAN file or interactively. The + or - symbol entered as the first character on the second line governs the initial status of the pump (+ for the simulation beginning with the pump STARTING, - for the simulation beginning with the pump OFF).

Stopping friction

A friction term can be entered to reduce the pump spin-down time when the pump is stopping. The torque lost to friction when stopping is:

$$T_{fric} = \text{stopping\_friction} \times \text{speed} \times \frac{\text{rated\_torque}}{\text{rated\_speed}}$$

Where T_{fric} is the torque corresponding to the rated speed (SPEED) and rated power (HPWR).

Fixed-speed pump

A fixed speed pump does not require an actuator. If no actuator is attached, pump startup at the beginning of the simulation is governed by the starting torque value entered under TORQ and the effective moment of inertia entered under INERT. A pump goes from starting to running when the pump reaches full speed or full driver power. Both maximum (rated) speed and maximum (rated) driver power are pokable.

Automatic pump trip

A pump with a controlling actuator may be made to trip off line (Power-down) by a signal from the control system. If the controlling actuator has an input signal that ultimately causes the actuator to output a value less than or equal to 0.005, then a RUNNING pump automatically trips off line. This mechanism is frequently used to drop a pump off line upon sensing an over-limit discharge or under-limit suction pressure.

Note: For pump startup when the pump has a controlling actuator, define the control system on startup in such a way that actuator output is > 0.005 at all times. This avoids the pump shutting down immediately.

A pump also trips when the temperature of the fluid reaches 350° F.

Note: When a pump is stopped, the PWR immediately goes to zero.

Best efficiency point (BEP)

BEP is the point on the pump performance map where a pump runs at its best efficiency. All of the input on the second format line is devoted to various data taken at BEP. The zero speed BEP data is used only during very low speed operation and to extend the user-defined performance curve.

Characteristic curves

Typically, the pump vendor only provides characteristic curves of developed head as a function of flow and power/efficiency as a function of flow at design speed for first quadrant operation. Where speed varies, *affinity laws* govern pump performance:

$$Q_1 / Q_2 = N_1 / N_2$$

$$H_1 / H_2 = (N_1 / N_2)^2$$

$$BHP_1 / BHP_2 = (N_1 / N_2)^3$$

where:

Q_1	=	Volumetric flow at RPM N_1 and Head H_1
Q_2	=	Volumetric flow at RPM N_2 and Head H_2
H_1	=	Developed head at RPM N_1 and flow Q_1
H_2	=	Developed head at RPM N_2 and flow Q_2
BHP_1	=	Brake horsepower at RPM N_1 and flow Q_1
BHP_2	=	Brake horsepower at RPM N_2 and flow Q_2

Note: The user-defined performance curves should be based on water. If the manufacturer's data is for a fluid other than water, you must convert the curves. If this is not done, the predicted running power is incorrect because the adjustment is based on the density of the fluid actually being pumped.

Viscosity correction

The Hydraulic Institute Standards viscosity correction procedures are applied when pumping viscous Newtonian fluids. Viscosity correction is made based on the pump's head at BEP divided by the number of stages. The default for the number of stages is chosen so that head per stage at BEP does not exceed the Hydraulic Institute Standards limit of 600 ft. However, if you enter a number of stages that makes head per stage greater than this limit, then your entry will be accepted with a warning.

Note: Performance curves and BEP data should be based on water and should represent overall, not per stage, performance. If the manufacturer's data is for a fluid other than water, you must convert the curves.

Note: The Hydraulic Institute Standards does not recommend extrapolation outside the range shown in their empirically-based charts. However, SPS extrapolates, if necessary, with a warning.

Multiple speed curves

If the pump manufacturer provides the necessary data and you want to model more accurately than can be represented by the affinity laws, a series of data curves (curve types HEAD, POWR, and/or EFFI) may be entered to define pump head and power at various reference speeds. In this case, as the speed changes, a modification of the affinity laws is used to interpolate between speed curves.

It is important to enter complete curves for the pump. The curves should start at shutoff head (zero flow) and should be extended to run-out flow (zero head). Doing so will aid in getting a proper curve fit as well as a smooth startup of the pump.

Entry of complete four-quadrant Knapp curves is especially important in cases where the pump is not protected from flow reversals and/or surges. In this case, the four-quadrant curve data is used to predict pump behavior under these conditions. If four-quadrant curves are not entered, the user-defined curves are extended into the missing regions using standard pump data built into SPS. The use of these standard curves is invoked automatically.

Integrated valves

The optional integrated valves are provided to simplify the specification of common pump piping arrangements and to simplify frequently used start and stop protocols. Any combination of the integrated valves may be used, from none to all.

Note: For multiple fluid simulations, the integrated valves are sized with the provided ΔP and Q using the first fluid entered on the equation of state.

If the pump is initialized as running, the series block valves are initialized as open; otherwise, they are initialized as closed.

Pump start and stop commands can be used to initiate a sequence of actions by using the delay parameters for the block valves. For example, if there is a discharge block valve and its start delay is 0.05 minutes, then 3 seconds after the pump is started the discharge block valve starts to open. The delays may be negative, in which case the valve operation precedes the pump status change. Even with negative delays the first action in the sequence takes place at the time the start or stop command is issued, not retroactively.

In defining the block and check valves, the ΔP and Q entries are used to determine the fully open valve coefficient. The various pressure drops in the connecting yard piping may be included with the pressure drop for the integrated valves.

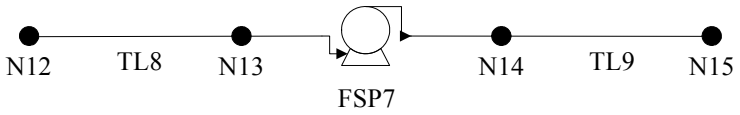
The optional valves do not have all of the flexibility that separate valves have, but they are simpler to enter and computationally more efficient than separately defined valves. Any integrated block valve uses the same curve for opening as for closing. The suction block valve may have a different opening/closing curve from the discharge block valve. The check valves open and close instantaneously to prevent back flow.

Note: The integrated valves cannot be used with either THERMAL or TRANSTHERMAL modes.

Control of flow and/or pressure set points

This function may be achieved by defining the appropriate control system with one or more sensors, any required relay(s), a controller, and an actuator. The sensor(s) and/or relay(s) provide the control signal input to the controller; the controller drives the actuator, which drives either the pump speed or power.

Example input



Example 1

A pump named FSP7 is connected to nodes named N13 and N14 and is installed between a pipe named TL8 and a pipe named TL9. The pump operates at a rated speed of 1200 RPM, with a driver rated horsepower of 2500 HP, and an effective moment of inertia of 1540 lbm-ft^2. The RPM at BEP is 1200, the HP at BEP is 1800, the pump head at BEP is 320 feet, and the flow at BEP is 655 AMB/D. The pump is bypassed with an integrated parallel check valve, and backflow is prevented by an integrated discharge check valve with an initial pressure drop of 0.25 and an initial flow of 655 MB/D through each check valve. Pump head and power curves are entered for a single speed (1200 RPM). Pump head as a function of flow is defined by a data curve named HFCV1. Pump power as a function of flow is defined by a data curve named PFCV1. The pump is initially stopped.

```
P FSP7 N13 N14 1200 2500 1.54
- 1200 1800 320 655
+ HFCV1 1 1 1200
+ PFCV1 1 1 1200
+ BYPASS CHECK 0.25 655
+ DISCHARGE CHECK 0.25 655
```

Note: Performance at BEP is based on water.

Example 2

The scenario is the same as Example 1, except the pump is initially starting and has no integrated valves. The pump uses standard pump performance curves built into SPS.

```
P FSP7 N13 N14 1200 2500 1.54
+ 1200 1800 320 655
```

Example 3

The following example shows input head as a function of flow and power as a function of flow data curves at three different speeds for a pump operating in the first quadrant. In this case, as the speed changes, TRANS uses a modification of the affinity laws to interpolate the data for the several speeds.

```
D HD_3550 HEAD 7                  /* HEAD VS. FLOW - 3550 RPM
900  870  850  780  690  610 0     /* HEAD - FT
  0 1150 2125 2950 3800 4500 5000  /* FLOW - GPM
D PW_3550 POWR 7                   /* POWER VS. FLOW - 3550 RPM
320  450  590  660  720  800 800   /* POWER - HP
  0 1150 2125 2950 3800 4500 5000  /* FLOW - GPM
D HD_2600 HEAD 7                   /* HEAD VS. FLOW - 2600 RPM
480 470  450  430  380  330 0      /* HEAD - FT
  0 860 1600 2130 2630 3200 3600   /* FLOW - GPM
D PW_2600 POWR 7                   /* POWER VS. FLOW - 2600 RPM
 80 150  225  260  280  310 310    /* POWER - HP
  0 860 1600 2130 2630 3200 3600   /* FLOW - GPM
D HD_2000 HEAD 7                   /* HEAD VS. FLOW - 2000 RPM
280 270  270  250  230  200 0      /* HEAD - FT
  0 640 1250 1750 2130 2575 2900   /* FLOW - GPM
D PW_2000 POWR 7                   /* POWER VS. FLOW - 2000 RPM
 25  40  50  100  125  140 140     /* POWER - HP
  0 640 1250 1750 2130 2575 2900   /* FLOW - GPM
```

Pump input must appear after any referenced data curves in the INPREP file. The following pump input incorporates multiple speed curves defined above.

```
P PUMP1 ND3  ND4  3550 700 .3 87.5 200
- 3550 700 730 3450    /* flow is in gpm
+ HD_2000 1 1 2000
+ PW_2000 1 1 2000
+ HD_2600 1 1 2600
+ PW_2600 1 1 2600
+ HD_3550 1 1 3550
+ PW_3550 1 1 3550
```

If the speed is 2250, the head and power values are determined by interpolation between the 2000 and 2600 RPM curves.

Sensor (S)

Peek and Poke Attributes

```
S NAME IN CURVE V1 V2 Z1 Z2

/* For values not available as unknowns
S NAME ±CNC TYPE CURVE V1 V2 Z1 Z2

/* For Named Fluid Sensors with the BWRS or SCL equations of state
S NAME ±CNC NFLD FLUIDNAME

/* For Named Fluid Sensors with the CNGA equation of state
S NAME ±CNC NFLD
```

Field	Units Key	Description
NAME	n/a	Unique name of the sensor.
IN	n/a	Source of the input signal. Use DEV:UNK to drive the sensor input with the unknown from device. The device can be any hydraulic or control device in the model; the unknown can be any unknown associated with the device. See "Input signals and set points" under "Take note" for more information. **Note:** Sensors can be used to sense values at nodes by using the DEV:UNK style. For example, to sense the pressure at node NODEA, use NODEA:P in the "IN" field.
±CNC	n/a	Name of the element or input reference where the sensor is connected (preceded by - for upstream side, + for downstream side). This cannot be a named node.
TYPE	n/a	Sensor type. See "±CNC style" under "Take note".
CURVE	n/a	Name of a data curve describing the output signal (V_n) of the sensor as a function of the sensed value (Z_n). Use V(Z)2 for a straight line mapping.
V_n		Minimum/maximum values for the sensor output.
Z_n		Minimum/maximum values for the sensor input.
FLUIDNAME	n/a	Name of the fluid or component whose mass fraction is to be sensed.

Description

The sensor (S) is used to model a sensing device (flow meter, transducer, gauge, etc.) that measures a given variable at a given location and produces output signals for use by interpretative devices such as controllers, relays, or actuators. Any number of sensors is allowed.

Sensors can measure and report to the control system a number of physical quantities.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Input signals and set points

Review the OUTPRP report to find the available unknowns in your model. Typically, the following unknowns are available:

Pipe	P-, P+, Q-, Q+
Header	P-, P+, Q
Valve	P-, P+, Q
Pump	P-, P+, Q, RPM, PWR, TQ
Compressor	P-, P+, Q, RPM, PWR, FUEL
Control elements	OUT

The P- and P+ references can be used even though they do not appear in the unknown summary because they are aliases for the node pressures.

Depending on the thermal mode selected, temperature unknowns may be available as well.

±CNC style

When using the ±CNC style, the measurement is made at the +end or -end of an element, not at a node. The measured quantity, Z, is determined by the TYPE designated, as shown in the following table:

Type	Quantity measured
DENS	The density at pipeline conditions in DENSITY units.
FLOW	Volume flow at custody transfer conditions.
IREF	The output of the named input reference.
MFLO	The mass rate of flow in MFLOW units.
NFLD	The concentration of the named fluid or gas gravity.
PFLO	The actual flow rate at pipeline conditions.
POWR	The pump/compressor driver power in POWER units.
PRES	Pressure, in PRESSURE units.
RPM	The pump/compressor speed in SPEED units.
STEM	The block, check or control valve stem position, fraction open.
TEMP	Temperature, in TEMPERATURE units.
UNK	Device:UNK.
VISC	The viscosity at pipeline conditions, in VISCOSITY units.

Subtype attribute

The :STYPattribute distinguishes the sensor type in a Show window, report, or text display. The :STYP peek attribute for a specific sensor is the same as what you entered for TYPE, except when using the `device:unk` style. In this case, the :STYP is the name of the unknown being sensed. For example, the :STYP peek attribute for a density sensor is DENS. Also, this peek attribute provides a way to use the TRENDLIST and PEEKLIST commands in TRANS to group sensors for trending and/or summing up variables by subtype.

Sensor map

The sensor can optionally map (or scale) the value sensed using an arbitrary data curve. Enter the name of the data curve in the CURVE field. Enter the minimum and maximum sensor outputs as V_1 and V_2. Enter the minimum and maximum sensor inputs as Z_1 and Z_2. For a straight line mapping, enter CURVE as V(Z)2.

The sensor output will be:

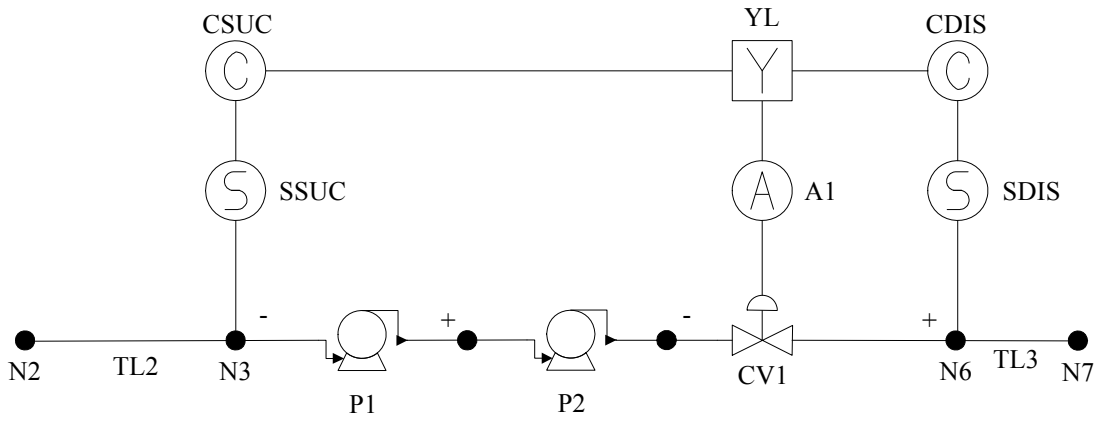
$$\text{output} = \begin{cases} V_1 & \text{if} \quad \text{input} < Z_1 \\ \text{curve(input)} & \text{if} \quad Z_1 \le \text{input} \le Z_2 \\ V_2 & \text{if} \quad \text{input} > Z_2 \end{cases}$$

Named fluid sensor

The named fluid sensor senses the concentration of a named fluid (SCL) or component (BWRS) in mass fraction. The named fluid sensor senses the gas gravity if used with the CNGA equation of state.

Example input

The input examples for the sensor, controller, actuator and LO-SELECT relay refer to the following diagram:



Example 1

A sensor named SSUC is monitoring pressure at the upstream side of a unit named P1 for input to a control loop to control that pressure to a desired set point. Sensor SSUC has a linear response described by output as a function of corresponding pressure sensed values. The sensor has a pressure range from 0-1000 psig:

```
S SSUC -P1 PRES V(Z)2 0 1000 0 1000
```

Example 2

A sensor named SDIS is monitoring pressure at the downstream side of a control valve named CV1. The sensor has a curved response function described by a data curve named VZ023 with the first table pair corresponding to (1,0) and the last table pair corresponding to (2000,500):

```
S SDIS +CV1 PRES VZ023 1 2000 0 500
```

Input reference (I) (INPREP)

Peek and Poke Attributes

```
I NAME CURVE FUNC1 FUNC2 t1 t2 [SC] [ST]
/* --or--
I NAME CONSTANT VALUE
[+ DLY [RESET]]
```

Field	Units Key	Description
NAME	n/a	Unique name of the input reference.
CURVE	n/a	Name of a data curve F(t) used to describe input reference output signal as a function of time since activation. If F is a straight line, use F(T)2 for the CURVE name.
$FUNC_n$	n/a	Function end-point values for adjusting the data curve function data.
t_n	TIME	Beginning and ending reference times for adjusting the data curve function data.
SC	n/a	Scale factor—a factor used to adjust the IR time scale relative to the simulation time scale. {1}
ST	TIME	Start time—simulation time (minutes) used as a time-shift in evaluating the input reference output as a function of simulated time. {1e20}
VALUE	n/a	Input constant.
DLY	TIME	Minutes delay for initiating an event-programmed response. {0}
RESET	n/a	Enables reset during event-programmed response.

Description

The input reference (I) provides a *sensor* with a time-varying input signal that can be programmed. The input reference is attached to a sensor by naming it as the connecting input in the second field of the sensor (S). The signal generation is initiated either by assigning an appropriate value for START or by directives in the INTRAN file or in the interactive mode.

The input reference signal function may also be used to enter constants that can be entered and/or changed from the INTRAN file or from pokes on a display.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Event-programmed

The input reference can be programmed to monitor output limits of some control element. (See “Input reference (I) (INTRAN)” on page 461). Once the control crosses the limit, the delay period begins. If the monitored value remains

past the limit through the delay period, the signal generation is activated by replacing the value of START below with the current time.

If the string RESET appears on the continuation line, it changes the response during an event-activated signal generation. If the triggering condition becomes false during the course of the signal generation, the input reference immediately resets to $F(t_1)$.

Time-programmed

The input reference can be manually activated by entering the activation time (START) in the input format. The time entered may be changed later by an entry on the input reference in the INTRAN file (see "Input reference (I) (INTRAN)" on page 461) or by poking it on the display.

Function output

During the transient simulation, the input reference has the following output, as a function of simulation time (T):

$$\tau = \text{SCALE} \bullet (\text{T} - \text{START})$$

where SCALE and START are defined in the input format and may be modified interactively. In addition, START may be replaced by current time to begin signal generation if the input reference is event-programmed. The output signal of the input reference is

$$\text{output} = \begin{cases} \text{FUNC}_1 & \text{if } \tau < t_1 \\ \text{curve(input)} & \text{if } t_1 \le \tau \le t_2 \\ \text{FUNC}_2 & \text{if } \tau > t_2 \end{cases}$$

For a straight line mapping, enter CURVE as F(T)2.

Example input

An input reference named IR050 provides an input signal to a sensor named PS050. The data may be entered in several ways, depending on how the input reference is programmed:

Example 1

The input reference is time-programmed to begin at five minutes into the simulation, with a data curve named IRCRV to describe the input reference signal over time.

```
I IR050 IRCRV 0. 0. 0. 0. 1. 5.0
```

Example 2

The scenario is the same as Example 1, except for a linear input reference function (input signal ramped from 0.01 to 1.0 over 2 minutes).

```
I IR050 F(T)2 0.01 1.0 0. 2.0 1. 5.0
```

Example 3

The scenario is the same as Example 2, but the input signal is ramped from 0 to 1 over 1 minute with a delay of 0.5 minutes to initiate an event-programmed response. RESET is enabled.

```
I IR050 F(T)2 0 1  0  1
+ .5 RESET
```

P-I-D Controller (C)

Peek and Poke Attributes

```
C NAME IN SP KC TD TI VS NC
[+ I = INTEG
```

Field	Units Key	Description
NAME	n/a	Unique name of the controller.
IN	n/a	Source of the input signal. Use DEV:UNK to drive the controller input with the unknown from device. The device can be any hydraulic or control device in the model; the unknown can be any unknown associated with the device. See "Input signals and set points" under "Take note" for more information.
SP	n/a	Source of the set point signal. Uses the same conventions as IN, or a constant.
KC		Gain. May be either a number for a constant gain or the name of a data curve for the gain as a function of the error signal.
TD	SIG.TIME	Derivative time. {0}
TI	SIG.TIME	Reset (integral) time, T_I. {Entry of 0 gives $T_I = 10^{20}$}
VS	n/a	Output bias. {0}
NC	n/a	Normalization constant, NC.
INTEG	n/a	Initial value for the integral term used only with an active initial state.

Description

The P-I-D controller (C) models a standard P-I-D control element.

A controller is used to interpret input signals from other elements (sensors, controllers, relays) by comparing them to a set point (which can be either a constant value, or the output from some other control element). Based on the difference (or error) between the input signal and the set point, the controller produces an output signal of varying magnitude (voltage) to drive other elements.

The controller closely mimics the operation of a standard P-I-D controller, down to the exact settings of its tuning knobs.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Input signals and set points

Review the OUTPRP report to find the available unknowns in your model. Typically, the following unknowns are available:

Pipe	P-, P+, DP, Q-, Q+
Header	P-, P+, DP, Q
Valve	P-, P+, DP, Q
Pump	P-, P+, DP, Q, RPM, PWR, TQ
Compressor	P-, P+, DP, Q, RPM, PWR, FUE
Control elements	OUT

The P- and P+ references can be used even though they do not appear in the unknown summary because they are aliases for the node pressures. Similarly, DEV:DP (= DEV:P+ - DEV:P-) can be used even though it does not appear in the unknown summary.

Depending on the thermal mode selected, temperature unknowns may be available as well.

Error signal

The normalized difference between the input control signal C(t) and the set point signal S(t) is called the error signal:

$$E(t) = \frac{C(t) - S(t)}{NC}$$

where:

NC is a normalization constant used to adjust the error signal (usually so that -1 < E(t) < 1), and S(t) is the value of SP.

Output voltage

P-I-D output signal voltage V(t) is calculated by:

$$V(t) = K_C \left[E(t) + T_D E'(t) + \left(\frac{1}{T_I} \right) \int_0^t E(t)\,dt \right] + V_S$$

where:

K_c	=	Controller gain
T_I	=	Reset time
T_D	=	Derivative time
Vs	=	Manually-adjusted bias
$E'(t)$	=	dE/dt
$T_D E'(t)$	=	derivative term
$\left(\frac{1}{T_I} \right) \int_0^t E(t)\,d\tau$	=	integral term

Tuning knobs

Use the following data entries to mimic knob settings on an actual P-I-D controller:

Knob	Data Entry
Proportional band setting	Constant gain, K_c
Reset adjustment	T_I
Derivative adjustment	T_D
Reverse switch	NC > 0 for increase; NC < 0 for decrease
Set point	SP
Output limits	Controller and integral limits {LO=0; HI=1}

P-I-D variations

For P only, $T_D = 0$ and $T_I = \infty$. For non-linear proportional controllers, enter a data curve describing K_c as a function of error signal. For a P-I controller, $T_D = 0$. For a P-D controller, $T_I = \infty$.

Anti-reset windup

Windup can occur from a sustained error signal for a controller in a system that cannot attain its set point. The integral can continue to increase (wind up) even though the controller has reached its limit. This large integral causes the controller to be unresponsive even if the error subsequently changes sign (indicating that the system can now make its set point). This period of unresponsiveness is occasioned by having to wind-down the integral after the error changes sign.

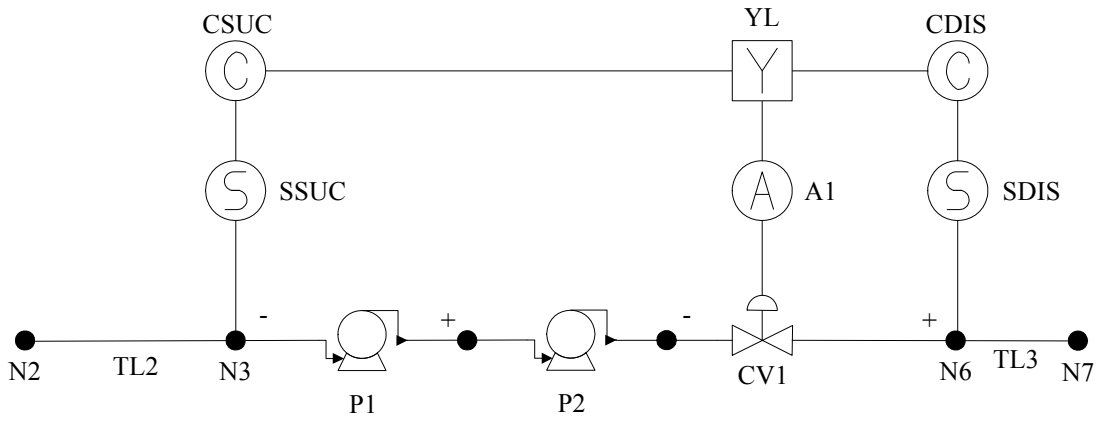
The anti-reset windup feature helps avoid this problem by freezing the integral value when the controller is at the full limit (upper or lower) of its output.

To activate the anti-reset windup feature, enter a negative value for T_I. The absolute value of the entry is used in the V(t) equation. In control loops involving relays, the use of a feedback relay can help with a related problem called *bump*. Control bump involves wind up in a controller that is unselected by a select relay. A HI/LO select relay has automatic feedback for controllers directly connected to it. (See "HI/LO Select Relay (Y HI/LO)" on page 403.)

Direct or reverse acting

A direct acting controller tries to maintain the output at or above the set point (positive normalization constant). A reverse acting controller tries to maintain the output at or below the set point (negative normalization constant).

Example input



Example 1

A controller named CSUC is monitoring the output of a sensor named SSUC and comparing it to a fixed set point of 50 psig. The gain is constant at 1.0, the derivative time is 0.30 minutes, the reset time is 1 minute, the output bias is 0 and the normalization constant is 100:

```
C CSUC SSUC:OUT 50 1.0 0.30 1.0 0 100
```

Example 2

A controller named CDIS is monitoring the output of a sensor named SDIS and comparing it to a fixed set point of 900 psig. The gain is constant at 1.0, the derivative time is 0.30 minutes, the reset time is 1 minute, the output bias is 0 and the normalization constant is -1000. (The negative value for the normalization constant makes this a reverse-acting controller):

```
C CDIS SDIS:OUT 900 1.0 0.30 1.0 0 -1000
```

Actuator (A)

Peek and Poke Attributes

```
/* Second Order type:
A NAME LNAME PEQUIP CURVE V0 V1 A1 A2 A3

/* Rate Constant type:
A NAME LNAME PEQUIP CURVE V0 V1 R

/* Travel Time type:
A NAME LNAME PEQUIP CURVE V0 V1
+ TRAVEL.TIME Topen [Tclose]
```

Field	Units Key	Description
NAME	n/a	Unique name of the actuator.
LNAME	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively).
PEQUIP	n/a	Name of the equipment controlled by the actuator, preceded by H (pump or compressor HP), R (pump or compressor RPM), Q (external supply/delivery flow), P (external supply/delivery pressure), or V (control valve), or A (KV compressor angle) to describe the controlled variable.
CURVE	n/a	Name of a data curve defining actuator position X as a function of input signal V. X must always lie between 0 and 1, and it is recommended that the slope of X(V) not change sign. A default data curve can be used by using X(V)2 to yield the straight line from (0,V_0) to (1,V_1).
V_n	n/a	Input limits where X(V_0) = 0 and X(V_1) = 1.
A_1, A_2, A_3	n/a	Coefficients for the second order function if they are to be entered directly. You can obtain direct response by setting A_2 and A_3 to zero.
R	n/a	Rate constant for the actuator, approximately the reciprocal time for full scale travel following a full scale step change in input, in reciprocal minutes.
T_{open}	n/a	When opening, the actuator position follows the scaled input signal (no second-order smoothing), but does not open at a rate faster than indicated by T_{open}.
T_{close}	n/a	When opening, the actuator position follows the scaled input signal (no second-order smoothing), but does not open at a rate faster than indicated by T_{close}.

Description

The actuator (A) models an actuator. The actuator is the part of a control loop that attaches directly to the process equipment and adjusts equipment settings in response to time-varying input signals from a controller, sensor, or relay. Actuators are used *only* to control pump/compressor speed or power, external flow or pressure, or control valve openings.

Actuators are modeled by describing the varying relation between actuator position, X (expressed as a fraction of total travel), input signal, V (expressed as a voltage), and rate of travel, R (expressed as a fraction of travel per minute).

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Second-order behavior

The relation between the position (y) and the input signal (V) is:

$$\frac{A_3 d^2 y}{dt^2} + \frac{A_2 dy}{dt} + A_1 \left(y - X(V)\right) = 0$$

where t is time in minutes, A_1, A_2, and A_3 are rate constants, entered as data, and X(V) is the desired actuator position for input V.

If the rate constant, R, is entered in reciprocal minutes, the values of A_1 and A_2 are computed as

$$A_1 = 20.25\ R^2 \quad A_2 = 2\sqrt{A_1}$$

while A_3 becomes 1.

Time travel behavior

The actuator goal, G, is given by:

$$G = \text{bound}\ (0,\ X(V),\ 1)$$

where:

V	=	Input signal
X	=	CURVE scaled to go from [0,1] as input goes from [v0,v1]
bound(a,b,c)	=	MAX(a, MIN(b,c))

"G" is the desired actuator position, without regard to open/close time.

The actuator position, P, is given by:

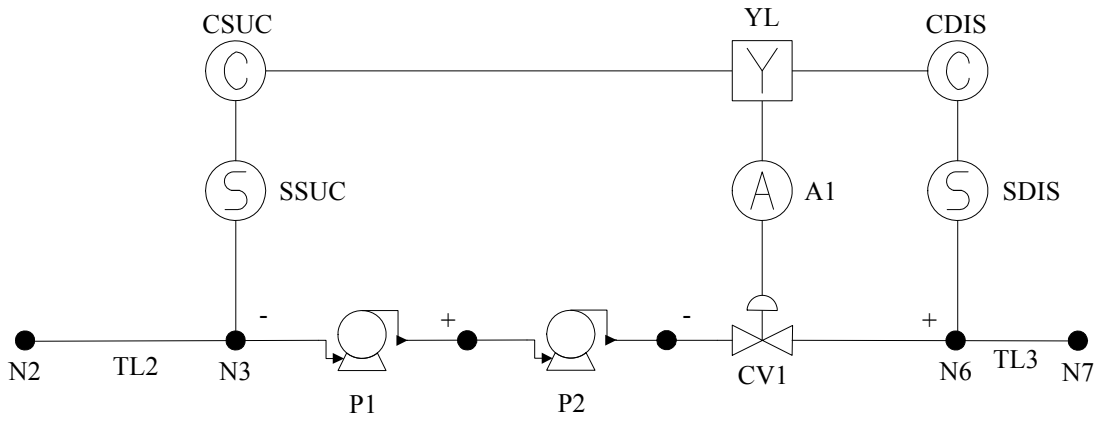
$$P = \text{bound}\ (y_{lower},\ G,\ y_{upper})$$

where:

y_{lower}	=	Position lower bound (= previous position - position corresponding to T_{close})
y_{upper}	=	Position upper bound (= previous position + position corresponding to T_{open})

"P" is the goal, G, with the open/close rates applied.

Example input



Example 1 – Second Order type

An actuator named A1 accepts its input signal from a Lo-Select relay named YL and controls a control valve named CV1. The actuator position as a function of the input signal is defined by a data curve named XV023. The actuator constants are A1 = 60.75, A2 = 15.6, and A3 = 0.

```
A A1 Y YL V CV1 XV023 0 1 60.75 15.6  0
```

Example 2 – Rate Constant type

An actuator named A1 accepts its input signal from a Lo-Select relay named YL and controls a control valve named CV1. The actuator position as a function of the input signal is linear from 0 to 1. The rate constant is 15 (the actuator takes 4 seconds to travel full scale).

```
A A1 Y YL V CV1 X(V)2 0 1 15
```

HI/LO Select Relay (Y HI/LO)

Peek and Poke Attributes

```
Y NAME TYPE IN1 IN2...
+ ...  INi ...  /* Any number of inputs can be on the various lines and
+ ... INn       /* continued on as many lines as necessary.
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
TYPE	n/a	HI or LO.
IN_n	n/a	Name of two to 20 sensors, relays, or controllers that supply the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). If input is to be a constant, enter the constant value, preceded by V. The constants, if any, should be entered last.

Description

The HI/LO-select relay (Y) is used to switch control from one signal to another depending upon which of several signals is largest (or smallest). It accepts two to 20 input signals, and generates an output signal equal to the highest (or lowest) value. It is typically used to control one quantity subject to a constraint on another quantity.

Take note

Common syntax for equipment

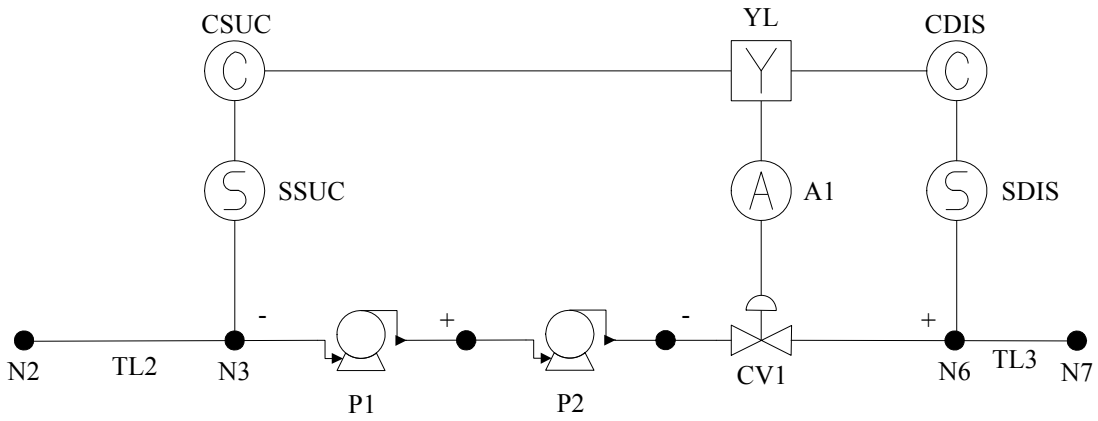
Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Bumpless control

The HI/LO-select relay has a built in feedback relay. This results in bumpless control if the relay switches between controllers. The non-controlling controller’s integral term is set equal to the controlling controller’s integral term.

Example input

A Lo-Select relay named YL is monitoring input signals from controllers named CSUC and CDIS.




```
Y YL LO-SELECT C CSUC C CDIS
```

Derivative Relay (Y DERIV)

Peek and Poke Attributes

```
Y NAME DERIV IN RATE VS
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). If input is to be a constant, enter the constant value.
RATE	1/TIME	The name of a sensor, controller, or relay, preceded by its keyletters (S, C, or Y, respectively), for which the output is the maximum absolute value of the rate of change of the output of the derivative relay, or a constant value.
V_S	n/a	Constant output bias.

Description

The derivative relay (Y) is used to transmit a signal unchanged unless the rate of change exceeds a limit RATE. The relay specifies a voltage signal originating from a named device for the maximum rate of change of the output signal. The (unconstrained) output signal V(t) of the derivative relay is:

$$V(t) = C(t) + V_S$$

but the actual output has V(t) constrained so that

$$\left|V(t)'\right| \leq \mathrm{RATE}$$

where:

$C(t)$ is the input signal.

Note: The DERIV relay cannot receive input signals from another DERIV relay.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

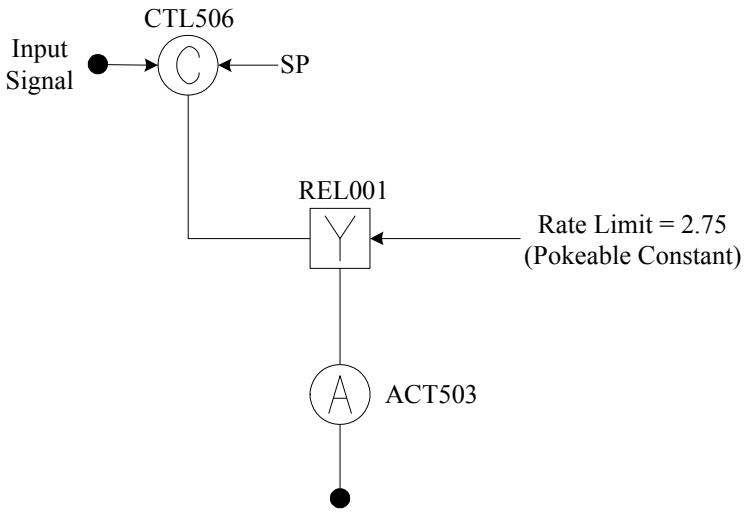
Example input

Example 1

A derivative relay named RLYDRV is receiving a signal from a sensor, SENS1. The rate-signal element is a sensor named SR1. The constant output bias is 10.

```
Y RLYDRV DERIV S SENS1 S SR1 10.
```

Example 2



A derivative relay named REL001 is interposed between a controller named CTL506 and an actuator named ACT503. The relay is programmed to prevent the control signal from increasing or decreasing at a rate faster than 2.75 volts per minute. The output bias is .001:

```
Y REL001 DERIV C CTL506 2.75 0.001
```

Multiply Relay (Y MULTIPLY)

Peek and Poke Attributes

```
Y NAME MULTIPLY IN1 M1 through IN3 M3
[+ IN4 M4 through IN7 M7 ]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN_n	n/a	Name of one or more sensors, relays, or controllers that supply the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). Instead of a name, you may enter BIAS as the first input. In this case, M_1 is interpreted as the constant (pokable) bias.
M_n	n/a	A corresponding number of multipliers. M may be a constant value, or the name of a sensor, relay, or controller preceded by S, Y, or C, respectively. Up to three input/multiplier pairs may be entered on the first line. Up to four more may be entered on the second line.

Description

The multiply relay (Y) combines up to seven input signals, each multiplied by some multiplier (either a constant or a signal from a sensor, relay, or controller), to produce an output signal.

Take note

Common syntax for equipment

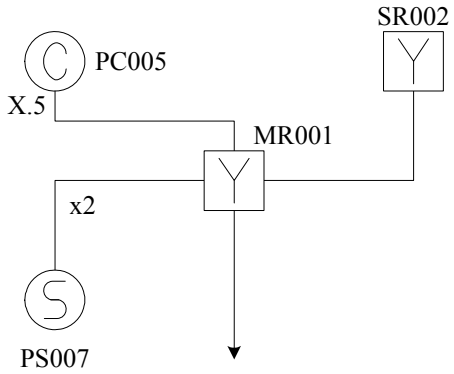
Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Multiplication

The multiply relay uses the following formula to determine its output signal, V(t):

$$V(t) = IN_1 \bullet M_1 + IN_2 \bullet M_2 + ... + IN_n \bullet M_n$$

Example input



A multiply relay named MR001 is monitoring the output signals from a sensor named PS007, a controller named PC005 and a relay named SR002. MR001 is determining its output signal by adding twice the value of the signal from PS007 to half the value of the signal from PC005 to the signal from SR002.

```
Y MR001 MULTIPLY   S PS007 2.0    C PC005 0.5     Y SR002 1
```

Integrator Relay (Y INTEG)

Peek and Poke Attributes

```
Y NAME INTEG IN [TIMESCALE] [INITIAL]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). No constant may be entered here.
TIMESCALE	n/a	Number of integrator time units in one minute. {0.00069444}
INITIAL	n/a	Initial value for the integral. {0}

Description

The integrator relay (Y) is used to integrate a signal with respect to time.

Any control signal can be integrated with respect to time. The time scale for this integration can be controlled by the TIMESCALE input. The integrator time units are specified by entering the number of integrator time units that correspond to 1 minute. For example, if the integration is to be with respect to time in seconds, enter TIMESCALE as 60 (60 sec/min).

Often it is desired to integrate a flow. If the IN signal is derived from a flow rate measured by a FLOW type sensor, then it is measured in units specified on the FLOW record. The default units for the FLOW record are thousands of barrels per day. Thus the proper choice for TIMESCALE using the default FLOW units is 0.000694 (MB/D * 1 DAY/1440 MIN ==> 1/1440 = 0.000694). The time is converted to minutes. The integrator output is then in thousands of barrels.

If the units for FLOW are cubic feet/second, and IN is a signal derived from a FLOW sensor its units are cubic feet per second. In this case TIMESCALE is entered as 60 (ft^3/sec * 60 sec/min). The time is converted to minutes. The integrator output is then in cubic feet. If instead, the TIMESCALE was entered as 0.060, (i.e., 60/1000), the integrator output is then in thousands of cubic feet.

A signal from other elements in the control system might not have units with a factor of reciprocal time. In this case, the output of the integrator is in the units of the input signal multiplied by days if the TIMESCALE default value is used, or by minutes, for example, if TIMESCALE is entered as 1. For example, if the input signal is temperature in °F. and the TIMESCALE is entered as 1, the output is °F · minutes.

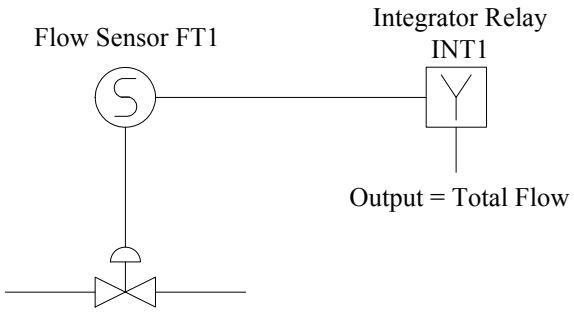
The initial value of the integral is taken to be the entered value of INITIAL. The value of the integral is also changeable by a poke entry in the INTRAN file.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Example input



An integrator relay named INT1 is monitoring the flow reported by a flow sensor named FT1. The number of integrator time units in a one-minute span has been set equal to one. The initial value of the integral that represents the flow at the beginning of the simulation defaults to zero.

```
Y INT1 INTEG S FT1  1.
```

Feedback Relay (Y FEEDBACK)

Peek and Poke Attributes

```
Y NAME FEEDBACK IN VL VH
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). No constant may be entered here.
VL	n/a	Name and prefix (S, Y, or C for sensor, relay or controller, respectively) of the element limiting the low-side output of attached controllers). May be zero (as placeholder), when VH is derived from a HI-select relay.
VH	n/a	Name and prefix (S, Y, or C for sensor, relay or controller, respectively) of the element limiting the high-side output of attached controllers. May be zero (as a placeholder) or omitted, when VL is derived from a LO-select relay.

Description

The feedback relay (Y) is the control loop element used to select, switch or convert signals traveling from one control instrument to another. The feedback relay is used to provide control stability to multiple controllers by preventing windup in a non-selected controller.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Multiple controllers

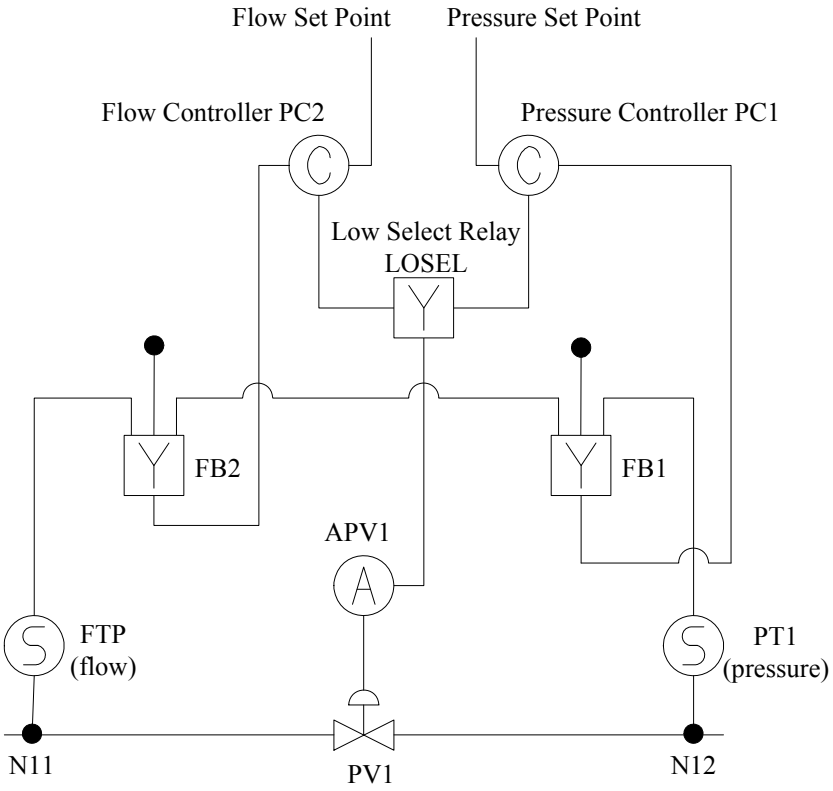
Where multiple controllers are attached to a single actuator using a HI-select or LO-select relay, the feedback relay can provide stability by causing the integral of the non-selected controller to track the selected controller, so that transitions from one controller to the other is smoother. Such systems in the field are frequently called *bumpless* control systems.

If a HI-select relay is used, VL should be fed a signal that is higher than any signal that is transmitted by VH. Conversely, if a LO-select relay is used, VH should be fed a signal that is lower than any signal that is transmitted by VL.

Feedback relay is usually not needed

The anti-windup of non-selected controllers is automatic, provided the controllers are directly connected to the Select relay.

Example input



Design a feedback relay to avoid the bump associated with the windup of the out-of-control controller integral used with a Low Select relay to provide flow control with a discharge pressure override.

A sensor named PT1 senses the pressure on the discharge side of a control valve named PV1, and its output is the control signal to a controller named PC1. The set point of PC1 is the pressure override control value. This means that the controller is to act to prevent the pressure downstream of PV1 exceeding the set point of PC1.

A sensor named FTP senses the flow through PV1, and its output is the control signal to a controller named PC2. The set point of PC2 is the desired flow through the valve PV1. An actuator named APV1 receives its signal from PC1 and PC2 through a Low Select relay named LOSEL and controls PV1.

A sensor named SZER senses the pressure from an input reference named ZERO. A feedback relay named FB1 receives its input signal from sensor PT1. A feedback relay named FB2 receives its input signal from sensor FTP:

```
V PV1 +PIPE2 -PIPE3 EQPC 0. 80.
S PT1 +PV1 PRES V(Z)2 0. 750. 0. 750.
S FTP -PV1 FLOW V(Z)2 0. 10000. 0. 10000.
I ZERO  F(T)2 0. 0. 0. 1.
S SZER +ZERO PRES V(Z)2 0. 1. 0. 1.
Y FB1  FEEDBACK S PT1 Y LOSEL S SZER
Y FB2  FEEDBACK S FTP Y LOSEL S SZER
Y LOSEL LO-SELECT C PC1 C PC2
A APV1  Y LOSEL V PV1 X(V)2 0. 1. V2.5
C PC1  FB1:OUT 450. 1.5 0. -0.4 0. -1000.
C PC2  FB2:OUT 1250. .4 0. -0.4 0. 100.
```

Switch Relay (Y SWITCH)

Peek and Poke Attributes

```
Y NAME SWITCH IN1 IN2 SS C1 [C2] [DLY]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN_n	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). If the input is a constant, enter the value.
SS	n/a	Set point signal (same rules as for IN_n).
C1	n/a	Value or source of limit signals (same rules as for IN_n).
C2	n/a	Value or source of limit signals (same rules as for IN_n).
DLY	TIME	A time delay for the changing of the switch connection. {0}

Description

The switch relay (Y) compares a set point value (or a signal from a named source) with two limit values (either or both of which may be signals from a named source). From the results of the comparison it selects as output one of two input signals. Each input signal may be chosen to be either a named source or a constant.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see "POKE and RAMP" on page 423 and "X and Y coordinates" on page 422.

Signal selection

The switch relay uses the following relationships to determine its output signal.

When $C_1 < C_2$:

$$\text{output} = \begin{cases} in_1 \ \text{ if } \ ss < c_1 \\ in_1 \ \text{ if } \ c_1 \le ss \le c_2 \ \text{ and previous output was } \ in_1 \\ in_2 \ \text{ if } \ c_1 \le ss \le c_2 \ \text{ and previous output was } \ in_2 \\ in_2 \ \text{ if } \ ss > c_2 \end{cases}$$

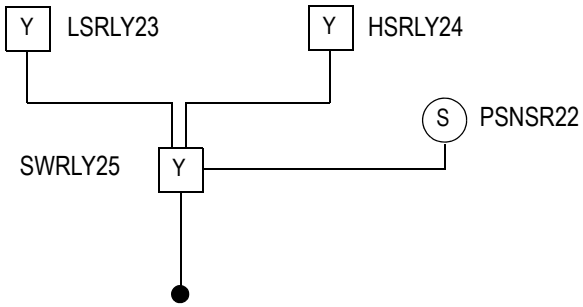
If $C_1 > C_2$, or C_2 is not entered, the program redefines $C_2 = C_1$ for the foregoing choice.

Delay

If the DLY entry is included with a positive number for DLY, this value is the delay time in minutes for changing the connection of the switch. The delay time period begins at the time the selection rules first indicate a change. If the rules continuously call for the change to the end of the delay period, the change is made. Otherwise, the switch connection

remains unchanged. Any new change that might be indicated could occur only upon initiating and completing a new DELAY period.

Example input



A switch relay named SWRLY25 is monitoring the output signals from relays named LSRLY23 and HSRLY24. The set point signal is derived from a sensor named PSNSR22. The input signal limits are defined as 45 for the minimum and 320 for the maximum.

```
Y SWRLY25 SWITCH  Y LSRLY23  Y HSRLY24  S PSNSR22  45  320
```

Noise Relay (Y NOISE)

Peek and Poke Attributes

```
Y NAME NOISE IN [SC] [PER] [PEAK] [RELATIVE]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). No constant may be entered.
SC	n/a	Scale of the noise used to perturb the input signal. {0} (The range of the scale is from 0 to 50% of the entered signal.)
PER	SIG.TIME	Time interval (minutes) between changing the value of the noise. Its default is {0}, which means the noise changes with each time step.
PEAK	n/a	If present, the noise, which lies in the interval [-SC, +SC], is peaked 0. This means the probability of a noise value of X in the interval decreases linearly to zero as |X| approaches SC. If not present, noise is generated uniformly in the interval [-SC, +SC]. This means that each value of the noise in the interval is equally likely.
RELATIVE	n/a	If omitted, the scale is interpreted as absolute. If included, the scale is interpreted as relative to the signal size.

Description

The noise relay (Y) is used to introduce a perturbing noise into a control signal. Noise may be used to study leak detection feasibility and to study control loop stability.

Any control signal can be perturbed by a noise relay. If the optional flag, RELATIVE, does not appear, the output signal is the input signal plus X, where X is the noise defined above. X is updated each simulated time interval of length PERIOD.

The random variable X is given by

$$X = \begin{cases} \text{uniform}(-sc, +sc) & \text{without PEAK flag} \\ \text{peak}(-sc, +sc) & \text{with PEAK flag} \end{cases}$$

The output V of the noise relay is given by

$$V = \begin{cases} IN + X & \text{without RELATIVE flag} \\ IN \cdot (1 + X) & \text{with RELATIVE flag} \end{cases}$$

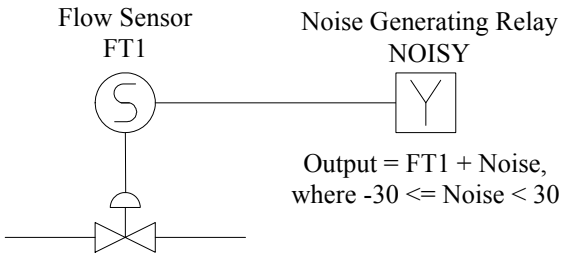
If the input IN is a polling sensor and PER for the noise relay is the same as that for the sensor, the random variable X is updated each time the sensor output value changes.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Example input



A noise relay named NOISY1 is monitoring the flow reported by a flow sensor named FT1. It is to report the value of the measured flow perturbed by noise that lies in the interval between -30 and +30. The value of the noise is changed every 45 seconds.

```
Y NOISY1 NOISE S FT1 60 0.75
```

Time-Averaging Relay (Y AVERAGE)

Peek and Poke Attributes

```
Y NAME AVERAGE IN [PER]
```

Field	Units Key	Description
NAME	n/a	Unique name of the relay.
IN	n/a	Name of the sensor, relay, or controller that supplies the input signal, preceded by the corresponding keyletter (S, Y, or C, respectively). (No constant may be entered here).
PER	TIME	Time period in minutes on which the averaging is to occur. {0}

Description

The time-averaging relay (Y) is used to time-average a signal. Any control signal can be time averaged over a time interval of length PER. If a zero PER is entered, the output of the relay is equal to its input.

Take note

Common syntax for equipment

Each piece of equipment may contain POKE, RAMP, and XY data. For more information, see “POKE and RAMP” on page 423 and “X and Y coordinates” on page 422.

Example input

A time-averaging relay named AVG1 is monitoring the flow reported by a flow sensor named FT1. The relay time-averages the input signal every 10 minutes:

```
Y AVG1 AVERAGE S FT1 10.
```

Data curve (D)

Peek and Poke Attributes

INPREP

Data curves may be expressed using several different input formats. The first format uses SPS's CSV input format. For more information on this format, see "CSV input syntax" on page 52.

```
+ TABLE
{
X, Y
X1, Y1
X2, Y2
...
Xn, Yn
}
```

Alternatively, data curves may be expressed as follows:

```
D NAME TYPE
+ Y1 Y2 Y3...Yn
+ X1 X2 X3...Xn
```

Data curves may also be expressed as follows:

```
D NAME TYPE
+ TABLE X Y
+ X1 Y1
+ X2 Y2
...
+ Xn Yn
```

Finally, data curves may also be expressed as follows:

```
+ TABLE Y X
+ Y1 X1
+ Y2 X2
...
+ Yn Xn
```

Field	Units Key	Description
NAME	n/a	Unique name for the curve.
TYPE	n/a	Type of curve. See "Curve types" in the *Description* section of this topic.

Field	Units Key	Description
X_j	Based on curve type	X coordinates.
Y_j	Based on curve type	Y coordinates.

Description

Data curves (D) allow you to input a function curve of X and Y coordinates that can be referenced by one or more elements in a model. When you reference a data curve from an element, you may apply a multiplier or an ending value to scale the curve. In this way, a curve with the proper shape can be used by multiple elements.

Data curve entries can be placed anywhere in the INPREP file after the =EQUIPMENT line but must precede the elements that reference them.

Curve types

The X and Y coordinates have various units, dependent on the type of curve being defined. Curves can be entered for the following types of functions:

Keyword	Function	Y-units Keyword	X-units Keyword
CLSE	Valve coefficient as a function of time (close)	VALVE.COEFF	TIME
C(X)	Valve coefficient as a function of stem	VALVE.COEFF	Dimensionless
EFFI	Pump efficiency as a function of flow	Dimensionless	PUMP.AFLOW
F(D)	Valve correction coefficient as a function of P	Dimensionless	PRESSURE
F(T)	Input reference output or opening/closing curve for relief valve (RV)	Dimensionless	Dimensionless for input reference or TIME for RV.
HEAD	Pump head as a function of flow	HEAD	PUMP.AFLOW
K(E)	Controller gain as a function of error	Dimensionless	Dimensionless
OPEN	Valve coefficient as a function of time (open)	VALVE.COEFF	TIME
POWR	Pump power as a function of flow	POWER	PUMP.AFLOW
Q(P)	External flow as a function of pressure	FLOW	PRESSURE
TEMP	Temperature as a function of distance	TEMPERATURE	LENGTH.PIPE
TKVL	Volume as a function of level. **Note:** A level of zero corresponds to the elevation of the discharge node.	VOLUME	ELEVATION
V(Z)	Sensor output as a function of input	Dimensionless	Variable
X(V)	Actuator position as a function of signal	Dimensionless	Dimensionless

Example input

Example 1

Define a data curve named PMP_HD of pump head as a function of flow. The head data ranges from 2900 to 1930 (HEAD units) and the flow data ranges from 0 to 149 (PUMP.AFLOW units).

```
D PMP_HD  HEAD
2900  2880  2850  2800  2740  2630  2550  2460
2350  2200  1930 0
  0   17   34   51   68   85   94  102
 111  120  133 149
```

Example 2

Define a data curve named GATE_CS of valve coefficient as a function of time for closing. The valve coefficient data ranges from 11000 to 0.01 (VALVE.COEFF units) and the valve closes in 2 minutes (TIME units).

```
D GATE_CS CLSE
+ TABLE Y X
+ 11000  0
+ 5000  .5
+ 2000  1
+ 1150  1.5
+ 0.01  2
```

STATION

Peek and Poke Attributes

INPREP

```
STATION NAME CNC [LEVEL]
```

Field	Units Key	Description
NAME	n/a	Unique name of the station.
CNC	n/a	Name of a node or a element-end in the station.
LEVEL	n/a	Integer from 1 to 4 that provides a means for controlling the stations displayed on a distance plot. When defining a distance plot, all stations with a level less than or equal to the level you specify are displayed on the distance plot. For example, if you choose 3, you will see all stations assigned level 3, 2, and 1. {1 for user-defined stations; 4 for automatically named stations.}

Description

STATION is used to name a station in order to make distance plots more descriptive and easier to read. Choosing a level allows you to control what stations are displayed on a distance plot.

A *station element* is defined as a node or any hydraulic element other than a pipe. A station “contains” all station elements that can be reached from CNC by traveling only through station elements. TRANS displays a node name for each station when the Stations option is invoked when you create or edit a distance plot. See “Distance plots” on page 169.

Do not specify a station that has previously been defined in another station.

Take note

Automatic naming

A station that has not been named, but contains a named node, is automatically named: ST_nodename where nodename is the first node, alphabetically, in the station. Automatically generated stations have a level of 4.

OUTPRP

The OUTPRP report contains a list of elements within a station.

Example input

Name the equipment connected to ND5, STA100 with a level of 1.

```
STATION STA100 ND5 1
```

X and Y coordinates

```
...
+XY = x  y
```

Field	Units Key	Description
x	n/a	X coordinate value set through Model Builder.
y	n/a	Y coordinate value set through Model Builder.

Description

You can assign coordinate values to each device through Model Builder. These coordinate values are saved in the INPREP file.

POKE and RAMP

```
...
+POKE attribute = expression
+RAMP attribute = expression
```

Field	Units Key	Description
attribute	n/a	Device attribute to be poked at the beginning of the simulation. See "Peek and Poke Keyletters and Attributes" on page 639.
expression	n/a	New value for the attribute; that is, the value to be poked. Arithmetic expressions and other peek names are valid. See "Expressions, Operators, and Functions" on page 587 and "Peek and Poke Keyletters and Attributes" on page 639.

Description

This syntax allows you to define POKEs and RAMPs in INPREP, which are executed at the start of the simulation.

The RAMP expression is evaluated every time step, and the device attribute is poked to that value every time step. For more information on specifying time or delay parameters at specified times, see "POKE" on page 493 and "RAMP" on page 504.

CHAPTER

11

The INTRAN File

The INTRAN file is a required text file. It controls the transient simulation whether or not it is run interactively.

You can control the following aspects of the simulation through the INTRAN file:

- Specify simulation starting and ending times
- Specify whether the simulation is interactive or not
- Request that trend and/or profile data be saved
- Request output at desired times during the simulation
- Specify output data format
- Archive the simulation at any time
- Load a previously archived state or steady state file
- Define a simulation scenario in which pumps or compressors start or stop, set points change, valves open or close, etc.
- Define a new variable based on other variables
- Use text handling facilities like macros
- Define operational sequences, such as station startup or shutdown
- Define alarm conditions and messages
- Set up a conditional control sequence so that certain events may be triggered based on what is happening in the model

Required input for the INTRAN file

Every INTRAN file requires the BEGIN command. However, you may also want to include the following minimal commands:

- BEGIN (must be the first line in the file)
- INTERACTIVE (if you want to run the simulation interactively)
- TRENDLIST (if you want to produce interactive and/or time plots or run TPORT)
- PROFILE (INTRAN) (if you want to produce profile plots)
- SHARE (if you want to run TPORT)

Note: The INTRAN file must be entered in capital letters. For more syntax rules, see "Input syntax" on page 47.

INTRAN file input

The following commands may be used in the INTRAN file:

- "SHARE" on page 524
- "SHOW.STEADY" on page 527
- "START" on page 528
- "STOP" on page 531
- "SUBMIT.SEQUENCE" on page 534
- "TIMEPAGE" on page 536
- "TRENDLIST" on page 537
- "WAIT, WAIT.UNTIL" on page 541
- "WHENEVER" on page 542

In addition, the following common commands may be used in the INTRAN file:

- "DEFINE" on page 568
- "DEFINE.PATH" on page 571
- "INCLUDE" on page 573
- "IFELSE" on page 576
- "IFISMACRO" on page 579
- "MACRO" on page 580
- "TESTMACRO" on page 584

For more information on using logic with commands, see "Expressions, Operators, and Functions" on page 587.

Sample INTRAN file

```
BEGIN 0,                    /* start a new simulation with the time
+ BEGIN.TIME = 0,           /* equal to zero
+ END.TIME = 1440           /* run the simulation for 1 day (1440 minutes)
INTERACTIVE                 /* specify the simulation as interactive
                            /* using Windows
MACRO(INIT, SHOW PIPE1)     /* when the interactive run begins, the
                            /* Show window for the element PIPE1
                            /* initially comes up
TRENDLIST *, KEY.LETTER = T B
                            /* request trend data to be stored to the REVIEW
                            /* file for all transfer lines and block valves
PROFILE 10                  /* request profile data for transfer lines
                            /* to be stored every 10 minutes to the
                            /* REVIEW file
DEFINE TOTHP = UNIT1:PWR + UNIT2:PWR
                            /* define a new variable that is the total
                            /* pump/compressor horsepower for
                            /* units named UNIT1 and UNIT2
START UT1, TIME = 60        /* start a pump/compressor named
                            /* UT1 at time = 60 minutes
START UT2, TIME = 61        /* start a pump/compressor named UT2
                            /* at time = 61 minutes
```

```
OPEN BLOCK1, TIME = 92      /* open a valve named BLOCK1 at time 92 minutes
POKE OUTLET:SP = 100, TIME = 102
                            /* change the pressure at an external
                            /* named OUTLET to 100 psig at time = 102 minutes
ARCHIVE ARK1, TIME = 120    /* archive the simulation at 2 hours to a
                            /* file named ARK1.ARK
```

ALARM

INTRAN

```
ALARM(ALN) = CONDITIONAL[,
+ FIRST.MESSAGE = "ALARM-MESSAGE"][,
+ REPEAT.MESSAGE = "ALARM-MESSAGE"][,
+ REPEAT.PERIOD = TIME][,
+ CLEAR.MESSAGE = "ALARM-MESSAGE"][,
+ CATEGORY = ALARM-CATEGORY]
```

Field	Units Key	Description
ALN	n/a	Alarm name or number. If a number is given to the alarm as ALN, the alarm name is *ALARM_number.* If a name is given to the alarm as ALN, the alarm name is *name*, and TRANS automatically numbers the alarm sequentially. Avoid mixing numbers and names for ALN.
CONDITIONAL	n/a	Alarm condition
ALARM-MESSAGE	n/a	Message triggered by the alarm
TIME	TIME	Time interval at which the repeat message is generated
ALARM-CATEGORY	n/a	User-defined category for the alarm. The ALARM.CATEGORY must be defined prior to being referenced by an alarm command. {ALMCAT_DEFAULT}

Description

ALARM allows any number of alarm conditions to be defined, as well as up to three different messages per alarm. An alarm is triggered initially when a CONDITIONAL that was false becomes true.

Messages are generated as follows:

- When the alarm condition that was false becomes true, the FIRST.MESSAGE is generated.
- While the alarm condition remains true, the REPEAT.MESSAGE is generated at a time interval specified by the REPEAT.PERIOD time entered.
- At the first time step when the alarm condition, which was true, becomes false, the CLEAR.MESSAGE is generated. No additional messages generate until the alarm condition is again true.

The alarm messages are appended to the "OUTTRN file", the "EVENTS file" and the "ALMLOG file" every time step. (The ALMLOG file contains *only* the alarm messages whereas the OUTTRN and EVENTS files contain the messages *and* the times associated with the messages).

When you run the simulation interactively and an alarm is triggered, an E# appears next to the <Running> message at the top of the display. (The # is sequential and corresponds to the number of alarms that have sounded since the beginning of the simulation to the current time). Type TYPE ALMLOG or TYPE EVENTS at the > prompt to look at the alarm messages. Once the alarm is cleared, the E# disappears from next to the <Running> message.

Example input

Specify a set of alarms named LOW_SUCTION_PRESSURE to be generated when the suction pressure of an element named UNIT falls below 300. The alarm message repeats every one minute until the alarm condition clears.

```
ALARM(LOW_SUCTION_PRESSURE) = UNIT:P- < 300,
+ FIRST.MESSAGE = "SUCTION PRESSURE, [UNIT:P-], TOO LOW",
+ REPEAT.MESSAGE = "SUCTION PRESSURE STILL TOO LOW",
+ REPEAT.PERIOD = 1,
+ CLEAR.MESSAGE = "SUCTION PRESSURE OK NOW"
```

The peek attributes can be embedded in the message text. In the example, the value of [UNIT:P-] is displayed as part of the first alarm message.

Note: The alarm name for this example is LOW_SUCTION_PRESSURE.

ALARM.CATEGORY

INTRAN

ALARM.CATEGORY $NAME_1$ [$NAME_2$ ... $NAME_i$ + $NAME_i$ ...]

Field	Units Key	Description
$NAME_i$	n/a	Name of alarm category to create

Description

ALARM.CATEGORY defines categories to which alarms may be assigned. This command is used in conjunction with the CATEGORY field in the ALARM command and must appear in the INTRAN file *before all* alarm commands, regardless of whether they reference a category or not.

Categories can be defined to segregate alarms by severity such as INFO, WARN, or ERROR. Other uses may be to maintain information on alarms by pipeline system (LINE_1_ALARMS, LINE_2_ALARMS) or by a combination of severity and pipeline (LINE_1_INFO, LINE_2_INFO, LINE_1_WARN, etc.). There is no practical limit to the number of categories that can be defined or the number of alarms that can be assigned to a given category.

Take note

Enabling or disabling alarms

An alarm category can be poked to DISABLE, which disables all of the alarms assigned to that category.

One command per INTRAN file

Only one alarm category command is allowed in an INTRAN file; however, the command may contain multiple lines of input.

Alarm categories

Schematic supports four alarm categories (INFO, WARNING, ALERT, and FATAL). These alarm categories share the same warning sound or beep and have distinct icons. If a different category is defined, it will be in the DEFAULT category, which has a generic icon.

Example input

Define alarm categories allowing alarms to be assigned to one of four categories depending on the severity. The alarm categories are INFO, WARNING, ERROR, and FATAL.

```
ALARM.CATEGORY INFO WARNING ERROR FATAL
```

ARCHIVE

INTRAN

```
ARCHIVE FILENAME [, TIME = T1 T2 ... Tn] [, DT = INTERVAL]
```

Interactive

```
ARCHIVE FILENAME
```

Field	Units Key	Description
FILENAME	n/a	Name of the archive file. The ARK extension is automatically appended for most installations, unless a specific extension is input or a peek attribute has been specified. The file name may contain a peek attribute enclosed in square brackets [ ]. The peek attribute will be replaced with the variable's value. In this case, no extension is automatically appended to the resulting file name.
T_i	TIME	Simulation time at which an archive should occur. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Interval in minutes at which the archive occurs.

Description

ARCHIVE takes a "snapshot" of the system state at a specified time and stores the data (every value, equipment status, etc.) at the specified time interval. If multiple archives are written to one file, the file contains only the latest archived state.

An archive file contains a single restart record (i.e., system status at a given point in the simulation) and may be used as the initial status for a subsequent simulation. The archived state may be used to initialize another simulation, or to re-initialize the current (ongoing) simulation using the LOAD.STATUS command. For more information on initialization, see "Preparing to run a simulation" on page 153. For more information on the archive file, see "Relationship of TRANS and related files" on page 55.

Example input

Example 1

Archive the state of the simulation to a file named STEADY.ARK.

```
ARCHIVE STEADY
```

The ARK extension is automatically appended.

Example 2

Archive the state of the simulation and name it “d” followed by the time value that the archive was executed:

```
DEFINE ARK_TIME=TOSTR(TIME)
ARCHIVE d[ARK_TIME], DT=1440
```

The file name has no extension.

Example 3

Archive the state of the simulation and name it to the value of P+ of an element named TL1 at the time the archive was executed:

```
ARCHIVE [TL1:P+].ARK
```

Example 4

Archive the simulation at the clock time November 6, 2002 at 1:40 p.m. Update the archive file every 5 minutes. Name the file STEADY.ARK.

```
ARCHIVE STEADY, TIME = “02/11/06 13:40:00”, DT = 5
```

Example 5

Create a file called save_1_12.ark, which is the pipeline state at noon on November 6, 2002.

```
ARCHIVE "SAVE_[DAY(TIME),2]_[HOUR(TIME + 30),2]",
TIME = "02/11/06 12:00:00"
```

The DAY and HOUR are time functions. For more information, see “Time function operators” on page 632.

The +30 indicates that if the archive file is created slightly before noon as a result of the minimum time step restriction, the file name will be save_1_12.ark instead of save_1_11.ark.

The 2 indicates that you wish the value to be displayed with a maximum of two significant digits to the left of the decimal and zero significant digits after the decimal place (in other words, an integer with two significant digits).

BEGIN

INTRAN

```
BEGIN RECNO[,
+ BEGIN.TIME = BEGIN.TIME][,
+ END.TIME = END.TIME][,
+ PRESSURE.TOLERANCE = PRES.TOLER][,
+ TEMP.TOLERANCE = TEMP.TOLER][,
+ DELAY.PRINT = DELAY]
```

Field	Units Key	Description
RECNO	n/a	Initial condition for the simulation. If the field is 0, TRANS uses restart record 0 (generated by PREPR) or an archive file (generated by TRANS) as the initial starting point.
BEGIN.TIME	TIME	Start time for the simulation. For more information on formats, see "Elapsed time versus clock format" on page 199.
END.TIME	TIME	End time for the simulation. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** Overrides the time value in the TIMEPAGE command. TIMEPAGE is supported for older models but END.TIME should be used. {endtime specified in the TIMEPAGE command or 1E20}
PRES.TOLER	ΔPRESSURE	A numeric value representing the maximum error tolerance for pressure in the simulation. For most simulations, TRANS runs both faster and more accurately with a moderately small error tolerance, such as 1 or 2 psi. {2 psi}
TEMP.TOLER	ΔTEMPERATURE	A numeric value representing the maximum error tolerance for temperature in the simulation. {PRES.TOL/20 °F, where PRES.TOLER is converted to PSI}
DELAY	TIME	Amount of simulation time to delay until the echo print of the POKE/SET commands is enabled. For more information on formats, see "Elapsed time versus clock format" on page 199. Used to reduce the size of the OUTTRN file and/or improve model startup time.

Description

BEGIN is used to start or restart a simulation and specify some tolerance settings. This command must be the first line of input in the INTRAN file. For more information on initialization settings, see "Preparing to run a simulation" on page 153.

Example input

Example 1

Start a new simulation from an initial state. The start time is zero. The simulation will run to the default end time. Run for 1 day (1440 minutes).

```
BEGIN 0,
+ BEGIN.TIME = 0,
+ END.TIME = 1440
```

Example 2

Start a new simulation from an initial state. The start time is defined as six hours (360 minutes). Run for 1 day (1440 minutes).

```
BEGIN 0,
+ BEGIN.TIME = 360,
+ END.TIME =  1440
```

Example 3

Start the simulation from an initial record from the RESTRT file and November 6, 2002 at 1:40 p.m. Run for 2 days.

```
BEGIN 0,
+ BEGIN.TIME = TIMEVALUE("02/11/06 13:40:00")
+ END.TIME = TIMEVALUE(“02/11/08 13:40:00”)
```

CALL.SEQUENCE

INTRAN

```
CALL.SEQUENCE NAME [(ARG1, ... ARGn)] [, TIME = TIME] [, DT = INTERVAL]
```

Field	Units Key	Description
NAME	n/a	Name of the defined sequence to be called.
ARG_j	n/a	Optional sequence arguments.
TIME	TIME	Time when the defined sequence is to be called. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval when the defined sequence is to be called (referenced back to time 0 rather than the BEGIN.TIME value).

Description

CALL.SEQUENCE calls a defined sequence within another command list. The called sequence must have previously been defined using the DEFINE.SEQUENCE command. CALL.SEQUENCE must always be entered within a another command list. If you want to call a defined sequence as a stand-alone command, use "SUBMIT.SEQUENCE" on page 534.

If arguments were included in the DEFINE.SEQUENCE definition, they must be passed to the DEFINE.SEQUENCE when the defined sequence is called.

The CALL.SEQUENCE command differs from the SUBMIT.SEQUENCE command in that the SUBMIT.SEQUENCE continues execution immediately when the defined sequence is submitted while CALL.SEQUENCE does not continue execution until the called command list finishes. Consider the following example:

```
DEFINE.SEQUENCE ABC
{
   WAIT 1
   OPEN BLOCK1
}

WHENEVER(XXX)
{
    SUBMIT.SEQUENCE ABC
    OPEN BLOCK2
}

WHENEVER(YYY)
{
    CALL.SEQUENCE ABC
```

```
    OPEN BLOCK2
}
```

When the XXX WHENEVER executes, BLOCK2 will be opened one minute before BLOCK1. The WHENEVER will not wait before opening BLOCK2. When the YYY WHENEVER executes, BLOCK2 will not be told to open until ABC completes, one minute after the YYY event occurred.

Example input

Example 1

Within a WHENEVER statement, call a sequence named ABC.

```
WHENEVER (xxx){
 CALL.SEQUENCE ABC}
```

CLOSE

INTRAN

```
CLOSE NAME [, TIME = T1 T2 ... Tn] [, DT = INTERVAL] [, KEEP.OLD=YES]
```

Interactive

```
CLOSE NAME [, TIME = TIME] [, DELAY = DELAY]
```

Field	Units Key	Description
NAME	n/a	Name of a block valve to be closed.
T_j	TIME	Simulation time or times at which the block valve starts to close. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the block valve starts to close (referenced back to time 0 rather than the BEGIN.TIME value).
TIME	TIME	Simulation time at which the block valve starts to close. For more information on formats, see "Elapsed time versus clock format" on page 199.
DELAY	TIME	Time from the current simulation time at which the block valve starts to close. Avoid using with models running in TPORT.

Description

CLOSE causes a block valve to start closing. The time at which the valve starts to close depends on which time option is used when the command is entered. Different time options are available in the INTRAN file and interactively.

INTRAN

Command setup	Effect
CLOSE is by itself and does not target a specific time	Begins to close the valve when the simulation begins
CLOSE is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Begins to close the valve when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
CLOSE targets a specific time, multiple times, or a time interval	Begins to close the valve at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

Command setup	Effect
CLOSE does not target a time	Begins to close the valve at the beginning of the next time step of the simulation
CLOSE targets a specific time	Begins to close the valve at the specified time
CLOSE specifies a delay	Begins to close the valve after the delay period expires

Take note

Closing other valves

The CLOSE command does not work for check valves, regulators, or actuated control valves. Check valves close automatically based on the pressure differential across the valve. Regulators close due to pressure, flow, or fraction set points. Actuated control valves close in response to an actuator's output signal. Relief valves open and close based on pressure set points.

Command may be overridden

STOP or OPEN override CLOSE if either is given after the CLOSE command.

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

KEEP.OLD feature—INTRAN

You may enable "old" actions to occur by adding the KEEP.OLD = YES statement to the CLOSE command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 60
CLOSE VALVE1, TIME = 90
CLOSE VALVE2, TIME = 30
CLOSE VALVE3, TIME = 0
```

Since the simulation starts at time = 60, the "CLOSE VALVE2" command will not execute, since its execution time has already passed at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the OPEN command execute anyway. For example, the following CLOSE would occur at a simulation start time = 60, despite its specified time:

```
CLOSE VALVE2, TIME = 30, KEEP.OLD = YES
```

In the sample, note that the "CLOSE VALVE3" command will execute regardless of the simulation start time, because the parameter TIME = 0 indicates that the CLOSE command should be executed at start time.

Example input

Example 1

Start closing a valve named VALVE1 as soon as the simulation begins.

```
CLOSE VALVE1
```

Example 2

Start closing a valve named VALVE1 at a simulation time of four hours (240 minutes).

```
CLOSE VALVE1, TIME = 240
```

Example 3

Start closing a valve named VALVE1 10 minutes from the current simulation time.

```
CLOSE VALVE1, DELAY = 10
```

Example 4

Start closing a valve named VALVE1 at 1:40 p.m. on November 6, 2002.

```
CLOSE VALVE1, TIME = "02/11/06 13:40:00"
```

COLSEP

INTRAN

```
COLSEP [COLAPS = COLAPS,] [BAND = BAND,] [RATE = RATE]
```

Field	Units Key	Description
COLAPS	n/a	A constant associated with bubble collapse rate {20 reciprocal seconds}
BAND	ΔPRES	The value of dead band below the bubble point before separation occurs {2 psi}
RATE	VEL/ΔPRES	A constant associated with bubble growth {0.001 ft/(sec*psi)}

Description

COLSEP simulates:

- Fast acting column separation and subsequent collapse due to upset conditions
- Slower acting slack line flow in which a perhaps relatively large section of the pipe is flowing but is only partially full because the hydraulic grade line is below the elevation.

Column separation is supported by the SCL and SCLPROP equations of state. See "STATE SCL" on page 230. It occurs only in transient pipes (T and GP). Column separation is supported with any thermal mode, but heat gain or loss associated with the vaporization and subsequent condensation is ignored even with TRANSTHERMAL modeling, and this can be an important effect.

If the COLSEP option is used, the vapor pressure, P_0, at which column separation occurs is determined from the equation of state. Below (P_0 - BAND), column separation occurs. SPS assumes that vapor volume is negligible and that vapor/liquid flow is homogeneous. Two-phase transient slip flow is not simulated.

Once the pressure in a pipe falls below (P_0 - BAND), vapor cavity formation occurs. The volumetric rate of growth for the bubbles is proportional to the surface area of the bubble and the difference between P_0 and P, the pressure in the liquid. RATE is the proportionality constant.

During a pressure increase, when P exceeds P_0, bubble collapse begins. It follows the same proportionality for collapse as for growth until the ratio of the bubble collapse rate to the bubble volume exceeds the value COLAPS, at which time the collapse rate is taken to be proportional to the value of COLAPS multiplied by the bubble size.

TRANS can reject time steps when column separation occurs.

Take note

Use when slack line flow is anticipated

Use COLSEP when simulating a liquid system in which slack line conditions might exist, such as at the top of a hill. If a pressure below the vapor pressure is calculated and no COLSEP command is present in the INTRAN file, erroneous results may occur.

Profiling liquid fraction

On a distance plot, a liquid fraction (LIQ.FRAC) variable can be profiled to determine if slack line flow is occurring. The liquid fraction range is from 0 to 1. LIQ.FRAC < 1 indicates slack line flow is occurring.

Example input

Include the column separation feature with a vapor cavity collapse rate of 9 sec^{-1}:

```
COLSEP 9
```

Command list

INTRAN

```
{COMMAND LIST}
```

Description

Command lists are used with the "DEFINE.SEQUENCE", "WHENEVER", and "IF" commands and may be composed of one or more commands. Multiple commands may be separated by semicolons on a single line or placed on separate lines in the INTRAN file. The following types of commands are allowed in a command list.

POKE/SET commands

```
POKE CNTL1:SP = 450
SET ABC:CF = 0
```

Status change commands (START, OPEN, etc.)

```
START UNIT1
CLOSE VALVE1
```

WAIT statements

```
WAIT 5
WAIT.UNTIL ( OUTLET:P+ < 300 )
```

SUBMIT.SEQUENCE commands

```
SUB.SEQ START_SEQUENCE
```

IF statements

```
IF ( OUTLET:P+ < 300 )
  { OPEN BB1 }
ELSE
  { OPEN BB2 }
```

Example input

Set up a command list for system shutdown whenever the pressure of an external named EDELIV rises above 600.

```
WHENEVER( EDELIV:P+ > 600 )
{
  STOP UNIT3
  WAIT 2
  START UNIT4
  WAIT.UNTIL (OUTLET:P+ > 750)
  CLOSE VALVE112
  IF (INLET:P+ > 300) { CLOSE INVALVE }
}
```

DEFINE.FUNCTION

INTRAN

The DEFINE.FUNCTION input may be expressed in one of several different formats. The following input format uses the keyword value format:

```
DEFINE.FUNCTION name = expression
/*or
DEFINE.FUNCTION name,
+ Y = y1 y2 y3 ... yn,
+ X = x1 x2 x3 ... xn[,
+ REPEAT = YES | NO]
```

As an alternative, you my use SPS's CSV input format, which was introduced in SPS version 9.6. For more information on this format, see "CSV input syntax" on page 52.

```
DEFINE.FUNCTION name,
+ TABLE
{
X Y
x1 y1
x2 y2
x3 y3
...
xn yn
}
[+ REPEAT = {YES | NO}]
```

Field	Units Key	Description
name	n/a	Name of the defined function
expression	n/a	Valid expression. When used, do not enter Y and X values.
$y_1...y_n$	n/a	Value associated with the corresponding X value in the command
$x_1...x_n$	n/a	Value associated with the corresponding Y value in the command
REPEAT	n/a	If time is specified as the X variable, the function can be repeated. The first and last Y value should be the same because the first x and last x are considered to be the same point. {NO}

Description

DEFINE.FUNCTION sets up a schedule (supplies/deliveries that are sawtooth in character, controller set point changes, etc.) that may be used for multiple devices. A RAMP command must be included in the INTRAN file to invoke the function name. This command is similar to the Data curve (D) command in PREPR. Alternatively, DEFINE.FUNCTION may be set to an expression.

Take note

Interactive editing

For the following discussion, see the following defined function and see “DEFINE.FUNCTION (DF) attributes” on page 654 for a list of attributes displayed on a DEFINE.FUNCTION Show window. Only the first five points will be displayed.

A defined function named DEFF is set up as follows.

```
DEFINE.FUNCTION DEFF,
+ Y = 10 9 8 7,
+ X = 1 4 9 12
```

DEFF has four pairs: (10,1) (9,4) (8,9) (7,12).

To delete the third pair, type the following.

```
POKE DEFF:DEL(3) = YES
```

This deletes the (8,9) pair. The (7,12) pair then become the third pair. The count will decrease from four to three. (The count will never go below one. A defined function will always have at least one pair in it, no matter how many times the DEL(i) variable is poked).

To insert a new pair into the original defined function DEFF, type the following.

```
POKE DEFF:NEW_X = 3
```

This inserts a new pair where X = 3. Because the X variables must always be in increasing order, the 3 will be inserted between the 1 and 4 (now the second pair) and the new Y value will be the same as the Y value of the original second pair (which is 9, corresponding to the X value of 4). The new function will look like:

```
+ Y = 10 9 9 8 7,
+ X = 1 3 4 9 12
```

To change the Y value for the second pair (X = 3) from 9 to a value of 12, type the following.

```
POKE DEFF:Y(2) = 12
```

To see the value of the X or Y for a given pair, type the following.

```
DEFINE VAR = DEFF:X(4)
DEFINE VAR = DEFF:Y(1)
```

where:

VAR is the name of a defined variable (a different name for each DEFINE).

This can be done in the INTRAN file or interactively, and the Show window for the defined variable displays the value of the defined X or Y point.

Archives

DEFINE.FUNCTIONs survive archives. If the DEFINE.FUNCTION is duplicated, the duplication is resolved, based on the number of parameters. When both have the same number of parameters, the new one overwrites the old; otherwise, the new one is ignored.

Example input

Example 1

```
DEFINE.FUNCTION SAWTOOTH,
+ Y = 0  1.0  2.0  1.0  0.0  1.0  2.0  1.0  0.0,
+ X = 2  4    6    8    10   12   14   16   18,
+ REPEAT = YES
RAMP EINLET:NQ = 50*(SAWTOOTH(TIME)) + 100
```

In this example, the data curve is named SAWTOOTH. A dimensionless saw-tooth repeating function with a natural period of 16 minutes is defined. (The SAWTOOTH(TIME) value input into the RAMP command is the Y value associated with the corresponding time (X) value.) The RAMP command invokes the SAWTOOTH curve by defining the NQ (nominal flow) of an external named EINLET based on the X and Y values entered.

The RAMP command will cause EINLET:NQ to track a saw-tooth function that starts at 100 at time = 2 minutes, increases to 150 at time = 4 minutes, increases to 200 at time = 6 minutes, decreases to 150 at time = 8 minutes, decreases to 100 at time = 10 minutes, increases to 150 at time = 12 minutes, increases to 200 at time = 14 minutes, decreases to 150 at time = 16 minutes, and decreases to 100 at time = 18 minutes. Because the function repeats, the value [EINLET:NQ] will be 150 at time = 20 minutes (because time 20 and time 4 are the same point).

Example 2

```
DEFINE.FUNCTION FLOW,
+ TABLE
{
X Y
0 .75
120 .8
240 .85
360 1.1
480 .99
600 .86
720 .68
840 .64
960 .72
1080 .79
1200 .82
1320 .8
1440 .75
}
+ REPEAT = YES
RAMP E1:NQ = 96 * (FLOW(TIME))
RAMP E2:NQ = 8 * (FLOW(TIME))
RAMP E3:NQ = 20 * (FLOW(TIME))
```

In this example, the X values correspond to time (in minutes). This function (named FLOW) repeats daily (the :NQ value at the beginning of the day is the same as the :NQ value at the end of the day). The :NQ values of the 3 different externals are ramped to different set points every 2 hours (120 minutes) by multiplying each Y value by the specified factor.

This example shows how a generic function may be set up and referenced differently by each external. For example, each Y value (:NQ set point) for external E1 is multiplied by 96. So at time 240, the :NQ value for external E1 is targeted at 81.6 (.85 * 96). Each Y value (:NQ set point) for external E2 is multiplied by 8. So at time 240, the :NQ value for external E2 is targeted at 6.8 (.85 * 8). Each Y value (:NQ set point) for external E3 is multiplied by 20. So at time 240, the :NQ value for external E3 is targeted at 17 (.85 * 20).

DEFINE.SEQUENCE

INTRAN

```
DEFINE.SEQUENCE[.REPLAY] NAME [( ARG1, ARG2, ... ARGn )]
{
  command list
}
```

Field	Units Key	Description
REPLAY	n/a	Indicates whether to write the DEFINE.SEQUENCE to the REPLAY file when it is executed.
NAME	n/a	Name of the command sequence
ARG_j	n/a	Optional arguments passed to command list that allow numbers or peek values to be passed to the command list at the time the command sequence is executed. These arguments may be used in statements in the command list. The argument may also be a complete peek name. The DEF.SEQUENCE cannot be used to construct peek names. Some devices have default peek attributes that permit only the device name to be passed as an argument. For pumps/compressors and valves, the default is :ST; for DEFINEs the default is :VAL. For these devices, the attributes are automatically appended only if the device name is passed as an argument.
command list	n/a	See "Command list" on page 443.

Description

DEFINE.SEQUENCE defines a new command sequence in terms of existing commands. The new command sequence may be executed in the INTRAN file or interactively from the command line.

The command list is an arbitrary list of scenario control statements, WAIT statements, and nested IF statements. See "Command list" on page 443 for more information. The command list will be executed in the INTRAN file when the sequence identified by NAME is submitted by the SUBMIT.SEQUENCE command. Interactively, a defined sequence may be executed by entering the sequence name only.

Example input

Example 1

Define a command sequence named OPEN_START that opens a valve named VALVE1, waits two minutes and then starts a pump/compressor named UNIT1.

```
DEFINE.SEQUENCE OPEN_START
{
   OPEN VALVE1
   WAIT 2.0
```

```
   START UNIT1
}
```

Example 2

Define a command sequence named DELPRES that sets the set point pressure argument at two delivery externals named DEL1 and DEL2.

```
DEF.SEQ DELPRES(PRESS)
{ /* Set all delivery pressures
  POKE DEL1:SP = PRESS + 20
  POKE DEL2:SP = PRESS
}
```

DEFINE.TIMETABLE

INTRAN

```
DEFINE.TIMETABLE timetable [(arg1,...,argn)]
{
  [REPEAT = [NONE | HOURLY | DAILY | MONTHLY | ANNUALLY | trep][,
  START.INTERVAL = tj][,
  EVERY = [NONE | NOSCHEDULE | CONTINUOUS | tevery]] [,
  SET arg1 = val1] ... [, SET argn = valn ]
}
```

Field	Description
timetable	Name of the time table
arg_i	Variables defined and controlled by the timetable
REPEAT	How often the time table, as a whole, repeats itself {NONE}
t_j	Schedule-time of a schedule-interval. SPS attempts to end a step exactly at each schedule-time. The schedule-time is relative to the REPEAT period. SPS is at a schedule-time if it is within 1/2 of DTMIN of a schedule-time. Time is usually entered in clock format. For more information on formats, see "Elapsed time versus clock format" on page 199. Multiple START.INTERVALs are allowed.
tevery	During a schedule-interval, additional schedule-times may be defined on a periodic basis in minutes.
NOSCHEDULE	Indicates that this START.INTERVAL time is NOT a schedule-time, it is used mainly to cancel an earlier START.INTERVAL and EVERY combination.
CONTINUOUS	The value of arg_i is updated continuously. Also the peek, timetable:SCHED, is set to 1 every step of schedule-interval.
val_i	Each schedule-time may set the value of arg_i. If an interval does not SET an arg_i that was given in the initial list of variables, the arg_i maintains the value last SET in a previous schedule-interval, or 0 if no SET appears in any interval. This value can be either a string or numerical constant.

Description

DEFINE.TIMETABLE defines times when actions are to occur. The time table allows for varying schedules on a daily, weekly, or monthly basis. A time table may be used in place of the standard syntax used for time entries. Commands accepting time(s) have following syntax:

```
...... [ , TIMES = t1 ... tm ] [ , DT = dt ]
```

A TIMETABLE may be substituted using the following syntax:

```
...... [ , TIMETABLE = timetable ]
```

The command is executed at every schedule-time as designated, and timetable-related variables are set if timetable is used.

Example input

Example 1

The current model should initiate ARCHIVES according to the following schedule.

Time Range	Frequency	Casename
0600 - 1200	Never	N/A
1200 - 1500	3 Hours	AFTERNOON
2100 - 0530	30 Minutes	MORNING
1800 - 2000	1 Hour	EVENING

Enter the following:

```
DEFINE.TIMETABLE ARCSCHED ( CASE )
{
  START.INTERVAL = "06:00", EVERY = NOSCHEDULE
  START.INTERVAL = "12:00", EVERY = 180, SET CASE = "AFTERNOON"
  START.INTERVAL = "18:00", EVERY = 60, SET CASE = "EVENING"
  START.INTERVAL = "21:00", EVERY = 30, SET CASE = "MORNING",
    REPEAT=DAILY
}
ARCHIVE [ ARCSCHED:CASE], TIMETABLE = ARCSCHED
```

Example 2

Print a report regularly every hour.

```
DEFINE.TIMETABLE HOURLY
{
  START.INTERVAL = "00:00", EVERY = 60,
  REPEAT=DAILY
}
PRINT REG, TIMETABLE = HOURLY
```

DO.INTERACTIVE

INTRAN

```
DO.INTERACTIVE "NAME"
/* or
DO.INTERACTIVE "STRING"
```

Field	Units Key	Description
NAME	n/a	Name of the display
STRING	n/a	Name of a text string or name of a string of commands to run

Description

DO.INTERACTIVE generates interactive commands from the INTRAN file. This command accepts a quoted string as an argument. The quoted string is treated as text entered from the keyboard at the interactive prompt (>). See the following example INTRAN command:

```
WHENEVER(PIPE1:P+ > 1000)
{
  POKE STATUS.MSG = "PRESSURE AT XYZ STATION TOO HIGH"
  DO.INTERACTIVE "SHOW PIPE1"
}
```

Whenever PIPE1:P+ is greater than 1000, the user-defined variable named STATUS.MSG is poked from its current value to the text string “PRESSURE AT XYZ STATION TOO HIGH” and then the Show window for the element named PIPE1 is displayed.

DO.INTERACTIVE strings are not executed immediately upon reading the DO.INTERACTIVE command. They are executed at the beginning of the next interaction. If several DO.INTERACTIVE commands are entered in the INTRAN file, the commands are executed in the order that the DO.INTERACTIVE commands were entered in the INTRAN file.

Example input

In the following example, a display named ABC.DSP displays, the simulation takes a time step, the display named XYZ.DSP displays, and then the simulation pauses waiting for your input.

```
DO.INTERACTIVE "ABC; RUN"
DO.INTERACTIVE "XYZ; PAUSE"
```

External (Named Fluid)

INTRAN

Single component case:

```
E EXTNAM FLUIDNAM [,
+ INVENTORY = INV    /*  INVENTORY FOR TANK EXTERNALS ONLY]
```

Multiple component case:

```
E EXTNAM FLU1 FLU2 ...  FLUn ,
+ FRACTION =  FR1 FR2 ... FRn [,
+ INVENTORY = INV    /*  INVENTORY FOR TANK EXTERNALS ONLY]
```

Connection-defined case:

```
E EXTNAM ±CNC

E EXTNAM SG HHV             /*  CNGA ONLY

E EXTNAM SG CO2 HHV[,       /*  AGA ONLY
+ USER LAB1 ... LAB8][,
+ VAL VAL1 ... VAL8]
```

Field	Units Key	Description
EXTNAM	n/a	Name of an external that delivers fluid into the system
FLUIDNAM	n/a	A Named fluid (STATE SCL) or a component (STATE BWRS) that enters into the system at this external
INV	VOLUME	The initial volume of fluid at the external. Used to simulate a tank.
FLU_j	n/a	A Named fluid (STATE SCL) or a BWRS component name
FR_j	n/a	Relative mass (mole) fraction of FLU_j in the mixture to be delivered into the system at this external
±CNC	n/a	Name of a connection point. (A node name may *not* be entered here.)
SG	n/a	Gas specific gravity
CO2	n/a	Mole fraction of CO2 (AGA) in the gas
HHV	HEAT.VALUE	Higher heating value of the gas (AGA)
LAB_j	n/a	Name of the user-defined property (AGA)
VAL_j	n/a	Value for the user-defined property (AGA)

Description

This command can also be used with the CNGA equation of state to specify a gas with a different specific gravity entering the system at the specified external. If the fluid and initial fractions have been specified as part of the SALE/TAKE type external, then this command will be disabled. The named fluid external (E) command is used to define the

fluid type entering an external (for STATE SCL or STATE BWRS multiple fluid simulations). Any component specified in this command must be included in the STATE SCL or the STATE BWRS equation of state in the INPREP file, even if a value is zero.

Initially all pipes and equipment are filled with either the first fluid named in the equation of state in the INPREP file or the initial fluid batches defined by using the LINE.FILL command in the INTRAN file. At runtime, all externals used as input points by default deliver the initial fluid or the fluid specified in INPREP.

In the single component case, the fluid delivered into the system is the SCL name or BWRS component designated by FLUIDNAM.

In the multiple component case, the fluid entering the system at this external is a mixture of fluids whose SCL names or BWRS components are designated by FLU_1 FLU_2... with the relative mass ratios of each component given by FR_1 FR_2... IF BWRS.MOLE is used the fractions are the mole ratios.

In the connection-defined case, the fluid delivered into the system at this external has the same composition as the fluid flowing at the location identified by ±CNC. It is permissible to specify ±CNC as the connection location of EXTNAM. In this case, the fluid entering is always of the same composition as the fluid present previously. One use of ±CNC as the node for EXTNAM is for SURGETANKS, which are modeled as externals. The SURGETANK external should usually have a record in the INTRAN file specifying its

±CNC as the same ±CNC used for the connection of the SURGETANK in the INPREP file.

If the flow direction changes so that EXTNAM becomes a point of outflow, the fluid exiting the system at that point is, of course, that which is contained in the system at the node to which EXTNAM is joined. However, should the flow direction change so that the fluid once more enters the system from EXTNAM, the fluid entering is of the composition defined by this record.

Example input

Example 1

Define the named fluids that are supplied at specific externals. An external named SUPPLY1 supplies a fluid named 1.FLUID into the system. An external named SUPPLY2 supplies a fluid named 2.FLUID into the system. An external named SUPPLY3 supplies a fluid named 3.FLUID into the system. An external named SUPPLY4 supplies a fluid named 4.FLUID into the system.

```
E SUPPLY1 1.FLUID
E SUPPLY2 2.FLUID
E SUPPLY3 3.FLUID
E SUPPLY4 4.FLUID
```

Note: The fluid names must be specified in the equation of state in the INPREP file. The externals must also be defined in the INPREP file.

Example 2

A surge tank named SRG2 is defined in the INPREP file as follows.

```
E SRG2 +PIPE1  SURGETANK VCYL 10. .1 300. 8.
+ 3
```

The following INTRAN input line causes the fluid that may enter the connection point at NOD2 as the Surge Tank breathes in and out to be the same as that at the connection point.

```
E SRG2 +PIPE1
```

FORMAT

INTRAN

```
FORMAT MATCH DECIMAL
```

Field	Units Key	Description
MATCH	n/a	A string of characters or user-defined variables
DECIMAL	n/a	The number of decimal places to be displayed to the right of the decimal point

Description

FORMAT controls the display format of numeric peek values or user-defined variables. It identifies the fields to control by matching a character string that is entered as data and designates the number of places displayed to the right of the decimal point. Variables are a combination of an element name and a peek attribute.

You may select for format control certain specific classes of displayed peek items (i.e., all the from-end pressures) or any particular field (as determined by the string of characters it contains). The selected fields have numeric values displayed with a designated number of places to the right of the decimal point. This is accomplished by entering the match string of characters that is the same as the string appearing in the peek attribute to be controlled.

Take note

Any number allowed

As many FORMAT commands as desired may be included in the INTRAN file. This allows a high degree of format control.

Order dependent

The order that FORMAT commands appear in the INTRAN file is important. The latest command in the INTRAN file overrides the previous command.

Example input

Example 1

Specify that all peek attributes associated with an element named PIPE1 are displayed with two digits to the right of the decimal point.

```
FORMAT PIPE1:* 2
```

Example 2

Specify that the pressure at the from-end of an element named PIPE1 are displayed with three digits to the right of the decimal point.

```
FORMAT PIPE1:P- 3
```

Note: If the FORMAT commands from Example 1 and Example 2 were both included in the INTRAN file (in the same order as of the examples), then the result is that all the peek attributes for PIPE1 have two digits to the right of the decimal point *except* for the P- pressure, which has three.

If the following is in the INTRAN file:

```
FORMAT PIPE1:P- 3
FORMAT PIPE1:* 2
```

Then the second command overrides the first one and all peek attributes for PIPE1 (including P-) have two digits to the right of the decimal point.

Example 3

Specify that all defined variables that start with TOTHP and end with one character (e.g., TOTHP1) are displayed with no digits to the right of the decimal point.

```
FORMAT TOTHP? 0
```

Example 4

Specify that the following peek values, P-, P+, PD, PK, Q-, Q+, for all of the elements in the model are displayed with two digits to the right of the decimal point.

```
FORMAT *:P? 2
FORMAT *:Q? 2
```

IF

INTRAN

```
IF(CONDITIONAL)
{COMMAND LIST}
/*or
IF(CONDITIONAL)
{COMMAND LIST1}
ELSE
{COMMAND LIST2}
/*or
IF(CONDITIONAL1)
{COMMAND LIST1}
ELSE IF(CONDITIONAL2)
{COMMAND LIST2}
ELSE IF(CONDITIONAL3)
{COMMAND LIST3}
...
ELSE
{COMMAND LISTn}
```

Field	Units Key	Description
CONDITIONAL	n/a	Relational test on numerical or status values
COMMAND LIST	n/a	See “Command list” on page 443.

Description

IF provides a way to conditionally execute one or another command list based on whether or not a certain specified condition is true. There are two forms of the IF command.

The first form of the IF command is actually an “if-then” construction, where the command list is executed only when the condition (described by CONDITIONAL) is true. That is, if the condition is true then the command list is executed. If the condition is false, the IF has no effect on TRANS.

The second form of the IF command is an “if-then-else” construction, where the first command list is executed if the condition is true. If the condition is false, the second command list is executed. For the second form of the IF, one of the two command lists is always executed. For more information, see “Command list” on page 443.

The condition can be one of the three following logical tests:

- A simple relational test such as “PIPE1:P+ ≤150.0”. “PIPE1:P+≤UNIT2:P-”, etc.
- A status test such as “UNIT1:ST = STARTING”, “VALVE1:ST = OPENED”. The status test keywords for pumps and compressors are = STARTING, STOPPING, RUNNING, and STOPPED. The status test keywords for valves are STOPPED, OPENING, CLOSING, OPENED, and CLOSED.
- A combination of two or more simple relational tests using and (&) and or (|). An example of a combined test is as follows:

```
"UNIT1:ST = STARTING & VALVE1:ST = CLOSED"
```

Take note

Where the IF command can be used

The IF command can be used either in a command list (for a WHENEVER or a DEFINE.SEQUENCE command) or on a stand-alone basis in the INTRAN file.

When the IF is used on a stand-alone basis (not in a command list), then the IF is evaluated only once at the start of the simulation (the first time step).

Nesting

IF statements may be nested.

Example input

Example 1

In a WHENEVER command list, examine the pressure of an external named DELIV and conditionally open a block valve named VLV1 if that pressure is greater than 300.

```
WHENEVER( PSTAT = 0 )
{IF( DELIV:P > 300 )
  {OPEN VLV1}
}
```

For this example, the IF statement is evaluated only when the WHENEVER conditional is true (i.e., PSTAT=0).

Example 2

As part of a defined sequence command, examine the pressure of an external named DELIV and either start a pump/compressor named UNIT1 if that pressure is greater than a specified value of 300; otherwise, stop another pump/compressor named UNIT2.

```
DEF.SEQ EXAMPLE_SEQ
{
  IF ( DELIV:P > 300 ) { START UNIT1 }
  ELSE { STOP UNIT2 }
}
```

The IF in the DEF.SEQ can also be entered in the INTRAN file as follows:

```
IF( DELIV:P > 300 )
{ START UNIT1 }
ELSE
{ STOP UNIT2 }
```

Example 3

At the start of a simulation, examine the pressure of an external named DELIV and either start a pump/compressor named UNIT1 if the pressure is below a specified value of 300, stop a pump/compressor named UNIT2 if the pressure is

above a specified value of 1000, open block valve B1 if the flow is less than 100, or close B1 if the flow is greater than 300.

```
IF ( DELIV:P < 300 )
{ START UNIT1 }
ELSE IF (DELIV:P > 1000 )
{ STOP UNIT2 }
ELSE IF (DELIV:NQ < 100)
{ OPEN B1}
ELSE IF (DELIV:NQ > 300)
{ CLOSE B1}
```

Example 4

At the start of a simulation, change the set point pressure of an external named INLET to a value equal to 500 if the NQ of an external named DELIV is either below 100 or above 300.

```
IF ( DELIV:NQ < 100 | DELIV:NQ > 300 )
{ POKE INLET:SP = 500 }
```

Input reference (I) (INTRAN)

INTRAN

```
/* time-programmed:
I NAME SCALE START

/* event-programmed:
I NAME KEY CNAME EVENTREF
```

Field	Units Key	Description
NAME	n/a	Name of the input reference being activated
SCALE	n/a	Time factor used to alter the time scale
START	TIME	Simulation time at which the input reference is activated. For more information on formats, see “Elapsed time versus clock format” on page 199.
KEY	n/a	Limit/response key (See the table in “Event-programmed” under “Take note”.)
CNAME	n/a	Name of a control element being monitored by this input reference. This may be the name of a controller, sensor, relay, or another input reference. Do not use a prefix (C, S, Y, or I) as part of the entry.
EVENTREF	n/a	The set point that triggers this input reference. A V with a space must precede the set point value. Alternately, this field may contain the name of another control element being monitored (without a prefix).

Description

The input reference (I) command is used to activate signal generation by means of an input reference (I) (see “Input reference (I) (INPREP)” on page 393). The input reference signal can be either time-programmed or event-programmed. Commands for a time-programmed input reference must be listed for each point in the simulation where the input reference is time-activated. The command for an event-programmed input reference is only listed once for each sensor and type of reference. It can activate itself, responding each time (or only the first time) the given conditions occur. Time and event programs may both be specified for the same input reference.

Take note

Time-programmed

START manually activates the input reference. SCALE can be used to adjust the time scale for the input reference. Input SCALE equal to 1 to keep the ratio 1:1. This command is used to override similar data entries made for the input reference in the equipment data in the INPREP file.

Event-programmed

The input reference can be programmed to monitor and respond to the output signal of a control element CNAME. The programmed response may be initiated based on either a fixed set point EVENTREF or a specified limit on the output

signal level KEY. The limit/response KEY may be designated by one of four available options shown in the following table:

Key		Limit Position	Response Frequency
UL	Respond to:	Upper Limit	once only
LL	Respond to:	Lower Limit	once only
UR	Respond to:	Upper Limit	as often as required
LR	Respond to:	Lower Limit	as often as required

These four options represent the combination of two separate choices: (1) Limit position (upper limit or lower limit) and (2) Response frequency (respond once only or respond as often as required).

Specifically, entering the keyword U on a time page beginning at time T0 sets the input reference to detect the first time following T0 at which the output of the control element CNAME exceeds the specified limit. If this event first occurs at time T1, then the input reference functions thereafter just as if the time-programmed command had been given with a start time of T1. Similarly, if the keyword had been LL, the input reference has been armed to detect the first time the output of the control element CNAME fell below the specified limit, and thereafter the function reverts to that of the time-programmed input reference with START=T1.

The UR and LR cases are similar in function to U and LL, respectively, but permit the input reference to reset and rearm. For example, if the keyword entered is LR, then T1 is taken as the first occurrence of the CNAME output dropping below the specified limit (either the input value or the value of the output of element CNAME). Thereafter the input reference outputs in accordance with its time function until it attains the upper limit of its function. Then if the output of CNAME remains below the lower limit (which of course can be variable), the output of the input reference remains at the upper limit of its function. However, should the output of CNAME ever exceed the lower limit, then the input reference is enabled to repeat the function by being rearmed to detect the event that the CNAME output might once again fall below the limit. As noted, the function with keyword UR is similar except that the designated limit is interpreted as an upper limit.

Entry of a time-programmed reference to an input reference that has previously been event-programmed causes the event-activation to be preempted.

Example input

Example 1 - Time-programmed

Activate an input reference named IR450 at time = 112.

```
I IR450 1.0 112.
```

Example 2 - Event-programmed

Have an input reference named IR451 monitor the output signal from a controller named PC1.F532 and activate whenever the signal voltage exceeds 1.23:

```
I IR451 UR PC1.F532  V 1.23
```

INTERACTIVE

INTRAN

```
INTERACTIVE [CRT] [,ROWS=ROWS, COLS=COLS] [, FONT=font], [fg=fgcolor],
[bg=bgcolor], [ffg=ffgcolor], [fbg=fbgcolor]
```

Field	Units Key	Description
CRT	n/a	Terminal type {MSWIN}.
ROWS	n/a	Number of rows in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
COLS	n/a	Number of columns in the CRT terminal display. **Notes:** Each installation has a default terminal display size (typically 24 rows x 80 columns). If supported by the CRT terminal, the ROWs and COLS commands can be used to specify other terminal display sizes for large displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
font	n/a	Font or font size to be used for displays. Entering new rows and columns values requires the CRT terminal type to be entered also.
fgcolor	n/a	Text foreground color for X-windows only. Foreground and background colors may be specified by one of two methods: • Specifying the English word for the color, i.e. red. Any color that requires spaces must be enclosed in two sets of quotation marks, i.e., ""medium violet red"". • Specifying the color bits, i.e., #RGB or #RRRGGGBBB.
bgcolor	n/a	Text background color for X-windows only. See the description for “fgcolor” above for more information.
ffgcolor	n/a	Figure foreground color for X-windows only. See the description for “fgcolor” above for more information.
fbgcolor	n/a	Figure background color for X-windows only. See the description for “fgcolor” above for more information.

Description

INTERACTIVE tells TRANS that the simulation will be performed interactively and will activate interactive features such as peek/poke, displays, the report, time plots, distance plots, etc.

Example input

Example 1

Run TRANS interactively using the default CRT terminal type and size.

```
INTERACTIVE
```

Example 2

Run a non-interactive TRANS run (command is commented out).

```
/*INTERACTIVE
```

LINE.FILL

INTRAN

Keyword value format:

```
LINE.FILL FROM TO,
+ FLUID F1 F2 ...  Fn,
+ VOLUME V1 V2 ...,
```

Table format:

```
LINE.FILL FROM TO,
+ TABLE FLUID VOLUME ID1 ID2 DRA.FRAC
+ Fi Vi id1i id2i fri
```

The following are alternate formats for LINE.FILL that are generated using SAVE.LINE.FILL for AGA, SCL, SCLPROP, BWRS and CNGA:

```
LINE.FILL pipe,
+ TABLE fluid1 ... fluidi
+ fluidj ... fluidn,
+ LOCATION loc1, FRAC frac1 ... fraci
+ fracj ...fracn,
+ ...
+ LOCATION loci, FRAC frac1 ...
```

SCLPROP equation of state outputs fluid properties of each fluid. The following table input applies:

```
LINE.FILL pipe,
+ FLUID fluid p0 t0 d0 pm0 tm0 ptmult ppmult ttmult
+ FRAC frac
[+ ID1 id1]
[+ ID2 id2]
[+ ID3 id3]
+ PRESSURE pres
+ TEMPERATURE temp
+ VISC [u0 vpmi vtmi] | [a b]
+ HCAP cp0 cpt
+ HVAL hval
+ HCOND k0 kt
[+ BING s0 spc stc]
+ VAPOR.PRESSURE vp1 ti vp2 t2 /* heat-of-vap int
[+ DRA.DR dr1 dr2 /* curve fit values are saved ]
[+ DRA.CON con1 con2]
[+ DRA.DROP pd]
[+ DRA.FRAC fr]
+ AGE age
+ LOCATION loc
```

The following table format is used with AGA, SCL, SCLPROP, BWRS, and CNGA equations of state. This example uses the CSV table input format. For more information on this format, see "CSV input syntax" on page 52.

```
LINE.FILL TABLE
{
PIPE, LOCATION, key1, key2, key3
 ....
}
```

Field	Units Key	Description
FROM	n/a	Beginning pipe name. If batches are specified in one pipe only, the same name is entered for both the FROM and TO pipes.
TO	n/a	Ending pipe name. If batches are specified in one pipe only, the same name is entered for both the FROM and TO pipes.
F_j	n/a	List of fluid names. Fluid names must be defined in the equation of state in the INPREP file. Fluid names should be input from the from-end to the to-end.
V_j	VOLUME	Volumes associated with the fluid names listed for F_j.
$id1_i$	n/a	Identifier ID1 for the fluid.
$id2_i$	n/a	Identifier ID2 for the fluid.
$fluid_j$	n/a	Available fluids or components.
loc_j	PIPE.LENGTH	Location where an interface between batches (FMVs) occurs.
$frac_j$	FRACTION	Corresponding fraction of the fluid for the batch.
g_j	n/a	Gas specific gravity for the batch.
key1, key2, and key3	—	Keywords corresponding to fluid names for SCL and BWRS, or fluid property names for AGA, SCLPROP, and CNGA. Refer to the individual equation of state sections for the keywords: • "STATE AGA" on page 221 • "STATE BWRS" on page 224 • "STATE CNGA" on page 228 • "STATE SCL" on page 230

Description

LINE.FILL allows you to initialize pipes with specified fluid names and/or properties in order to set up the alignment of batch interfaces. You may use this feature when using any of the following equations of state: AGA, SCL, SCLPROP, BWRS or CNGA.

The fluid names and properties used for LINE.FILL correspond to those specified for the model's equation of state. In addition to the fluid name and property keywords, you can use the location or volume keywords to specify the pipe location or volume for the specified fluid mixture vector (FMV).

Keywords may be entered using the keyword/value syntax as shown on the equation of state or the TABLE syntax.

For each pipe, any number of fluids are listed by name along with the volume of each of the fluids in the section of pipe.

Take note

How TRANS sets up alignment of batches

TRANS sets up the alignment of the batches on a proportional basis. The total pipe volume (PV_T) is calculated based on the pipes entered. The total fluid volume (V_T) is calculated by summing the volume inputs (V_n) of each of the fluids. The batch alignment is set up based on the calculated fluid volume (FV_n) of each fluid. The FV_n for each fluid is calculated by the following formula:

$$FV_n = PV_T \bullet (V_n / V_T)$$

If a LINE.FILL is used in a system with established slack line flow (LINE.FILL follows a LOAD.STATUS with slack line flow), the batch interfaces are adjusted to take into account the void fraction of the slack region.

Specific batch sizes

If specific batch sizes are desired, the total volume specified must exactly match the pipe volume. However, the volume numbers do not have to be exact. If the numbers are not exact, the following message is written to the OUTTRN file:

```
Pipe volume #########, fluid volume #########
```

LOAD.INPUT

LOAD.INPUT can be used to load batch information, or the LINE.FILL command may be included in INTRAN. The batch information can be saved using the SAVE.LINE.FILL command. For more information, see "LOAD.INPUT" on page 469 or "SAVE.LINE.FILL" on page 512.

This format includes an input for each pipe and then a table specifying the batches. There is a definition for each batch in the pipe. Batch volumes are calculated based on the locations and the operating temperature and pressure.

Example input

Example 1 (SCL and/or SCLPROP)

The LINE.FILL command is entered as follows. This example is for SCL and/or SCLPROP, with fluid names ONE, TWO, THREE, and FOUR.

```
LINE.FILL TL1 TL2,
+ FLUID  ONE  TWO  THREE  FOUR,
+ VOLUME  5   10   20   5
```

Assume the total pipe volume (PV_T) for the two pipes is 80. (This is calculated by TRANS.) The specified volumes (5 + 10 + 20 +5 = 40) are scaled to completely fill the pipes (volume 80).

Example 2 (AGA)

The following is an example of the LINE.FILL command for AGA. In this example, the names COST through NC4 are user-defined labels from the STATE AGA input in the INPREP file.

```
LINE.FILL TABLE,
```

```
{
PIPE, LOCATION, SG, CO2, HHV, COST, H2O, HS, C1, C2, C3, IC4, NC4
PIPE1, 0, 0.7, 0.01, 1100, 3, 0.05, 0.0003, 0.922, 0.04, 0.008, 0.02, 0.001
PIPE1, 10, 0.7, 0.01, 1100, 3, 0.05, 0.0003, 0.922, 0.04, 0.008, 0.02, 0.001
PIPE2, 0, 0.6518, 0.004, 1144.52, 2.25, 0.02, 0.0005, 0.9, 0.02, 0.04, 0.02,
0.01
PIPE2, 10, 0.6518, 0.004, 1144.52, 2.25, 0.02, 0.0005, 0.9, 0.02, 0.04,
0.02, 0.01
PIPE3, 0, 0.675, 0.00694922, 1122.64, 2., 0.03, 0.0004, 0.91, 0.029, 0.02,
0.02, 0.005
PIPE3, 25, 0.5896, 0.001, 1061.59, 1.48938, 0.08, 0.0001, 0.95, 0.03, 0.01,
0.004, 0.003
}
```

LOAD.INPUT

INTRAN

```
LOAD.INPUT FILENAME [, TIME = TIME] [,DT = INTERVAL]
```

Interactive

```
LOAD.INPUT FILENAME
```

Field	Units Key	Description
FILENAME	n/a	Name of the SAVE.LINE.FILL (optionally containing a peek attribute in [ ]) to be loaded
TIME	n/a	Simulation time to process the command. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval when the command will be processed (referenced back to time 0 rather than the BEGIN.TIME value)

Description

LOAD.INPUT loads RAMP or LINE.FILL commands from a file. The LINE.FILL commands may have been generated by the SAVE.LINE.FILL command. For more information, see "LINE.FILL" on page 465, "RAMP" on page 504, and "SAVE.LINE.FILL" on page 512.

Take note

Units

The units of the existing model are used. No units conversion occurs.

Fluids

All fluids and/or components in the saved file must be in the current model.

Pipes

Only those elements with matching names will be updated. Pipes that are in the model but are not in the saved file will remain unchanged.

Pressure profiles

When a previously saved batch alignment is loaded into a running model, there may be discrepancies in the resulting pressure profile due to changes in fluid density. This may cause node diagnostic flows for a couple of time steps to

adjust pressures. Take care in loading a previous state into a running model, especially if it is a liquid model with significant elevation changes.

Example input

Load the fluid information from a file called steady.sav:

```
LOAD.INPUT STEADY.SAV
```

LOAD.STATUS

INTRAN and Interactive

```
LOAD.STATUS file[,
+ SET.TIME = settime][,
+ DEVICES.MATCH = dev_names_include][,
+ DEVICES.EXCLUDE = dev_names_exclude][,
+ KEYLETTERS.MATCH = dev_kl_include][,
+ KEYLETTERS.EXCLUDE = dev_kl_exclude][,
+ SUBTYPES.MATCH = subtypes_include][,
+ SUBTYPES.EXCLUDE = subtypes_exclude][,
+ ECHO = echo]
```

Field	Units Key	Description
file	n/a	Name of an archive file or status file from which the status is loaded. For an interactive LOAD.STATUS, the extension may be omitted if it is ARK. In all other cases, the extension is required. If a full path is specified, it must be enclosed with quotation marks.
settime	TIME	Time value or expression to define the begin time of the simulation. For more information on formats, see "Elapsed time versus clock format" on page 199.
dev_names_include	n/a	List of devices to include, by name
dev_names_exclude	n/a	List of devices to exclude, by name
dev_kl_include	n/a	List of device types include, by keyletter
dev_kl_exclude	n/a	List of device types exclude, by keyletter
subtypes_include	n/a	List of device subtypes to include
subtypes_exclude	n/a	List of device subtypes to exclude
echo	n/a	Indicates whether to print to the OUTTRN file the name of every device in the ARK or STA file, and whether the information for that device is loaded or skipped.

Description

LOAD.STATUS initializes a TRANS run with data contained in an archive file. LOAD.STATUS attempts to change the conditions of the current simulation to match the archived status, as closely as possible.

You can specify the start time of the simulation, with the SET.TIME argument. If you do not specify a start time, SPS starts the simulation at the time when the archive was taken.

You may use LOAD.STATUS if the current model has been modified since the archive time. Any data that does not match, perhaps due to the addition or deletion of devices, is ignored. However, performing a LOAD.STATUS following model alterations could produce unexpected results.

The following information from the archive file is available for each device and overrides information specified in the INPREP file, for any device not excluded by the LOAD.STATUS command:

Device	Overriding information from archive file
Pipes	friction, pipe end temperatures, ground temperatures (TRANSTHERMAL only), elapsed time since pressure was low, mass inventory, pressures, velocities, composition, flowing temperature, wall temperature, wrap temperature and wax buildup.
Pumps	run status, power, speed, head, suction block status, discharge block status, suction check status, discharge check status, bypass check status, pump sequence status, time since start, time since stop, suction and discharge valve: open time, close time, travel time and current valve coefficient, flow, suction pressure, discharge pressure, temperature, maximum (rated) driver power, maximum (rated) speed constraint, and stages.
Tanks	Volume, height, temperature, and fluid.
Centrifugal compressors (KC)	compressor speed, status, time left for set point change, new set point target, number of units running, flow resistance, ambient temperature, check valve status, surge control status, stonewall control status, maximum power constraint status, effective maximum speed constraint status, maximum rate of speed increase, maximum rate of speed decrease, discharge pressure set point status, suction pressure set point status, flow set point status, minimum speed trip status, maximum discharge temperature trip status, discharge pressure set point interval, speed target, speed interval, discharge pressure set point, suction pressure set point, flow set point, polytropic index (if BWRS equation of state), number running target, surge constraint control used, surge constraint type bound on control, number of constraint mode changes, pressures, flows, recycle flow, head, temperatures, heat rate
Idealized compressor	current power target, rated power, mechanical efficiency, power, speed, head, status, pressures, flow, temperatures, heat rate
Reciprocating compressors	status, pocket mode, last pocket, run key, accumulated bad time, rated rpm, rated power, minimum load, maximum load, pocket number, swept volume, number of units running, mechanical efficiency, time left for speed change, speed target, time factor for underload or overload conditions, capacity correction factor, gas polytropic index (if BWRS equation of state), CT factor, number of control mode changes, pressures, flows, temperatures, heat rate
Theoretical horsepower-flow compressor	status, check valve control status, maximum rate of power increase status, maximum rate of power decrease status, maximum power constraint status, minimum power constraint status, power set point status, maximum discharge pressure constraint status, discharge pressure set point status, minimum suction pressure constraint status, suction pressure set point status, maximum flow constraint status, minimum flow constraint status, flow set point status, maximum ratio constraint status, minimum ratio constraint status, ratio set point status, run status, power equation, rated power, K1, K2, K3, efficiency, polytropic exponent, fuel usage ratio, maximum power, minimum power, power set point, maximum discharge pressure, discharge pressure set point, minimum suction pressure, suction pressure set point, maximum flow, minimum flow, flow set point, maximum ratio, minimum ratio, ratio set point, number running, temperature rise ratio, number of control mode changes, pressures, flows, temperatures, heat rate

Device	Overriding information from archive file
Control valve	check valve status, minimum valve constant, maximum valve constant, fraction, pressure, flows, temperatures
Idealized control valve	check valve status, minimum valve constant, maximum valve constant, maximum rate of increase for valve position, maximum rate of decrease for valve position, downstream pressure set point status, upstream pressure set point status, flow set point status, travel time, fraction open set point, downstream set point, upstream set point, flow set point, maximum fraction open constraint, number of control mode changes, pressures, flows, temperatures, fraction
Block valve	block valve movement status :SCV, valve status :ST, check valve status, cumulative flow :CF, open time, close time, valve coefficient, open valve coefficient :CVO, closed valve coefficient :CVC, pressure :P- :P+, flows :Q- :Q+, temperatures :T- :T+; initial fraction open :SFR
G887	cumulative flow, A1, B, C1, lowest value of sleeve pressure, highest value of sleeve pressure, A2, C2, pressures, flows, temperatures
Header	cumulative flow, if a heater header outlet temperature or header delta temperature, pressures, flows, duty
Heat exchanger	cumulative flow, pressures, flows, duty
Externals/nodes	current value of flow if flow controlled, current value of pressure if pressure controlled, temperature, set point pressure or flow, temperature set point, for BWRS or SCL fluid set points, number of control mode changes
Air valve controller	current air mass in system, liquid mass in system, maximum air mass set point
Surge tank	pressure of gas, volume of gas, height of liquid in tank, tank flow
Controller	gain, integral, set point, normalization constant, bias, reset time, derivative multiplier, low bound, high bound, error, in, out
Sensor	input, output, period
Actuator	old dt, current output period dt, in, out
Relays	
- noise	seed value, scale, period, pokable input, output, feedback status flag
- HI/LO select	set point, high limit, low limit, feedback status flag, output
- switch	set point, source1, source2, delay time, end delay, output, feedback status flag
- ramp	ramp rate, bias, pokable input, output, feedback status flag
- deriv	rate, bias, pokable input, output, feedback status flag
- feedback	five inputs, hi source signal, low source signal, output, status flag
- multiply	seven multipliers, output, feedback status flag
- integrator	timeliest, initial, pokable input, output, feedback status flag
- average	period, pokable input, output, feedback status flag

Currently, LOAD.STATUS does not consider the following:

- Column-separation
- Differing equations of state
- Differing fluid components
- Differing phases (gas in one model, liquid in the other)
- Differing base or custody conditions

Take note

Devices

LOAD.STATUS operates on a device name/device-type basis. Only devices that have the same name and type in the two archive files are set in the current simulation.

LOAD.STATUS is only minimally aware of subtypes. For example, if a device is a reciprocating compressor in one model and a polytropic compressor in the other model, the results may be unpredictable. For models with varying subtypes, it is recommended that you use LOAD.STATUS interactively.

Cascading effect of conditions

Many of the arguments in a LOAD.STATUS command are used set conditions that determine precisely which devices should be restored to the archived state. In most cases, you may not want to exclude any devices, and will not use these arguments.

If you do use these arguments, they function in a cascading manner, each further narrowing the device “list” as altered by the previous argument. For example, consider the following POKEALL statement:

```
LOAD.STATUS STEADY_STATE.ARK
+ DEVICES.MATCH = *P*
+ KEYLETTERS.MATCH = E
```

The initial LOAD.STATUS command assumes the inclusion of all devices in the model. The DEVICES.MATCH argument narrows this list to only devices that contain a “P” in their names. The KEY.LETTER.MATCH argument further narrows the list to include externals only.

Using LOAD.STATUS to reset a simulation interactively

If an archive was created during the current simulation, LOAD.STATUS can be used interactively to reset the state of the system to the archived state, rather than halting and restarting the simulation. Each time the status is loaded, it has the effect of resetting the simulation to the state of the archived file, and the REVIEW file is deleted and recreated. This may be useful when experimenting with different modeling scenarios and you are not sure how the system will respond.

After a load status is completed, TRANS evaluates the POKEs and SUBMIT.SEQUENCES and disregards those whose execution times are earlier than the time associated with the loaded status.

Behavior of command lists in progress at the time of a LOAD.STATUS

If a LOAD.STATUS is issued when a WHENEVER or SUBMIT.SEQUENCE is in progress, unexpected results may occur if commands are processing at WAIT and/or WAIT.UNTIL commands. Following a LOAD.STATUS event, the following actions may occur in a command list, as applicable:

Condition	Action
A WHENEVER blocked at a WAIT statement	Continues execution immediately, at the same location in the command list
A WHENEVER blocked at a WAIT.UNTIL statement	Continues to reevaluate the statement
A SEQUENCE blocked at a WAIT.UNTIL	The statement will either be deleted, or continue execution at the WAIT.UNTIL.

Time expression

The time expression can be any valid expression in TRANS, such as 2 + 2.5. If the expression given is invalid or contains undefined variables, TRANS produces an error and starts the archived simulation at the time of the archive creation.

LOAD.STATUS from different architectures

Archive files and/or status files created on one architecture can be loaded into a model running on another architecture.

Example input

Example 1

Initialize a TRANS run with a previously created archive file named STEADY_STATE.ARK.

```
LOAD.STATUS STEADY_STATE.ARK
```

Tip: For an interactive command, the ARK extension is optional.

Example 2

An archive of the model was taken at time = 4 minutes and was written to TIME4.ARK. To load the archived state and reset the time to 1 minute, enter one of the following commands.

```
LOAD.STATUS TIME4.ARK, SET.TIME = 1
LOAD.STATUS TIME4.ARK, SET.TIME = 3.5 - 2.5
```

Example 3

Load an archive file name ARK1.ARK and set the simulation time to December 31, 2002.

```
LOAD.STATUS ARK1.ARK, SET.TIME=TIMEVALUE("02/12/31 00:00:00")
```

The TIMEVALUE function is used if you want to specify the date and time in clock format.

LOAD.STEADY

INTRAN and Interactive

```
LOAD.STEADY [STEADY.FILE=name1] [,STEADY.FILE=name2][,
+ ECHO=echo][,
+ WARN.EXTRA=extra][,
+ WARN.MISSING=missing][,
+ WARN.TOLERANCE=warn_tol][,
+ ABSOLUTE.TOLERANCE=general_abs][,
+ RELATIVE.TOLERANCE =general_rel][,
+ ABSOLUTE.TOLERANCE.PRESSURE=pres_abs][,
+ RELATIVE.TOLERANCE.PRESSURE=pres_rel][,
+ ABSOLUTE.TOLERANCE.FLOW=flow_abs][,
+ RELATIVE.TOLERANCE.FLOW=flow_rel][,
+ LOG.FILE[.APPEND]=logname][,
+ BALANCE=balance]
```

Field	Units Key	Description
name	n/a	Name(s) of the steady state file(s) you want to load
echo	n/a	YES echoes the input to the log. NO does not echo the input. {NO}
extra	n/a	YES warns data in the steady state file that is not in the SPS model. NO does not warn this data. {NO}
missing	n/a	YES warns data in the SPS model that is not in the steady state file. NO does not warn the data. {NO}
warn_tol	n/a	YES warns balance results that are out of tolerance with the values in the steady state file. NO does not warn the results. {YES}
general_abs	n/a	Absolute tolerance used for all values, but may be overridden for pressures and/or flows. This value is in the user units for whatever value is being tested.
general_rel	n/a	Relative tolerance used for all values, but may be overridden for pressures and/or flows
pres_abs	PRESSURE	Absolute tolerance used for pressures
pres_rel	n/a	Relative tolerance used for pressures
flow_abs	FLOW	Absolute tolerance used for flows
flow_rel	n/a	Relative tolerance used for flows
balance	n/a	YES indicates a balance should occur after the steady state file is loaded. NO indicates a balance should not occur. {YES}
logname	n/a	Name of the log file

Description

LOAD.STEADY allows you to load in a previously saved steady state file and then rebalance the simulation. The steady state file (usually file type STS) may have been generated by:

- SynerGEE
- SPS SAVE.STEADY

In either case, the steady state file may be edited by hand before loading into SPS. The steady state file is formatted as a comma-separated value (CSV) file so that it can be edited using either Microsoft Excel or a text editor.

Immediately after loading the file, you will usually want SPS to perform a balance. However, if you load several different steady state files in succession (perhaps they initialize different parts of your model), then you will probably want to specify BALANCE=NO on all files except the last file.

Because of differences in models, modeling software, and possible inconsistencies from hand edits, the model may not balance perfectly after loading. This is generally acceptable because SPS is inherently a transient model; small “bumps” due to the various inconsistencies usually settle out quickly.

Additionally, several mechanisms help identify the various bumps that come :

- Message destinations
- Tolerances
- Message destinations
- Extra values
- Missing values

Missing values

All of the various balance-related messages automatically go to the OUTTRN file. They will also go to the “logname” file if it is specified on the LOG.FILE[.APPEND] keyword.

Tolerances

There are two types of tolerances: absolute and relative. These tolerances are used to compare absolute and relative differences, respectively.

If x is a value from the steady state file and y is the corresponding value after SPS has balanced, then you can define absolute difference as:

$$\text{adiff} = |\, x - y \,|$$

Relative difference is:

$$\text{rdiff} = \begin{cases} 0 \quad \text{if both } x \text{ and } y \text{ are } 0 \\ \dfrac{|\, x - y \,|}{\max(|\, x \,|, |\, y \,|)} \quad \text{otherwise} \end{cases}$$

For each value loaded from the steady state file, SPS calculates adiff and rdiff, and then compares these against the appropriate tolerance.

adiff is compared to:

- pres_abs for pressures
- flow_abs for flows

- general_abs for everything else

Likewise, rdiff is compared to:

- pres_rel for pressures
- flow_rel for flows
- general_rel for everything else

If both adiff and rdiff are larger than the corresponding tolerance, then the corresponding value is out of tolerance and SPS writes a message.

You may choose to change tolerances in SPS using the SET.LIMIT command.

Extra Values

The steady state file may contain attributes for one or more elements that SPS does not expect. WARN.EXTRA controls whether these extra attributes should be warned or quietly skipped.

Missing Values

The steady state file may be missing some attributes for one or more elements that SPS expects. WARN.MISSING controls whether these missing attributes should be warned or quietly skipped.

Continue Simulation

After loading one or more steady state files and balancing, the simulation can proceed from the newly calculated steady state. The simulation time is held constant during the balance.

Supported Features

The isothermal mode and the AGA equation of state are supported.

MAXMIN

INTRAN

```
MAXMIN name[,
+ DEVICES.MATCH = dev_name_include][,
+ DEVICE.EXCLUDE = dev_name_exclude][,
+ KEYLETTERS.MATCH = dev_kl_include][,
+ KEYLETTERS.EXCLUDE = dev_kl_exclude][,
+ SUBTYPES.MATCH = subtypes_include][,
+ SUBTYPES.EXCLUDE = subtypes_exclude][,
+ PRINT.PERIOD = period][,
+ ECHO = echo][,
+ TOLERANCE = tolerance]
```

Field	Units Key	Description
name	n/a	Name of the MAXMIN "device." If you want to produce a MAXMIN device for each matching device name, enter `%1%`.
dev_name_include	n/a	Devices to include, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_name_exclude	n/a	Devices to exclude, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See "Device keyletters" on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See "Device keyletters" on page 47.
subtypes_include	n/a	Device subtypes to include (STYP attribute values). For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
subtypes_exclude	n/a	Device subtypes to exclude (STYP attribute values). For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
period	minutes	Time interval for printing the MAXMIN report to OUTTRN.
echo	n/a	Indicates whether to produce a list in the OUTTRN file of all elements that are included in this MAXMIN device. Valid values are YES and NO.
tolerance	ΔPRESSURE	The tolerance by which a current value must be exceeded before it is replaced. The tolerance can be used to prevent unnecessary fluctuation of MAXMIN values due to unimportant pressure changes.

Description

MAXMIN is used to monitor maximum and/or minimum pressures with regards to location and time, within a specified collection of elements. This collection of elements is defined by the MAXMIN command arguments, which function as

conditional parameters to build the collection. Generally, you should use wildcards to form the conditions that define this collection. For more information, see "Wildcards" on page 50.

When you use the MAXMIN command, you are essentially creating a new "device" in the model, whose attributes represent the identities, pressures, and other characteristics of the individual elements in the collection. This device is not an actual piece of equipment that integrates with the model. Rather, it is a reporting mechanism whose primary purpose is to maintain values related to other elements in the collection based on maximum and minimum criteria. A MAXMIN device is not a part of the model topology and has no effect on the hydraulics of the model.

As an example, the following command creates a MAXMIN device named "MYCOLLECTION":

```
MAXMIN MYCOLLECTION
```

Using the MYCOLLECTION device example, the following peek name represents the name of the element in the MYCOLLECTION collection that has experienced the highest actual pressure during the current step:

```
MYCOLLECTION:PMAX.IDEV
```

The following peek name represents the maximum pressure value experienced by any element within the collection, since the creation or most recent reset of the MAXMIN device:

```
MYCOLLECTION:PMAX.HVAL
```

The MAXMIN device maintains information on pressure maximums/minimums during the current time step, as well as historical pressure maximums/minimums since the creation of the device, or the most recent reset. When you reset a MAXMIN device, all historical data is discarded and historical maximums/minimums are maintained from that moment forward. A reset is performed by poking the RESET attribute of the MAXMIN device. For example:

```
POKE MYCOLLECTION:RESET = YES
```

Attributes

A MAXMIN device provides attributes that represent names, pressures, locations, and other characteristics of elements within the applicable collection.

Single value (ungrouped) attributes

The following MAXMIN attributes do not belong to groups and can form peek names without any additional group syntax:

Attribute	Pokable	Units	Description
NDEV	No	none	Number of individual elements in the MAXMIN collection
RESET	Yes	none	When poked to Yes, resets the recording of historical pressure information to current moment.
SINCE	No	TIME	Time of the most recent reset. If the MAXMIN device has not been reset since its creation, a peek name with this attribute will return the time of MAXMIN device creation, often zero (0).

As an example, the following command resets the MAXMIN device MYCOLLECTION:

```
POKE MYCOLLECTION:RESET = YES
```

Grouped attributes

Most MAXMIN attributes belong to groups, and you must specify the appropriate group when forming peek names. For example, the PMAX group represents a collection of maximum pressure-related attributes, some of which include:

- The highest pressure experienced during the current time step; for example:

  `MYCOLLECTION:PMAX.IVAL`

- The name of the element which experienced the highest pressure during the current time step; for example:

  `MYCOLLECTION:PMAX.IDEV`

- The location on the element that experienced the highest pressure during the current time step; for example:

  `MYCOLLECTION:PMAX.ILOC`

The following table lists the MAXMIN attribute groups:

Group	Attributes	Units	Description
PMAX	IVAL	PRESSURE	Highest pressure experienced by any element in the collection, during the current time step
	IDEV	none	Name of the element that experienced the highest pressure, represented by IVAL
	ILOC	none	Specific location where the highest pressure took place on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	HVAL	PRESSURE	Historically maximum pressure experienced by any element in the collection, since the creation of the MAXMIN device or the most recent reset.
	HDEV	none	Name of the element that experienced the historically maximum pressure, represented by HVAL
	HLOC	none	Specific location where the historically maximum pressure occurred on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the historically maximum pressure, HVAL, occurred.

Group	Attributes	Units	Description
PMIN	IVAL	PRESSURE	Lowest pressure experienced by any element in the collection, during the current time step
	IDEV	none	Name of the element that experienced the lowest pressure, represented by IVAL
	ILOC	none	Specific location where the lowest pressure took place on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note"
	HVAL	PRESSURE	Historically minimum pressure experienced by any element in the collection, since the creation of the MAXMIN device or the most recent reset.
	HDEV	none	Name of the element that experienced the historically minimum pressure, represented by HVAL
	HLOC	none	Specific location where the historically minimum pressure occurred on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the historically minimum pressure, HVAL, occurred.

Group	Attributes	Units	Description
MAOP	IDIF	PRESSURE	For all elements in the collection, the minimum value of the applicable MAOP minus the maximum pressure experienced, during the current time step. That is: MAOP.IDIF = min(MAOP - pressure) ...where the difference is calculated for each pressure of each element in the collection, using the applicable MAOP at that location.
	IDEV	none	Name of the element that came the closest to its MAOP (or exceeded it the most, as applicable), according to the value represented by IDIF
	ILOC	none	Specific location where the IDIF value was recorded on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	HDIF	PRESSURE	For all elements in the collection, the minimum value of the applicable MAOP minus the maximum pressure experienced, since the creation of the MAXMIN device or the most recent reset. That is: MAOP.HDIF = min(MAOP - pressure) ...where the difference is calculated for each pressure of each element in the collection, using the applicable MAOP at that location.
	HDEV	none	Name of the element that came the closest to its MAOP (or exceeded it the most, as applicable), according to the value represented by HDIF
	HLOC	none	Specific location where the HDIF value was recorded on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the HDIF value was recorded on HDEV.

Group	Attributes	Units	Description
LAOP	IDIF	PRESSURE	For all elements in the collection, the minimum value of the minimum pressure experienced minus the applicable LAOP, during the current time step. That is: LAOP.IDIF = min(pressure - LAOP) ...where the difference is calculated for each pressure of each element in the collection, using the applicable LAOP at that location.
	IDEV	none	Name of the element that came the closest to its LAOP (or exceeded it the most, as applicable), according to the value represented by IDIF
	ILOC	none	Specific location where the IDIF value was recorded on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	HDIF	PRESSURE	For all elements in the collection, the minimum value of the minimum pressure experienced minus the applicable LAOP, since the creation of the MAXMIN device or the most recent reset. That is: LAOP.HDIF = min(pressure - LAOP) ...where the difference is calculated for each pressure of each element in the collection, using the applicable LAOP at that location.
	HDEV	none	Name of the element that came the closest to its LAOP (or exceeded it the most, as applicable), according to the value represented by HDIF
	HLOC	none	Specific location where the HDIF value was recorded on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the HDIF value was recorded on HDEV.

Group	Attributes	Units	Description
MASP	IDIF	PRESSURE	For all elements in the collection, the minimum value of the applicable MASP minus the maximum pressure experienced, during the current time step. That is: MASP.IDIF = min(MASP - pressure) ...where the difference is calculated for each pressure of each element in the collection, using the applicable MASP at that location.
	IDEV	none	Name of the element that came the closest to its MASP (or exceeded it the most, as applicable), according to the value represented by IDIF
	ILOC	none	Specific location where the IDIF value was recorded on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	HDIF	PRESSURE	For all elements in the collection, the minimum value of the applicable MASP minus the maximum pressure experienced, since the creation of the MAXMIN device or the most recent reset. That is: MASP.HDIF = min(MASP - pressure) ...where the difference is calculated for each pressure of each element in the collection, using the applicable MASP at that location.
	HDEV	none	Name of the element that came the closest to its MASP (or exceeded it the most, as applicable), according to the to value represented by HDIF
	HLOC	none	Specific location where the HDIF value was recorded on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the HDIF value was recorded on HDEV.

Group	Attributes	Units	Description
LASP	IDIF	PRESSURE	For all elements in the collection, the minimum value of the minimum pressure experienced minus the applicable LASP, during the current time step. That is: LASP.IDIF = min(pressure - LASP) ...where the difference is calculated for each pressure of each element in the collection, using the applicable LASP at that location.
	IDEV	none	Name of the element that came the closest to its LASP (or exceeded it the most, as applicable), according to the value represented by IDIF
	ILOC	none	Specific location where the IDIF value was recorded on the element represented by IDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	HDIF	PRESSURE	For all elements in the collection, the minimum value of the minimum pressure experienced minus the applicable LASP, since the creation of the MAXMIN device or the most recent reset. That is: LASP.HDIF = min(pressure - LASP) ...where the difference is calculated for each pressure of each element in the collection, using the applicable LASP at that location.
	HDEV	none	Name of the element that came the closest to its LASP (or exceeded it the most, as applicable), according to the value represented by HDIF
	HLOC	none	Specific location where the HDIF value was recorded on the element represented by HDEV, as a percentage of its length. For more information, see "Specific location of reported pressures (ILOC/HLOC)" under "Take note".
	TIME	TIME	The time that the HDIF value was recorded on HDEV.

Note: Attributes that begin with an "I" are Instantaneous; that is, their values represent conditions at the current moment. Attributes that begin with an "H" are Historical; that is, their values represent recorded values from past events that satisfied the applicable criteria.

Take note

Specific location of reported pressures (ILOC/HLOC)

Many of the attributes of a MAXMIN device include ILOC and HLOC values, each of which specifies the precise location where the applicable pressure was experienced. This type of value indicates a percentage, represented by a fractional equivalent, of the length of the element in question. In turn, this length indicates the distance from the to-end of the element that the measurement occurred.

For example, assume that you know the following information, as related to the MAXMIN device MYCOLLECTION:

MYCOLLECTION:PMAX.IDEV = PIPE1

MYCOLLECTION:PMAX.ILOC = .25

PIPE1:LEN = 10 (miles)

In this situation, the pressure represented by MYCOLLECTION.PMAX.IVAL (not shown) occurred 2.5 miles from the to-end of the PIPE1 transfer line.

Negative values for IDIF/HDIF

The values of IDIF and HDIF attributes are represented by a simple difference between pressures and specified limits. For example, the attribute MAOP.IDIF is represented by the lowest value of:

min(MAOP - pressure)

...during the current time step, for all pressures of all elements in the MAXMIN collection, using MAOPs at the respective locations. The MAOP attribute group, therefore, allows you to find out which element in the MAXMIN collection came the closest to its MAOP, and by how much.

If the pressure for any element in the MAXMIN collection exceeded its MAOP during the current time step, the value for :MAOP.IDIF will be negative. Through a similar logic, the IDIF and HDIF values for :LAOP, :MASP, and :LASP can also be negative. If one of these values is negative, you can view it as a positive number to see how much associated limit was exceeded.

If no limits are exceeded, IDIF and HDIF values are always positive.

Whole-system MAXMIN device

By default, SPS creates a system-wide MAXMIN device when INTRAN is initiated, named MM_SYSTEM. This MAXMIN device includes all hydraulic elements in the model, and can be used like any other MAXMIN device. For example, the following peek name represents the total number of hydraulic elements in the current model:

```
MM_SYSTEM:NDEV
```

Cascading effect of conditions

Most of the arguments in a MAXMIN command are used to set conditions that determine which elements should be included in the MAXMIN collection. These arguments function in a cascading manner, each further narrowing the element “list” as altered by the previous argument. For example, consider the following MAXMIN statement:

```
MAXMIN MYCOLLECTION,
+ DEVICES.MATCH = *P*,
+ KEYLETTER.MATCH = T
```

In this case, the MAXMIN * part of the command names the device “MYCOLLECTION.” The DEVICES.MATCH argument narrows the collection to elements with a “P” in their names. The KEYLETTER.MATCH argument further narrows the collection to transfer lines only.

Not interactive

The MAXMIN command may not be used interactively. Like most peek names, though, all peek names of an existing MAXMIN device may be accessed interactively.

Example input

Example 1

Create a MAXMIN device named MYCOLLECTION, which includes all transfer lines in the model except those that begin with a “T”:

```
MAXMIN MYCOLLECTION,
+ DEVICES.EXCLUDE = T*,
+ KEYLETTER.MATCH = T
```

Example 2

The following example will produce a MAXMIN device MM_<devicename> every 60 minutes for each device matching the specified criteria.

```
MAXMIN MM_%1%,
+ KEYLETTER.MATCH = T,
+ PRINT.PERIOD 60
```

OPEN

INTRAN

```
OPEN NAME [, TIME = T1 T2 ... Tn] [, DT = INTERVAL] [, KEEP.OLD=YES]
```

Interactive

```
OPEN NAME [, TIME = TIME] [, DELAY = DELAY]
```

Field	Units Key	Description
NAME	n/a	Name of a block valve to be opened
T_j	TIME	Simulation time or times at which the block valve starts to open. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the block valve starts to open (referenced back to time 0 rather than the BEGIN.TIME value)
TIME	TIME	Simulation time at which the block valve starts to open. For more information on formats, see "Elapsed time versus clock format" on page 199.
DELAY	TIME	Time from the current simulation time at which the block valve starts to open. Avoid using with models running in TPORT.

Description

OPEN causes a block valve to start opening. The time at which the valve starts to open depends on which time option is used when the command is entered. Different time options are available in the INTRAN file and interactively.

INTRAN

Command setup	Effect
OPEN is by itself and does not target a specific time	Begins to open the valve when the simulation begins
OPEN is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Begins to open the valve when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
OPEN targets a specific time, multiple times, or a time interval	Begins to open the valve at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

Command setup	Effect
OPEN does not target a time	Begins to open the valve at the beginning of the next time step of the simulation
OPEN targets a specific time	Begins to open the valve at the specified time
OPEN specifies a delay	Begins to open the valve after the delay period expires

Take note

Check valves and control valves

The OPEN command does not work for check valves, idealized control valves, or non-idealized control valves. Check valves open automatically based on the pressure differential across the valve. Idealized control valves open due to pressure, flow, or fraction set points. Non-idealized control valves open in response to an actuator's output signal.

Command may be overridden

A STOP command or a CLOSE command overrides an OPEN command if either of these is given after the OPEN command.

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

KEEP.OLD feature—INTRAN

You may enable "old" actions to occur by adding the KEEP.OLD = YES statement to the OPEN command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 60
OPEN VALVE1, TIME = 90
OPEN VALVE2, TIME = 30
OPEN VALVE3, TIME = 0
```

Since the simulation starts at time = 60, the "OPEN VALVE2" command will not execute, since its execution time has already passed at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the OPEN command execute anyway. For example, the following OPEN would occur at a simulation start time = 60, despite its specified time:

```
OPEN VALVE2, TIME = 30, KEEP.OLD = YES
```

In the sample, note that the "OPEN VALVE3" command will execute regardless of the simulation start time, because the parameter TIME = 0 indicates that the OPEN command should be executed at start time.

Example input

Example 1

Start opening a valve named VALVE1 at the current simulation time (when the next time step is taken).

```
OPEN VALVE1
```

Example 2

Start opening a valve named VALVE1 at a simulation time of 4 hours (240 minutes).

```
OPEN VALVE1, TIME = 240
```

Example 3

Interactively start opening a valve named VALVE1 10 minutes from the current simulation time.

```
OPEN VALVE1, DELAY = 10
```

Example 4

Start opening a valve named VALVE1 at 1:40 p.m. on November 6, 2002.

INTRAN:

```
OPEN VALVE1, TIME = "02/11/06 13:40:00"
```

OUTLIST

Description

OUTLIST is obsolete. Use “MAXMIN” on page 479.

POKE

Note: A POKE command may also be issued in the INPREP file for an individual device. For more information, see "POKE and RAMP" on page 423.

INTRAN

```
POKE peek_name = val_intran [, TIME=t1 ... tn] [, DT=int] [, KEEP.OLD=YES]
```

Interactive

```
POKE peek_name = val_inter [, TIME=time] [, DELAY=delay]
```

Field	Units Key	Description
peek_name	n/a	Peek name to be poked, in the form of {device name}:{attribute}
val_intran	n/a	New value for the peek name; that is, the value to be poked. Arithmetic expressions and other peek names are valid. See "Expressions, Operators, and Functions" on page 587 and "Peek and Poke Keyletters and Attributes" on page 639.
t_j	TIME	Simulation time or times when the poke is to be executed. For more information on formats, see "Elapsed time versus clock format" on page 199.
int	TIME	Time interval when the poke is to be executed (referenced back to time 0 rather than the BEGIN.TIME value)
val_inter	n/a	New value for the peek name; that is, the value to be poked. Other peek names and defined variables are valid, but expressions are not.
time	TIME	Simulation time when the poke is to be executed. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
delay	TIME	Time from the current simulation time when the poke is to be executed. Avoid using with models running in TPORT.

Description

The POKE command may be used to change the value of any pokable peek name during the course of the simulation. A number of device-related attributes (such as a controller set point) as well as simulation parameters are pokable. All user-defined variables are also pokable. "Peek and Poke Keyletters and Attributes" on page 639 contains a list of the available poke keywords.

A POKE command forces the time step to its minimum value. To change a peek name value without changing the time step, use SET instead. See "SET" on page 518. In addition, if you want to have the variable change over the course of the simulation, you would want to use the RAMP command rather than POKE. See "RAMP" on page 504 or "POKE and RAMP" on page 423.

Any numbers in the value to be poked are assumed to be in user-defined units.

The time at which the POKE command executes depends on the time setup in the command, and the command's location in the file. INTRAN and Interactive modes react differently to different POKE time setups, as follows:

INTRAN

POKE command setup	Effect
POKE is by itself and does not target a specific time	Changes the value when the simulation begins
POKE is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Changes the value when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
POKE targets a specific time, multiple times, or a time interval	Changes the value at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

POKE command setup	Effect
POKE does not target a time	Changes the value at the beginning of the next time step of the simulation
POKE targets a specific time	Changes the value at the specified time
POKE specifies a delay	Changes the value after the delay period expires

Take note

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

IF.EXISTS feature—INTRAN

You can use IF.EXISTS within a POKE command to avoid an error if the target device is not in the model. For example, if CTL1 does not exist, the following POKE command would not cause an error:

```
POKE IF.EXISTS(CTL1:SP) = 50, TIME = 10
```

IF.EXISTS is most useful if an INTRAN file is used by several models. When using IF.EXISTS, note the following:

- No spaces are permitted within the parentheses. For example, IF.EXISTS(TOTPWR:VAL) is invalid.
- If the device exists but the attribute is not pokable, an error occurs.
- If the device exists but the target value does not, an error occurs. For example, if YYY does not exist, the following is invalid, regardless of XXX:

```
POKE IF.EXISTS(XXX:VAL) = YYY
```

- If there are any syntax errors elsewhere in the POKE command, an error occurs.
- The poke still occupies space in IF/ELSE logic. Consider the following example:

```
IF (ZZZ:VAL = 2)
{
POKE IF.EXISTS(B1:ST) = OPEN, TIME = 5
POKE QQQ:VAL = 10
}
```

Even if the model does not have a device named B1, this sample *does not* become the following:

```
IF (ZZZ:VAL = 2)
POKE QQQ:VAL = 10
```

KEEP.OLD feature—INTRAN

You may enable “old” actions to occur by adding the KEEP.OLD = YES statement to the POKE command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 5
DEFINE A = 0
DEFINE B = 0
DEFINE C = 0
POKE A = 10, TIME = 10
POKE B = 10, TIME = 1
POKE C = 10, TIME = 0
```

Since the simulation starts at time = 5, the “B” poke will not execute, since the execution time is already past at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the “B” poke execute anyway. For example, the following poke would occur at a simulation start time = 5, despite its specified time:

```
POKE B = 10, TIME = 1, KEEP.OLD = YES
```

In the sample, note that the “C” poke executes regardless of the simulation start time, because the parameter TIME = 0 indicates that the POKE command should be executed at start time.

Interaction with RAMPS

If you poke a peek name that is being ramped, the peek name is updated by the poke and the ramp is disabled.

Expression evaluation - INTRAN

Use caution when performing a time-dependent poke with a non-numeric expression, because the value that is poked may not be as expected. POKEs and associated expressions are evaluated when they are read, which is prior to the first time step unless they are part of a WHENEVER or DEFINE.SEQUENCE.

For example, assume that you want to change ABC to the value of PIPE1:P+ at time = 100. Consider the following command:

```
POKE ABC = PIPE1:P+, TIME = 100
```

The value of ABC is changed at time = 100, but it is changed to the value that PIPE1:P+ was at initialization time, not at time = 100. To avoid this situation, you must put the POKE command in a WHENEVER or a DEFINE.SEQUENCE that is evaluated at time = 100.

Example input

Example 1

Poke the pressure of a TAKE external named INLET to a value of 750, immediately.

```
POKE INLET:SP = 750
```

Example 2

Poke a previously defined variable named TOTHP to a value of zero after 30 minutes of simulation from time zero.

```
POKE TOTHP = 0, DELAY = 30
```

Example 3

Poke the maximum time step to a value of 10 at time = 60.

```
POKE DTMAX = 10, TIME = 60
```

Example 4

Poke the pressure of a TAKE external named INLET to a value of 750 at a simulation time of November 6, 2002 at 1:40 p.m.

```
POKE INLET:P+ = 750, TIME= "02/11/06 13:40:00"
```

POKEALL

INTRAN

```
POKEALL dev_names][,
+ KEY.LETTER = dev_kl_include][,
+ DEVICE.EXCLUDE = dev_kl_exclude][,
+ PEEK.MATCH = peek_names_incl][,
+ PEEK.EXCLUDE = peek_names_excl][,
+ UNITS.MATCH = units_include][,
+ SUB.TYPE = subtypes_include][,
+ TO = val][,
+ ECHO = echo][,
+ TIMES = t1 ... tn]
```

Field	Units Key	Description
dev_names	n/a	Devices to include, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See "Device keyletters" on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See "Device keyletters" on page 47.
peek_names_incl	n/a	Peek names to be included, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
peek_names_excl	n/a	Peek names to be excluded, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
units_include	n/a	Units keywords. If this field is used, only peek names that use the specified units are included. See "Default units" on page 129.
subtypes_include	n/a	Device subtypes (STYP attribute values). If this field is used, only peek names of the specified subtypes are included. For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
val	n/a	New value for the included peek names; that is, the value to be poked. Arithmetic expressions and attribute names are valid. See "Expressions, Operators, and Functions" on page 587.
echo	n/a	Indicates whether to produce a list of all peek names that are poked. Valid values are YES and NO.
t_j	n/a	Simulation time(s) when the poke is to be executed. For more information on formats, see "Elapsed time versus clock format" on page 199.

Description

POKEALL is used to poke a set of peek names at the same time, to the same value. Generally, you should use wildcards to form the conditions that define this peek name set. For more information, see “Wildcards” on page 50.

Take note

Cascading effect of conditions

Most of the arguments in a POKEALL command are used to set conditions that determine precisely which peek names should be poked. These arguments function in a cascading manner, each further narrowing the peek name “list” as altered by the previous argument. For example, consider the following POKEALL statement:

```
POKEALL *,
+ KEY.LETTER = E,
+ PEEK.MATCH = *N*:PMAX,
+ PEEK.EXCLUDE = *T*:*,
+ TO 1000
```

In this case, the POKEALL * part of the command includes all peek names in the model; that is, all devices and associated attributes. The KEY.LETTER argument narrows the list to externals only. The PEEK.MATCH argument further narrows the list to only externals whose names contain an “N”, and only the PMAX attribute of those externals. Finally, the PEEK.EXCLUDE argument narrows the list again by removing any externals whose names contain a “T”.

Example input

Example 1

Poke the PMAX at all externals to 1200 psig.

```
POKEALL *,
+ KEY.LETTER = E,
+ PEEK.MATCH = *:PMAX,
+ TO = 1200
```

Example 2

Poke the QMIN for all TAKE externals to 0.

```
POKEALL *,
+ KEY.LETTER = E,
+ SUB.TYPE = TAKE,
+ PEEK.MATCH = *:QMIN,
+ TO = 0
```

PRINT

INTRAN

```
PRINT NAME [(ARG1,...,ARGn)] [,TO=FILENAME][,TIME = T1 ...Tn] [,DT =
INTERVAL]
```

Interactive

```
PRINT [FILENAME]
```

Field	Units Key	Description
NAME	n/a	Name of the print form to be printed
ARG_j	n/a	Arguments to be passed to the print form
FILENAME	n/a	Name of a file other than the OUTTRN file to which to print the form. See "May append or overwrite" under "Take note".
T_j	TIME	Simulation time or times that the print(s) occur. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the prints occur (referenced back to time 0 rather than the BEGIN.TIME value).

Description

PRINT causes reports and text displays to be written to the OUTTRN file (or to another file) at a specified interval and/or selected times. The PRINT command does not work for graphic displays.

INTRAN

Print forms are completely interchangeable with displays that have been built for interactive use. The FORMAT command may be used to control the number of decimal places printed and applies to print forms as well as displays. If a report is too long or wide to be fully displayed, the information beyond the limits is not displayed, though it is printed. If a report exceeds the viewing area in width, printing to a file or to the OUTTRN file results in wrapped text.

Interactive

An interactive print writes what is currently visible to the OUTTRN file or the specified file.

Take note

May append or overwrite

When printing the same display to the OUTTRN file at different times, each display is appended to the OUTTRN file. When printing the same display to another file at different times, the latest display over-writes the previous display in the file (does not append).

File name specification

The file name specification for the print form in the print command is system dependent. Typically, the file name for a print form (or display) is NAME.DSP, where the DSP extension tells TRANS that the file is a display type file. For installations where this convention is used, only the file name prefix NAME needs to be specified. If a directory path is specified for the file, the full path name needs to be quoted.

Use of REPORT command

The interactive REPORT command provides a very easy way to generate print forms. See “REPORT (Interactive)” on page 554.

Print extents

When printing a report that is not visible on the screen, SPS uses a default window size to determine the extent of the printed report. To obtain better results, Windows users may want to consider displaying the report prior to printing. This will allow users to print the report at its full extents, rather than at the default extents.

Example input

Example 1

From the INTRAN file, print a pre-defined display named SUMMARY_TABLE.DSP every 60 minutes of simulation time to the OUTTRN file.

```
PRINT SUMMARY_TABLE, DT = 60
```

Example 2

From the INTRAN file, print a pre-defined display named SUMMARY_TABLE.DSP at 30 and 40 minutes of simulation time to the OUTTRN file, and only pass the name of two pipes named PIPE1 and PIPE2 to the form.

```
PRINT SUMMARY_TABLE(PIPE1,PIPE2), TIME = 30 40
```

Example 3

From the INTRAN file, print a pre-defined display named GAS.DSP to a new file named GASBACK.DSP at 60 minutes of simulation time.

```
PRINT GAS, TO=GASBACK, TIME = 60
```

Example 4

From the INTRAN file, print a pre-defined display named SUMMARY_TABLE.dsp at simulation time November 6, 2002.

```
PRINT SUMMARY_TABLE, TIME = "02/11/06 13:40:00"
```

Example 5

Interactively print a display named OVERVIEW.DSP to the OUTTRN file. First, display the display by entering its name:

```
OVERVIEW
```

Now print the display:

```
PRINT
```

Example 6

Interactively print the Show window for a element named PIPE1 to the OUTTRN file. First display the Show window:

```
SHOW PIPE1
```

Now print the display:

```
PRINT
```

Example 7

Interactively print the current text to a file named VIEWBACK.DSP.

```
PRINT VIEWBACK
```

PROFILE (INTRAN)

INTRAN

```
PROFILE INT
```

Field	Units Key	Description
INT	TIME	Simulation interval (minutes) at which profiles are requested

Description

PROFILE, entered in the INTRAN file, is used to request generation of profile data by TRANS at a given frequency (i.e., every five minutes), which is written to the REVIEW file. Profile data (i.e., pressure, elevation, temperature, flow rate, etc.) is collected at each knot and is measured along the entire length of the pipe.

This command causes TRANS to save to the REVIEW file all pipe data for subsequent use by SimPlot. The profile data is used by to produce distance plots and/or reports of variables like pressure and flow as a function of distance.

You can plot the following distance variables from a cold review fle if a PROFILE statement was included in INTRAN in the review file.

Variable	Description
ELEVATION	Pipeline elevation
PRESSURE	Pipeline pressure
TEMPERATURE	Pipeline temperature
VELOCITY	Pipeline fluid velocity
DENSITY	Pipeline density
STANDARD FLOW	Volumetric flow rate at standard condtions
PIPE.FLOW	Volumetric flow rate at pipeline condtions
MASS.FLOW	Flow rate in mass units
HEAD	Total fluid head
MAX.PRESSURE	Maximum pressure overa time interval
MIN.PRESSURE	Minimum pressure over a time interval
MAOH	Maxmum allowable operating head. Calculated based on user-defined MAOP
MAOP	Maximum allowable operating pressure
MASP	Maximum allowable surge pressure
LAOP	Lowest allowable operating pressure
LASP	Lowest allowable surge pressure

Take note

Not needed for interactive profiles

This command is required *only* to generate distance plots and/or reports with SimPlot. The command is not required for generating interactive distance plots. This is unlike interactive time plots, which do require the use of the TRENDLIST command.

Choosing time interval INT

The time interval at which the data is saved must match the times for which profiles are desired. The time interval is found by adding the time record in the BEGIN command to the time value in the PROFILE command.

Note: For profiles of variables such as PMAX or PMIN, the values are reset at each profile interval.

Example input

Request profile data to be written to the REVIEW file every 10 minutes of simulation time:

```
PROFILE 10
```

RAMP

Note: A RAMP command may also be issued in the INPREP file for an individual device. For more information, see "POKE and RAMP" on page 423.

INTRAN

The RAMP input may be expressed in one of several different formats. The first format shown below uses SPS's CSV input format, which was introduced in SPS version 9.6. For more information on this format, see "CSV input syntax" on page 52.

```
RAMP [IF.EXISTS(POKENAME)],
+ TABLE
{
TIMES VALUES
T1 V1
T2 V2
T3 V3
...
Tn Vn
}
[,
+ MULT = SCALE][,
+ ADD = SHIFT][,
+ REPEAT = {YES | NO}][,
+ TARGET = {YES | NO}][,
+ EFFECTIVE.TIME = EFF]
```

As an alternative, the vector input format may be used:

```
RAMP [IF.EXISTS(POKENAME)] = V1  V2 V3 ... Vn [,
+ TIME = T1 T2 T3 ... Tn][,
+ MULT = SCALE][,
+ ADD = SHIFT][,
+ REPEAT = {YES | NO}][,
+ TARGET = {YES | NO}]
```

Another alternative is the expression input format, which is as follows:

```
RAMP [IF.EXISTS(POKENAME)] = EXPRESSION[,
+ EFFECTIVE.TIME = EFF]
```

Field	Units Key	Description
POKENAME	n/a	Name of the value to be ramped
V_n	n/a	A target value for the parameter POKENAME at each time T_n
T_n	TIME	A target time to achieve the value V_n. For more information on formats, see "Elapsed time versus clock format" on page 199.

Field	Units Key	Description
SCALE	n/a	Dimensionless scale factor to multiply the target values. Scaling is applied before shifting, regardless of the order of appearance. {1}
SHIFT	n/a	Shift factor to be added to the target values. Scaling is applied before shifting, regardless of the order of appearance. {0}
REPEAT	n/a	Repeat the ramp cyclically {NO}
TARGET	n/a	Specifies that TRANS controls the time step to land on each of the T_1, ..., T_n as closely as possible
EXPRESSION	n/a	Valid expression. Note that for the expression format, the only keyword supported is EFFECTIVE.TIME.
EFF	TIME	Simulation time when the ramp goes into effect. For more information on formats, see "Elapsed time versus clock format" on page 199. If used, must be the last phrase in the RAMP command. {0}

Description

RAMP is used to enter a ramped schedule of any pokable value. Often it is used with either the flow rate or pressure for a SALE or TAKE external. There are two forms of the RAMP command. In the first form, a series of target values are entered together with corresponding time values. The item to be changed ramps from one target value to the next from one specified time to the next.

Any number of values and times may be specified, but there must be the same number of values and times. The pokable value is ramped linearly in time from the initial value (as specified in the INPREP file or from a RESTRT or ARCHIVE state) to V_n at time T_n.

In the alternative form of the RAMP command, the item to be ramped is specified as equal to an expression. The expression may or may not be a strict function of time. The item being ramped tracks the value of the expression at any point in the simulation.

Take note

RAMP survives restart

If the model's state is saved in an archive file, the RAMP schedule is also saved to that file. When a new simulation is begun using the previous saved state for initialization, then the RAMP schedule defined in the previous simulation begins where it was saved and then continues. A consequence of this feature is that a simulation may have unintended ramping if you do not know how or when a particular archive was saved.

Use of expressions

An expression may be entered instead of a list of target values for the ramp schedule. The expression is evaluated at the end of each time step, and the poke name set to that value. If the value of the expression is changing rapidly, particularly if the expression is non-linear, the result may be an unstable simulation.

Ramping text strings

A ramp command can be used to change the value of a field that is a text string, such as compressor/pump unit status, SCADA mode, etc. This INTRAN logic can require less memory than the alternative approach of using multiple WHENEVER commands. Additionally, this RAMP syntax does not require switches of flags to prevent multiple commands for changing the valve status unnecessarily as is needed with the use of WHENEVER logic.

Poking of ramped variables

Poking any ramped variable disables the ramp.

LOAD.INPUT

LOAD.INPUT can be used to load RAMP statements.

TIME.ADD

The TIME.ADD directive has been added to the INTRAN ramp syntax. This allows a RAMP sequence to be initiated when a particular condition becomes true. An example is as follows:

```
DEFINE TEST1 = 0
DEFINE TEST2 = 0
```

.

```
WHENEVER (B5:ST = OPENED & TEST2 >= 1)
{
SET TEST1 = TIME
RAMP B5:FR = 0.8 0.6 0.4 0.2 0,
+ TIME = 5 10 15 20 25,
+ TARGET = YES,
+ TIME.ADD = TEST1
}
```

IF.EXISTS

You may specify that it is not an error if the device value being ramped is not in the model. This is done by putting the POKENAME in IF.EXISTS(). An example is as follows:

```
RAMP IF.EXISTS(CTL1:SP) = 50, EFFECTIVE.TIME = 10
```

If the model has a controller or an external named CTL1, the :SP value is ramped to 50 by time = 10. If the model does not have a device named CTL1, TRANS ignore this RAMP statement. The IF.EXISTS feature is most useful when an INTRAN file is used by several models. Things to note with IF.EXISTS:

- No spaces are allowed, i.e., IF.EXISTS(TOTPWR:VAL) is an error.
- It is still an error if the device value being ramped exists, but isn't pokable.
- If the following is entered: RAMP IF.EXISTS(XXX:VAL) = YYY, it is an error if YYY is not a valid peek attribute, expression or constant (whether or not XXX:VAL exists).
- Any syntax errors in the rest of the RAMP statement are not ignored.

REPEAT

The period for repeating is last time minus first time. The corresponding values for last time and first time should be the equal to prevent a discontinuity.

Square waves

Duplicate times are allowed to define square waves.

Using a RAMP with LOAD.STATUS

To prevent unexpected results when issuing a LOAD.STATUS command with a RAMP command, you should do one of the following:

- Set EFFECTIVE.TIME equal to the time for the ramp's first point.
- Set REPEAT = YES.
- Give the RAMP an initial time of zero.

Example input

Example 1

Define a 24 hour load pattern for a SALE external named DELIV with the following data:

TIME (hours)	LOAD (mmscfd)	TIME (hours)	LOAD (mmscfd)
0	12.00	9.5	20.19
2	11.67	10	19.24
5	11.35	12	16.40
6	12.00	16	14.19
8	19.24	20	15.14
8.5	20.19	21	15.14
9	20.50	24	12.00

```
RAMP DELIV:NQ = 12.00  11.67  11.35  12.00  19.24  20.19  20.50
+ 20.19  19.24  16.40  14.19  15.14  15.14  12.00,
+ TIME =  1  120  300  360  480  510  540
+ 570  600  720  960  1200  1260  1440
```

Example 2

Insert a multiplier of 3 to the load pattern described in Example 1.

```
RAMP DELIV:NQ  =  12.00  11.67  11.35  12.00  19.24  20.19  20.50
+ 20.19  19.24  16.40  14.19  15.14  15.14  12.00,
+ TIME  =  1  120  300  360  480  510  540
+ 570  600  720  960  1200  1260  1440,
+ MULT  =  3
```

Each NQ value is multiplied by 3.

Example 3

Multiply each load value by 2 and then add 5 to each load value to the load pattern described in Example 1.

```
RAMP DELIV:NQ  =  12.00  11.67  11.35  12.00  19.24  20.19  20.50
+ 20.19  19.24  16.40  14.19  15.14  15.14  12.00,
+ TIME  =  1  120  300  360  480  510  540
+ 570   600  720  960  1200  1260  1440,
+ MULT  =  2,
+ ADD  =  5
```

Each NQ value will be multiplied by 2 and then have 5 added to it.

Example 4

Ramp the set point of a controller named PID1 from 150 to 200 from time 0 to time 60, and from 200 back to 150 from time 60 to time 120.

```
RAMP PID1:SP = 150 200 150, TIME = 0 60 120
```

Example 5

Ramp the set point of a PID controller named CONT1 to be equal to the pressure at an external named EXT1 minus 150.

```
RAMP CONT1:SP = EXT1:P+ - 150
```

Example 6

Create a ramp that automatically closes a pump or compressor discharge valve named UNIT_NAME whenever the unit is stopped or stopping. Reopen the discharge valve when the unit status is starting or running.

```
RAMP UNIT_NAME.BV:ST = (UNIT_NAME:ST == STOPPED |
+                       UNIT_NAME:ST == STOPPING) ? CLOSE : OPEN
```

The unit is stopped or stopping as reflected by the unit status peek attribute :ST. The discharge valve opens when the unit status is STARTING or RUNNING. This condition is logically the same as *not* STOPPED and *not* STOPPING.

Example 7

Create a ramp with times entered in clock format.

```
RAMP MTOUT:SP = 30  14  14
+               32  24   1
+                1  31  32
+ TIME = "01/03/25 12:00:00"  "01/03/25 12:01:00"  "01/03/25 12:25:00"
+        "01/03/25 12:26:00"  "01/03/25 13:30:00"  "01/03/25 13:31:00"
+        "01/03/25 14:40:00"  "01/03/25 14:41:00"  "01/03/25 15:00:00"
```

REOPEN

INTRAN

```
REOPEN FILE [, OPEN.AS NEWNAME] [, SAVE.AS OLDNAME][,
+DT = PERIOD | TIME =T1 ...Tn]
```

Interactive

```
REOPEN FILE [, OPEN.AS NEWNAME] [, SAVE.AS OLDNAME]
```

Field	Units Key	Description
FILE	n/a	Name of the file to be reopened (OUTTRN, REPLAY, LOGTRN or ALMLOG). The file extension is *not* automatically appended to the given file name.
NEWNAME	n/a	New name of the file {defaults to previous file name}
OLDNAME	n/a	Old name of the file. If no data is provided, TRANS does not close and copy the specified file. {defaults to previous file name}
PERIOD	TIME	Interval at which the given file is re-opened
T_j	TIME	Times to reopen the file. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.

Description

REOPEN allows you to manage the size of files created by TRANS. REOPEN causes TRANS to close the specified file, rename it to the name provided in the SAVE.AS field, and re-direct output from TRANS for the specified file to the name provided in the OPEN.AS field.

Take note

File extensions

The file extension is not automatically appended to the given file name. When re-opening the OUTTRN file, if the file name provided in the SAVE.AS field is "last_day", TRANS saves the current Model.outtrn file to Last_day and not Last_day.outtrn.

Example input

Example 1

Reopen the OUTTRN file.

```
REOPEN OUTTRN
```

Example 2

Use the INTRAN file to reopen the OUTTRN file, save the contents of the current file to a file called previous_day.outtrn and open a new file called current_day.outtrn to record the output from TRANS at a time of 24 hours.

```
REOPEN OUTTRN, OPEN.AS CURR_DAY.OUTTRN, SAVE.AS PREV_DAY.OUTTRN,
TIME = 1440
```

REVIEW SIZE

INTRAN

```
REVIEW SIZE FILESIZE
```

Field	Units Key	Description
FILESIZE	n/a	Size limit of REVIEW file in bytes {1,000,000,000}

Description

REVIEW SIZE limits the size of the REVIEW file to a specified limit. Once the REVIEW file reaches the specified size, the oldest records are overwritten by the new data.

Example input

Limit the size of the REVIEW file to 200000 bytes.

```
REVIEW SIZE 200000
```

SAVE.LINE.FILL

INTRAN

```
SAVE.LINE.FILL
FILENAME [, TIME = T1 ...Tn] [, DT=INT]
```

Interactive

```
SAVE.LINE.FILL FILENAME
```

Field	Units Key	Description
NAME	n/a	Name of the file to be saved.
T_j	TIME	Simulation time or times at which the file is loaded. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INT	TIME	Time interval at which the file is to be loaded

Description

SAVE.LINE.FILL saves FMV (fluid mixture vector) information per pipe. With the SCLPROP equation of state, this will include all fluid properties.

The SAVE.LINE.FILL command saves fluid information, as defined in the INPREP file and/or as defined by the LINE.FILL command, in a network of pipes.

The file can be viewed, edited, and used for loading the fluid information into a model. The LOAD.INPUT command is used to load the SAVE.LINE.FILL or RAMP (or both) information into a simulation.

The information stored when using the SAVE.LINE.FILL command is at model pressure and temperature conditions and the location of batch interfaces is in model length units.

Take note

File name

If FILENAME is used without an extension, the default extension for the saved file is INC. However, FILENAME can use any extension and the saved file will reflect this. A SAVE.LINE.FILL file can, therefore, be used in an INCLUDE statement.

File format

For each equation of state type (AGA, BWRS, CNGA, SCL, and SCLPROP), the line fill data is saved in SPS's CSV format, which is compatible with Excel. The format, which is described in more detail in "CSV input syntax" on page 52, will appear as follows:

```
LINE.FILL TABLE
{
PIPE, LOCATION, key1, key2, ...
pipea, loca, value1a, value2a, ...
pipeb, locb, value1b, value2b, ...
...
pipen, locn, value1n, value2n, ...
}
```

The key values in the header row represent the following:

- For AGA, the keys correspond to SG, CO2, HHV, and the user-defined labels.
- For BWRS the keys correspond to the component names entered on the "+ NAMES" line.
- For CNGA, the keys correspond to SG and HHV.
- For SCL, the keys correspond to the fluid names on each "+ FLUID" line.
- For SCLPROP, the keys correspond to the properties entered on the STATE SCLPROP directive.

A convenient method for updating the line fill of a simulation is to do the following:

1 Use the SAVE.LINE.FILL command to save the line fill data to a *.csv file.

2 Use Excel to make the desired changes to the *.csv file.

3 Load the file back into the model using the LOAD.INPUT command.

Simulation use

A SAVE.LINE.FILL output file can be repeatedly used with a model, which is run under various scenarios. Repeated SAVE.LINE.FILL FILENAME will overwrite the previous file.

Equation of state

The equation of state must remain the same in order to use a SAVE.LINE.FILL file during various simulations. The exception to this, however, is SCL and SCLPROP. A SAVE.LINE.FILL file created from a model using the SCL equation of state can be used in a SCLPROP simulation. For SCLPROP, all applicable fluid properties are saved.

Example input

Example 1

In the INTRAN file, save the LINE.FILL information after 10 minutes of simulation time to a file called line10.sav.

```
SAVE.LINE.FILL LINE10.SAVE, TIME=10
```

Example 2

Interactively, save the LINE.FILL information to a file named try4.

```
SAVE.LINE.FILL TRY4
```

SAVE.STATUS

INTRAN

```
SAVE.STATUS FILENAME [, TIME = T1 T2...Tn] [, DT = INTERVAL]
```

Interactive

```
SAVE.STATUS FILENAME
```

Field	Units Key	Description
FILENAME	n/a	Name of the file. The STA extension is automatically appended for most installations, unless a specific extension is input or a peek attribute has been specified. The file name may contain a peek attribute enclosed in square brackets [ ]. The peek attribute will be replaced with the attribute's value. In this case, no extension is automatically appended to the resulting file name.
T_j	TIME	Simulation time at which an archive should occur. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the archives are written (referenced back to time 0 rather than the BEGIN.TIME value)

Description

SAVE.STATUS takes a "snapshot" of the system state at a specified time and stores the data (every value, equipment status, etc.) at the specified time interval.

Note: If multiple archives are written to one file, the file contains only the latest archived state.

A status file contains a single restart record (i.e., system status at a given point in the simulation) and may be used as the initial status for a subsequent simulation. The archived state may be used to initialize another simulation, or to re-initialize the current (ongoing) simulation using the LOAD.STATUS command.

For more information on initialization and the differences between a status file and an archive file (ARK), see "LOAD.STATUS initialization" on page 156.

Example input

Example 1

In the INTRAN file, save the state of a model to a file named ARCH1.STA at 5.1 minutes of simulation time.

```
SAVE.STATUS ARCH1, TIME = 5.1
```

Example 2

From the INTRAN file, save the state of a model to a file named STEADY.STA at six hours and ten hours of simulation time.

```
SAVE.STATUS STEADY, TIME = 360 600
```

Example 3

From the INTRAN file, save the state of a model to a file named INT90.STA every 90 minutes from time 0 and save the state at time 10.

```
SAVE.STATUS INT90.STA, TIME = 10, DT = 90
```

Example 4

Interactively SAVE.STATUS the state of the simulation to a file named STEADY.STA.

```
SAVE.STATUS STEADY
```

Example 5

Interactively save the state of the simulation and name it “d” followed by the time value that the SAVE.STATUS was executed.

```
SAVE.STATUS d[TIME]
```

SAVE.STEADY

INTRAN and Interactive

```
SAVE.STEADY name [, APPEND=YES | NO]
```

Field	Units Key	Description
name	n/a	Name of the file to which you want to save or append model conditions

Description

SAVE.STEADY allows you to save the current simulation conditions from an SPS run for subsequent reloading either into the same simulation or another one. The SAVE.STEADY command can be executed whether the simulation is currently steady or not. Any transients in the simulation will result in “out of tolerance” messages after the file is loaded and then balanced.

When saving a steady state from SPS, all elements are supported.

SELECT (INTRAN)

INTRAN

```
SELECT PRODUCT
[+ RTU.FILES = file1 file2 ... filen]
```

Field	Units Key	Description
PRODUCT	n/a	Name of the product to be run: SIMULATOR, TRAINER, STATEFINDER, LEAKFINDER or PREDICTOR. {SIMULATOR} **Note:** Must be the same in both INPREP and INTRAN command.
$FILE_i$	n/a	Name of the file(s) where the RTU data is stored. (Required for STATEFINDER and LEAKFINDER only)

Description

SELECT specifies a product and is validated against the license file. The SELECT statement in the INPREP file must match the SELECT statement in the INTRAN file. For more information, see "SELECT (INPREP)" on page 211.

Take note

Statefinder and Leakfinder

The software first opens file1 and the RTU data in file1 drives the model until all of the data is exhausted. Then, the software switches to file2. This process is repeated up to the last file. The software continues to read the last file, expecting to receive live data from SCADA.

Trainer

This command turns *sticky mode* off (the simulation automatically continues running after any command is given) and attempts to match clock time, i.e., an hour of simulation takes an hour of actual time. A SHARE command and a TRENDLIST command must also be included in the INTRAN file to enable the shared memory interface. For more information, see "SHARE" on page 524 and "TRENDLIST" on page 537.

Example input

Use Statefinder in the INTRAN file. The RTU data is stored in a file named RTUDATA.DT.

```
SELECT STATEFINDER
+RTU.FILES = RTUDATA.DT
```

SET

INTRAN

SET POKENAME **=** EXP [**, TIME=**T_1 ... T_n] [**, DT=**INT] [**, KEEP.OLD=YES**]

Interactive

SET POKENAME **=** VALUE [**, TIME =** TIME] [**, DELAY =** DELAY]

Field	Units Key	Description
POKENAME	n/a	Value to be changed
EXP	n/a	A number, peek name, or arithmetic expression of numbers and peek names
T_j	TIME	Simulation time or times when the poke executes. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INT	TIME	Time interval when the poke executes (referenced back to time 0 rather than the BEGIN.TIME value)
VALUE	n/a	A number, peek name or a defined variable
TIME	TIME	Simulation time when the poke executes. For more information on formats, see "Elapsed time versus clock format" on page 199.
DELAY	TIME	Time from the current simulation time when the poke executes. Avoid using with models running in TPORT.

Description

SET may be used to change the value of a pokable parameter during the course of the simulation. SET is similar to the POKE command but it does not cause the time step to go to the minimum. The SET command should be used to change values that do not affect the hydraulics. A number of device-related items (such as a controller set point) as well as simulation parameters are pokable. All user-defined variables are also pokable. "Peek and Poke Keyletters and Attributes" on page 639.

The EXPRESSION can be a number, a peek name, or an expression. Any numbers in the EXPRESSION are taken to be in user-defined units.

The time at which the SET command executes depends on which time option is used when the command is entered. Different time options are available in the INTRAN file and interactively.

INTRAN

Command setup	Effect
SET is by itself and does not target a specific time	Changes the value when the simulation begins
SET is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Changes the value when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
SET targets a specific time, multiple times, or a time interval	Changes the value at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

POKE command setup	Effect
SET does not target a time	Changes the value at the beginning of the next time step of the simulation
SET targets a specific time	Changes the value at the specified time
SET specifies a delay	Changes the value after the delay period expires

Take note

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

IF.EXISTS feature—INTRAN

You may specify that it is not an error if the device value being poked is not in the model. This is done by putting the set name in IF.EXISTS(). An example is as follows:

```
SET IF.EXISTS(CTL1:SP) = 50, TIME = 10
```

If the model has a controller or an external named CTL1, the :SP value changes to 50 at time = 10. If the model does not contain device CTL1, TRANS ignores this SET statement. The IF.EXISTS feature is most useful when an INTRAN file is used by several models.

Things to note with IF.EXISTS:

- No spaces are allowed, i.e., IF.EXISTS(TOTPWR:VAL) is an error.
- It is still an error if the device value being poked exists, but is not pokable.
- If the following is input: SET IF.EXISTS(XXX:VAL) = YYY, it is still an error if YYY does not exist (whether or not XXX:VAL exists).
- Any syntax errors in the rest of the SET statement are not ignored.
- The set still occupies space in IF/ELSE logic. In other words:

```
IF (ZZZ:VAL = 2)
{
SET IF.EXISTS(B1:ST) = OPEN, TIME = 5
SET QQQ:VAL = 10
}
```

does not become:

```
IF (ZZZ:VAL = 2)
SET QQQ:VAL = 10
```

just because a device named B1 is not in the model.

KEEP.OLD feature—INTRAN

You may enable “old” actions to occur by adding the KEEP.OLD = YES statement to the SET command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 60
DEFINE A = 0
DEFINE B = 0
DEFINE C = 0
SET A = 5, TIME = 90
SET B = 10, TIME = 30
SET C = 30, TIME = 0
```

Since the simulation starts at time = 60, the “SET B = 10” command will not execute, since its execution time has already passed at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the SET command execute anyway. For example, the following SET would occur at a simulation start time = 60, despite its specified time:

```
SET B = 10, TIME = 30, KEEP.OLD = YES
```

In the sample, note that the “SET C = 30” command will execute regardless of the simulation start time, because the parameter TIME = 0 indicates that the SET command should be executed at start time.

Interaction with RAMPS

If you define a variable that is being ramped, the variable is updated by the poke and the ramp is disabled.

Expression evaluation - INTRAN

Be careful when setting a variable to a non-numeric expression. Sets are evaluated when they are read, which is prior to the first time step unless they are part of a WHENEVER or DEFINE.SEQUENCE. For example, you want to change the value of variable ABC to the value of PIPE1:P+ at time 100 and you enter the following SET in the INTRAN file:

```
SET ABC = PIPE1:P+, TIME = 100
```

The value of ABC changes to the value of PIPE1:P+ at initialization rather than the value at time 100. In other words, if the value of PIPE1:P+ was 50 at time 0 and 300 at time 100, ABC has a value of 50 after the SET. To obtain the proper

value for PIPE1:P+, the set cannot be evaluated until time 100. For this to occur, the SET needs to be in a WHENEVER or a DEFINE.SEQUENCE, which is evaluated at time 100.

Example input

Example 1

SET a previously defined variable named TOTHP to a value of zero after 30 minutes of simulation from time zero.

```
SET TOTHP = 0, TIME = 30
```

Example 2

Set the maximum time step to a value of 10 at time = 60.

```
SET DTMAX = 10, TIME = 60
```

Example 3

Set the maximum time step to a value of 10 at 1:40 p.m. on November 6, 2002.

```
SET DTMAX = 10,TIME = "02/11/06 13:40:00"
```

SETLIST

INTRAN

```
SETLIST [dev_names][,
+ KEY.LETTER = dev_kl_include][,
+ DEVICE.EXCLUDE = dev_kl_exclude][,
+ PEEK.MATCH = peek_names_incl][,
+ PEEK.EXCLUDE = peek_names_excl][,
+ UNITS.MATCH = units_include][,
+ SUB.TYPE = subtypes_include]
```

Field	Units Key	Description
dev_names	n/a	Devices to include, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See "Device keyletters" on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See "Device keyletters" on page 47.
peek_names_incl	n/a	Peek names to be included, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
peek_names_excl	n/a	Peek names to be excluded, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
units_include	n/a	Units keywords. If this field is used, only peek names that use the specified units are included. See "Default units" on page 129.
subtypes_include	n/a	Device subtypes (STYP attribute values). If this field is used, only peek names of the specified subtypes are included. For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.

Description

SETLIST redefines Show window poke characters to use the SET command instead of the POKE command. By default, changing a value in a Show window executes a POKE command action. Using a SETLIST command causes the Show window to execute a SET command action instead, for the included peek names. Using a SET command action instead of a POKE avoids the minimization of the current time step. For more information on SET, see "SET" on page 518.

Take note

Exclusion of the :ST attribute

For pumps, compressors, and valves, the :ST attribute (status) is automatically excluded by SETLIST. This exclusion is necessary because starting and stopping these devices requires a reduction in the time step for proper simulation.

Cascading effect of conditions

Most of the arguments in a SETLIST command are used set conditions that determine precisely which peek names should be affected by the command. These arguments function in a cascading manner, each further narrowing the peek name “list” as altered by the previous argument. For example, consider the following SETLIST statement:

```
SETLIST *,
+ KEY.LETTER = T,
+ PEEK.MATCH = *P*:P+,
+ PEEK.EXCLUDE = *T*:*,
```

In this case, the SETLIST * part of the command includes all peek names in the model; that is, all devices and associated attributes. The KEY.LETTER argument narrows the list to transfer lines only. The PEEK.MATCH argument further narrows the list to only transfer lines whose names contain an “P”, and only the P+ attribute of those transfer lines. Finally, the PEEK.EXCLUDE argument narrows the list again by removing any transfer lines whose names contain a “T”.

Performance

The SET command is intended to be used to change values that are not hydraulically significant. Using the SET command to change values that are hydraulically significant can result in both a loss of fidelity and in an increase in the computer time required to perform the simulation.

Example input

Specify that all DEFINE values be changed using the SET command.

```
SETLIST *, KEY.LETTER = DE
```

SHARE

INTRAN

```
SHARE [dev_names[,
+ KEY.LETTER = dev_kl_include][,
+ DEVICE.EXCLUDE = dev_kl_exclude][,
+ PEEK.MATCH = peek_names_incl][,
+ PEEK.EXCLUDE = peek_names_excl][,
+ UNITS.MATCH = units_include][,
+ SUB.TYPE = subtypes_include]]
```

Interactive

```
SHARE peekname [For n STEPS]
```

Field	Units Key	Description
dev_names	n/a	Devices to include, by name. Wildcards are permitted. See “Wildcards” on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See “Device keyletters” on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See “Device keyletters” on page 47.
peek_names_incl	n/a	Peek names to be included, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see “Wildcards” on page 50.
peek_names_excl	n/a	Peek names to be excluded, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see “Wildcards” on page 50.
units_include	n/a	Units keywords. If this field is used, only peek names that use the specified units are included. See “Default units” on page 129.
subtypes_include	n/a	Device subtypes (STYP attribute values). If this field is used, only peek names of the specified subtypes are included. For valid STYP values, see “Peek and Poke Keyletters and Attributes” on page 639.
peekname	n/a	The model variable you want to include in shared memory for n time steps.
n	n/a	Number of time steps you want to include the variable in shared memory.

Description

SHARE enables an external TPORT process or other module to access the active simulation, by producing a shared area of memory, called *shared memory*, which contains peek names and values. The amount of peek name data included in the shared area is controlled by command parameters, such that you can limit the amount of data that an

external TPORT process can access. If you do not specify any limitations, SPS shares the same data as specified by the TRENDLIST command(s) in the INTRAN file.

Note: A TRENDLIST command must be included in the INTRAN file for the SHARE command to function properly. For more information, see "TRENDLIST" on page 537.

Multiple SHARE commands are permitted in an INTRAN file. SPS shares the combination of all peek name data found in all SHARE commands.

For more information, see "Shared memory" on page 57.

Take note

Cascading effect of conditions

Most of the arguments in a SHARE command are used set conditions that determine precisely which peek names should be shared. These arguments function in a cascading manner, each further narrowing the peek name "list" as altered by the previous argument. For example, consider the following SHARE statement:

```
SHARE *,
+ KEY.LETTER = E,
+ PEEK.MATCH = *N*:PMAX,
+ PEEK.EXCLUDE = *T*:*,
+ TO 1000
```

In this case, the SHARE * part of the command includes all peek names in the model; that is, all devices and associated attributes. The KEY.LETTER argument narrows the list to externals only. The PEEK.MATCH argument further narrows the list to only externals whose names contain an "N", and only the PMAX attribute of those externals. Finally, the PEEK.EXCLUDE argument narrows the list again by removing any externals whose names contain a "T".

Note: By narrowing the list of shared peek names, you may be able to improve the performance of the simulation, especially if you are using a large model.

Shared memory sizing

For shared data, SPS allocates the amount of memory required for the specified share list, plus an extra amount for temporary use during a TPORT session. The additional memory is used for distance plot information and any interactively shared data used in Show windows or reports. For information on limitations and how to manage shared memory, see "Shared memory" on page 57.

Example input

Example 1

Share all peek name data in the model.

```
SHARE *
```

Example 2

Share peek name data for all externals and block valves only.

```
SHARE *, KEY.LETTER = E B
```

Example 3

Share data only for transfer lines and exclude lines PIPE1, PIPE2 and PIPE12.

```
SHARE *, KEY.LETTER = T,
+ DEVICE.EXCLUDE = PIPE1 PIPE2 PIPE12
```

Example 4

Share data for peek names whose attribute begins with either a “P” or “Q”.

```
SHARE *, PEEK.MATCH = *:P* *:Q*
```

Example 5

This example demonstrates the effect of two separate SHARE commands within the same INTRAN file. The first command shares any transfer line peek name whose attribute contains either a “Q” or “P”, excluding those containing the PKF or AMP attribute. The second command shares any controller peek name whose attribute is either SP, IN, or OUT.

```
SHARE *, KEY.LETTER = T, PEEK.MATCH = *:*Q* *:*P*,
+ PEEK.EXCLUDE = *:PKF *:AMP
SHARE *, KEY.LETTER = C, PEEK.MATCH = *:SP *:IN
+ *:OUT
```

Example 6

Share all peek name data for TAKE externals only.

```
SHARE *, KEY.LETTER = E, SUB.TYPE = TAKE
```

SHOW.STEADY

INTRAN and Interactive

```
SHOW.STEADY [DEVICE=device,] [SORT=method,] [FILE[.APPEND]=filename]
```

Field	Units Key	Description
device	n/a	Name of a device to which you want to limit the report. If you do not enter a device, a full report will be generated.
method	n/a	Method used to sort data (ABSOLUTE, RELATIVE, NAME). ABSOLUTE sorts by absolute difference between modeled value and requested value, with largest absolute difference first. RELATIVE sorts by relative difference between modeled value and requested value, with largest relative difference first. See "LOAD.STEADY" on page 476 for information on absolute and relative differences. NAME sorts alphabetically by peek name.
filename	n/a	The name of the steady state file to which you want to overwrite or append data.

Description

SHOW.STEADY generates a report that shows how closely the current model values are to the values last imported from a steady state file. You may want to generate this report:

- Before a balance to show how far the model is from the desired state.
- Immediately after a balance to show how well the balance worked.
- A few steps after a balance to show how well the model is holding the steady state.

You can customize the report by limiting it to a single device and by changing the sort order. You can choose to save the data to a steady state file and/or display it in an SPS window.

START

INTRAN

```
START NAME [, TIME = T1 T2 ... Tn ] [, DT = INTERVAL] [, KEEP.OLD=YES]
```

Interactive

```
START NAME [, TIME = TIME] [, DELAY = DELAY]
```

Field	Units Key	Description
NAME	n/a	Name of a compressor or pump to be started
T_j	TIME	Simulation time or times at which the compressor or pump begins to start. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the compressor or pump begins to start (referenced back to time 0 rather than the BEGIN.TIME value)
TIME	TIME	Simulation time at which the compressor or pump begins to start. For more information on formats, see "Elapsed time versus clock format" on page 199.
DELAY	TIME	Time from the current simulation time at which the compressor or pump begins to start. Avoid using with models running in TPORT.

Description

START causes a compressor or pump to begin starting. The time at which the compressor or pump begins to start depends on which time option is used when the command is entered. Different time options are available in the INTRAN file and interactively.

INTRAN

Command setup	Effect
START is by itself and does not target a specific time	Begins to start the compressor or pump when the simulation begins

Command setup	Effect
START is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Begins to start the compressor or pump when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
START targets a specific time, multiple times, or a time interval	Begins to start the compressor or pump at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

Command setup	Effect
START does not target a time	Begins to start the compressor or pump at the beginning of the next time step of the simulation
START targets a specific time	Begins to start the compressor or pump at the specified time
START specifies a delay	Begins to start the compressor or pump after the delay period expires

Take note

Start time & inertia multipliers

The pump or compressor begins to start when the START command is invoked. However, these devices require a certain period of time to actually start. Compressor start time is specified explicitly in the INPREP file or in some cases may be set interactively or in the INTRAN file.

The time it takes a pump to start depends on its inertia, which is specified in the INPREP file. To change a pump's start time, poke a multiplier of the pump's inertia interactively. There is a start inertia multiplier called GIMSTART. This is a global multiplier and will consequently change the starting inertia of all pumps in the model. See "GLOBALS (GB) attributes" on page 665.

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

KEEP.OLD feature—INTRAN

You may enable "old" actions to occur by adding the KEEP.OLD = YES statement to the START command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 60
START UNIT1, TIME = 90
START UNIT2, TIME = 30
START UNIT3, TIME = 0
```

Since the simulation starts at time = 60, the "START UNIT2" command will not execute, since its execution time has already passed at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the START

command execute anyway. For example, the following START would occur at a simulation start time = 60, despite its specified time:

```
START UNIT2, TIME = 30, KEEP.OLD = YES
```

In the sample, note that the “START UNIT3” command will execute regardless of the simulation start time, because the parameter TIME = 0 indicates that the START command should be executed at start time.

Example input

Example 1

Start a pump/compressor named UNIT2 at a simulated time of 15 minutes.

```
START UNIT2, TIME = 15
```

Example 2

Start a pump/compressor named UNIT3 immediately.

```
START UNIT3
```

Example 3

Interactively start a pump/compressor named UNIT3 10 minutes from the current simulation time.

```
START UNIT3, DELAY = 10
```

Example 4

Begin starting a pump/compressor named UNIT1 at 1:40 p.m. on November 6, 2002.

```
START UNIT1, TIME = “02/11/06 13:40:00”
```

STOP

INTRAN

```
STOP NAME [, TIME = T1 T2 ... Tn] [, DT = INTERVAL] [, KEEP.OLD=YES]
```

Interactive

```
STOP NAME [, TIME = TIME] [, DELAY = DELAY]
```

Field	Units Key	Description
NAME	n/a	Name of a compressor or pump to be stopped or a block valve to be stopped
T_j	TIME	Simulation time or times at which the compressor or pump begins to stop. Simulation time or times at which the block valve stops its transition. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval at which the compressor or pump begins to stop. Time interval at which the block valve stops its transition (referenced back to time 0 rather than the BEGIN.TIME value).
TIME	TIME	Simulation time at which the compressor or pump begins to stop. Simulation time at which the block valve stops its transition. For more information on formats, see "Elapsed time versus clock format" on page 199.
DELAY	TIME	Time from the current simulation time at which the compressor or pump begins to stop. Simulation time at which the block valve stops. Avoid using with models running in TPORT.

Description

STOP has two functions, depending on the type of equipment. For compressors and pumps, STOP causes the compressor or pump to begin stopping. For a block valve that is opening or closing, STOP causes the block valve to stop its transition, which forces the valve position (fraction open) to remain at the current value. STOP is often used to simulate the failure of a valve actuator resulting in a block valve being stuck in a partially open condition.

The time at which the compressor or pump begins to stop (or the block valve actually stops) depends on which time option is used when the command is entered. Different time options are available in the INTRAN file and interactively.

INTRAN

Command setup	Effect
STOP is by itself and does not target a specific time	Begins to stop the compressor or pump when the simulation begins
STOP is in a command list (within a WHENEVER or DEFINE.SEQUENCE) and does not target a specific time	Begins to stop the compressor or pump when (and if) the command list is executed. Multiple pokes are executed in the order in which they appear inside the command list.
STOP targets a specific time, multiple times, or a time interval	Begins to stop the compressor or pump at each specified simulation time and/or time interval. To specify the time as relative to the start time of the simulation, precede it with a plus sign (+). Otherwise, the time is used as an absolute simulation time.

Interactive

Command setup	Effect
STOP does not target a time	Begins to stop the compressor or pump at the beginning of the next time step of the simulation
STOP targets a specific time	Begins to stop the compressor or pump at the specified time
STOP specifies a delay	Begins to stop the compressor or pump after the delay period expires

Take note

Time and delay parameter

If a time parameter and a delay parameter are both entered, the time parameter overrides the delay parameter.

Stop time and inertia multipliers

A pump or compressor begins to stop when the STOP command is invoked. However, these devices require a certain period of time to actually stop. Compressor stop time is specified explicitly in the INPREP file or in some cases may be changed interactively or in the INTRAN file.

The time it takes a pump to stop depends on its inertia, which is specified in the INPREP file. To change a pump's stop time, change (poke) the multiplier of the pump's inertia interactively. There is a stop inertia multiplier called GIMSTOP. This is a global multiplier and will consequently change the stopping inertia of all pumps in the model. (See "GLOBALS (GB) attributes" on page 665.

KEEP.OLD feature—INTRAN

You may enable "old" actions to occur by adding the KEEP.OLD = YES statement to the STOP command. For example, consider the following INTRAN file sample:

```
BEGIN 0,
+ BEGIN.TIME = 60
STOP UNIT1, TIME = 90
STOP UNIT2, TIME = 30
```

```
STOP UNIT3, TIME = 0
```

Since the simulation starts at time = 60, the “STOP UNIT2” command will not execute, since its execution time has already passed at the simulation start time. In this situation, you can add the KEEP.OLD statement to have the STOP command execute anyway. For example, the following STOP would occur at a simulation start time = 60, despite its specified time:

```
STOP UNIT2, TIME = 30, KEEP.OLD = YES
```

In the sample, note that the “STOP UNIT3” command will execute regardless of the simulation start time, because the parameter TIME = 0 indicates that the STOP command should be executed at start time.

Example input

Example 1

Stop a pump/compressor named UNIT1 immediately.

```
STOP UNIT1
```

Example 2

Stop a pump/compressor named UNIT2 at a simulated time of 15 minutes.

```
STOP UNIT2, TIME = 15
```

Example 3

A block valve named BLOK starts to open at a simulation time of 59 minutes. The valve has an opening time of 3 minutes. Stop the block valve at a simulation time of 60 minutes so that it gets stuck in a partially open state.

```
STOP BLOK, TIME = 60
```

Example 4

Stop a pump/compressor named UNIT3 10 minutes from the current simulation time using an interactive command.

```
STOP UNIT3, DELAY = 10
```

Example 5

Begin STOPPING a pump/compressor named UNIT1 at 1:40 p.m. on November 6, 2002.

```
STOP UNIT1, TIME = “02/11/06 13:40:00”
```

SUBMIT.SEQUENCE

INTRAN

SUBMIT.SEQUENCE NAME [(ARG$_1$, ... ARG$_n$)] [, **TIME** = TIME] [, **DT** = INTERVAL]

Interactive

NAME [(ARG$_j$)] [, **TIME** = TIME]

Field	Units Key	Description
NAME	n/a	Name of the defined sequence to be submitted
ARG$_j$	n/a	Optional sequence arguments
TIME	TIME	Time when the defined sequence is to be submitted. For more information on formats, see "Elapsed time versus clock format" on page 199. **Note:** In place of a TIME keyword, you can also use the TIMETABLE keyword. A time table defines a schedule of times when actions are to occur. Time tables are created using the DEFINE.TIMETABLE command, as described in "DEFINE.TIMETABLE" on page 450.
INTERVAL	TIME	Time interval when the defined sequence is to be submitted (referenced back to time 0 rather than the BEGIN.TIME value)

Description

SUBMIT.SEQUENCE is used to submit (execute) a defined sequence that was previously defined using the DEFINE.SEQUENCE command.

If arguments were included in the DEFINE.SEQUENCE definition, they must be passed to the DEFINE.SEQUENCE when the defined sequence is submitted.

If a TIME is included then the defined sequence is submitted at that time. If a TIME is not included, then the defined sequence is submitted according to the following rules:

INTRAN

- If the SUBMIT.SEQUENCE command is not in a command list associated with a WHENEVER or DEFINE.SEQUENCE command, then the defined sequence is invoked at the start of the simulation according to its sequential order in the INTRAN file.
- If the SUBMIT.SEQUENCE command is in a command list associated with a WHENEVER or a DEFINE.SEQUENCE command, then the defined sequence is invoked during execution of the command list, based on its sequential order in the command list.

Interactive

The sequence is invoked when the command is given.

Take note

Abbreviation

The SUBMIT.SEQUENCE command may be abbreviated as SUB.SEQ.

Example input

Example 1

In the INTRAN file, submit a defined sequence named OPEN_START at time = 60.

```
SUBMIT.SEQUENCE OPEN_START, TIME = 60
```

Example 2

In the INTRAN file, submit a defined sequence named OPEN_START at a simulation time of 1:40 p.m. on November 6, 2002.

```
SUBMIT.SEQUENCE OPEN_START, TIME = "02/11/06 13:40:00
```

Example 2

In the INTRAN file, submit a defined sequence named DELPRES at the start of the simulation and pass an argument setting the set point pressure equal to 300.

```
SUB.SEQ DELPRES(300)
```

Example 3

The following defined sequence is in the INTRAN file.

```
DEF.SEQ INT1(VV1,T2,VV3)
{
  OPEN VV1
  WAIT T2
  OPEN VV3
}
```

To invoke (submit) the sequence while running TRANS interactively, type the following at the command line:

```
INT1(B1,5,B2)
```

This sequence opens a block valve named B1, waits five minutes, then opens a block valve named B2.

TIMEPAGE

Description

Use BEGIN instead of TIMEPAGE. See “BEGIN” on page 434.

TRENDLIST

INTRAN

```
TRENDLIST dev_names[,
+ KEY.LETTER = dev_kl_include][,
+ DEVICE.EXCLUDE = dev_kl_exclude][,
+ PEEK.MATCH = peek_names_incl][,
+ PEEK.EXCLUDE = peek_names_excl][,
+ UNITS.MATCH = units_include][,
+ SUB.TYPE = subtypes_include][,
+ ECHO = echo]
```

Field	Units Key	Description
dev_names	n/a	Devices to include, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See "Device keyletters" on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See "Device keyletters" on page 47.
peek_names_incl	n/a	Peek names to be included, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
peek_names_excl	n/a	Peek names to be excluded, in the form of {device name}:{attribute}. Wildcards are permitted, but should be used carefully. Before using wildcards in this type of field, see "Wildcards" on page 50.
units_include	n/a	Units keywords. If this field is used, only peek names that use the specified units are included. See "Default units" on page 129.
subtypes_include	n/a	Device subtypes (STYP attribute values). If this field is used, only peek names of the specified subtypes are included. For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
echo	n/a	Indicates whether to produce a list of all peek names that are trended. Valid values are YES and NO.

Description

TRENDLIST is required if you want to generate time plots. TRENDLIST generates a list of trend data from which you can create time plots, either interactively or through . Based on the peek attributes you specify in the TRENDLIST, SPS generates a list of the corresponding values and saves them to a REVIEW file.

While you could choose to include all peek attribute data in the REVIEW file, SPS would not run efficiently. Therefore, you should carefully select the attributes you anticipate plotting using wildcards. For more information, see "Wildcards" on page 50. For more information on improving model performance, related to TRENDLIST, see "TRENDLISTs" on page 195.

Multiple TRENDLIST commands are permitted in an INTRAN file. SPS combines all trend data found in all TRENDLIST commands.

Take note

TRENDLIST required for TPORT

Because a valid TRENDLIST command is required for the SHARE command, a TRENDLIST command is required to run TPORT.

Cascading effect of conditions

Most of the arguments in a TRENDLIST command are used set conditions that determine precisely which peek names and values should be included (or *trended*). These arguments function in a cascading manner, each further narrowing the peek name "list" as altered by the previous argument. For example, consider the following TRENDLIST statement:

```
TRENDLIST *,
+ KEY.LETTER = E,
+ PEEK.MATCH = *N*:PMAX,
+ PEEK.EXCLUDE = *T*:*,
```

In this case, the TRENDLIST * part of the command includes all peek names in the model; that is, all devices and associated attributes. The KEY.LETTER argument narrows the list to externals only. The PEEK.MATCH argument further narrows the list to only externals whose names contain an "N", and only the PMAX attribute of those externals. Finally, the PEEK.EXCLUDE argument narrows the list again by removing any externals whose names contain a "T".

Note: By narrowing the list of trended peek names, you may be able to improve the performance of the simulation, especially if you are using a large model.

Wildcard syntax

Wildcard syntax for TRENDLIST is identical to other commands, with the exception of the field following the main command, TRENDLIST. Normally, you should use this field to set conditions on which devices will be included in the list, by name. However, this field also accepts full peek names for setting conditions to narrow the list, as would normally be specified with the PEEK.MATCH argument. If the field item contains a colon, SPS assumes that it is a full peek name, and performs matching appropriately. If it does not, SPS assumes that matching should occur based on device name only.

This functionality is maintained to ensure backwards compatibility with older versions of SPS, which did not support a PEEK.MATCH argument. However, with any new modeling and analysis, you should avoid using peek names in this manner, reserving them for the PEEK.MATCH argument only.

Global variables

All global variables are automatically saved to the REVIEW file.

Example input

Example 1

Request trend data to be stored for a valve named VALVE1.

```
TRENDLIST VALVE1
```

Example 2

Request trend data to be stored for the following devices.

```
TRENDLIST UNT956 BLK433 EXS355 HDR001 UNT958 UNT959 HDR01
+ ACT012 SNS032
```

Example 3

Request all trend data to be stored.

```
TRENDLIST *
```

Example 4

Request trend data to be stored for all devices with names that begin with PIPE and have exactly one character after the word PIPE.

```
TRENDLIST PIPE?
```

Example 5

Request trend data to be stored for all devices that begin with UNT.

```
TRENDLIST UNT*
```

Example 6

Request trend data to be stored only for all of the externals and block valves.

```
TRENDLIST *, KEY.LETTER = E B
```

Example 7

Request trend data to be stored only for transfer lines but exclude the following lines: PIPE1, PIPE2 and PIPE12.

```
TRENDLIST *, KEY.LETTER = T,
+ DEVICE.EXCLUDE = PIPE1 PIPE2 PIPE12
```

Example 8

Request trend data to be stored only for all the peeks that begin with the letter P and the letter Q in all of the devices.

```
TRENDLIST *, PEEK.MATCH = *:P* *:Q*
```

Example 9

Request trend data to be stored only for all the 2-character peeks that begin with the letter P and the letter Q in all of the devices but exclude the :PK peeks.

```
TRENDLIST *,
+ PEEK.MATCH = *:P? *:Q?, PEEK.EXCLUDE = *:PK
```

Example 10

Request trend data to be stored only for the reciprocating compressors and only trend the pressure and speed peeks. List all of the variables that are included in the TRENDLIST command to the OUTTRN file.

```
TRENDLIST *, KEY.LETTER = KR,
+ UNITS.MATCH = PRESSURE SPEED,
+ ECHO = YES
```

Example 11

This example contains two separate TRENDLIST commands within the same INTRAN file. The first one trends all peeks that contain the letters Q and P for transfer lines only but excludes the :PKF and :AMP peeks. The second one trends all the :SP, :IN and :OUT peeks for controllers only.

```
TRENDLIST *, KEY.LETTER = T, PEEK.MATCH = *:*Q* *:*P*,
+ PEEK.EXCLUDE = *:PKF *:AMP
TRENDLIST *, KEY.LETTER = C, PEEK.MATCH = *:SP *:IN
+ *:OUT
```

Example 12

Request trend data to be stored only for all of the TAKE externals.

```
TRENDLIST *, KEY.LETTER = E, SUB.TYPE = TAKE
```

WAIT, WAIT.UNTIL

INTRAN

```
WAIT TIME
/*or
WAIT.UNTIL (CONDITIONAL)
```

Field	Units Key	Description
TIME	TIME	Interval of time to wait before executing the next command in a command list
CONDITIONAL	n/a	Relational test on numerical or status values. The next command is executed when this is true.

Description

WAIT is valid only inside a command list and causes TRANS to wait before issuing the next command. There are two versions of the wait statement available for use in a command list. The first is the WAIT command where a period of time to elapse is specified before the next statement in a command list is executed. In this context, “next” means the next statement in sequential order in the command list.

The WAIT.UNTIL form of the wait statement makes the wait period dependent on some condition. When the conditional becomes true, the next statement in the command list is executed.

Example input

Example 1

Inside a command list, open a block valve named BB1, wait 2 minutes and then close a block valve named BB2.

```
{OPEN BB1
 WAIT 2
 CLOSE BB2}
```

Example 2

Inside a command list, wait until a pump/compressor named UNIT1 is running before starting a second unit named UNIT2.

```
{WAIT.UNTIL ( UNIT1:ST = RUNNING )
 START UNIT2}
```

Example 3

Inside a command list, wait until the simulation time is 1:40 p.m. on November 6, 2002 to start UNIT2.

```
WAIT.UNTIL  TIME > TIMEVALUE("02/11/06 13:40:00")
START UNIT2
```

WHENEVER

INTRAN

```
WHENEVER (CONDITIONAL)
{
  COMMAND LIST
}
```

Field	Units Key	Description
CONDITIONAL	n/a	Relational test on numerical or status values
COMMAND LIST	n/a	See “Command list” on page 443.

Description

WHENEVER is used to monitor a value or status and take some action when the monitored conditional becomes true.

The conditional can be one of the three following logical tests:

- A simple relational test such as “PIPE1:P+ <= 150.0”, “PIPE1:P+ <= UNIT2:P-”, etc.
- A status test such as “UNIT1:ST = STARTING”, “VALVE1:ST = OPENED”. The status test keywords for pumps and compressors are = STARTING, STOPPING, RUNNING, and STOPPED. The status test keywords for valves are STOPPED, OPENING, CLOSING, OPENED, and CLOSED.
- A combination of two or more simple relational tests using and (&) and or (|.) An example of a combined test is as follows:

```
"UNIT1:ST = STARTING & VALVE1:ST = CLOSED"
```

TRANS evaluates the WHENEVER conditional every time step during the simulation, and executes it every time it is true, provided that the command list is not already in sequence. Execution of the command may make the time step go to its minimum value.

Take note

Specifying the conditional

Because the command list is executed each time step that the conditional is true, the conditional should be defined so that it is usually false. If the conditional is usually true, the result may be TRANS running with a minimum time step, because certain TRANS commands force a minimum time step.

Turning a WHENEVER on and off

A WHENEVER can be defined such that it can be turned on or turned off using a user-defined variable as a switch.

Use of braces

The use of braces ({ }) is optional for a command list that has one item. If the case of multiple lines, the braces must be used to show where the command list begins and ends.

Caution using ARCHIVE and LOAD.STATUS

Performing an ARCHIVE or a LOAD.STATUS during the execution of a WHENEVER may cause the WHENEVER to hang. That is all the actions associated with the WHENEVER will not be executed. This is especially true if the LOAD.STATUS has affected any WAITs or WAIT.UNTILs.

Example input

Example 1

Use a WHENEVER to define a low suction trip. In this example, whenever the suction pressure of UNIT1 falls below 20, UNIT1 stops.

```
WHENEVER ( UNIT1:P- < 20 ) STOP UNIT1
```

Example 2

"Example 1" on page 543 has the effect of not allowing UNIT1 to keep running if the suction pressure gets too low. Revise the conditional so that UNIT1 trips *only* if it is running *and* the suction pressure falls below 20.

```
WHENEVER ( UNIT1:P- < 20 & UNIT1:ST = RUNNING ) { STOP UNIT1 }
```

Example 3

Define a WHENEVER for a sequence where unit U1 stops when units U1 and U2 are both running and either the suction pressure of U1 falls below 20 or the discharge pressure of control valve V1 rises above 1100.

```
WHENEVER (( U1:P- < 20 | V1:P+ > 1100 ) & U1:ST = RUNNING &
+       U2:ST = RUNNING)
         { STOP U1 }
```

Example 4

Define a WHENEVER for a stop sequence that can be turned on or off when desired.

```
DEFINE SWITCH = 0
WHENEVER ( UNIT2:P- < 40 & SWITCH = 1 ) {
              STOP UNIT2
              OPEN VALVE1
             }
```

In this example, SWITCH is a user-defined variable. If the stop sequence is to be executed, then POKE SWITCH = 1. The defined variable SWITCH is used as a switch to turn the conditional on or off. The stop sequence is executed only when the SWITCH variable is equal to 1 and the suction pressure of UNIT2 falls below 40.

Example 5

Pause the simulation whenever the simulation time is November 6, 2002 at 1:40 p.m.

```
WHENEVER ( TIME > TIMEVALUE("02/11/06 13:40:00") )
   {
    DO.INTERACTIVE "PAUSE"
   }
```

CHAPTER

12

Interactive Commands

The following commands may be used interactively. For more information on setting up your model to enter commands interactively, see "Preparing to run TRANS interactively" on page 56.

- "SPAWN" on page 561
- "START" on page 528
- "STOP" on page 531
- "SUBMIT.SEQUENCE" on page 534
- "TIMEPLOT" on page 562
- "TYPE" on page 563
- "ZZSTICK" on page 565

BACKGROUND

INTERACTIVE

```
BACKGROUND "system command"
```

Field	Units Key	Description
system command	n/a	Operating system command to be executed at the system level.

Description

BACKGROUND is used to submit operating system commands from within a TRANS environment.

Example input

Example 1

Start a TPORT process on the “predict” model in another window from the TRANS command line.

```
BACKGROUND "tport predict -CRT=xw &" /* windows/dos
```

Example 2

Run TRANSSHARE on the predict model from within the INTRAN file at time = 10 minutes using the BACKGROUND command.

```
DEFINE FLAG_T/0/ = 0
WHENEVER(TIME > 10  &  FLAG_T = 0)
{
    SET FLAG_T = 1
    DO.INTERACTIVE  "BACKGROUND transshare predict"
}
```

The DO.INTERACTIVE command in this example is used to generate interactive commands from within the INTRAN file. See “DO.INTERACTIVE” on page 452.

DISTPLOT

(Interactive)

```
DISTPLOT [FIGNAME]
```

Field	Units Key	Description
FIGNAME	n/a	Name of a previously defined distance plot

Description

DISTPLOT is used to create a distance plot or open an existing distance plot that is in memory. To create a new distance plot, enter the command DISTPLOT only. To edit a distance plot, enter DISTPLOT followed by the name of the plot (i.e., DISTPLOT PROF1). If you want to replace the version of the plot in memory with a saved version, use "REREAD" on page 555. For more information on creating, displaying, and editing distance plots, see "Distance plots" on page 169. For a list of keywords corresponding to data you can plot, see "Distance plot items" on page 172. See the device type you want to plot for more information on issues specific to plotting that device's data.

Example input

Example 1

Create a new distance plot.

```
DISTPLOT
```

Example 2

Change an existing distance plot named PROG1.

```
DISTPLOT PROG1
```

Example 3

Display a distance plot named PROG1 from any window.

```
PROG1
```

FLIP

Interactive

FLIP

Description

FLIP is used when working with time plots, distance plots, or reports to alternate between the display of the plot or report and the editor or menu used to edit the plot or report.

Example input

To display a plot or report from the associated menu or editor, type the following:

```
FLIP
```

To display the associated menu or editor from the plot or report, type the following:

```
FLIP
```

HALT/QUIT

Interactive

HALT | **QUIT**

Description

HALT and QUIT both terminate an interactive simulation. These commands are interchangeable and have the same effect.

Take note

Restart record

Using either of these commands causes the current simulation to be terminated. When the simulation is terminated by means of a HALT or a QUIT, a restart record is not written. To save the current status of the simulation for possible later use, archive the model before issuing the HALT or QUIT command to terminate the simulation.

Note: Once a simulation is terminated, all interactive defined variables and macros are no longer valid.

Example input

Example 1

Terminate an interactive simulation using the HALT command.

```
HALT
```

Example 2

Terminate an interactive simulation using the QUIT command.

```
QUIT
```

When halt or quit is entered, the following message appears on the command line:

```
<HALT/QUIT accepted - terminating>
```

HELP

Interactive

```
HELP
```

Description

HELP displays the syntax and a short description of the most-used interactive commands.

PRINTALL

Interactive

```
PRINTALL [FILENAME]
```

Field	Units Key	Description
FILENAME	n/a	Name of a file where the current report is to be written

Description

PRINTALL entered by itself on a report prints the entire contents of the report to the OUTTRN file whether or not the entire report is being displayed. Additional width and length of the report are printed. Multiple reports are appended when printed to the OUTTRN file. PRINTALL accepts reports that are greater than 80 characters in width and greater than one page in length.

The PRINTALL command followed by a file name prints the entire contents of the report to that file name. (A DSP extension is automatically appended to the file name.) If the file name already exists, a prompt appears asking if the file is to be over-written. If the answer is N, the display is not printed and the > prompt re-appears. If the answer is Y, the old file is overwritten. (A time step is taken before the > prompt re-appears.)

PRINTALL only works for reports, and only works when SPS for Windows is put in compatibility mode. When PRINTALL is used from the TPORT window or terminal, a file name must be included with the PRINTALL command.

Take note

Print extents

When printing a report that is not visible on the screen, SPS uses a default window size to determine the extent of the printed report. To obtain better results, Windows users may want to consider displaying the report prior to printing. This will allow users to print the report at its full extents, rather than at the default extents.

Example input

Example 1

A report defined in RPTEST.DSP generates a report of 100 lines. To print the entire report (all 100 lines) to the OUTTRN file, first display the report.

```
RPTEST
```

Then enter the following command:

```
PRINTALL
```

Example 2

A report defined in RPTEST.DSP generates a report of 100 lines. To print the entire report (all 100 lines) to a file called TLINES.DSP, first display the report.

```
RPTEST
```

Then enter the following command.

```
PRINTALL TLINES
```

REPORT (Interactive)

Interactive

```
REPORT [REPNAME]
```

Field	Units Key	Description
REPNAME	n/a	Name of previously defined report

Description

REPORT is used to create a report or open an existing report that is in memory. To create a new report, enter the command REPORT only. To edit a report, enter REPORT followed by the name of the report (i.e., REPORT MYREPORT). If you want to replace the version of the report in memory with a saved version, use "REREAD" on page 555. For more information on creating, displaying, and editing reports, see "Reports" on page 182.

Example input

Example 1

Create a new report.

```
REPORT
```

Example 2

Edit an existing report named REP11.

```
REPORT REP11
```

Example 3

Display a report named REP11 from any display.

```
REP11
```

REREAD

Interactive

```
REREAD FIGNAME
```

Field	Units Key	Description
FIGNAME	n/a	Name of the display to be reread.

Description

REREAD is used when working with time plots, distance plots, or reports to replace the version of the plot or report in memory with the version saved on disk.

Example input

To load in the disk version of a saved report called REPORT1 enter the following command.

```
REREAD REPORT1
```

To load in the disk version of a saved distance plot called PR_PROF enter the following command.

```
REREAD PR_PROF
```

RUN, RUN UNTIL, RUN WHILE, RUN FOR

Interactive

```
RUN
/* or
RUN UNTIL PEEKNAME COMPARISON VALUE
/* or
RUN WHILE PEEKNAME COMPARISON VALUE
/* or
RUN FOR TIMEUNITS
```

Field	Units Key	Description
PEEKNAME	n/a	Name of a peek attribute used in the comparison
COMPARISON	n/a	Relational comparison (>, <, =, <=, >=, !=)
VALUE	n/a	Constant numerical value used in the comparison or time value in clock format (in single quotes). For more information on time formats, see "Elapsed time versus clock format" on page 199.
TIMEUNITS	n/a	Run duration in SECONDS, MINUTES, HOURS, DAYS, or STEPS {minutes}

Description

RUN is used to resume a simulation that you have paused, to start running and continue running until some condition has been met and then pause, or to run for a specified amount of simulation time or a specified number of time steps.

A simulation is paused when the simulation time is not progressing. When a simulation is paused, the > prompt is observed in the upper-left corner of the display. When a simulation is running, the > prompt is replaced by the word <Running> (or by the user-defined value of RUNNING.MESSAGE).

The RUN command may be used in any one of four forms.

- The simplest form is the RUN command without any condition. Entering this command by itself causes the simulation to progress from the current paused state, and the simulation continues running until it is either paused again or terminated by reaching the time limit.
- The RUN UNTIL form of the RUN command causes the simulation to progress until the condition specified becomes true, then the simulation pauses. The condition is based on a comparison between any valid peek attribute name and some numerical value. (RU: appears on the command line along with the peek name specified and the actual value compared with the comparison value).
- The RUN WHILE form of the RUN command causes the simulation to progress while some condition is true, then the simulation pauses when the condition becomes false. The condition is based on a comparison between any valid peek attribute name and some numerical value. (RW: appears on the command line along with the peek name specified and the actual value compared with the comparison value).
- The RUN FOR form of the RUN command causes the simulation to progress either for a specified amount of simulation time or for a specified number of time steps, then the simulation pauses. (RF: appears on the command line and counts down the time or the time steps remaining).

Take note

General

Use RUN UNTIL or RUN WHILE to monitor a simulation value. For example, if the current simulation time is 100 minutes and the next event of interest is at a simulation time of 1200 minutes, then the simulation could be run until it approaches the time of interest without monitoring it closely. The input for this could be any of the following:

```
RUN UNTIL TIME >= 1195
RUN UNTIL TIME >= "19:55"
RUN WHILE TIME <= 1195
```

All of these examples would have the same effect; that is, the simulation pauses when the simulation time equals or exceeds 1195 minutes.

Continue with a conditional RUN

If the RUN UNTIL or RUN WHILE commands are being used and the simulation is paused manually, the conditional run may be resumed by entering either RUN UNTIL or RUN WHILE without the conditional. These may be abbreviated RUN U or RUN W.

Use of equals sign (=)

When using the RUN UNTIL or RUN WHILE form of the RUN command, avoid using the equals sign (=) relational operator as the basis of the comparison. For example, if the pressure at the suction side of a station is being monitored and the simulation is to pause when the pressure reaches a certain level, such as 300 psig, then do not expect the suction pressure to be exactly 300 psig. It is unlikely that the pressure calculated will be exactly 300 psig because of the numerical methods used. SPS takes time steps with each time step being some finite time period. So if at one time step the pressure is 299 psig and the pressure is changing 10 psi per minute then a 1 minute time step changes the pressure to 309 psig.

Based on this example, the following command would not pause the simulation at the desired condition:

```
RUN UNTIL STA:P- = 300
```

The calculated pressure would not equal exactly 300 psig so the simulation would not pause.

A better approach is to pause the simulation before the desired condition is reached. For example, pause the simulation when the pressure exceeds 280 psig with the following command:

```
RUN UNTIL STA:P- > 280
```

Example input

Example 1

From a paused simulation, start the simulation running again.

```
RUN
```

Example 2

From a paused simulation, run the simulation until the simulation time exceeds 300 minutes.

```
RUN UNTIL TIME > 300
```

Note that when the simulation pauses, the simulation time may be significantly larger than 300 minutes depending on the current time step.

Example 3

Run the simulation for as long as the pressure at the suction side of a station named STA2 is above 275 psig.

```
RUN WHILE STA2:P- > 275
```

Example 4

Run the simulation for a duration of 10 minutes.

```
RUN FOR 10
```

Example 5

Run the simulation for 10 time steps.

```
RUN FOR 10 STEPS
```

SHOW

Interactive

```
SHOW [NAME]
```

Field	Units Key	Description
NAME	n/a	Name of a device, node name, defined variable, defined sequence, defined path name, fluid name, alarm name, globals, audits, events, or running.message.

Description

SHOW generates a Show window that provides a list of attributes for a device or property set that you specify. The current simulation time and the time step are displayed on all Show windows.

For more information on using the Show window, see "Show windows" on page 188.

You can use the following forms of the SHOW command to generate a Show window:

SHOW	When entered by itself, the following is listed: the names of all the devices, nodes, defined variables, defined sequences, fluid names, alarm names, alarm categories, globals, audits, station names, data curve names, and running.message. The names are listed in alphabetical order downward, not across.
SHOW GLOBALS	Lists all of the global peeks/pokes. For more information, see "GLOBALS (GB) attributes" on page 665.
SHOW AUDITS	Lists all of the audit-related peeks, such as DCPU, NREJECT, etc. For more information, see "AUDITS (AU) attributes" on page 643.
SHOW <device>	Displays information on the named device.
SHOW <node>	Displays information on the named node. For more information, see "Node (NO) attributes" on page 676.
SHOW <defined variable>	Displays information on the defined variable, its value and the expression for the variable named. For more information, see "DEFINE (DE) attributes" on page 654.
SHOW <defined path name>	Displays information on the defined path. For more information, see "DEFINE.PATH (DP) attributes" on page 654.
SHOW <defined function>	Displays information for the defined function. For more information, see "DEFINE.FUNCTION (DF) attributes" on page 654.
SHOW <alarm name>	Displays information for the specified alarm. For more information, see "Alarm name (AL) attributes" on page 643.
SHOW <alarm category>	Displays information for the specified alarm category. For more information, see "Alarm category (AC) attributes" on page 643.
SHOW <fluid name>	Displays information on a fluid name (defined in the SCL equation of state). For more information, see "Fluid name (FL) attributes" on page 662.

SHOW MONFLOWS	Displays information the sum of the monitor flows, the sum of the absolute value of the monitor flows, and a few other items of interest for online modeling. For more information, see "Monitor flows (MONFLOWS) (MF) attributes" on page 848 and "MONFLOWS for online modeling" on page 817.
SHOW DPLOT	Displays plot overrides for distance plots.
SHOW TPLOT	Displays plot overrides for time plots.
SHOW RUNNING.MESSAGE	Displays the running message as a defined variable.

Take note

Using the short or long form for the SHOW window

The Show window has two forms:

- *SL.ALL* displays all the peeks (default)
- *SL.SHORT* displays the most commonly used peeks

To change the form of the Show window, poke SL.USE to the appropriate name in quotes:

```
POKE SL.USE = "SL.SHORT"
```

The change is effective for all devices.

Example input

Example 1

List everything that can be displayed (showed) in the model.

```
SHOW
```

Note: Press the Enter to return to the > prompt.

Example 2

Display the Show window for a block valve named BLOK.

```
SHOW BLOK
```

Example 3

Display the Show window for a node named NDE1.

```
SHOW NDE1
```

Example 4

Display the GLOBALS Show window.

```
SHOW GLOBALS
```

SPAWN

Interactive

SPAWN

Description

SPAWN pauses SPS and creates a sub-process at the operating system level. Control is returned to the operating system level, where mail may be read, files may be copied, removed, etc. Return to the simulation by logging out of the sub-process.

Note: For DOS systems, type EXIT to return to the simulation). All spawn commands are logged to the REPLAY and OUTTRN files as comments.

Take note

Useful for displays

The SPAWN command is particularly useful for building text displays for TRANS. Because these displays are text files created by using an editor, you may spawn out of TRANS in order to create a new display or edit an existing one. To see any changes that were made, use “REREAD” on page 555; otherwise the old version of the display is used.

Example input

Spawn out of TRANS.

```
spawn
```

TIMEPLOT

Interactive

```
TIMEPLOT [FIGNAME]
```

Field	Units Key	Description
FIGNAME	n/a	Name of a previously defined time plot

Description

TIMEPLOT is used to create a time plot or open an existing time plot that is in memory. To create a new time plot, enter the command TIMEPLOT only. To edit a time plot, enter TIMEPLOT followed by the name of the plot (i.e., TIMEPLOT TREND1). If you want to replace the version of the plot in memory with a saved version, use "REREAD" on page 555. For more information on creating, displaying, and editing time plots, see "Time plots" on page 179.

Example input

Example 1

Create a new time plot.

```
TIMEPLOT
```

Example 2

Edit an existing time plot named PROG1.

```
TIMEPLOT PROG1
```

Example 3

Display a time plot named PROG1 from any window.

```
PROG1
```

TYPE

Interactive

```
TYPE FILENAME
```

Field	Units Key	Description
FILENAME	n/a	Name of a text file to be displayed. If you type only the file extension (e.g. INPREP, INTRAN, etc.), the command defaults to the name of the model being run and searches in the directories specified in the DREMPATH_DSP variable. See "Display directory (DREMPATH_DSP)" on page 94 to specify in which directories SPS should look for the files. **Note:** You may not specify the HTML OUTPRP report.

Description

TYPE displays the contents of a text file, one page at a time. You cannot edit a file by using the TYPE command.

Typical page size is 22 rows by 80 columns, of which the top 20 rows are available for displaying the file, row 21 is blank, and row 22 displays the prompt for moving through the file. If you want to set the CRT size, see "INTERACTIVE" on page 463.

Depending on which part of the file you are viewing, you may navigate the file using the following options:

Quit	To quit displaying the file, type Q and then press Enter. Control is returned back to the system prompt in the simulation.
Top	To go to the top (first) page in the file, type T and then press Enter.
Previous	To go to the previous page in the file, type P and then press Enter.
Next	To go to the next page, type N and then press Enter or just press Enter.
Bottom	To go to the bottom (last) page in the file, type B and then press Enter.
Events	To obtain a listing of the events that have occurred during the simulation while it is running, type TYPE EVENTS at the command line. Inside the listing, type Q to return to the command line. The OUTTRN file and the EVENTS file contain a complete listing of the events that occurred during the simulation.

Example input

Example 1

List the contents of an INPREP file named EXAMPLE.INPREP while running TRANS interactively.

```
TYPE EXAMPLE.INPREP
```

Example 2

While running a simulation, get a listing of the errors that have occurred so far during the simulation.

```
TYPE EVENTS
```

ZZSTICK

Interactive

```
ZZSTICK
```

Description

ZZSTICK is used to toggle between the two available run modes. If the simulation is paused by default, this mode is STICKYMODE ON. This means that after you enter a command, the > prompt appears and the simulation stays paused until the run command is given. If the simulation is not paused by default, this mode is STICKYMODE OFF.

When a simulation begins, the default mode is paused (STICKYMODE ON). To switch it to STICKYMODE OFF, enter the command ZZSTICK.

Because ZZSTICK is a toggle command, the way to switch back to paused by default is to again enter the command ZZSTICK.

Example input

Example 1

Change the run mode from default paused to default run.

```
ZZSTICK
```

SPS responds with the message:

```
(Stickymode OFF)
```

Example 2

Then change the run mode from default run back to default paused.

```
ZZSTICK
```

SPS responds with the message:

```
(Stickymode ON)
```

CHAPTER

13

User-defined Variables, Macros, and Include Files

The following groups of commands may be used in INPREP and INTRAN files:

- *Macros.* Simplify input of data.
- *Include files.* Organize input into manageable chunks.
- *User-defined variables.* Automatically calculate data not readily available as a standard in SPS.

"Expanding files that contain macros and include statements" on page 585 may be used to expand input files that contain macros or reference include files.

User-defined variables

User-defined variables allow you to automatically calculate data not readily available as a standard in SPS. In addition, user-defined variables may be used as the basis for output for show windows, reports, and plots. For more information, see:

- DEFINE
- DEFINE.PATH

DEFINE

INPREP, INTRAN, and Interactive

```
DEFINE NAME [/INIT/] = EXP[,
+ UPDATE.IF CONDITIONAL]
```

Field	Units Key	Description
NAME	n/a	Name of this user-defined variable. Colons (:) are not permitted in the names of defined variables. In addition, names should start with an alpha character.
INIT	n/a	Initial value of the define
EXP	n/a	A number, peek attribute, defined variable, arithmetic expression of numbers and variables, or a text string enclosed within double quotes (" ").
CONDITIONAL	n/a	Relational test on a numerical or status value

Description

DEFINE allows you to define a user-defined variable, which may be displayed or used to define other variables or trends. Any user-defined value may be poked. Defines are evaluated every time step.

Take note

Output of defined variables

You can output defined variables on a display or print form exactly like any other peek attribute; however, no peek attribute is required after the variable name. By default :VAL is the attribute.

For example, if you defined a variable PIPEMIN that is equal to the from-end pressure on a pipe named PIPE1, then you could obtain that pressure value by entering only PIPEMIN.

Time plot variables

One of the principal uses of the DEFINE is to create a new variable for time plots. Data for all user-defined variables are written to the REVIEW file for plotting interactively (on a time plot) or by using to produce a trend plot *only if* a TRENDLIST command is in the INTRAN file. The keyletters of DE are used for defined variables.

Initialization

User-defined variables are initialized to zero and are not updated until the first time step is taken. To initialize the variable to a non zero value use the INIT option.

Order

Variables can be defined and used in any order. They are evaluated in the order that they are defined.

INPREP versus INTRAN

A variable may be defined in both INPREP and INTRAN. The INTRAN definition overrides and a warning message is written to the OUTTRN file. Defining variables in the INPREP file instead of the INTRAN file saves some TRANS initialization time. Variable definitions created in INPREP are stored in the RESTRT file.

Interactive

The DEFINE may also be used interactively. Interactive DEFINEs are echoed to the REPLAY file, not the OUTTRN file. When the simulation halts, the defined variable is not stored. The defined variable is lost unless the simulation was archived after the variable was defined.

Use of UPDATE.IF

UPDATE.IF indicates that the defined variable is updated to be EXP only when UIF is true. The UIF is pokable.

LOAD.STATUS

If the DEFINE is specified as a constant (single numeric value or text string), the value is updated by the LOAD.STATUS. If a DEFINE is specified as an expression (algebraic or function including sign; for example, -5 is an expression), the value from the ARCHIVE is ignored and the expression is re-evaluated after the LOAD.STATUS. This is true even if the DEFINE was poked to a constant prior to the ARCHIVE.

Example input

Example 1

Define a variable named TORQUE1 that is equal to 5252 multiplied by the ratio of the power of a pump/compressor named UNIT1 to its speed. Initialize the variable at 5000.

```
DEFINE TORQUE1/5000/ = 5252. * (UNIT1:PWR / UNIT1:RPM)
```

Example 2

Define a variable named TDP1 that is the product of TORQUE1 (the name of a previously defined variable) multiplied by the head of a pump/compressor named UNIT1.

```
DEFINE TDP1 = TORQUE1 * UNIT1:HD
```

Example 3

Define a variable named TOTPWR that is the sum of the power used by two pump/compressors. Make the variable equal to zero for all time less than 6 hours.

```
DEFINE TOTPWR = ( TIME > 360 ) * (UNIT1:PWR + UNIT2:PWR)
```

The expression TIME > 360 has a value of zero if it is not true (i.e., TIME ≤ 360) or one if it is true (i.e., TIME > 360). The result is a defined variable that is zero until the simulation time reached 6 hours (360 minutes). After it reaches 360 minutes, the defined variable is equal to the sum of the powers of pumps/compressors UNIT1 and UNIT2.

Example 4

Define a variable named CC to be equal to half the pressure of an external named EIN when the pressure of the external is greater than or equal to 600:

```
DEFINE CC = EIN:P / 2,
+ UPDATE.IF EIN:P >= 600
```

CC is equal to zero until EIN:P reaches a value of 600. After that, CC is equal to 1/2 of the value of EIN:P+. If EIN:P then falls below 600, CC remains constant until EIN:P goes above 600.

Example 5

Define a variable named TXT1 to be the following expression: “DEFINES WORK FOR TEXT STRINGS”.

```
DEFINE TXT1 = "DEFINES WORK FOR TEXT STRINGS"
```

Example 6

Define a variable named MINP that is the minimum of the following three pressures: TL1:P-, TL2:P- and TL3:P-.

```
DEFINE MINP = MIN(TL1:P-,TL2:P-,TL3:P-)
```

DEFINE.PATH

INPREP, INTRAN, Interactive

DEFINE.PATH PATHNAME **= PATH** (NAME$_1$, NAME$_2$, ..., NAME$_j$)

Field	Units Key	Description
PATHNAME	n/a	Name of a user-defined path
NAME$_j$	n/a	Name of a pipe, header, external, valve, pump, and/or node and/or defined path name. NAME$_1$, NAME$_2$, NAME$_3$, ... , NAME$_i$ does not need to be connected to have continuous flow in the path. (The shortest interconnected path between NAME$_1$ and NAME$_2$; NAME$_2$ and NAME$_3$; and so on is found.)

Description

DEFINE.PATH defines a specific path in the model for distance. DEFINE.PATH may be used in either the INPREP or INTRAN file.

Take note

OUTTRN information

If DEFINE.PATH is entered in the INPREP file and PREPR runs successfully, when you run TRANS, the OUTTRN report contains a DEFINE.PATH summary. The summary gives the name and a device list of each path name that is defined.

If you see an asterisk (*) in this section of the OUTTRN, the flow direction through the element is backwards relative to the rest of the path. Flow goes through the tagged element(s) from the downstream side to the upstream side because that is how the element was oriented in INPREP.

Creating a distance plot of a defined path

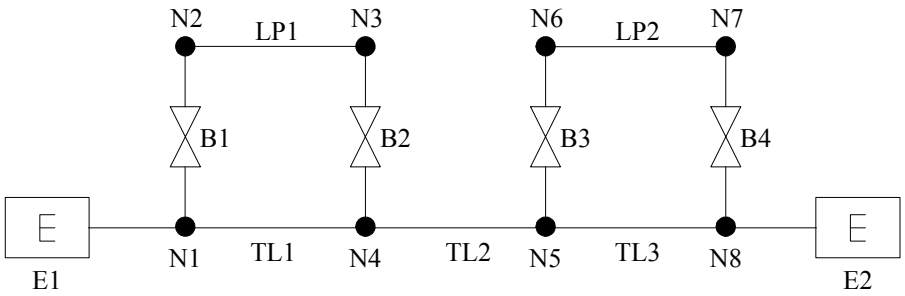
To display the distance plot of the defined path, enter the path name in the from and to fields in the Distance Plot Editor. For more information, see “DISTPLOT” on page 548 and “Distance plots” on page 169.

Define paths nesting

Define paths may be nested. A defined path may refer to another defined path.

Example input

A sample model is as follows:



Specify a path named LOOP that goes from external E1 and through the two loops to external E2:

```
DEFINE.PATH LOOP = PATH(E1, B1, LP1, B2, TL2, B3, LP2, B4, E2)
```

You can view information the path using peek commands. See "DEFINE.PATH (DP) attributes" on page 654. For example, you can determine the number of devices (ND) in the path. The LOOP:ND variable is 9 (as shown in the example). The total number of devices in the path, including nodes (NP), determined through the LOOP:NP variable is 17 (E1, N1, B1, N2, LP1, N3, B2, N4, TL2, N5, B3, N6, LP2, N7, B4, N8, E2). The fourth device specified in the path, LOOP:D(4), is B2. The fourth device, including nodes, specified in the path, LOOP:P(4), is N2.

The following input specifies the same path as above:

```
DEFINE.PATH SHORT = PATH(E1, LP1, LP2, E2)
```

You do not need to enter intermediate points.

Define a path for the first loop and a second path for the second loop and a path similar to the ones above:

```
DEFINE.PATH LOOP1 = PATH(E1,LP1,B2)
DEFINE.PATH LOOP2 = PATH(B3,LP2,E2)
/* PATH THROUGH THE LOOPS FROM END TO END
DEFINE.PATH LOOPNESTED = PATH(LOOP1,LOOP2)
```

INCLUDE

INPREP or INTRAN

```
INCLUDE FILENAME [,[NO]LIST][,[NO]QUIT]]
```

Field	Units Key	Description
FILENAME	n/a	Name of file (data set) to be included in the INPREP or INTRAN file. May include a complete path name to another directory.
LIST	n/a	The included file to be listed in OUTPRP or OUTTRN. {NOLIST}
QUIT	n/a	Indicates that the session should terminate if FILENAME is not found. {Default is QUIT}

Description

INCLUDE allows you to include the contents of any other text file into an INPREP or INTRAN file. Standard, or commonly used, input may be stored in files that are then included anywhere in any number of input files. When information contained in the include files changes, you need only make the change in one place.

Any number of include files may be used. In fact, an input file may consist of a title and then various INCLUDEs. In addition, a file can be included that contains other INCLUDEs. The maximum level of nesting is 10. A file cannot include itself (either directly or indirectly).

Take note

Naming and storing include files

Provide meaningful names for include files. For example, if an include file contains standard MACROs, you might want to use the name STANDARD_MACROS.INC. The extension INC is not required, but using a common extension for include files is a good idea.

Placing include files in a common directory is also a good idea. This helps if the input file and the include file are not in the same directory.

The specification of the FILENAME for inclusion must follow the rules imposed by the operating system on file names and directory paths. The current directory is checked for the FILENAME prior to checking the directory path. If the FILENAME exists in the current directory, it is accessed instead of the FILENAME in the given path. This permits “what ifs” without changing the standard include file. The file name should not be quoted.

Using include files with Model Builder

While using Model Builder, you may either overwrite or preserve include files, depending on your needs. Through Model Builder, you may set an overwrite command on each include file. Then when you save the model, you may choose to observe the setting or override it.

A file generated from Model Builder will be overwritten by default while a manually created file will not be overwritten by default. Model Builder detects whether a file was manually created or generated through Model Builder based on the comment on the second line. A file generated in Model Builder has the following comment in the second line. `/*{sps}`

```
Include file generated ...
```

Note that there is no other syntax for this setting in the INCLUDE file and deleting this comment through a text editor could change the status. If you want to preserve the include file, you should remove the comment line and use Windows features to ensure protection of the file.

Example input

Example 1 - INPREP

Use the INCLUDE command to bring standard input into an INPREP file.

This INPREP input is contained in a file named STANDARD.INC:

```
GAS
ENGLISH
ISOTHERMAL 60
STATE CNGA .62
NOTRACK
CUSTODY PRESSURE = 14.696, TEMPERATURE = 60
PIPEPARMS
+ FRICTION COLE 0.0028
+ INITIAL 300
+ KNOT 2
```

Use STANDARD.INC as part of an INPREP file. These two files are in the same directory:

```
Example Title
INCLUDE STANDARD.INC
=EQUIPMENT
  ...
```

Example 2 - INPREP

Use multiple INCLUDE commands in an INPREP file.

A file named ABOVE_EQUIP.INC contains all the INPREP input needed for a particular model that is above the =EQUIPMENT command (except for the title).

A file name BELOW_EQUIP.INC contains all the INPREP input needed that follows the =EQUIPMENT command.

Use INCLUDE commands to build an INPREP file.

```
SAMPLE TITLE LINE HERE
INCLUDE ABOVE_EQUIP.INC
=EQUIPMENT
INCLUDE BELOW_EQUIP.INC
```

If the include files used in this example contain all the needed input for the model, then the INPREP file contains only four lines. A title needs to be the first input line in the INPREP file.

Example 3 - INTRAN

Use the INCLUDE command to bring standard MACROs into an INTRAN file. The MACROs are contained in a file named INTRAN_MACROS.INC. The INTRAN file and the include file are located in the same directory:

```
...
/* Include standard MACRO file
INCLUDE INTRAN_MACROS.INC.
...
```

Macros

Use of macros helps to simplify input of data. Using macros, you can populate repetitive information and potentially reduce the number of mistakes that typically result from entering data manually and maintain cleaner files with less duplication.

For more information, see:

- IFELSE
- IFISMACRO
- MACRO
- TESTMACRO
- SPS_VERSION

IFELSE

INPREP or INTRAN

```
IFELSE(COMPARE1,COMPARE2,[REPLACE1][,REPLACE2[,LIST]])
```

Field	Units Key	Description
$COMPARE_1$ $COMPARE_2$	n/a	Text or a MACRO that expands into text
$REPLACE_1$ $REPLACE_2$	n/a	Text or a MACRO that expands into text. The text can be other data processing statements or /* comments.
LIST	n/a	Induces extra output in OUTPRP or OUTTRN to be used as a diagnostic aid.

Description

IFELSE is a text substitution facility that may be used in the INPREP and INTRAN files. If the resulting text from $COMPARE_1$ and $COMPARE_2$ match (based on a case-sensitive string comparison), then the IFELSE is replaced with the resulting text from $REPLACE_1$. If there is not a match, then the IFELSE is replaced with the resulting text from $REPLACE_2$.

$REPLACE_2$ is optional. To omit $REPLACE_2$, use one of the following statement formats:

```
IFELSE(COMPARE1,COMPARE2,{REPLACE1})
```

—or—

```
IFELSE(COMPARE1,COMPARE2,{REPLACE1},)
```

Both are acceptable and function in the same manner.

Enclose the $REPLACE_1$ and $REPLACE_2$ text in braces to reduce the chance for errors by making the IFELSE easier to read. Brace syntax replaces the quotation syntax previously used by SPS.

How IFELSE works

The following provides some background on how the IFELSE command works:

- $COMPARE_1$, $COMPARE_2$, $REPLACE_1$, and $REPLACE_2$ are read, and any data processing commands are expanded to create the resulting text. Note that comments are not included in the resulting text.
- After the resulting text from $COMPARE_1$ and $COMPARE_2$ is generated, all leading and trailing blanks are removed.
- The resulting text from $COMPARE_1$ and $COMPARE_2$ are then compared as text. (This is a case-sensitive comparison).
- If the resulting text from $COMPARE_1$ and $COMPARE_2$ *do* match, then the IFELSE is replaced with the resulting text from $REPLACE_1$, and the resulting $REPLACE_2$ is ignored.

- If the resulting text from $COMPARE_1$ and $COMPARE_2$ *do not* match, then the IFELSE is replaced with the resulting text from $REPLACE_2$, and the resulting $REPLACE_1$ is ignored.
- If the comparison of $COMPARE_1$ and $COMPARE_2$ indicate a substitution of an omitted $REPLACE_1$ or $REPLACE_2$, then nothing is substituted for the IFELSE; it is as if it were never there.

Take note

Expanding macros

You may use the DEMAC utility to expand macros. For more information, see "DEMAC" on page 1044.

Example input

Example 1 - INPREP

Use IFELSEs to set up an INPREP file that contains two cases. The first case is for a model that is run in the isothermal mode, and the second case is run in the TRANSTHERMAL mode.

Define a MACRO for use in the test comparison (i.e., COMPARE1, COMPARE2).

```
MACRO(CASE_NAME, ISOTHERM)
```

Changes the case to be run by editing the INPREP file and changing the COMPARE2 text string. In this example, change ISOTHERM to something else to change the case to be run.

Define an IFELSE that sets the TRANSTHERMAL mode to be used in the INPREP file.

```
IFELSE(CASE_NAME, ISOTHERM, {
    =ISOTHERMAL
     },{
     =TRANSTHERMAL
     })
```

Define an IFELSE that adds some additional pipe input if the simulation is to be transient thermal.

```
T PIPE2 +PIPE1 -PIPE3 10. 20. 0.5 60. 60.
+ COLEBROOK 0.0012
IFELSE(CASE_NAME, ISOTHERM,,{
+ WRAP THIK 0.5
+ FILL THIK 24.0
+ GRND THIK 36.0
})
```

The IFELSE in this example has no "if" text substitution but rather only an "else" substitution. (The second comma after ISOTHERM ends any input information for the ISOTHERM case.) If CASE_NAME is ISOTHERM, then the additional three lines of pipe data are not entered.

To cause the transient thermal input to be included, change the MACRO definition.

```
MACRO(CASE_NAME, TRANSTHERM)
```

Because of the “else” construction of the IFELSE, any entry in the case name MACRO other than ISOTHERM results in the transient thermal input being included.

```
MACRO(CASE_NAME, ANYTHING)     /* same effect as previous MACRO
```

Example 2 - INPREP

Create an IFELSE that either includes or excludes a block valve depending on the case to be run. Case 1 is with the valve, and Case 2 is without.

Define a MACRO to set up the case number switch.

```
MACRO(case_num, case_1)
```

Define an IFELSE that includes or excludes a block valve.

```
IFELSE(case_num, case_1,{
    B BLOCK +PIPE1 -PIPE2 OPEN2 CLSE2 0 1000 1
    },)
```

This IFELSE does not have an “else” substitution. Nothing is included if the case_num is not case_1. This is because nothing is after the last comma before the closing parenthesis.

Example 3 - INTRAN

Set up an INTRAN file for running two simulation cases, each with its own simulation scenario, and use the IFELSE command to determine which case is run.

```
MACRO(CASE_NUM, CASE_1)
/*MACRO(CASE_NUM, CASE_2)
IFELSE(CASE_NUM, CASE_1,
      { START UNIT1, TIME = 5
       STOP UNIT2, TIME = 10
       POKE INLET:P+ = 750, TIME = 30
       }
       ,
        { START UNIT1, TIME = 10
         POKE INLET:P+ = 800, TIME = 60
        }
       )
```

Choose which string ($REPLACE_1$ or $REPLACE_2$) becomes part of the INTRAN file by commenting out the unwanted MACRO statement.

IFISMACRO

INPREP or INTRAN

```
IFISMACRO(NAME,[{REPLACE1}][,REPLACE2[,LIST]])
```

Field	Units Key	Description
NAME	n/a	Text or a MACRO that expands into text
$REPLACE_1$ $REPLACE_2$	n/a	Text or a MACRO that expands into text. The text can be other data processing statements or /* comments.
LIST	n/a	Induces extra output in OUTPRP or OUTTRN.

Description

IFISMACRO is a text substitution facility that checks to see if the MACRO is defined. It may be used in INPREP and INTRAN files. If the resulting text from NAME matches (based on a case sensitive string comparison), then the IFISMACRO is replaced with the resulting text from $REPLACE_1$. If there is not a match, then the IFISMACRO is replaced with the resulting text from $REPLACE_2$.

$REPLACE_2$ is optional. To omit $REPLACE_2$, use the following statement format:

```
IFISMACRO(NAME,REPLACE1)
```

Enclose the $REPLACE_1$ and $REPLACE_2$ text in braces to reduce the chance for errors by making the IFISMACRO easier to read.

How IFISMACRO works

The following provides some background on how the IFISMACRO command works:

- $REPLACE_1$, and $REPLACE_2$ are read, and any data processing commands are expanded to create the resulting text. Note that comments are not included in the resulting text.
- If the resulting text from NAME matches a MACRO, then the IFISMACRO is replaced with the resulting text from $REPLACE_1$, and the resulting $REPLACE_2$ is ignored.
- If the resulting text from NAME does not match a MACRO, then the IFELSE is replaced with the resulting text from $REPLACE_2$, and the resulting $REPLACE_1$ is ignored.
- Typically REPLACE will be the MACRO call with associated variables.

Take note

Expanding macros

You may use the DEMAC utility to expand macros. For more information, see "DEMAC" on page 1044.

MACRO

INPREP or INTRAN

MACRO(SHORT-FORM[(ARG$_1$...ARG$_n$)], LONG-FORM)

Field	Units Key	Description
SHORT-FORM	n/a	Short form of text (MACRO name)
ARG$_j$	n/a	Optional arguments passed when the MACRO is invoked
LONG-FORM	n/a	Long form of text

Description

MACRO is a way to substitute one text string for another. This simple concept of text substitution can be used to great advantage throughout the INPREP and INTRAN files.

MACRO has the following uses:

- Define custom input formats.
- Shorten standard input formats by combining MACROs with standard input.
- Develop quick input forms for a single input item or for a combination of input items.
- Define a generic set of commands for use in more than one model. These generic commands can be customized by passing arguments. They may also be stored in separate files then may be brought into an input file using the INCLUDE command.
- Establish test criteria for use with an IFELSE command.
- Define abbreviations for interactive TRANS.
- Define INIT display for TRANS.

The use of MACROs is a two-step process. To use a MACRO, you must first define it, and then invoke it. You define a MACRO with the actual MACRO command, then invoke a MACRO, using the short form (or MACRO name), further down in the file. MACRO definitions may appear anywhere in an input file, but the definition must appear sequentially before the MACRO is invoked.

The MACRO must be called with exactly the same number of arguments for which it was defined.

Take note

MACRO libraries

A useful concept is that of the MACRO library. A MACRO library is a collection of MACROs for frequently used input. The MACROs are stored in one or more separate files. The MACROs in the library are referenced by using the INCLUDE command to include the MACRO files in the input files where the MACROs are needed.

INCLUDE statement

A MACRO may contain one or more INCLUDE statements.

MACRO names

The name of a MACRO defined in the INPREP file should not be the name of a node or element or a fluid name. Special characters ("_ . $ #) are allowed in MACRO names. For MACRO parameters, % may also be used.

INTERACTIVE

MACROS may also be created interactively. The interactive MACRO command must be uppercase. When a simulation is terminated in any manner, all MACROs that have been created interactively are lost.

Expanding macros

You may use the DEMAC utility to expand macros. For more information, see "DEMAC" on page 1044.

Example input

Example 1 - INPREP

Define a MACRO that replaces the standard input for a pipe named PIPE2. Typical input for a pipe.

```
T PIPE2 NODE1 NODE2 10.0 20.0 0.25 60 60
+ ELEVATION 200 300
```

Now create a MACRO for a pipe. This MACRO assumes that all pipes have the same diameter, wall thickness, end temperatures, and roughness. Also assumed is that two elevations points are sufficient.

Define the MACRO.

```
MACRO(T_LINE(NAME,UP_CON,DN_CON,LEN,ELEV1,ELEV2),
 T NAME UP_CON DN_CON LEN 20.0 0.25 60 60
 + ELEVATION ELEV1 ELEV2
       )
```

Invoke the MACRO.

```
T_LINE(PIPE2,NODE1,NODE2,10,200,300)
```

Invoking the MACRO in this manner is exactly the same as this example. However, once the MACRO has been defined it may be invoked as many times as you want for as many pipes as you want.

Invoke the MACRO again (for a second pipe named PIPE3).

```
T_LINE(PIPE3,NODE5,NODE6,27.2,530,590)
```

Example 2 - INPREP

Be careful when defining MACROs that the arguments for the MACRO match only the text that you want to substitute. Avoid using short names for arguments because you may experience some unexpected matches, as in the following example.

```
MACRO(DEFSUM(D,E,F),    /* DO NOT USE THIS STYLE
DEFINE D = E + F
)
```

The MACRO is then called.

```
DEFSUM(A, FLOW1, FLOW3)
```

This MACRO expands to the following.

```
AFLOW1FLOW2INFLOW1 A = FLOW1 + FLOW2
```

Every instance of D is replaced with A, every instance of E is replaced with FLOW1, and every instance of F is replaced with FLOW2. One way to prevent this type of problem is to use special characters in the definition.

```
MACRO(DEFSUM(%D%,%E%,%F%),
DEFINE %D% = %E% + %F%
)
```

When this MACRO is called as before the following results.

```
DEFINE A = FLOW1 + FLOW2
```

Example 3 - INTRAN

When an interactive run begins, TRANS looks for a display called INIT.DSP. If TRANS finds it, then it displays initially when the model begins running. Instead of having a display called INIT, you can define an existing display, report, or figure as the INIT display as in the following. This is done with a MACRO statement like the following.

```
MACRO(INIT, OVERVIEW SHOW TL2 SHOW B1 SHOW TL1 REP1)
```

TRANS displays OVERVIEW.DSP as the initial display instead of INIT.DSP, along with the SHOW screens for TL2, B1, and TL1 and REP1. Note that REPORTS do not need a preceeding SHOW command.

For more information, see "To automatically open a display when you run TRANS" on page 164.

Example 4 - INTRAN

Define abbreviations for interactive commands.

```
MACRO(RR, RUN)
MACRO(HH, HALT)
MACRO(SS, SHOW)
```

In the interactive mode, you can type RR to run the simulation, HH to halt the simulation, and SS to display a Show window. The MACRO names are case sensitive.

Example 5 - INTRAN

Establish test criteria for use with the IFELSE command.

```
MACRO(CASE_NUM, CASE_1)
/*MACRO(CASE_NUM, CASE_2)
IFELSE(CASE_NUM, CASE_1,
        { START UNIT1, TIME = 5
         STOP UNIT2, TIME = 10
         POKE INLET:P+ = 750, TIME = 30
```

```
 }
,
 { START UNIT1, TIME = 10
  POKE INLET:P+ = 800, TIME = 60
 }
 )
```

In this example UNIT1 is started at TIME = 5, etc. If the commented MACRO were active, and the MACRO(CASE_NUM,CASE_1) were commented out, then UNIT1 starts at TIME = 10.

TESTMACRO

INPREP or INTRAN

TESTMACRO(CONDITIONAL,[REPLACE$_1$][,REPLACE$_2$[,LIST]])

Field	Units Key	Description
CONDITIONAL	n/a	A simple relational expression comparing two numbers, either of which may be a MACRO. Supported operations are: < <= == != >= >
REPLACE$_1$ REPLACE$_2$	n/a	Text or a MACRO that expands into text. The text can be other data processing statements or /* comments.
LIST	n/a	Induces extra output in OUTPRP or OUTTRN.

Description

TESTMACRO is a text substitution facility that compares two numbers, either of which may be a MACRO. It may be used in INPREP or INTRAN files. If CONDITIONAL is true, then the TESTMACRO is replaced with REPLACE$_1$. If CONDITIONAL is false, then the TESTMACRO is replaced with REPLACE$_2$.

REPLACE$_2$ is optional. To omit REPLACE$_2$, use the following statement format:

TESTMACRO(CONDITIONAL,REPLACE$_1$)

How TESTMACRO works

The following provides some background on how the TESTMACRO command works:

- REPLACE$_1$ and REPLACE$_2$ are read, and any data processing commands are expanded to create the resulting text. Note that comments are not included in the resulting text.
- If the resulting text from CONDITIONAL is true, then the TESTMACRO is replaced with the resulting text from REPLACE$_1$, and the resulting REPLACE$_2$ is ignored.
- If the resulting text from CONDITIONAL is not true, then the IFELSE is replaced with the resulting text from REPLACE$_2$, and the resulting REPLACE$_1$ is ignored.

Take note

Expanding macros

You may use the DEMAC utility to expand macros. For more information, see “DEMAC” on page 1044.

SPS_VERSION

```
SPS_VERSION
```

Description

SPS_VERSION is a built-in macro intended to ease transition from one SPS version to the next. The SPS_VERSION macro is replaced by the current version you are running. This is valid for SPS 9.20 and greater.

Example input

Use SPS_VERSION and TESTMACRO to create version-specific logic. SPS will evaluate the expression based on the version in use. The :OHTC pipe attribute was added in release 9.4. If you are using 9.4 or greater, the expression evaluates to PIPE1:OHTC. For earlier versions, the expression evaluates to 500.

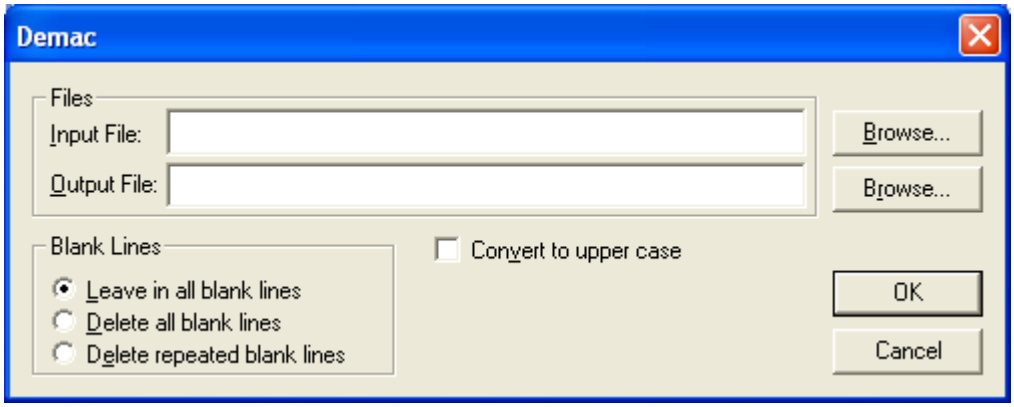
```
DEFINE PIPE1_OHTC = TESTMACRO( SPS_VERSION >= 9.4, PIPE1:OHTC, 500)
```

Expanding files that contain macros and include statements

DEMAC is an SPS utility used to process INPREP and INTRAN files to expand IFELSE, INCLUDE and MACRO commands to their full text equivalents. For more information on using the DEMAC utility on Windows, see "To DEMAC a file" on page 585. For more information on running DEMAC from the command line, see "DEMAC" on page 1044.

To DEMAC a file

1 From the SPS startup window, select **Tools > Demac**.



2 In the **Demac window**, specify the **Input File** to be expanded.

3 Specify the **Output File** to be created.

Tip: If you click **Browse**, you can either navigate to an existing file to be overwritten, or type the path of a new file, including the extension, to be created.

4 Select options for blank lines and letter case conversion.

Note: The default case convention is to leave the cases as they are.

5 Click **OK**.

CHAPTER

14

Expressions, Operators, and Functions

Operators and functions that may be used with SPS commands to express logical controls are contained in this section. For easy reference, all of the functions have been organized in alphabetical order.

Expressions

An expression is not a command but rather an important part of a command. An expression may be a simple algebraic expression, such as “y + 5”, or it may be more involved, consisting of simple algebra together with logical or relational operators and special functions. The order in which operators in an expression are evaluated follows standard mathematical rules for precedence. Parentheses may be used to explicitly establish the order. For more information on valid operators, see “Relational and logical operators” on page 590.

Conditional logic

You can apply conditional logic through if-then-else construction. The general syntax is:

```
Condition ? valueIfTrue : valueIfFalse
```

This expression can be used anywhere any other expression can be used. For example, in a defined variable:

```
DEFINE A = B ? C : D
```

which means that if B is true (that is, B is not 0), set A to C; otherwise, set A to D.

For more complex expressions, use this multi-line style for clarity:

```
DEFINE STN_ABC_PRES_LIMIT = ABC:P+ > MAX(ABC1:P-, ABC2:P-)
+      ? ABC3:P+ + 15
+      : ABC4:P- + 10
```

In this example:

- Condition is `ABC:P+ >MAX (ABC:P-, ABC2:P-)`
- valueIfTrue is `ABC3:P+ + 15`
- valueIfFalse is `ABC4:P- + 10`

Multiple conditional expressions may be included in a single DEFINE. See “Example 4” in “DEFINE” on page 568.

Note: The entire expression is evaluated every time step, so special care must be taken to avoid mathematical errors (see "Example 5" under "Examples of expressions" on page 588).

Multiple colon syntax

Expressions also support names with multiple colons. A name with multiple colons, such as A:B:C, is evaluated left to right. First, A:B is evaluated. If A:B evaluates to the name of another device, then that device's :C field is used as the result of the A:B:C expression.

Typical applications for this notation may include online models where it is helpful to see which device is being read from which monitor through the SCADA. The :CB "controlled by" attribute commonly evaluates to another device. The expression A:CB:REP first expands to the device that is the :CB field for A, then gets the :REP for that device.

Names with multiple colons may be created within the SPS for Windows Report Editor. See "To create or edit a Report" on page 183. When selecting attributes for any given device, a second attribute is inserted after the colon.

For example, in the Report Editor, select the monitor (E) as the equipment type, then select the CB keyword from the report items, then manually type :EREP after the CB keyword. In the custom display, the selected attribute would retain this colon notation – CB:EREP.

Multiple colon expressions may also be declared using the DEFINE statements in either the INPREP or INTRAN files (see "Example 6" under "Examples of expressions" on page 588).

Examples of expressions

Example 1

Define a variable named TSECS equal to the simulation time (the default is minutes) expressed in seconds:

```
DEFINE TSECS = TIME * 60
```

Example 2

Poke the pressure of an external named EXT1 to a value equal to some other pressure plus 100 at a simulation time of 60 minutes. The other pressure in this example is given by a previously defined variable named OTHPRESS.

```
POKE EXT1:P = OTHPRESS + 100, TIME = 60
```

Example 3

Define a variable named VAL1 that has a value of either 0 or 100 depending on whether a second variable named VAL2 (previously defined) is greater than 0. If VAL2 is greater than zero, VAL1 is equal to zero. If VAL2 is less than or equal to zero, VAL1 is equal to 100:

```
DEFINE VAL1 = ( VAL2 <= 0 ) * 100
```

In this example the part of the expression inside the parentheses has a numerical value of either one or zero depending on whether the inequality is true or false. If the inequality is true, VAL2 is <= 0, then the parenthetical expression has a value of 1 and the defined variable VAL1 has a value of 100 (because 1 * 100 = 100). If the inequality is false, VAL2 is >

0, then the parenthetical expression has a value of 0 and the defined variable VAL1 has a value of 0 (because 0 * 100 = 0).

This example illustrates how a relational test produces a numerical result in an expression in SPS.

Example 4

Define a variable named BLKST that passes a numerical value for valve BLK status:

```
DEFINE BLKST =
+ BLK:ST == OPENED ? 1 :
+ BLK:ST == CLOSED ? 0 : 2
```

For this example, if valve BLK is opened, BLKST has a value of 1. If valve BLK is closed, BLKST has a value of 0. Any other status (e.g. OPENING) results in BLKST having a value of 2.

Example 5

Define a variable that computes the average flow at an external over the last 24 hours. Re-initialize the variable every morning at 7 AM.

```
DEFINE.TIMETABLE AEXT.SC {
REPEAT = DAILY
START.INTERVAL = "07:00"
} /* End DEFINE.TIMETABLE
DEFINE AEXT.ACCUM/0/ = TINT(EXT:Q)
DEFINE AEXT.TIME/0.00001/ = AEXT.TIME +DT
DEFINE AEXT.ACCUM.OLD/0/ = AEXT.ACCUM,
+ UPDATE.IF AEXT.SC:SCHED = 1.0
DEFINE AEXT.TIME.OLD/0/ = AEXT.TIME,
+ UPDATE.IF AEXT.SC:SCHED = 1.0
DEFINE AEXT = ( (AEXT.TIME - AEXT.TIME.OLD) = 0)
+ ? 0
+ : (AEXT.ACCUM - AEXT.ACCUM.OLD) / (AEXT.TIME - AEXT.TIME.OLD)
```

In this example, at 07:00, the evaluation of AEXT has a divide by zero error even though an if-then-else is used to check to see if the change in time is zero. This is because the entire expression is evaluated and not just the valid pieces according to the “if” conditional. A message similar to the following appears in the OUTTRN file each time an error occurs in this DEFINE.

```
Error, line nnnn, directive DEFINE, name AEXT.
```

The error can be eliminated by making the following change to the DEFINE for AEXT.

```
DEFINE AEXT = ( (AEXT.TIME - AEXT.TIME.OLD) = 0 )
+ ? 0
+ : (AEXT.ACCUM - AEXT.ACCUM.OLD) / MAX(AEXT.TIME - AEXT.TIME.OLD,DT)
```

In this example, the denominator of the “else” expression is never equal to zero.

Example 6

Using multiple colon syntax, show two different DEFINE notations for the repeatability correction term (RC) for SCADA points driving monitor pressure or external flows. Confirm output values using the Show window.

Place the following two expressions in either the INPREP or INTRAN file, assuming MON_IN is the name of the external and INLET.P is the name of the SCADA.

```
DEFINE READING_1 = MON_IN:CB:RC
DEFINE READING_2 = INLET.P:RC
```

While running TRANS, selecting Device Type “define” in the Show window, and selecting Device Names Reading_1 and Reading_2 should open two windows which show the same value for both the defined variables.

Example 7

Using multiple colon syntax, show two different DEFINE notations for the repeatability correction term (RC) for SCADA points driving monitor pressure or external flows. Confirm output values using the Show window.

Place the following two expressions in either the INPREP or INTRAN file, assuming MON_IN is the name of the external and INLET.P is the name of the SCADA.

```
DEFINE READING_1 = MON_IN:CB:RC
DEFINE READING_2 = INLET.P:RC
```

While running TRANS, selecting Device Type “define” in the Show window, and selecting Device Names Reading_1 and Reading_2 should open two windows which show the same value for both the defined variables.

Relational and logical operators

Relational and logical operators used in an expression result in all or some part of the expression having a numerical value of either zero or one. A relational or logical expression that is FALSE is converted to 0 when used in a numeric context. Likewise, a relational or logical expression that is TRUE is converted to 1 when used in a numeric context.

The practical application of relational and logical operators is that the numerical value of an expression can be dependent on some relational or logical test (see “Example 3” under “Examples of expressions” on page 588).

Care must be taken when using the relational operators = and != because the various values reported by SPS are the results of various numerical procedures with built-in approximations and convergence criteria.

Mathematical operators

Note: The + and - operators must be preceded and followed by a blank space.

Operation	Operator
Addition	+
Subtraction	-
Multiplication	*

Operation	Operator
Division	/
Raise to a power	**
Parenthesis	()

Relational operators

Operation	Operator
Less than	<
Less than or equal	<=
Greater than	>
Greater than or equal	>=
Equal	= (or ==)
Not equal	!= (or ~=)

Logical operators

Operation	Operator
And	&
Or	|
Not	! (or ~)
If/Then/Else	? :

String operators

Operation	Operator
Concatenation	+
Equal	= (or ==)
Not equal	!= (or ~=)
If/Then/Else	? :

Functions

Functions are used in expressions in all input files. This section documents the functions available for use.

Function	Argument type(s)	Return type	Return value
"ABS" on page 594	numeric	numeric	Absolute value
"AVG" on page 595	history data	numeric	Arithmetic average
"CEIL" on page 597	numeric	numeric	Rounds up to the nearest integer
"COUNT" on page 598	history data	numeric	Count of the historical points
"DELTA_ENTHALPY" on page 599	node and numeric	numeric	Change in enthalpy from given pressures and temperatures
"DENS" on page 600	node and numeric	numeric	Density of fluid at specified pressure and temperature
"DENS_DDP" on page 601	node and numeric	numeric	Derivative of density with respect to pressure
"DENS_DDT" on page 602	node and numeric	numeric	Derivative of density with respect to temperature
"DESCRIPTION" on page 603	peek name	text	Description of a peek name
"DEVICELIST" on page 604	string	device list	Returns a list of devices based on a string
"FLOOR" on page 606	numeric	numeric	Rounds down to the nearest integer
"HEAT_CAPA" on page 607	node and numeric	numeric	Heat capacity of fluid
"HISTORY" on page 608	numeric	history data	History data for a specified number of steps
"INT" on page 609	numeric	numeric	Rounds towards zero
"INTEGRATE" on page 610	history data, numeric	numeric	Integral of the history item over a specified time interval
"INTERNAL" on page 611	pipe and numeric	numeric	Value of a distance plot variable at a specified location
"ISPEEK" on page 612	string	logical	TRUE(=1) if the argument resolves to a peek name; FALSE(=0) otherwise
"LN" on page 613	numeric	numeric	Natural logarithm
"LOOKBACK" on page 614	history data, numeric	numeric	Historical value of the history data at a specified time
"MAX" on page 615	numeric	numeric	Maximum of two or more variables
"MAXDIFF" on page 616	distance plot items	numeric	Maximum difference of two pipeline distance plot variables

Function	Argument type(s)	Return type	Return value
"MIN" on page 618	numeric	numeric	Minimum of two or more variables
"MOD" on page 619	numeric	numeric	Remainder after division
"MOVING_AVG"	numeric	numeric	Average over the specified time at the given interval
"PEEKLIST" on page 621	string	peek list	Array of peek names that satisfies specified conditions
"PREV" on page 623	numeric or string	numeric or string	Previous value of a variable
"SCRAPER" on page 624	path and numeric	numeric	Scraper (pig) tracking function
"STDV" on page 626	history data	numeric	Standard deviation
"SUM" on page 627	peek list	numeric	Sum of the variables given
"SUMA" on page 628	peek list	numeric	Absolute sum of the variables given
"TAVE" on page 629	numeric	numeric	Running average of a variable over a decay period
"Time function operators" on page 632	time	numeric	Various functions to manipulate time values
"TIME_HISTORY" on page 633	numeric	history data	History data for a specified time interval
"TINT" on page 634	numeric	numeric	Integral of a variable over time
"TOSTR" on page 635	numeric	string	Converts expressions to strings
"TRUNC" on page 636	string	string	Truncates a character string
"UNITS" on page 637	peek name	string	Units of a peek name

ABS

```
ABS (VALUE)
```

Field	Description
VALUE	Peek name or expression.

Description

This function calculates the absolute value of VALUE.

Example input

Set ABS_DP equal to the absolute value of the :DP for valve VALVE1.

```
DEFINE ABS_DP = ABS(VALVE1:DP)
```

AVG

```
/* time history input
AVG(THIST(VALUE, TP))

/* multiple value input
AVG(VALUE1 ,..., VALUEn)

/* pipe input
AVG(PIPE:DPLOT_ITEM)

/* path input
AVG(PATH:DPLOT_ITEM)
```

Field	Description
THIST	Expression representing TIME_HISTORY or HISTORY
VALUE	Peek name or expression. When used in the multiple value example, at least two arguments are required.
TP	Time period over which data is accumulated.
PIPE	Pipe for which internal information is requested.
PATH	Path for which internal information is requested. See "DEFINE.PATH" on page 571 for more information.
DPLOT_ITEM	Any valid distance plot item. For standard distance plot variables, see "Distance plot items" on page 172.

Description

When used in conjunction with the TIME_HISTORY or HISTORY function, the AVG function calculates the arithmetic average of VALUE over the time period. When used with a pipe or path as an argument, the AVG function calculates the arithmetic average of a distance plot item over the length of a pipe or path, respectively.

Example input

Time history example

Set AVG_TEMP equal to the average of the last five minutes of values from TEMP_DN.

```
DEFINE THIST_TEMP_DN = TIME_HISTORY(TEMP_DN, 5)
DEFINE AVG_TEMP = AVG(THIST_TEMP_DN)
```

Alternatively, you may use the following syntax.

```
DEFINE AVG_TEMP = AVG(TIME_HISTORY(TEMP_DN, 5))
```

Pipe example

Set PIPE1_AVG equal to the average pressure across PIPE1.

```
DEFINE PIPE1_AVG = AVG(PIPE1:PRESSURE)
```

Path example

Set PATH1_AVG equal to the average pressure across the path from PIPE1 to PIPE6.

```
DEFINE.PATH PATH1 = PATH(PIPE1, PIPE6)
DEFINE PATH1_AVG = AVG(PATH1:PRESSURE)
```

CEIL

```
CEIL(VALUE)
```

Field	Description
VALUE	Peek name or expression.

Description

This function rounds a real number up to the nearest integer.

Example input

Round DP up to the nearest integer value.

```
DEFINE TEST = CEIL(DP)
```

As a result, if DP = 1.2, then TEST = 2, and if DP = -1.2, then TEST = -1.

COUNT

```
COUNT(function)
```

Field	Description
function	A TIME_HISTORY or HISTORY data filter

Description

This function is used in conjunction with the TIME_HISTORY or HISTORY function to return a count of the number of points in the function.

Example input

Define TEMP_STABLE.COUNT as the number of points in the TIME_HISTORY in the last five minutes.

```
DEFINE THIST_TEMP_STABLE = TIME_HISTORY(TEMP_DN,5)
DEFINE TEMP_STABLE.COUNT = COUNT(THIST_TEMP_STABLE)
```

Alternatively, you may use this syntax.

```
DEFINE TEMP_STABLE.COUNT = COUNT(TIME_HISTORY(TEMP_DN,5))
```

DELTA_ENTHALPY

```
DELTA_ENTHALPY(NODE, P1, T1, P2, T2[, DCHDP1, DCHDT1, DCHDP2, DCHDT2])
```

Keyword	Units	Description
NODE	n/a	Node from which density is to be determined.
P1	PRESSURE	Pressure for enthalpy calculation (either a constant or a peek).
T1	TEMPERATURE	Temperature for enthalpy calculation (either a constant or a peek).
P2	PRESSURE	Pressure for enthalpy calculation (either a constant or a peek).
T2	TEMPERATURE	Temperature for enthalpy calculation (either a constant or a peek).
DCHDP1	n/a	Partial derivative of change in enthalpy with respect to P1.
DCHDT1	n/a	Partial derivative of change in enthalpy with respect to T1.
DCHDP2	n/a	Partial derivative of change in enthalpy with respect to P2.
DCHDT2	n/a	Partial derivative of change in enthalpy with respect to T2.

Description

This function is used to calculate the change in enthalpy and/or partial derivatives based on the given pressures and temperatures.

Take note

If they are specified, the values DCHDP1, DCHDT1, DCHDP2, and DCHDT2 are DEFINEs that must be defined before being called. The DEFINEs are then updated with the values.

Example input

Calculate the change in enthalpy of the composition at NODE2 when changing from the pressure and temperature at ND1 to the pressure and temperature at ND3.

```
DEFINE DE2 = DELTA_ENTHALPY(NODE2, ND1:P, ND1:T, ND3:P, ND3:T)
```

DENS

```
DENS(NODE[, PRES, TEMP])
```

Keyword	Unit	Description
NODE	n/a	The composition at this node is used for the density calculation.
PRES	PRESSURE	Pressure for density calculation (either a constant or a peek). {node pressure}
TEMP	TEMPERATURE	Temperature for density calculation (either a constant or a peek). {node temperature}

Description

This function is used to calculate the density of the fluid at a node.

Example input

Calculate the density of the NODE1 composition using the NODE3 pressure and temperature.

```
DEFINE RHO1 = DENS(NODE1, NODE3:P, NODE3:T)
```

DENS_DDP

```
DENS_DDP(NODE[, PRESS, FLOW])
```

Keyword	Units	Description
NODE	n/a	The composition at this node is used for the density derivative calculation.
PRES	PRESSURE	Pressure for density derivative calculation (either a constant or a peek). {node pressure}
TEMP	TEMPERATURE	Temperature for density derivative calculation (either a constant or a peek). {node temperature}

Description

This function is used to calculate the derivative of density with respect to pressure at constant temperature.

Example input

Calculate the derivative of density with respect to pressure of the NODE1 composition using the NODE3 pressure and temperature.

```
DEFINE DELTA_RHOP = DENS_DDP(NODE1, NODE3:P, NODE3:T)
```

DENS_DDT

```
DENS_DDT(NODE[, PRES, FLOW])
```

Keyword	Units	Description
NODE	n/a	The composition at this node is used for the density derivative calculation.
PRES	PRESSURE	Pressure for density derivative calculation (either a constant or a peek). {node pressure}
TEMP	TEMPERATURE	Temperature for density derivative calculation (either a constant or a peek). {node temperature}

Description

This function is used to calculate the derivative of density with respect to temperature at constant pressure.

Example input

Calculate the derivative of density with respect to temperature of the NODE1 composition using the NODE3 pressure and temperature.

```
DEFINE DELTA_RHOT = DENS_DPT(NODE1, NODE3:P, NODE3:T)
```

DESCRIPTION

```
DESCRIPTION(PEEKNAME)
```

Field	Description
PEEKNAME	A peek name.

Description

This function returns the description corresponding to a peek name.

Example input

Get the descriptive text for VALVE2:ST

```
DEFINE DESC = DESCRIPTION(VALVE2:ST)
```

As a result, DESC will be the text “Status”.

DEVICELIST

```
DEVICELIST(" dev_names_incl
[DEVICES.EXCLUDE = dev_names_excl,]
[KEYLETTERS.MATCH = dev_kl_include,]
[KEYLETTERS.EXCLUDE = dev_kl_exclude,]
[SUBTYPES.MATCH = subtypes_include,]
[SUBTYPES.EXCLUDE = subtypes_exclude,]
[ECHO = echo"])
```

Field	Units Key	Description
dev_names_incl	n/a	Devices to include, by name. Wildcards are permitted. See “Wildcards” on page 50.
dev_names_excl	n/a	Devices to exclude, by name. Wildcards are permitted. See “Wildcards” on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See “Device keyletters” on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See “Device keyletters” on page 47.
subtypes_include	n/a	Device subtypes (STYP attribute values) to include. For valid STYP values, see “Peek and Poke Keyletters and Attributes” on page 639.
subtypes_exclude	n/a	Device subtypes (STYP attribute values) to exclude. For valid STYP values, see “Peek and Poke Keyletters and Attributes” on page 639.
echo	n/a	Indicates whether to produce a list of all devices that are included. Valid values are YES and NO.

Description

The DEVICELIST function returns a list of device names that satisfy all of the conditions specified in the function. Before using DEVICELIST, note the following:

- The conditions specified in the DEVICELIST function must be enclosed in quotation marks (" "). Generally, you should use wildcards to form these patterns or conditions. For more information, see “Wildcards” on page 50.
- The DEVICELIST function is set up as an expression within the DEFINE command.
- Multiple DEVICELIST defines are allowed in the INTRAN file.
- The DEVICELIST argument is limited to one line in the INTRAN file (225 characters).

Take note

Cascading effect of conditions

Most of the arguments in a DEVICELIST functions are used to determine precisely which devices should be included in the final list. These arguments act as conditions which function in a cascading manner, each further narrowing the “list” as altered by the previous argument. For example, consider the following DEVICELIST example:

```
DEFINE DL1 = DEVICELIST("*, DEVICES.EXCLUDE = *X*, KEYLETTERS.MATCH = T")
```

In this case, the original DEVICELIST function includes all devices in the model. The DEVICES.EXCLUDE removes any devices with an "X" in the name. The final argument further narrows the list to transfer lines only.

FLOOR

```
FLOOR(VALUE)
```

Field	Description
VALUE	Peek name or expression.

Description

This function rounds a numeric value down to the nearest integer.

Example input

Round DP down to the nearest integer value.

```
DEFINE TEST = FLOOR(DP)
```

As a result, if DP = 1.2, then TEST = 1, and if DP = -1.2, then TEST = -2.

HEAT_CAPA

```
HEAT_CAPA(NODE[, PRESS, FLOW])
```

Keyword	Units	Description
NODE	n/a	Node from which heat capacity is to be determined.
PRESS	PRESSURE	Pressure for heat capacity calculation (either a constant or a peek). {node pressure}
TEMP	TEMPERATURE	Temperature for heat capacity calculation (either a constant or a peek). {node temperature}

Description

This function is used to calculate the heat capacity at constant volume.

Example input

The heat capacity at NODE1 at the NODE3 pressure and temperature is desired.

```
DEFINE HCAP1 = HEAT_CAPA(NODE1, NODE3:P, NODE3:T)
```

HISTORY

```
HISTORY(VALUE, NP)
```

Field	Description
VALUE	Peek name
NP	Number of points

Description

This function may be used with the other functions (AVG, COUNT, INTEGRATE, LOOKBACK, and STDV) or independently to specify the name and number of points that may be used in a calculation. You may specify a larger history range than you intend to use in a calculation.

Take note

Because the time step (DT) can vary dramatically, in most cases you will want to use TIME_HISTORY and INTEGRATE instead of HISTORY and AVG for these types of calculations.

Example input

Create a user-defined variable called P_HIST that keeps three steps of history data for NODE2:P. Then create another user-defined variable that finds the average pressure of these values.

DEFINE P_HIST = HISTORY(NODE2:P, 3)

DEFINE P_AVE = AVG(P_HIST)

INT

```
INT(VALUE)
```

Field	Description
VALUE	Peek name or expression.

Description

This function rounds VALUE towards zero.

Example input

Round DP towards zero.

```
DEFINE TEST = INT(DP)
```

As a result, if DP = 1.2, then TEST = 1, and if DP = -1.2, then TEST = -1.

INTEGRATE

```
/* time history input
INTEGRATE(THIST, [HOWFAR])

*/ distance input
INTEGRATE(PIPE:DPLOT_ITEM)

/* path input
INTEGRATE(PATH:DPLOT_ITEM)
```

Field	Description
THIST	Expression representing TIME_HISTORY or HISTORY.
HOWFAR	Time period to which the simulation should look back. If no value is given, the function will integrate all the available data.
PIPE	Pipe for which internal information is requested.
PATH	Path for which internal information is requested. See "DEFINE.PATH" on page 571 for more information.
DPLOT_ITEM	Distance plot item to be integrated over the specified pipe. For standard distance plot variables, see "Distance plot items" on page 172.

Description

INTEGRATE finds the integral with respect to time or distance. Used in conjunction with the TIME_HISTORY or HISTORY function, the INTEGRATE function defines a peek name and time period to take the integral of with respect to time. INTEGRATE may also be used to return the distance integral of the specified item.

Example input

Time example

Create a user-defined variable called P_INT that finds the change in pressure with respect to time over the past 5 minutes. Create another user-defined variable called P_IAVE that finds the average change in pressure over the past five minutes.

```
DEFINE P_INT = INTEGRATE(TIME_HISTORY(NODE2:P, 5)) /* time integral of
NODE2:P for past 5 /* minutes
DEFINE P_IAVE = P_INT / 5 /* time average of NODE2:P for past 5 minutes
```

Distance example

Create a user-defined variable named P1_AVG_PRES that finds the average pressure in pipe P1.

```
DEFINE P1_AVG_PRES = INTEGRATE(P1:PRESSURE) / P1:LEN
```

INTERNAL

```
INTERNAL(PIPE:LEN, "QUANT", MILEPOST)
```

Field	Description
PIPE	Pipe for which internal information is requested.
QUANT	Any quantity that can be plotted on a distance plot, enclosed in quotes. For standard distance plot variables, see "Distance plot items" on page 172.
MILEPOST	Milepost where the pipe internal value is to be reported. If the pipe was configured in INPREP with survey data, then enter the corresponding (interpolated) value in the survey array. Otherwise, enter the distance from the from-end of the pipe, in PIPE.LENGTH units.

Description

This function gives a calculated value of a distance plot quantity (such as pressure or flow) for any point in a pipe including each pipe end. This function allows the value of any distance plot quantity to be peeked at any point along a pipe.

Take note

- Although useful, this function is relatively compute-intensive. The presence of these functions may degrade performance.
- If the INTERNAL function is used inside a DEFINE.FUNCTION, any DEFINE.FUNCTION dummy argument may not be used as the first argument to the INTERNAL function.

Example input

Determine the pressure at a point 3.5 miles from the from-end of pipe PIPE1 and the viscosity at each end and in the middle.

```
DEFINE PEAKPR = INTERNAL(PIPE1:LEN, "PRESSURE",3.5)
DEFINE VISC_IN = INTERNAL(PIPE1:LEN, "VISCOSITY", 0.0)
DEFINE VISC_OUT = INTERNAL(PIPE1:LEN, "VISCOSITY", PIPE1:LEN)
DEFINE VISC_MID = INTERNAL(PIPE1:LEN, "VISCOSITY", PIPE1:LEN/2)
```

ISPEEK

```
ISPEEK(VALUE)
```

Field	Description
VALUE	A string that may contain a valid peek name.

Description

If the string VALUE contains a valid peek, this function returns TRUE(=1). Otherwise, this function returns FALSE(=0).

Take note

This function can be used to determine if devices, etc. have been included in the simulation.

Example input

Activate a WHENEVER only when device UNIT27 is included in the model. A DEFINE is set up on whether UNIT27 status is a valid peek. The WHENEVER is defined to be active only when the switch has a value of one and other conditions are met.

```
DEFINE U27SWITCH = ISPEEK("UNIT27:ST")
WHENEVER(U27SWITCH & ...)
-or-
WHENEVER(ISPEEK("UNIT27:ST") & ...)
```

LN

LN(VALUE)

Field	Description
VALUE	Peek name or expression.

Description

This function returns the natural logarithm of VALUE.

Example input

Take the logarithm of a value.

DEFINE LN_VAL = LN(VAL).

As a result, LN_VAL is the logarithm of VAL.

LOOKBACK

```
LOOKBACK(THIST, [HOWFAR])
```

Field	Description
THIST	Expression representing TIME_HISTORY or HISTORY
HOWFAR	Time period to which the simulation should look back

Description

Used in conjunction with the TIME_HISTORY or HISTORY function, the LOOKBACK function returns a previously computed value at some time in the simulated past.

Example input

Create a user-defined variable called P_THIST that keeps the past ten minutes of history data for NODE2:P. Then create another user-defined variable that finds the pressure value at 5 minutes back in time from the current simulation time.

```
DEFINE P_THIST = TIME_HISTORY(NODE2:P, 10)
DEFINE P_OLD = LOOKBACK(P_THIST, 5)
```

MAX

```
/* standard input
MAX(VALUE1 ,..., VALUEn)

/* pipe input
MAX(PIPE:DPLOT_ITEM)

/* path input
MAX(PATH:DPLOT_ITEM)
```

Field	Description
$VALUE_j$	Peek names and/or expressions. When used in the standard example, requires at least two arguments.
PIPE	Pipe for which internal information is requested.
PATH	Path for which internal information is requested. See "DEFINE.PATH" on page 571 for more information.
DPLOT_ITEM	Any valid distance plot item. For standard distance plot variables, see "Distance plot items" on page 172.

Description

When used in the standard example, the MAX function returns the maximum of the VALUEs in the argument list. When used with a pipe or path as an argument, the MAX function determines the maximum value of a valid distance plot item over the length of a pipe or path, respectively.

Example input

Standard example

Track the value of TEST, treating values below zero as zero.

```
DEFINE TEST1 = MAX(TEST,0)
```

As a result, if TEST > 0, then TEST1 = TEST, and if TEST ≤ 0, then TEST1 = 0.

Pipe example

Set PIPE1_MAX equal to the maximum pressure on PIPE1.

```
DEFINE PIPE1_MAX = MAX(PIPE1:PRESSURE)
```

Path example

Set PATH1_MAX equal to the maximum pressure on the path from PIPE1 to PIPE6.

```
DEFINE.PATH PATH1 = PATH(PIPE1, PIPE6)
DEFINE PATH1_MAX = MAX(PATH1:PRESSURE)
```

MAXDIFF

```
/* pipe input
MAXDIFF(PIPE:DPLOT_ITEM1 , PIPE:DPLOT_ITEMn , MILEPOST)

/* path input
MAXDIFF(PATH1:DPLOT_ITEM1, PATHn:DPLOT_ITEMn, MILEPOST[, VERSUS])
```

Field	Description
MILEPOST	Milepost where the maximum difference occurred. If the pipe was configured in INPREP with survey data, then the corresponding (interpolated) value in the survey array is reported. Otherwise, the distance from the from-end of the pipe, in PIPE.LENGTH units, is reported.
PIPE	Pipe for which internal information is requested.
PATH	Path for which internal information is requested. See "DEFINE.PATH" on page 571 for more information.
DPLOT_ITEM	Any valid distance plot item. For standard distance plot variables, see "Distance plot items" on page 172.
VERSUS	Optional argument that specifies whether the MILEPOST is reported using HORIZ.DIST or PIPE.DIST. • VERSUS = 0 means that MILEPOST is reported using HORIZ.DIST (default). HORIZ.DIST is used exactly as specified on the pipe; it is not accumulated for paths. • VERSUS = 1 means that MILEPOST is reported using PIPE.DIST and is accumulated across all pipes in the path.

Description

This function returns the maximum difference (preserving the sign) between the two pipe distance plot variables or path variables and the location where the difference occurred.

Take note

- MAXDIFF is often used to compare pressure with MAOP or with LAOP.
- If the MAXDIFF function is used inside a DEFINE.FUNCTION, any DEFINE.FUNCTION dummy argument may not be used as either the first or second argument to the MAXDIFF function.

Example input

Standard example

Compare the pressure in PIPE1 with its MAOP.

```
DEFINE L.PIPE1 = 0
DEFINE DIFPIPE1/0/ = MAXDIFF(PIPE1:MAX.PRESSURE,PIPE1:MAOP,L.PIPE1)
```

DIFPIPE1 is the maximum difference between the historical maximum pressure vector for PIPE1 and the MAOP for PIPE1. The difference is negative as long as the historical maximum pressure vector is completely below MAOP. The value of L.PIPE1 is the location where the maximum difference occurred. The location is in survey distance if available; otherwise, it is the distance from the from-end of the pipe in PIPE.LENGTH units.

Path example

```
DEFINE.PATH PATH1 = PATH(PIPE1, PIPE6)
DEFINE MXDIF = MAXDIFF(PATH1:PRESSURE, PATH1:MAOP, LOC, 1)
```

MIN

```
/* standard input
MIN(VALUE1 ,..., VALUEn)

/* pipe input
MIN(PIPE:DPLOT_ITEM)

/* path input
MIN(PATH:DPLOT_ITEM)
```

Field	Description
$VALUE_j$	Peek names and/or expressions. When used in the standard example, requires at least two arguments.
PIPE	Pipe for which internal information is requested.
PATH	Path for which internal information is requested. See "DEFINE.PATH" on page 571 for more information.
DPLOT_ITEM	Any valid distance plot item. For standard distance plot variables, see "Distance plot items" on page 172.

Description

When used in the standard example, the MIN function returns the minimum of the VALUEs in the argument list. When used with a pipe or path as an argument, the MIN function determines the minimum value of a valid distance plot item over the length of a pipe or path, respectively.

Example input

Standard example

Assume that you do not want a variable to ever be greater than 1, but the calculated variable TEST is sometimes greater than 1. Define a new variable that is never greater than 1.

```
DEFINE TEST1 = MIN(TEST,1)
```

As a result, if TEST < 1, then TEST1 = TEST, and if TEST ≥ 1, then TEST1 = 1.

Pipe example

Set PIPE1_MIN equal to the minimum pressure on PIPE1.

```
DEFINE PIPE1_MIN = MAX(PIPE1:PRESSURE)
```

Path example

Set PATH1_MIN equal to the minimum pressure on the path from PIPE1 to PIPE6.

```
DEFINE.PATH PATH1 = PATH(PIPE1, PIPE6)
DEFINE PATH1_MIN = MIN(PATH1:PRESSURE)
```

MOD

MOD(VALUE$_1$, VALUE$_2$)

Field	Description
VALUE$_1$	Peek name or expression.
VALUE$_2$	Peek name or expression.

Description

This function gives the remainder of VALUE$_1$/VALUE$_2$. As an example, if A = 5 and B = 2, then MOD(A,B) = 1 because 5/2 has a remainder of 1. If A = 4 and B = 2, MOD(A,B) = 0.

Take note

This is most useful when used with integers, but it may be used with any real numbers.

Example input

Compute the number of minutes into the hour.

```
DEFINE M = MOD(TIME, 60)
```

Therefore, if TIME is 75, then MOD(TIME,60) is 15.

MOVING_AVG

```
MOVING_AVG(VALUE, TP, GRAN)
```

Field	Description
VALUE	Peek name or expression.
TP	Time period over which data is accumulated.
GRAN	Granularity time period at which averages are taken.

Description

The MOVING_AVG function determines the approximate average for each of a series of subsets (granularity time period) of data during a designated time period, and then returns the average of those subsets. For example, a moving average could be defined as the average value of each 1-minute average during the course of a 1440-minute time period. As time progresses, the moving average will change, as the newest data will replace the oldest data in the overall time period. This significantly reduces the amount of information saved in memory if the time steps are very small.

A time-weighted moving average of a simulation value, V, over a specified time period, P, can be defined as follows:

$$\text{MovingAverage}(V,P) = \frac{1}{P}\int_{T=\text{now}-P}^{T=\text{now}} V\,dT$$

To reduce memory requirements, the MOVING_AVG function returns the MovingAverage VALUE over a time period which is approximately TP. At any instant the period is within GRAN of TP. This gives the following:

$$P' = TP +/- GRAN$$

$$\text{MOVING\_AVG}(VALUE,\ TP,\ GRAN) = \text{MovingAverage}(VALUE,\ P')$$

For example, MOVING_AVG(NODE1:P, 1440, 5) will return the average value of NODE1:P over the last 1440 +/- 5 minutes.

Keep in mind the following:

- If the simulation has not yet run for TP minutes, MOVING_AVG will return the average value over the time of the simulation.
- The model does not restrict its timestep to GRAN. If the model takes larger time steps, GRAN is effectively DT during those large time steps.

Example input

Set AVG_TEMP equal to the approximate average over 60 minutes of each 1 minute average. (Since time steps can vary, the average may be greater or less than exactly 60 minutes.)

```
DEFINE THIST_TEMP_DN = TIME_HISTORY(TEMP_DN, 60, 1)
DEFINE AVG_TEMP = MOVING_AVG(THIST_TEMP_DN)
```

Alternatively, you may use the following syntax:

```
DEFINE AVG_TEMP = MOVING_AVG(TIME_HISTORY(TEMP_DN, 60, 1))
```

PEEKLIST

```
PEEKLIST(" dev_names_incl
[DEVICES.EXCLUDE = dev_names_excl,]
[KEYLETTERS.MATCH = dev_kl_include,]
[KEYLETTERS.EXCLUDE = dev_kl_exclude,]
[SUBTYPES.MATCH = subtypes_include,]
[SUBTYPES.EXCLUDE = subtypes_exclude,]
[PEEKS.MATCH = peek_names_incl,]
[PEEKS.EXCLUDE = peek_names_excl,]
[UNITS.MATCH = units_include,]
[UNITS.EXCLUDE = units_exclude,]
[ECHO = echo")
```

Field	Units Key	Description
dev_names_incl	n/a	Devices to include, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_names_excl	n/a	Devices to exclude, by name. Wildcards are permitted. See "Wildcards" on page 50.
dev_kl_include	n/a	Device types to include, by keyletter. See "Device keyletters" on page 47.
dev_kl_exclude	n/a	Device types to exclude, by keyletter. See "Device keyletters" on page 47.
subtypes_include	n/a	Device subtypes (STYP attribute values) to include. For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
subtypes_exclude	n/a	Device subtypes (STYP attribute values) to exclude. For valid STYP values, see "Peek and Poke Keyletters and Attributes" on page 639.
peek_names_incl	n/a	Peek names to be included, in the form of `device name:attribute`. Wildcards are permitted. See "Wildcards" on page 50.
peek_names_excl	n/a	Peek names to be excluded, in the form of `device name:attribute`. Wildcards are permitted. See "Wildcards" on page 50.
units_include	n/a	Units keywords. If this field is used, only peek names that use the specified units are included. See "Default units" on page 129.
units_exclude	n/a	Units keywords. If this field is used, peek names that use the specified units are excluded. See "Default units" on page 129.
echo	n/a	Indicates whether to produce a list of all peek names that are included. Valid values are YES and NO.

Description

The PEEKLIST function returns a list of peek names that satisfy all of the conditions specified in the function. Before using PEEKLIST, note the following:

- The conditions specified in the PEEKLIST function must be enclosed in quotation marks (" "). Generally, you should use wildcards to form these patterns or conditions. For more information, see “Wildcards” on page 50.
- The PEEKLIST function is set up as an expression within the DEFINE command.
- Multiple PEEKLIST defines are allowed in the INTRAN file.
- The PEEKLIST argument is limited to one line in the INTRAN file (225 characters).
- A PEEKLIST function is required to produce input for the SUM and SUMA functions. For more information, see “SUM” on page 627 and “SUMA” on page 628.

Take note

Cascading effect of conditions

Most of the arguments in a DEFINE functions are used to determine precisely which devices should be included in the final list. These arguments act as conditions which function in a cascading manner, each further narrowing the “list” as altered by the previous argument. For example, consider the following DEVICELIST example:

```
DEFINE DL1 = PEEKLIST(“*, DEV.EX = *X*, KEY.MAT = T”, PEEK.MAT = *:P*)
```

In this case, the original DEVICELIST function includes all devices in the model. The DEVICES.EXCLUDE argument removes any devices with an “X” in the name. The KEYLETTERS.MATCH argument further narrows the list to transfer lines only. Finally, the PEEKS.MATCH argument narrows the list to peek names whose attribute begins with “P”.

Example input

Example 1

Define a variable named PKLS1 and populate it with a list that contains the PK (pack rate) attribute peek names for all GP pipes in the model:

```
DEFINE PKLS1 = PEEKLIST("*, KEY.LETTER = GP, PEEK.MATCH = *:PK")
```

Note: A “show PKLS1” reveals:

```
PKLS1:VAL = 0
PKLS1:EXP = PEEKLIST("*, KEY.LETTER = GP, PEEK.MATCH = *:PK")
```

Example 2

Define a variable named PK1 and populate it with a list that contains the Q (standard flow) attribute peek names for all of the TAKE externals in the model except for the external named EIN1:

```
DEFINE PK1 = PEEKLIST(" *, K.L = E," +
   "P.M = *:Q, " +
   "D.E = EIN1," +
   "S.T = TAKE ")
```

Note: Continuation syntax has been used.

PREV

```
PREV(VALUE,[INIT])
```

Field	Description
VALUE	Peek name or expression, either numeric or string valued.
INIT	The value to return the first time step of the simulation. If VALUE is numeric, this argument defaults to 0. If VALUE is a string, this argument is required.

Description

This function returns the value of a peek name or expression calculated during the previous time step.

Take note

- This function may be used only in the expression for a DEFINE (not in the conditional for a WHENEVER, for example).
- This function may be used with string variables, such as statuses. To use with a string value an initial value must be supplied.

Example input

Determine the difference in pressure from one time step to the next at external ABC.

```
DEFINE ABCP = ABC:P+ - PREV(ABC:P+)
```

Determine if UNIT1 has changed status from one time step to the next. The expression has a value of 1 if the statuses are the same or 0 if they differ.

```
DEFINE UNIT1STC = (UNIT1:ST = PREV(UNIT1:ST, " "))
```

SCRAPER

```
SCRAPER(PATH, LAUNCH,SLIP,LOC,[ETA,[RVOL,[DEV,[DLOC,[DETA,[VEL]]]]]])
```

Keyword	Units	Description
PATH	n/a	DEFINE.PATH of devices in the scraper path.
LAUNCH	TIME	Scraper launch time. If time is less than current simulation time, launch immediately.
SLIP	n/a	Velocity slip factor. Scraper velocity = (1-SLIP) * pipe velocity.
LOC	PIPE.LENGTH	Distance from the beginning of PATH (updated)
ETA	TIME	Estimated time of arrival at end of PATH (updated).
RVOL	INVENTORY	Remaining volume to the end of PATH (updated).
DEV	n/a	Name for the pipe/header where the scraper is located (updated).
DLOC	PIPE.LENGTH	Distance into the DEV (updated).
DETA	TIME	Estimated time of arrival at end of DEV (updated).
VEL	VELOCITY	Velocity of the scraper (updated).

Description

This function is used to track a scraper (pig) through a piping system. The SCRAPER is a virtual device that keeps track of a scraper without putting a real device into the model. SCRAPER may be used in INTRAN only.

Before putting in a scraper, you need to define a path (DEFINE.PATH) through the model. Flow in each pipe and header of the path should be in the positive direction. This restriction may require some reconfiguring of the system. Define the path carefully because SPS does not restrict the scraper from moving through inappropriate elements, such as pumps or compressors.

Take note

Launching

A scraper is inactive if LAUNCH = 0. To launch a scraper, set launch to the appropriate time. If LAUNCH is less than the simulation time, the scraper is launched immediately.

Reporting

When the scraper launches, the function automatically calculates LOC, ETA, RVOL, DEV, DLOC, DETA, and VEL.

Example input

A scraper is being launched at the beginning of the system and will be caught at the first station. The scraper is launched 10 minutes into the simulation.

```
DEFINE.PATH PATH1 = PATH(T1, T2, H1, T3, T4, H2)
DEFINE P1LT = 10
DEFINE P1SLIP = 0.05
```

```
DEFINE P1LOC = 0
DEFINE P1ETA = 0
DEFINE P1RVOL = 0
DEFINE P1DEV = " "
DEFINE P1DLOC = 0
DEFINE P1DETA = 0
DEFINE P1VEL = 0
DEFINE P1SCRAPER =
+ SCRAPER(PATH1,P1LT,P1SLIP,P1LOC,P1ETA,P1RVOL,P1DEV,P1DLOC,P1DETA,P1VEL)
```

STDV

```
STDV(THIST(VALUE, TP))
```

Field	Description
THIST	Expression representing TIME_HISTORY or HISTORY
VALUE	Peek name or expression.
TP	Time period over which data is accumulated.

Description

Used in conjunction with the TIME_HISTORY and HISTORY function, the STDV function calculates the standard deviation of VALUE over the time period defined by TP. The standard deviation is defined as follows:

$$\text{Standard Deviation} = \sqrt{\Sigma \left(X_1 - X_{AVE} \right)^2 / \left(n - 1 \right)}$$

where:

X = Peek name
n = number of points in time period

Example input

Define TEMP_STABLE as a function of the standard deviation of TEMP_DN over the last 5 minutes. TEMP_STABLE will be TRUE if the standard deviation of TEMP_DN is less than one, and FALSE otherwise.

```
DEFINE THIST = TIME_HISTORY(TEMP_DN, 5)
DEFINE TEMP_STABLE = STDV(THIST) < 1
```

—or—

```
DEFINE TEMP_STABLE = STDV(TIME_HISTORY(TEMP_DN, 5)) < 1
```

SUM

```
SUM(X)
```

Field	Units Key	Description
X	n/a	The values to be summed from a PEEKLIST

Description

The SUM function returns the sum of all of the numerical values included in the input. The input is an array of peek names satisfying the given conditions (see "PEEKLIST" on page 621). The input may be a defined variable that has been previously defined containing an array of peek names, or the input may contain the PEEKLIST function itself.

The variable gets evaluated (the sum is re-calculated) every time step. SUM skips the peek names for which values are strings, such as :ST.

Example input

Example 1

Referring to Example 1 under the PEEKLIST function, define a variable named SMPKLS1, which returns the sum of all of the :PK peeks for all GP pipes in the model:

```
DEFINE PKLS1 = PEEKLIST("*, KEY.LETTER = GP, PEEK.MATCH = *:PK")
DEFINE SMPKLS1 = SUM(PKLS1)
-or-
DEFINE SMAPKLS1 = SUM(PEEKLIST("*, K.L = GP, P.MA = *:PK"))
```

When the model is at steady state, SMPKLS1 is zero.

Example 2

Referring to Example 2 under the PEEKLIST function, define a variable named SMEXPK1, which returns the sum of all of the :Q peeks (standard flow values) for all TAKE externals in the model except for the external named EIN1:

```
DEFINE PK1 = PEEKLIST(" *, K.L = E," +
   "P.M = *:Q, " +
   "D.E = EIN1," +
   "S.T = TAKE ")
DEFINE SMEXPK1 = SUM(PK1)
```

SUMA

SUMA(X)

Field	Units Key	Description
X	n/a	The values to be summed from a PEEKLIST.

Description

The SUMA function returns the sum of all of the absolute values of the numerical values included in the input. The input is an array of peek names satisfying the given conditions (see "PEEKLIST" on page 621). The input may be a defined variable that has been previously defined containing an array of peek names or the input may contain the PEEKLIST function itself.

The variable gets evaluated (the sum is re-calculated) every time step. SUMA skips the peek names for which values are strings, such as :ST.

Example input

Example 1

Referring to Example 1 under "PEEKLIST" on page 621, define a variable named SMAPKLS1, which returns the sum of all of the absolute values of the :PK peeks for all transfer lines in the model:

```
DEFINE PKLS1 = PEEKLIST("*, KEY.LETTER = GP, PEEK.MATCH = *:PK")
DEFINE SMAPKLS1 = SUM(PKLS1)
-or-
DEFINE SMAPKLS1 = SUMA(PEEKLIST("*, K.L = T, P.MA = *:PK")
```

When the model is at steady state, SMPKLS1 is zero.

Example 2

Referring to Example 2 under "PEEKLIST" on page 621, define a variable named SMAEXPK1, which returns the sum of all of the absolute values of the :Q peeks (standard flow values) for all TAKE externals in the model except for the external named EIN1:

```
DEFINE PK1 = PEEKLIST(" *, K.L = E," +
   "P.M = *:Q, " +
   "D.E = EIN1," +
   "S.T = TAKE ")
DEFINE SMAEXPK1 = SUMA(PK1)
```

TAVE

```
TAVE(VALUE , DECAY [, INITIAL])
```

Field	Description
VALUE	Peek name or expression to be averaged.
DECAY	The decay period over which the value is averaged (minutes).
INITIAL	Initial value of the variable to be averaged. Must be numeric.

Description

This function gives a running average of some value for some decay period:

$$wt = e^{-dt / decay}$$

$$tave_k = val_k + wt \bullet (tave_{(K-1)} - val_K)$$

where K is the step number.

Take note

TAVE may be used only in a DEFINE (not in the conditional for a WHENEVER, for example).

Example input

Average the flow at external ABC with a decay rate of 15 minutes and initial flow of 25.

```
DEFINE ABCFA = TAVE(ABC:Q,15,25)
```

TAYLOR1

TAYLOR1(X, a_0, a_1, a_2, ..., a_n)

Field	Description
X	Peek name representing independent variable in Taylor Series
a_0, a_1, a_2, …, a_n	Constants representing differentiated terms in Taylor Series

Description

The TAYLOR1 data filter is used to calculate the nth-degree Taylor polynomial of one independent variable. The expanded Taylor Series is:

$$\text{TAYLOR1}\ (X, a_0, a_1, a_2, \ldots, a_n) = a_0 + a_1 \cdot X + a_2 \cdot X^2 + \ldots + a_n \cdot X^n$$

Take note

Any number of terms can be used in the TAYLOR1 function.

Example input

Calculate DENS using the TAYLOR1 Series approximation for the following function:

```
/* DENS = RHOT * (1 + PCOEF1*PBAR + PCOEF2*PBAR**2)
CALC DENS,
 + VAL = RHOT * TAYLOR1( PBAR, 1, PCOEF1, PCOEF2)
```

TAYLOR2

TAYLOR2(X, Y, a_0, a_1, b_1, a_2, c_2, b_2)

Field	Description
X	Peek name representing independent variable in Taylor Series
Y	Peek name representing independent variable in Taylor Series
a_0, a_1, b_1...b_2	Constants representing differentiated terms in Taylor Series

Description

The TAYLOR2 data filter is used to calculate the second order Taylor polynomial of two independent variables. The expanded Taylor Series is:

$$\text{TAYLOR2}\,(X, Y, a_0, a_1, b_1, a_2, c_2, b_2) = a_0 + a_1 \cdot X + b_1 \cdot Y + a_2 \cdot X^2 + c_2 \cdot X \cdot Y + b_2 \cdot Y^2$$

Take note

The TAYLOR2 Series approximation is limited to second order.

Example input

Calculate the specific volume using the TAYLOR2 series approximation in the following equation:

```
/* SPV = (1/DBASE - PDIFF*PCOEF - TDIFF*TCOEF
/*          - PDIFF*TDIFF*PTCOEF +PDIFF2*PPCOEF)
CALC SPV,
+ VAL = TAYLOR2(PDIFF,TDIFF,1/DBASE,-PCOEF,-TCOEF, PPCOEF,-PTCOEF),
+ RTUID = SPV,
+ UPDATE.IF CYCLE_CHECK
```

Time function operators

```
SECOND (X)
MINUTE (X)
HOUR (X)
DAY (X)
MONTH (X)
YEAR (X)
DATE (X,Y,Z)
TIMEVALUE (X)
```

Description

The following are available time function operators:

SECOND(X)	Returns truncated seconds part of time, 0-59
MINUTE(X)	Returns truncated minutes part of time, 0-59
HOUR(X)	Returns truncated hours part of time, 0-23
DAY(X)	Returns truncated days part of time, 1-31
MONTH(X)	Returns truncated months part of time, 1-12
YEAR(X)	Returns truncated years part of time, 1970-2050
DATE (X,Y,Z)	Converts (year, month, day) to internal TIME value
TIMEVALUE (X)	Converts “yy/mm/dd hh:mm:ss” string to internal TIME value

These functions can be used to convert values to SPS internal time and vice versa. This may be used to reset the program clock to a new time or to convert program time to strings that can be used in file names, etc.

Example input

Generate an archive file name case1_XXd_YYh.ark where “XX” is the day of the month (1-31) and “YY” is the hour of the day (0-23).

```
ARCHIVE "case1_[DAY(TIME),2.0]d_[HOUR(TIME),2.0]h.ark"
```

TIME_HISTORY

```
TIME_HISTORY(VALUE, TP)
```

Field	Description
VALUE	Peek name
TP	Time period over which data is accumulated.

Description

This function must be used with other functions (for example, AVG, COUNT, INTEGRATE, LOOKBACK, and STDV) to specify the name and time period that may be used in a calculation; it cannot be used as a standalone function. You may specify a larger history range than you intend to use in a calculation.

Take note

The storage requirements for TIME_HISTORY will be large when the time step is small for an extended period.

Example input

Define TEMP_STABLE as a function of the standard deviation of TEMP_DN over the last five minutes.

```
DEFINE TEMP_STABLE = STDV(THIST, 5)
```

TINT

```
TINT(VALUE [, INIT])
```

Field	Description
VALUE	A peekable variable or expression that is to be integrated.
INIT	Initial value of the variable to be integrated.

Description

This function gives the time integral of the variable. The integral is for the time interval from the start of the simulation to the current time step.

$$TINT_K = TINT_{(K-1)} + dt\ (VAL_K + VAL_{(K-1)}\ ) / 2$$

where K is the step number.

Take note

This function can be used only within a DEFINE (not in the conditional for a WHENEVER, for example).

Example input

Integrate flow at external ABC over the simulation time to obtain a total flow value.

```
DEFINE ABCTOT = TINT(ABC:Q)
```

TOSTR

```
TOSTR(EXPRESSION [,N1][,N2][,N3])
```

Field	Description
EXPRESSION	Expression to be converted to a string.
N1	Integer that specifies the number of decimal places to be displayed. {-1}
N2	Integer that specifies the maximum length of the final string. {80}
N3	Integer that specifies the minimum length of the final string. {1}

Description

This function converts the expression to a string.

Take note

- Any value included in quotes (" ") is passed as a string.
- A value of -1 for N1 corresponds to no decimal point.
- If the string length is less than N3, the final string will be padded with enough leading zeroes to meet the minimum length requirement.

Example input

Define a variable that is the simulation time in hours and minutes only. The TOSTR function may be used in conjunction with the Time functions.

```
DEFINE TIMESTR = TOSTR(HOUR(TIME),-1,2) + ":"
+ TOSTR(MINUTE(TIME),-1,2)
```

If five hours and 20 minutes of simulation time has passed since the start time of 0, the result is:

```
TIMESTR = "5:20"
```

With N3 set equal to 2, the result from the previous example would be:

```
TIMESTR = "05:20"
```

TRUNC

```
TRUNC(STRING , N1)
```

Field	Description
STRING	String to be truncated.
N1	Integer that specifies the length of the final string

Description

This function truncates a character string.

Take note

Any value included in quotes (" ") is passed as a string.

Example input

Truncate a character string to 4 characters. The TRUNC function may be used to truncate a character string such as a pump status.

```
TRUNC(PUMP1:ST,3)
```

This would return “RUN” if PUMP1:ST = “RUNNING”

TRUNC could also be used in conjunction with a DEFINE or other logic.

```
DEFINE LEAK_ST = TRUNC(LF.STATUS,4)
```

This would set LEAK_ST = “STAR” if LF.STATUS = “STARTING”.

UNITS

```
UNITS(PEEKNAME)
```

Field	Description
PEEKNAME	A peek name.

Description

This function returns the units, as a string, corresponding to a peek name.

Example input

Get the unit name for VALVE2:P-

```
DEFINE UNAME = UNITS(VALVE2:P-)
```

As a result, UNAME will be the text “PSIG”.

CHAPTER

15

Peek and Poke Keyletters and Attributes

This section contains a listing of device types with their associated keyletters and attributes, which may be used to view, get, set, or modify data. Some attributes are common to all devices, for more information, see "Common peek and poke attributes" on page 641. If you are working with an online model, see "Peek and poke keyletters and attributes for online modeling" on page 835.

Device or property set	Keyletters	Subtype
"Actuator (A) attributes" on page 642	A	{n/a}
"Alarm category (AC) attributes" on page 643	AC	{n/a}
"Alarm name (AL) attributes" on page 643	AL	{n/a}
"AUDITS (AU) attributes" on page 643	AU	{n/a}
"Batch tracking (BT) attributes" on page 644	BT	{n/a}
"Block valve (B) attributes" on page 646	B	{n/a}
"Block valve (BV) attributes" on page 647	BV	{n/a}
"Centrifugal compressor (CC) attributes" on page 647	CC	{n/a}
"Centrifugal compressor (KC) attributes" on page 649	KC	{n/a}
"Check valve (BC) attributes" on page 650	BC	{n/a}
"Check valve (CV) attributes" on page 651	CV	{n/a}
"Control valve (V) attributes" on page 652	V	{n/a}
"Controller (C) attributes" on page 653	C	{n/a}
"Data curve (D, D2, D3) attributes" on page 653	D	{n/a}
"DEFINE (DE) attributes" on page 654	DE	{n/a}
"DEFINE.FUNCTION (DF) attributes" on page 654	DF	{n/a}
"DEFINE.PATH (DP) attributes" on page 654	DP	{n/a}
"DEFINE.SEQUENCE (DS) attributes" on page 655	DS	{n/a}
"DEFINE.TIMETABLE (TT) attributes" on page 655	TT	{n/a}
"Derivative relay (Y) attributes" on page 656	Y	DERIV

Device or property set	Keyletters	Subtype
"External (P-CONTROL/Q-CONTROL/Q(P)) (E) attributes" on page 656	E	P-CONTROL —or— Q-CONTROL —or— P(Q)
"External (SALE/TAKE) (E) attributes" on page 657	E	SALE —or— TAKE
"Feedback relay (Y) attributes" on page 661	Y	FEEDBACK
"Flow meter (FM) attributes" on page 662	FM	{n/a}
"Fluid name (FL) attributes" on page 662	FL	{n/a}
"General compressor (GC) attributes" on page 663	GC	{n/a}
"General pipe (GP) attributes" on page 664	GP	{n/a}
"GLOBALS (GB) attributes" on page 665	GB	{n/a}
"Grove G887 relief valve (V) attributes" on page 668	V	{n/a}
"Guide vane compressor (KV) attributes" on page 669	KV	{n/a}
"Header (H) attributes" on page 671	H	{n/a}
"Heat exchanger (HE) attributes" on page 672	HE	{n/a}
"HI/LO select relay (Y) attributes" on page 672	Y	HI —or— LO
"Idealized regulator (RE) attributes" on page 673	RE	{n/a}
"Idealized regulator (RG) attributes" on page 674	RG	{n/a}
"Input reference (I) attributes" on page 675	I	{n/a}
"Integrator relay (Y) attributes" on page 675	Y	INTEG
"MAXMIN" on page 479	MM	{n/a}
"Multiply relay (Y) attributes" on page 675	Y	MULTIPLY
"Node (NO) attributes" on page 676	NO	TAKE
"Noise relay (Y) attributes" on page 680	Y	NOISE
"Plot limits (PL) attributes" on page 680	PL	
"Pump (P) attributes" on page 680	P	{n/a}
"Reciprocating compressor (RC) attributes" on page 682	RC	{n/a}
"Relief valve (RV) attributes" on page 683	RV	{n/a}

Device or property set	Keyletters	Subtype
"Sensor (S) attributes" on page 685	S	See "Sensor (S)" on page 389
"Shared memory (SH) attributes" on page 685	SH	{n/a}
"Station (ST) attributes" on page 685	ST	{n/a}
"Span (SP) attributes" on page 686	SP	{n/a}
"Surge tank (E) attributes" on page 687	E	SURGETANK
"Switch relay (Y) attributes" on page 689	Y	SWITCH
"Tank (TK) attributes" on page 689	T	{n/a}
"Theoretical horsepower flow compressor (KP) attributes" on page 690	KP	{n/a}
"Time-averaging relay (Y) attributes" on page 692	Y	AVERAGE
"Transfer line (T) attributes" on page 692	T	{n/a}
"TRANSTHERMAL (TR) attributes" on page 694	TR	{n/a}
"Wax deposition (WX) attributes" on page 695	WAX	{n/a}

Common peek and poke attributes

Hydraulic devices

Attribute	Pokable	Units	Description
MAOP	Yes	PRESSURE	Maximum allowable operating pressure. When MAOP is poked, MASP goes to 110% of MAOP.
MASP	Yes	PRESSURE	Maximum allowable surge pressure. When MAOP is poked, goes to 110% of MAOP. Uses input value if MASP is poked.
LAOP	Yes	PRESSURE	Lowest allowable operating pressure
LASP	Yes	PRESSURE	Lowest allowable surge pressure
NN+	No	none	Node at from-end of device
NN-	No	none	Node at to-end of device

All devices

Attribute	Pokable	Units	Description
DESC	Yes	none	Description string for comments. If entered, the string must be enclosed in single quotation marks (' ').
KYLT	Yes	none	Keyletters
SL.USE	Yes	none	SL.ALL displays all peeks. SL.SHORT displays short peek list.

Actuator (A) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN	No	none	Actuator input signal
OUT	No	none	Actuator output signal
TYPE	Yes	none	Device type. Available choices are: • SECOND ORDER • RATE CONSTANT • TRAVEL TIME
CURVE	Yes	none	Name of a data curve defining actuator position X as a function of input signal V.
V0	Yes	none	Input limit where $X(V_0) = 0$.
V1	Yes	none	Input limits where $X(V_1) = 1$.
R	Yes	none	Rate constant for the actuator, approximately the reciprocal time for full scale travel following a full scale step change in input, in reciprocal minutes.
A1	Yes	none	Coefficient for the second order function if they are to be entered directly.
A2	Yes	none	Coefficient for the second order function if they are to be entered directly.
A3	Yes	none	Coefficients for the second order function if they are to be entered directly.
TOPEN	Yes	none	When opening, the actuator position follows the scaled input signal (no second-order smoothing), but does not open at a rate faster than indicated by T_{open}.
TCLOSE	Yes	none	When opening, the actuator position follows the scaled input signal (no second-order smoothing), but does not open at a rate faster than indicated by T_{close}.

Alarm name (AL) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
MSG	No	none	Current alarm message (blank if alarm is not triggered)
ST	No	none	Current status of the alarm. The alarm status descriptions are {Corresponding integer value}: ON—The alarm is currently on. {1} OFF—The alarm is currently off. {2}
NTRIG	Yes	none	Total number of times this alarm has been triggered
CAT	Yes	none	Alarm category this alarm is assigned to
MODE	Yes	none	To enable or disable the generation of the alarm messages {ENABLE/DISABLE}
ALNUM	No	none	Alarm number

Alarm category (AC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
NUMON	No	none	The total number of alarms in this category with a status of ON
COUNT	No	none	The sum of the total number of times that each alarm in the category has been triggered
MODE	Yes	none	To enable or disable the generation of all of the alarm messages within this category {ENABLE/DISABLE}
NUM	No	none	Number of alarms in this category

AUDITS (AU) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
DCLOCK	No	TIME	Elapsed wall clock time on a per-step basis
DCPU	No	TIME	Elapsed CPU time on a per-step basis
NISOLVE	No	none	Number of calls to parse matrix solution routine on a per-step basis (NISOLVE > 1 means post-mortems).
NREJECT	No	none	Number of steps that were rejected on a per-step basis

Attribute	Pokable	Units	Description
NBATCH	No	none	Number of batch interfaces
NBSTEP	No	none	Number of batch tracking steps per time step
NBITER	No	none	Number of batch tracking composition iterations per time step
NFMV	No	none	Number of fluid mixture vectors in the model
NBATCH	No	none	Number of fluid mixture vectors in pipes
$MAIN	No	SECONDS	CPU time in driver program plus all otherwise un-audited CPU time (per step)
$SINGLE	No	SECONDS	CPU time in pipe solution (per step)
$UPSN	No	SECONDS	CPU time in pipe update (per step)
$MBATCH	No	SECONDS	CPU time in batch movement (per step)
$LNGN	No	SECONDS	CPU time to form linearized equations (per step)
$ISOLVE	No	SECONDS	CPU time to solve sparse matrix (per step)
$REFRSH	No	SECONDS	CPU time for INTERACTIVE processing (per step)
$REVIEW	No	SECONDS	CPU time for REVIEW file update (per step)
$PARSE	No	SECONDS	CPU time for parse of INTRAN (per step)
$UPDATE	No	SECONDS	CPU time to update data structures after solving
$PMORT	No	SECONDS	CPU time in post-mortem
$SCADA	No	SECONDS	CPU time spent reading RTU data file (online only)
$DEFLANG	No	SECONDS	CPU time spent processing DEFINEs and WHENEVERs
$LFANAL	No	SECONDS	CPU time spent doing leak analysis (online only)

Batch tracking (BT) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ID1	Yes	none	Auxiliary Identifier of form NNNNL where N is an integer and L is a letter
ID2	Yes	none	Auxiliary Identifier of form NNNN where N is an integer.
ID3	Yes	none	Auxiliary Identifier of the form NNN where N is an integer
FLU	Yes	none	Name of the fluid in the batch
AVOL	No	VOLUME	Actual volume of the batch at pipeline conditions
VOL	Yes	VOLUME	Volume of the batch at custody conditions
ADIR	Yes	none	Adjustment direction for the batch - options are: HEAD, TAIL, or BOTH. Default is TAIL
HPIP	No	none	Name of the pipe where the batch head is located

Attribute	Pokable	Units	Description
HLOC	Yes	LENGTH.PIPE	Location of the batch head in the pipe
HVL	Yes	VOLUME	Volume location of the batch head in the pipe
TPIP	No	none	Name of the pipe where the batch tail is located
TLOC	Yes	LENGTH.PIPE	Location of the batch tail in the pipe
TVL	Yes	VOLUME	Volume location of the batch tail in the pipe
LEN	Yes	LENGTH.PIPE	Length of the batch
INJ	No	none	External where the batch was injected
INJT	No	TIME	Time the batch was injected
NSEG	No	none	Number of disconnected segments with this batch name
NFMV	No	none	Number of FMVs in the batch
P	No	PRESSURE	Average pressure of the batch
T	No	TEMPERATURE	Average temperature of the batch
DEN	No	DENSITY	Average density of the batch at pipeline pressure and temperature
D0	Yes	DENSITY	Average reference density
PM	Yes	BULK.MOD	Average bulk modulus at pipeline pressure and temperature
TM	Yes	ΔTEMP	Average temperature modulus at pipeline pressure and temperature
PTM	No	none	Average multiplier
PPM	No	none	Multiplier for the $(\Delta P)^2$ term
TTM	No	none	Average multiplier
VISC	No	VISCOSITY	Average viscosity of the batch at pipeline pressure and temperature
V0	Yes	VISCOSITY	Average reference viscosity
VPMI	No	1/ΔP	Average viscosity pressure multiplier
VTMI	No	1/ΔT	Average viscosity temperature multiplier
ASTMA	No	none	Constant for ASTM formula of viscosity
ASTMB	No	none	Constant for ASTM formula of viscosity
CV	No	HEAT.CAP	Average heat capacity at constant volume
CVT	No	HEAT.CAP/ΔT	Average heat capacity temperature coefficient
HV	No	HEAT.VAL	Average heating value of the batch
HCON	No	HEAT.COND	Average heat conductivity of the batch
HCONC	No	HEAT.COND/ΔT	Average heat conductivity temperature coefficient
VP	No	PRESSURE	Average vapor pressure

Attribute	Pokable	Units	Description
BME	No	none	Average bulk modulus error
DE	No	DENSITY	Average density error
VE	No	none	Average viscosity error
SDD	No	none	Standard deviation of density
SDPM	No	none	Standard deviation of bulk modulus
SDTM	No	none	Standard deviation of temperature modulus
SDV	No	none	Standard deviation of viscosity

Block valve (B) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of valve
P+	No	PRESSURE	Pressure at *to-end* of valve
T-	No	TEMP	TEMP at from-end of valve
T+	No	TEMP	TEMP at *to-end* of valve
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of *to-end*, Standard conditions
FR	Yes	FRACTION	Fraction Open (Valve coefficient percentage)
CF	Yes	VOLUME	Cumulative flow through valve
ST	Yes	none	Valve status. The valve status descriptions are: {Corresponding integer value} • STOPPED – Valve has been stopped in mid-travel. {1} • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PD	No	ΔPRESSURE	Pressure drop across valve: (P-) - (P+)
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of *to-end*, pipeline conditions
SCV	No	none	Status of series check valve (appears only when the series check valve is entered)
CV	No	VALVE.COEF	Actual valve coefficient.
CVC	Yes	VALVE.COEF	Valve coefficient at full-close position

Attribute	Pokable	Units	Description
CVO	Yes	VALVE.COEF	Valve coefficient at full-open position
T	Yes	TIME	Valve travel time from fully closed to fully open (or vice versa)

Block valve (BV) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
ST	Yes	none	Status
QTYPE	No	none	Flow type
PU	No	PRESSURE	Upstream pressure
PD	No	PRESSURE	Downstream pressure
RATIO	No	none	Pressure ratio
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow
FU	No	AFLOW	Actual upstream flow
FD	No	AFLOW	Actual downstream flow
FR	Yes	FR	Fraction Open (Valve Coefficient percentage)
CG	No	VALVE.CG	Actual valve coefficient.
CGMAX	Yes	VALVE.CG	Max valve coefficient (open)
CGMIN	Yes	VALVE.CG	Min valve coefficient (close)
TT	Yes	TIME	Travel time
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature

Centrifugal compressor (CC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
PU	No	PRESSURE	Suction pressure
PD	No	PRESSURE	Discharge pressure

Attribute	Pokable	Units	Description
PSMIN	Yes	PRESSURE	Suction pressure set point
PDMAX	Yes	PRESSURE	Discharge pressure set point
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow, Standard conditions
QI	No	FLOW	Through flow, Standard conditions
ST	Yes	none	Unit Status. The unit status descriptions are: {Corresponding integer value} • STOPPED – Unit is stopped. {1} • STARTING – Unit is starting. {2} • RUNNING – Unit is running. {3} • STOPPING – Unit is stopping. {4}
QR	No	FLOW	Recycle flow, standard conditions
QF	No	FLOW	Fuel flow, standard conditions
HF	No	TFLOW	Thermal fuel flow
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
SQ	Yes	FLOW	Flow set point, standard conditions
FU	No	AFLOW	Suction flow, pipeline conditions
FD	No	AFLOW	Discharge flow, pipeline conditions
TU	No	TEMP	Suction TEMP
TD	No	TEMP	Discharge TEMP
S	No	SPEED	Compressor speed
SS	Yes	SPEED	Set speed for compressor
SMIN	Yes	SPEED	Minimum permissible speed
SMAX	Yes	SPEED	Maximum permissible speed
POWER	No	POWER	The brake power delivered
PWMAX	No	POWER	Maximum available power
HD	No	HEAD	Head
RATIO	No	none	Compression ratio
EFF	No	none	Compressor efficiency
NPOLY	Yes	none	Polytropic exponent. If you are using the AGA or BWRS equation of state, entering a poke value other than zero will hold the value constant. If, however, you enter a zero, then the calculated value will be used.
TRR	Yes	none	Temperature rise ratio
MODE	No	none	What is controlling compressor operation

Centrifugal compressor (KC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Suction pressure
P+	No	PRESSURE	Discharge pressure
SP-	Yes	PRESSURE	Suction pressure set point
SP+	Yes	PRESSURE	Discharge pressure set point
PD	No	ΔPRESSURE	Differential pressure
Q-	No	FLOW	Suction flow, Standard conditions
Q+	No	FLOW	Discharge flow, Standard conditions
QI	No	FLOW	Through flow, Standard conditions
ST	Yes	none	Unit Status. The unit status descriptions are: {Corresponding integer value} • STOPPED – Unit is stopped. {1} • STARTING – Unit is starting. {2} • RUNNING – Unit is running. {3} • STOPPING – Unit is stopping. {4}
QR	No	FLOW	Recycle flow, standard conditions
QF	No	FLOW	Fuel flow, standard conditions
SQ+	Yes	FLOW	Flow set point, standard conditions
F-	No	AFLOW	Suction flow, pipeline conditions
F+	No	AFLOW	Discharge flow, pipeline conditions
FI	No	COMP.AFLOW	Inlet flow, Suction conditions
FIS	No	COMP.AFLOW/ SPEED	Suction flow index (:FI/:S)
T-	No	TEMP	Suction TEMP
T+	No	TEMP	Discharge TEMP
MXT+	Yes	TEMP	Maximum discharge TEMP
TE	No	TEMP	Power turbine exhaust TEMP
MXTE	Yes	TEMP	Maximum power turbine TEMP
TA	Yes	TEMP	Ambient TEMP
TR	No	TEMP	Reference TEMP from manufacturer data
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
S	No	SPEED	Compressor speed
SS	Yes	SPEED	Set speed for compressor

Attribute	Pokable	Units	Description
MNS	Yes	SPEED	Minimum permissible speed
MXS	Yes	SPEED	Maximum permissible speed
PWR	No	POWER	The brake power delivered
MXP	No	POWER	Maximum available power
NRN	Yes	none	Number of units running
HD	No	HEAD	Head
R	No	none	Compression ratio
RN	No	none	Compression ratio based on node pressures
HE	No	EFFICIENCY	Hydraulic efficiency
NPOLY	Yes	none	Polytropic exponent. If you are using the AGA or BWRS equation of state, entering a poke value other than zero will hold the value constant. If, however, you enter a zero, then the calculated value will be used.
TRR	Yes	none	Temperature rise ratio
HEC	Yes	none	Multiplier for hydraulic efficiency curve data
ME	Yes	EFFICIENCY	Mechanical efficiency
LHV	Yes	HEAT.VAL	Lower heating value for fuel gas
MNE	No	EFFICIENCY	Minimum efficiency
MXE	No	EFFICIENCY	Maximum efficiency
MODE	No	none	What is controlling compressor operation
VCNS	No	none	Violated constraints
SCV	No	none	Series check valve status
BCV	No	none	Bypass check valve status
CB	No	none	Name of controlling actuator, if applicable
PI+	No	PRESSURE	Intermediate discharge pressure of unit
NCMC	Yes	none	Number of control mode changes
HF	No	TFLOW	Thermal fuel flow
FSRG	No	AFLOW	Actual flow corresponding to surge line.

Check valve (BC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of valve
P+	No	PRESSURE	Pressure at *to-end* of valve

Attribute	Pokable	Units	Description
T-	No	TEMP	TEMP at from-end of valve
T+	No	TEMP	TEMP at *to-end* of valve
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of *to-end*, Standard conditions
FR	No	FRACTION	Fraction Open (Valve Coefficient percentage)
CF	Yes	VOLUME	Cumulative flow through valve
ST	No	none	Valve status. The valve status descriptions are {Corresponding integer value}: • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PD	No	ΔPRESSURE	Pressure drop across valve: (P-) - (P+)
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
CV	No	VALVE.COEF	Actual valve coefficient.
CVC	Yes	VALVE.COEF	Valve coefficient at full-close position
CVO	Yes	VALVE.COEF	Valve coefficient at full-open position
T	Yes	TIME	Valve travel time from fully closed to fully open (or vice versa)

Check valve (CV) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
ST	No	none	Status
QTYPE	No	none	Flow type
PU	No	PRESSURE	Upstream pressure
PD	No	PRESSURE	Downstream pressure
RATIO	No	none	Pressure ratio
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow
FU	No	AFLOW	Actual upstream flow

Attribute	Pokable	Units	Description
FD	No	AFLOW	Actual downstream flow
FR	No	FR	Fraction Open (Valve Coefficient percentage)
CG	No	VALVE.CG	Actual valve coefficient.
CGMAX	Yes	VALVE.CG	Max valve coefficient (open)
CGMIN	Yes	VALVE.CG	Min valve coefficient (close)
TT	Yes	TIME	Travel time
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature

Control valve (V) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of valve
P+	No	PRESSURE	Pressure at to-end of valve
T-	No	TEMP	TEMP at from-end of valve
T+	No	TEMP	TEMP at to-end of valve
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
FR	No	FRACTION	Valve fraction open
CF	Yes	VOLUME	Cumulative flow through valve
ST	No	none	Valve status. The valve status descriptions are: {Corresponding integer value} • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PD	No	ΔPRESSURE	Pressure drop across valve: (P-) - (P+)
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
CV	No	VALVE.COEF	Actual valve coefficient.
CVC	Yes	VALVE.COEF	Valve coefficient at full-close position
CVO	Yes	VALVE.COEF	Valve coefficient at full-open position

Attribute	Pokable	Units	Description
SCV	No	none	Status of series check valve (appears only when the series check valve is entered)
CB	No	none	Name of controlling actuator, if applicable

Controller (C) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
SP	Yes	none	Set point for controller
ER	No	none	Error for controller
KC	Yes	none	Gain for controller
TD	Yes	SIG.TIME	Derivative multiplier for controller
TI	Yes	SIG.TIME	Reset time for controller
VS	Yes	none	Bias for controller
NC	Yes	none	Normalization constant for controller
INT	Yes	none	Integral of error for controller
IN	No	varies	Input signal to controller
OUT	No	none	Output signal from controller
HI	Yes	none	Upper limit of controller output signal (Also upper limit for the integral term if the anti-windup feature is triggered)
LO	Yes	none	Lower limit of controller output signal (Also lower limit for the integral term if the anti-windup feature is triggered)
AW	Yes	TIME	Reset time (for anti-windup controllers only)

Data curve (D, D2, D3) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
NPTS	No	none	Number of points
X(i)	No	special	i^{th} X value. Units depend on curve type.
Y(i)	No	special	i^{th} Y value. Units depend on curve type.
A(i)	No	none	i^{th} Curve fit coefficient
B(i)	No	none	i^{th} Curve fit coefficient
C(i)	No	none	i^{th} Curve fit coefficient

DEFINE (DE) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
VAL	Yes	none	Value of defined expression
EXP	Yes	none	Expression of the defined variable
UIF	No	none	Test conditional for the variable

DEFINE.FUNCTION (DF) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
COUNT	No	none	Number of pairs currently in the define function (set up in the INTRAN file)
REPEAT	Yes	none	Sets the define function to be periodic {YES/NO}
X(i)	Yes	none	The i^{th} X value in the DEFINE FUNCTION
Y(i)	Yes	none	The i^{th} Y value in the DEFINE FUNCTION
DEL(i)	Yes	none	Deletes the i^{th} pair in the define function if poked to YES
NEW_X	Yes	none	If poked with a value X, computes Y = FUNC(X) and inserts the pair into the define function

DEFINE.PATH (DP) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ND	No	none	Number of devices specified in the path in the INPREP file
D(i)	No	none	i^{th} device specified
NP	No	none	Total number of devices in the path (including nodes)
P(i)	No	none	i^{th} device in the path

DEFINE.SEQUENCE (DS) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
TIME(i)	No	none	Start time for the i^{th} scheduled occurrence of the sequence
DEL(i)	Yes	none	Poking to YES means delete the i^{th} scheduled occurrence of the sequence.
arg(i)	Yes	none	Argument assignment associate with the i^{th} schedule-interval. You can poke it to be either string or numerical value.

DEFINE.TIMETABLE (TT) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
SCHED	No	none	1 if | Time - a schedule-time |vDTMIN/2, or current interval has EVERY = CONTINUOUS; 0 otherwise
NEXTTIME	No	none	next schedule-time.
PERIOD	Yes	none	Overall period for the timetable, i.e., REPEAT-clause.
START (i)	Yes	none	The i^{th} start time, all start times are ALWAYS sorted in ascending order, i==0 is the currently active start time.
EVERY (i)	Yes	none	Scheduled period associated with the i^{th} schedule-interval.
DEL(i)	Yes	none	Peek value is always NO, when it is poked to YES, the i^{th} schedule-interval is deleted.
DUP(i)	Yes	none	Peek value is always NO, when it is poked to YES, the i^{th} schedule-interval and the associated values are copied into a new (i+1) schedule-interval, the old (i+1) schedule-interval is moved to (i+2), etc.
arg(i)	Yes	none	Argument assignment associate with the i^{th} schedule-interval. You can poke it to be either string or numerical value.

Derivative relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	Yes	none	Relay input signal **Note:** Pokable only if a constant was entered in the input.
OUT	No	none	Relay output signal
JR	Yes	none	Maximum ramp rate
VS	Yes	none	Relay output bias
STYP	No	none	Type of relay (DERIV)

External (P-CONTROL/Q-CONTROL/Q(P)) (E) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P	No	PRESSURE	Pressure at external
ST	Yes	TEMP	TEMP set point. Always the TEMP of the fluid when it flows into the pipeline.
Q	No	FLOW	Flow through external, standard conditions
CF	Yes	VOLUME	Cumulative flow through external
F	No	AFLOW	Flow though external, pipeline conditions
NN+	No	none	Node where external is connected
NQ	No	FLOW	Nominal flow; same as Q, except for SALEs where it is -Q
NF	No	AFLOW	Nominal flow; same as F, except for SALEs where it is –F
STYP	No	none	Type of external
WOB	No	HEAT.VALUE	Wobbe number
SWOB	Yes	HEAT.VALUE	Set point Wobbe number
NWOB	No	HEAT.VALUE	Node Wobbe number
SG	No	none	Actual specific gravity of the gas
SSG	Yes	none	Set point actual specific gravity of the gas
NSG	No	none	Node actual specific gravity of the gas
LHV	No	HEAT.VALUE	Calculated lowest heating value
SLHV	Yes	HEAT.VALUE	Set point calculated lowest heating value
NLHV	No	HEAT.VALUE	Node calculated lowest heating value

Attribute	Pokable	Units	Description
HHV	No	HEAT.VALUE	Calculated higher heating value
SHHV	Yes	HEAT.VALUE	Set point calculated higher heating value
NHHV	No	HEAT.VALUE	Node calculated higher heating value
SSG	No	none	Set point specific gravity of the gas (as input in INPREP)
T	Yes	TEMP	Actual TEMP of the fluid, which is flow direction dependent. When fluid flows into the system, :T = :ST; otherwise, :T = :NT.
NT	No	TEMP	Node TEMP
VP	No	PRESSURE	Vapor pressure
SVP	Yes	PRESSURE	Set point vapor pressure
NVP	No	PRESSURE	Node vapor pressure
<FL1>	No	none	Actual composition of the first fluid input in the SCL or BWRS equation of state at the node. Use the name of the fluid as the key.
<SFL1>	Yes	none	Set point composition of the first fluid input in the SCL or BWRS equation of state at the external. Use the name of the fluid as the key.
<NFL1>	No	none	Node actual composition of the first fluid input in the SCL or BWRS equation of state at the node. Use the name of the fluid as the key.
<FL2>	No	none	Actual composition of the input in the SCL or BWRS equation of state at the node second fluid. Use the name of the fluid as the key.
<SFL2>	Yes	none	Set point composition of the second fluid input in the SCL or BWRS equation of state at the external. Use the name of the fluid as the key.
<NFL2>	No	none	Node actual composition of the input in the SCL or BWRS equation of state at the node second fluid. Use the name of the fluid as the key.

External (SALE/TAKE) (E) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P	Yes	PRESSURE	Pressure at the external
SP	Yes	PRESSURE	Set point pressure at the external
ST	Yes	TEMP	TEMP set point. Always the TEMP of the fluid when it flows into the pipeline.

Attribute	Pokable	Units	Description
Q	Yes	FLOW	Flow through external, standard conditions
CF	Yes	VOLUME	Cumulative flow through external
F	No	AFLOW	Flow though external, pipeline conditions
NN+	No	none	Node where the external is connected
P0	No	PRESSURE	Base pressure of the fluid
SP0	Yes	PRESSURE	Set point base pressure of the fluid
NP0	No	PRESSURE	Node base pressure of the fluid
T0	No	TEMP	Base temp of the fluid
ST0	Yes	TEMP	Set point base temp of the fluid
NT0	No	TEMP	Node base temp of the fluid
D0	No	DENSITY	Density of fluid at P0 and T0
SD0	Yes	DENSITY	Set point density of fluid at P0 and T0
ND0	No	DENSITY	Node density of fluid at P0 and T0
PM0	No	BULK.MOD	Bulk modulus of the fluid at P0 and T0
SPM0	Yes	BULK.MOD	Set point bulk modulus of the fluid at P0 and T0
NPM0	No	BULK.MOD	Node bulk modulus of the fluid at P0 and T0
TM0	No	ΔTEMP	Temperature modulus of the fluid at P0 and T0
STM0	Yes	ΔTEMP	Set point temperature modulus of the fluid at P0 and T0
NTM0	No	ΔTEMP	Node temperature modulus of the fluid at P0 and T0
PTM	No	none	Multiplier for the ΔPΔT term
SPTM	Yes	none	Set point multiplier for the ΔPΔT term
NPTM	No	none	Node multiplier for the ΔPΔT term
PPM	No	none	Multiplier for the $(\Delta P)^2$ term
SPPM	Yes	none	Set point multiplier for the $(\Delta P)^2$ term
NPPM	No	none	Node multiplier for the $(\Delta P)^2$ term
V0	No	VISCOSITY	Base viscosity of the fluid
SV0	Yes	VISCOSITY	Set point base viscosity of the fluid
NV0	No	VISCOSITY	Node base viscosity of the fluid
VPMI	No	1/PRESSURE	Pressure coefficient of viscosity
SVPMI	Yes	1/PRESSURE	Set point pressure coefficient of viscosity
NVPMI	No	1/PRESSURE	Node pressure coefficient of viscosity
VTMI	No	1/TEMP	Temperature coefficient of viscosity
SVTMI	Yes	1/TEMP	Set point temperature coefficient of viscosity
NVTMI	No	1/TEMP	Node temperature coefficient of viscosity

Attribute	Pokable	Units	Description
VISC	No	VISCOSITY	Viscosity
SVISC	No	VISCOSITY	Set point viscosity
NVISC	No	VISCOSITY	Node viscosity
ASTMA	No	none	Constant for ASTM formula of viscosity
SASTMA	No	none	Set point constant for ASTM formula of viscosity
NASTMA	No	none	Node constant for ASTM formula of viscosity
ASTMB	No	none	Constant for ASTM formula of viscosity
SASTMB	No	none	Set point constant for ASTM formula of viscosity
NASTMB	No	none	Node constant for ASTM formula of viscosity
HC	No	HEAT.CAPACITY	Heat capacity of the fluid
SHC	Yes	HEAT.CAPACITY	Set point heat capacity of the fluid
NHC	No	HEAT.CAPACITY	Node heat capacity of the fluid
FLU	No	none	Name of the fluid
SFLU	Yes	none	Name of the set point fluid. **Note:** The fluid must be present in the external definition, even if it has no mass fraction.
NFLU	No	none	Name of the node fluid
NQ	Yes	FLOW	Nominal flow; same as Q, except for SALEs where it is –Q
NF	No	AFLOW	Nominal flow; same as F, except for SALEs where it is –F
SQ	Yes	FLOW	Set point flow
SNQ	Yes	FLOW	Set point nominal flow
DP	No	PRESSURE	Difference between SP and P+
DQ	No	FLOW	Difference between SQ and Q
PMAX	Yes	PRESSURE	Highest allowable pressure
PMIN	Yes	PRESSURE	Lowest allowable pressure
QMAX	Yes	FLOW	Highest allowable flow
QMIN	Yes	FLOW	Lowest allowable flow
STYP	No	none	Type of external (SALE or TAKE)
WOB	No	HEAT.VALUE	Wobbe number
SWOB	Yes	HEAT.VALUE	Set point Wobbe number
NWOB	No	HEAT.VALUE	Node Wobbe number
SG	No	none	Actual specific gravity of the gas
SSG	Yes	none	Set point specific gravity of the gas

Attribute	Pokable	Units	Description
NSG	No	none	Node specific gravity of the gas
LHV	No	HEAT.VALUE	Calculated gas lower heating value
SLHV	Yes	HEAT.VALUE	Set point calculated gas lower heating value
NLHV	No	HEAT.VALUE	Node calculated gas lower heating value
HHV	No	HEAT.VALUE	Calculated gas higher heating value
SHHV	Yes	HEAT.VALUE	Set point gas higher heating value
NHHV	No	HEAT.VALUE	Node gas higher heating value
T	Yes	TEMP	Actual TEMP of the fluid, which is flow direction dependent. When fluid flows into the system, :T = :ST; otherwise, :T = :NT.
NT	No	TEMP	Node TEMP
H	No	T.FLOW	Thermal flow
NH	No	T.FLOW	Nominal thermal flow
SH	Yes	T.FLOW	Thermal flow set point
SNH	Yes	T.FLOW	Nominal thermal flow set point
DH	No	T.FLOW	Thermal flow discrepancy
HMAX	Yes	T.FLOW	Maximum thermal flow
HMIN	Yes	T.FLOW	Minimum thermal flow
NCMC	Yes	none	Number of control mode changes
MODE	No	none	Control mode
VP	No	$PRESSURE_{abs}$	Vapor pressure
SVP	Yes	$PRESSURE_{abs}$	Set point vapor pressure
NVP	No	$PRESSURE_{abs}$	Node vapor pressure
<FL1>	No	none	Actual composition of the first fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
S<FL1>	Yes	none	Set point composition of the first fluid input in the SCL, AGA or BWRS equation of state at the external. Use the name of the fluid as the key.
N<FL1>	No	none	Node composition of the first fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
<FL2>	No	none	Actual composition of the second fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.

Attribute	Pokable	Units	Description
S<FL2>	Yes	none	Set point composition of the second fluid input in the SCL, AGA or BWRS equation of state at the external. Use the name of the fluid as the key.
N<FL2>	No	none	Node composition of the second fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
DEN	No	DENSITY	Flowing density
NDEN	No	DENSITY	Node density
SDEN	Yes	DENSITY	Set point density
QERR	No	FLOW	Flow modeling error
SVL0	No	VELOCITY	Flowing reference sonic velocity
NSVL0	No	VELOCITY	Node reference sonic velocity
SSVL0	Yes	VELOCITY	Set point reference sonic velocity

Feedback relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	No	none	Feedback relay input signal
IN2	No	none	Feedback relay input signal
IN3	No	none	Feedback relay input signal
IN4	No	none	Feedback relay input signal
IN5	No	none	Feedback relay input signal
OUT	No	none	Feedback relay output signal
VH	No	none	High source signal
VL	No	none	Low source signal
STYP	No	none	Type of relay (FEEDBACK)

Flow meter (FM) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P±	No	PRESSURE	Pressure at the upstream (-) and downstream (+) connections
T±	No	TEMP	Temperature at the upstream (-) and downstream (+) connections
Q±	Yes	FLOW	Flow at the upstream (-) and downstream (+) connections, standard conditions
PD	No	ΔPRESSURE	Pressure drop across the element
CF	Yes	VOLUME	Cumulative flow through the element
F±	No	AFLOW	Flow at the upstream (-) and downstream (+) connections, pipeline conditions
SQ	Yes	FLOW	Set point flow
CB	No	none	Controlling SCADA element

Fluid name (FL) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P0	No	PRESSURE	Base pressure of the fluid
T0	No	TEMP	Base TEMP of the fluid
D0	No	DENSITY	Base density of the fluid
PM0	No	ΔPRESSURE	Bulk modulus of the fluid at P0 and T0
TM0	No	ΔTEMP	TEMP modulus of the fluid at P0 and T0
PTM	No	none	Multiplier for the ΔPΔT term
PPM	No	none	Multiplier for the $(\Delta P)^2$ term
V0	No	VISCOSITY	Base viscosity of the fluid
VPMI	No	1/ΔPRESSURE	Pressure coefficient of viscosity
VTMI	No	1/ΔTEMP	Temperature coefficient of viscosity
ASTMA	No	none	Constant for ASTM formula of viscosity.
ASTMB	No	none	Constant for ASTM formula of viscosity.
HC	No	HEAT.CAP	Heat capacity of the fluid
HV	No	HEAT.VAL	Heating value of the fluid
FC	No	none	Friction correction factor

Attribute	Pokable	Units	Description
COLOR	Yes	none	Color used to display fluid on batch bars. If entered, the string must be enclosed in single quotation marks (‘ ’).
VP1	No	PRESSURE	User-defined vapor pressure
VT1	No	TEMP	Corresponding temperature
VP2	No	PRESSURE	User-defined vapor pressure
VT2	No	TEMP	Corresponding temperature

General compressor (GC) attributes

Also see “Common peek and poke attributes” on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
STATUS	Yes	none	Status
MODE	No	none	Control mode
PU	No	PRESSURE	Upstream pressure
PSMIN	Yes	PRESSURE	Suction pressure set point
PD	No	PRESSURE	Downstream pressure
PDMAX	Yes	PRESSURE	Discharge pressure set point
RATIO	No	none	Compression ratio
RMAX	Yes	none	Maximum ratio
RMIN	Yes	none	Minimum ratio
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow
QMAX	Yes	FLOW	Maximum flow
QMIN	Yes	FLOW	Minimum flow
FU	No	AFLOW	Actual upstream flow
FD	No	AFLOW	Actual downstream flow
QF	No	FLOW	Fuel flow
HF	No	TFLOW	Thermal fuel flow
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
POWER	No	POWER	Power
SPPW	Yes	POWER	Power set point
PWMAX	Yes	POWER	Maximum power

Attribute	Pokable	Units	Description
K1	Yes	POWER/FLOW	Input constant
K2	Yes	POWER/FLOW	Input constant
K3	Yes	none	Input constant
START	No	TIME	Start time
STOP	No	TIME	Stop time
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature
NPOLY	Yes	none	Polytropic exponent.
TRR	Yes	none	Temperature rise ratio

General pipe (GP) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
LEN	Yes	LENGTH.PIPE	Length
OD	Yes	DIAM	Outside diameter
DIAM	Yes	DIAM	Inside diameter
WT	Yes	WALL	Wall thickness
PU	Yes	PRESSURE	Upstream pressure
PD	Yes	PRESSURE	Downstream pressure
DP	No	ΔPRESSURE	Pressure drop
QU	No	FLOW	Upstream flow
QD	No	FLOW	Downstream flow
FU	No	AFLOW	Actual upstream flow
FD	No	AFLOW	Actual downstream flow
HU	No	TFLOW	Upstream thermal flow
HD	No	TFLOW	Downstream thermal flow
ROUGH	Yes	ROUGHNESS	Roughness
FRIC	No	none	Friction factor
EFF	No	EFFICIENCY	Pipe efficiency
PACK	No	LINEPACK	Line pack
VU	No	VELOCITY	Upstream velocity

Attribute	Pokable	Units	Description
VD	No	VELOCITY	Downstream velocity
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature
OHTC	No	none	Overall heat transfer coefficient

GLOBALS (GB) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
TIME	No	TIME	Current simulation time
BEGIN.TIME	No	TIME	Start time for the simulation
END.TIME	No	TIME	End time for the simulation
DELAY	Yes	TIME	Minimum real time between steps
DT	No	TIME	Time step
GIMSTART	Yes	none	Global inertia multipliers for pump start
GIMSTOP	Yes	none	Global inertia multipliers for pump stop
DTMIN	No	TIME	Minimum allowable step size
DTMAX	Yes	TIME	Maximum allowable step size
MEMAVAIL	No	none	Memory statistics - total number of words currently free
NUMFREE	No	none	Memory statistics - number of free blocks
MAXBLOCK	No	none	Memory statistics - size in words; of the largest continuous block of free memory
ALARM (i)	No	none	Alarm status
DEBUG	Yes	none	Debug status
MINPRES	Yes	PRESSURE	Lower limit pipe pressure
MAXPRES	Yes	PRESSURE	Upper limit pipe pressure
MINVELO	Yes	VELOCITY	Lower limit pipe velocity
MAXVELO	Yes	VELOCITY	Upper limit pipe velocity
CS.COLAPS	Yes	TIME	COLSEP collapse rate
CS.BAND	Yes	PRESSURE	COLSEP dead band value
CS.RATE	Yes	TIME	COLSEP bubble growth rate
SL.PLOW	Yes	PRESSURE	Pressure set point below which the pipe is in or near slack line flow {0} (online only)

Attribute	Pokable	Units	Description
SL.DTRESET	Yes	TIME	Timer used to determine slack line flow {30 min} (online only)
VPR.SUM	No	none	COLSEP cumulative bubble loss value on closure
DIFFUSE	Yes	none	Used to tune diffusion between batches SCLPROP {1}
BATCHBOUNDARY	Yes	none	Fluid label to determine the batch boundary (the line between the fluids) on the batch tracking bar.
SHOW. PIPEENDS	Yes	none	If YES, display vertical bars at the end of each pipe in the batch tracking bar. If NO, do not display these bars at the end of pipes. {NO}.
DTGRAN	Yes	none	DT granularity diagnostic tool for GL (leave it at the default value of 0).
SYSTIME	n/a	none	Host computer system's clock time (in format hh:mm:ss)
CASENAME	No	none	INTRAN file name

Attribute	Pokable	Units	Description
TRANSSTATUS	No	none	Current status of trans. The TRANSSTATUS descriptions are: {Corresponding TRANSSTATUS value}
			EXITED Trans has been halted/quit. {1}
			RUNNING Trans is taking time steps. {2}
			PAUSED Trans is paused. {3}
			ARCHIVE Trans is writing an archive. {4}
			LOAD_STATUS Trans is reading an archive. {5}
			WAIT.SCADA Trans is waiting for SCADA system data. {6}
			WAIT.KEYBOARD Trans is waiting for the keyboard enter command. {7}
			WAIT.SPAWNED The SPAWN command has been executed during Trans. {8}
			WAIT.FILELOCK Trans is waiting to get a lock on the REVIEW file. {9}
			WAIT.STOPPED Trans has been suspended by<control>-z from the UNIX system prompt. {10}
TRANSISTATUS	No	none	Numerical value of TRANSSTATUS which can be used in INTRAN logic.
PIPE.EXTRAP	Yes	none	Obsolete
MEM.DYNAMIC	No	none	Memory statistics - total dynamic memory in use

Attribute	Pokable	Units	Description
DISTPLOT.HIST	Yes	none	Number of history curves on a distance plot. The default is four curves. You may select from one to 25 curves. A value of 0 (zero) reverts to the behavior of previous versions.
INTERACTIVE. COMMAND	Yes	none	Setting INTERACTIVE.COMMAND to a string value causes that string to be executed. Example “SET INTERACTIVE.COMMAND = 'RUN FOR 1 STEP'”.
PRINT_START	Yes	TIME	Time delay for an OUTTRN report to be generated.
SPEED	Yes	none	Multiplier to speed up or slow down a Trainer simulation. A speed greater than one will speed up the simulation; a speed less than one will slow down the simulation.
LAG	No	TIME	How far the model time step is behind real time, including the SPEED factor, for a Trainer model. Displayed in clock format.

Grove G887 relief valve (V) attributes

Also see “Common peek and poke attributes” on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of valve
P+	No	PRESSURE	Pressure at to-end of valve
T-	No	TEMP	TEMP at from-end of valve
T+	No	TEMP	TEMP at to-end of valve
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
FR	No	FRACTION	Valve fraction open
CF	Yes	VOLUME	Cumulative flow through valve
ST	No	none	Valve status. The valve status descriptions are: {Corresponding integer value} • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PD	No	ΔPRESSURE	Pressure drop across valve: (P-) - (P+)

Attribute	Pokable	Units	Description
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
A1	Yes	VALVE.COEF	Flow factor used to calculate the valve coefficient for valve flow from the from-end to the to-end
B	Yes	none	Flow equation exponent
C1	Yes	VALVE.COEF	Flow coefficient for valve flow from the from-end to the to-end when back pressure is limiting the flow
PS0	Yes	PRESSURE	Lowest value of sleeve pressure
PS1	Yes	PRESSURE	Highest value of sleeve pressure
A2	Yes	VALVE.COEF	Flow factor or constant used to calculate the valve coefficient for valve flow from to-end to from-end
C2	Yes	VALVE.COEF	Flow coefficient for valve flow from to-end to from-end when back pressure is limiting the flow
CB	No	none	Name of controlling actuator, if applicable

Guide vane compressor (KV) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Suction pressure
P+	No	PRESSURE	Discharge pressure
SP-	Yes	PRESSURE	Suction pressure set point
MNP-	Yes	PRESSURE	Minimum suction pressure constraint
SP+	Yes	PRESSURE	Discharge pressure set point
MXP+	Yes	PRESSURE	Maximum discharge pressure constraint
PI-	No	PRESSURE	Internal suction pressure
PI+	No	PRESSURE	Internal discharge pressure
ST	Yes	none	Unit status. The unit status descriptions are: {Corresponding integer value} • STOPPED – Unit is stopped. {1} • STARTING – Unit is starting. {2} • RUNNING – Unit is running. {3} • STOPPING – Unit is stopping. {4}
PD	No	ΔPRESSURE	Differential pressure
Q-	No	FLOW	Suction flow, standard conditions

Attribute	Pokable	Units	Description
Q+	No	FLOW	Discharge flow, standard conditions
QI	No	FLOW	Through flow, standard conditions
QR	No	FLOW	Recycle flow, standard conditions
QF	No	FUEL	Fuel flow, standard conditions
SQ+	Yes	FLOW	Flow set point, standard conditions
F-	No	AFLOW	Suction flow, pipeline conditions
F+	No	AFLOW	Discharge flow, pipeline conditions
FI	No	COMP.AFLOW	Inlet flow, Suction conditions
T-	No	TEMP	Suction TEMP
T+	No	TEMP	Discharge TEMP
MXT+	Yes	TEMP	Maximum discharge TEMP
SA	Yes	ANGLE	Set vane angle for the compressor
ANG	No	ANGLE	Guide vane angle
MNA	Yes	ANGLE	Minimum permissible vane angle
MXA	Yes	ANGLE	Maximum permissible vane angle
PWR	No	POWER	The brake power delivered
MXPW	Yes	POWER	Maximum available power
MNPW	Yes	POWER	Minimum available power
SPWR	Yes	POWER	Power set point
NRN	Yes	none	Number of units running
HD	No	HEAD	Head
S	No	SPEED	Speed
R	No	none	Compression ratio
RN	No	none	Compression ratio based on node pressures
HE	No	EFFICIENCY	Hydraulic efficiency
HEC	Yes	none	Multiplier for hydraulic efficiency curve data
ME	Yes	EFFICIENCY	Mechanical efficiency
LHV	Yes	HEAT.VAL	Lower heating value for fuel gas
MNE	No	EFFICIENCY	Minimum efficiency
MXE	No	EFFICIENCY	Maximum efficiency
MODE	No	none	What is controlling compressor operation
VCNS	No	none	Violated constraints
SCV	No	none	Series check valve status
BCV	No	none	Bypass check valve status
AMT	Yes	TIME	Time to move minimum angle to maximum angle.

Attribute	Pokable	Units	Description
TRR	Yes	none	Temperature rise ratio
NPOLY	Yes	none	Polytropic exponent. If you are using the AGA or BWRS equation of state, entering a poke value other than zero will hold the value constant. If, however, you enter a zero, then the calculated value will be used.
CB	No	none	Name of controlling actuator, if applicable
NCMC	Yes	none	Number of control mode changes
FSRG	No	AFLOW	Actual flow corresponding to surge line
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
HF	No	TFLOW	Thermal fuel flow

Header (H) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of header
P+	No	PRESSURE	Pressure at to-end of header
T-	No	TEMP	TEMP at from-end of header
T+	No	TEMP	TEMP at to-end of header
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
PD	No	ΔPRESSURE	Pressure drop of header: (P-) - (P+)
CF	Yes	VOLUME	Cumulative flow through header
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
TMP	Yes	TEMP	Outlet TEMP of the simple heat exchanger
DLT	Yes	ΔTEMP	Difference between outlet TEMP and inlet TEMP of the simple heat exchanger
DTY	No	DUTY	Duty (valid when the header is a simple heat exchanger)
LEN	Yes	LEN.HEADER	Length of header
OD	Yes	DIAM	Outside diameter of header
WT	Yes	WALL	Wall thickness of header pipe
H-	No	T.FLOW	Thermal flow into from-end of header
H+	No	T.FLOW	Thermal flow out of to-end of header

Heat exchanger (HE) attributes

Also see "Common peek and poke attributes" on page 641.

Note: A heat exchanger entered in INPREP becomes two devices, one for the tube-side, and one for the shell-side. If the INPREP heat exchanger is called name, the tube-side exchanger is called `name.t` and the shell-side exchanger is called `name.s`.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at from-end of heat exchanger
P+	No	PRESSURE	Pressure at to-end of heat exchanger
T-	No	TEMP	TEMP at from-end of heat exchanger
T+	No	TEMP	TEMP at to-end of heat exchanger
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
PD	No	ΔPRESSURE	Pressure drop across the heat exchanger: (P-) - (P+)
CF	Yes	VOLUME	Cumulative flow through header
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
DTY	No	DUTY	Duty

HI/LO select relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	Yes	none	First input signal **Note:** Pokable only if a constant was entered in the input.
IN2	Yes	none	Second input signal **Note:** Pokable only if a constant was entered in the input.
IN3	Yes	none	Third input signal **Note:** Pokable only if a constant was entered in the input.
IN4	Yes	none	Fourth input signal **Note:** Pokable only if a constant was entered in the input.
IN5	Yes	none	Fifth input signal **Note:** Pokable only if a constant was entered in the input.
IN6	Yes	none	Sixth input signal **Note:** Pokable only if a constant was entered in the input.

Attribute	Pokable	Units	Description
OUT	No	none	Relay output
STYP	No	none	Type of relay (HI or LO)

Idealized regulator (RE) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Pressure at -side of regulator
P+	No	PRESSURE	Pressure at +side of regulator
T-	No	TEMP	TEMP at from-end of regulator
T+	No	TEMP	TEMP at to-end of regulator
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
FR	No	FRACTION	Regulator valve fraction open
CF	Yes	VOLUME	Cumulative flow through regulator
ST	No	none	Regulator valve status. The valve status descriptions are {Corresponding integer value}: • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PD	No	ΔPRESSURE	Pressure drop across regulator: (P-) - (P+)
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
SCV	No	none	Status of series check valve (appears only when the series check valve option is selected)
CV	No	VALVE.COEF	Actual valve coefficient.
CVC	Yes	VALVE.COEF	Valve coefficient at full-closed position
CVO	Yes	VALVE.COEF	Valve coefficient at full-open position
TT	Yes	TIME	Regulator valve full travel time
SQ	Yes	FLOW	Flow set point, Standard conditions
SP-	Yes	PRESSURE	Upstream pressure set point
SP+	Yes	PRESSURE	Downstream pressure set point
SFR	Yes	FRACTION	Fraction open set point
MODE	No	none	Control mode (SQ, SP-, SP+, TT, SFR, OPEN, CLOSED)

Attribute	Pokable	Units	Description
VCNS	No	none	Violated constraints
NCMC	Yes	none	Number of control mode changes

Idealized regulator (RG) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
ST	No	none	Status
MODE	No	none	Control mode
QTYPE	No	none	Flow type
CHECK	No	none	Status of in-line check valve
PU	No	PRESSURE	Upstream pressure
SPU	Yes	PRESSURE	Upstream pressure set point
PD	No	PRESSURE	Downstream pressure
SPD	Yes	PRESSURE	Downstream pressure set point
RATIO	No	none	Pressure ratio
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow
SQ	Yes	FLOW	Flow set point
FU	No	AFLOW	Actual upstream flow
FD	No	AFLOW	Actual downstream flow
FR	No	FR	Stem position
CG	No	VALVE.CG	Actual valve coefficient.
CGMAX	Yes	VALVE.CG	Max valve coefficient (open)
CGMIN	Yes	VALVE.CG	Min valve coefficient (close)
TT	Yes	TIME	Travel time
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature

Input reference (I) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
OUT	No	none	Input reference output signal
SC	Yes	none	SCALE multiplier for time (poke this to change the output value)
ST	Yes	TIME	START time for the input reference (I)
DLY	Yes	TIME	Delay time for activation
SP	Yes	none	Set point for activation
IN	No	none	Constant input

Integrator relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	No	none	Relay input signal
OUT	Yes	none	Relay output signal
STYP	No	none	Type of relay (INTEG)

MAXMIN (MM) attributes

For more information on the attributes of a MAXMIN device, see "MAXMIN" on page 479. Also see "Common peek and poke attributes" on page 641.

Multiply relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	Yes	none	First input signal **Note:** Pokable only if BIAS was entered in the input.
IN2	No	none	Second input signal
IN3	No	none	Third input signal
IN4	No	none	Fourth input signal
IN5	No	none	Fifth input signal

Attribute	Pokable	Units	Description
IN6	No	none	Sixth input signal
IN7	No	none	Seventh input signal
OUT	No	none	Relay output
M1	Yes	none	First input signal multiplier
M2	Yes	none	Second input signal multiplier
M3	Yes	none	Third input signal multiplier
M4	Yes	none	Fourth input signal multiplier
M5	Yes	none	Fifth input signal multiplier
M6	Yes	none	Sixth input signal multiplier
M7	Yes	none	Seventh input signal multiplier
STYP	No	none	Type of relay (MULTIPLY)

Node (NO) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P	Yes	PRESSURE	Pressure at the node
SP	Yes	PRESSURE	Set point pressure at the node
ST	Yes	TEMP	TEMP set point. Always the TEMP of the fluid when it flows into the pipeline.
Q	Yes	FLOW	External flow through node, standard conditions
CF	Yes	VOLUME	Cumulative external flow through node
F	No	AFLOW	External flow through node, pipeline conditions
CNCS	No	none	Connected devices
P0	No	PRESSURE	Base pressure of the fluid
SP0	Yes	PRESSURE	Set point base pressure of the fluid
NP0	No	PRESSURE	Node base pressure of the fluid
T0	No	TEMP	Base temp of the fluid
ST0	Yes	TEMP	Set point base temp of the fluid
NT0	No	TEMP	Node base temp of the fluid
D0	No	DENSITY	Density of fluid at P0 and T0
SD0	Yes	DENSITY	Set point density of fluid at P0 and T0
ND0	No	DENSITY	Node density of fluid at P0 and T0
PM0	No	BULK.MODULUS	Bulk modulus of the fluid at P0 and T0

Attribute	Pokable	Units	Description
SPM0	Yes	BULK.MODULUS	Set point bulk modulus of the fluid at P0 and T0
NPM0	No	BULK.MODULUS	Node bulk modulus of the fluid at P0 and T0
TM0	No	ΔTEMP	Temperature modulus of the fluid at P0 and T0
STM0	Yes	ΔTEMP	Set point temperature modulus of the fluid at P0 and T0
NTM0	No	ΔTEMP	Node temperature modulus of the fluid at P0 and T0
PTM	No	none	Multiplier for the ΔPΔT term
SPTM	Yes	none	Set point multiplier for the ΔPΔT term
NPTM	No	none	Node multiplier for the ΔPΔT term
PPM	No	none	Multiplier for the $(\Delta P)^2$ term
SPPM	Yes	none	Set point multiplier for the $(\Delta P)^2$ term
NPPM	No	none	Node multiplier for the $(\Delta P)^2$ term
V0	No	VISCOSITY	Base viscosity of the fluid
SV0	Yes	VISCOSITY	Set point base viscosity of the fluid
NV0	No	VISCOSITY	Node base viscosity of the fluid
VPMI	No	1/ΔPRESSURE	Pressure coefficient of viscosity
SVPMI	Yes	1/ΔPRESSURE	Set point pressure coefficient of viscosity
NVPMI	No	1/ΔPRESSURE	Node pressure coefficient of viscosity
VTMI	No	1/ΔTEMP	Temperature coefficient of viscosity
SVTMI	Yes	1/ΔTEMP	Set point temperature coefficient of viscosity
NVTMI	No	1/ΔTEMP	Node temperature coefficient of viscosity
VISC	No	VISCOSITY	Viscosity
SVISC	No	VISCOSITY	Set point viscosity
NVISC	No	VISCOSITY	Node viscosity
ASTMA	No	none	Constant for ASTM formula of viscosity
SASTMA	No	none	Set point constant for ASTM formula of viscosity
NASTMA	No	none	Node constant for ASTM formula of viscosity
ASTMB	No	none	Constant for ASTM formula of viscosity
SASTMB	No	none	Set point constant for ASTM formula of viscosity
NASTMB	No	none	Node constant for ASTM formula of viscosity
HC	No	HEAT.CAPACITY	Heat capacity of the fluid
SHC	Yes	HEAT.CAPACITY	Set point heat capacity of the fluid
NHC	No	HEAT.CAPACITY	Node heat capacity of the fluid
FLU	No	none	Name of the fluid

Attribute	Pokable	Units	Description
SFLU	Yes	none	Name of the set point fluid **Note:** The fluid must be present in the external definition, even if it has no mass fraction.
NFLU	No	none	Name of the node fluid
NQ	Yes	FLOW	Nominal flow; same as Q, except for SALEs where it is –Q
NF	No	AFLOW	Nominal flow; same as F, except for SALEs where it is –F
SQ	Yes	FLOW	Set point flow
SNQ	Yes	FLOW	Set point nominal flow
DP	No	PRESSURE	Difference between SP and P+
DQ	No	FLOW	Difference between SQ and Q
PMAX	Yes	PRESSURE	Highest allowable pressure
PMIN	Yes	PRESSURE	Lowest allowable pressure
QMAX	Yes	FLOW	Highest allowable flow
QMIN	Yes	FLOW	Lowest allowable flow
STYP	No	none	Type of external (SALE or TAKE). Always TAKE for nodes.
WOB	No	HEAT.VALUE	Wobbe number
SWOB	Yes	HEAT.VALUE	Set point Wobbe number
NWOB	No	HEAT.VALUE	Node Wobbe number
SG	No	none	Actual specific gravity of the gas
SSG	Yes	none	Set point specific gravity of the gas
NSG	No	none	Node specific gravity of the gas
LHV	No	HEAT.VALUE	Calculated gas lower heating value
SLHV	Yes	HEAT.VALUE	Set point calculated gas lower heating value
NLHV	No	HEAT.VALUE	Node calculated gas lower heating value
HHV	No	HEAT.VALUE	Calculated gas higher heating value
SHHV	Yes	HEAT.VALUE	Set point gas higher heating value
NHHV	No	HEAT.VALUE	Node gas higher heating value
T	Yes	TEMP	Actual TEMP of the fluid, which is flow direction dependent. When fluid flows into the system, :T = :ST; otherwise, :T = :NT.
NT	No	TEMP	Node TEMP
H	No	T.FLOW	Thermal flow
NH	No	T.FLOW	Nominal thermal flow

Attribute	Pokable	Units	Description
SH	Yes	T.FLOW	Thermal flow set point
SNH	Yes	T.FLOW	Nominal thermal flow set point
DH	No	T.FLOW	Thermal flow discrepancy
HMAX	Yes	T.FLOW	Maximum thermal flow
HMIN	Yes	T.FLOW	Minimum thermal flow
NCMC	Yes	none	Number of control mode changes
MODE	No	none	Control mode
VP	No	PRESSURE	Vapor pressure
SVP	Yes	PRESSURE	Set point vapor pressure
NVP	No	PRESSURE	Node vapor pressure
<FL1>	No	none	Actual composition of the first fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
<SFL1>	Yes	none	Set point composition of the first fluid input in the SCL, AGA or BWRS equation of state at the external. Use the name of the fluid as the key.
<NFL1>	No	none	Node composition of the first fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
<FL2>	No	none	Actual composition of the second fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
<SFL2>	Yes	none	Set point composition of the second fluid input in the SCL, AGA or BWRS equation of state at the external. Use the name of the fluid as the key.
<NFL2>	No	none	Node composition of the second fluid input in the SCL, AGA or BWRS equation of state at the node. Use the name of the fluid as the key.
DEN	No	DENSITY	Flowing density
NDEN	No	DENSITY	Node density
SDEN	Yes	DENSITY	Set point density
QERR	No	FLOW	Flow modeling error
SVL0	No	VELOCITY	Flowing reference sonic velocity
NSVL0	No	VELOCITY	Node reference sonic velocity
SSVL0	Yes	VELOCITY	Set point reference sonic velocity

Noise relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	No	none	Noise Generator input signal
OUT	No	none	Noise Generator output signal
SC	Yes	none	Scale of noise introduced
PER	Yes	SIG.TIME	Period on which the noise is changed
STYP	No	none	Type of relay (NOISE)

Plot limits (PL) attributes

For more information, see the explanation of dplot and tplot in "SHOW" on page 559. Also see "Common peek and poke attributes" on page 641.

Pump (P) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Suction pressure of unit
P+	No	PRESSURE	Discharge pressure of unit
T-	No	TEMP	Suction temperature of unit
T+	No	TEMP	Discharge temperature of unit
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
PWR	No	POWER	Driver power used by unit. Power is calculated using the pump head and flow plus either the power curve or the efficiency curve.
HYPW	No	POWER	Hydraulic power used by unit. Hydraulic power is calculated based on the moment of inertia of the pump.
ST	Yes	none	Pump status. The pump status descriptions are: {Corresponding integer value} • STOPPED – Pump is stopped {1} • STARTING – Pump is starting. {2} • RUNNING – Pump is running. {3} • STOPPING – Pump is stopping. {4}

Attribute	Pokable	Units	Description
RPM	No	SPEED	Speed of unit
TQ	No	TORQUE	Driver torque
HTQ	No	TORQUE	Hydraulic torque
HYPR	No	POWER	Hydraulic driver power with respect to the torque of the unit.
PD	No	ΔPRESSURE	Pressure drop across unit: (P-) - (P+)
HD	No	HEAD	Head developed by unit
F-	No	PUMP.AFLOW	Flow into from-end, pipeline conditions
F+	No	PUMP.AFLOW	Flow out of to-end, pipeline conditions
PI-	No	PRESSURE	Suction pressure for integrated assembly
PI+	No	PRESSURE	Discharge pressure for integrated assembly
TI-	No	TEMP	Suction temperature for integrated assembly
TI+	No	TEMP	Discharge temperature for integrated assembly
QB-	No	FLOW	Suction standard flow through integrated bypass check valve
QB+	No	FLOW	Discharge standard flow through integrated bypass check valve
FB-	No	AFLOW	Suction actual flow through integrated bypass check valve
FB+	No	AFLOW	Discharge actual flow through integrated bypass check valve
DBV	No	none	Status of discharge block valve
DCV	No	none	Status of discharge check valve
SBV	No	none	Status of suction block valve
SCV	No	none	Status of suction check valve
BCV	No	none	Status of bypass check valve
EFF	No	EFFICIENCY	Hydraulic efficiency
MXS	Yes	SPEED	Maximum (rated) speed constraint
MXPW	Yes	POWER	Maximum (rated) driver power
MNE	No	EFFICIENCY	Minimum hydraulic efficiency (for water)
MXE	No	EFFICIENCY	Maximum hydraulic efficiency (for water)
STG	Yes	none	Number of stages for viscosity correction
VCQ	No	none	Flow viscosity correction factor
VCE	No	none	Efficiency viscosity correction factor
VCH	No	none	Head viscosity correction factor
MODE	No	none	What is controlling pump operation
VCNS	No	none	Violated constraints
NCMC	Yes	none	Number of control mode changes
CB	No	none	Name of controlling actuator, if applicable

Reciprocating compressor (RC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
FROM	No	none	From-node (upstream)
TO	No	none	To-node (downstream)
ST	Yes	none	Status
CB	No	none	Name of controlling actuator, if applicable
MODE	No	none	Control mode
NCMC	Yes	none	Number of control mode changes
VCNS	No	none	Violated constraints
PU	No	PRESSURE	Upstream pressure
PSMIN	Yes	PRESSURE	Suction pressure set point
PD	No	PRESSURE	Downstream pressure
PDMAX	Yes	PRESSURE	Discharge pressure set point
RATIO	No	none	Compression ratio
RMAX	Yes	none	Max compression ratio
HD	No	ELEV	Head
DP	No	ΔPRESSURE	Differential pressure
Q	No	FLOW	Flow
SPQ	Yes	FLOW	Flow set point
FU	No	AFLOW	Actual upstream flow
FD	No	AFLOW	Actual downstream flow
QF	No	FLOW	Fuel flow
HF	No	TFLOW	Thermal fuel flow
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
S	No	SPEED	Speed
SPS	Yes	SPEED	Speed set point
SMAX	Yes	SPEED	Max speed
SMIN	Yes	SPEED	Min speed
POWER	No	POWER	Compressor power
PWMAX	Yes	POWER	Max power
ME	No	EFFICIENCY	Mechanical efficiency
CE	No	EFFICIENCY	Compressor efficiency
VE	No	EFFICIENCY	Volumetric efficiency

Attribute	Pokable	Units	Description
LC	Yes	none	Volumetric efficiency correction constant
LM	Yes	none	Volumetric efficiency correction multiplier
SV	No	SWEPT.VOL	Swept volume
CL	No	none	Clearance ratio
STEP	Yes	none	Unloading step number - poke changes Unloading Goal to MANUAL.
UG	Yes	none	Unloading goal (SPT/SPS/MANUAL)
SPT	Yes	FR.TORQUE	Torque set point
TQ	No	FR.TORQUE	Fraction rated torque
TQMAX	Yes	FR.TORQUE	Max fraction rated torque
TQMIN	Yes	FR.TORQUE	Min fraction rated torque
TTB	Yes	TIME	Time torque bad
NPOLY	Yes	none	Polytropic exponent.
TRR	Yes	none	Temperature rise ratio
LHV	No	HEAT.VALUE	Lower heating value
DBS	Yes	SPEED	Speed dead band
DBT	Yes	FR.TORQUE	Torque dead band
TODB	Yes	TIME	Time outside (active) dead band
START	Yes	TIME	Start time
STOP	Yes	TIME	Stop time
DEL	Yes	TIME	Trip delay and unloading delay
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature

Relief valve (RV) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
PU	No	PRESSURE	Upstream pressure
PD	No	PRESSURE	Downstream pressure
TU	No	TEMP	Upstream temperature
TD	No	TEMP	Downstream temperature
QU	No	FLOW	Upstream flow, standard conditions
QD	No	FLOW	Downstream flow, standard conditions

Attribute	Pokable	Units	Description
FU	No	AFLOW	Actual upstream flow pipeline conditions
FD	No	AFLOW	Actual downstream flow, pipeline conditions
ST	No	n/a	Valve status. The valve status descriptions are: {Corresponding integer value} • CLOSED – Valve is fully closed. {2} • OPENING – Valve is opening. {3} • OPENED – Valve is fully opened. {4} • CLOSING – Valve is closing. {5}
PO	Yes	PRESSURE	Opening pressure set point
PC	Yes	PRESSURE	Closing pressure set point
MODE	No	n/a	Control mode
NCMC	Yes	n/a	Number of control mode changes
VCNS	No	n/a	Number of violated constraints
FR	No	n/a	Fraction open
VEL	No	SPEED	Velocity
DIAM	Yes	LENGTH	Diameter
PDROP	No	PRESSURE	Pressure drop across valve: (PU) - (PD)
CF	Yes	FLOW	Cumulative flow
NNU	No	n/a	Upstream node
NND	No	n/a	Downstream node
CV	No	n/a	Current valve coefficient
CVO	Yes	n/a	Open valve coefficient
CVC	Yes	n/a	Closed valve coefficient
TTO	Yes	TIME	Opening travel time
TTC	Yes	TIME	Closing travel time
NCO	Yes	n/a	Opening normalization constant
NCC	Yes	n/a	Closing normalization constant
DF	No	FLOW	Diagnostic bypass flow

Sensor (S) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN	No	special	Sensor input signal, units appropriate for quantity being sensed.
OUT	No	none	Sensor output signal
STYP	No	none	Type of sensor

Shared memory (SH) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
SH.SIZE	No	none	Shared memory size
SH.MEMAVAIL	No	none	Free table size
SH.NUMFREE	No	none	Number of free blocks
SH.NUMUSED	No	none	Number of used blocks
SH.MAXBLOCK	No	none	Largest available block
SH.FREEPTR	No	none	Address of free memory table
SH.HIWATER	No	none	Highest memory used
SH.NUMFAIL	No	none	Number of SHMALO fails
SH.NUMREQ	No	none	Number of SHARE requests
SH.WAIT	No	none	TPORT update period

Station (ST) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
NDEV	No	none	Number of devices, including externals
DEV(i)	No	none	Name of the devices
NNOD	No	none	Number of nodes, including boundary
NOD(i)	No	none	Name of the nodes
NPE	No	none	Number of pipe ends
PE(i)	No	none	Name of the pipe ends

Attribute	Pokable	Units	Description
QAST	No	none	Status (QAS relative to QASMX). • If QAS < QASMX, then QAST = LOW • If QASMX <= QAS <= 10*QASMX, then QAST = MEDIUM • If 10*QASMX < QAS, then QAST = HIGH Used for online modeling.
EXCL	Yes	none	YES | NO switch to determine whether to exclude <node>:DQ from QS and QAS calculations. Default is NO. Used for online modeling.
QS	No	FLOW	Sum of all <monitor>:Q and, if EXCL=NO, all <node>:DQ flows. Used for online modeling.
QAS	No	FLOW	Sum of the absolute values of each <monitor>:Q and, if EXCL=NO, the absolute value of each <node>:DQ flows. Used for online modeling.
QASMX	Yes	FLOW	User-entered flow rate used to determine :QAST. Used for online modeling.

Span (SP) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
NDEV	No	none	Number of devices
DEV	No	none	Name of the devices
LEN	No	LENGTH.PIPE	Total length of the span
P-	No	PRESSURE	Pressure at from-end of span
P+	No	PRESSURE	Pressure at to-end of span
T-	No	TEMP	TEMP at from-end of span
T+	No	TEMP	TEMP at to-end of span
Q-	No	FLOW	Flow into from-end, standard conditions
Q+	No	FLOW	Flow out of to-end, standard conditions
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
FPD	No	ΔPRESSURE	Total friction pressure drop
EFF	No	none	Pipe efficiency. EFF = SQRT(1/(1+FC)).
FC	Yes	none	Friction correction factor

Attribute	Pokable	Units	Description
FCR	No	none	Friction correction ratio
PD	No	ΔPRESSURE	Pressure drop of span: (P-) - (P+)
PK	No	FLOW	Pack rate of span: (Q-) - (Q+), standard conditions
PKF	No	AFLOW	Pack rate of span: (F-) - (F+), pipeline conditions
INV	No	VOLUME	Span inventory
NBAT	No	none	Number of batch interfaces in span
PKM	No	MFLOW	Pack rate of span
FFE	No	none	Effective friction factor
LPDT	No	TIME	Low pressure delta time
NINT	No	none	Number of pipe intervals (# pipe knots - 1)
FLU(i)	No	none	Name of fluid(s) in span
ID1(i)	No	none	Additional fluid identifier
ID2(i)	No	none	Additional fluid identifier
DT	No	TIME	Nominated time step
GPD	No	ΔPRESSURE	Total gravitational pressure drop

Surge tank (E) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P	No	PRESSURE	Pressure
ST	No	TEMP	Temperature set point
Q	No	FLOW	Flow
CF	Yes	VOLUME	Cumulative flow
F	No	AFLOW	Actual flow
NN+	No	none	Node where surge tank is connected
P0	No	PRESSURE	Fluid reference pressure
SP0	Yes	PRESSURE	Set point fluid reference pressure
NP0	No	PRESSURE	Node fluid reference pressure
T0	No	TEMP	Fluid reference temperature
ST0	Yes	TEMP	Set point fluid reference temperature
NT0	No	TEMP	Node fluid reference temperature
D0	No	DENSITY	Reference density
SD0	Yes	DENSITY	Set point reference density

Attribute	Pokable	Units	Description
ND0	No	DENSITY	Node reference density
PM0	No	BULK.MOD	Fluid reference bulk modulus
SPM0	Yes	BULK.MOD	Set point fluid reference bulk modulus
NPM0	No	BULK.MOD	Node fluid reference bulk modulus
TM0	No	ΔTEMP	Fluid reference temperature modulus
STM0	Yes	ΔTEMP	Set point fluid reference temperature modulus
NTM0	No	ΔTEMP	Node fluid reference temperature modulus
PTM	No	none	Multiplier for the ΔPΔT term
SPTM	Yes	none	Set point multiplier for the ΔPΔT term
NPTM	No	none	Node multiplier for the ΔPΔT term
PPM	No	none	Multiplier for the $(\Delta P)^2$ term
SPPM	No	none	Set point multiplier for the $(\Delta P)^2$ term
NPPM	No	none	Node multiplier for the $(\Delta P)^2$ term
V0	No	VISCOSITY	Fluid reference viscosity
SV0	Yes	VISCOSITY	Set point fluid reference viscosity
NV0	No	VISCOSITY	Node fluid reference viscosity
VPMI	No	none	Viscosity pressure coefficient
SVPMI	Yes	none	Set point viscosity pressure coefficient
NVPMI	No	none	Node viscosity pressure coefficient
VTMI	No	none	Viscosity temperature coefficient
SVTMI	Yes	none	Set point viscosity temperature coefficient
NVTMI	No	none	Node viscosity temperature coefficient
HC	No	HEAT.CAPA	Fluid heat capacity
SHC	Yes	HEAT.CAPA	Set point fluid heat capacity
NHC	No	HEAT.CAPA	Node fluid heat capacity
NQ	No	FLOW	Nominal flow
NF	No	AFLOW	Nominal actual flow
STYP	No	none	Subtype
LHV	No	HEAT.VALUE	Low heating value
SLHV	Yes	HEAT.VALUE	Set point low heating value
NLHV	No	HEAT.VALUE	Node low heating value
HHV	No	HEAT.VALUE	High heating value
SHHV	Yes	HEAT.VALUE	Set point high heating value
NHHV	No	HEAT.VALUE	Node high heating value

Attribute	Pokable	Units	Description
T		TEMP	Flowing temperature
NT	No	TEMP	Node temperature
VP	No	PRESSURE	Vapor pressure
SVP	No	PRESSURE	Set point vapor pressure
NVP	No	PRESSURE	Node vapor pressure

Switch relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	Yes	none	Switch input signal **Note:** Pokable only if a constant was entered in the input.
IN2	Yes	none	Switch input signal **Note:** Pokable only if a constant was entered in the input.
OUT	No	none	Relay output signal
SS	Yes	none	Switch set point signal
C1	Yes	none	Limit value for input signal 1
C2	Yes	none	Limit value for input signal 2
DLY	Yes	TIME	Time delay for changing the connection switch
STYP	No	none	Type of relay (SWITCH)

Tank (TK) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ST	No	none	Status: EMPTY, FULL, EMPTYING, FILLING, STATIC, OVER_EMPTY, SPILLING
PI	No	PRESSURE	Inlet node pressure. (Two connection tanks only.)
PO	No	PRESSURE	Outlet node pressure. (Two connection tanks only.)
P	No	PRESSURE	Node pressure. (Single-connection tanks only.)
PB	No	PRESSURE	Internal bottom pressure
T	Yes	TEMPERATURE	Internal temperature
NQ	No	FLOW	Nominal flow into tank. (Single-connection tanks only.)
Q	No	FLOW	Flow into node. (Single-connection tanks only.)

Attribute	Pokable	Units	Description
NQI	No	FLOW	Nominal flow in through inlet. (Two-connection tanks only.)
QI	No	FLOW	Flow out through inlet. (Two-connection tanks only.)
NQO	No	FLOW	Nominal flow out through outlet. (Two-connection tanks only.)
QO	No	FLOW	Flow out through outlet. (Two-connection tanks only.)
SPQ	No	FLOW	Spill rate
SPVL	Yes	VOLUME	Spill inventory
CF	Yes	VOLUME	Cumulative flow into tank
ETE	No	TIME	Estimated time until empty
ETF	No	TIME	Estimated time until full
INV	Yes	VOLUME	Fluid inventory (standard conditions)
VOL	Yes	VOLUME	Fluid volume (tank conditions)
RVOL	No	VOLUME	Remaining volume
LEV	Yes	ELEVATION	Fluid height
HT	Yes	ELEVATION	Tank height
HIHI	Yes	VOLUME	Nominal full volume
HI	Yes	VOLUME	Capacity
LO	Yes	VOLUME	Nominal empty volume
LOLO	Yes	VOLUME	Volume at outlet height
NNU	No	none	Inlet node. (Two-connection tanks only.)
NND	No	none	Outlet node. (Two-connection tanks only.)
NN	No	none	Node. (Single-connection tanks only.)
FLU	Yes	none	Dominant fluid

Theoretical horsepower flow compressor (KP) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	No	PRESSURE	Suction pressure of unit
PI-	No	PRESSURE	Intermediate suction pressure of unit
PI+	No	PRESSURE	Intermediate discharge pressure of unit
P+	No	PRESSURE	Discharge pressure of unit

Attribute	Pokable	Units	Description
Q-	No	FLOW	Suction flow, standard conditions
Q+	No	FLOW	Discharge flow, standard conditions
T-	No	TEMP	Suction temperature
T+	No	TEMP	Discharge temperature
ST	Yes	none	Status of the compressor. The unit status descriptions are: {Corresponding integer value} • STOPPED – Unit is stopped. {1} • STARTING – Unit is starting. {2} • RUNNING – Unit is running. {3} • STOPPING – Unit is stopping. {4}
R	No	none	Compression ratio
RN	No	none	Compression ratio based on node pressures
PWR	No	POWER	Power used by unit
QF	No	FUEL	Fuel flow, standard conditions
MODE	No	none	Operating mode of the unit
BCV	No	none	Status of the bypass
QB	No	FLOW	Bypass flow
PD	No	ΔPRESSURE	Pressure drop for compressor: (P-) - (P+)
F-	No	AFLOW	Actual flow into from-end
F+	No	AFLOW	Actual flow into to-end
SP-	Yes	PRESSURE	Suction pressure set point
SP+	Yes	PRESSURE	Discharge pressure set point
SQ+	Yes	FLOW	Flow set point
SR	Yes	FRACTION	Ratio set point
SPW	Yes	POWER	Power set point
NRN	Yes	none	Number of running units
MNP-	Yes	PRESSURE	Minimum suction pressure
MXP+	Yes	PRESSURE	Maximum discharge pressure
MNQ	Yes	FLOW	Minimum flow through unit
MXQ	Yes	FLOW	Maximum flow through unit
MNR	Yes	none	Minimum compression ratio
MXR	Yes	none	Maximum compression ratio
MNPW	Yes	POWER	Minimum compressor power
MXPW	Yes	POWER	Maximum compressor power
MXIP	No	POWER	Maximum rate of increase in power

Attribute	Pokable	Units	Description
MXDP	No	POWER	Maximum rate of decrease in power
RP	Yes	POWER	Rated power
K1	Yes	POWER/FLOW	Compressor equation coefficient
K2	Yes	POWER/FLOW	Compressor equation coefficient
K3	Yes	none	Compressor equation exponent
EFF	Yes	EFFICIENCY	Efficiency of the compressor
NPOLY	Yes	none	Polytropic exponent.
LHV	No	HEAT.VALUE	Lower heating value of fuel gas
HR	Yes	HEAT.RATE	Heat rate (Not pokable when entered as a curve.)
TRR	Yes	none	Temperature rise ratio
VCNS	No	none	Violated constraints
SCV	No	none	Series check valve status
NCMC	Yes	none	Number of control mode changes
HF	No	TFLOW	Thermal fuel flow

Time-averaging relay (Y) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
IN1	No	TIME	Input signal
OUT	No	TIME	Output signal
PER	Yes	TIME	Time period
STYP	No	none	Type of relay (AVERAGE)

Transfer line (T) attributes

Attribute	Pokable	Units	Description
P-	Yes	PRESSURE	Pressure at from-end of pipe
P+	Yes	PRESSURE	Pressure at to-end of pipe
T-	Yes	TEMP	TEMP at from-end of pipe
T+	Yes	TEMP	TEMP at to-end of pipe
Q-	No	FLOW	Flow into from-end, Standard conditions
Q+	No	FLOW	Flow out of to-end, Standard conditions
PD	No	ΔPRESSURE	Pressure drop of pipe: (P-) - (P+)

Attribute	Pokable	Units	Description
PK	No	FLOW	Pack rate of pipe: (Q-) - (Q+)
LEN	Yes	LENGTH.PIPE	Length of pipe
OD	Yes	DIAM	Outside diameter of pipe
WT	Yes	WALL	Wall thickness of pipe
V-	No	VELOCITY	Fluid velocity into from-end of pipe
V+	No	VELOCITY	Fluid velocity out of to-end of pipe
F-	No	AFLOW	Flow into from-end, pipeline conditions
F+	No	AFLOW	Flow out of to-end, pipeline conditions
PKF	No	AFLOW	Pack rate of pipe: (F-) - (F+)
FLU(i)	Yes	none	Name of fluid(s) in pipe
ETA(i)	No	none	Estimated time of arrival of batch interface
POS(i)	Yes	LENGTH.PIPE	Position of batch interface
INV	No	LINEPACK	Pipe inventory
AMP	Yes	none	Flow area multiplier
FF	Yes	none	Interval-by-interval average of the effective friction factor
RUF	Yes	ROUGHNESS	Pipe roughness
EFF	No	none	Pipe efficiency. EFF = SQRT(1/(1+FC))
FC	Yes	none	Friction correction factor **Note:** Pokable only if spans are not enabled. (See "SELECT (INPREP)" on page 211.) If you change the span:FC setting, you can change all of the :FC in all of the pipes in the span. For more information, see "Span (SP) attributes" on page 686.
FPD	No	ΔPRESSURE	Frictional pressure drop
GPD	No	ΔPRESSURE	Gravitational pressure drop
ID1(i)	Yes	none	Additional fluid identifier
ID2(i)	Yes	none	Additional fluid identifier
VPR	No	none	Percentage of pipe volume that is vapor.
NBAT	No	none	Number of batch interfaces in the pipe
BPP(i)	No	LENGTH.PIPE	Position of batch interface
HMUL	Yes	none	Holdup multiplier (for tuning slack line flow)
HMIN	Yes	none	Holdup minimum (for tuning slack line flow)
MERR	No	none	Bound on mass imbalance due to slack line flow modeling error (numerical truncation)
PKM	No	MFLOW	Pack rate of pipe

Attribute	Pokable	Units	Description
FFE	No	none	Effective friction factor
NINT	No	none	Number of pipe intervals (# pipe knots - 1)
TG-	Yes	TEMP	Inlet ground layer TEMP
TG+	Yes	TEMP	Outlet ground layer TEMP (When TG- or TG+ is poked, the ground TEMP along the pipe becomes the linear interpolation of :TG- and :TG+)
VP-	No	$PRESSURE_{abs}$	Vapor pressure at from-end of pipe
VP+	No	$PRESSURE_{abs}$	Vapor pressure at to-end of pipe
WAX	Yes	WALL	Average wax thickness
DT	No	TIME	Nominated time step
E-	No	ELEVATION	Elevation at from-end of pipe
E+	No	ELEVATION	Elevation at to-end of pipe
H-	No	T.FLOW	Thermal flow into from-end of pipe
H+	No	T.FLOW	Thermal flow into to-end of pipe
OHTC	No	none	Overall heat transfer coefficient

TRANSTHERMAL (TR) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
TEMP	Yes	TEMP	Default flow TEMP
SP.HEAT	Yes	none	Ratio of heat capacity of fluid to the heat capacity of water
COLBURNA	Yes	none	TRANSTHERMAL scale coefficient in Colburn equation.
COLBURNB	Yes	none	TRANSTHERMAL Reynolds number exponent in Colburn equation
COLBURNC	Yes	none	TRANSTHERMAL Prandtl number exponent in Colburn equation
GRASH.A	Yes	none	Scale coefficient in laminar term.
GRASH.B1	Yes	none	Viscosity ratio exponent for heating.
GRASH.B2	Yes	none	Viscosity ratio exponent for cooling.
GRASH.C	Yes	none	Scale coefficient representing effective d/L value
GRASH.D	Yes	none	Reynolds * Prandtl number exponent in Grashof term.
GRASH.E	Yes	none	Coefficient in laminar term
GRASH.F	Yes	none	Exponent of Grashof *Prandtl number
GRASH.G	Yes	none	Exponent of d/L

Attribute	Pokable	Units	Description
GRASH.H	Yes	none	Exponent in laminar term
CG.RE1	Yes	none	Reynolds number separating laminar and transition region
CG.RE2	Yes	none	Reynolds number separating transition and turbulent region
MIN.FILM	Yes	HEAT.TRANS	Minimum film coefficient
HFE.LH	Yes	none	HFE laminar heating
HFE.LC	Yes	none	Value of HFE for cooling in laminar flow
HFE.TH	Yes	none	Value of HFE for heating in turbulent flow
HFE.TC	Yes	none	Value of HFE for cooling in turbulent flow
FORC.CON	No	none	TEMP used in forced convection term
FREE. CON	No	none	TEMP used in free convection term
layer.HCAPA	Yes	HEAT. CAPACITY	Heat capacity where layer is WALL, WRAP, FILL, or GRND
layer.HCOND	Yes	HEAT.COND	Heat conductivity where layer is WALL, WRAP, FILL, or GRND
TT.RESET	Yes	none	Enables steady-state radial temperature calculation
TT.ERRTOL	Yes	TEMP	TRANSTHERMAL error tolerance

Wax deposition (WX) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
MXR	Yes	ROUGHNESS	Wax roughness
HCON	Yes	HEAT.COND	Heat conductivity of the wax
DENS	Yes	DENSITY	Density of the wax
MXT	Yes	WALL	Maximum thickness of the wax

CHAPTER

16

Model Builder

Model Builder is a tool that allows you to build and maintain SPS models within a graphical user interface. With Model Builder, you can create and maintain a complete and functional INPREP file, using a visual schematic and dialog box editors. You may edit directly in the model text files if you choose.

Model Builder provides a graphic interface for working with SPS models, including a schematic view, device editors, and a variety of drawing tools to simplify the modeling process. You can create an entirely new model with Model Builder, or you can import an existing model, such as an older INPREP file that was previously created and maintained with a text editor. Model Builder supports the maintenance of all INPREP file features, including facilities data and model settings.

Model Builder also provides access to several SPS-related tasks, including the generation of the RESTRT file with PREPR and the initiation of TRANS. If desired, you can use Model Builder as a comprehensive tool that largely replaces the traditional SPS startup window. Or, you can simply use Model Builder to create and maintain your models, then use the exported INPREP files with the traditional startup window. In either case, analyses will run the same and produce the same results.

See "Before using Model Builder" on page 697 for caveats to using Model Builder.

Before using Model Builder

Before you consider using Model Builder, note the following:

- Model Builder runs on the Windows platform only.
- When a model is imported with Model Builder, all MACRO and IFELSE statements are expanded and not otherwise preserved. If your current workflow relies on these features as is, you should not use Model Builder.
- Model Builder supports the creation and maintenance of INPREP files only. All tasks related to other SPS source files, such as INTRAN files, must be performed directly in the text files with a text editor.
- When you first open an existing INPREP file in Model Builder, it creates an initial schematic based on connectivity. This initial schematic, however, may not look as expected, and you may need to move some elements to create a better visual representation of the system. For more information on moving elements, see "To move an element" on page 711. When you re-save from Model Builder, coordinate information will be added to each element, and it will appear as expected when you reload it. SPS uses this coordinate information only for Model Builder schematics and it has no effect on analysis results.
- Although Model Builder supports a "note" feature meant to facilitate comments, it may not place those comments in the same places that an author in a text editor might. Model Builder assumes that all comments belong above the item to which they are associated. Therefore, if you import an existing file with comments, comments in the exported equivalent might be moved around somewhat.

Steps for using Model Builder

If you choose to use Model Builder, you will likely follow these general steps:

1. Create and build a model within the Model Builder interface.

   —or—

   Import an existing model into Model Builder and perform maintenance as needed.
2. Run the Model Builder validation feature to detect topological errors, such as connectivity problems.
3. Save (export) the model as an INPREP file.
4. Run PREPR to create the RESTRT file, either from the Model Builder window, or from the traditional SPS startup window.
5. View OUTPRP reports and correct any deficiencies found during the PREPR run, until you can produce a valid RESTRT file.
6. Run TRANS on the RESTRT file, either from the Model Builder window, or from the traditional SPS startup window.

These steps are flexible, depending on whether you are:

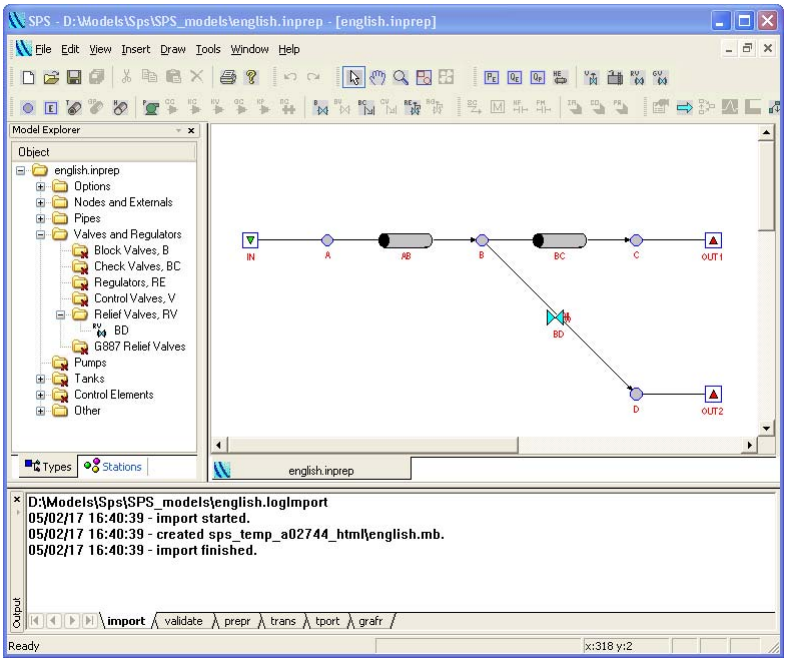
- Doing all modeling exclusively with Model Builder, or continuing to do work directly in the text files.
- Using Model Builder to replace traditional SPS startup window functions, such as running PREPR and TRANS.

Model Builder environment

The Model Builder environment provides several windows, tools, and other features that facilitate the construction and maintenance of SPS models. You may save a desired layout of the Model Builder environment, including the models you work with on a regular basis. You may set other preferences that control Model Builder behavior when building models, running a simulation, and viewing results.

The Model Builder window

The Model Builder window contains menus, toolbar buttons, a model explorer, a schematic view, an output window, a status bar, and editors.



Model Builder window

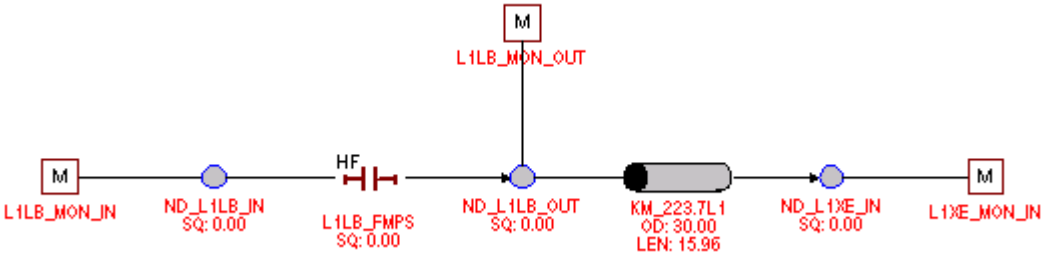
Schematic view

The schematic view provides a visual diagram of your model. It uses symbols and lines to represent elements, including transfer lines and other equipment.

You may display node and element names, select attributes to view, and use shapes to mark up the schematic. For more information, see "Customizing the schematic" on page 726.

For more information on adding, copying, pasting, and deleting elements and stations, see "Building a model" on page 710.

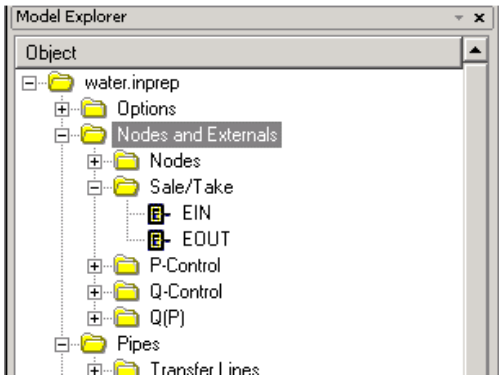
Note: The schematic is not a map, only a connectivity diagram. The size, location, and scale of elements in the schematic view have no relationship to actual physical or geographical characteristics of the model data.



Portion of a sample schematic

Model explorer

The model explorer provides a quick means of accessing model data and locating elements in the schematic. All model items, including parameters such as units and limits, can be accessed within the folder structure of the explorer.



Portion of a sample model in the model explorer

If a folder displays a red x next to it, the folder contains no data. If a folder displays a red exclamation point, the folder contains inappropriate data for the license or phase of the model.

Toolbars

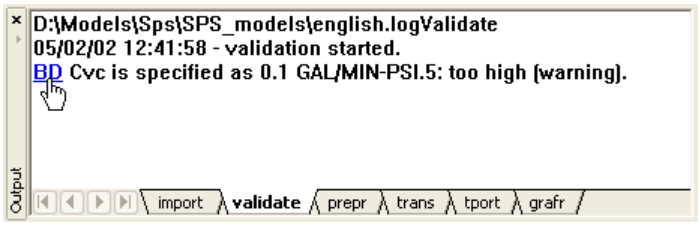
Toolbars provide quick access to commonly-used menu commands. You can hide or show a toolbar by right-clicking in the toolbar area and selecting that toolbar from the popup list. You also may customize toolbars by selecting **Tools > Customize**.

Editors

Editors are dialog boxes that permit model data to be input, including data for multiple devices at once. For more information, see "Editing data through Model Builder" on page 722.

Output window

The output window is a multi-tabbed window for messages and warnings from various Model Builder processes, such as model validation. Links in the output window open the relevant editor.



Certain PREPR error messages may appear in this window as well, although the main OUTPRP file opens in an HTML window in the schematic area following a PREPR run. For more information, see "HTML OUTPRP reports" on page 53.

Status bar

The status bar is a small area at the bottom of the Model Builder window that displays pertinent information, such as the name of the selected item in the schematic, the coordinates of the cursor, and the current zoom percentage.

Layouts

Layouts specify information on window locations, toolbar setup, and the current model, if any. You can use a layout to quickly reconfigure the appearance of the Model Builder environment to your preferred settings.

You will most likely want to configure a blank layout with the buttons, windows, and toolbars that you use and save it in a logical location. Layouts are stored in a WSP file with the name and folder of your choice.

To load or save a layout

Select **File > Layout >** {*command*}.

Tip: To revert to the default SPS layout, select **File > Layout > Factory Layout**.

Setting program options and preferences

Model Builder preferences are a collection of settings that serve two primary purposes:

- Specify certain SPS environment settings associated with models, such as colors, fonts, and time zones.
- Control Model Builder behavior during a modeling session and while running SPS modules, such as PREPR and TRANS.

Setting backup preferences

Model Builder's Backup preferences control how often automatic model backups occur, and how many backup copies are maintained. Backups are stored in a subdirectory that is automatically created under the model directory. Backup files and folders are named by appending a "_bu" suffix to the original name. For example, the backup folder for a model named Water.inprep would be named Water_bu, and the backup files would be named Water_1.inprep_bu, Water_2.inprep_bu, and so on.

You should note the following backups:

- If you are saving files in MB format, Model Builder creates duplicate backups in both INPREP and MB formats. For more information on exported file types, see "Saving model data from Model Builder" on page 735.
- Backups are performed only when you save the model or run the validation module, provided that the minimum wait time has been exceeded.

To set backup preferences

1. From the Model Builder main menu, select **Tools > Preferences**.
2. In the Preferences dialog box, select the **Backup** tab.
3. Use the Backup tab to set the following options:
   - *Number of copies to keep.* Controls how many old backup files are maintained in the backup directory. When the backup limit is reached, each new backup file overwrites the oldest backup file in the directory.
   - *Minimum wait time (minutes).* Controls the minimum time interval, in minutes, between backups.
4. Click **OK** to accept the changes and close the dialog box.

Setting module dependencies in Model Builder

When you run an SPS module through Model Builder, settings in the Dependencies tab control whether different modules run or not, based on the existence of changes to the source file(s) since the last module run. For example, when you attempt to run PREPR, you can have Model Builder prompt you for confirmation first if there have been no changes to the source INPREP file. Or, you can set PREPR to run in any case, regardless of whether there have been changes.

Dependency settings also control the automatic launching of "prerequisite" modules when another module is launched through the Model Builder interface. The concept of prerequisite modules is based on the normal sequence of module usage. For example, PREPR is considered a prerequisite to TRANS, because it produces the source RESTRT file that TRANS needs to function. Therefore, you may want PREPR to run automatically before TRANS, especially if there have been changes to the source INPREP or MB file.

To set module dependencies in Model Builder

1 From the Model Builder main menu, select **Tools > Preferences**.

2 In the Preferences dialog box, select the **Dependencies** tab.

3 Use the options in the Dependencies tab to set module dependency relationships for the Validate, PREPR, and TRANS modules. For each module, you can set different options for when the module is launched directly from Model Builder (by using the Tools menu or the appropriate toolbar button) and when the module is launched automatically as a prerequisite for another module). The following options are available:

Method of module launch	Options
As a command (launching a module directly with the Tools menu or the toolbar button)	• *Always.* The module runs whenever directed, regardless of whether the source file(s) have changed. • *When no changes, ask.* If the source file(s) have not changed, you are prompted for confirmation before the module runs. • *When no changes, don't run.* If the source file(s) have not changed, the module does not run. You do not receive a notification.
As a prerequisite (launching a module automatically as a prerequisite to another) **Note:** It is best to always run Validate as a prerequisite to PREPR.	• *When changes, run.* As applicable, the module runs automatically as a prerequisite if its source file(s) have changed. You do not receive a notification. • *When changes, ask.* As applicable, the module runs automatically as a prerequisite if its source file(s) have changed, following your confirmation. • *Never run as a prerequisite.* Disables the module as an automatic prerequisite.

4 Next to **Tport** and , select whether you want Model Builder to warn you if TRANS input files have changed prior to running TPORT or .

5 Click **OK** to accept the changes and close the dialog box.

Setting up SPS case files

Cases, as described in "Cases" on page 99, are collections of file locations that allows you to specifically direct SPS modules to certain input and output file locations. By default, SPS uses the main model directory for both input files and output files associated with the model. Case files allow you to specify different paths for different file types associated with an SPS model. For example, cases can allow you to use the same source files for different simulations, but produce different output files based on simulation events. Note that cases are available on Windows only and are used by the SPS startup window and/or Model Builder.

If you have a case associated with a model, SPS will load it when you run a module and handle files as specified. To automatically associate a case with a model, the associated case file must have the same root name as the model. For example, if you open the MyModel.inprep file and SPS finds a MyModel.cas file in the same directory, that case will automatically be applied. Otherwise, you must load the case after the model is loaded.

If no case file is associated with a model, SPS assigns a default case to that model, in which all input and output files are given the same root name and stored in the same original directory.

To set up SPS case files

1 From the Model Builder main menu, select **Tools > Preferences**.

2 In the Preferences dialog box, select the **Files** tab.

3 Use the Files tab to perform the following tasks:

To:	Do this:
Create a new case file	Click **New**, then type the name of the case file under **Case file**.
Open an existing case file	Click **Open**, then navigate to the folder of the case file that you want to open.
Change the file location for different SPS file types	In the **Files** area, click the button for the file type that you want to set (such as **.inprep** for INPREP files) and then navigate to the folder where the new file is located.
Save a case file	Click **Save** to save the case file under its current name, or click **Save As** to save the case file with a new name and location. If you are saving a case file for the first time, you will be prompted to select a file name and folder location where the case file will be saved.

4 Click **OK** to accept the changes and close the dialog box.

Setting miscellaneous Model Builder preferences

The Settings tab allows you to specify miscellaneous settings, as listed in the following procedure.

To set miscellaneous Model Builder preferences

1 From the Model Builder main menu, select **Tools > Preferences**.

2 In the Preferences dialog box, select the **Settings** tab.

3 Use the Settings tab to set the following options:

Don't warn changes that are not saved	This option, when selected, disables a warning message that appears when you perform certain Model Builder actions that are not saved with a model, such as grouping, flipping, and rotating objects. The warning messages can also be disabled by selecting the **Suppress this message in the future** check box on the appropriate warning message.
Text editor	Specifies the name of the text editor that will be used to view text input files from Model Builder.
Edit sps.settings	Opens the sps.settings text file, which controls certain settings for all models within the same directory. For more information on the settings contained in an sps.settings file, see "sps.settings files" on page 91.
Size of current drawing area (pixels)	Defines the height and width, in pixels, of the current Model Builder drawing area.

4 Click **OK** to accept the changes and close the dialog box.

Setting color preferences

The Colors tab allows you to choose colors to represent valve and compressor or pump status.

Note: Valve status is based on FR or SFR (for RE and RG). The control valve (V) does not have an initial fraction and therefore it does not have a color assignment. Relief valves (G887 and RV)] also do not have color assignments.

To set Model Builder color preferences

1 From the Model Builder main menu, select **Tools > Preferences**.

2 In the Preferences dialog box, select the **Colors** tab.

3 Click on an existing color that you want to change, and then select a new color in the resulting dialog box. The default colors are as follows:

- Running Unit: Green
- Stopped Unit: Red
- Open Valve: Green
- Closed Valve: Red
- Partially Open Valve: Yellow

4 Click **OK** to accept the changes and close the dialog box.

Creating a new model

You may create a new model even if you already have a model open in Model Builder. Simply select **File > New**. You will most likely want to first set global settings, equation of state, units, and other defaults before you begin adding devices to the schematic. For more information, see "Setting model options" on page 706.

Importing an existing model

You may import an existing Model Builder (MB) file or an INPREP file. Simply select **File > Open/Import** and select the MB or INPREP file you want to work on.

While opening a file is straightforward, migrating an INPREP file that has never been imported into Model Builder will require some additional work. After opening the file, the schematic window will display the basic connectivity of the model, but you will most likely need to refine the spatial representation of the model. You may need to move individual or groups of elements to create a more accurate depiction of the pipeline. For more information, see "To move an element" on page 711 and "Positioning objects in the schematic" on page 730.

In addition to the graphical representation of the model, you may also need to consider the organization and maintenance of your model. Model Builder allows you to maintain include files and to preserve the sequence of data within files through line numbers. For more information, see "Maintaining Include files" on page 736 and "Preserving INPREP sequence using line numbers" on page 738.

Setting model options

Before building a model, you will most likely want to edit global settings, such as the equation of state, units, and limits, among others. You can define and edit this in the Options folder of the model explorer. Topics in this section provide tips for editing data in Model Builder and provide references to the corresponding INPREP syntax for detailed descriptions.

Global Settings editor

In the Global Settings editor, you can set general information such as pipeline custody conditions. If the model is exported to an INPREP file, this editor provides data for the following INPREP commands:

- "TITLE" on page 210
- "CUSTODY" on page 215
- "PIPEPARMS" on page 216

Note: Note that you must set default friction factor or roughness either through the Limits editor or by right-clicking on the field in an editor and selecting Properties. For more information, see "Limits editor" on page 709 and "Setting defaults, limits, and other attribute properties" on page 724.

Equation of State editor

In the Equation of State editor, you can set the phase and the equation of state and associated data. You may also determine whether to turn batch tracking on or off.

If the model is exported to an INPREP file, this editor provides data for one or more of the following INPREP commands:

- "GAS" on page 213 or "LIQUID" on page 214
- "NOTRACK" on page 218
- "STATE AGA" on page 221
- "STATE BWRS" on page 224
- "STATE CNGA" on page 228
- "STATE SCL" on page 230
- "WAX" on page 242
- "STATE TABLE" on page 245

Within traditional INPREP syntax, the SCL, SCLPROP, and BWRS equation of state commands include data fluid definitions. Model Builder continues this syntax; however, the fluids themselves are defined in a separate editor. For more information on defining fluids to accompany these commands, see "Fluids editor" on page 709. For more information on choosing an appropriate equation of state, see "Equations of state" on page 125.

Thermal Modes editor

In the Thermal Modes editor, you can set the thermal mode and associated data. If the model is exported to an INPREP file, this editor provides data for one of the following INPREP commands:

- "ISOTHERMAL" on page 247
- "THERMAL" on page 248
- "TRANSTHERMAL" on page 249

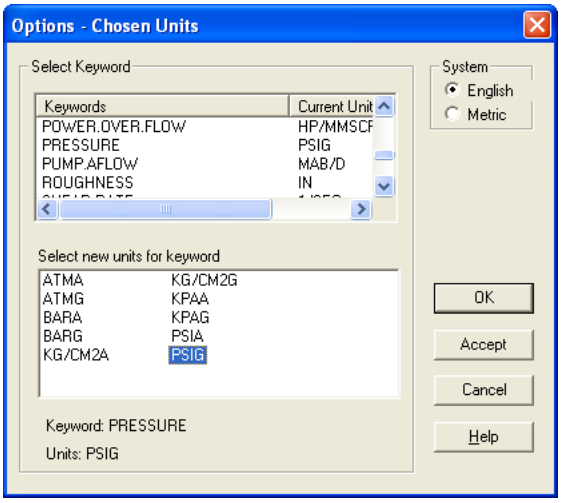
For more information on thermal modes, see "Thermal modes" on page 126.

Select editor

In the Select editor, you choose whether to run the model in off-line or online mode. Depending on what mode you select, different options may appear. For more information, see "SELECT (INPREP)" on page 211.

Chosen Units editor

In the Chosen Units editor, you can specify the units to be used for various model parameters, by assigning actual units to SPS units keywords. For example, if you assign "DC" (°C) to the TEMPERATURE units keyword, all parameters that are associated with the TEMPERATURE keyword will be represented in degrees Celsius, such as PIPE1:T- (temperature at from-end of transfer line PIPE1).



In this editor, you can select either the English or Metric system of measurement for your model.

Note: Note that all units will be reset to factory default settings if you change the system of an existing model.

If the model is exported to an INPREP file, this editor provides data for the following command:

"USEUNITS" on page 258

Note: A USEUNIT command is generated only if you change a units assignment from the factory-specified default. For example, the factory default assignment for the TEMPERATURE keyword is "DF" (°F) or "DC" (°C). Unless you change this assignment, no USEUNIT command will be generated for TEMPERATURE. For more information on default units, see "Default units" on page 129.

To change a units assignment

1 In the model explorer under **Options**, double-click **Chosen Units**.

2 In the editor, select the units keyword you would like to reassign.

3 In the lower box, select the new unit for the keyword.

Tip: The units that appear in this box represent all those that are applicable to the selected keyword, including user-defined units. For example, if you define a unit named "YARDS" based on the built-in unit FEET, YARDS will appear as an option for the LENGTH units keyword, along with normal built-in units. For more information on defining your own units with Model Builder, see "Defined Units editor" on page 708.

4 Click **Accept**.

Defined Units editor

In the Defined Units editor, you can view the mathematical relationships between SPS built-in units, as well as create your own units. If the model is exported to an INPREP file, this editor provides data for the following command:

"DEFUNITS" on page 256

Much of the information displayed in the editor represents the setup of built-in units, and is not found in an INPREP file. Built-in units are factory-configured within SPS and are available for any simulation. Only those units that you specifically create will become DEFUNITS commands in an exported INPREP file. For more information on built-in units, see "Built-in units" on page 131.

To view details on a currently defined unit

1 In the model explorer, under **Options**, double-click **Defined Units** to open the editor.

2 Select the unit in the **Units** box.

The mathematical relationships with other units, built-in and user-defined as applicable, appear to the right.

To define a new unit

1 In the model explorer, under **Options**, double-click **Defined Units** to open the editor.

2 Click **New**.

3 In the **Define Units** dialog box, select the unit that you want to use as the basis for the new unit.

4 In the equation, replace **new_unit** with the new unit name.

5 In the other text boxes, enter the appropriate conversion factors to convert from your new unit to the selected unit.

For example, if you were creating a new unit called "YARD," you could select **FEET** and specify **3** as the multiplication factor. After finishing this procedure and creating the YARD, you could select **FEET** in the **Defined Units** editor and see that its reverse relationship to your new YARD has also been established, as a "multiply by 0.333333" factor.

6 Click **OK**.

To edit, rename, or delete a user-defined unit

Note: You can only edit, rename, or delete the definitions of user-defined units. The mathematical relationships of built-in units are permanent and cannot be altered.

1. In the model explorer, under **Options**, double-click **Defined Units** to open the editor.
2. Select the unit in the **Units** box.
3. Click **Edit**, **Rename**, or **Delete**, as applicable.

Fluids editor

The Fluid editor allows you to define and configure any number of fluids and their parameters for the model when using the SCL or SCLPRP equation of state. For more information on specifying the equation of state for the model, see "Equation of State editor" on page 706.

If the model is exported to an INPREP file, this data is saved in "STATE SCL" on page 230. If you change to any other equation of state, fluid definitions will be lost when you save the model.

You may enter a non-Newtonian viscosity table on the Fluids editor by clicking **Edit** next to **VISC TBL**. For more information, see "Viscosity table editor" on page 709.

Viscosity table editor

The Viscosity table editor allows you to input a table of data for each named fluid. The viscosity table includes viscosity versus shear rate and temperature. If you save as an INPREP file, the data is saved as described in "VISCOSITY (non-Newtonian)" on page 239.

In the Viscosity editor, you may insert and delete rows, undo and redo entries, and copy and paste values to or from an external software product. When copying and pasting values, be sure to select the entire range of data you wish to copy or paste.

Limits editor

The Limits editor allows you to define and configure limits and defaults for the model. You can define any number of limits. If the model is exported to an INPREP file, this editor provides data for the following command:

"SET.LIMIT" on page 219

All limits defined in Model Builder are compiled into a single SET.LIMIT command. In addition, only those limits that are altered from SPS defaults are included in the command.

Tip: You can also set limits for individual attributes directly in the equipment editors, by right-clicking on the attribute in the editor and selecting Properties. These limits perform an identical function as those specified in the Limits editor, and a change in one place is automatically reflected in the other. For more information on this functionality, see "Setting defaults, limits, and other attribute properties" on page 724.

Building a model

After you have set model options, you are ready to begin building the connectivity of the model using elements. In addition to elements, you may need to add other data, such as curves. All model elements and devices can be added from the model explorer and, if applicable, drawn on the schematic.

The size and scale of the model in the schematic have no hydraulic significance in the model. For example, if you stretch the length of a transfer line graphically, the length parameter does not change. You must edit all physical dimensions directly in the editors. However, the spatial coordinates do change based on positioning of elements in the schematic.

The schematic represents the actual connectivity of the model. You should use care when moving and creating lines and nodes to prevent errors due to connectivity problems.

For more information on model building theory, see "Modeling the Physical System" on page 119.

To insert a new element

Tips: You may insert a new element by copying an existing one. For more information, see "Cutting, copying, and pasting" on page 715.

If you want to add an element that extends beyond the drawing area, you can manually adjust the size of the drawing area in the schematic view by selecting **Tools > Preferences > Settings** and entering the number of pixels you want the area to have. You can view the current number of pixels by looking at the status bar at the bottom of the Model Builder window.

You can turn on the grid by selecting **Draw > Grid > Grid**.

1 Select the element type in the **Insert** menu.

—or—

Click the appropriate button on the toolbar.

Tip: Hover the mouse pointer over toolbar buttons to see a tool tip for the button.

2 On the schematic, press and hold the left mouse button while dragging to draw the new element.

Tip: You can draw either an isolated element, or you can have it automatically connected to an existing node by left-clicking directly on that node.

3 Release the left mouse button to finish the element.

While adding elements, note the following:

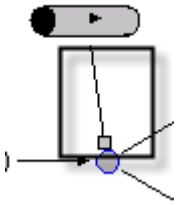
- When adding items from the Insert menu, all elements are automatically placed on the schematic, except heat exchangers.
- In some cases, the editor for the new element will appear. If you change your mind and want to immediately delete the new element, click the Delete button. If you only want to cancel the changes you have made in the editor, click the Cancel Edits button.
- As an alternative to left-clicking and dragging, for many node-connecting elements you can left-click once to designate the from-node, then left-click elsewhere to designate the to-node. If you do not click on a valid node for the to-node, Model Builder will generally create a new node.
- For externals and single-connection tanks, the first left-click designates the location of the element itself, rather than a node.

To move an element

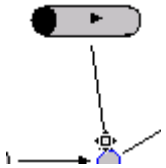
Model Builder tools allow you to:

- Move the location of an element on the schematic, by dragging its symbol. This action applies to the schematic only, and does not affect the hydraulic properties of the model. For more information, see "Positioning objects in the schematic" on page 730.
- Change model connectivity by dragging one of the "legs" of an element. You can drag it to another existing node, or allow Model Builder to create a new node for you.

  Dragging one node to another until they overlap does not change connectivity. You must select the leg as shown below.



The selected "leg" connects an element to a node.



Click cursor on the end of the selected leg and drag it to the desired node or new location.

  This action affects the hydraulic connectivity of the model.

  Note: Actions in the schematic and in the editors can override one another. For example, you can move a transfer line connection on the schematic, then override that move by specifying a different node connection in the transfer line editor. As such, it is recommended that you close any unnecessary editors before you perform schematic operations.

For more information on movement and positioning in the schematic view see "Positioning objects in the schematic" on page 730.

To delete an item from the model

Select the item on the schematic and select **Edit > Delete**.

—or—

Select the item on the schematic and press the **Delete** key on your keyboard.

—or—

Right-click on the item in the model explorer and select **Delete**.

—or—

Open the item in the editor window and click **Delete**.

Element editors

Elements, such as transfer lines and valves, can be added and edited within the Model Builder environment. You can find details on the data required to model elements in the main INPREP documentation. To assist with navigation to those sections, the following table provides information on INPREP commands related to elements. Follow the cross-references for more information on the INPREP commands and their arguments, which represent the same data managed within Model Builder editors.

If you have imported from an INPREP file, the sequence of your data is preserved and recorded as line numbers. When you add or modify data through Model Builder, you may want to specify a particular location to insert the data in the text file. You may do this using line numbers in the editors. For more information, see "Preserving INPREP sequence using line numbers" on page 738.

Furthermore, you may still employ include files for data storage. For more information on using include files within model builder, see "Maintaining Include files" on page 736.

Model Builder element	Cross-reference to INPREP documentation
Node	"NODE" on page 283
Sale/Take External	"E SALE/TAKE" on page 287
P-Control External	"E P-CONTROL" on page 291
Q-Control External	"E Q-CONTROL" on page 293
Q(P) External	"E Q(P)" on page 295
Transfer Line	"Transfer line - transient (T)" on page 263
Header	"Header (H)" on page 276
Heat Exchanger	"Heat exchanger (HE)" on page 281
Block Valve	"Block valve (B)" on page 308
Block Valve	"Block valve (BV)" on page 306
Check Valve	"Check valve (BC)" on page 313
Check Valve	"Check valve (CV)" on page 311
Control Valve	"Control valve (V)" on page 316
Regulator	"Idealized regulator - control valve (RE)" on page 330
Regulator	"General regulator (RG)" on page 329
G887 Relief Valve	"Grove G887 relief valve (V G887)" on page 325
Centrifugal compressor	"Centrifugal compressor (CC)" on page 339
Centrifugal compressor	"Idealized controllable centrifugal compressor (KC)" on page 346
Variable Guide Vane Compressor	"Variable guide vane compressor (KV)" on page 364
Theoretical compressor	"Theoretical horsepower-flow compressor (KP)" on page 359
Theoretical compressor	"General compressor (GC)" on page 343
Reciprocating compressor	"Reciprocating compressor (RC)" on page 371
Pump	"Pump (P)" on page 382

Model Builder element	Cross-reference to INPREP documentation
Atmospheric Tank	"Tank (TK)" on page 303
Surge Tank	"E SURGETANK" on page 300
Sensor	"Sensor (S)" on page 389
Actuator	"Actuator (A)" on page 400
High/Low Select Relay	"HI/LO Select Relay (Y HI/LO)" on page 403
Multiply Relay	"Multiply Relay (Y MULTIPLY)" on page 407
Switch Relay	"Switch Relay (Y SWITCH)" on page 413
Derivative Relay	"Derivative Relay (Y DERIV)" on page 405
Integrating Relay	"Integrator Relay (Y INTEG)" on page 409
Time Averaging Relay	"Time-Averaging Relay (Y AVERAGE)" on page 417
Noise Relay	"Noise Relay (Y NOISE)" on page 415
Feedback Relay	"Feedback Relay (Y FEEDBACK)" on page 411
PID Controller	"P-I-D Controller (C)" on page 396
Input Reference	"Input reference (I) (INPREP)" on page 393
Composition Controller	"Composition controller" on page 763
Flow Meter	"FLOWMETER" on page 279
Header Flow Element	"Header flow (HF)" on page 770
Interface Alignment	"Interface alignment (IA)" on page 773
Monitor	"E MON" on page 786
Property Element	"PROPERTY" on page 788
SCADA	"SCADA" on page 791

For more information on drawing these elements onto the schematic, see "To insert a new element" on page 710.

Other model items

Model Builder provides for other miscellaneous model items and parameters, such as curve and variable definitions. In the model explorer, these items are found in the Other folder.

Curves editor

In the Curve editor with its associated tools, you can define data curves for reference by other elements. If the model is exported to an INPREP file, this editor provides data for the following command:

"Data curve (D)" on page 418

When adding a D command to an INPREP file with a text editor, you must specify each coordinate in the curve as numerical values. You can simplify this process with Model Builder's graphical curve designer.

To use the graphical curve designer

Open the editor for the desired curve and click **Edit Curve**.

In the Edit Curve dialog box, you can perform the following actions:

If you click	Result
Anywhere within the graphical curve window	Populates the X and Y boxes based on where you clicked
Add	Adds the coordinates from the X and Y boxes to the list and updates the graphical curve shown to the right
Delete	Removes the selected coordinates from the list
Undo	Reverts the most recent action in the editor
Copy	Copies the selected coordinates in the list to the clipboard
Paste	Pastes coordinates from the clipboard into the list, if any exist

Tip: You may be able to select all your desired coordinates by clicking within the graphical curve window. However, if the range of the window is too small, you can still enter an X or Y coordinate directly in the X or Y box. If the manually entered value is beyond the range of the current window, the window scale will adjust automatically when you click Add.

Defines editor

In the Define editor, you can define variables for the simulation. If the model is exported to an INPREP file, this editor provides data for the following command:

"DEFINE" on page 568

To add a new defined variable

Select **Insert > Other > Define Variable**.

Paths editor

In the Path editor, you can define a distance path used to generate distance plots. If the model is exported to an INPREP file, this editor provides data for the following command:

"DEFINE.PATH" on page 571

To add a new distance path

Select **Insert > Other > Distance-plot Path**.

Stations editor

In the Station editor, you can name stations for use in distance plots. If the model is exported to an INPREP file, this editor provides data for the following command:

"STATION" on page 421

To add a new station for a distance plot

Select **Insert > Other > Station**.

Cutting, copying, and pasting

You can cut, copy, and paste the following items within Model Builder:

- Elements in the schematic
- Items in the model explorer
- A whole column or row of data into or from a Model Builder editor.

In addition you may copy and paste various alphanumeric data or graphical model representations into other software products.

Copying elements

You can copy and paste one or more element into the following areas:

- The current working model
- A different model in the same Model Builder session
- A different Model Builder session

When you select a device to cut or copy, the selection is extended to include the connected nodes of the selected items. When cutting, the connected nodes will not be deleted if they are used by any other elements. In contrast, when pasting, the connected nodes are included in the paste.

If you cut/copy and paste a device that references another device, the second device is not included in the cut/copy and paste. For example, data curves and control systems are not automatically copied but rather need to be cut/copy and pasted as a second step using the model explorer. For more information, see "To copy and paste items in the model explorer" on page 716. See "Viewing device relationships" on page 720 to identify device references.

To copy and paste an element in the schematic

Tip: If you want to add an element that extends beyond the drawing area, you can manually adjust the size of the drawing area in the schematic view by selecting **Tools > Preferences > Settings** and entering the number of pixels you want the area to have. You can view the current number of pixels by looking at the status bar at the bottom of the Model Builder window.

1. Select the item(s) you want to copy, using one of the selection methods described in "Selecting items" on page 719.
2. With the item(s) selected in the schematic, select **Edit > Copy**, or right-click in the schematic and select **Copy**.
3. Select **Edit > Paste**, or right-click on the schematic and select **Paste**.

   Note: If there is a conflict with the names of the pasted item(s) and the existing item(s), use the Name Conflict dialog box to specify how the item name(s) should be resolved. For more information, see "Resolving name conflicts when copying and pasting in a model" on page 717.

When the item is pasted, it is “floating” unattached to any other elements. You will need to connect the new device to an existing node by double-clicking on the new device and editing the FROM and TO fields or by attaching the connected node to an existing node. For more information see “To move an element” on page 711.

If you have copied and pasted from a different model, the X and Y coordinates for the station may not be positioned ideally. You may select **Edit > Redo XY’s** to have Model Builder reposition the elements to a default position. *Beware that this option will reposition all of the items in the schematic, not just the ones you just pasted.*

To copy and paste items in the model explorer

1 Select the element you want to copy by clicking it once, and select **Edit** > **Copy**.

—or—

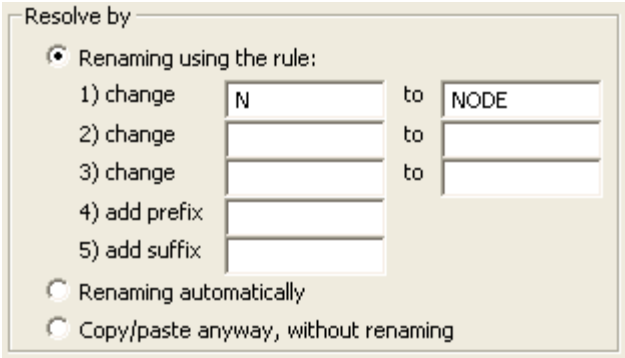
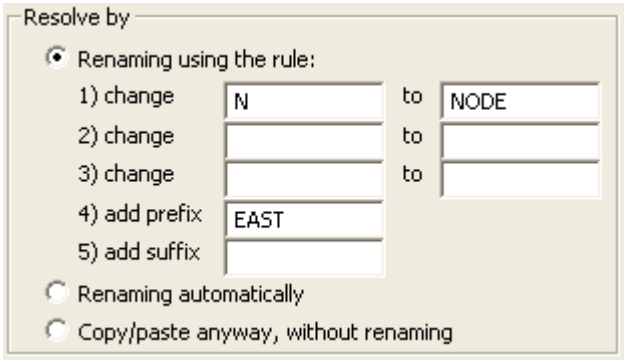
Right-click on the element you want to copy and select **Copy**.

2 Right-click anywhere in the model explorer and select **Paste**.

Note: If there are name conflicts, a dialog box describes the conflicts and provides some options to resolve the issue. For more information, see “Resolving name conflicts when copying and pasting in a model” on page 717.

Resolving name conflicts when copying and pasting in a model

The Name Conflict dialog box appears any time you copy and paste within a model or from one model to another, and the pasted objects share names with existing objects in the model. Use this dialog box to resolve the naming conflicts through any of the methods described below.

Method	How it works
Renaming using rules	You may change up to three character strings found in the listed conflicts. For example, if you wanted to change N to NODE everywhere it occurs in the name conflicts, you would enter the following:  You may also add a suffix or prefix to all of the listed conflicts. For example, if you wanted to also add EAST to each of conflicts, you would enter the following:  Keep in mind that the renaming rules are sequential—that is, rule 2 is applied to the results of rule 1, and rule 3 is applied to the results of rule 2, etc.
Rename automatically	A second option is for Model Builder to automatically rename conflicting devices. When you select this option, Model Builder will automatically add `_1` to the end of each pasted device name. If you paste more than once, the second set of conflicts would be `_2`, etc.
Copy/paste without renaming	With the third option, **Copy/paste anyway, without renaming**, Model Builder will keep the duplicate names. It is very important for you to go back into the editors and rename the devices to avoid conflicts when running the model.

Viewing data and navigating in Model Builder

You may use the model explorer, toolbar buttons, and keyboard shortcuts to locate elements in the schematic, navigate to them, and zoom in to an appropriate level. Furthermore, Model Builder provides tools to show you relationships among devices, link to editors through reports, and view model data as text input. For more information on viewing data through the schematic, see "Customizing the schematic" on page 726.

Zooming and navigation

Model Builder provides tools for you to navigate the schematic and control the magnification. These tools can be found in the View menu, on the toolbar buttons, and by using shortcut key combinations. In addition, the model explorer provides some shortcuts for zooming and navigation.

Note: The current zoom percentage is displayed in the Model Builder status bar.

View menu

Command	Function
Pan	Allows you to pan the schematic by dragging it with the mouse.
Zoom	Allows you to draw a box with the mouse around the area to which you want to zoom.
Zoom to Normal	Zooms to the default magnification (100% zoom).
Zoom to Fit	Scales the model to fit it exactly within the current boundaries of the schematic window.
Zoom to Selection	Scales the model to fit the selected item(s) within the current boundaries of the schematic window.
Zoom Percent	Allows you to select a zoom percentage from a preset list.
Zoom Custom	Allows you to specify a custom zoom percentage.

Toolbar buttons

Select/Pan/Zoom toolbar

Keyboard and mouse shortcuts

Scroll up	**Page Up** Up arrow key Mouse wheel
Scroll down	**Page Down** Down arrow key Mouse wheel
Page left	**Shift+Page Up** Left arrow key **Shift+MouseWheel**
Page right	**Shift+Page Down** Right arrow key **Shift+MouseWheel**
Go to top left	**Ctrl+Home**
Go to bottom right	**Ctrl+End**
Zoom in or out	**Ctrl**+Mouse wheel

Model explorer

If you right-click on an item in the model explorer, the resulting popup menu provides access to functions such as:

- Opening the editor for the item
- Selecting the item in the schematic
- Showing INPREP syntax for the item

Selecting items

To perform actions such as copying and pasting, you must first select the applicable items in the schematic view. In "Select" mode (**View > Select**), you can select items in several ways:

- Click to select individual items
- Hold Shift and click to select multiple items
- Drag a box over the schematic to select anything within that box. By holding Shift, you can draw multiple boxes.
- Right-click on the item in the model explorer and select **Select on Schematic**.
- From the main menu, select **Edit > Select All**.

You may also find the items you want to select by using the Go To dialog box or the Uses/Used By dialog box. For more information, see "To find an element" on page 720 and "Viewing device relationships" on page 720.

Finding model items

You can locate items in the model by sorting and filtering by name, type, or details.

To find an element

Select **Edit > Go To Item** and select the desired item from the list.

- In the **Go To** dialog box, you can sort items by name or type by clicking the button in the upper-left corner of the dialog box.
- You can display the device type in the **Go To** dialog box by clicking the second button in the upper-left corner and then selecting **Details**.
- If you know the name of the device you want to find, you can go directly to the device by typing in the **Device name** text box.
- You can filter the list of items by typing the first and additional letters in the **Filter Pattern** text box.

Viewing device relationships

You can locate items in the model by viewing relationships among various devices. For example, you may view curves associated with an element, SCADA devices that drive an element, and from and to connections. You may navigate the connections through the dialog box to open other editors or select an item on the schematic.

To view a device relationship

1 Select an item in the schematic.

—or—

Select an item in the model explorer.

2 Right-click and select **Uses**.

3 In the **Uses/Used By** dialog box, navigate the relationships by expanding the tree view.

Tips: Select an item in the dialog box and click **Edit** to open the editor or click **Show in Map** to select the item in the schematic.

Viewing model data as text

As an experienced SPS user, you may be used to viewing and working with model data in text files. Sometimes you may prefer to view your data as INPREP syntax or you feel an edit is easier made through the INPREP syntax. In these cases, Model Builder allows you to:

- View the INPREP syntax for a selected element through the output window.
- Open INPREP, INTRAN, and other model data files in your favorite text editor.

To show INPREP syntax for an element

Select the element in the schematic view and select **Edit** > **Show Inprep**.

—or—

Right-click on the item in the model explorer and select **Show Inprep**.

The INPREP syntax displays on the prepr tab of the output window.

To open an INPREP file as a text file

From the Model Builder window, select **File > Open INPREP As Text**.

—or—

Click the Open INPREP as Text toolbar button. 

Notes: The text file opens in your default text editor. To change the default text editor, see "Setting program options and preferences" on page 701.

If you have not saved your recent changes in Model Builder, then those changes may not be reflected in the text file. Also, with both the text file and Model Builder open, you risk overwriting data if you make changes to both of the files. The most recently saved file will overwrite any changes made during a previous save.

To open an INTRAN file as a text file

From the Model Builder window, select **File > Open INTRAN As Text**.

—or—

Click the Open INTRAN as Text toolbar button. 

Notes: The text file opens in your default text editor. To change the default text editor, see "Setting program options and preferences" on page 701.

If you have not saved your recent changes in Model Builder, then those changes may not be reflected in the text file. Also, with both the text file and Model Builder open, you risk overwriting data if you make changes to both of the files. The most recently saved file will overwrite any changes made during a previous save.

To open any model data file as a text file

1. From the Model Builder window, select **File > Open Any As Text**.
2. In the Open dialog box, select the file you want to open. The default file type is INPREP files, but you can use the **Files of type** drop-down list to open INTRAN, INC, OUTTRN, DSP, or TXT files.
3. Click **Open**.

   Notes: The text file opens in your default text editor. To change the default text editor, see "Setting program options and preferences" on page 701.

   If you have not saved your recent changes in Model Builder, then those changes may not be reflected in the text file. Also, with both the text file and Model Builder open, you risk overwriting data if you make changes to both of the files. The most recently saved file will overwrite any changes made during a previous save.

Printing

Model Builder schematics are printable. Use the Page Setup, Print Setup, Print Preview, and Print commands in the File menu to prepare and launch the print job. You can also select whether to print at the current zoom percentage or at the default setting (100% zoom).

Editing data through Model Builder

Model Builder provides a set of standard editors for entering and editing model data. These editors all function in basically the same manner, and look very similar except for the content being edited. In a Model Builder editor, data items have a label and a description. The label is often derived from the syntax of the corresponding attribute found in an INPREP file.

To edit model options and other data in Model Builder

In the model explorer tree view, double-click on the data you want to edit.

—or—

If it is an item that appears on the schematic view, double-click the item symbol.

—or—

Select the item on the schematic and select **Edit** > **Edit Item**.

To edit an element

Find the desired item in the model explorer tree view and double-click on it.

—or—

If it is an item that appears on the schematic view, double-click the item symbol.

—or—

Select **Edit > Go To Item** and select the desired item from the list. For tips on sorting and filtering items, see "To find an element" on page 720.

For more information on using the editors, including tools for multiple item edits, see "Editing data through Model Builder" on page 722.

Types of editors

Standard editors include the following:

- Editors for hydraulic and online elements, described under "Element editors" on page 712
- "Global Settings editor" on page 706
- "Equation of State editor" on page 706
- "Thermal Modes editor" on page 706
- "Select editor" on page 707

- "Chosen Units editor" on page 707
- "Defined Units editor" on page 708
- "Fluids editor" on page 709
- "VISCOSITY (non-Newtonian)" on page 239
- "Limits editor" on page 709
- "Curves editor" on page 713
- "Defines editor" on page 714
- "Paths editor" on page 714
- "Stations editor" on page 715

Symbols in Model Builder editors

Symbols in Model Builder editors help you keep track of whether the data is a user-defined or default value and whether it meets validation criteria:

Symbol	Description
	The value is a user-entered value and is valid.
	The value is optional or was not entered. SPS default is used.
	The value is outside of low to high range.
	The value is outside of low low or high high range.

For more information on setting default properties from the editor, see "Setting defaults, limits, and other attribute properties" on page 724.

Buttons in Model Builder editors

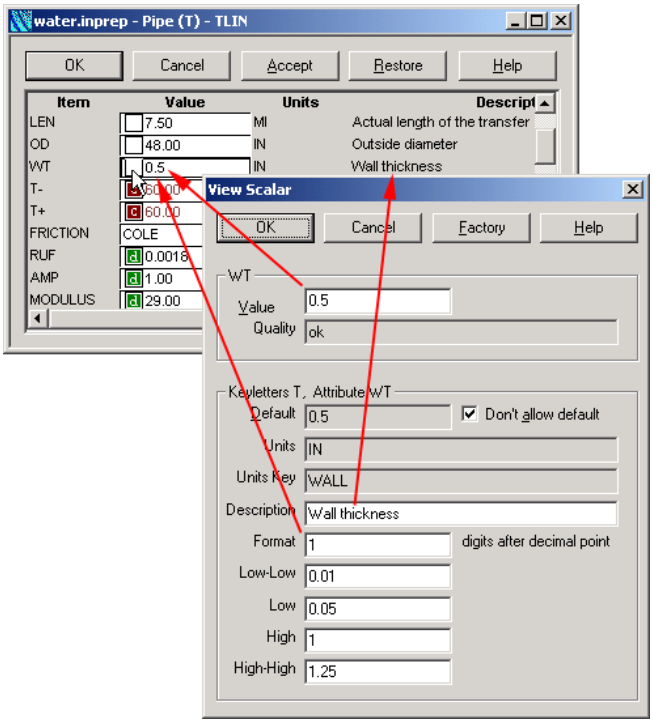
Along with the lists and fields used to specify the data items, a Model Builder editor provides a set of buttons to assist you with data entry. These buttons include:

Button	Function
OK	Saves all data in the editor and closes it.
Accept	Saves all data in the editor and leaves it open.
Restore	Reverts the settings in the editor to the last saved settings, such as the last time **OK** or **Accept** was clicked.
Cancel	Closes the editor without saving any changes to the data.
Insert (multiple editors only)	Inserts a new item. For more information, see "Editing multiple items" on page 724.

Setting defaults, limits, and other attribute properties

In many cases, you can access a secondary editor that allows you to specify more detailed information, such as defaults and limits. To access this secondary editor, right-click in the data entry area of the attribute and select Properties.

For example, the following secondary editor is produced for the WT attribute of a transfer line:



In this secondary editor, you can specify items such as:

- The value of the attribute
- The number of decimal places shown in the main editor
- The description of the main editor
- Defaults and limits

  Note: By specifying limits that are different from factory settings, you will likely cause Model Builder to generate a SET.LIMIT command if the model is exported as an INPREP file. This functionality mirrors that of the Limits editor, accessible through the model explorer. For more information on the Limits editor and SET.LIMIT, see "Limits editor" on page 709.

Editing multiple items

Some editors allow you to edit multiple items at the same time. For example, you can edit all the transfer lines in the model, within a single editor. This type of editor can be especially useful if you have many similar edits to make over a range of facilities.

Multiple item editors also provide an Insert button, which adds a new item to the model, and likewise to the editor. For example, in the transfer lines editor, you can insert a new transfer line into the model. After insertion, you will probably need to check the schematic and move the new element to the appropriate location.

To access a multiple items editor

In the model explorer, double-click the lowest-level folder that contains the items you wish to edit. For example, if you wish to edit all transfer lines, double-click on the Transfer Lines folder icon.

Other special editors

Some Model Builder editors operate in a unique fashion to accommodate non-tabular types of data. These editors are described in this section.

Curve Selection editor

In certain element editors, such as the pump editor, you can select curves for the device by name in the Curve Selection editor. In this editor, you can select the curve name from the list of available curves, and specify speed and multiplier values.

For a curve to appear in the list, it must first be defined using the general Curves editor. For more information, see "Curves editor" on page 713.

Poke/Ramp editor

Poke/ramp settings can be set for transfer lines and other hydraulic elements. For more information on poke/ramp settings and values, see "POKE and RAMP" on page 423.

Pipe Distance editor

The Pipe Distance editor allows you to enter elevation and pressure limit information for a transfer line, according to distance intervals along the line. The information specified in this editor is identical to that which can be specified in the INPREP command for transfer lines, except the editor provides a more intuitive table format and a visual graph representing your entries. For more information on these entries, as they relate to transfer lines, see "Transfer line - transient (T)" on page 263.

Workbook mode

Like most windows, you can resize and move Model Builder editors with standard Windows controls. You can have multiple editors open at once.

To assist with navigation among numerous editors, Model Builder provides a workbook mode that shows tabs at the bottom of the schematic view area. Each open window, such as a schematic view or an editor, has an associated tab. When the tab is clicked, that particular item is brought to the front.

To enable workbook mode

Select **View** > **Workbook Mode**.

Customizing the schematic

Model Builder provides a visual representation of your model, called the schematic. It also provides a variety of tools to help you manage the appearance of the schematic. While using the schematic, keep in mind that it is for visual reference only. The size, scale, and location of symbols does not affect the model hydraulics.

Schematic properties

You can set a variety of properties that control how the schematic appears and the amount of control that you have over editing tools.

To set schematic properties

Right-click on the schematic view and select **Default Properties**.

The **Default Properties** dialog box contains the following tabs:

Tab	Function
Edit	Controls the access that you have over editing tools, such as enabling/disabling the ability to move items. Generally, you should use these settings to prevent accidental alteration of a schematic by yourself or another user.
Labels	Controls the orientation of labels with respect to their associated schematic items
Line	Controls the appearance of connecting lines, such as pattern and thickness.
Fill	Controls the pattern and color of the schematic background.
Font	Affects the font size of text objects, inserted by selecting Draw > Shapes > Text.

Note: Customized schematic properties are not saved when you close Model Builder.

Setting grid properties

The schematic can display a grid in the background to assist you with object alignment. The grid can be shown or hidden, and its appearance can be customized.

To customize the grid

Tip: You may turn the grid on or off, enable to snap-to-grid, or angle snap properties quickly by right-clicking in the schematic or clicking the toolbar buttons.

Right-click on the schematic view and select **Grid Properties**.

In the **Grid Properties** dialog box, you can set:

- Grid visibility
- Snap properties
- Grid color
- Grid spacing

Viewing data on the schematic

You can display element and/or node names as well as selected attributes on the schematic. You can display a different set of attributes for each element type. Furthermore, you can customize the font size, color, etc. for the names and attributes.

To view node or element names on the schematic

From the Model Builder window, select **View** > **Element Names** or **View** > **Node Names**.

To set up attributes for viewing on the schematic

1 From the Model Builder window, select **View** > **Select Attributes**.

2 In the **Schematic Attributes** editor, click the … button next to a device for which you want to display attributes.

Tip: You may use the * character as a wildcard. For example, if you wanted to view all peeks beginning with S for a particular element, you would enter S*.

3 In the dialog box, click on a attributes you want to display and then click **Add**. You may repeat this step to add additional attributes.

4 Repeat step 2 and step 3 to select attributes for other element types.

5 After selecting the attributes, you need to show them. See "To view attributes on the schematic" on page 727.

To view attributes on the schematic

From the Model Builder window, select **View** > **Show Attributes**.

To customize the font on the schematic

From the Model Builder window, select **View** > **Font**.

Marking up the schematic

In the Model Builder schematic, you can add lines, shapes, pictures, and text. These markings are saved in a Model Builder Graphics (MBG) file when you save the model.

Note: If you want another user to see the same markups, you must be sure to make the MBG file available with the model files in the same folder.

To add a shape

In the Model Builder window, select **Draw > Shapes** and then select the shape you want to add.

Shape	Process
Line	Click in the schematic at the beginning of the line and drag to the end of the line. Release to complete the line.
Polyline	Click in the schematic at the beginning of the polyline. Click again at the vertex (bend) in the line. Continue clicking at other vertices. Double-click to end the polyline.

Shape	Process
Polygon	Click in the schematic to begin the polygon. Continue clicking at each vertex. Double-click to complete the polygon.
Rectangle	Click in the schematic at a corner of the rectangle. Drag to the opposite corner of the rectangle and release.
Polycurve	Click in the schematic to begin the polycurve. Continue clicking to give shape to the curve. Model Builder smooths the curve between the points. Double-click to end the polycurve.
Closed Curve	Click in the schematic to begin the curve. Continue clicking to make the shape of the curve. Model Builder smooths the curve between the points. Double-click to close the curve.
Ellipse	Click in the schematic and drag to the desired shape of the ellipse. Release to complete the ellipse.
Port	Click in the schematic to insert a port symbol. 

To move a shape

In the schematic, click on the shape and drag it to the new location. For information on moving elements, see "To move an element" on page 711.

To move vertices on a shape

Note: Moving vertices on an element may change the connectivity. See "To move an element" on page 711.

1. In the schematic, click on an existing shape so that the vertices display.
2. Place your cursor over a vertex. When the cursor changes, click on the vertex and drag it to a new location.

To edit properties for a shape

1. In the schematic, right-click on a shape and select **Properties**.
2. Click on the tabs in the Component Properties dialog box to set properties for the shape. The following table summarizes the settings that are available on each tab. Some of the tabs listed below may not be available for some shapes.

Tab	Available Settings
General	Specifies the name of the shape and the shape type. To change the shape's name, type a new name in the Name text box.
Edit	Specifies which edit actions can be performed on a shape. Available options include selecting the shape, moving and rotating it, editing its vertices, and performing scaling, stretching, and containment actions. You can disable all editing of the shape by selecting the Read Only check box.
Line	Specifies the color, style, and width of any lines in the shape. Available style options include Solid, Dashed, Dotted, Dash-Dot, and Dash-Dot-Dot. You can also make the lines invisible by selecting the Transparent check box.

Tab	Available Settings
Fill	Specifies the fill color and pattern for the shape. Available patterns include a solid pattern and several lined and hatched designs. You can specify both foreground and background colors for the patterns, or you can make the foreground and/or background colors invisible by selecting the appropriate Transparent check box. **Note:** The Fill tab is not available for all shape types.
Position and Size	Provides read-only information the size of the shape plus the location of the shape in the schematic. For information on moving a shape, see "To move a shape" on page 728 and "To move vertices on a shape" on page 728.
Font	Specifies font type, size, color, and formatting. This tab is only available for the Text shape.
Text	Specifies text, text alignment, and whether the text is allowed to wrap in multiple lines in the schematic. This tab is only available for the Text shape.

3 Click **OK** to close the dialog box.

To delete a shape

In the schematic, select the shape and press **Delete**.

To insert text in the schematic

1 In the Model Builder window, select **Draw > Shapes > Text**.

—or—

Copy text from another software product and paste it onto the schematic.

2 Double-click the shape and then type the text that you want to appear in the schematic.

3 To change the font type, size, color, and other text settings, right-click the shape and select **Properties**. For more information on editing text properties, see "To edit properties for a shape" on page 729.

To insert an image in the schematic

In the Model Builder window, select **Draw > Shapes > Image**.

—or—

Copy an image from another software product and paste it onto the schematic.

Tip: Model Builder may have trouble pasting some images. As a workaround, insert the image directly using the menu or first copy and paste the image into another software product like Word or Paint and then copy and paste from there.

Positioning objects in the schematic

You can position objects, including both elements and shapes in the schematic using a variety of tools. Depending on whether you are working with a shape, an element, or multiple objects, the changes you make to the schematic may or

may not be saved. For more information on whether changes are saved, see the procedure for the action you want to take.

To group objects

Note: Groups are useful for manipulating several objects at once; however, group information is not saved.

1. Press **Ctrl** and click on each of the objects you want to include in the group.
2. In the schematic, right-click and select **Grouping > Group**.

To rotate a single object

Notes: Rotation of an element symbol is a temporary action that is not saved. However, rotation angle of shapes is saved.

You may rotate groups of objects. For example, if you need to mirror the elements in a station, you could copy and paste the group and then rotate the group. For more information, see "To group objects" on page 730 and "Cutting, copying, and pasting" on page 715.

1. In the schematic, select an item you want to rotate.
2. From the Model Builder window, select **Draw > Rotate**.

To rotate multiple objects in a group

1. Group the objects you want to rotate. For more information, see "To group objects" on page 730.
2. From the Model Builder window, select **Draw > Rotate**.

To move an object

1. In the schematic, click on an object to select it.

   Tip: Press **Ctrl** and click on other objects if you want to move multiple objects.
2. From the Model Builder window, select **Draw > Nudge**.

   —or—

   Press the arrow keys.

To align objects

1. In the schematic, click on an object to select it.
2. Press **Ctrl** as you click on at least one more object.
3. From the Model Builder window, select **Draw > Align**.

To evenly space objects

1. In the schematic, click on an object to select it.
2. Press **Ctrl** as you click on at least two more objects.
3. From the Model Builder window, select **Draw > Spacing**.

To order objects

Note: Ordering objects allows you to move one object in front of another. This may be helpful for viewing and printing purposes, but ordering is *not* saved.

1 In the schematic, select an object that overlaps another object.

2 From the Model Builder window, select **Draw > Order**.

Exporting and importing model data

Model Builder allows you to export data from any device in your model to CSV files, and also to import data from CSV files back into your model. This data exchange capability allows you to easily move data in and out of your model for any number of different tasks, including the following:

- Export key data from your model for use with other software products.
- Import data from external sources into your model.
- Update model data using Excel or some other external software product, and then import the updated data back into your model.

Refer to the topics listed below for more information.

- "Exporting model data" on page 731
- "Importing model data" on page 732

Exporting model data

Perform the following procedure to export model data from your Model Builder file to CSV file format.

To export model data

1 From the main menu, select **File > Import/Export > Export**.

2 Use the Export dialog box to select which device types, devices, and attributes you want to write to the CSV file. Each category – Types, Devices, and Attributes – has its own tab in the dialog box, and each tab includes columns for Included items and Excluded items. Items in the Included columns will be exported to the CSV file, while items in the Excluded columns will not be exported.

- To move items from one column to another, select the item and click the **<-** and **->** buttons, as appropriate.
- To move all items from one column to another, click the **<<<** and **>>>** buttons, as appropriate.

In general, use the following process to select the data that will be exported to the CSV files:

a Use the **Types** tab to select which device types (such as general pipes, nodes, and regulators) you want to export. Each included device type will be exported to a separate CSV file.

b Use the **Devices** tab to select specific devices that you want to export data for. You must first select a device type under **Show devices for**, and then set up specific device names in the Included and Excluded columns, as appropriate. Repeat this step to select devices for each device type that you included for export on the Types tab.

c Use the **Attributes** tab to select specific attributes that you want to import for each included device. You must first select a device type under **Show attributes for**, and then set up specific attribute names in the Included and Excluded columns, as appropriate. Repeat this step to select attributes for each device type that you included for export on the Types tab.

3 Click **OK**.

4 Use the Export dialog box to select the folder where you want to export the data.

5 Click **OK**.

Model Builder exports the data to the specified directory. A separate CSV file will be created for each device type that you selected to export.

Importing model data

Perform the following procedure to import data from CSV files into your model.

To import model data

1 From the main menu, select **File > Import/Export > Import**.

2 In the Import dialog box, select the name of the folder that contains the CSV files that you want to import.

3 Click **OK**.

4 Use the Import dialog box to select which device types, devices, and attributes you want to import to your Model Builder file. Model Builder performs a quick scan of the designated CSV files to determine which types, devices, and attributes to include in the dialog box.

Each category – Types, Devices, and Attributes – has its own tab in the dialog box, and each tab includes columns for Included items and Excluded items. Items in the Included columns will be imported into your model, while items in the Excluded columns will not be imported.

- To move items from one column to another, select the item and click the **<-** and **->** buttons, as appropriate.
- To move all items from one column to another, click the **<<<** and **>>>** buttons, as appropriate.

In general, use the following process to select the data that will be imported into your model:

a Use the **Types** tab to select which device types (such as general pipes, nodes, and regulators) you want to import.

b Use the **Devices** tab to select specific devices that you want to import data for. You must first select a device type under **Show devices for**, and then set up specific device names in the Included and Excluded columns, as appropriate. Repeat this step to select devices for each device type that you included for import on the Types tab.

c Use the **Attributes** tab to select specific attributes that you want to import for each included device. You must first select a device type under **Show attributes for**, and then set up specific attribute names in the Included and Excluded columns, as appropriate. Repeat this step to select attributes for each device type that you included for import on the Types tab.

5 Click **OK**.

Running an SPS module

You can run PREPR, TRANS, and TPORT directly from Model Builder.

Validating the model from Model Builder

After making changes to a model, you should validate the model. Validation checks the basic integrity of the model and can help avoid certain problems associated with connectivity and invalid settings, before you attempt to run PREPR. Validation results are printed to the Output window.

To validate the model from Model Builder

Select **Tools** > **Modules** > **Validate**.

Note: Validation automatically saves your model. Be sure that you have no unwanted changes before validating.

Note: The Output window displays the results of the validation. This information is also automatically written to a text file found in the same directory as the model. The Output window displays the name of this file, and you can use WordPad to open and view it at a later time. Each validation overwrites the previous validation output file.

Running PREPR from Model Builder

When you run PREPR from Model Builder, it attempts to create a RESTRT file from the currently open model, or the currently selected model if multiple models are open. The functionality is identical to running PREPR from the normal SPS startup window. After you successfully run PREPR from Model Builder, you can run TRANS on the resulting RESTRT file, either with Model Builder again, or using the normal SPS startup window. For more information on PREPR functionality, including PREPR reporting, see “Overview of PREPR” on page 53.

When you run PREPR from Model Builder, it produces an HTML-based report that opens in the Model Builder window. This report contains important information the PREPR run, such as messages and warnings. If you are an experienced SPS user, you may recognize this report as a replacement for the former OUTPRP file.

To run PREPR from Model Builder

Note: PREPR requires an INPREP file, which is generated automatically if the current model is in MB format. If it is already in INPREP format, it will be saved automatically before the PREPR run. For more information on the differences between MB and INPREP files, see “Saving model data from Model Builder” on page 735.

Select **Tools** > **Modules** > **Prepr**.

After running PREPR, a summary report displays in the Output window and a detailed report, labeled “SPS/prepr,” displays in a new window within Model Builder. This detailed report is the same OUTPRP report that is generated when you run PREPR from the SPS startup window. For more information on this report, see “HTML OUTPRP reports” on page 53.

Running TRANS from Model Builder

When you run TRANS from Model Builder, it launches a simulation from a RESTRT file, just as the traditional SPS startup window.

Because TRANS requires a RESTRT file produced by PREPR, you must run PREPR at some point to get a RESTRT file. You have several options for doing this, including:

- Manually running PREPR from Model Builder to create the RESTRT file on the currently open model. For more information, see "Running PREPR from Model Builder" on page 733.
- Allowing Model Builder to automatically run PREPR before TRANS, as a prerequisite. For more information, see "Setting module dependencies in Model Builder" on page 702.
- Running PREPR on an INPREP file that is not currently open in Model Builder. For more information, see "To run PREPR from the SPS startup window" on page 84.

Note: TRANS also requires an INTRAN file to define the simulation. You must create this file manually outside of Model Builder. For more information on INTRAN files, see "The INTRAN File" on page 425.

For more information, see "Overview of TRANS" on page 55.

To run TRANS from Model Builder

Select **Tools** > **Modules** > **Trans**.

Tip: If you have a specified a case for the model, TRANS will use the RESTRT file specified there. Otherwise, TRANS will look for a RESTRT file with the same root name as the associated INPREP file, within the same directory. For more information on cases and file locations, see "Setting up SPS case files" on page 703.

Accessing a Show window from Model Builder

During a TRANS run, you may access a Show window directly from the item on the schematic or from the model explorer.

To access a Show window from Model Builder

1. Run TRANS or TPORT. See "Running TRANS from Model Builder" on page 734 or "Running TPORT from Model Builder" on page 735.
2. In the schematic window, right-click on the device and select **Show Runtime**.

   —or—

   In the model explorer, right-click on the device and select **Show Runtime**.

Running TPORT from Model Builder

You can run TPORT from Model Builder. For more information on TPORT, see "Overview of TPORT" on page 57.

To run TPORT from Model Builder

Select **Tools** > **Modules** > **Tport**.

Saving model data from Model Builder

When managing files in Model Builder, you need to consider several issues to determine the best practices for saving your work. You may save in one of two different file formats: INPREP or MB. The format that you choose depends on how you use Model Builder and SPS.

- If you use Model Builder for all of your modeling and running PREPR, you can use the MB format exclusively and take advantage of faster loading and saving times.
- If you have a need to edit the files directly, you want to maintain include files and/or sequence of INPREP data, or you prefer to use the traditional SPS startup window to run PREPR, you must save your models in INPREP format.

If a model is incomplete, or the Validate module produces errors, Model Builder may not be able to export an INPREP file. In this situation, you should use the MB format until the model is complete. However, it is recommended that you maintain INPREP archives of the model as soon as possible to help avoid MB file compatibility issues between Model Builder releases.

Model Builder automatically maintains INPREP backups of your models, even if you use MB format exclusively to save your models. For more information, see “Setting backup preferences” on page 701.

To save a model in Model Builder

Note: If you are maintaining include files, see “Maintaining Include files” on page 736.

1. Select **File > Save** or **File > Save As**.
2. In the Save As dialog box, next to **File**, browse to the location where you want to save the file. Make sure the file extension is MB or INPREP, as applicable.
3. If you are saving in INPREP format, under **Include File Options (INPREP only)**, choose the most appropriate option. For more information, see “Maintaining Include files” on page 736.
4. Click **OK**.

   Note: If you used drawing tools to mark up the schematic, an additional MBG file will be saved with the model. If you want to view the markups the next time you open the model, this file must be present in the same folder as the model.

Maintaining Include files

Include files provide a convenient way to share configuration data among several models. Additionally, if you have a large volume of data, you may prefer to manage your data in smaller files. If you have managed your model data this way in conjunction with INPREP files, you may continue to do so through Model Builder. On the other hand, if you no longer wish to maintain include files, Model Builder offers a command that incorporates all include files into the main model file to produce a “flat” file.

For more information on specifying include files directly from the INPREP file, see “INCLUDE” on page 573.

Preserving or overwriting include files

If you want to maintain include files, you need to decide whether you want to preserve the data in each of the include files or whether you want changes made in Model Builder to be transferred to the include file when you save the model.

If you are using include files to share configuration data among several models, you may elect to not overwrite the files from Model Builder, so inadvertent changes won't be saved. You should also consider making the files read only (using Windows features).

In order to manage include files, you need to:

1 Verify that the include file will be preserved or overwritten as desired by setting the OVERWRITE=YES|NO option. For more information, see "To edit the overwrite status for an include file" on page 736.

2 Assign elements to include files in the element editor and specify a line number for the element, as desired. For more information, see "To assign an include file for INPREP data" on page 737 and "Preserving INPREP sequence using line numbers" on page 738.

3 Select the appropriate option when saving. For more information, see "To overwrite selected include files when saving" on page 737 or "To overwrite all include files when saving" on page 737.

To edit the overwrite status for an include file

1 In the model explorer, expand **Other**.

2 Under Include files, double-click the include file for which you want to change the status.

—or—

If you are creating a new include file, right-click on **Include Files** and select **Insert**.

3 In the include file editor, change the **OVERWRITE** field to **YES** or **NO**.

4 Click **OK**.

Note: A file generated from Model Builder will be set to YES by default while a manually created file will be set to NO. Model Builder detects whether a file was manually created or generated through Model Builder based on the comment on the second line. A file generated in Model Builder has the following comment in the second line. `/*{sps} Include file generated ...` Note that there is no other syntax for this setting in the INCLUDE file and deleting this comment through a text editor could change the status.

To assign an include file for INPREP data

1 In a model builder editor, click on the **Common** tab.

2 In the FROM FILE field, type the name and folder path of the include file you want to reference.

Tips: The FROM FILE field may be populated by any file name, but the common convention is to use INC file extension.

If you do not specify a file path, Model Builder assumes the Include file resides in the same folder as the model. However, you may also specify a relative path or full path.

3 Click **OK**.

To overwrite selected include files when saving

Note: Only include files with OVERWITE=NO specified will be overwritten. This overwrite status is initially set based on whether the file looks like it was written by model builder. Specifically, if the second line of the file matches `/*{sps} Include file generated ...`, it is assumed to have been written by Model Builder and is set to YES. Otherwise, it is set to NO and must be manually changed.

1 After you have made changes to the model through Model Builder, select **File > Save As**.
2 In the **Save As** dialog box, next to **File**, make sure the file type has an INPREP extension.
3 Under **Include File Options (INPREP only)**, choose **Save INPREP file and include files**.
4 Also choose **Save only those include files with OVERWRITE=YES**.
5 Click **OK**.

To overwrite all include files when saving

Note: This procedure will result in all of your include files being overwritten, regardless of whether you have manually set OVERWRITE+NO in the include file. For you do not want overwrite all include files, see "To overwrite selected include files when saving" on page 737.

1 After you have made changes to the model through Model Builder, select **File > Save As**.
2 In the **Save As** dialog box, next to **File**, make sure the file type has an INPREP extension.
3 Under **Include File Options (INPREP only)**, choose **Save INPREP file and include files**.
4 Also choose **Save all include files**.
5 Click **OK**.

Incorporating include file data into the main model

If you no longer want to maintain separate include files, you may use the command in Model Builder to incorporate all data into a single model file. This change is permanent, so you may want to back up your model files before performing this action. After you have flattened the model, you will need to save the model in whichever format you prefer.

To flatten include files

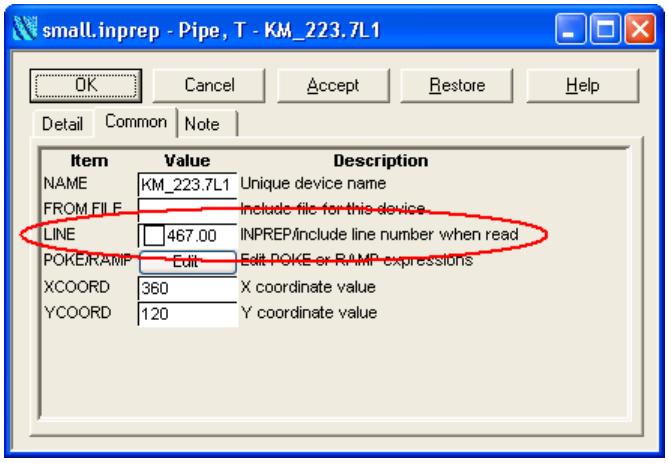
Note: This change is permanent, so you may want to back up your model files before continuing.

1 From the Model Builder window, select **Edit > Flatten Model**.
2 Click **OK** in the message box.
3 Select **File > Save** or **Save As**.

Preserving INPREP sequence using line numbers

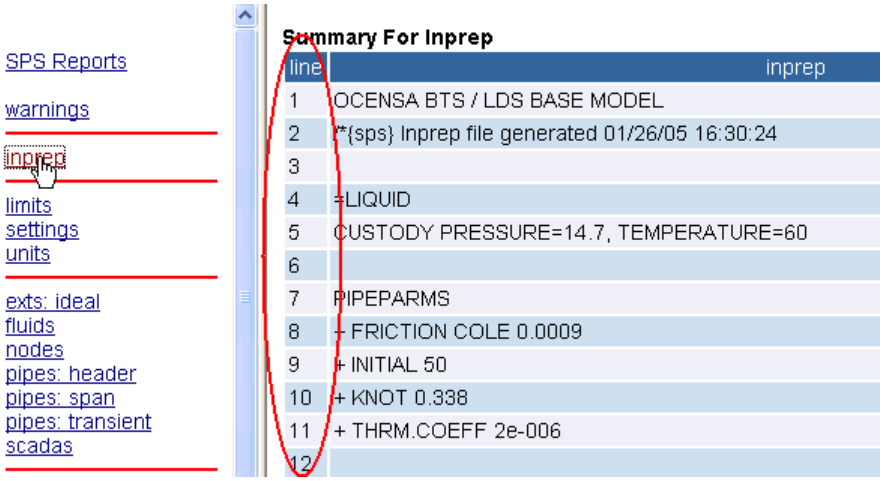
When importing INPREP files, Model Builder keeps track of the INPREP line numbers, which enables you to retain your preferred organization of your INPREP file and insert devices within that structure as you continue to build and maintain your model.

You may view and edit the line number for each element through the various device editors.



In any element editor, the LINE field on the Common tab preserves INPREP sequence.

In addition, after running the validation and PREPR you can view the line numbers in the **SPS Reports** window. On the left side of this window, select **Inprep**. **Summary for Inprep** will appear, displaying the attributes of the current model and their corresponding line numbers.



In the OUTPRP report, click in inprep to reference the line number.

For more information on running a validation see "Validating the model from Model Builder" on page 733. For more information on running PREPR see "Running PREPR from Model Builder" on page 733.

Notes: Although sequence is preserved, line numbers may change from when the INPREP file is imported to when it is exported. For example, if a valve has line 100, and you add/delete/modify anything above the valve, then the valve will still have a line attribute of 100 during that session. However, when exported the valve may have a higher line number. This will become apparent the next time the file is imported.

You can also insert data at a desired location in the text file via Model Builder by using decimal places. For example, if you want to insert a element between line 100 and 101, you would enter 100.5).

Blank lines in the INPREP document will count as numbered lines.

If the data is stored in an include file, the number reflects the line number in the include file rather than the INPREP file.

Finding conversion factors

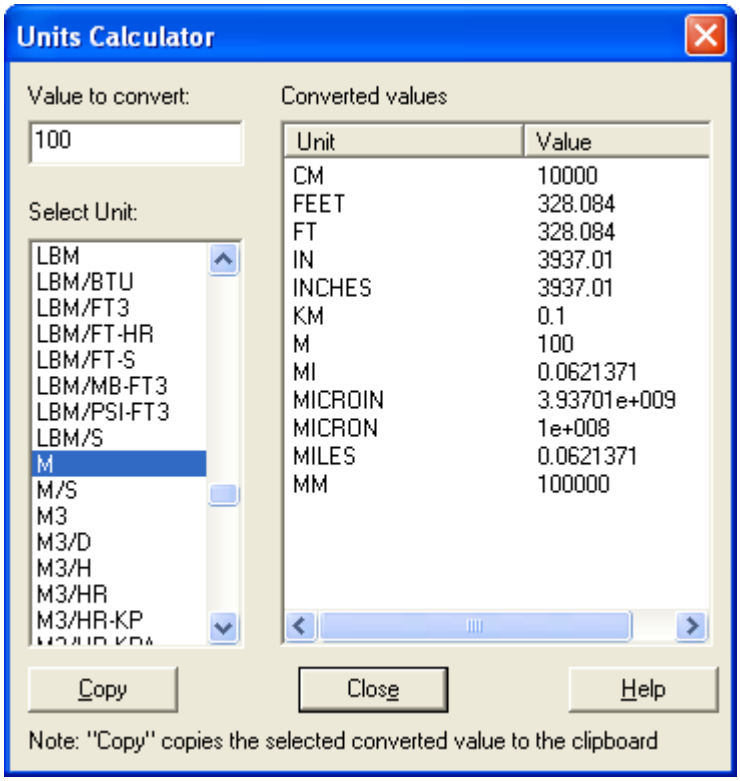
Model Builder provides quick access to commonly-used utilities from either the Tools menu or by toolbars. While most utilities are available through Windows, the Units Calculator is a utility available only with Model Builder.

To find conversion factors in Model Builder

1. From the Model Builder window, select **Tools > Utilities > Units Calculator**.
2. In the **Units Calculator**, enter a number under **Value to convert**.
3. Under **Select Unit**, select the current unit for which you want to obtain conversions.

   Under **Converted values**, Model Builder provides all of the applicable conversions for the value and unit you selected.

   For example, the following figure shows how to obtain relevant conversions from 100 meters to other distance units.



Units Calculator in Model Builder

CHAPTER

17

Statefinder and Leakfinder

The On-line suite of products—Statefinder, Leakfinder, and Predictor—are built upon SPS technology and are used as aids by pipeline operators in performing their daily tasks. Unlike SPS, Statefinder and Leakfinder simulate the behavior of a pipeline using real data from a SCADA system. The Predictor then predicts the future state of the pipeline by extending the current state of the pipeline, obtained from Statefinder, forward in time.

This section of the *SPS Help and Reference* provides the necessary information to build models and run online simulations. You may need to see "Interfacing SPS to External Applications" on page 925, which describes the function calls necessary for creating interfaces between the SPS family of products and other software.

On-line products

Statefinder, Predictor, and Leakfinder are based on SPS. The input structure and the model interaction are the same as SPS. Statefinder and Leakfinder have additional elements that allow SCADA information to drive the model.

Statefinder

Using real-time measurement data from a SCADA system, Statefinder maintains a constantly available picture of the current operation of the pipeline. It provides detailed information the ongoing operation of the pipeline even when operating with degraded or limited SCADA data.

Extending the SCADA system

By providing constantly available distance plots of pressure, flow, density and other parameters along the pipeline, Statefinder acts as an extension to the SCADA system. Indeed, these variables can be passed back to the SCADA system just as if they came from installed field equipment. This enables the operators to observe system performance even without installed telemetry, thus minimizing investment in telemetry equipment.

For example, for gas models, while in-line station flows are commonly unavailable in the control room, they are continuously calculated by Statefinder and can be routinely passed to the SCADA system for presentation in the control room. For petroleum and product pipelines, Statefinder maintains a constantly available view of the location of all batch interfaces, along with their estimated time of arrival.

Operational surveillance

Statefinder automatically adjusts and calibrates incoming SCADA data to compensate for errors associated with meter drift or changing pipeline roughness. Alarms can be configured to alert such problems as an instrument reaching its maximum calibration limit, thus assisting in determining maintenance schedules.

Statefinder also acts as an operational surveillance aid through its constant scrutiny of the pipeline for maximum and lowest allowable operating pressure (MAOP and LAOP) violations. Again, alarms can alert the control room staff when problems are detected.

Leakfinder

Today's climate of environmental and safety regulation, together with the economic consequences of product loss and cleanup, and facility downtime and repair, has focused the attention of pipeline operators on effective means of detecting leaks. Leakfinder is a very effective tool for pipeline operations surveillance and leak detection.

Leakfinder combines Statefinder with the Leakanalyzer to form an effective leak detection system. Statefinder uses real-time SCADA data to track pipeline hydraulics and to signal measurement anomalies. The Leakanalyzer scrutinizes those anomalies looking for possible leak signatures. When a leak is detected, Leakfinder raises an alarm that shows the leak location, the onset time, the leak rate, and the total volume released. Leakfinder also assigns a confidence level to the leak.

Statefinder can operate with degraded or limited SCADA measurement data. When this occurs, as when a liquid pipeline runs slack, the Leakanalyzer automatically adjusts its leak detection thresholds (LDTs) to maintain the best possible system performance while avoiding false alarms. An LDT is a distance plot along the pipeline that represents the smallest leak that is not masked by the aggregate uncertainties in the measurement data, current operating conditions, fluid properties, elevation data, and many other factors. These LDTs are profiled graphically and provide a dynamic and constantly available measure of how well the pipeline is protected.

Predictor

Automatic Predictor

The Automatic Predictor runs "look-ahead" hydraulic simulations automatically at a specified frequency and duration.

The current results from Statefinder are used as the starting point for the Automatic Predictor's calculations. Using a list of timed events (flow and pressure changes) from a Load Forecaster (for weather dependent loads) or from a batch scheduling SPS, the Automatic Predictor steps forward in time to show how the network will behave. If you maintain a Control Schedule for planned set point or equipment changes, this schedule can be used for changes in the model.

Predicted alarm conditions are signaled so that preemptive action can be taken before the real-life conditions occur.

Planning Predictor

The Planning Predictor is used to run "what-if?" cases upon request. By comparing the economic and hydraulic results of two or more forecasts, you can select the better strategy for pipeline operation.

A trial Control Schedule (typically a modified version of the schedule used by the Automatic Predictor) is used as the operating strategy for the predictive simulation. Similarly, a modified weather forecast might be used to derive a different

forecast of model loads. Results from Statefinder or from the Automatic Predictor can be used as the starting point of the Planning Predictor.

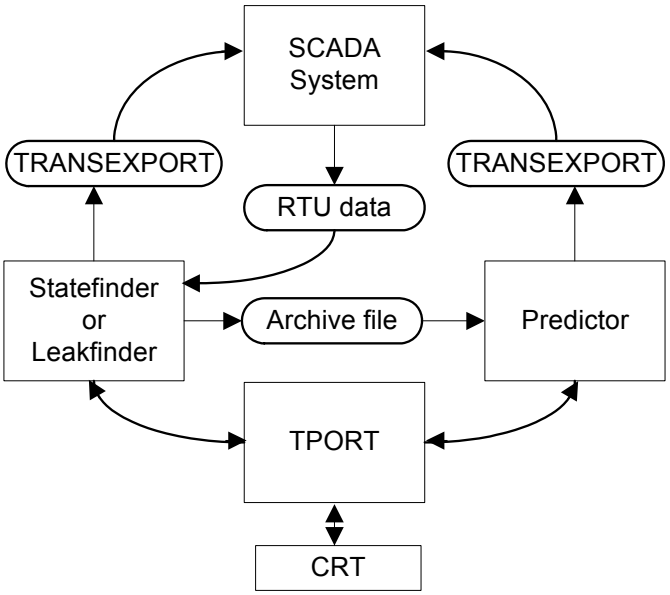
Features of Statefinder and Leakfinder

The following are common features of Statefinder and Leakfinder:

- *State Estimation.* State Estimation adjusts SCADA data and pipeline friction within user-defined limits to calculate an overall model state that is as consistent as possible with all the data.
- *Autocalibration.* Biases in the State Estimation results are fed back into the model so that the model improves over time.
- *Alarms.* You may develop alarms and categorize the alarms to aid in the detection of equipment outages, pressure violations, and other significant events.
- *Detection of equipment malfunctions.* The autocalibration process detects instrument outages, out-of-range calibration errors, fouled pipes, inefficient pump/compressor operation, valve malfunction, etc.
- *Archiving of the pipeline state.* You can schedule archives and/or perform archives on demand.
- *Data Estimation.* Statefinder estimates non-telemetered data and the pipeline state during data outages.
- *Batch tracking.* Statefinder tracks liquid batch movement through a piping model, and/or changes in gas composition and related qualities, and/or gas gravity.

Relationship of TRANS and the SCADA system

The following flow diagram shows the relationship between TRANS and the SCADA System for On-line Systems.



Online System Flow Diagram

Differences between off-line and online models

Because off-line and online models have a few basic differences, the online models require a few additional settings and elements from those in off-line models.

Changes to the INPREP file for online modeling

Model settings

You must enter the command "SELECT (INPREP)" on page 211 to run Statefinder and Leakfinder.

Additional elements

Online models use the following additional elements:

- *SCADA* elements drive model set points with telemetered data.
- *Monitors* impose telemetered pressures on the model.
- *Header Flow (HF)* elements can be used to augment and/or replace pumps, compressors, valves, and stations.
- *Flowmeter* elements model in-line flow meters rather than headers.

Liquid models use additional settings (such as Property, composition controller, and interface alignment) to adjust fluid properties and align batches.

Schematization

Online systems affect the schematization of the pipeline network. In general, HF elements replace stations. An entire station can be replaced with a single HF or individual units may each be replaced with an HF. The following are specific points schematizations:

- External flows are modeled using SCADA elements driving SALE/TAKE externals.
- Pressures are usually controlled through Monitors.
- All HF elements must have a Monitor somewhere upstream and somewhere downstream.
- Two HFs in series must have a Monitor in between.
- Two Monitors should not be connected to the same node.
- HFs should not be connected to only a flow-controlled external.

If a Predictor model is also to be run, detached elements are also modeled. The detached elements usually consist of two externals (these may be pressure controlled SALE/TAKE externals), the element(s) that the HF replaces in the online model, and applicable control systems.

Changes to the INTRAN file for online modeling

The INTRAN file must reference the RTUDATA file(s) where SCADA data is written. Also, the RTU logic may be duplicated in the INTRAN file if required. SCADA set points must be saved to drive elements in the Predictor.

Solution technique for online modeling

State Estimation

The three representative states of a pipeline system are

- *Process State.* The state of the dynamic fluid process as exhibited by the actual fluid(s) in the actual pipeline network.
- *SCADA State.* The state as reported by the SCADA System via discrete measurements of variables at discrete times and locations.
- *Model State.* The state calculated by Statefinder or Leakfinder

Statefinder and Leakfinder attempt to represent the Process State of the pipeline network by adjusting the SCADA State. The SCADA State, by its nature, does not represent the complete Process State of the pipeline network. The SCADA State is known only at discrete points and discrete times. In addition, there are instrumentation errors and limitations and the polling time of RTUs are different. Statefinder minimizes the effects of instrumentation errors using State Estimation so that the Model State is a close approximation of the Process State.

State Estimation interpolates SCADA values using the fundamental laws of fluid mechanics, meaningfully distributes errors among the data, provides a pipeline network state during partial data outages, and expands SCADA data to create a complete hydraulic state of the pipeline network.

The information that is gathered by SCADA in some ways *over-specifies* a pipeline fluid mechanics problem. To see this, consider a single gun-barrel pipeline with measured pressures and flows at each end. In such a pipeline, the supply flow and delivery pressure provided continuously in time completely specify the state of the pipeline, even under transient conditions. Thus, the inlet pressure measurement and the outlet flow measurement are redundant. Moreover, because of possible measurement errors, the redundant data may inconsistently specify the problem. This means that one or both of the redundant data values may not agree with the calculated state based on the original supply-flow-delivery-pressure conditions.

The information that is gathered by SCADA in other ways *under-specifies* a pipeline fluid mechanics problem. For example, SCADA systems do not generally report key pipeline data such as connectivity, pipe geometry, pipe efficiency, valve opening/closing characteristics, pump/compressor performance curves, or detailed fluid properties.

Plus, by their nature, SCADA systems report values at discrete times and not continuously. The exact details of what happens in the time between measurements are generally unknown.

One function of State Estimation is to make best use of redundant data. This is much like a chemist using regression to curve fit data taken at widely disparate conditions.

Each SCADA measurement has an associated error tolerance. In selecting the most representative value for each SCADA point, State Estimation adjusts each SCADA-reported value within its tolerance to achieve a Model State closest to the Process State. It has thereby used the redundancy to improve the closeness to the estimate of the Process State.

Among the most important adjusted SCADA measurements are pressures and flows. Because of errors in flow or density measurements and errors in user-defined pipe friction properties, State Estimation also makes adjustments in the pressure drop in each span of pipe as a correction called Pressure Drop Force (PDF). In addition to correction for

errors in friction, PDF also corrects for the effects of head errors caused by incorrect batch interface locations or by incorrect density in hilly terrain.

You have the ability to set the tolerances that State Estimation uses in adjusting pressures, flows, and PDFs. Some of the limits are set on the SCADA elements, including driving set points, individual fluids, or pipes pokes. Other limits are set on a global basis.

In any given set of data not all the values are equally reliable. For example, it might be known that one pressure measurement is more accurate than another. Also, errors in pressure and errors in flow must each be accounted for in the results in an appropriate way. Clearly, if the pressures are trusted and the flows are not, State Estimation should have a means of developing its estimate by putting much more weight upon pressure measurements than flow measurements, or vice versa.

To exercise such control, a set of "weights" can be entered that influences the way State Estimation pays attention to the input data. Qualitatively, paying attention to data or other specifications is done by assigning a penalty to *not* satisfying the specification, then finding the solution that minimizes the total penalty assigned.

The penalties are controlled by a set of weights referred to as JTS Weights. These weights are values that you may enter to penalize deviations from SCADA values in certain quantities. Increasing a weight, and thus the corresponding penalty, decreases the size of the corresponding deviation in the solution. It is important to note that increasing all weights by the same factor is equivalent to leaving the original weights alone. State Estimation cannot be forced to agree 100% with all inputs. The values of the weights and their corresponding penalties are available as attributes of the JTS element via the "SHOW JTS" interactive command.

The following defines each of the JTS weights.

Weight	Description
DF_WT	Penalizes diagnostic flows. Formerly jtswt(4).
P_ADJ_WT	Penalizes deviation between model pressures and corresponding SCADA pressures. Formerly jtswt(1).
P_RATE_WT	Penalizes the rate of change of model pressure from time step to time step. Formerly jtswt(2).
Q_ADJ_WT	Penalizes deviation between model flows and corresponding SCADA flows including: SCADA driven HFs without equations SCADA driven flow meters Formerly jtswt(3).
Q_ERR_WT	Penalizes pressure differences that do not match frictional drop for flow meters. Range of 0.01 – 10. Formerly jtswt(8).
FPDC_WT	Penalizes the magnitude of the frictional component of PDF. Formerly jtswt(7).
FPDC_RATE_WT	Penalizes rate of change of the frictional component of PDF from time step to time step. Formerly jtswt(5).
GPDC_WT	Penalizes the magnitude of the gravitational component of PDF. Formerly jtswt(10).

Weight	Description
GPDC_RATE_WT	Penalizes rate of change of the gravitational component of PDF from time step to time step. Formerly jtswt(9).
BMC_WT	Penalizes bulk modulus error correction. Formerly jtswt(6).

The equations for the corresponding penalties are:

$$DF = \sum_i \left[\frac{DF\_WT}{1.2} \times DF_i \right]^2$$

$$P\_ADJ = \sum_i \left[\frac{P\_ADJ\_WT}{EREP_i} \times \left(PV_i - PM_i \right) \right]^2$$

$$P\_RATE = \sum_i \left[\frac{P\_RATE\_WT}{1000 \times dt} \times \left(PM_i - PM_{i,prev} \right) \right]^2$$

$$Q\_ADJ = \sum_i \left[\frac{Q\_ADJ\_WT}{EREP_i} \times \left(QV_i - QM_i \right) \right]^2$$

$$Q\_ERR = \sum_i \left[Q\_ERR\_WT \times \left(P_i^- - P_i^+ - \frac{k_i Q_i \mid Q_i \mid}{\rho_i} \right) \right]^2$$

$$FPDC = \sum_i \left[FPDC\_WT \times fpdc_i \right]^2$$

$$FPDC\_RATE = \sum_i \left[FPDC\_RATE\_WT \times \left(fpdc - fpdc_{prev} \right) \right]^2$$

$$GPDC = \sum_i \left[GPDC\_WT \times gpdc_i \right]^2$$

$$GPDC\_RATE = \sum_i \left[GPDC\_RATE\_WT \times \left(gpdc_i - gpdc_{i,prev} \right) \right]^2$$

$$BMC = \sum_i \left[BMC\_WT \times \frac{BMC_i}{BMC_{i,allowed}} \right]^2$$

where:

EREP	=	Effective repeatability for the SCADA element
PV	=	Measured pressure, psig
PM	=	Pressure used in the model, psig
QV	=	Measured flow, user units
QM	=	Flow used in the model, user units
DF	=	Diagnostic flow at pressure monitors, standard MB/D
prev	=	Value at the previous time step
fpdc	=	Frictional component of PDF, dimensionless
gpdc	=	Gravitational component of PDF, dimensionless

$BMC_{allowed}$	=	Maximum allowed change in bulk modulus per time step
BMC	=	Bulk modulus correction factor
PDF	=	Pressure drop force (see "Pressure drop forces (PDFs)" on page 748)
P^-	=	Upstream pressure of flow meter, psig
P^+	=	Downstream pressure of flow meter, psig
k	=	Frictional pressure drop constant for flow meter
Q	=	Flow through flow meter.
ρ	=	Fluid density in flow meter

State Estimation minimizes, subject to constraints, the sum of the penalties plus the sum of the equations from HF elements with user-defined equations. Occasionally, several of the individual penalty values may become large. The largest penalty values take on the most significance. Inconsistencies may result in other areas of the model, which are manifest by values being pushed to their constraint limits.

As a cautionary note, flow controlled externals, which are driven by SCADA elements that violate a user-defined PMIN, PMAX, QMIN, or QMAX and therefore switch control mode, can result in unexpected results. Therefore, QMIN and QMAX should not be set at the external; they should be set with OMIN and OMAX of the SCADA element. Also, avoid setting PMIN and PMAX unless not doing so could result in model inaccuracies.

Pressure drop forces (PDFs)

PDFs are modeling forces that increase or decrease the pressure difference across a span. A span is a series connection of pipes, headers and valves without tees or parallel lines; it may be a single pipe. Spans are automatically created based on the connectivity in the model. State Estimation uses PDFs as means of accommodating data errors. PDFs are in user units of pressure and are calculated on a per span basis.

Usually, a difference between the computed and measured pressure exists at every measurement. Many possible sources can contribute to this discrepancy. In liquid batched systems, the largest difference is usually the gravitational effect associated with the misalignment of batch interfaces interacting with hilly terrain. Even in single fluid systems, density errors still have gravitational effects. Density errors can be introduced by poor or missing instrumentation or by equation of state inexactness. Errors in the computation of frictional pressure drop have several sources, not the least of which are uncertainties the pipe itself. For example, the pipe is not really round, wall thickness varies along pipes, roughness varies because of welding slag and pitting, and length measurements may contain errors. Uncertainties in the density of the fluid or the pipe roughness and errors in the friction factor correlation all contribute to the uncertainties. Measurement errors and changes in values between sampling times also contribute to the difference between the pressure changes across pipes in the field and the calculated pressure change.

PDF is one mechanism that State Estimation uses to reconcile the differences between the measured and the computed pressure drops. You influence the development of PDF by setting the weights for FPDC, FPDC_RATE, GPDC, and GPDC_RATE. The PDF for a span is the sum of the PDFs for the individual pipes in the span:

$$\text{PDF (span)} = \text{PDF (pipe}_1\text{)} + \ldots + \text{PDF (pipe}_n\text{)}$$

The PDF in pipe i in span j is given by:

$$\text{PDF (pipe}_1\text{)} = \text{FPDC}_j \bullet \text{FPD}_i + \text{GPDC}_j \bullet \text{GPD}_i$$

where:

$$FPD_i = \text{frictional pressure drop in pipe}_i$$

$$GPD_i = SpanGPD_j \times \left[\frac{\text{length } (Pipe_i)}{\text{length } (Span_j)} \right]$$

$$SpanGPD_j = 0.1 \bullet D0 \bullet MaxElevChange\ (Span_j) \bullet factor$$

All pipes within a span will have the same friction calibration value. Values for FPDC and GPDC are determined each time step so as to yield a best agreement with data consistent with known error magnitudes.

Both the gravitational and frictional parts of PDF are subject to bounds on the rate of change; the recent pressure drop history determines these bounds.

The bound for the FPDC at any time is no more than 15% of the frictional pressure drop in the pipeline augmented by viscosity error effects. This 15% bound is not alterable by data entry, but viscosity error is a user-defined parameter.

The constraints on the GPDC, the span gravitational control variable, are determined by estimating the uncertainty of the pressure drop due to gravitational effects. The uncertainty of the gravitational part of each pipe is:

$$\max (PipeDeltaRho, DRHO.MIN) \bullet PipeElevError + PipeDeltaElev \bullet PipeRhoError$$

where:

PipeDeltaRho	=	Maximum density - minimum density in the pipe
DRHO.MIN	=	Global minimum density error (tuning parameter)
PipeElevError	=	Maximum vertical length of batch misalignment
PipeDeltaElev	=	Maximum elevation - minimum elevation of the pipe
PipeRhoError	=	Average DENSITY.ERROR in the pipe

The DRHO.MIN can be changed using the “show online” display. The PipeElevError (vertical error in batch interface location) is user-definable for each pipe in INPREP. PipeRhoError is based on the fluids in the pipe, an estimate of the accuracies of the densitometers, and SCADA availability at the time the fluids entered the pipeline network.

Density error

The density error within a pipe can be determined in the following ways:

- When the pipe is initialized from PREPR, the DENSITY.ERROR for all FMVs is initially:

  $$MAX\ (base\ density) - MIN\ (base\ density) + MAX\ (DENSITY.ERROR)$$

- Because the density is uncertain (first fluid assumed), the worst case density error is assumed.
- When the pipe is initialized with a LINE.FILL, DENSITY.ERROR is set to the user-defined value for each fluid.
- When the pipe is initialized with a restart, LOAD.INPUT, or LOAD.STATUS, DENSITY.ERROR is set to the value saved with each FMV.
- When density is controlled by a composition controller, DENSITY.ERROR is set to the effective repeatability of the SCADA element driving the composition controller. If the SCADA element is out, the density and DENSITY.ERROR defaults to the user-defined values for the fluid.

Diagnostic flows (DFs)

Every SCADA point that is configured to drive the model has a user-defined parameter called repeatability, REP. The repeatability is a quantitative statement of the maximum deviation that is reasonably expected between the process value and the SCADA measurement. REP is usually somewhat higher than the manufacturer's specification because of A/D conversion error and SCADA processing. A measurement that is late because of a missed polling cycle, or a measurement that is known to be unreliable because of equipment malfunction may have a much larger effective repeatability than that stated in its original definition. The effective repeatability, EREP, is adjusted for a number of conditions that could lead to data degradation.

The flow through the monitor is given by the following equation:

$$Q = LM.WT \bullet (P - SP)$$

where:

Q	=	monitor flow
LM.WT	=	weight for minimizing monitor flow
P	=	pressure at the monitor
SP	=	pressure set point for the monitor

Also JTS:DF_WT and/or JTS:P_ADJ_WT can be adjusted to restrict monitor flow.

Node capacitance

Nodes have a small capacitance (volume) associated with them to minimize solution problems when multiple steady-state elements are connected in series. This is a problem with HF elements because each element typically tries to achieve a given flow rate. If the flows are not exactly the same through the HFs, the intervening point must have the capacitance to absorb the mismatch in flow. Without a small capacitance, pressures would become unstable and either go to a very low number (out flow higher than in flow) or go to a large pressure (out flow less than in flow). This capacitance does not affect the overall fluid volume in the system because it is limited to a small value.

Other locations that benefit from node capacitance are closed valves in series and closed valves tied to flow control externals.

Leak Analysis

Leak Analysis is the capability that Leakfinder adds to the functionality of Statefinder. During Leak Analysis, time averaged diagnostic flow patterns are analyzed for flow abnormalities. Any of the three types of abnormalities that are analyzed—leaks, injections, and circulations—may cause leak analysis alarms. The detectability of the leak often depends on the user-defined error bounds.

Leaks

Briefly, a *leak* is indicated if over the averaging period the diagnostic flows indicate a sufficiently large outflow from the system. The reason a leak in the pipeline produces an outflow in the diagnostic flow (which appears as a negative number for the external flow at the monitor external) can be interpreted by a simple thought experiment.

For example, assume that you have a gun-barrel pipeline with four monitors: A, B, C, and D. The flow direction is from A toward D. If the pipe segment between monitors B and C has a leak, the pipeline segment upstream of the leak between monitors A and B must supply the flow both to the leak and to the underlying flow, which continues to flow downstream between monitors C and D. Thus the pipeline segment downstream of the leak between monitors C and D will have less flow than the flow between A and B. The difference in flow is the amount of the leak. If the Statefinder Model is operating properly, the pressure differences between the monitors is consistent with flows in the Statefinder model, which conform to the flows in the corresponding segments in the pipeline. In the model, the excess fluid (corresponding to the amount of the leak delivered into the modeled pipe segment between B and C) has nowhere to go because the C to D segment is conforming to its pressure measurements. In order to stay in step with the monitored pressures from the field, the amount of the leak flow must come out through the Statefinder monitors.

In a well tuned system, this flow would flow out of the model mostly through monitors B and C. There would be some flow through A and D as well because of the least-squares nature of the solution process: The solution minimizes the sum of the squares of the diagnostic flows. Leakfinder analysis detects these flows as an indication of a leak and produces a LEAK ALARM status and a leak location position.

Injections

An *injection* is the complement of a leak. If the analysis code detects a sufficiently large positive flow in the monitors over the averaging period, it assigns an INJECTION ALARM status. Clearly, it is unlikely that a real injection is taking place. However, a faulty flow meter that is not reporting the actual flow into the line could be at fault. Other causes such as a rapid change in pressure over a significant part of the line, such as by initiation of a new supply, can appear as an injection if the bulk modulus of the fluid is assigned incorrectly.

Circulations

A *circulation* is indicated when time-averaged monitor flow at one or more monitor locations is negative, indicating flow out of the system, while it is positive by the same amount at one or more neighboring monitors. A circulation is caused by one or more modeling errors.

A typical cause of a circulation is the misplacement of a batch interface between fluids of differing density moving up or downhill in the pipeline. If the interface is in a different location in the model from the corresponding location in the pipeline, the hydrostatic head developed by the modeled fluid will be different from that developed in the pipeline. If the modeled hydrostatic head difference between monitors above and below the interface is less than the head difference between monitors in the pipeline, then the lower monitor in the model will take in fluid while the one above the interface will deliver fluid out.

Another typical cause of circulation is having defined incorrectly the frictional pressure drop in a pipe segment between monitors, or having the viscosity defined incorrectly, or both.

Leak analysis alarms

Leaks, Injections, and Circulations, which can all produce Leak Analysis alarms, depend upon a balance between modeling errors and predetermined quality of the data. For example, the presumed pressure and flow measurements have a given accuracy and repeatability, and presumed density changes with pressure or between batches are known to satisfy certain bounds, and uncertainties in the elevation with distance along the pipeline are presumed to have certain bounds. With this data the question to be resolved by Leak Analysis is, “Can any combination of the uncertainties in the modeling be consistent with the measurements from the pipeline if the pipeline has no leak?” If the answer to this

question is “Yes,” then there is no reason to a signal a leak alarm because the leak is undetectable given the data quality. If the answer to this question is “No,” then the Leak Analysis code in Leakfinder should alarm a leak.

Error bounds

In order to determine whether any combination of the uncertainties in the modeling can be consistent with the measurements from the pipeline if the pipeline has no leak, many of the quantities maintained by Leakfinder have error bounds. Some of the quantities with bounds come from measurements that are continuously autocalibrated while some are entered directly as data and never changed. Ultimately, the ability to excuse declaring that a set of data must imply a leak depends upon examining how all the measurements can be adjusted within their bounds to avoid the diagnostic flows that point to the leak. Thus, the ability to detect a leak depends upon your definition of the quality of the measurements and the accuracy with which the modeling data have been entered. This in turn depends on your supplying appropriate bounds for a number of quantities.

The user-defined error bounds used in Leak Analysis are:

Pressure repeatabilities	Entered on SCADA elements using the REP keyword.
Flow repeatability	Entered on SCADA elements using the REP keyword.
Bulk modulus error	Entered on SCL FLUID command using the BULK. MOD.ERR keyword.
Density error	Entered on SCL FLUID command using the DENSITY.ERROR keyword.
FRIC.ACC	Pokable on the “show online” display.
DRHO.MIN	Pokable on the “show online” display.
Elevation error	Entered on the pipe command using the ELEV.ERR keyword.
Slack volume bound	Entered on the pipe command using the SLACK.VOL.BOUND keyword.

Autocalibration of pressure drop errors

Modeling errors of pressure drops in spans can be contributed to three principal sources:

- Offset errors in pressure measurements
- Modeled pipe friction (because pipe roughness is not properly input). Pipe roughness is difficult to measure and changes slowly with time.
- Errors in gravity head in the span. Sources of head errors are:
    - Incorrect elevation profile
    - Errors in modeling fluid densities
    - Errors in batch interface location

In gas systems, gravity head errors are usually negligible because gas densities are usually low and have only small variations, and there are usually no sharp batch interfaces. In liquid statefinding, injection and in-line measurements usually provide adequate characterization of fluid density or sonic velocity, and the Interface Alignment (IA) element provides for adequate characterization of the interface location. Together, these often limit the error levels due to gravity head errors to approximately the magnitude of pressure measurement accuracy. When the gravity effects are negligible

or properly compensated by the gravity component of PDF, the pressure measurement and frictional errors can be reconciled using autocalibration.

At steady state, pressure calibration and friction errors are indistinguishable: a constant 1 psi calibration error in pressure measurement has the same signature as a 1 psi frictional pressure drop calibration error. This means simply that at a single steady state, the pipe friction correction can always be chosen to compensate for whatever pressure errors exist. Whether this friction calibration is “right” cannot be determined until the flow rate changes. When the change occurs, the friction calibration compensation changes with the frictional pressure drop, and an improperly calibrated system in balance at the first flow rate will appear unbalanced at the new flow rate.

The system will balance for all steady flow rates only if both the calibrated pressure measurements are correct and the frictional pressure calibration is correct. The purpose of autocalibration of pressure drop errors is to make use of these concepts to arrive at the single correct calibration for pressure measurements and the span-by-span friction (pipe roughness tuning).

Autocalibration requirements

To determine the correct calibrations, two or more sufficiently different almost steady states must be observed. The state must be almost steady for sufficient time for autocalibration to occur. The closeness to steady state and the time period are determined based on parameters of the span.

Moreover, for calibration to occur, it is necessary that there be good pressure measurements at each span end. Practically, the measurement points must be separated from the span ends only by equipment with minimal pressure drop. Minimal drop means pressure drop that is comparable to the pressure measurement repeatabilities. Further, calibration will only occur if the frictional pressure drop in the span is on average at least 10 times the sum of the span end monitor repeatability errors plus the pressure drop in the elements separating the pressure measurement from the span ends. Also, for liquid systems autocalibration will occur only when a span has not been in slack flow for some period of time.

Autocalibration control and display

You have control over autocalibration using the following peeks and pokes. For more information, see “Peek and poke keyletters and attributes for online modeling” on page 835.

TAP.AUTOCAL	This global poke value (show online) is the Time Averaging period for autocalibration. This is a time interval during which the pipeline data is expected to reflect two or more flow states that are approximately steady for comparable intervals. The default is 1440 minutes.
ACP.PRESS	This global poke value, Accuracy Correction Period for Pressure, is the period of time during which calibration corrections are made to the :AC values of span end SCADA pressures and to the span :FC values. The default is 60 minutes.

SPAN:DIF	This span peek is a measure of the suitability of the observed differences in flow through the span during the preceding TAP.AUTOCAL period. The value is evaluated by integrating functions representing the quality of the data, the observed frictional pressure differential and the modeled friction differential. It increases with separation of flow rates, and autocalibration of span end :AC's and span :FC will proceed while span:DIF exceeds DIF.LBOUND. :DIF = 10 * (1 - :MF^2 / (:MFMF * :DAT)). Note: :DIF = 0 at steady state.
DIF.LBOUND	This is a global poke, which is the lower bound of the span difference value for autocalibration. The default and minimum value is .5.
SPAN:DAT	This span peek is a measure of the quality of the data during the previous TAP.AUTOCAL period. It is determined by integrating SPAN:DATI. Autocalibration of the span end :AC's and the span :FC will proceed while SPAN:DAT exceeds DAT.LBOUND.
SPAN:DATI	This span peek is a measure of the instantaneous quality of the autocalibration data for the span.
DAT.LBOUND	This is a global poke, which is the lower bound of the span data quality value for autocalibration. The default and minimum value is .5.
SPAN:MF	This span peek is the approximate integral over the previous TAP.AUTOCAL period of the modeled frictional pressure drop weighted with the quality function, SPAN:DATI.
SPAN:SF	This span peek is the approximate integral over the previous TAP.AUTOCAL period of the SCADA inferred frictional pressure drop weighted with the quality function, SPAN:DATI.
SPAN:MFSF	This span peek is the approximate integral over the previous TAP.AUTOCAL period of the modeled frictional pressure drop multiplied by the SCADA inferred frictional pressure drop weighted with the quality function, SPAN:DATI.
SPAN:MFMF	This span peek is the approximate integral over the previous TAP.AUTOCAL period of the square of modeled frictional pressure drop weighted with the quality function, SPAN:DATI.

Expected results from autocalibration

This autocalibration method should result in lower diagnostic flows being generated with changes in flow rate. This is because the error in pressure between the SCADA data and the model data is divided more accurately between problems in pressure measurement and in pipe roughness values. This is true as long as the errors associated with the gravity terms are effectively handled with batch alignment and fluid property data. The corrections for :AC and :FC should be nearly constant after running for several Time Averaging Periods for Autocalibration.

Batch tracking in online models

For batch tracking in liquid pipelines, SPS models use either the Slightly Compressible Liquid (SCL), or SCL-properties (SCLPROP) equation of state. SPS tracks fluid compositions based on fluid names (SCL) or properties (SCLPROP). SCL is typically used for off-line models. For online models with telemetered fluid properties, SCLPROP is often preferred.

SPS tracks interfaces using fluid mixture vectors, FMVs, which identify the composition at a particular point in the pipeline. This approach is used so that SPS can appropriately represent blending that occurs due to valve swings and side stream injections.

Composition between any two FMVs is calculated by interpolating between the two FMVs. In this way, SPS calculates the composition at any point in the model. FMVs are automatically launched as needed so that this interpolation results in modeling accuracy consistent with user-defined tolerances.

Batch interfaces move through the model at the fluid velocity. When side stream injections occur, the fluid composition becomes a mixture of the flowing streams.

The shape of the interface is based on the compositions and flow rates, which were initially injected into the model, as adjusted by later mixing with side stream injections. When using SCLPROP, diffusion occurs between each pair of FMVs. All of the properties diffuse at a rate determined by the rate of density diffusion. Interfaces last until they leave the model at some delivery point or tank.

In this example, an interface between Fluid A and Fluid B is represented by several FMVs whose compositions change from 100% Fluid A to 100% Fluid B. The properties within each FMV will be determined by the previously mentioned rate of diffusion.

The exact definition of an interface may be subtle. If fluid composition changes from 80% fluid A to 70% fluid A, some might call this a new interface while others would not. The *batch bars*, which are available on distance plots or custom text displays, report an interface whenever the dominant fluid changes. The MIXWT parameter when using STATE SCL[PROP] can be used to control the definition of the dominant fluid.

For online models using SCLPROP, an Interface Alignment (IA) element is available to compare the shape of density or sonic velocity curves within the model to the output of the densitometers or flow meters in the field, and then automatically adjust modeled batch locations accordingly. A well-tuned online system with pump stations at 60 mile intervals should usually show batch locations to within a few hundred feet of their actual locations.

Errors

As a performance trade off, you can specify a volume at which interface detail is not needed. In other words, composition bumps that are small will be ignored. This is typically set at a few hundred barrels for pipelines in the thirty inch range.

The only element in SPS with transient volume is the pipe. This fact means that FMVs move through pump stations instantly. This simplification typically introduces only a very small error, but can be a problem if headers are used to model larger pipes. Note that two hundred feet of twelve-inch pipe has a volume of less than thirty barrels.

SPS stores explicit compositions at moving points within pipes. The distance between such points is typically a few hundred meters. Between these points, compositions are assumed to vary linearly as a function of distance along the pipe. Because fluid expansion/contraction due to temperature or pressure fluctuations between points may not be linear, there may be small errors in reported values between points.

Launching of FMVs

In order to minimize the number of FMVs in the model, SPS launches a new FMV only when necessary. It becomes necessary to launch a new FMV when the interpolation rules would otherwise calculate a composition that is in error by more than the user-defined tolerance (TRIVC - See "STATE SCL" on page 230). Also, SPS attempts to keep batches above the user-defined trivial batch size (TRIVB).

Batch tracking algorithm

The following steps are a part of the batch tracking algorithm. They take place every time step during a simulation.

1. Set up new FMVs for out flow pipe ends. This step moves the FMVs that are nearly out of any pipe to the pipe end so that the nodes will have the correct composition.
2. Iterate the stations. Some stations have multiple in-flows that blend together; SPS calculates the composition at every node, iterating if required.
3. Move the FMVs that are inside pipes. SPS moves each FMV through pipes using the fluid velocity profile in the pipe.
4. Smear composition discontinuities that move into a node. LINE.FILL commands and fast acting valves cause composition to be discontinuous; this discontinuity is represented by two FMVs at the same location in the pipe. SPS smears the discontinuity by separating the FMVs.
5. Launch new FMVs. For every pipe end, SPS keeps track of the change in fluid properties between FMVs. The TRIVC parameter from the SCLPROP command acts as a bound in determining whether FMVs should be added or removed. Because the calculation is a little complicated, you should just be aware that an increase in TRIVC decreases the number of FMVs while a decrease in TRIVC increases the number of FMVs. The variable NBATCH on the AUDITS display is an important parameter to look at when modifying TRIVC or TRIVB.
6. Update composition at in-flow pipe ends. As a result of step 2, SPS knows the composition at every node. In step 6, SPS moves the composition.
7. Remove trivial batches. For three sequential batches, A, B, and C, SPS removes B if the FMVs A and C are close together. FMVs at the pipe end and one in from the pipe end are never removed so that SPS always injects high fidelity FMVs.

Side stream injections

Side stream injections are handled primarily in steps 2 and 5 of the batch tracking algorithm. In the batch tracking algorithm, SPS uses a broad definition of *station*: every hydraulic element except pipes (T and GP) is in a station. Using this broad definition, all blending occurs at stations (some stations are simply nodes that connect pipes).

Painting of fluid properties

Some fluid properties for SCLPROP can be “painted” onto FMVs using the PROPERTY (PR) element in INPREP. The PR element functions by setting the property in the element-end FMV to the desired value. The effects spread through the model as described in the batch tracking algorithm.

Tuning an online model

This section supplements information in “Validating, Troubleshooting, and Tuning” on page 193.

Slack line tuning

Slack line flow occurs in a liquid pipeline when the pressure falls below the vapor pressure of the liquid at some point. Slack line flow is characterized by having a section of the pipeline in which the liquid no longer fills the pipe, and there is an interface between liquid and a vapor volume. The vapor in a slack section is essentially static, that is, the vapor is not transported with the liquid.

In a slack section, liquid may be flowing in channel flow along the lower pipe wall in the vapor section. This liquid trickles down into the interface separating the vapor section from the full section below. If the flow into the slack line section is lower than the flow of the fluid in the full section below, the slack line section will increase at a rate that is consistent with the depletion of the liquid by the difference of the flow out versus the flow in. If the rate of trickle flow into the vapor section exceeds the rate at which the liquid is flowing out in the full pipe, or if the flow in the fluid pipe is toward the vapor-liquid interface, then the vapor-liquid interface will rise, and may eventually fill the slack section so that the pipe will again run full.

Leak detection in full pipes is possible because pressures telemetered from monitors allow the consistency of the material balance within the pipeline to be correctly inferred. When the pipe is full, a change in the type of fluid present in a particular pipe volume can result in pressure changes. Therefore, knowing the pressure up and down the pipeline gives an accurate measure of the fluid present.

In the slack condition, this relationship is no longer true. If the pipe is almost level at the point of the vapor-liquid interface, enormous amounts of pipeline liquids can be removed without making much change in any of the pressures measured in the pipeline. Thus, in the presence of slack line flow the detectability of leaks is seriously degraded. Because large volumes of liquid may correspond to pressure uncertainties that typically correspond to minute amounts of liquid, the modeling process will often result in large diagnostic flows when actually a slack section is present. These diagnostic flows do not generally indicate a leak, and their presence up to an appropriate magnitude must be excused from the leak alarm process. The tuning parameters are associated with the adjustments of Leak Analysis near slack sections.

This section describes the tuning parameters that help you tune out false alarms due to slack line flow modeling errors. The basic idea is that if a pipe is in slack, or is close to slack, or has recently been in slack, a moderately large diagnostic flow should be excused as a modeling error. Nevertheless, a moderately large diagnostic flow should not be excused indefinitely.

Note that the model may not go into slack at exactly the same time as the pipeline because of the various uncertainties such as the uncertainty in vapor pressure. Therefore, the “close to slack” concept is important. Whenever the model is close to slack, the larger uncertainties associated with slack line flow are used.

For each pipe, you may enter a parameter to control the size of the diagnostic flow to be excused. The parameter is a volume and Leakfinder divides the volume by each time averaging period to calculate a flow rate. The syntax used to enter the volume is:

```
+ SLACK.VOLUME.BOUND volume
```

The peek/poke name is pipe:SVB.

Two global parameters control whether or not the pipe is thought to be close to slack. They are SL.PLOW (a low pressure) and SL.DTRESET (a time period). Whenever any pressure in a pipe gets lower than SL.PLOW, the pipe goes into *slack mode* and stays there until the pressure is above SL.PLOW for SL.DTRESET minutes. After the pressure has been above SL.PLOW for SL.DTRESET, the pipe comes out of slack mode. Slack mode means that the uncertainty in the flow change in pipe inventory for each time averaging period is increased by SVB/period. This increased uncertainty shows up in both the leak analysis and the leak thresholds. It does not have a direct effect on State Estimation, but it does also increase the calculated bounds in PDF.

A peek for each pipe says how long the pipe pressure has been above SL.PLOW. The peek is pipe:LPDT and has units of minutes. The algorithm for calculating LPDT is:

If any pressure in the pipe is less then SL.PLOW,

then set LPDT = 0

else set LPDT = LPDT + DT.

The algorithm for determining whether or not a pipe is in slack mode is:

If LPDT ≤ SL.DTRESET,

then the pipe is in slack mode

else the pipe is not in slack mode.

The LPDT parameter is pokable, and can be used to take a pipe out of slack mode more quickly than it would come out by the forgoing algorithm. If the pressure stays below SL.PLOW, the pipe will stay in slack mode regardless of any pokes to LPDT.

Note: SVB defaults to 0.01. Note that it does not hurt anything to have SVB nonzero even for a pipe that does not go into slack.

SL.PLOW defaults to 14.696 psia, which may be too low. A value of 15.0 psia will prevent more false alarms. It will also prevent more real alarms, but it is quite likely that any pipe that makes it below 15.0 psia will probably go into slack anyway. The vapor pressure is not used when setting slack mode because the uncertainty in the SVB dominates the other errors.

Performance tuning a Statefinder model

Tuning an online model for performance requires many of the same techniques used for an off-line model. For information on tuning an off-line model and the syntax used in this section, see "Performance tuning" on page 194. Performance tuning a Statefinder model requires additional considerations.

Statefinder tuning is an iterative process: you run through the same RTU data over and over until the model is capable of simulating the real pipeline, and the faster the model runs, the sooner benefits can be derived from its use in a production environment.

Once in the production environment, the model must keep up with real time to be useful. Consequently, model run speed continues to play an important role because it affects model sensitivity (selection of DTMIN) and ancillary capabilities (ARCHIVE intervals, TRENDLIST variables). Therefore, it is critical that the model runs as efficiently as possible so that it will keep up with real time, produce an accurate hydraulic simulation, and perform all desired tasks such as archiving and trending.

Each area for tuning lists those audits that will be affected and the probable amount of effort required to make the necessary analysis and changes. See "AUDITS (AU) attributes" on page 643 for a description of the relevant audits.

Equipment

Audits:	$MAIN
Effort:	High
Hydraulic impact:	High

Talk to field personnel and pipeline operators the purpose and use of each piece of equipment in the field to avoid the addition of any superfluous equipment (valves, compressors, control systems). Extra equipment will only slow the model down. A survey of this type should also be used in determining what elements should be used to model field equipment.

For example, a downstream regulator can be modeled with an idealized regulator (RE), a header flow element (HF), or a control valve with a full-blown control system. If the regulator is rarely used, it would probably be best to use an RE because it is both faster and easier than the other two options. On the other hand, if the element does play an integral role in daily operations, an HF might be the better choice.

TRENDLISTs

Audits:	$REVIEW
Effort:	Low
Hydraulic impact:	None

The TRENDLIST statement in the INTRAN file specifies the peeks that will be written to the REVIEW file. Once a model is fairly well tuned, you should define a TRENDLIST that is more discriminating. By reducing the TRENDLIST, $REVIEW will be reduced.

Consider the following example:

```
TRENDLIST *, KEY.LETTER = SC,
+ PEEK.MATCH = *:RAW *:USE *:EREP *:RC
```

In this TRENDLIST, rarely trended SCADA peeks such as :BMAX or :TOUT have not been included in the TRENDLIST and therefore will not be written to the REVIEW file and may not be trended. However, important peeks such as :USE and :RAW have been included in the TRENDLIST and therefore will be written to the REVIEW file and may be trended. You should develop similar TRENDLISTs for all the elements in the model after inspecting all element peeks to determine what needs to be trended in a production environment. For more information, see “TRENDLISTs” on page 195.

$REVIEW can also be affected by the amount of IMEM used with TRANS. For more information on memory settings, see “Memory allocation” on page 92.

SCADA pressures

Audits:	$MAIN, $ISOLVE, OL.NUMITR, NISOLVE
Effort:	Medium
Hydraulic impact:	High

The speed of the model is highly dependent upon pressure transients within the model. With this in mind, consider smoothing *all* of the :USE SCADA pressures by increasing JTS:P_RATE_WT. This will stabilize the model, reduce the OL.NUMITR, and increase DT. Smoothing the simulation by increasing JTS:P_RATE_WT is a far better approach to increasing the average DT than is increasing the knot spacing. Doing the latter usually condemns the model to run always at DTMIN. A model that always runs at DTMIN typically has greater errors in the hydraulic solution than one that runs at an even larger time step but takes smaller steps when needed.

IA element tuning

One of the most difficult sources of error in batched liquid pipelines crossing hilly terrain is batch interface misplacement.

Suppose the specific gravity difference across a batch interface is 0.05. This would correspond to 0.05 • 62.4 or a 3 lb/cu.ft difference in density. Also suppose the inlet flow meter at the point where the batch interface is created is 100 miles from a moderately steep hill, a 20% grade. And further suppose the flow meter accuracy is 1%. Then, by the time the interface has traversed the 100 mile pipe segment to the hill, its location may be modeled in error by plus or minus 1 mile. Suppose the side of the hill is 1 mile long, corresponding to a rise of 1000 feet (20% multiplied by 5280 feet).

If the interface location error is 1 mile along the pipeline (based upon the flow meter uncertainty) the modeled interface may be at the bottom of the hill whereas the pipeline interface may be at the top of the hill. The difference in pressure due to the error in hydrostatic head is 1000 feet multiplied by 3 lb/ft^3 divided by 144 to convert to psi units. This error is then 21 psi. This error is many times the typical error of modern SCADA pressure measurements. If this entire error must be corrected out (typically by invoking a 21 psi PDF) the leak detection sensitivity is essentially degraded by an amount comparable to that which would occur if the pressure monitors were 10 to 20 times worse than they are in the field.

If the error accumulates, a reasonable flow meter error translates to a damaging error. The purpose of the IA element in Statefinder is to allow intermediate correction of modeled batch interface location.

Assume that 50 miles from the launch point is an in-line densitometer. Then an IA element can be inserted into the model at the point of the densitometer. The IA element senses that the modeled batch interface does not arrive at the modeled point corresponding to the in-line densitometer at the same time as the batch interface arrives at the corresponding point in the pipeline. The IA element then moves the FMVs associated with the fluids on both sides of the interface at a different rate than the actual velocity in the modeled pipeline to cause the interface in the model to catch up with or slow down to match the location of the interface in the pipeline as revealed by the densitometer. Further, output from the IA element can be used to adjust the tuning of the flow meter so that subsequent corrections become smaller. In any event, the alignment error at the top of the hill discussed previously will be only half as much because the realigned interface must now travel only 50 miles to the hill.

A number of parameters are associated with optimizing the IA's ability to realign an errant modeled interface. These are summarized as follows:

- IA Mode and Status
- IA element mode (:MODE)
- Flow conditions (LOW_FLOW, REV_FLOW, EXT_FLOW)
- Condition of SCADA liquid property (BAD_VAL)
- State of property arrays (NOT_FULL)
- IA Event Discrimination
- Population of property curves (USD0/USV, DSD0/DSV, and EMD0/EMV)
- Range of property within signal range (:DDEN>:DDENTOL or :DVEL>:DVELTOL)
- Set :DDENTOL or :DVELTOL to smallest change in density between adjacent batches.
- Containment of entire interface within signal range (:SIWITHIN)
- Variation in density within signal range (:DDENV>:DDENVTOL or :DVELV>:DVELVTOL)
- To calculate a reasonable :DDENVTOL or :DVELVTOL, trend :DDENV or :DVELV over a large number of interfaces.
- IA Match Recognition
- Reasonable match is found (:MINU<:MINTOL)

- Comparison with the next best match (:R1U<:R1TOL)
- Comparison with a straight line (:R2U<:R2TOL)
- Comparison of matches found in EMD0/EMV and DMD0/DMV. Only occurs for leading model interfaces. (:DSF < :DSFTOL)
- IA Configuration
- Volume sizes
- Shift ranges
- Location
- Interaction with other IA elements
- Shift velocity (:VELLIM)
- CPU time per step

Differences in simulation time for online and off-line models

When running an off-line model, simulation start times and end times, as well as pipeline operation changes, are usually specified in minutes. However, for online modeling, you usually want the simulation time to correspond to the actual time. Therefore, a clock format is used that specifies the date and time. See “Elapsed time versus clock format” on page 199. When using clock format, displaying and keeping track of time in an online model is complex because:

- Computer networks may have a number of systems on them, and the times are usually not well synchronized.
- Pipelines commonly pass through several time zones.
- The system administrator can reset computer system time.
- Various computer operating systems have different treatments of time zones.
- Proper treatment of daylight savings time (also called summer time)

Many of these issues may be resolved by using the recommended time settings. Time-related environment settings for SPS should be stored in a settings file called sps.settings. For more information on the recommended settings, see “Display of simulation time on displays and reports” on page 200. For more information on how to set environment variables in the sps.settings file, see “sps.settings files” on page 91.

The simulation does not directly use the current system time, except to set the SYSTIME global variable for display. (See “GLOBALS (GB) attributes” on page 665.) The simulation step control for the online products is based on the time in the RTUDATA file.

Time zones and Daylight Savings Time

If using Daylight Savings Time, in many locales, but not all, the display of one hour per year is skipped and one hour per year is repeated. If the TZ environment variable is set to recognize Daylight Savings Time, then each year has one hour that occurs twice. This happens in the fall because when the clock says 2:00 a.m., it is set back to 1:00 a.m. This setting causes an ambiguity because, for example, the date/time “98/10/25 01:45:00” occurred twice. SPS supports a notation to resolve this ambiguity. The suffix, WIN, when specified after the time, means the time is interpreted using the winter clock. The suffix SUM means that the time is interpreted using the summer clock. For example, the following command will be executed the first 01:45.

POKE ABC = 5, TIME = "98/10/25 01:45:00 SUM"

The SUM and WIN suffixes can be specified on any time string, regardless of the date. WIN means suppress the Daylight Savings Time conversion, regardless of the season; SUM means perform the Daylight Savings Time conversion, regardless of the season.

Note: Southern Hemisphere users: SUM should mean the time around January, but some operating systems may not handle this correctly. You can check this with an INTRAN define. DEFINE TESTTIME = TIMEVALUE("2000/1/1 WIN") - TIMEVALUE("2000/1/1 SUM"). If you move your clock ahead one hour for the summer months, TESTTIME should have the value 60 (minutes).

INPREP file input for online modeling

When creating an INPREP file online modeling, you will need to add or modify the INPREP files used for off-line modeling. See "Differences between off-line and online models" on page 744 for more information on the modification required. This section contains commands and additional syntax used for online modeling.

Composition controller

INPREP

```
/* use this format for STATE SCL and STATE BWRS[.MOLE]
COMPOSITION NAME EXTERNAL
[+ FLUIDNAME1 FRAC1]
[+ FLUIDNAME2 FRAC2]
...

/* use this format for STATE SCLPROP
COMPOSITION NAME EXTERNAL
[+ FLUIDNAME]
[+ PRESSURE PRES]
[+ TEMPERATURE TEMP]
[+ PROPERTY VAL]
[+ PIPE ±pipename]
```

Field	Units Key	Description
NAME	n/a	Unique name of the composition controller.
EXTERNAL	n/a	Name of external to be controlled.
$FLUIDNAME_i$	n/a	The names of the fluids (or components) that are to be injected by the external.
$FRAC_i$	n/a	The fluid (or component) fractions that the external is to inject, corresponding to $FLUIDNAME_i$.
FLUIDNAME	n/a	Specifies the name of the fluid to use for properties that are not otherwise specified on the COMPOSITION element. {first fluid}
PRES	PRESSURE	Pressure at which the following fluid properties are measured.
TEMP	TEMPERATURE	Temperature at which the following fluid properties are measured.
PROPERTY	n/a	Specifies the property being controlled. One of DENSITY, BULK.MODULUS, TEMPERATURE.MODULUS, PTMULT, PPMULT, VISCOSITY, VPMI, VTMI, SONIC.VELOCITY, or INTERPOLATION.
VAL	user units	Value for the property.
±PIPENAME	n/a	Pipe end connected to the Composition controller for monitoring FMVs; use only if also using INTERPOLATION property.

Description

The COMPOSITION CONTROLLER (CO) controls the fluid composition of fluid flowing from an external into the model. The composition can be controlled by specifying fluid/component names, or in the case of STATE SCLPROP, specifying fluid properties.

The Composition Controller is used in conjunction with SCADA element(s) in online systems to set the fluid properties at a specified external. In general, the Composition Controller receives input from a densitometer (designated by CD0) or a viscometer (designated by CV0).

The densitometer ideally provides measured values of base density, bulk modulus, temperature modulus, PTMULT and PPMULT. In reality, the densitometer provides a base density, or possibly an API gravity that must be corrected to density units. Other parameters, used in calculating the density, will likely default to values supplied with the SCL equation of state definition for the selected fluid. An alternative to using default values is to measure these properties.

The viscometer ideally provides values of base viscosity and VTMI. The value of VPMI is commonly negligible. In reality, the viscometer may provide a measured viscosity at flowing pressure and temperature, which must then be converted into base viscosity. The Composition Controller does not currently support the ASTM parameters for viscosity definition. Therefore, fluid viscosity parameters specified in the SCL equation of state must use the base viscosity and VTMI parameters.

Some flow meters provide values for sonic velocity as well as flowing pressure and temperature. The pressure and temperature can then be used to convert the measured sonic velocity into base sonic velocity. The Composition Controller will convert the sonic velocity into bulk modulus, which is used to drive the bulk modulus of the fluid entering the pipeline.

The measured values of MD0, MBM, MTM, MPT, and MPP are typically configured to receive their values from a SCADA element that drives CD0, CBM, CTM, CPT, or CPP, respectively. These parameters are automatically ramped to their measured values whenever the corresponding SCADA element status is “good for use”.

Note: When controlling density, density.error is automatically ramped to :EREP from the density signal. When controlling bulk modulus, bulk.mod.error is automatically ramped to :EREP from the bulk modulus signal.

Likewise, the measured values of MV0, MVP and MVT are typically configured to receive their values from a SCADA element that drives CV0, CVP, or CVT, respectively. These parameters are automatically ramped to their measured values whenever the viscosity SCADA element status is “good for use”. See “SCADA” on page 791 for details on when a measured value is good for use.

Note: When controlling viscosity, viscosity.error is automatically ramped to :EREP from the viscosity signal.

The weight parameters are automatically set based on the quality of the corresponding SCADA element. If the SCADA element is good for use, then the weight is set to 1; otherwise, the weight is set to zero.

The property value to be used is calculated as:

$$\text{value}_i = \text{default}_i + \text{weight}_i \bullet (\text{measured}_i - \text{default}_i)$$

For a SCADA signal to be “good for use”, i.e., weight=1, the following must be true:

- The :MODE must be ON
- The :LIM must be GOOD and
- The :ST must be GOOD or NOCHANGE

Take note

Correcting “bad” FMVs caused by batch changes

During times of batch change, “bad” FMVs may be created due to instabilities in reported densities. These bad FMVs are created because the various user-defined calculations for the derived fluid properties may break down during periods of rapid composition change. The SPS identifies an FMV as bad if the :EREP of the SCADA driven density is more than 95% of the user-defined DENSITY.ERROR.

Several minutes after a batch change starts, the batch properties stabilize and the newly launched FMVs become good once again. After this occurs, the bad FMVs that have been launched into the model can be adjusted to have more reasonable properties, provided they are contained in one pipe and have one or more good FMVs at both the upstream and downstream ends of the pipe.

To correct bad FMVs, define a Composition Controller with both the INTERPOLATION and PIPE keywords. A SCADA element should drive composition:MINT to a line density using a density measurement. (Any property can be used instead of density provided it varies approximately linearly as the composition changes from one fluid to another one.) Every time step, the specified pipe is scanned for groups of bad FMVs that are sandwiched between good FMVs. If any groups are found, the fluid properties in the bad FMVs are adjusted as follows:

For base density, the updated values follow the shape of the :INT curve, scaled so they match the good FMVs that bound either side of the section of bad FMVs. For properties other than density, the updated values follow the shape of the :INT curve scaled for the specific property, but the maximum and minimum new values are bounded by the interpolation variables for the good FMVs. No update occurs if the difference between the :INT of the two bounding good FMVs is less than 10^{-6}.

Autocalibration of fluids

The fluid properties are initialized from the +FLUID definitions and are updated when the driving SCADA point is “good for use”. The properties are updated using time average exponential decay:

$$\text{default}_i = \text{default}_i + \left(\text{measured}_i - \text{default}_i\right) \times e^{\frac{1}{\text{ACP\_FLUID}}}$$

where:

default_i	=	default value associated with the fluid property, :D*
measured_i	=	measured value associated with the fluid property, :M*
ACP_FLUID	=	tuning factor from the “show online” display

A fluids.inc file, suitable for using in INPREP, is output every DT_FLUIDS interval using the so that the updated default values can be used on a subsequent run.

Example input

It is unlikely that sufficient instrumentation exists on a pipeline to measure all of the parameters that can be set using the composition controller. Probably, the only telemetered instrumentation available is that of fluid pressure, temperature, density and possibly viscosity. Fluid data for other properties, such as bulk modulus, temperature modulus, PPMULT, PTMULT, and TTMULT density multipliers, and viscosity multiplier indices VTMI and VPMI, would likely require custom fabricated instrumentation, which may be cost-prohibitive. These values can be set with a fake SCADA element, i.e., one that gets its value from an equation instead of from the RTUDATA file.

The first three examples in this section provide sample input for the Composition Controller element and its associated SCADA elements for cases in which density, temperature, and viscosity data is available from SCADA. The fourth example, by comparison, shows how much simpler the input data becomes when telemetered instrumentation is available for the less-often-measured and more obscure fluid parameters.

Example 1

Assume that density and viscosity instrumentation exist at a point downstream of a crude oil tank, TANK_123, which is injecting a fluid, CRUDE, into a pipeline. The measured density and viscosity drive the fluid property data used in the model. The SCADA point name for the density instrument is TANK_123_D, and the SCADA point name for the viscosity instrument is TANK_123_V.

First, the STATE SCL equation of state is specified using default values. See "STATE SCL" on page 230.

```
STATE SCLPROP   0.   5.
+ FLUID CRUDE
+ DENSITY     14.696 60  55.61 120000 -3041
+VISC  25.6    0.0  -.02009      /*EXPONENTIAL CURVE FIT APPROX.
+HCAP   0.465
```

Second, the fluid supply and its associated COMPOSITION element must be specified. See "E SALE/TAKE" on page 287.

For example:

```
/*<ext_name> <node> <type> <p> <q> <t> <Prate> <Qrate>)
E TANK_123   ND.123  TAKE   0.  0.  60.  1E6     1E6
/*             <name>      <ext>    <pressure>   <temp>
COMPOSITION  COMP_CRUDE  TANK_123  PRESSURE  0.0  TEMP 60
```

Third, the SCADA points that are used to drive the applicable parameters of the composition controller are specified. See "SCADA" on page 791.

If the measured density and measured viscosity are at base conditions, and in the same user units specified for density and viscosity, then the SCADA elements in the model would be:

```
/*   <name>      <limit macro>      <identifier>
SCADA  CRUDE_DENS, DENS_LIMITS, RTU.ID=TANK_123_D,
+ DRIVES=COMP_CRUDE:MD0 /* base density
/*   <name>      <limit macro>      <identifier>
SCADA  CRUDE_VISC, VISC_LIMITS, RTU.ID=TANK_123_V,
+ DRIVES=COMP_CRUDE:MV0 /* viscosity
```

where DENS_LIMITS and VISC_LIMITS are macros, which expand into several lines of text concerning default values of I.MIN, I.MAX, O.MIN, O.MAX, REP, etc.

:MD0 is the peek parameter for measured base density, and :MV0 is the peek parameter for measured base viscosity. The measured value, :MD0 or :MV0, is driven by the SCADA :VAL, even if the SCADA element status is BAD or OUT. The associated weight is 1 if the SCADA element is usable and 0 otherwise.

Example 2

If the density instrument output signal is in some unit of measure that is different from the density user units specified in the model, then a fake SCADA element must be used to convert the real SCADA signal into the appropriate user units. A variable to provide the “manual” density value must be defined, when the real SCADA is OUT, because the fake SCADA is never OUT.

The same fluid and composition controller as in “Example 1” above are used, except the density SCADA output signal is in units of API gravity (60/60).

```
/*  defined value can be a constant, or picked from the EOS
DEFINE FAKE_COMP_123_D = CRUDE:D0
/*  Dummy density value (lb/cu.ft)
/*   Real SCADA point outputs units of density in degrees API
/*      <name>  <limit macro>  <identifier>      Density
SCADA  CRUDE_API,  API_LIMITS,  RTU.ID=analog.TANK_123_DAPI(60/60)
/*   Fake SCADA point converts units from API to lb/cu.ft
/*   and provides a valid manual value when the real SCADA is out.
SCADA  CRUDE_DENS_F, DENS_LIMITS,
+  VAL = (CRUDE_API:ST = GOOD) * (( 141.5 / (131.5 + CRUDE_API)) *
+    62.37)  +   (CRUDE_API:ST!= GOOD) * FAKE_COMP_123_D,
+ DRIVES=COMP_CRUDE:MD0 /*<--- Composition Element
/*Density (lb/cu.ft)
```

Example 3

If the viscosity instrument measures kinematic viscosity (e.g. centistokes) at flowing temperature rather than absolute viscosity (e.g. centipoise) at base temperature, then a fake SCADA element must be used to convert the real SCADA signal into the appropriate user units. This also requires input from a density SCADA point and a temperature SCADA point. Two conversions are performed in order to obtain the correct output.

- Convert measured kinematic viscosity at line temperature into absolute viscosity at line temperature.
- Convert absolute viscosity at line temperature into base viscosity at base temperature.

The same fluid and composition controller as in “Example 1” above are used, except the viscosity SCADA output signal is in units of centistokes at pipeline conditions. The density SCADA point name is TANK_123_D and it is measured in lb/cu.ft at pipeline conditions. The temperature SCADA point is TANK_123_T and it is measured in °F (same as user units for temperature). Assume that fluid pressure has no effect on viscosity.

```
/* <name>   <limit macro>   <identifier>
SCADA  CRUDE_DENS, DENS_LIMITS,  RTU.ID=TANK_123_D
/* Density (lbs/cu.ft)

/* <name>   <limit macro>   <identifier>
SCADA  CRUDE_VISC, VISC_LIMITS,  RTU.ID=TANK_123_V
/*  Kinematic Viscosity  (Cst.)

/* <name>   <limit macro>   <identifier>
SCADA  CRUDE_TEMP, TEMP_LIMITS,  RTU.ID=TANK_123_T
/*    Temp.  (F)

/* define a default viscosity to use if SCADA is out
DEFINE FAKE_VISC_123 = 15
```

```
 /* Dummy Viscosity value (Cst)

/* define a default temperature to use if SCADA is out
DEFINE FAKE_TEMP_123 = 75
/* Dummy temperature (Deg. F)

/* Define a variable to convert units from CST to Cp at flowing temp.
/* and provide a reasonable alternative value if SCADA is bad.

DEFINE FAKE_VISC_AT_TEMP =
+    ( (CRUDE_VISC:ST = GOOD)  & (CRUDE_DENS:ST = GOOD) ) *
+      (CRUDE_VISC:USE * (CRUDE_DENS:USE / 62.37) )  +
+    ( (CRUDE_VISC:ST = GOOD)  & (CRUDE_DENS:ST != GOOD) ) *
+      (CRUDE_VISC:USE * (ND.123:DEN / 62.37) ) +
+    ( (CRUDE_VISC:ST != GOOD) * FAKE_VISC_123 )

/*   Fake SCADA point converts CP at flowing temperature
/*   to Cp at base temperature, based on SCL equation using VTMI.
/*        Visc(T) = Visc(60) * exp( VTMI * (T - 60 ))
/*     therefore...
/*        Visc(60) = Visc(T) / exp( VTMI * (T - 60 ))
/*         ^^^                      ^^^
/*        :MV0                     :VT

SCADA  CRUDE_VISC_F, VISC_LIMITS,
+  VAL = ( (CRUDE_TEMP:ST = GOOD) *
+ (FAKE_VISC_AT_TEMP / (2.7182818**(COMP_CRUDE:VT *
+ (CRUDE_TEMP:-SE - 60))))) + ( (CRUDE_TEMP:ST != GOOD) *
+ (FAKE_VISC_AT_TEMP / (2.7182818**(COMP_CRUDE:VT *
+  (FAKE_TEMP_123 - 60))))),
+ DRIVES=COMP_CRUDE:MV0  /*         <--- Composition Element
```

Example 4

The same fluid and composition controller as in "Example 1" above are used, except SCADA is available for every parameter desired, and in the proper units.

```
/* <ext_name> <node> <type> <p> <q> <t> <P rate> <Q rate)
E   TANK_123  ND.123  TAKE   0.  0.  60.   1E6     1E6
/*              <name>      <ext>         <pressure>      <temp>
COMPOSITION  COMP_CRUDE  TANK_123  PRESSURE  0.0  TEMP 60

/* <name>  <limit macro>  <identifier>  (pressure)
SCADA  CRUDE_PRES, PRES_LIMITS,  RTU.ID=TANK_123_P,
+ DRIVES=COMP_CRUDE:MP

/* <name>  <limit macro>  <identifier>   (temp)
SCADA  CRUDE_TEMP, TEMP_LIMITS,  RTU.ID=TANK_123_T,
+ DRIVES=COMP_CRUDE:MT

/* <name>  <limit macro>  <identifier>   (density)
SCADA  CRUDE_DENS, DENS_LIMITS,  RTU.ID=TANK_123_D,
+ DRIVES=COMP_CRUDE:MD0
```

```
/* <name>  <limit macro>  <identifier>  (bulk mod)
SCADA  CRUDE_BMOD, BMOD_LIMITS, RTU.ID=TANK_123_BM,
+ DRIVES=COMP_CRUDE:MBM

/* <name>  <limit macro>  <identifier>  (temp mod)
SCADA  CRUDE_TMOD, TMOD_LIMITS,  RTU.ID=TANK_123_TM,
+ DRIVES=COMP_CRUDE:MTM

/* <name>  <limit macro>  <identifier>  (viscosity)
SCADA  CRUDE_VISC, VISC_LIMITS,  RTU.ID=TANK_123_V,
+ DRIVES=COMP_CRUDE:MV0

/* <name>  <limit macro>  <identifier>  (vtmi)
SCADA  CRUDE_VTMI, VTMI_LIMITS, RTU.ID=TANK_123_VT,
+ DRIVES=COMP_CRUDE:MVT
```

Note: Assume that VPMI, PPMULT, PTMULT AND TTMULT are all zero. Macros for each set of limits must be previously defined.

Header flow (HF)

INPREP

```
HF NAME FROM TO
[+ JWT1  = JWT1,]
[+ JQ1   = JQ1,]
[+ JPS1  = JPS1,]
[+ JPD1  = JPD1,]
[+ JRHS1 = JRHS1,]
[+ JWT2  = JWT2,]
[+ JQ2   = JQ2,]
[+ JPS2  = JPS2,]
[+ JPD2  = JPD2,]
[+ JRHS2 = JRHS2]
```

Field	Units Key	Description
NAME	n/a	Unique name of the header flow element.
FROM	n/a	From-node or connection point.
TO	n/a	To-node or connection point.
JWT_n	n/a	The JTS weights for the equations.
JQ_n	n/a	Multiplier for the flow term in the equation.
JPS_n	n/a	Multiplier for the inlet pressure in the equation.
JPD_n	n/a	Multiplier for the outlet pressure in the equation.
$JRHS_n$	n/a	Desired value of the equation.

Description

The Header Flow (HF) element is used in online models to control flow and/or pressure change between two nodes in the model. It may be configured to represent a block valve, pump, compressor, regulator, check valve, or control valve. It is designed to produce a flow without regard to the pressure differential across the nodes. Conversely, it can produce a pressure differential without regard to the flow. The HF is typically driven by a SCADA element.

The flexibility offered by the HF element allows the model state to better match the process state than if an actual pump/compressor or valve element were entered. In the case of a valve or pump/compressor, various curves are input describing the behavior of the respective element. Because these curves are theoretical, some variance from the curves is expected in actual operation. Errors would be introduced to the model, thereby increasing the possibility of diagnostic flows. With the HF element, no curves are entered; it relies instead on the surrounding conditions and any user-defined goals to determine flow and pressures.

HF flow may be controlled in one of two ways:

- HF:Q is driven by a SCADA element
- HF is driven by one or both of the following goal equations:

$$JWT_1 \bullet (JQ_1 \bullet Q + JPS_1 \bullet P_- + JPD_1 \bullet P_+) = JWT_1 \bullet JRHS_1$$
$$JWT_2 \bullet (JQ_2 \bullet Q + JPS_2 \bullet P_- + JPD_2 \bullet P_+) = JWT_2 \bullet JRHS_2$$

Each of the coefficients are entered, while the remaining terms (Q, P_-, and P_+) are calculated. Note that since these are goal equations, they are solved in a least squares sense and not exactly. The JWT_i term, even though it appears on both sides of the equations, affects the scaling of the residual and hence the value of the least squares solution.

Take note

HF driven by SCADA

If a SCADA point is available that is an accurate measurement of the flow at the corresponding location in the pipeline, this method of control is preferable.

The HF element is defined with no goals, while the SCADA element drives the HF with the flow value available in the RTU data file, ± its effective repeatability based on the SCADA element driving the HF. The difference between :SQ and :Q are included in the calculations of the JTS:Q_ADJ equation. Therefore, JTS:Q_ADJ_WT will affect how closely :SQ and :Q match

HF driven by equations

In this case, a SCADA point for a particular location in the pipeline may or may not be available, or the SCADA point may be from a check meter and the measurement may not be accurate.

Both an HF and SCADA element are defined, but with a different functionality. Goals and weights are applied to the HF with the intent that the resulting flow is dictated by the individual equations as well as the global JTS equations.

The individual equations are specific to each element. A value greater than zero for either of the JTS coefficients (JWT_1 or JWT_2) activates the related equation. The greater that weight, the higher the penalty function for any deviation from the equation. The remaining coefficients are turned on and off by setting them to a non-zero value or zero.

The SCADA element for the HF is either a real SCADA point, such as a check meter flow, with a large repeatability or a fake SCADA point with an initial value and large repeatability. The SCADA definition will limit output values and repeatabilities.

The global JTS equations are also in the mix because the entire least squares minimization is done as one solution process. The individual HF equations become part of the overall equation for State Estimation as additional equations to solve. HFs with equations do not affect the Q_ADJ equation so changes to JTS:Q_ADJ_WT would not affect an individual HF with equations. Changing a global weight to correct a specific problem may affect the results of a different global weight, thereby changing the results of State Estimation. Achieving an overall balance of global weights, while ensuring the HF equations are active, is important.

Example input

Example 1

An in-line flow meter is modeled as an HF element. The following assumptions are made:

- A SCADA point is available that defines the flow at this location in the pipeline.
- The repeatability associated with this point is no larger than 3.

```
/* STA100_Q is the SCADA point name identifying the flow
/* in the RTU data file
SCADA STA100_Q, DRIVES = HF100:Q, REP = 3
HF HF100 ND100.IN ND100.OUT  /* HF as a flow meter
```

Example 2

Consider the difference in modeling a block valve as a Block Valve element or as an HF element. SCADA reports a block valve in the “not open” (meaning closed or in transit state) with pressure instruments on either side of the valve indicating a large pressure differential. The valve begins to open, has a 3 minute travel time, and has only one limit switch that is activated when the valve is full open.

The next SCADA scan (one per minute) reveals that the pressure differential has substantially decreased, but the valve status has not changed. As a result, the modeled block valve creates a diagnostic flow circulation in which flow exits the upstream pressure monitor and enters the downstream pressure monitor so that the pipes bypass the closed block valve.

An HF element driven by the internal equations takes into account the status of the block valve.

```
/*#STA100_Q is a fake SCADA point
SCADA #STA100_Q, DRIVES = HF100:Q, VAL = 500, REP = 500

/*Real SCADA point identifying the block valve limit switch signal
/*(0 = NOT OPEN,  1 = OPEN)
SCADA STA100.BLK.VLV
/*Block valve modeled as an HF
HF HF100 ND100.IN ND100.OUT
/*When valve is open, pressure differential = 0
+ JWT1  = STA100.BLK.VLV:VAL,
+ JQ1   =  0,
+ JPS1  = -1,
+ JPD1  =  1,
+ JRHS1 =  0,
/*When a valve is not open, flow = 0
+ JWT2  = (STA100.BLK.VLV:VAL = 0) * 1,
+ JQ2   =  1,
+ JPS2  =  0,
+ JPD2  =  0,
+ JRHS2 =  0
```

When valves are modeled using HFs as shown, the HF automatically compensates (at least partially) for the inevitable errors associated with the valve status from SCADA. The HF reduces the diagnostic flow circulation by taking on some of the effects of the error itself.

Interface alignment (IA)

INPREP

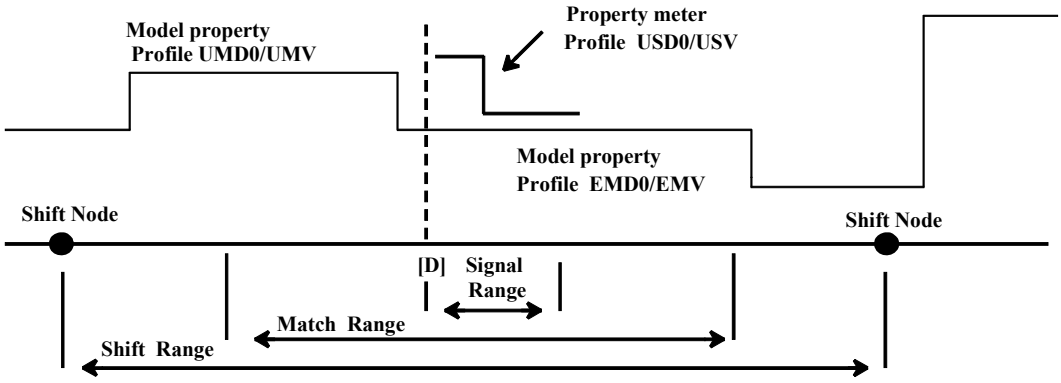
```
IA name cnc
+ SHIFT snu [snd]
+ MATCH matu [matd]
+ SIGNAL sig
[+ DENSITY | SONIC]
```

Field	Units Key	Description
name	n/a	Unique Name of the IA element.
cnc	n/a	From/to connection point or named node.
snu	n/a	Node name or connection point of the upstream end of the shift range. The upstream shift range is a section of pipe upstream of the IA element over which incremental velocity corrections will be made to the fluid interface.
snd	n/a	Node name or connection point of the downstream end of the shift range. The downstream shift range is a section of pipe downstream of the IA element over which incremental velocity corrections will be made to the fluid interface.
matu	VOLUME	Pipe volume upstream of the IA element. The IA element will attempt to find a modeled batch interface over this volume that matches the batch interface in the signal range. {10 MBBLS}
matd	VOLUME	Pipe volume downstream of the IA element. The IA element will attempt to find a modeled batch interface over this volume that matches the batch interface in the signal range. {10 MBBLS}
sig	VOLUME	Total volume of fluid downstream of the IA element's property meter for which the property meter values will be stored. As the property meter measures the property at a point with respect to time, this data is converted into property values with respect to volume of fluid downstream of the property meter. The property data that is available at any given time is designed to emulate the real property curve downstream of the property meter, as if there were a series of property meters at equal pipe volume intervals, until the total volume being sampled equals SIGNAL_VOL. {5 MBBLS}
DENSITY | SONIC	n/a	Property used by the IA element for pattern matching. {DENSITY}

Description

The Interface Alignment (IA) element, liquid only, is used to improve batch interface alignment. The IA element functions by *pattern matching* the measured liquid property with the modeled liquid property, and then shifting the modeled property as appropriate. The liquid properties that are currently supported by the IA element are density and sonic velocity.

The pattern matching is accomplished through the comparison of in-line measured property data with modeled property data to calculate the volume between the SCADA-derived batch interface and the model batch interface. The IA element causes the velocity of the fluid interface to change within the shift range in order to align the batch interfaces. The fluid property interface velocity is changed, but the actual fluid velocity used for hydraulic calculation does not change.



Sample Signal Range, Match Range, Shift Range, and Shift Nodes

After a new batch has been launched in the model, the resulting batch interface travels through the model's pipes at a velocity that is approximately, but not exactly, the same as the velocity of the real interface through the real pipeline. Depending on the differences between the real pipeline and the modeled pipeline, the model interface will be located in one of three locations relative to the real interface when the real interface passes a property meter at the IA element connection point:

- The model interface may lag the real interface, existing in the model pipe upstream of the property meter/IA element.
- The model interface may lead the real interface, existing in the model pipe downstream of the property meter/IA element.
- The model interface may lead the real interface, and may have already exited the model. This is typical in a situation where a full stream delivery is being made immediately downstream of the property meter.

Statefinder attempts to recognize batch misalignment by comparing the property meter data with respect to downstream volume to the model property with respect to volume upstream and downstream of the IA element. Statefinder converts the property meter data over time into property curves over some volume of downstream fluid. This data may be compared to property curves that are designed to represent the three possible locations of misaligned batch interfaces.

Statefinder takes the following steps to realign batches:

- *Check for the Existence of Any Conditions That Should Prevent Shifting.* There are various reasons to cancel a shift. See "IA mode and status" under "Take note".
- *Locate the Real Batch Interface.* Check the existence of a possible interface against discriminating criteria. This process is called event discrimination. Do not proceed until an interface passes the event discrimination checks.

- *Attempt to Find the Matching Model Batch Interface.* Once a batch interface is located from the signal data, SPS attempts to locate the corresponding interface in the model. If no match is found, then return to Step 1 and locate the next real interface.
- *Shift the Model Batch Interface Location.* If the corresponding interface is located in the model, then an incremental velocity is applied to the fluid interface in the shift range for each time step in an effort to cause the model batch interface to realign with the real batch interface as it reaches the IA element connection.

Take note

SCADA element

A SCADA element is typically used to drive the IA element. It is the SCADA element that provides the particular SCADA value that is used to feed densitometer or sonic meter data to the IA element SCADA density or sonic velocity set point. The general format for the SCADA element is:

```
SCADA DENS_001, DRIVES IA1:SD0    /* for density
SCADA SVEL_001, DRIVES IA1:SV     /* for sonic velocity
```

For sonic velocity, the pressure and temperature at which the measurement is made can be input into the IA using SCADA elements. The general format for the SCADA element for sonic velocity temperature and pressure is:

```
SCADA SVEL_P_001, DRIVES IA1:MP   /* pressure
SCADA SVEL_T_001, DRIVES IA1:MT   /* temperature
```

Shift range

It is generally appropriate for the shift range of an IA element to extend upstream to the previous IA element. If there is no IA element upstream, then the shift range may extend to an upstream external or the beginning of the pipeline. Keep in mind that if an external is located within the shift range, shifting will be disabled whenever non-trivial flow rates occur at the external. Also, loop or branch line flow will disable shifting if flow enters or leaves the system.

The downstream shift node is optional. You may elect to shift only in upstream pipes in cases where downstream interfaces are not related to upstream pipe activity. For example, if full stream delivery of the upstream fluid is made into a tank at the IA station while a full stream injection is made into the downstream pipe at the same flow rate. If this is the normal mode of operation, then the IA element may be defined without the downstream segment.

Multiple paths

If there are multiple paths from the IA connection node to a shift node, then the shortest path will be chosen.

Signal volume

The signal volume should be at least twice the size of the largest batch interface volume, yet smaller than half the sum of the match volumes. In addition, the larger the signal volume the less likely interface alignment will occur due to small spikes in the property curves.

Match volumes

The sum of the match volumes should be at least two times larger than the signal volume. They should be large enough to contain the misaligned batches when the SCADA batch interface is entirely within the signal volume. Because the real

interfaces are not recognized until they are well past the property meter, the following equations should yield reasonable match volumes:

Upstream Match Volume = U – S

Downstream Match Volume = D + S

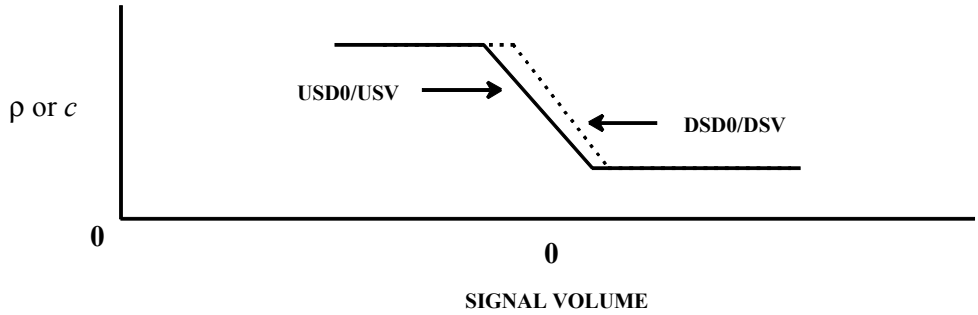
where:

U	=	Maximum misalignment volume when model batch interfaces lag real batch interfaces
D	=	Maximum misalignment volume when model batch interfaces lead real batch interfaces
S	=	Signal Volume

The ratio of match volume to signal volume is also critical. The larger the ratio of match volume to signal volume, the more computations will be required by Statefinder, thereby reducing CPU performance.

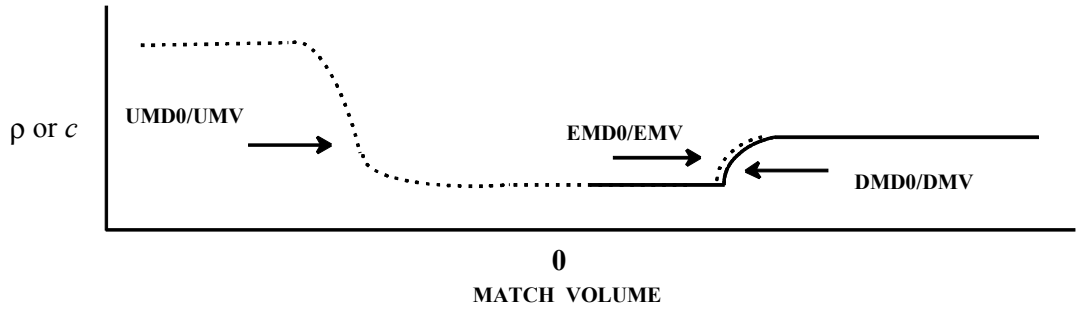
Density curves

There are five property curves that are generated by the IA element. Two of the curves are based on the signal property (typically from the SCADA system). The other three curves are based on the model property.



Sample Profile - SCADA Signal Property

UPSTREAM SCADA DENSITY/SONIC VELOCITY - (USD0/USV)	Signal property (density or sonic velocity) curve that uses the signal property value, the model fluid flow rate at the outlet of the *upstream* pipe, and the time step (DT) to construct a curve of the property as a function of volume over a volume equal to the signal volume, based on the *upstream* pipe's exit flow rate. The result is a curve that represents the property profile just downstream of the IA connection, as if flow were fed into a single pipe at the same rate as it left the upstream pipe.
DOWNSTREAM SCADA DENSITY/SONIC VELOCITY - (DSD0/DSV)	Signal property curve that uses the signal property value, the model fluid flow rate at the inlet of the *downstream* pipe, and the time step (DT) to construct a curve of the property as a function of volume over a volume equal to the signal volume, based on the *downstream* pipe's inlet flow rate. The result is a curve that represents the property profile just downstream of the IA connection, as if flow in the pipe at the same rate as it entered the downstream pipe. This property curve may be viewed using the distance plot item DSD0/DSV.
UPSTREAM MODEL DENSITY/SONIC VELOCITY - (UMD0/UMV)	Model property curve over the upstream match volume. This curve is comprised of the same data used to create a DENSITY or SONIC VELOCITY profile on a distance plot and will lie entirely upstream of the IA element. This curve is designed to display lagging model batch interfaces.
EXTENDED MODEL DENSITY/SONIC VELOCITY - (EMD0/EMV)	Model property curve that uses the model property associated with the IA element's connection node with the upstream pipe's exit flow rate to create a downstream model property profile. This curve is designed to display model batch interfaces, whether they lie in the downstream pipe or have exited the shift range through a pipe or external at the IA station.
DOWNSTREAM MODEL DENSITY/SONIC VELOCITY - (DMD0/DMV)	Model density curve over the downstream match volume This curve is comprised of the same data used to create a DENSITY or SONIC VELOCITY profile on a distance plot and will lie entirely downstream of the IA element. This curve is designed to display model batch interfaces, in the downstream pipe.



Sample Profile - Model Property

USD0/USV and DSD0/DSV curves will be exactly the same if the flow rate out of the upstream pipe matches the flow rate into the downstream pipe. This may not occur if there are external flows or other pipe flows within the IA station. (The IA station is defined as all of the equipment that can draw a path to the IA connection point without passing through a pipe.)

The USD0/USV, DSD0/DSV, and EMD0/EMV property curves are typically updated every time step using the appropriate liquid property, time, and flow information. Under certain conditions, however, the data in these curves will be purged (replaced with zeros). Updating of the curves will be suspended until the condition clears.

When the condition clears, updating of the curves is resumed. The curves are not fully constructed until sufficient volume has flowed past the IA connection point. For USD0/USV and DSD0/DSV, the volume required is equal to the signal volume, whereas for EMD0/EMV, the volume required is equal to the downstream match volume. Shifting upstream of the IA element is essentially disabled until USD0/USV and EMD0/EMV are fully constructed, while shifting downstream of the IA element cannot occur until all three curves are completely constructed.

IA mode and status

In order for the IA element to begin looking for interfaces to match and shift, the IA element must be enabled by poking :MODE = ON. Typically the status, :ST, is equal to RUNNING when the element is attempting to match and shift. :ST will reflect RUNNING until an interface match is made and a shift is activated. :ST will display SHIFTING when the IA element is shifting.

There are a number of conditions that may cause an IA element to be disabled. When these conditions exist, one or more curves may be purged, with the effect of disabling the IA element until all of the data can be replaced. The following is a description of the conditions and their effect on the IA element:

LOW_FLOW	The flow rate at the exit of the upstream pipe (:Q) is less than the tolerance (:QLOLIM). The test is designed to disable the IA element when the pipeline is being shut in. EMD0/EMV is purged.
REV_FLOW	The flow rate in any pipe within the shift range has reversed its normal flow direction. EMD0/EMV is purged.

EXT_FLOW	The time averaged flow rates through adjacent pipe ends within the shift range differ by more than 10%. The difference in flow rates is computed as follows: Difference in flow rates = ABS(Q1-Q2) / MAX(ABS(Q1), ABS(Q2)) where: Q1 and Q2 are 30 minute time-averaged flows at adjacent pipe ends Pressure monitor flows, branch line flows, loop flows, or external flows initiate this condition. EMD0/EMV is purged.
BAD_VALUE	The status of the SCADA element driving the IA signal property is not usable. New shifts will be disabled (but shifts in progress will not be canceled) and the USD0/USV and DSD0/DSV curves are purged.
NOT_FULL	At least one of three property curves, USD0/USV, DSD0/DSV, or EMD0/EMV, is not filled with data.
PROF_MISMATCH	The minimum $\Delta(\lambda)$ from the EMD0/EMV or DMD0/DMV curve was greater than the resultant $\Delta(\lambda)$ from the SCADA data match with a constant property profile OR :DSF was greater than :DSFTOL. PROF_MISMATCH can only occur if the model interface leads the SCADA interface.
OFF	:MODE is poked to OFF.

Note: The IA element will be disabled whenever the status is not RUNNING or SHIFTING. Except for the :ST=OFF, the IA element will go back to RUNNING automatically whenever the offending condition clears, and the data arrays necessary to enact a shift are full.

Locate the real batch interface

To determine whether or not USD0/USDV or DSD0/DSV includes an entire batch interface, the signal data in these two curves are subjected to four tests:

1. Is there a enough data to populate the required property curves? Previously it was stated that a number of conditions could cause the IA element to purge one or more of the property curves USD0/USV, DSD0/DSV, and EMD0/EMV. Once the condition clears, the IA element begins to fill the curves with property data once again. You may ascertain whether or not the curves are full of data by viewing a distance plot.
2. Do the highest and lowest points on the curves differ by a value greater than :DDENTOL or :DVELTOL? The default value for :DDENTOL is 0.5 lbm/ft^3 (8.0 kg/m^3). The default value for :DVELTOL is 40 ft/s (12.2 m/s).
3. Is the interface fully within the signal range? This is determined by looking for a relatively flat range in the most recent property curve data with the following test: Is the variation in the most recent one-third of the curve less than 25% of the variation in the entire curve? Mathematically:

   Let ϕ' be (dρ/dv), the derivative of the signal property with respect to volume. Then:

$$4\left[\int_0^{X/3}(\phi')^2\,dv\,/(X/3)\right]^{0.5} < \left[\int_0^{X}(\phi')^2\,dv\,/\,X\right]^{0.5}$$

   where X = signal volume

The peek name :SIWITHIN is calculated to be–[1 - (4 X recent variation) / (total variation)]. Therefore, :SIWITHIN ≥ 0 indicates the test is passed.

4 Does the interface have a sufficient variation in signal property over the entire range to imply the existence of a batch interface? This is determined by comparing the integration of the square of the derivative of the property with respect to volume over the signal range volume with a tolerance, :DDENVTOL or :DVELVTOL.

$$\left[\int_0^X (\phi')^2 \, dv \; / \; X \right]^{0.5} >: \text{DDENVTOL} \quad \text{or} \quad : \text{DVELVTOL}$$

where X = signal volume

The default value for :DDENVTOL is 0.001 (lbm/ft)/MB. The default value for :DVELVTOL is 0.08 (ft/sec)/MB. A value this small will effectively allow this test to be passed for all but the smallest differences in the signal property.

If each of these tests is passed, then Statefinder has determined that an entire batch interface has passed the IA element, and is within the signal range.

Sonic velocity temperature and pressure correction for pattern matching

For an IA element using sonic velocity as its input property, a correction to the data is needed to account for the effects of temperature and pressure on sonic velocity. The following steps occur before the IA attempts to match the model and signal interfaces:

1 The average pressure and temperature of the signal volume is determined.

2 The signal's sonic velocities are each converted to a velocity appropriate for the average pressure and temperate, using the following relationship:

$$c_S^2(p^*, T^*) = c_S^2(p, T) + k_p(p^* - p) - k_T(T^* - T)$$

where

c_s	=	sonic velocity
p*	=	average pressure in the signal range
T*	=	average temperature in the signal range
p	=	pressure at which the sonic velocity was measured (MP)
T	=	temperature at which the sonic velocity was measured (MT)
k_p	=	Pressure constant (2497 (ft^2/sec^2)/psi)
k_T	=	Temperature constant (78909 (ft^2/sec^2)/°R

3 The isentropic sonic velocity (at the average P,T determined in step 1) of the match volume will be computed at each point.

Once the previous steps are complete, the IA element has the signal volume data and the match volume data at comparable conditions for use in matching model and signal interfaces discussed in "Attempt to find the matching model batch interface" under "Take note".

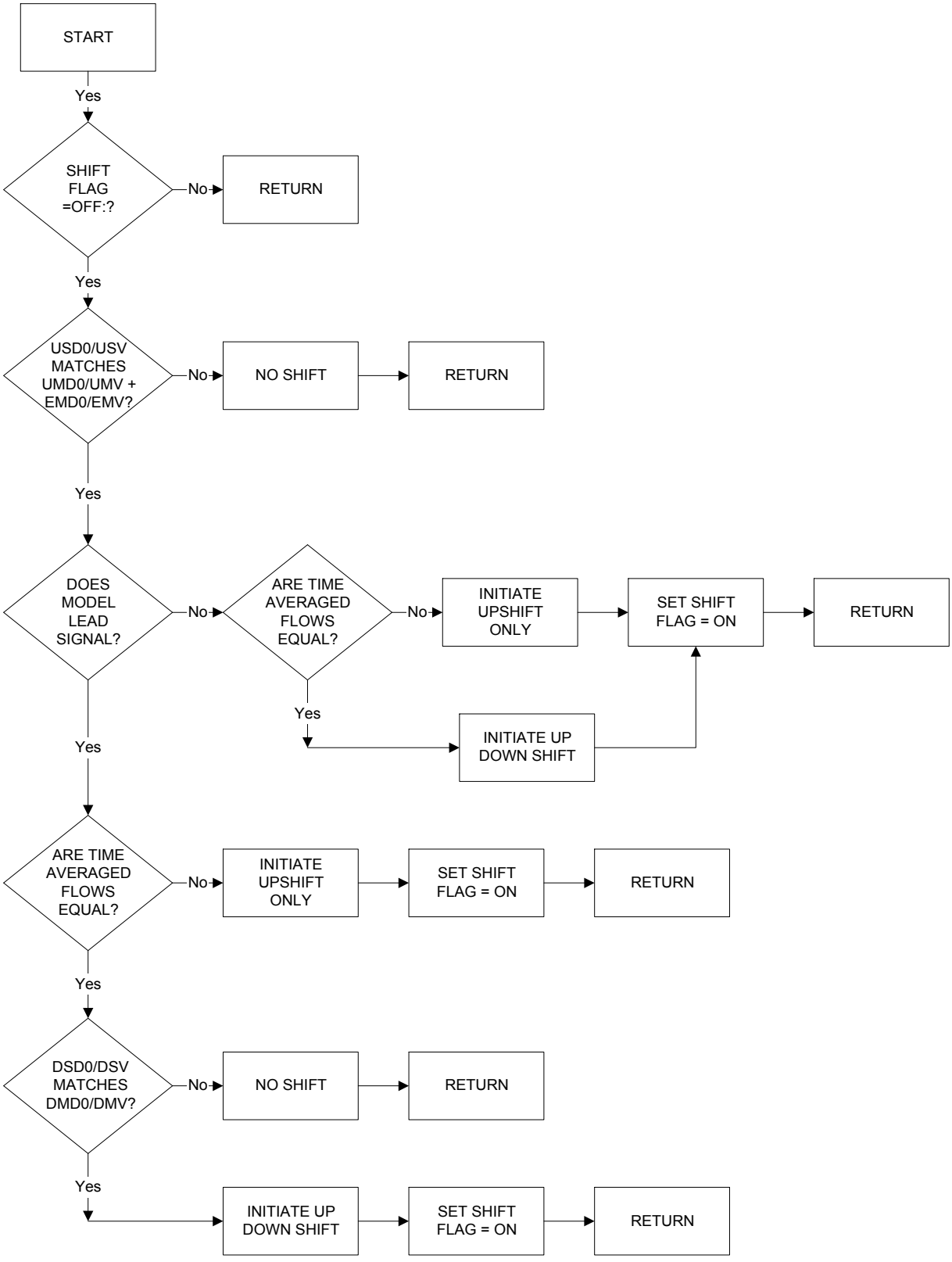
Attempt to find the matching model batch interface

Previously it was stated that there are five property curves that are updated each time step. Two of the curves are based on SCADA values from the IA element's property meter. The other three are curves are based on model data rather than SCADA data.

The property curve USD0/USV is compared first with a concatenation of the model property curve UMD0/UMV and EMD0/EMV to determine if there is a similarly shaped batch interface. If a match is found entirely in the downstream profile EMD0/EMV (the shift volume, λ, is positive), then another comparison is made of DSD0/DSV and DMD0/DMV to see if a match can be made in the same position as that found in EMD0/EMV. If a match is found so that :DSF is less than :DSFTOL, and there are no reasons to cancel a shift, then a shift will be invoked.

If the comparison of USD0/USV with a concatenation of UMD0/UMV and EMD0/EMV results in a negative shift volume, λ (the model interface lags the real interface), a shift is invoked if there are no reasons to cancel a shift.

The sequence of events that ensue once a shift is initiated is displayed in Matching Event Flow Chart.



Matching Event Flow Chart

A pattern recognition algorithm is used in trying to find a match. In effect, Statefinder "slides" a curve of model property values (UMD0/UMV, EMD0/EMV, or DMD0/DMV) along the smaller curve of SCADA density values (USD0/USV or DSD0/DSV) at constant volume intervals in an effort to find the shift volume (λ) that results in the closest match between the property values and the derivative of the property values. A positive value of λ indicates that the modeled batch interface is ahead of the real interface, and that the model fluid properties must be shifted in the upstream direction in order to find a match.

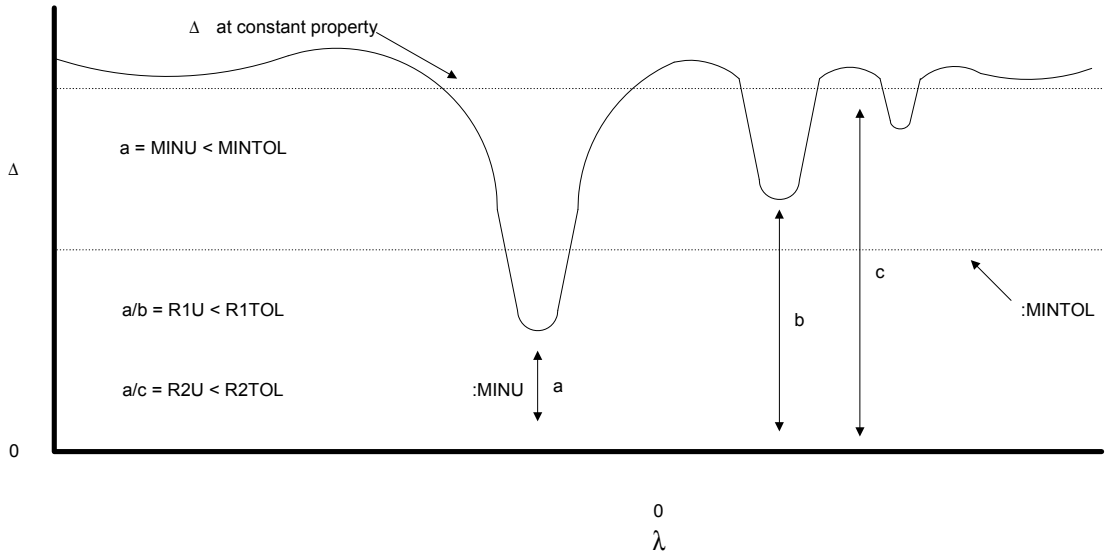
The pattern recognition algorithm attempts to minimize the following equation:

$$\Delta(\lambda) = \int_0^V \left[\alpha\left(\phi(X) - \psi(0, X-\lambda)\right)^2 + \beta\left(\phi'(X) - \psi'(0, X-\lambda)\right)^2 \right] dV$$

where:

Φ	=	change in property meter data with respect to volume
Ψ	=	change in model property data with respect to volume at some point, X, on the pipeline
Φ	=	property meter data as a function of volume
Ψ	=	model property data as a function of volume at some point, X, on the pipeline
λ	=	shift volume required to align the interfaces (ϕ with ψ).
α	=	:DENWT/:VELWT, the weight associated with the property values
β	=	:DDENWT/:DVELWT, the weight associated with the change in the property with respect to volume
V	=	:SIG, the signal volume

By choosing weights appropriately, you can dictate how much emphasis is placed on a particular component of the match (pattern recognition) equation. With a relatively high weight on :DENWT/:VELWT, a match will not be found unless the signal property values closely match the model property values. With a high weight on :DDENWT/DVELWT, a match will not be found unless the slope of the signal property curve closely matches the slope of the model property curve.



Definition of Match Algorithm Peeks

The algorithm confirms a match when the following occur:

- A minimum value of $\Delta(\lambda)$ is less than the tuning parameter :MINTOL.
- If more than one local minimum $\Delta(\lambda)$ exists, the value of the minimum $\Delta(\lambda)$ must be sufficiently smaller than the next smallest local minimum of $\Delta(\lambda)$. This condition is satisfied with the test:

  a < (:R1TOL) (b),

  where

  a = minimum value of $\Delta(\lambda)$

  b = second smallest local minimum value of $\Delta(\lambda)$
- The minimum $\Delta(\lambda)$ must be small when compared to the $\Delta(\lambda)$ resulting from trying to match SCADA data with a constant property profile. This condition is satisfied with the test:

  a < (:R2TOL) (c),

  where

  a = minimum value of $\Delta(\lambda)$ when comparing SCADA with the model property.

  c = $\Delta(\lambda)$ resulting from comparing SCADA with constant property value

  For leading model interfaces, a fourth test is also conducted.
- The $\Delta(\lambda)$ from DSD0/DSV and DMD0/DMV must be similar to the $\Delta(\lambda)$ from the DSD0/DSV and EMD0/EMV match. This condition is satisfied with the test:

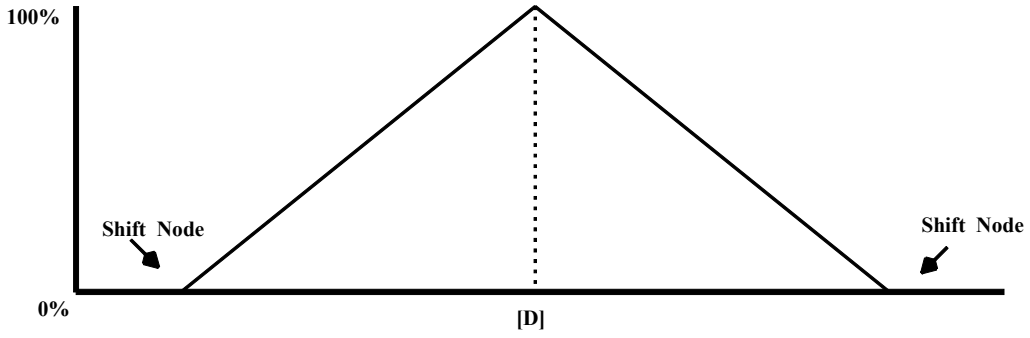
  :DSF < :DSFLIM

  where :DSF = $\Delta(\lambda)$DSD0/DSV with DMD0/DMV - $\Delta(\lambda)$DSD0/DSV with EMD0/EMV

Shift mechanism

Once an event is declared, the shift mechanism is started in order to bring the real interface and the modeled interface back into alignment with one another. The realignment occurs by applying an incremental velocity (either positive or negative) to the fluid properties in the pipe so that the property curve in the model changes to match the property curve in the real pipeline.

Once a valid realignment event is recognized, an incremental velocity profile is enforced upon the fluid properties between the shift nodes.



Sample Velocity Profile Enforced during Shifting when the Model Interface Lags the Signal Interface

This incremental velocity profile is enforced each time step until the shifted volume at the IA element is equal to the volume between the real interface and the model interface (λ = volume between matching interfaces) that existed prior to the shift. This can be demonstrated graphically with a time plot of :SVU (remaining shift volume in upstream pipe) or :SVD (remaining shift volume in downstream pipe).

Shift volume = Σ{(incremental velocity at IA element)(Pipe Area)(DT)}

Plotting IA information

If you want to plot IA information based on volume instead of length, specify NOMINAL.VOLUME as the first plot parameter and specify the X axis.

Example input

An IA element is installed at the discharge node of station RST. Because the misalignment of model interfaces with real interfaces can be as much as 15 MB in either direction, the `matu` and `matd` are set at 15 MB. The interfaces at RST are typically 5 MB in length. The `sig` is set arbitrarily higher at 7 MB. Because you want interface shifting to affect all of the fluid between station ABC and XYZ, the `snu` is set to ND.ABC.OUT while the `snd` is set to NO.XYZ.IN.

```
IA  RST_IA  ND.RST.OUT
+ SHIFT ND.ABC.OUT  ND.XYZ.IN
+ MATCH 15  15
+ SIGNAL  7
```

E MON

INPREP

```
E NAME CNC
+ MON PRES FLOW * [PRATE] [QRATE]
```

Field	Units Key	Description
NAME	n/a	Unique name of the external.
CNC	n/a	Connection point or named node.
PRES	PRESSURE	Initial pressure.
FLOW	FLOW	Initial flow rate. This must be zero.
*	n/a	Place holder.
PRATE	PRESSURE/MIN	Maximum rate of change of pressure.
QRATE	FLOW/MIN	Maximum rate of change of flow.

Description

The monitor external (E MON) is used to control the pressure in the model at a location where pressure is measured. Pressure measurement and quality information are passed to the model via the monitor and a SCADA element that controls the set point pressure of the Monitor (the :SP peek). A monitor is used only in the Statefinder and Leakfinder models. While a SALE or TAKE EXTERNAL is used in an online model to control flow into or out of the pipeline, the MONITOR EXTERNAL is used to control the pressure at a node.

During model startup (or after a LOAD.STATUS that uses the SET.TIME option), Monitor externals allow diagnostic flow into or out of the system in order to adjust the modeled pressure to agree with the measured pressure. After a certain period of time (:MAGE peek for the monitor), the monitor will close, preventing further diagnostic flow. However, monitors may be manually opened to correct pressure imbalances due to various data errors. :MAGE defaults to 1E20 minutes, which means by default monitors will stay open.

The leak detection function of LEAKFINDER models is built on top of the diagnostic flow. For leak detection to work, key monitors in the pipeline system must be open.

Example input

A MONITOR EXTERNAL named MON1 is connected to node CON2. This monitor external has an initial pressure of 900 psig.

```
E  MON1  CON2  MONITOR  900  0
```

FLOWMETER for online modeling

Peek and Poke Attributes

Description

This section supplements the documentation for modeling a flow meter for an offline model. For more information see “FLOWMETER” on page 279. This section documents the flow meter behavior when using the online products.

FLOWMETER models physical flow meters in the system. In online systems, a SCADA element controls the flow through the element.

The flow through the FLOWMETER targets the SCADA element’s :USE. The associated SCADA element’s :EREP determines the allowable variance in flow. The weight for reaching the target flow is Q_ADJ_WT/:EREP. An additional JTS goal is written to match the pressure loss across the FLOWMETER with a global weight of Q_ERR_WT. This weight should be small to allow the monitor pressures to be consistent with pipeline flow.

Example input

Flow meter has a normal flow of 250 MMCFD, an inlet pressure of 600 psig and an outlet pressure of 599 psig. The SCADA element M1234_Q drives the flow of the flow meter.

```
FLOWMETER M1234 N0123 N0124
+ PPQ  600 599 250
SCADA M1234_Q,
+ DRIVES = M1234:SQ,
+ REP = 5
```

PROPERTY

INPREP

```
PROPERTY NAME CNC SUBTYPE
```

Field	Units Key	Description
NAME	n/a	Unique name for the Property element.
CNC	n/a	Connection point or named node.
SUBTYPE	n/a	Key name for the fluid property being painted. For SCLPROP, the options are: DENSITY, DENSITY.ERROR, BULK.MODULUS, BULKMOD.ERROR, VISCOSITY, VISCOSITY.ERROR, and VAPOR.PRESSURE. For AGA, the options are: SG, CO2, HHV, and user-defined label.

Description

The Property (PR) element “paints” fluid property information (SCLPROP or AGA equation of state only) telemetered from field instrumentation onto the passing fluid, replacing the existing value with a new value, which is (presumably) more accurate. Property elements are typically located in the model where corresponding field instrumentation is located.

The PROPERTY element is normally driven by one or more SCADA elements. Some of the properties being painted require multiple SCADA elements driving different peeks such as the pressure and/or temperature to fully define the property being painted.

The connection point of the PROPERTY element is critical. To paint continuously, even during back flow, connect the PROPERTY element to a node. To paint only if the flow is leaving the element at the connection point, connect to the element end.

Take note

Equation of State details for PROPERTY

SCLPROP equation of state

The following refer to the SCLPROP equation of state

- **DENSITY**—A subtype of DENSITY allows for the passing of density and the associated pressure and temperature at which the density is measured. If the pressure and temperature are not driven by a SCADA element, the pressure and temperature at the connection point are used. The density plus the temperature and pressure are used to calculate the density at reference pressure and temperature, which is stored in the passing fluid.
- **DENSITY.ERROR**—A subtype of DENSITY.ERROR allows the passing of the density error.
- **BULK.MODULUS**—A subtype of BULK.MODULUS allows for the passing of bulk modulus and the associated pressure and temperature at which the bulk modulus was measured. If the pressure and temperature are not driven by a SCADA element, the pressure and temperature at the connection point are used. The bulk modulus along with the pressure and temperature are used to calculate the bulk modulus at reference pressure and temperature that is stored. The following equation is used:

$$PM_0 = PM + PM_P \bullet (P_0 - P) + PM_T \bullet (T_0 - T)$$

where:

PM_0	=	The bulk modulus at P_0 and T_0
PM	=	The measured bulk modulus at P and T
PM_P	=	The partial derivative of bulk modulus with respect to pressure
PM_T	=	The partial derivative of bulk modulus with respect to temperature

- *BULKMOD.ERROR.* A subtype of BULKMOD.ERROR allows the painting of the bulk modulus error.
- *VISCOSITY.* A subtype of VISCOSITY allows the passing of viscosity and the associated pressure and temperature at which the viscosity is measured. If the pressure and temperature are not driven by a SCADA element, the pressure and temperature at the connection point are used. The viscosity along with the pressure and temperature are used to calculate the viscosity at reference pressure and temperature. The reference viscosity is stored in the passing fluid. The viscosity can be painted for either the standard viscosity equation or the ASTM equation.
- *VISCOSITY.ERROR.* A subtype of VISCOSITY.ERROR allows the painting of the viscosity error.
- *VAPOR.PRESSURE.* A subtype of VAPOR.PRESSURE allows for the passing of a vapor pressure and the temperature associated with the vapor pressure. If the temperature is not driven by a SCADA element, the temperature at the connection point is used. These values along with the latent heat of vaporization are used to calculate the vapor pressure at pipeline temperature.

AGA equation of state

The following types refer to the AGA equation of state:

- *SG - Specific Gravity.* A type of SG allows the passing of the gas specific gravity.
- *CO2 - Carbon Dioxide.* A type of CO2 allows the passing of the carbon dioxide mole fraction.
- *HHV - Higher Heating Value.* A type of HHV allows the passing of the higher heating value of the gas.
- *Label.* A label is the user-defined name of the label specified in the STATE AGA definition. A new value for that label is passed.

FMV Updating

Standard FMV value rules apply when painting an FMV. If the painted value does not vary significantly from the existing FMV value, the value will not be painted. The allowable variance is defined using TRIVC on the SCLPROP input. In addition each fluid property has an assigned weight.

For each property in an FMV, a value S is calculated:

$$S = \left| \frac{fmv1\ [\ k\ ] - fmv2\ [\ k\ ]}{(\ wt\ [\ k\ ]\ )\ (\ max\ (\ fmv1\ [\ k\ ],\ fmv2\ [\ k\ ]\ )\ )} \right|$$

where:

fmv1[k]	=	The value of the property in the first FMV.
fmv2[k]	=	The value of the property in the next FMV.
wt[k]	=	The weight assigned to the property, not user-defined.

Then if S > TRIVC/25 launch a new FMV, otherwise do not.

Example input

Paint the vapor pressure with a PROPERTY element VP_100 connected to node ND100. The SCLPROP fluid input results in a vapor pressure of -10 psig at ND100. The measured value from SCADA element VP100 is -7 psig. When the model is run the vapor pressure upstream of ND100 will be -10 and the vapor pressure downstream will be -7. The input is:

```
PROPERTY  VP_100  ND100  VAPOR.PRESSURE
SCADA  VP100, DR  VP_100:MVP           /* vapor pressure
SCADA  VT100, DR  VP_100:MT            /* corresponding temperature
```

SCADA

INPREP

```
SCADA name,
[+ DRIVES                  = dr,]
[+ IDENTIFIER              = identifier,]
[+ VALUE                   = val,]
[+ INPUT.MINIMUM           = imin,]
[+ INPUT.MAXIMUM           = imax,]
[+ OUTPUT.MINIMUM          = omin,]
[+ OUTPUT.MAXIMUM          = omax,]
[+ RATE.BOUND              = rb,]
[+ TIME.AVE.PERIOD         = tave,]
[+ ACCURACY                = ACC,]
[+ REPEATABILITY           = rep,]
[+ TIME.TAG.ERR.B          = teb,]
[+ OFF.VALUE               = off,]
[+ BAD.MINIMUM             = bmin,]
[+ BAD.MAXIMUM             = bmax,]
[+ BAD.RATE.BOUND          = brb,]
[+ BAD.REP                 = brep,]
[+ MODE                    =ON | OFF,]
[+ REP.DECAY.RATE          = rdr,]
[+ TIME.OUT.PERIOD         = tout,]
[+ AUTOCAL.PERIOD          = acp,]
[+ FORECAST                = fore,]
[+ MANUAL.REPEATABILITY    = mrep,]
[+ NOCHANGE.REPEATABILITY  = nrep,]
[+ OFF.REPEATABILITY       = orep,]
[+ SCAN.PERIOD             = sp,]
[+ SECOND.DIFF.BOUND       = SDB,]
[+ INTERPOLATION.RULE      = LINEAR | EARLY | MIDDLE | LATE]
```

Field	Units Key	Description
NAME	n/a	Generally, this is the SCADA point name as defined in the RTU data file. For fake SCADA points (points not defined in a RTU data file), this is a name chosen to represent the fake SCADA point name.
DR	n/a	The poke point to be driven by this data point. For more information see "DRIVES keyword" under "Take note".
IDENTIFIER	n/a	This must correspond to a name in the RTU data file when NAME does not match the name in the RTU data file. The use of this keyword is mutually exclusive with +VALUE=val.

Field	Units Key	Description
VAL	user units	Used to define a “fake SCADA point” (one not driven by data from the RTU data file) in terms of any valid expression. Use of this keyword inhibits TRANS from searching for this point in the RTU data file. Mutually exclusive with +IDENTIFIER=identifier.
IMIN	user units	Minimum value accepted as GOOD from the SCADA system. If the value from the SCADA system falls below the value of :IMIN, :LIMR is set to LOW and :FILT is set to the interpolation of good SCADA values. {-1E10}
IMAX	user units	Maximum input accepted as GOOD from the SCADA system. If the value from the SCADA system rises above the value of :IMAX, :LIMR is set to HIGH and :FILT is set to the interpolation of good SCADA values. {1E10}
OMIN	user units	Minimum value accepted as GOOD after the :RAW value has been filtered by :IMIN, :IMAX, and :RB. {-1E10}
OMAX	user units	Maximum value accepted as GOOD after the :RAW value has been filtered by :IMIN, :IMAX, and :RB. {1E10}
RB	user units/TIME	Maximum allowable change in the value with respect to time. If :RB is exceeded, :LIMR is set to TOO FAST and :FILT is set to the interpolation of good SCADA values. {1E10}
TAVE	TIME	This is the period of time over which :VAL will be averaged. {0}
ACC	user units	The accuracy (and autocalibration) limits for the measurement. {50}
REP	user units	Repeatability associated with GOOD points. Used in calculating goodrep. It should indicate the maximum possible error in successive measurements of this parameter. {10}
TEB	TIME	The maximum time difference between the time a value is measured at the instrument and the time stamp associated with the point in the RTU data file. {0}
OFF	user units	When MODE is ON, :OFF is set to :USE. When MODE is OFF or :ST is not GOOD, then :OFF remains fixed at the last good value unless poked.
BMIN	user units	Minimum value accepted for this SCADA element when :ST is BAD. This is analogous to the role :IMIN plays when the point status is not BAD. {-1E10}
BMAX	user units	Maximum value accepted for this SCADA element when :ST is BAD. This is analogous to the role :IMAX plays when the point status is not BAD. {1E10}
BRB	user units/ TIME	Input filter used to constrain rate of change limits when :ST is BAD. {1E10}

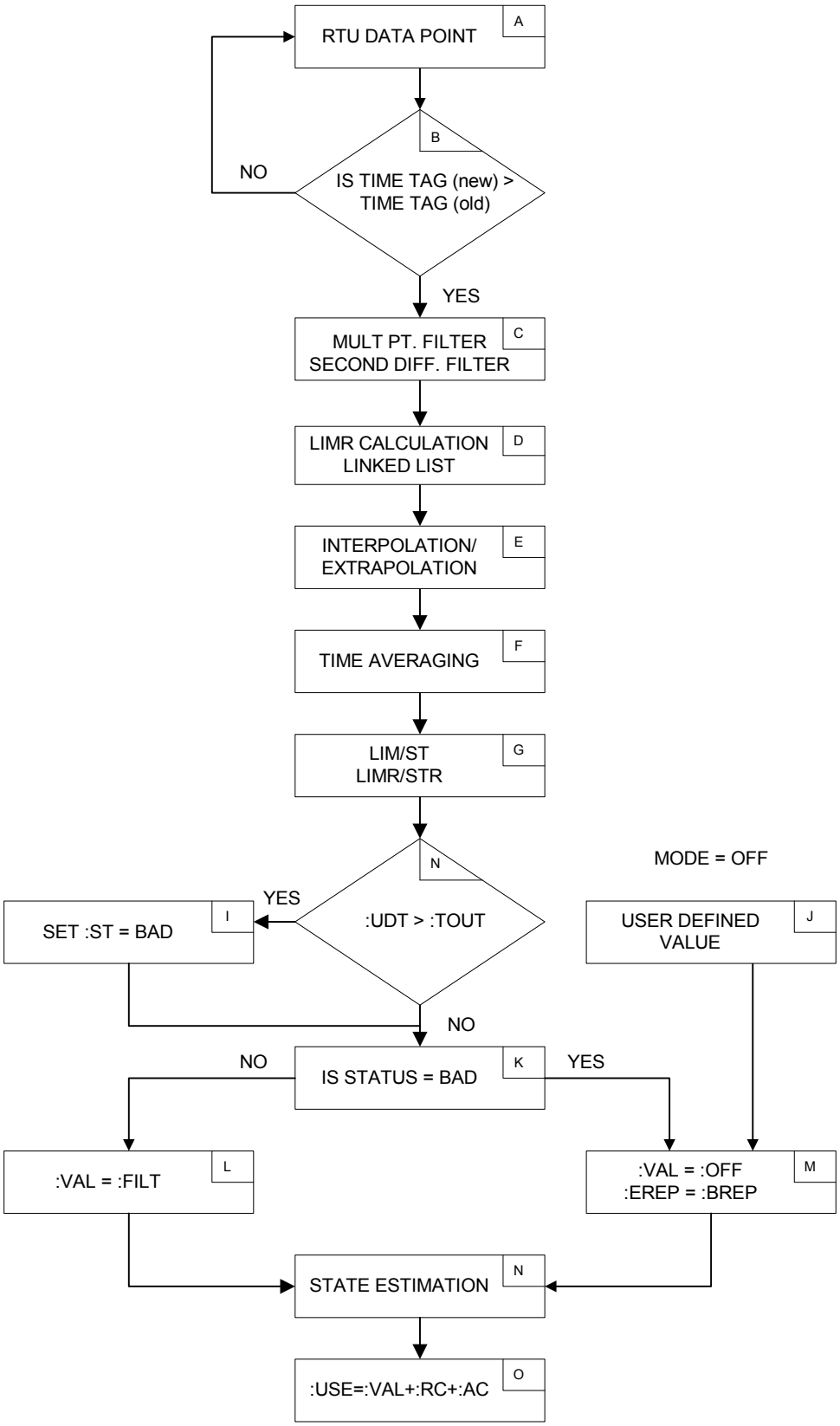
Field	Units Key	Description
BREP	user units	Repeatability associated with BAD points. Used in calculating badrep. {1000}
ON or OFF	n/a	If MODE is OFF (or :ST is BAD), then :VAL is set to :OFF. Otherwise, :VAL is set to :FILT.
TOUT	TIME	Period of time allowed between simulation time and closest data point in the RTU data file before a data point is considered out. :ST will be set to BAD when :UDT > :TOUT {5}
ACP	TIME	The period of time over which :AC is changed when autocalibrating. {43200}
FORE	user units	When using forecast data, the current value of the forecasted data. {0}
RDR	user units/TIME	The rate at which repeatability is increased to account for the change in the field value after measurement. {1}
MREP	user units	Repeatability associated with MANUAL points. Used in calculating manrep. {10}
NREP	user units	Repeatability associated with NOCHANGE points. Used in calculating nochrep. {10}
OREP	user units	Repeatability to used when :MODE is OFF. {10}
SP	TIME	Expected scan period. Any adjacent duplicate values within :SP of the first value are ignored. :EREP is not decayed unless :UDT > :SP/2. {0}
SDB	n/a	Bound on the second difference filter. Any data point that fails the second difference test is ignored. {1E10}
LINEAR EARLY MIDDLE LATE	n/a	Interpolation method used when reading information from an RTUDATA file. {LINEAR} • LINEAR uses constant linear interpolation. • EARLY interpolates the curve early in the interval between two RTU points. • MIDDLE interpolates the curve in the middle of the interval between two RTU points. • LATE interpolates the curve late in the interval between two RTU points.

Description

The SCADA element is used in online systems to provide a means of driving model elements with data from the real pipeline. Typically, the SCADA system writes process values to a binary RTU data file. This file is read and adjusted based on the SCADA element input.

SCADA element flow chart

To describe the functionality of the SCADA element, the *SCADA Element Flow Chart* follows:



SCADA Flowchart

The SCADA Element Flow Chart is a graphical depiction of the tests and filters used by the SCADA element to transform points from the RTU data file into filtered points that reflect the known instrumentation and data collection errors. To facilitate discussion of the chart, boxes within the chart will be commonly referenced by a letter in the upper-right corner of each box.

- *Box A - RTU data point.* Box A represents a discrete point collected by the SCADA system and added to a binary, circular RTU data file. Every point collected will have a point number, identifier, time stamp, value, and quality flag.
- *Box B - Time tag filter.* Once the point is collected from the RTU data file, its time stamp is compared with the last recorded time stamp for that :ID. If the time stamp from the last point collected is less than the time stamp from the previous point, the point is stamped with BAD TIME TAG and the point is discarded, otherwise the process proceeds to BOX C.
- *Box C - :RAW limit check.* Incoming points are compared with the last point added to the linked list to find occurrences of spurious multiple points. If two points are defined by (t1, v1) and (t2, v2); where t1 and t2 represent the time stamps and v1 and v2 the values, a multiple point is defined as one where |t1 - t2| < :SP and v1 = v2.

  A second difference filter is applied to all GOOD and NOCHANGE points. In order to apply the filter, there must be three points with the same status and:

  |y3 - y1| < 2 * :REP

  where y1 is the first point and y3 is the third point.

  If these conditions are met, the second difference is calculated:

  SD = y3 - 2*y2 + y1

  IF |SD| > SDB, then y2 is removed.
- *Box D - Linked list.* If the quality of the point is BAD, (:ST=BAD), then :BMIN, BMAX, and BRB are used to determine :LIMR. If the quality is not BAD, then :IMIN, IMAX, and :RB are used to determine :LIMR.

  Every point that passes the Time Tag Filter will be added to a Linked List specific to the :ID. These Linked Lists store time stamps, point values, and limit checks for each point. Calculate :RAW, :LIMR and :STR based on available SCADA points, regardless of quality.
- *Box E - Interpolation/extrapolation and nomination.* :EREP, :UDT, :FILT, and :FILP (for GOOD points) are calculated for each type of quality flag. The set with the lowest repeatability is applied to the point and displayed.
- *Box F - Time averaging.* If :TAVE is non-zero, :FILT is time averaged with the previous :FILT values.
- *Box G - Limit and status determination.* :LIM and :ST are determined.
- *Box H - Data outage check.* If the GOOD :UDT is greater than the :TOUT, the point is considered to be OUT and the process moves to Box I. Otherwise, the process moves to BOX J.
- *Box I - Data outage.* If the point has timed out, then :ST is set to BAD and :BREP, :BMIN, :BMAX and :BRB are used. This is the only time that the minimum repeatability is not used.
- *Box J - OFF mode.* User-defined :MODE to OFF and sets :OFF to a value.
- *Box K - Status check.* If :ST is not BAD, then :VAL = :FILT and :EREP is the lowest calculated repeatability. Otherwise, :VAL=:OFF and :EREP = :BREP if :ST is BAD or :OREP if :MODE is OFF.
- *Box L - :VAL determination when :ST!=BAD.* :VAL is set to :FILT and :EREP is set to the minimum calculated repeatability or the appropriate smoothing repeatability if the point is returning from a data outage.

- *Box M - :VAL determination when :ST=BAD.* If :ST= BAD or :MODE is OFF (element is in the manual mode), then :VAL is set to :OFF and :EREP = :BREP if :ST is BAD or :OREP if :MODE is OFF.
- *Box N - State Estimation.* By this point, data has been verified and is assumed to be reasonable. In order to determine :USE, the data is processed through a complex suite of procedures for data adjustment. This process considers data from all other instruments, as well as the fundamental laws of fluid dynamics as it meaningfully distributes errors to individual instruments.
- *Box O - :USE = :VAL + :RC + :AC.* :RC is determined by State Estimation and :AC is determined by the autocalibration process. See "Autocalibration of pressure drop errors" on page 752. Both of these terms are added to :VAL to determine :USE, which is the value used to determine the hydraulic state of the pipeline. :RC is constrained to be between +/- :EREP; also :RC is constrained so that :USE will between the :OMIN/OMAX filter (or :BMIN/:BMAX).

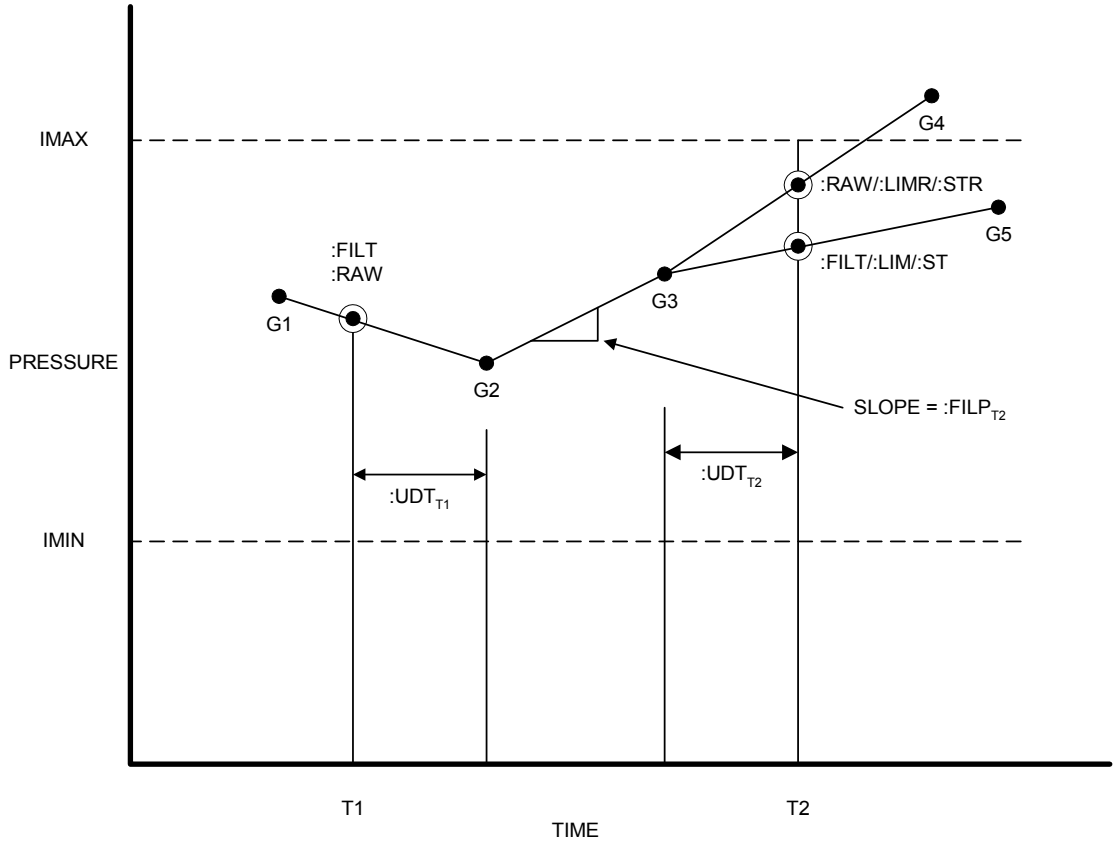
Take note

Philosophy

The TODREM routine captures the quality of each SCADA point and writes it to the RTU data file. Possible qualities include GOOD, BAD, MANUAL, NOCHANGE1 and NOCHANGE2. Definitions for each of these qualities may vary somewhat depending upon the individual SCADA system, but in general they are:

GOOD	SCADA system has no reason to doubt the point.
BAD	SCADA system doubts the point.
MANUAL	Point has been placed in manual and values are being manually entered.
NOCHANGE1	SCADA system passes in old value and indicates point has not changed.
NOCHANGE2	SCADA system indicates the value has not changed and TODREM uses old value, if available.

TRANS uses the quality flags to determine the believability of each value from the RTU data file. The quality flags help drive the effective repeatability of a SCADA element. This is done by calculating :FILT, :FILP, :UDT and :EREP, for each type of quality flag and then interpolating between each pair of points of quality that have passed the :IMIN, :IMAX and :RB limits. The set of values with the lowest :EREP value is selected. This prevents an erroneous increase in :EREP if MANUAL, NOCHANGE or BAD points are interspersed within GOOD points.



Calculation of :FILT, :RAW, :UDT, AND :FILP

Calculation of :FILT, :RAW, :UDT, and :FILP

In the previous graph, time step T1 is between two GOOD RTU data points that are within the :IMIN and :IMAX limits. As a result, :RAW and :FILT are simply the interpolated values between G1 and G2. :UDT$_{T1}$ is the time from T1 to the nearest RTU data point, which is G2.

Time step T2 falls between GOOD RTU data points but G4 is above the :IMAX limit. Therefore, :RAW is interpolated by using G3 and G4, but :FILT is interpolated by using G3 and G5. For time step T2, G3 is the nearest RTU data point once again, so :UDT$_{T2}$ is the difference between G3 and T2.

For a given time step, :FILP is calculated for each type of quality flag using the following equation:

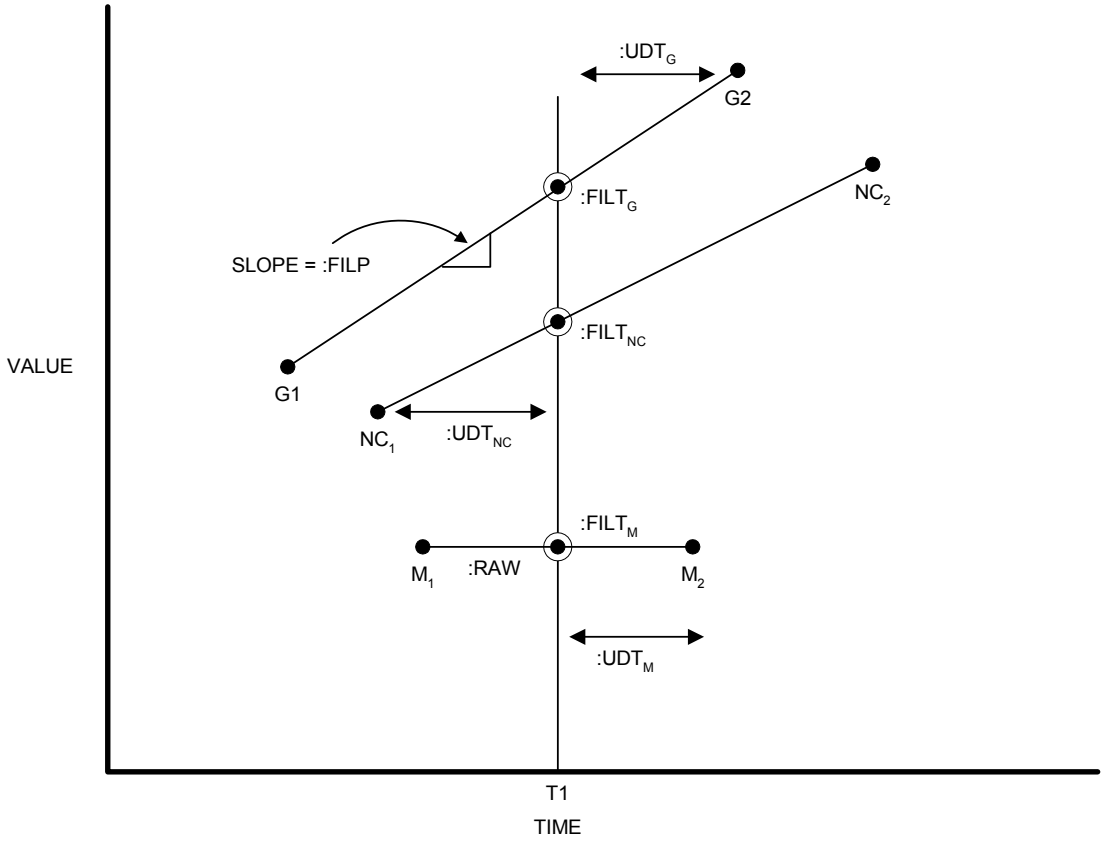
:FILP = MAX (ABS (SLOPE(i))

where SLOPE(i) is the slope of the interpolation line between a single pair of RTU data points.

The RTU data points that are included in the calculation of :FILP include the RTU data points between the current and previous time steps, and the RTU data point before the previous time step and the points after the current time step if available. In the diagram Calculation of :FILT, :RAW, :UDT, and :FILP, the data points used in the calculation of :FILP for the GOOD quality flag for time step T2 would be the GOOD RTU data point G1 immediately before time step T1, the GOOD RTU data point G5 immediately after time step T2, and all of the GOOD points, G2 and G3, between T1 and T2. Thus, :FILP for a GOOD quality flag would be the maximum of the absolute value of the slopes between the interpolation lines for G1 and G2, G2 and G3, and G3 and G5. The RTU data point G4 would be used in the calculation

of :FILP for the BAD quality flag because its value exceeds IMAX. The GOOD quality flag :FILP is used in the calculation of goodrep.

On the following graph, a :FILT and :UDT is shown for each type of quality flag by interpolating between each pair of RTU data points at time step T1. A :FILP is also shown for the GOOD interpolation. Because T1 falls directly between MANUAL points, :RAW is equal to the next MANUAL point in time, M2.



:FILT for Different Quality Flags

Effective repeatability

The raw repeatability of a SCADA point is the amount of random change in the output signal from the instrument while being supplied with a constant input signal, combined with the error induced in the analog to digital conversion of the value by the RTU. This error is in addition to the accuracy error, which accounts for the constant offset in the output signal from the true value.

The effective repeatability, when combined with the accuracy, determines the degree to which the SCADA value can be believed. The effective repeatability augments the repeatability of the instrument with three additional components:

- The temporal separation of the measurement and the model calculation
- The error associated with time stamp of the measurement
- An offset associated with changes between data outages and recovery

The equations used to calculate the effective repeatability for each status are :

goodrep = :REP + :RDR · max(:UDT - :SP/2, 0) + :TEB · :FILP
nochrep = :NREP + :RDR · max(:UDT - :SP/2, 0)
manrep = :MREP + :RDR · max(:UDT - :SP/2, 0)
badrep = :BREP
offrep = :OREP

where:

:RDR = is the raw repeatability decay rate
:UDT = is the time between the model time and the next or most recent SCADA time
:SP = SCADA system scan period for the SCADA point
:TEB = time error bound (the largest expected difference between the true measurement and the time indicated by the measurement time tag)
:FILP = is the :FILP calculated from RTU data points with a GOOD quality flag.

The repeatability decay rate (:RDR) is used to increase the effective repeatability for the instrument, depending on the degree of interpolation required to obtain a measurement estimate for a model time between two SCADA reported times.

The time error bound (:TEB) indicates the error introduced by uncertainties in the SCADA-reported time stamp. These errors may be introduced by such practices as assigning a single time stamp to a group of measurements on a single circuit poll, or to the entire set of measurements in a SCADA scan.

The process of finding the best fit of the data to describe the state of the pipeline involves the fact that the pressure and flow changes within the pipeline are fundamentally coupled by the laws of fluid dynamics. Thus each measurement is allowed to be adjusted from its SCADA-supplied value by the effective repeatability in order to find the best set of values that are hydraulically consistent.

Data outages can cause abrupt changes in :EREP, which can adversely affect the hydraulics of the system. Therefore, when data recovery occurs for more than one point, :EREP is gradually ramped down over a period :TOUT from :BREP to (:REP + :RDR * :TOUT) and then ramped over period :TOUT to the typically good value of :EREP. The :EREP that is applied is based on the following:

$$:EREP = \max(a, b, c)$$
$$a = \min(goodrep, nochrep, manrep, badrep, offrep)$$
$$b = :REP + dtrec1 \cdot :RDR$$
$$c = :REP + dtrec2 \cdot :BREP$$

where:

:EREP = effective repeatability
goodrep = effective repeatability if the :ST is GOOD
nochrep = effective repeatability if the :ST is NOCHANGE
manrep = effective repeatability if the :ST is MANUAL
badrep = effective repeatability if the :ST is BAD
offrep = effective repeatability if the ST: is OFF
dtrec1 = trec + 2 * :TOUT - dtime

dtrec2	=	trec - dtime
trec	=	(data recovery time) - :TOUT

Time averaging

The time averaging is:

$FILT = V_{n-1} + (V_n - V_{n-1}) \cdot e^{DT/TAVE}$

(V_i is the interpolation of GOOD :RAW values.)

This type of averaging is called (exponentially) weighed time averaging, and has the property that a step change in V will cause :FILT to move towards the new V with a difference that decays by a factor of e every time averaging period, TAVE.

Good for use

Many of the elements that are driven by SCADA elements will automatically switch to using defaults when the raw data is of low quality. The test used to determine if the data is Good For Use is:

```
Good For Use = (:LIM = GOOD) & (:ST = GOOD | :ST=NOCHANGE) & (:MODE = ON)
```

Examples of elements that use this logic are the "Composition controller" on page 763 and "PROPERTY" on page 788.

Autocalibration

Autocalibration is performed on GOOD For Use data, using the following:

$$AC_{ADJ} = \left(\frac{RC_n}{|RC_n|} - \frac{AC}{2 \times ACC} \right) \times \frac{DT}{ACP} \times EREP$$

$$AC_n = AC_{n-1} + \text{bound}(-1, AC_{ADJ}, +1)$$

where:

DT	=	current time step in minutes
AC_n	=	current accuracy correction
AC_{ADJ}	=	adjustment to the accuracy correction this step
RC_n	=	repeatability correction called for this time step by State Estimation
ACC	=	absolute value of the limit for accuracy correction :AC
ACP	=	autocalibration period
EREP	=	effective repeatability

bound(-1,$AC_{ADJ,}$1) is the value of $AC_{ADJ,}$ bounded by -1 and +1

Autocalibration does not occur when:

$$ACC \leq :EREP \text{ or } ACC \leq 0.0$$

Note: The autocalibration described here does not apply to SCADA elements driving Monitors that are at the endpoints of Spans. The autocalibration that applies for Spans is discussed in "Autocalibration of pressure drop errors" on page 752.

:USE

USE is defined as follows:

$$USE = VAL + RC + AC$$

USE assumes the value of the OFF peek if MODE is OFF or ST is BAD. If MODE is ON, RC is determined by SCADA element filtering and State Estimation.

Extrapolation conditions

The model will always interpolate instead of extrapolate unless one of the following conditions is met:

Condition 1

tnewpt - MODEL.TIME > 2 hours

where:

tnewpt = Time tag of the most recent point

Condition 2

of points in memory ≥ MAXSCANS * (# of RTU ids)

where:

MAXSCANS	=	Tuning parameter viewed with show online command defined as the number of RTU data points stored in memory.
# of RTU ids	=	The number of SCADA elements in INPREP file

Condition 3

tnewpt - LF.MAXWAIT > MODEL.TIME + dtmin

where:

LF.MAXWAIT	=	Tuning parameter viewed with show online command

Even if the model meets one of these conditions, the model will still interpolate all points for which there is data.

SCADA interpolation example

A sample RTU data file, containing values for a point named P.DELIV, is as follows. The input parameters for the SCADA element are shown. Note that user units are psig and time is in minutes:

```
REP = 2     RDR = 1     IMAX = 1000
BREP = 1000 SP = 0.5    TOUT = 5
NREP = 6    TEB = 0.5   MAXSCANS = 20
MREP = 50   IMIN = 0
Date       Time     SCADA ID      Value   Quality
94/08/15 00:01:00   P.DELIV        405      GOOD
94/08/15 00:02:00   P.DELIV        395      GOOD
94/08/15 00:03:00   P.DELIV        390      GOOD
94/08/15 00:04:00   P.DELIV        390      NOCHANGE1
```

```
94/08/15 00:05:00   P.DELIV        390      NOCHANGE1
94/08/15 00:06:00   P.DELIV        390      NOCHANGE1
94/08/15 00:07:00   P.DELIV        390      NOCHANGE1
94/08/15 00:08:00   P.DELIV        390      NOCHANGE1
94/08/15 00:09:00   P.DELIV        390      NOCHANGE1
94/08/15 00:10:00   P.DELIV        390      NOCHANGE1
94/08/15 00:11:00   P.DELIV        390      NOCHANGE1
94/08/15 00:12:00   P.DELIV        390      NOCHANGE1
94/08/15 00:13:00   P.DELIV        390      NOCHANGE1
94/08/15 00:14:00   P.DELIV        -999      BAD
94/08/15 00:15:00   P.DELIV        -999      BAD
94/08/15 00:16:00   P.DELIV        -999      BAD
94/08/15 00:17:00   P.DELIV        -999      BAD
94/08/15 00:18:00   P.DELIV        -999      BAD
94/08/15 00:19:00   P.DELIV        -999      BAD
94/08/15 00:20:00   P.DELIV        410      MANUAL
94/08/15 00:21:00   P.DELIV        410      MANUAL
94/08/15 00:22:00   P.DELIV        410      MANUAL
94/08/15 00:23:00   P.DELIV        410      MANUAL
94/08/15 00:24:00   P.DELIV        410      MANUAL
94/08/15 00:25:00   P.DELIV        410      MANUAL
94/08/15 00:26:00   P.DELIV        457      GOOD
94/08/15 00:27:00   P.DELIV        459      GOOD
94/08/15 00:28:00   P.DELIV        460      GOOD
94/08/15 00:29:00   P.DELIV        465      GOOD
94/08/15 00:30:00   P.DELIV        470      GOOD
```

The following table displays the model time and the raw value from the RTU data file.

```
Model Time    :RAW
00:02:30      392.50
00:04:00      390.00
00:06:30      390.00
00:09:00      390.00
00:11:15      390.00
00:13:30       -999.00
00:15:30      -999.00
00:17:30      -999.00
00:20:00      410.00
00:23:30      410.00
00:26:15      457.50
00:28:30      462.50
```

The following table displays the calculated repeatabilities for each possible status. The badrep is equal to :BREP (1000); the offrep is equal to :OREP (1000).

GOOD				NOCHANGE			MANUAL		
:EREP	:UDT	:FILP	:FILT	:EREP	:UDT	:FILT	:EREP	:UDT	:FILT
7.25	0.50	-10.00	392.50	7.25	1.50	390.00	67.25	17.50	410.00
5.25	1.00	5.00	392.91	6.00	0.00	390.00	65.75	16.00	410.00
6.71	3.50	2.91	400.19	6.25	0.50	390.00	63.25	13.50	410.00

GOOD				NOCHANGE			MANUAL		
:EREP	:UDT	:FILP	:FILT	:EREP	:UDT	:FILT	:EREP	:UDT	:FILT
9.21	6.00	2.91	407.48	6.00	0.00	390.00	60.75	11.00	410.00
11.41	8.25	2.91	414.03	6.00	0.25	390.00	58.50	8.75	410.00
13.71	10.50	2.91	420.59	6.25	0.5	390.00	56.25	6.50	410.00
13.71	10.50	2.91	426.41	8.25	2.50	390.00	54.25	4.50	410.00
11.71	8.50	2.91	432.24	10.25	4.50	390.00	52.25	2.50	410.00
9.21	6.00	2.91	439.52	12.75	7.00	390.00	50.00	0.00	410.00
5.71	2.50	2.91	449.72	16.25	10.50	390.00	50.25	0.50	410.00
3.46	0.25	2.91	457.50	19.00	13.25	390.00	53.00	3.25	410.00
4.75	0.50	5.00	462.50	21.25	15.50	390.00	55.50	5.50	410.00

The following table shows the nominated value where :EREP = min(goodrep, nochrep, manrep, badrep). The :UDT, :FILP, and :FILT values correspond to the selected repeatability. The :ST and :LIM are based on the selected repeatability.

NOMINATED				STATUS/LIMIT			
:EREP	:UDT	:FILP	:FILT	:ST	:LIM	:STR	:LIMR
7.25	0.50	10.00	392.50	GOOD	GOOD	GOOD	GOOD
5.25	1.00	5.00	392.91	GOOD	GOOD	NC	GOOD
6.25	0.50	0.00	390.00	NC	GOOD	NC	GOOD
6.00	0.00	0.00	390.00	NC	GOOD	NC	GOOD
6.00	0.25	0.00	390.00	NC	GOOD	NC	GOOD
6.25	0.50	0.00	390.00	NC	GOOD	NC	LOW
8.25	2.50	0.00	390.00	NC	GOOD	BAD	GOOD
10.25	4.50	0.00	390.00	NC	GOOD	BAD	GOOD
1000.00	6.00	2.91	439.52	BAD	GOOD	MANUAL	GOOD
5.71	2.50	2.91	449.72	GOOD	GOOD	MANUAL	GOOD
3.46	0.25	2.91	457.50	GOOD	GOOD	GOOD	GOOD
4.75	0.50	5.00	462.50	GOOD	GOOD	GOOD	GOOD

Calculations

Explanations of the calculations and the selections at different times are as follows:

Time 00:04:00

$$\text{goodrep} = 2 + 1\ (\ 1 - (\ 0.5 / 2\ )\ ) + 5.00 \bullet 0.5 = 5.25$$

$$\text{badrep} = 1000$$

$$\text{nochrep} = 6 + 1\ (\ 0\ ) = 6.00$$

$$manrep = 50 + 1 (16 – (0.5 / 2)) = 65.75$$

$$:EREP = min (goodrep, badrep, nochrep, manrep) = 5.25$$

Therefore, the GOOD :UDT, :FILP, and :FILT are used and :ST = GOOD. :LIM = GOOD because :FILT passed all the limit checks. :STR = NOCHANGE because at 00:04:00 the nearest quality flag in the RTU data file is NOCHANGE1. :LIMR = GOOD because :RAW passed all the limit checks.

Time 00:09:00

$$goodrep = 2 + 1 (6 – (0.5 / 2)) + 2.91 • 0.5 = 9.21$$

$$badrep = 1000$$

$$nochrep = 6 + 1 (0) = 6.00$$

$$manrep = 50 + 1 (11 – (0.5 / 2)) = 60.75$$

$$:EREP = min (goodrep, badrep, nochrep, manrep) = 6.00$$

Therefore, the NOCHANGE :UDT, :FILP, and :FILT are used and :ST = NOCHANGE. :LIM = GOOD because :FILT passed all the limit checks. :STR = NOCHANGE because at 00:09:00 the nearest quality flag in the RTU data file is NOCHANGE1. :LIMR = GOOD because :RAW passed all the limit checks.

Time 00:15:30

$$goodrep = 2 + 1 (10.5 – (0.5 / 2)) + 2.91 • 0.5 = 13.71$$

$$badrep = 1000$$

$$nochrep = 6 + 1 (2.5) = 8.25$$

$$manrep = 50 + 1 (16 – (0.5 / 2)) = 65.75$$

$$:EREP = min (goodrep, badrep, nohcrep, manrep) = 8.25$$

Therefore, the NOCHANGE :UDT, :FILP, and :FILT are used and :ST = N0 CHANGE. :LIM = GOOD because :FILT passed all the limit checks. :STR = BAD because the nearest quality flag in the RTU data file at 00:15:30 is BAD. :LIMR = GOOD because :RAW passed the limit checks.

Time 00:20:00

$$goodrep = 2 + 1 (6 – (0.5 / 2)) + 2.91 • 0.5 = 9.21$$

$$badrep = 1000$$

$$nochrep = 6 + 1 (7 – (0.5 / 2)) = 12.75$$

$$manrep = 50 + 1 (0) = 50$$

$$:EREP = 1000 \text{ since the GOOD } :UDT > TOUT$$

Therefore, :ST =BAD because it has timed out. :LIM = GOOD because :FILT passed all of the limit checks. :STR = MANUAL because the nearest quality flag in the RTU data file at 00:20:00 is MANUAL. :LIMR = GOOD because :RAW has passed all of the limit checks.

DRIVES keyword

The DRIVES keyword defines the value that the SCADA device controls. The controlled device is updated at the beginning of every time step as follows:

$$dr = :VAL + :AC$$

Then, during the time step, State Estimation calculates and applies the additive repeatability correction :RC, for monitor pressures, SALE/TAKE flows, and HF flows.

At the end of every time step, the accuracy correction :AC is "nudged" by autocalibration for monitor pressures, SALE/TAKE flows, and HF flows. The :AC term can be set manually regardless of the device the SCADA device is controlling.

See "Solution technique for online modeling" on page 745 for more details.

Example input

Example 1

The following example shows the SCADA elements for three pressure values.

```
DEFINE TIMEOUT = 7.5  /* Time to wait before pronouncing a data

SCADA  STA100_PD,  DR S100_DISC_MON:SP,
+ I.MIN=1,   I.MAX=1200,   O.MIN=1,   O.MAX=1200,
+ B.MIN=1,   B.MAX=1200,   ACC=12.,    REP=3.,
+ AUTO.PER=43200,   R.D.R=2,   B.REP=750,
+  TIME.OUT.PER=TIMEOUT

SCADA  STA110_PD,  DR S110_DISC_MON:SP,
+ I.MIN=1,   I.MAX=1200,   O.MIN=1,   O.MAX=1200,
+ B.MIN=1,   B.MAX=1200,   ACC=12.,   REP=3.,
+ AUTO.PER=43200,   R.D.R=2,   B.REP=750,
+ TIME.OUT.PER=TIMEOUT

SCADA  STA120_PD,  DR S120_DISC_MON:SP,
+ I.MIN=1,   I.MAX=1200,   O.MIN=1,   O.MAX=1200,
+ B.MIN=1,   B.MAX=1200,   ACC=24.,   REP=6.,
+ AUTO.PER=43200,   R.D.R=2,   B.REP=750,
+ TIME.OUT.PER=TIMEOUT
```

Example 2

The following example functions exactly like "Example 1" above, yet makes use of a DEFINE command and a macro so that repetitive data may be defined once for all like points.

```
DEFINE TIMEOUT = 7.5  /* Time to wait before pronouncing a data
/* outage for a single point
/* Macro to use with all pressure SCADA points
MACRO(PLIMITS,
+ I.MIN=1,   I.MAX=1200,   O.MIN=1,   O.MAX=1200,
+ B.MIN=1,   B.MAX=1200,   ACC=12.,   REP=3.,
+ AUTO.PER=43200,  R.D.R=2,    B.REP=750,
+ TIME.OUT.PER=TIMEOUT)

/* SCADA element
SCADA  STA100_PD,  PLIMITS,  DR S100_DISC_MON:SP
SCADA  STA110_PD,  PLIMITS,  DR S110_DISC_MON:SP
SCADA  STA120_PD,  PLIMITS,  DR S120_DISC_MON:SP,
```

```
+ ACC = 24, REP = 6.0
```

Note that keywords specified in macros may be superseded after invocation of the macro. In example 2, the SCADA point STA120_PD uses accuracy and repeatability values that differ from those specified in the PLIMITS macro. Because ACC=24 and REP=6.0 is specified after PLIMITS, this SCADA element will be reassigned the new ACC and REP values.

Example 3

The following example demonstrates how SCADA element keywords may be set to expressions in a macro.

```
/* Define the macro
MACRO(QLIMITS(PVID),
+   I.MIN =0,   I.MAX=1050,   O.MIN=0,   O.MAX=1050,
+   B.MIN=0,    B.MAX=1050,   B.REP=200.,  AUTO.PER=2880.,
+   TIME.OUT.PER=60.,
+   ACC   = MIN( 0.10 * ABS(PVID:VAL) , 20 ),
+   REP   =    ( 0.05 * ABS(PVID:VAL) ),
+   R.D.R = ( ( ABS(PVID:VAL) > 0) ? 1.0 : 0.0) )
/* Define SCADA points to be used
SCADA  METER123_Q,  QLIMITS(METER123_Q),  DR = MTR123:SNQ
SCADA  METER456_Q,  QLIMITS(METER456_Q),  DR = MTR456:SNQ
SCADA  METER234_Q,  QLIMITS(METER234_Q),  DR = MTR234:SNQ
```

Slightly compressible liquid equation of state (STATE SCL)

INPREP

```
STATE SCL[PROP] ...
+ FLUID FNAME
[+ ...]
[+ DENSITY.ERROR DENERR]
[+ BULKMOD.ERROR PMODERR]
[+ VISCOSITY.ERROR VERR]
[+ BFC BFC]
```

Field	Units Key	Description
FNAME	n/a	Name for this fluid.
DENERR	DENSITY	Density error for each fluid.
PMODERR	n/a	Bulk modulus error as a percentage of bulk modulus, 0-100. {20}
VERR	n/a	Viscosity error as a percentage of viscosity, 0-100. {20}
BFC	n/a	Batch friction correction. {0}

Description

This section supplements the documentation for using SCL and SCLPROP for an offline model. For more information see “STATE SCL” on page 230. This section documents the fields on the SCL and SCLPROP commands that are applicable when using the online products.

Take note

Friction correction

Inside a pipe, the effective friction factor, FFE, is:

FFE = :FF · (1 + :FC) · (1 + :BFC)

where:

:FC = pipe friction.correction, :FC from the pipe
:BFC = batch friction.correction, :FC from the batch
:FF = calculated friction factor

Note: FFE and FF are peekable for each pipe; the average displays because they both vary along the pipe.

The batch friction correction for each batch is bounded by BFC.BOUND. The pipe friction correction for each pipe is bounded by PFC.BOUND. These two bounds are available from the “show online” display.

-BFC.BOUND ≤ batch:BFC ≤ +BFC.BOUND

-PFC.BOUND ≤ pipe:FC ≤ +PFC.BOUND

batch:BFC and pipe:FC are both pokable, and they are both automatically adjusted (time scales are ACP.BFRIC and ACP.PFRIC, respectively) so as to reduce PDF.

Density error

For SCL, the DENSITY.ERROR is maintained by fluid, and should be set to the density variation expected within a batch of that fluid.

For SCLPROP, the DENSITY.ERROR is maintained both by fluid and by FMV. The DENSITY.ERROR for the fluid should be set to the density variation expected within a batch of that fluid. This value is used when the batch line up is set using a LINE.FILL command. If a composition controller is driving DENSITY, the DENSITY.ERROR for the FMV is set to the effective repeatability of the density measurement. A Property element can control the DENSITY.ERROR directly. The DENSITY.ERROR set from either a composition controller or a Property element will typically be much smaller than the DENSITY.ERROR for the fluid as a whole.

Example input

Specify the SCL equation of state input for two different crude oils:

```
STATE  SCL
+ FLUID  CRUDE1
+ DENSITY  14.696  60  58.2  270000
+ DENS.ERR  2
+ VISC      3.78
+ VISC.ERR 25
+ FLUID  CRUDE2
+ DENSITY  14.696  59  59.1  290200
+ DENS.ERR  1
+ VISC      4.01
+ VISC.ERR 15
```

General pipe - transient (GP) for online modeling

INPREP

```
GP ...
+ ELEV.ERROR EE
+ SLACK.VOLUME.BOUND SVB
```

Field	Units Key	Description
EE	ELEVATION	The maximum vertical distance that batch interfaces could be misaligned. {30 feet}
SVB	VOLUME	Volume of diagnostic flow to be excused due to inaccuracies in modeling slack line flow. {0}

Description

This documentation supplements the documentation for the transient general pipe (GP) element used in offline simulations. See “General pipe - transient (GP)” on page 261 for details of the offline portions of this command.

Transfer line - transient (T) for online modeling

INPREP

```
T ...
+ ELEV.ERROR EE
+ SLACK.VOLUME.BOUND SVB
```

Field	Units Key	Description
EE	ELEVATION	The maximum vertical distance that batch interfaces could be misaligned. {30 feet}
SVB	VOLUME	Volume of diagnostic flow to be excused due to inaccuracies in modeling slack line flow. {0}

Description

This documentation supplements the documentation for the transfer line (T) element used in offline simulations. See "Transfer line - transient (T)" on page 263 for details of the offline portions of this command.

Take note

Pipe DRA degradation rate

DRA degradation in pipes is simulated using a first-order decay equation of the following form:

$$\frac{dC}{dx} = -kC$$

where:

C	=	DRA concentration
k	=	DRA degradation coefficient (in units of 1/distance units)
x	=	Distance

If C_o is DRA concentration at the beginning of a pipe of length L, then the DRA concentration (C) at the end of the pipe can be obtained by integrating the above equation:

$$C = C_o e^{-kL}$$

The corresponding fractional degradation (f) is given by the following equation:

$$f = 1 - e^{-kL}$$

The following graph shows DRA destruction coefficients:

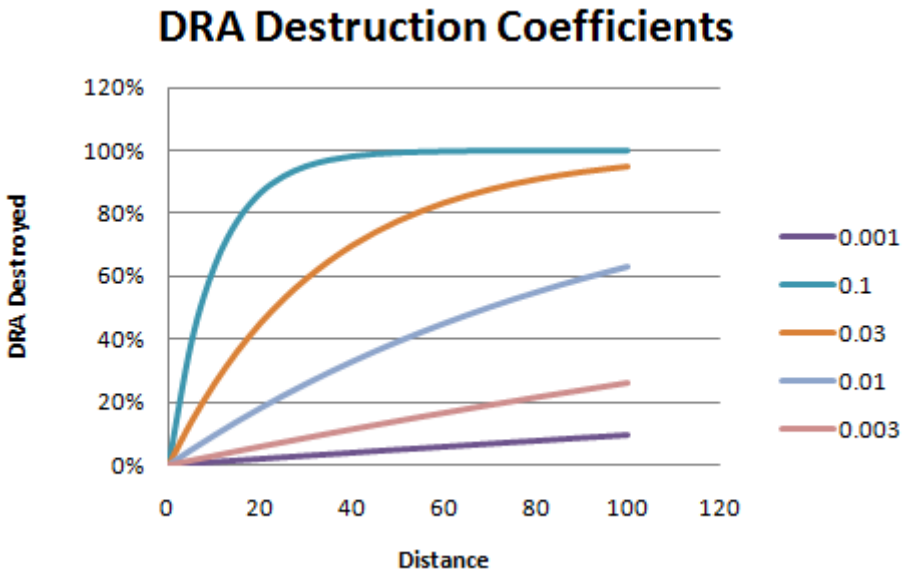

DRA Destruction Coefficients
120%
100%
80%
60%
40%
20%
0%
DRA Destroyed
0
20
40
60
80
100
120
Distance
0.001
0.1
0.03
0.01
0.003

Figure 17-1

INTRAN file input for online modeling

When creating an INTRAN file for online modeling, you will need to add or modify the INTRAN files used for offline modeling. See "Differences between off-line and online models" on page 744 for more information on the modification required. This section contains commands and additional considerations for using commands in online modeling.

- "GENERATE.SCHEDULE" on page 814
- "SELECT (INTRAN)" on page 517

Also see "LOAD.STATUS overrides for online modeling" on page 816.

GENERATE.SCHEDULE

INTRAN

```
GENERATE.SCHEDULE name1 name2 name3 ...,
[+ FILE = filename,]
[+ DTMAX = num,]
[+ MAXRECS = mr]
```

Field	Units Key	Description
$name_i$	n/a	List of peek names to be written to RTU data file.
filename	n/a	Name of RTU data file.
num	minutes	Maximum time between points. {no maximum}
mr	n/a	Maximum number of records in the RTU data file.

Description

GENERATE.SCHEDULE is used to create an RTU data file containing simulated data for testing purposes. Typically, each $NAME_i$ is a DEFINE corresponding to a SCADA identifier.

Take note

Data compression

Values are written to the RTU data file every time step, but values that are redundant because they lie on a line between two adjacent points are not written. You may force output times not to exceed a certain time interval by specifying DTMAX.

Note: The DTMAX specified with GENERATE.SCHEDULE does not limit the simulation time step

Example input

Write data information for PUMP1A:P-, PUMP1C:P+, PUMP2A:P-, PUMP2C:P+, PIPE10:P+, PIPE1:P-, PIPE10:Q+, PIPE1:Q-, and the output of density sensors: PIPEIN.DENS, PIPE6.IN.DENS, and PIPE10.OUT.DENS. The data gets written to an RTU file name RTUDATA.DT. There is no maximum interval:

```
/*
/* Define SCADA pressures
DEFINE INLET.P.STA1  =  PUMP1A:P-
DEFINE DISC.P.STA1   =  PUMP1C:P+
DEFINE SUCT.P.STA2   =  PUMP2A:P-
DEFINE DISC.P.STA2   =  PUMP2C:P+
DEFINE DEL.PRES.STA6 =  PIPE10:P+
DEFINE BEG.PRES.STA0 =  PIPE1:P-
DEFINE DEN.STA1 = PIPE1.IN.DENS:OUT
DEFINE DEN.STA4 = PIPE6.IN.DENS:OUT
DEFINE DEN.STA6 = PIPE10.OUT.DENS:OUT
/*
/* Define SCADA Flows
/*
```

```
DEFINE DEL.FLOW     = PIPE10:Q+
DEFINE INLET.FLOW   = PIPE1:Q-
/*
/*  Generate RTU data file
/*
GENERATE.SCHEDULE INLET.P.STA1  DISC.P.STA1 DEL.PRES.STA6
+ DEN.STA1  DEN.STA4
+ DEN.STA6  DEL.FLOW  DISC.P.STA2 SUCT.P.STA2 INLET.FLOW
+ BEG.PRES.STA0,
+ FILE=RTUDATA.DT
```

Interactive commands for online modeling

The same interactive commands available for offline modeling are also used for online modeling. This section provides supplemental information use of some interactive commands.

Distance plot items for online modeling

For more information on distance plots, see "Distance plots" on page 169 and "DISTPLOT" on page 548.

Distance plot item	Description
LDT(i)	Leak Detection Threshold.
BFC	Batch friction correction (by label).
BULKMOD.ERR	Bulk modulus error (by FMV).
DENSITY.ERROR	Density error (by FMV).
EFF.HEAD	Head with respect to PDF.
EFF.VPHEAD	Head with respect to vapor pressure and PDF.
FRIC.PRES.DROP	Frictional pressure drop.
GRAV.PRES.DROP	Gravitational pressure drop.
INCR.VELOCITY	Shift rate (velocity units) due to gravitational signature of misaligned density.
USD0	Density at custody conditions as a function of Distance curve based on densitometer data and upstream pipe flow rate. Use to compare with UMD0 and EMD0. (IA)
USV	Sonic Velocity at custody conditions as a function of Distance curve based on sonic velocity data and upstream pipe flow rate. Use to compare with UMV and EMV. (IA)
DSD0	Density at custody conditions as a function of Distance curve based on densitometer data and downstream flow rate. Use to compare with DMD0 to locate model batches that arrive early. (IA)
DSV	Sonic velocity at custody conditions as a function of Distance curve based on sonic velocity data and downstream flow rate. Use to compare with DMV to locate model batches that arrive early. (IA)
UMD0	Density at custody conditions as a function of Distance upstream of the IA element based on model density.

Distance plot item	Description
UMV	Sonic velocity at custody conditions as a function of Distance upstream of the IA element based on model sonic velocity.
EMD0	Density at custody conditions as a function of Distance downstream of the IA element based on node density at the IA element and the upstream pipe flow rate. Used to determine if the fluid that has left the system via an external contained a batch interface in USD0.
EMV	Sonic velocity at custody conditions as a function of Distance downstream of the IA element based on node sonic velocity at the IA element and the upstream pipe flow rate. Used to determine if the fluid that has left the system via an external contained a batch interface in USV.
DMD0	Density at custody conditions as a function of Distance downstream of the IA element based on model density.
DMV	Sonic velocity at custody conditions as a function of Distance downstream of the IA element based on model sonic velocity.
UP.MATCH	The error function $\Delta(\lambda)$. This is a measure of how closely the curve USD0/USV matches with the model density/sonic velocity represented by the concatenated curves, UMD0/UMV and EMD0/EMV. The low point of the curve represents the shift volume (λ) required to bring the model property into alignment with the real interface location.
DN.MATCH	The error function $\Delta(\lambda)$. This is a measure of how closely the curve DSD0/DSV matches with the model property curve DMD0/DMV. The low point of the curve represents the shift volume (λ) required to bring the model property into alignment with the real interface location.
ALIGN.FLOW	Batch alignment flow rate at pipeline conditions. (IA)
ALIGN.VOLUME	Accumulated batch alignment volume. (IA)
INCR.PROP.FLOW	Shift rate in flow units. (IA)
INCR.PROP.VELO	Shift rate in velocity units. (IA)
SONIC.VELOCITY	Sonic velocity.
VISCOSITY.ERROR	Viscosity error.

LOAD.STATUS overrides for online modeling

The following information from the archive or status file is used for each element and overrides information specified in the INPREP file. Only online elements are discussed here; see "LOAD.STATUS" on page 471 for off-line elements:

SCADA	input minimum, input maximum, output minimum, output maximum, rate bound, time average period, maximum accuracy correction, repeatability, time tag error bound, accuracy correction, repeatability correction, effective repeatability, limit, mode, off value, bad minimum, bad maximum, bad rate bound, bad repeatability, raw value, use value, repeatability decay rate, time out period, autocalibration period, fore, raw limit, raw status, off repeatability, manual repeatability, no change repeatability, scan period, status, second difference bound, recovery time, time difference between simulation and SCADA, recover time
IA	shift velocity limit, model density, model sonic velocity, set point density, set point sonic velocity, mode, density difference, sonic velocity difference, average derivative of density with respect to volume, average derivative of sonic velocity with respect to volume, maximum value of the error function, multiplier used in calculating the error function, multiplier used in calculating the error function for sonic velocity, multiplier used in calculating the error function, multiplier used in calculating the error function for sonic velocity, maximum ratio of minimum value of the error function, maximum ratio of minimum value of the error function, flow low limit, remaining shift volume, fraction of remaining shift, cumulative change in flow upstream, cumulative change in flow downstream
Composition controller	If SCL fluid and fluid fraction. If SCLPROP, pressure, temperature, density, bulk modulus, temperature modulus, pressure temperature multiplier, pressure multiplier, viscosity, interpolation value, sonic velocity, pressure weight, temperature weight, density weight, bulk modulus weight, temperature modulus weight, PTMULT weight, PPMULT weight, viscosity weight, VPMI weight, VTMI weight, interpolation weight, sonic velocity weight, base fluid
Monitors	current value of pressure, temperature, for BWRS or SCL fluid set points, number of control mode changes, the pressure set point is changed to the pressure value
Externals/ Nodes	current value of flow, temperature, set point pressure or flow, temperature set point, for BWRS or SCL fluid set points, number of control mode changes, the flow set point is set to the current flow value

MONFLOWS for online modeling

MONFLOWS is a built-in device for online models that calculates various useful items that help the user evaluate the overall health of the model. It calculates the sum of the monitor flows, the sum of the absolute value of the monitor flows, and a few other items of interest. See "SHOW" on page 559 for information on how to display monflow attributes. See "Monitor flows (MONFLOWS) (MF) attributes" on page 848 for descriptions of monflow attributes. See also "STATION for online modeling" on page 817.

STATION for online modeling

For online models, stations have some additional attributes that are useful when tuning and/or evaluating scenarios. The attributes of particular interest to the online modeler are :QAST, :EXCL, :QS. :QAS, and :QAMX. See "STATION" on

page 421 and “Station (ST) attributes” on page 685 for details. See also “MONFLOWS for online modeling” on page 817.

TRANSEXPORT program

TRANSEXPORT is used to send data and/or alarms back to the SCADA system or to any other process or data store. TRANSEXPORT reads the *model*.INEXPT file for a list of model variables and alarms that are to be exported. Upon startup, the list is validated against TRANS shared memory. Any errors are written to the *model*.OUTXPT file.

TRANSEXPORT

```
TRANSEXPORT MODEL [-TIMEOUT = TIME]
```

Field	Units Key	Description
MODEL	n/a	The *model*.INEXPT file contains the various INEXPT commands to configure the data and/or alarms to be exported. See "INEXPT" on page 820.
TIME	n/a	Time out period in seconds. If TRANS does not update shared memory within the time out period, TRANSEXPORT exits. {3600 seconds}

Description

The data and alarm export utility, TRANSEXPORT, is used to send data and/or alarms back to the SCADA system or to any other process or data store. TRANSEXPORT reads the *model*.INEXPT file for a list of model variables and alarms that are to be exported. Upon startup, the list is validated against TRANS shared memory. Any errors are written to the *model*.OUTXPT file.

TRANSEXPORT monitors the status of TRANS, if TRANS is exited TRANSEXPORT exits. If TRANS exits without a user-defined HALT or QUIT, TRANSEXPORT exits after the timeout period.

Take note

File naming requirements

MODEL must be the same name used to start the TRANS process.

Multiple TRANSEXPORT processes

For two or more TRANSEXPORT processes to export data from the same model, the *model*.VYDEF file must be copied to additional VYDEF files, one for each additional TRANSEXPORT process. This copy must be done after TRANS starts but before TRANSEXPORT starts. Launch TRANSEXPORT using the name from the newly copied VYDEF file(s).

API documentation

See the "TRANSEXPORT interface" on page 991 for details on how to interface TRANSEXPORT to the SCADA system or to any other process or data store.

INEXPT

```
[POKE EXPORTDATA.UPDATE = TIME]
EXPORTDATA SCADA_ID = MODEL_VARIABLE
EXPORTALARM SCADA_ID = MODEL_VARIABLE
```

Field	Units Key	Description
TIME	minutes	The period, in real clock time minutes, between data transfers. If no POKE EXPORTDATA.UPDATE appears in the INEXPT file, data transfers will take place following each model calculation interval (time step). All alarmed data is exported at every calculation interval.
SCADA_ID	n/a	The SCADA ID or foreign variable equating to the modeled point that is to be alarmed and/or exported.
MODEL_ VARIABLE	n/a	The source of the information to be alarmed and/or exported.

Description

The INEXPT file is an ASCII file containing a complete description of the model data export process. This file specifies the source (model variable) and destination (for example, SCADA point ID in an online modeling system) for the data transfer, the names of the model alarms for export, and the optional update period for the data transfer.

Example input

The format of the portion of the INEXPT file defining the model variables for export is shown in the following example:

```
POKE EXPORTDATA.UPDATE     =  5.0
EXPORTDATA  BS_TIME        =  MODEL_TIME
EXPORTDATA  BS_BCREEK_P    =  BATTLE_CREEK:P
EXPORTDATA  BS_ALBION_P    =  ALBION:P
EXPORTDATA  BS_T100.1_UF   =  T100.1:Q-
EXPORTDATA  BS_T100.1_DF   =  T100.1:Q+
EXPORTDATA  BS_PAN_PAK     =  PANHANDLE.INV
EXPORTDATA  BS_MUSK_SP     =  MUSKEGON_SP
EXPORTDATA  BS_PLYMO_POS   =  PLYMOUTH:FR
EXPORTDATA  BS_N_HUD_AF    =  NEW_HUDSON.AQ
```

The EXPORTDATA command describes the variable source and the “foreign” variable name of the data transfers. The left side indicates the foreign variable name and the right side indicates the model variable (peek) name. In this example each of the foreign variable names begin with “BS_” indicating “BASIN”. Such a prefix is not required, however, it may be useful as a means of indicating the source of the foreign destination data.

In addition to model peek names, the right side of the EXPORTDATA command can also be an expression. These expressions have the complete flexibility of expressions used in the DEFINE command. With EXPORTDATA expressions, the value exported can be any mathematical, relational or logical combination of any of the standard or defined variables. In this way, values are exported with the computational overhead placed on the TRANSEXPORT process rather than on the TRANS modeling process. Note, however, that if expressions are used in an EXPORTDATA command, the value for that command is not available to its parent model.

Examples of EXPORTDATA expressions are as follows:

```
EXPORTDATA BS_ABC  = T100.1:INV +
+                    T100.2:INV +
+                    T100.3:INV
EXPORTDATA BS_XYZ  = (STA_A_C1:ST == RUNNING) |
+                    (STA_A_C2:ST == RUNNING)
```

In the first example, foreign destination variable name BS_ABC receives a value representing the line pack in a pipeline sub-network containing model pipes T100.1, T100.2, and T100.3.

In the second example BS_XYZ receives a logical with value 0.0 if neither compressor STA_A_C1 nor STA_A_C2 is running or a value of 1.0 if either compressor is running.

Definition of Model Alarms in the INEXPT file

The format of the portion of the INEXPT file for the selection of the model-generated alarms for export is shown in the following example:

```
EXPORTALARM EVENTMAN.WARN  = COMP1.LSP
EXPORTALARM EVENTMAN.WARN  = T100-1.MAOP
EXPORTALARM EVENTMAN.WARN  = T100-2.MAOP
EXPORTALARM EVENTMAN.WARN  = T100-1.LAOP
EXPORTALARM EVENTMAN.WARN  = T100-2.LAOP
EXPORTALARM EVENTMAN.INFO  = ALM.RTU2.P17
```

The EXPORTALARM command specifies the model alarm source, defined in the model's simulation input file, and the foreign alarm string. The left side indicates the string, to be passed to the foreign event manager. This string can be used to assign individual foreign names to each alarm or, as shown, to specify the routing and priority for the alarm messages.

RTUFILT utility

The RTUFILT utility is a data filtering utility that extracts the values of telemetered points from an input RTU data file, performs user-defined calculations with the input points, and writes calculated results to an output RTU data file. This utility is run from the command line. For more information on running RTUFILT from the command line, see "RTUFILT" on page 828.

Operating modes

RTUFILT operates in one of three modes:

- READ
- COLLECTION
- CALCULATE

Upon startup, RTUFILT is in the READ mode, reading data points from the input RTU data file, looking for an INPUT point. As soon as an INPUT point is read, RTUFILT saves its time stamp, then switches to the COLLECTION mode.

In the COLLECTION mode, RTUFILT collects INPUT points for those points whose time stamps are within PERIOD of the saved time stamp. As soon as any point is read (INPUT or non-INPUT) whose time stamp is not within PERIOD of the saved time stamp, RTUFILT switches to the CALCULATE mode.

In the CALCULATE mode, RTUFILT performs the calculations for each CALC point ready for calculation and, if specified, writes the results to an output RTU data file.

Once all the calculations are complete, RTUFILT goes back into READ mode looking for the next INPUT point.

Input files

There are two input files to RTUFILT: an RTU data file and a *model*.INFILT file. The RTU data file is a circular binary file containing the telemetered values. Each point in this binary file contains a SCADA id name, value, quality indicator, and time stamp associated with the point. The *model*.INFILT file is an ASCII text file that contains the description and specification of the input points, calculated points, and output points. For more information, see "The INFILT file" on page 822.

Output files

There are two output files from RTUFILT: *model*.RTUOUT and *model*.OUTFLT. The RTUOUT file is a circular binary file with the same format as the input RTU data file. It contains all of the points in the input RTU data file plus any specified calculated points. The OUTFLT file is an ASCII text file, which echoes the *model*.INFILT file and contains any appropriate error or warning messages.

The INFILT file

The INFILT file is a required text file for the RTUFILT utility. It is used to:

- Specify collection name(s) (COLLECTION command)
- Specify collection period(s) (PERIOD subcommand)
- Specify the points to be read from the input RTU data (INPUT subcommand)

- Construct mathematical expressions to calculate parameters (CALC command)
- Specify calculated values to be written to the output RTU data file (+RTUID for CALC command)

Commands

The following commands and subcommands are supported in the INFILT file:

- "COLLECTION" on page 829
- "PERIOD" on page 830
- "INPUT" on page 831
- "CALC" on page 833

Expression language

INFILT files support the same expression language as PREPR and TRANS, except:

- String expressions are not supported.
- Functions that rely on nodes and/or elements, such as MAXDIFF, INTERNAL, SCRAPER, DENS, are not supported.

See "User-defined Variables, Macros, and Include Files" on page 567 and "Expressions, Operators, and Functions" on page 587.

Data filters

Built-in data filtering capabilities are provided to perform the following data checks for INPUT and specified output points:

- Minimum value (IMIN, OMIN),
- Maximum value (IMAX, OMAX), and
- Maximum rate of change (RB).

In addition, the following data filters are provided for use in expressions:

- HISTORY - store a specific number of historical values for other calculations
- TAYLOR1 - Taylor series for one independent variable
- TAYLOR2 - Taylor series for two independent variables

Example INFILT file

```
/* Purpose:   Input file for RTUFILT program
/*================================================================
/*          INPUT POINTS FROM RTU DATA FILE
/*================================================================
COLLECTION  COL1
{
  PERIOD 10*               Collect points for 10 secs
  INPUT PRES_SC,
  + RTUID = PRES_SC,     IMIN = 0,   IMAX=8000,   RB=1E10
  INPUT TEMP_UP_SC,
```

```
  + RTUID = TEMP_UP_SC, IMIN=-200, IMAX=800,   RB=1E10
  INPUT TEMP_DN_SC,
  + RTUID = TEMP_DN_SC, IMIN=-200, IMAX=800,   RB=1E10
  INPUT FLOW_SC,
  + RTUID = FLOW_SC,    IMIN=  0,  IMAX=1000,  RB=1E10
  INPUT DEN_SC,
  + RTUID = DEN_SC,     IMIN=5001, IMAX=9499,  RB=1E10
  INPUT VISC_SC,
  + RTUID = VISC_SC,    IMIN=  -1, IMAX=1000,  RB=1E10
  INPUT SVEL_SC,
  + RTUID = SVEL_SC,    IMIN= 501, IMAX=1510,  RB=1E10
  INPUT HEAT_ST_SC,
  + RTUID = HEAT_ST_SC, IMIN=   0, IMAX=1,     RB=1E10
  INPUT HEAT_KW_SC,
  + RTUID = HEAT_KW_SC, IMIN= 0,   IMAX=2000,  RB=1E10
}  /* End collection points
/*=================================================================
/*             CONVERT INPUT POINTS TO ACTUAL VALUES
/*=================================================================
CALC PRES,
+ VAL = PRES_SC / 10
CALC TEMP_UP,
+ VAL = TEMP_UP_SC / 10
CALC TEMP_DN,
+ VAL = TEMP_DN_SC / 10,
+ RTUID = TEMP_DN
CALC FLOW,
+ VAL = FLOW_SC / 100
CALC SVEL,
+ VAL = SVEL_SC,
+ RTUID = SVEL
CALC DEN,
+ VAL = DEN_SC / 10,
+ RTUID = DEN
CALC VISC,
+ VAL = (VISC_SC / 10)  + 1.2           /* viscometer is out
CALC HEAT_KW,
+ VAL = HEAT_KW_SC / 100
/*=================================================================
/*                       CALCULATED POINTS
/*=================================================================
/************************ Heat Status ***************************
CALC HS,
+ VAL = HEAT_KW>0,
+ RTUID = HS
/******************** Define stable conditions ******************
CALC HEAT_STABLE,
+ VAL = STDV(HISTORY(HEAT_KW,5)) < 1
CALC DEN_STABLE,
+ VAL = STDV(HISTORY(DEN,30)) < .75,
+ RTUID = DEN_STABLE
CALC DELTA_TEMP_STABLE,
+ VAL = STDV(HISTORY(HEATER_DELTA_TEMP,5)) < 10,
+ RTUID = DELTA_TEMP_STABLE
```

```
CALC DELTA_STABLE,
+ VAL = STDV(HISTORY(HEATER_DELTA_TEMP,5)),
+ RTUID = DT_STABLE
CALC BULK_STABLE,
+ VAL = STDV(HISTORY(BULK_MODULUS2, 15)) < 2000,
+ RTUID = BULK_STABLE
CALC HEATER_DELTA_TEMP,
+ VAL = (TEMP_DN - TEMP_UP),
+ RTUID = HEATER_DELTA_TEMP
CALC PMOD_STABLE,
+ VAL = STDV(HISTORY(PMOD,5)) < 300,
+ RTUID = PMOD_STABLE
/*** Calculate Hot and Cold values for temp, visc, density, bulk.mod ***
CALC HOT_TEMP,
+ VAL = AVG(HISTORY(TEMP_DN,15)),
+ UPDATE.IF (HEAT_FLAG_ON),
+ RTUID=HOT_TEMP
CALC COLD_TEMP,
+ VAL = AVG(HISTORY(TEMP_DN,15)),
+ UPDATE.IF (HEAT_FLAG_OFF),
+ RTUID=COLD_TEMP
CALC HOT_VISC,
+ VAL = AVG(HISTORY(VISC,15)),
+ UPDATE.IF (HEAT_FLAG_ON)
CALC COLD_VISC,
+ VAL = AVG(HISTORY(VISC,15)),
+ UPDATE.IF (HEAT_FLAG_OFF)
CALC HOT_DEN,
+ VAL = AVG(HISTORY(DEN,15)),
+ UPDATE.IF (HEAT_FLAG_ON),
+ RTUID = HOT_DEN
CALC COLD_DEN,
+ VAL = AVG(HISTORY(DEN,15)),
+ UPDATE.IF (HEAT_FLAG_OFF),
+ RTUID = COLD_DEN
CALC HOT_BULKMOD,
+ VAL = AVG(HISTORY(BULK_MODULUS,15)),
+ UPDATE.IF (HEAT_FLAG_ON),
+ RTUID = HOT_BULKMOD
CALC COLD_BULKMOD,
+ VAL = AVG(HISTORY(BULK_MODULUS,15)),
+ UPDATE.IF (HEAT_FLAG_OFF),
+ RTUID = COLD_BULKMOD
/*********** Calculate Density Temperature Modulus ***************
CALC DELTA_TEMP,
+ VAL = ABS(HOT_TEMP - COLD_TEMP),
+ UPDATE.IF (CYCLE_CHECK),
+ RTUID=DELTA
CALC RHO_AVG ,
+ VAL = ((COLD_DEN + HOT_DEN) / 2),
+ UPDATE.IF (CYCLE_CHECK),
+ RTUID = RHO_AVG
CALC DELTA_DEN,
+ VAL = ABS(HOT_DEN - COLD_DEN),
```

```
+ UPDATE.IF (CYCLE_CHECK),
+ RTUID = DELTA_DEN
/* degrees C
CALC DEN_TEMP_MODULUS,
+ VAL = -(DELTA_TEMP * RHO_AVG/DELTA_DEN),
+ UPDATE.IF (CYCLE_CHECK & DELTA_DEN!=0 & DELTA_DEN>5 &
!OUTAGE_CHECK2 & !BATCH_CHECK4),
+ RTUID=NAME_DEN_TEMP_MODULUS
/********* Calculate Viscosity Temperature Modulus Index *********
CALC VISC_AVG ,
+ VAL = ((COLD_VISC + HOT_VISC) / 2),
+ UPDATE.IF (CYCLE_CHECK)
CALC DELTA_VISC,
+ VAL = ABS(HOT_VISC - COLD_VISC),
+ UPDATE.IF (CYCLE_CHECK)
CALC VTMI,
+ VAL = -1/((DELTA_TEMP * VISC_AVG)/DELTA_VISC),
+ UPDATE.IF (CYCLE_CHECK & DELTA_VISC!=0),
+ RTUID = NAME_VTMI
/**************** Calculate mass flow rate ***********************
/* MASS_FLOW in units of kg/sec
/* Flow * 1ft3/7.481 gal * 1m3/35.31466 ft3 * 1 min /60 secs
CALC MASS_FLOW,
+ VAL = (FLOW * 1/7.481 * 1/35.31466 * 1/60 * DEN),
+ UPDATE.IF (CYCLE_CHECK),
+ RTUID=NAME_MASS_FLOW
/***************** Calculate Heat into fluid *********************
/* FLUID_Q in units of cal/sec
/* 1.341 HP/kW * 42.42 BTU/min/HP * min/60 secs * 252 cal/BTU = 238.92 /*
cal/sec
/* Ambient loss factor is 1, therefore multiply expression by 1
CALC FLUID_Q,
+ VAL = (1 * HEAT_KW * 238.92 )
/****************** Calculate Heat Capacity **********************
CALC WDT,
+ VAL = (MASS_FLOW * HEATER_DELTA_TEMP),
+ RTUID = WDT,
+ UPDATE.IF (CYCLE_CHECK)
/* In units of cal/kg*degree K
CALC HEAT_CAP ,
+ VAL = (FLUID_Q / WDT),
+ UPDATE.IF (HEAT_FLAG_ON & CYCLE_CHECK & (WDT > 0 )),
+ RTUID = HEAT_CAP
/***************** Calculate Bulk Modulus **********************
CALC DIV,
+ VAL = (4.168 * HEAT_CAP * (DEN_TEMP_MODULUS** 2)),
+ RTUID = DIV
CALC DIV2 ,
+ VAL = (1/SVEL ** 2) + ((TEMP_DN + 273.15)/DIV),
+ RTUID = DIV2
CALC ISOT_SVEL,
+ VAL = (1 / DIV2) ** 0.5,
+ RTUID=ISOT_SVEL
CALC BULK_MODULUS,
```

```
+ VAL = (DEN * ISOT_SVEL**2/1000),
+ UPDATE.IF (CYCLE_CHECK & !BATCH_CHECK4 & !OUTAGE_CHECK2),
+ RTUID=NAME_BULK_MODULUS
CALC BULK_MODULUS2,
+ VAL = (DEN * ISOT_SVEL**2/1000),
/* + UPDATE.IF (CYCLE_CHECK & !BATCH_CHECK4 & !OUTAGE_CHECK2),
+ RTUID=NAME_BULK_MODULUS2
/**** Calculate Derivative of Bulk Modulus (with respect to TEMP) ****
CALC BULKMOD_TEMPD,
+ VAL = ((BULK_MODULUS - COLD_BULKMOD)/(TEMP_DN - COLD_TEMP) *
HEAT_FLAG_ON) + ((BULK_MODULUS -  HOT_BULKMOD)/(TEMP_DN -
HOT_TEMP)  * HEAT_FLAG_OFF),
+ UPDATE.IF (CYCLE_CHECK & DELTA_TEMP>15),
+ RTUID = BULKMOD_TEMPD
CALC PTMULT,
+ VAL = -1 - (BULKMOD_TEMPD*DEN_TEMP_MODULUS)/MAX(1E5,BULK_MODULUS),
+ RTUID = NAME_PTMULT,
+ UPDATE.IF (CYCLE_CHECK)
```

RTUFILT

```
RTUFILT MODEL
[-INFILT = infilt-file]    /* string
[-OUTFLT = outfilt-file]  /* string
[-RTUIN = rtuin-file]  /* string
[-RTUOUT = rtuout-file]  /* string
[-TBEGIN = begin-time]  /* "YY/MM/DD-hh:mm:ss"
[-TEND = end-time]  /* "YY/MM/DD_hh:mm:ss"
[-MAXPTS = maximum-points-in-rtuout]  /* integer
[-DUMP]   /* dEBUG PRINT
```

Description

The RTUFILT command line utility used for data filtering. It extracts the values of telemetered points from an input RTU data file, performs user-defined calculations with the input points, and writes calculated results to an output RTU data file. For more information, see “RTUFILT utility” on page 822.

User interface and utility debugging

RTUFILT is not an interactive utility and is typically run as a background process. While running, you cannot directly interact with the utility other than to stop it.

A dump flag, DUMP, is provided to allow debugging of output. When DUMP is specified, all CALC points are written to the RTUOUT file. Caution should be exercised when setting DUMP because RTUFILT output can be quite voluminous. For more information, see “RTUFILT” on page 828.

COLLECTION

INFILT

```
COLLECTION NAME
{
  COMMAND1
  COMMAND2
  ...
}
```

Field	Units Key	Description
NAME	n/a	Name of COLLECTION group.
$COMMAND_i$	n/a	PERIOD and INPUT subcommands which are a part of the COLLECTION.

Description

The COLLECTION command is used to group information into dependent groups.

Take note

COLLECTIONs are typically comprised of points from the same field RTU, or points that arrive during the same SCADA poll.

Example input

Define a COLLECTION named Station130 with a collection PERIOD of 10 seconds.

```
COLLECTION STATION130
{
  PERIOD 10

  INPUT STA130_DP, RTUID = STA130DP,
  + I.MIN = 200, I.MAX = 950, RB = 500

  INPUT STA130_IP, RTUID = STA130IP,
  + I.MIN =   0, I.MAX = 600, RB = 100

}  /* End of STA130 collection
```

PERIOD

INFILT

```
PERIOD TPER
```

Field	Units Key	Description
TPER	seconds	Time interval for collecting data.

Description

The PERIOD command specifies the time period for data collection. As soon as any point is READ (INPUT or non-INPUT) whose time stamp is not within TPER of the start time stamp of the collection, the RTUFILT utility goes into CALCULATE mode and performs the specified calculations in each CALC command.

Take note

The PERIOD is typically set to a value less than the SCADA poll interval.

Example input

Set period to 10 seconds.

```
PERIOD 10
```

INPUT

INFILT

```
INPUT NAME,
+ RTUID = SCADA_ID,
[+ IMIN = IMIN,]
[+ IMAX = IMAX,]
[+ RB = RB]
```

Field	Units Key	Description
NAME	n/a	Unique name for the input point (up to 32 characters long).
SCADA_ID	n/a	Name of point in the input RTU data file.
IMIN	n/a	Minimum acceptable value. {no minimum}
IMAX	n/a	Maximum acceptable value. {no maximum}
RB	1/minutes	Maximum allowable change per minute in the value. {no limit}

Description

The INPUT command specifies an INPUT point for RTUFILT.

INPUT point status

INPUT points may have a status value of GOOD or OUT.

As the raw value for each INPUT point is read from the input RTU file, each value is subjected to these three filters:

- A minimum filter (defines the minimum acceptable value)
- A maximum filter (defines the maximum acceptable value)
- A maximum rate of change filter (defines the maximum rate of change with respect to time)

INPUT points are considered to be GOOD if:

- Data is collected during the COLLECTION mode and
- The RAW value passes all three data filters.

Otherwise, INPUT points are considered to be OUT.

:RAW and :VAL values

The value of :RAW is always equal to the most recent value from the input RTU data file. :VAL is equal to :RAW when :ST="GOOD". However, when :ST = “OUT”, the value of :VAL does not change and remains at its previous good value. If no points are available after the RTUFILT utility is started, :VAL is set to (:IMIN + :IMAX) / 2.

Take note

All telemetered values require the use of an input command in RTUFILT calculations.

Peek values are referenced by combining the INPUT point name with the :attribute (e.g. DENSITY:VAL, DENSITY:IMIN, etc.) and can be used in CALC commands. If :attribute is not specified, the expression will use :VAL as a default.

Example input

Set input CORR.DENS equal to the telemetered point EPEOS.DEN.

```
INPUT CORR.DENS, RTUID = EPEOS.DEN,
+ I.MIN = 600, I.MAX = 950, RB = 500
```

CALC

INFILT

```
CALC NAME,
[+ VAL          = EXPR,]
[+ CRITPTS      = CRITLIST,]
[+ UPDATE.IF    = EXPRESSION,]
[+ RTUID        = SCADA_ID,]
[+ O.MIN        = OMIN,]
[+ O.MAX        = OMAX,]
[+ RB           = RB]
```

Field	Units Key	Description
NAME	n/a	Unique name for the calculated point.
EXPR	n/a	A valid expression.
CRILIST	n/a	List of critical points required for calculation to be performed.
SCADA_ID	n/a	A unique name for the RTU identifier in the output RTU data file.
OMIN	n/a	Minimum acceptable output value.
OMAX	n/a	Maximum acceptable output value.
RB	n/a	Maximum allowable change in output value with respect to time (per min).

Description

The CALC command is used to define a calculated point and the calculations to be performed for that point, as well as its output point, if :RTUID is specified.

Take note

A CALC command is required for every calculated point that is to be written to the RTUOUT file.

Peek values are referenced by combining the calculated point name with the :attribute (e.g. BULK.MOD:VAL), and can be used in other INPUT commands or CALC commands. If :attribute is not specified, :VAL will be used.

A CALC point will be written to the RTUOUT file with the time stamp corresponding to the time stamp of the most recent operand.

:OMIN, :OMAX, and :RB, only apply if :RTUID is specified.

The CRITPTS field is used in CALCs that have INPUTs that are not collected during the same PERIOD as the rest of the inputs (this routine is common with SCADA systems that employ Report by Exception). In the following example, you may want to define B as a critical point because it carries a much greater weight in the determination of C than A.

$$C = A + B$$

where:

A ranges from 0.01-0.02, point collected every 10 seconds.

B ranges from 500-1000, point collected every 60 seconds.

If B is defined as a critical point, C will be calculated every 60 seconds. Without defining B as a critical point, C would be calculated every 10 seconds.

If no critical points are specified, then all points in the CALC are defined as critical and the CALC will be performed as soon as any point comes in. For this reason, C will be calculated every 10 seconds.

If a CALC is comprised of points from multiple COLLECTIONs, than the CALC will be executed as often as one of its INPUT points is collected, regardless of which COLLECTION the point came from. Consequently, the previous example would have the same outcome even if points A and B came from different COLLECTIONs.

Example input

Define a CALC to be used in other CALCs for converting ft^3 to m^3 ($1m^3/35.31466ft^3$).

```
CALC VOL_CONV,
+ VAL = 1/35.31466
```

Define a CALC based on the standard deviation of TEMP_DN over its last five points. TEMP_STABLE:VAL is set equal to 1 if the statement is true or 0 if it is false.

```
CALC TEMP_STABLE,
+ VAL = STDV(HISTORY(TEMP_DN,5)) < 1
```

Calculate the bulk modulus of a fluid using an INPUT defined as DENSITY. BULK.MOD will only be calculated if CYCLE_CHECK is true (equal to 1) and DENSITY was collected during the last collection period.

```
CALC BULK.MOD,
+ VAL = DENSITY * ( ISOT.SONIC.VELO:RAW ** 2 ),
+ CRITPTS = ( DENSITY),
+ UPDATE.IF (CYCLE_CHECK)
```

Peek and poke keyletters and attributes for online modeling

These peek and poke keyletters and attributes for online modeling supplement the main list. See "Peek and Poke Keyletters and Attributes" on page 639.

CALC attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
VAL	No	none	Current value for expression.
DFLT	Yes	none	Default value.
OMIN	Yes	none	Minimum acceptable value for output.
OMAX	Yes	none	Maximum acceptable value for output.
RB	Yes	none	Maximum acceptable rate of change for output.
ID	No	none	SCADA identifier.
ID	No	none	SCADA identifier.

Composition controller (CO) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
EXT	No	none	Name of monitor external.
FLU	Yes	none	Name of the SCL Fluid at the external.
ID1	Yes	none	Fluid identifier.
ID2	Yes	none	Fluid Identifier.
ST	No	none	Status of the FMV at pipe end. Descriptions include: {Corresponding integer value} • GOOD—FMV at pipe end is good. {1} • BAD—FMV at pipe end is bad. {2}
P	No	PRESSURE	Pressure.
WP	Yes	none	Pressure weight parameter (0-1).
DP	No	PRESSURE	Default Pressure.
MP	Yes	PRESSURE	Measured Pressure.
CP	No	none	Name of the SCADA element controlling pressure.
T	No	TEMP	Temperature.

Attribute	Pokable	Units	Description
WT	Yes	none	Temperature weight parameter (0-1).
DT	No	TEMP	Default temperature.
MT	Yes	TEMP	Measured temperature.
CT	No	none	Name of SCADA element controlling temperature.
D0	No	DENSITY	Base density at the external.
WD0[1]	Yes	none	Density weight parameter (0-1).
DD0	No	DENSITY	Default base density (from +FLUID line).
MD0[1]	Yes	DENSITY	Measured base density.
CD0	No	none	Name of the SCADA element controlling density.
BM	No	BULK.MOD	Bulk Modulus at the external.
WBM	Yes	none	Bulk Modulus weight parameter (0-1).
DBM	No	BULK.MOD	Default bulk modulus (from +FLUID line).
MBM	Yes	BULK.MOD	Measured bulk modulus.
CBM	No	none	Name of the SCADA element controlling bulk modulus.
TM	No	ΔTEMP	Temperature modulus at the external.
WTM	Yes	none	Temperature modulus weight parameter (0-1).
DTM	No	ΔTEMP	Default temperature modulus (from +FLUID line).
MTM	Yes	TEMP	Measured temperature modulus.
CTM	No	none	Name of SCADA element controlling temperature modulus.
PT	No	none	PTMULT for the SCL equation of state.
WPT	Yes	none	PTMULT weight parameter (0-1).
DPT	No	none	Default PTMULT (from +FLUID line).
MPT	Yes	none	Measured PTMULT.
CPT	No	none	Name of the SCADA element controlling PTMULT.
PP	No	none	PPMULT for the SCL equation of state.
WPP	Yes	none	PPMULT weight parameter (0-1).
DPP	No	none	Default PPMULT (from +FLUID line).
MPP	Yes	none	Measured PPMULT.
CPP	No	none	Name of the SCADA element controlling PPMULT.
V0	No	VISCOSITY	Absolute viscosity of the fluid at :MP and :MT.
WV0[1]	Yes	none	Viscosity weight parameter (0-1).
DV0	No	VISCOSITY	Default viscosity (from +VISC line).
MV0	Yes	VISCOSITY	Measured viscosity.
CV0	No	none	Name of the SCADA element controlling viscosity.

Attribute	Pokable	Units	Description
VP	No	1/ΔP	Viscosity Pressure Multiplier Index (VPMI).
WVP	Yes	none	VPMI weight parameter (0-1).
DVP	No	1/ΔP	Default VPMI (from +VISC line).
MVP	Yes	1/ΔP	Measured VPMI.
CVP	No	none	Name of the SCADA element controlling VPMI.
VT	No	1/ΔT	Viscosity Temperature Multiplier Index (VTMI).
WVT	Yes	none	VTMI weight parameter (0-1).
DVT	No	1/ΔT	Default VTMI (from +VISC line).
MVT	Yes	1/ΔT	Measured VTMI.
CVT	No	none	Name of the SCADA element controlling VTMI.
INT	No	none	Interpolation value.
WINT	Yes	none	Interpolation value weight.
MINT	Yes	none	Measured interpolation value.
CINT	No	none	Name of SCADA element controlling INT.
SVL	No	VELOCITY	Name of the SCADA element controlling sonic velocity.
WSVL	Yes	none	Sonic velocity weight parameter (0-1).
DSVL	No	VELOCITY	Default sonic velocity (from Bulk Modulus on +Fluid line).
MSVL	Yes	VELOCITY	Measured sonic velocity.
CSVL	No	none	Name of the SCADA element controlling sonic velocity.

Header flow (HF) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P±	No	PRESSURE	Pressure at the upstream (-) and downstream (+) connections.
T±	No	TEMP	Temperature at the upstream (-) and downstream (+) connections.
Q±	Yes	FLOW	Flow at the upstream (-) and downstream (+) connections, standard conditions.
PD	No	ΔPRESSURE	Pressure drop across the element.
CF	Yes	VOLUME	Cumulative flow through the element.
F±	No	AFLOW	Flow at the upstream (-) and downstream (+) connections, pipeline conditions.
SQ	Yes	FLOW	Set point flow.

Attribute	Pokable	Units	Description
CB	No	none	Controlling SCADA element.
JWT1	Yes	none	Weight for Goal 1 equation.
JQ1	Yes	none	Coefficient for the flow term in Goal 1.
JPS1	Yes	none	Coefficient for the P- term in Goal 1.
JPD1	Yes	none	Coefficient for the P+ term in Goal 1.
JRHS1	Yes	none	Value for the right side of Goal 1.
JWT2	Yes	none	Weight for Goal 1 equation.
JQ2	Yes	none	Coefficient for the flow term in Goal 2.
JPS2	Yes	none	Coefficient for the P- term in Goal 2.
JPD2	Yes	none	Coefficient for the P+ term in Goal 2.
JRHS2	Yes	none	Value for the right side of Goal 2.

Interface alignment (IA) attributes

Note: Many peeks are not computed (show 0) unless the value is required for shift purpose.

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
CB	No	none	Name of SCADA element controlling this element.
NN	No	none	Connection node.
SNU	No	none	Upstream shift node.
SND	No	none	Downstream shift node.
MATU	No	VOLUME	Upstream pipe volume. The IA element will attempt to find a modeled batch interface over this volume that matches the batch interface in the signal range.
MATD	No	VOLUME	Downstream pipe volume. The IA element will attempt to find a modeled batch interface over this volume that matches the batch interface in the signal range.
SIG	No	VOLUME	Volume of fluid downstream of the property meter for which property meter data is retained.
MD0	No	DENSITY	Model density at the IA element.
MV	No	VELOCITY	Model sonic velocity at the IA element.
SD0	Yes	DENSITY	Set point density at the IA element, usually the controlling SCADA element density. {0}
SV	Yes	VELOCITY	Set point sonic velocity at the IA element, usually the controlling SCADA element sonic velocity. {0}

Attribute	Pokable	Units	Description	
CDFU	Yes	FLOW	Cumulative change in flow rate of fluid properties, upstream end.	
CDFD	Yes	FLOW	Cumulative change in flow rate of fluid properties, downstream end.	
MODE	Yes	none	Switch used to disable or enable the element. {OFF}	
ST	No	none	Status of the IA element. IA status descriptions include: {Corresponding integer values}	
			OFF	:MODE has been poked to OFF. {1}
			ON	If :MODE has been poked to ON, the MODE becomes one of the other calculated modes, depending on model state.
			RUNNING	Statefinder is trying to locate misaligned interfaces. {2}
			SHIFTING	Statefinder is shifting an interface after recognizing a batch interface misalignment. {3}
			PROF MISMATCH	Statefinder has determined that the MISMATCH minimum $\Delta(\lambda)$ from the EMD0/EMV or DMD0/DMV curve was greater than the resultant $\Delta(\lambda)$ from the SCADA data match with a constant property profile *or* :DSF was greater than :DSFTOL. This status can only occur if the model interface leads the SCADA interface. {4}
			LOW_FLOW	Statefinder has determined that the pipeline is being shut in. {5}
			EXT_FLOW	Statefinder has determined that an external within the shift range is flowing. {6}
			REV_FLOW	Statefinder has determined that flow has reversed direction. {7}
			BAD_VALUE	Statefinder has determined that property meter data is bad. {8}
			NOT_FULL	Statefinder has determined that one of three property curves is not full of data. {9}

Attribute	Pokable	Units	Description
DDEN	No	DENSITY	Difference between the largest density value and the smallest density value within the signal range.
DVEL	No	VELOCITY	Difference between the largest sonic velocity value and the smallest sonic velocity value within the signal range.
DDENTOL	Yes	DENSITY	Minimum allowable :DDEN required to indicate a batch interface is within the signal range. {0.5 lb/ft^3}
DVELTOL	Yes	VELOCITY	Minimum allowable :DVEL required to indicate a batch interface is within the signal range. {40 ft/sec}
DDENV	No	DENSITY/ VOLUME	Average derivative of density with respect to volume in the signal range.
DVELV	No	VELOCITY/ VOLUME	Average derivative of sonic velocity with respect to volume in the signal range.
DDENVTOL	Yes	DENSITY/ VOLUME	Minimum allowable :DDENV required to indicate a batch interface in the signal range. {0.001 lb/ft^3/MB}
DVELVTOL	Yes	VELOCITY/ VOLUME	Minimum allowable :DVELV required to indicate a batch interface in the signal range. {0.08 ft/sec/MB}
MINU	No	DENSITY	Minimum value of the error function $\Delta(\lambda)$ in UP.MATCH.
MIND	No	DENSITY	Minimum value of the error function $\Delta(\lambda)$ in DN.MATCH.
MINTOL	Yes	none	Max value of $\Delta(\lambda)$ to accept as a valid match of interfaces. The absolute minimum must be less than :MINTOL. {$+\infty$}.
R1U	No	none	Ratio of the minimum value of the error function $\Delta(\lambda)$, to the next smallest local minimum of the function in UP.MATCH.
R1D	No	none	Ratio of the minimum value of the error function. $\Delta(\lambda)$, to the next smallest local minimum of the function in DN.MATCH.
R1TOL	Yes	none	Maximum allowable :R1U or :R1D that will allow the software to claim a match of interfaces. {0.50}
R2U	No	none	Ratio of the minimum value of the error function $\Delta(\lambda)$, to the error associated with comparing USD0/USV data with a match range made up of a single fluid of constant density/ sonic velocity (average density/sonic velocity of UMD0/UMV + EMD0/EMV).
R2D	No	none	Ratio of the minimum value of the error function $\Delta(\lambda)$, to the error associated with comparing DSD0/DSV data with a match range made up of a single fluid of constant density/ sonic velocity (average density/sonic velocity of DMD0).
R2TOL	Yes	none	Maximum allowable :R2U or :R2D that will allow the software to claim a match of interfaces. {0.20}

Attribute	Pokable	Units	Description
SVU	No	none	Volume of fluid that exists between the model's interface in UMD0/UMV and EMD0/EMV and the upstream set point interface (generally SCADA) seen in USD0/USV. This value becomes non-zero when shifting begins, and decreases its value each step that shifting occurs until the value reaches zero, and shifting is complete.
SVD	No	none	Volume of fluid that exists between the model's interface in DMD0/DMV and the downstream set point interface (generally SCADA) seen in DSD0/DSV. This value becomes non-zero when shifting begins, and decreases its value each step that shifting occurs until the value reaches zero, and shifting is complete.
SIWITHIN	No	none	Test to determine whether an interface has fully passed through the signal range. Positive value indicates that a SCADA interface is now fully within the signal range. A negative value indicates that the recent data still shows significant variation. Zero indicates that :SIWITHIN was not calculated during the time step. :SIWITHIN = [1 - (4R / T)], where R is the variation in the most recent one-third of the signal range property data, and T is the variation in all of the signal range.
VELLIM	Yes	none	Shift velocity limit as a fraction of model velocity (:V). Valid range = (0.01-1.00). {1.00}
DENWT	Yes	none	Multiplier applied to the difference between density in the signal range and density in the match range when calculating the error, $\Delta(\lambda)$, in the match algorithm. {10 ft^3/lb}
VELWT	Yes	none	Multiplier applied to the difference between sonic velocity in the signal range and sonic velocity in the match range when calculating the error, $\Delta(\lambda)$, in the match algorithm. {0.125 sec/ft}
DDENWT	Yes	none	Multiplier applied to the difference between the change in SCADA density with respect to volume in the signal range and the change in model density in the match range when calculating the error, $\Delta(\lambda)$, in the match algorithm. {0.1 MB/ (lb/ft^3)}
DVELWT	Yes	none	Multiplier applied to the difference between the change in SCADA sonic velocity with respect to volume in the signal range and the change in model sonic velocity in the match range when calculating the error, $\Delta(\lambda)$, in the match algorithm. {0.001 MB/(ft/sec)}

Attribute	Pokable	Units	Description
QLOLIM	Yes	FLOW	Flow low limit at the IA element. Whenever :Q < :QLOLIM, batch interface alignment will be disabled, and :ST will equal LOW_FLOW. {0}
Q	No	FLOW	Flow at the IA element. It is equated to the flow out of the last pipe in the upstream shift range.
DSFTOL	Yes	none	Maximum allowable :DSF. Whenever :DSF ≥ :DSFTOL, :ST will equal PROF_MISMATCH.
DSF	No	none	Difference between Δ(λ) from the DSD0/DSV and DMD0/DMV match and Δ(λ) from the DSD0/DSV and EMD0/EMV match. Delta shift (:DSF) is only calculated for leading model interfaces.

INPUT attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ST	No	none	INPUT point status (GOOD or OUT).
RAW	No	none	Most recent value read from input RTU data file.
VAL	No	none	Set to :RAW value if :RAW passes data filter checks; otherwise, remains at previous :VAL value.
IMIN	Yes	none	Minimum acceptable value for :RAW.
IMAX	Yes	none	Maximum acceptable value for :RAW.
RB	Yes	none	Maximum acceptable rate of change for :RAW.

JTS (JT)

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
DF	No	none	Diagnostic flow residual, jts(4).
P.ADJ	No	none	Pressure adjustment residual, jts(1).
P.RATE	No	none	Pressure rate of change residual, jts(2).
Q.ADJ	No	none	Flow adjustment residual, jts(3).
Q.ERR	No	none	Flow meter error residual (jts8).
FPDC	No	none	Frictional pressure drop correction residual, jts(7).
FPDC.RATE	No	none	Frictional pressure drop correction residual rate of change residual, jts(5).
GPDC	No	none	Gravitational pressure drop correction residual, jts(10).

Attribute	Pokable	Units	Description
GPDC.RATE	No	none	Gravitational pressure drop correction residual rate of change residual, jts(9).
BMC	No	none	Bulk modulus correction residual, jts(6).
DF.WT	Yes	none	Diagnostic flow rate, jts(6).
P.ADJ.WT	Yes	none	Pressure adjustment weight, jts(1).
P.RATE.WT	Yes	none	Pressure rate of change weight, jts(2).
Q.ADJ.WT	Yes	none	Flow adjustment weight, jtswt(3).
Q.ERR.WT	Yes	none	Flowmeter error weight, jtswt(8).
FPDC.WT	Yes	none	Frictional pressure drop correction weight, jtswt(7).
FPDC.RATE.WT	Yes	none	Frictional pressure drop correction rate of change weight, jtswt(5).
GPDC.WT	Yes	none	Gravitational pressure drop correction weight, jtswt(10).
GPDC.RATE.WT	Yes	none	Gravitational pressure drop correction rate of change, jtswt(9).
BMC.WT	Yes	none	Bulk modulus correction weight, jstwt(6).

Leak detection (LE) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
LF.OMEGA	Yes	none	Fraction of step taken in Leak Analysis minimization technique to determine local derivative.
LF.STATUS(i)	Yes	none	Status of Leak Analysis for each time period. POKE LF.STATUS(0) to OKAY to start Leak Analysis. Descriptions include: {Corresponding integer value}: • STARTING—Leak analysis is turned off. {1} • OKAY—Leak analysis is turned on and there is no leak alarm indicated for the given period. {2} • CIRCULATION—Leak alarm indicated when time-averaged monitor flow at one or more monitors is negative while it is positive by the same amount at one or more neighboring monitors. {3} • LEAK—Leak alarm indicated when a negative flow occurs in one or more monitors over the given period. {4} • INJECTION—Leak alarm indicated when a positive flow occurs in one or more monitors over the given period. {5}
LF.STATUS+(i)	Yes	none	Status of Leak Analysis for each time period based on only positive diagnostic flows. POKE LF.STATUS+(0) to OKAY to start Leak Analysis. Descriptions include: {Corresponding integer value} • STARTING—Leak analysis is turned off. {1} • OKAY—Leak analysis is turned on and there is no leak alarm indicated for the given period. {2} • INJECTION—Leak alarm indicated when a positive flow occurs in one or more monitors over the given period. {4}

Attribute	Pokable	Units	Description
LF.STATUS-(i)	Yes	none	Status of Leak Analysis for each time period based on only negative diagnostic flows. POKE LF.STATUS-(0) to OKAY to start Leak Analysis. Descriptions include: {Corresponding integer value} • STARTING—Leak analysis is turned off. {1} • OKAY—Leak analysis is turned on and there is no leak alarm indicated for the given period. {2} • LEAK—Leak alarm indicated when a negative flow occurs in one or more monitors over the given period. {5}
LF.CONF(i)	No	none	Confidence of alarms, {Corresponding integer value} N_A: LF.STATUS is STARTING or OKAY {1} LOW: (2/3)*LF.TOLER < LF.RESID< 3*LF.TOLER {2} MEDIUM: 3*LF.TOLER < LF.RESID < 10*LF.TOLER or (1/3)*LF.TOLER < LF.RESID < (2/3)*LF.TOLER {3} HIGH: 10*LF.TOLER < LF.RESID or (1/3)*LF.TOLER > LF.RESID {4}
LF.CONF+(i)	No	none	Confidence of alarms based on positive diagnostic flows, {Corresponding integer value} N_A: LF.STATUS+ is STARTING or OKAY {1} LOW: (2/3)*LF.TOLER+ < LF.RESID+ < 3*LF.TOLER+ {2} MEDIUM: 3*LF.TOLER+ < LF.RESID+ < 10*LF.TOLER+ or (1/3)*LF.TOLER+ < LF.RESID+ <(2/3)*LF.TOLER+ {3} HIGH: 10*LF.TOLER+ < LF.RESID+ or (1/3)*LF.TOLER+ > LF.RESID+ {4}
LF.CONF-(i)	No	none	Confidence of alarms based on negative diagnostic flows, {Corresponding integer value} N_A: LF.STATUS is STARTING or OKAY {1} LOW: (2/3)*LF.TOLER- < LF.RESID- < 3*LF.TOLER- {2} MEDIUM: 3*LF.TOLER- < LF.RESID- < 10*LF.TOLER- or (1/3)*LF.TOLER- < LF.RESID- < (2/3)*LF.TOLER- {3} HIGH: 10*LF.TOLER- < LF.RESID- or (1/3)*LF.TOLER- > LF.RESID- {4}

Attribute	Pokable	Units	Description
LF.INVENTORY.FRACTION	Yes	FRACTION/TIME	A tuning parameter that increases LDTs to help tune out unexplained leaks caused by unmodeled error.
LF.LOC(i)	No	DISTANCE	Milepost location of the (i)th leak. The milepost is the user-defined location from the pipe.
LF.LOC+(i)	No	DISTANCE	Milepost location of the (i)th leak based on positive diagnostic flows. The milepost is the user-defined location from the pipe.
LF.LOC-(i)	No	DISTANCE	Milepost location of the (i)th leak based on negative diagnostic flows. The milepost is the user-defined location from the pipe.
LF.ONSET(i)	No	TIME	Time the (i)th leak started.
LF.ONSET+(i)	No	TIME	Time the (i)th leak started based on positive diagnostic flow.
LF.ONSET-(i)	No	TIME	Time the (i)th leak started based on negative diagnostic flow.
LF.RATE(i)	No	FLOW	Rate of (i)th leak (in FLOW units).
LF.RATE+(i)	No	FLOW	Rate of (i)th leak (in FLOW units) based on positive diagnostic flows.
LF.RATE-(i)	No	FLOW	Rate of (i)th leak (in FLOW units) based on negative diagnostic flows.
LF.VOL(i)	No	VOLUME	Cumulative volume of (i)th leak (in VOLUME units).
LF.VOL+(i)	No	VOLUME	Cumulative volume of (i)th leak (in VOLUME units) based on positive diagnostic flows.
LF.VOL-(i)	No	VOLUME	Cumulative volume of (i)th leak (in VOLUME units) based on negative diagnostic flows.
LF.TOLER(i)	Yes	none	Tolerance for comparison of LF.RESID. Once LF.RESID(i) is greater than LF.TOLER(i), Leak Analysis will alarm LEAK, INJECTION, or CIRCULATION, depending upon the size and location of diagnostic flows.
LF.TOLER+(i)	Yes	none	Tolerance for comparison of LF.RESID+. Once LF.RESID+(i) is greater than LF.TOLER+(i), Leak Analysis will alarm LEAK, INJECTION, or CIRCULATION, depending upon the size and location of diagnostic flows.
LF.TOLER-(i)	Yes	none	Tolerance for comparison of LF.RESID-. Once LF.RESID-(i) is greater than LF.TOLER-(i), Leak Analysis will alarm LEAK, INJECTION, or CIRCULATION, depending upon the size and location of diagnostic flows.

Attribute	Pokable	Units	Description
LF.RESET	Yes	none	Used to erase Leak Analysis history. POKE Options include: {Corresponding integer value} • NO there is not presently a request to reset the Leak Analysis history. {1} • YES resets all time-averaged values involved in Leak Analysis to the current instantaneous value. {2} • ZERODF resets all time-averaged values involved in Leak Analysis to the current instantaneous value, and in addition resets the time-averaged diagnostic flows to zero. {3}
LF.RESID(i)	No	none	The error residual from the Leak Analysis flow equations.
LF.RESID+(i)	No	none	The error residual from the Leak Analysis flow equations based on positive diagnostic flows.
LF.RESID-(i)	No	none	The error residual from the Leak Analysis flow equations based on negative diagnostic flows.
LF.PIPE(i)	No	none	The pipe the (i)th leak is in.
LF.PIPE+(i)	No	none	The pipe the (i)th leak is in based on positive diagnostic flows.
LF.PIPE-(i)	No	none	The pipe the (i)th leak is in based on negative diagnostic flows.
LF.PLOC(i)	No	DISTANCE	Location of (i)th leak from the from-end of the pipe.
LF.PLOC+(i)	No	DISTANCE	Location of (i)th leak from the from-end of the pipe based on positive diagnostic flows.
LF.PLOC-(i)	No	DISTANCE	Location of (i)th leak from the from-end of the pipe based on negative diagnostic flows.
LF.MINRES	Yes	none	Value of zero, means do not minimize LF.RESID. Value of one, means minimizes LF.RESID.

Monitor flows (MONFLOWS) (MF) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
QAST	No	none	Status (QAS relative to QASMX). • If QAS < QASMX, then QAST = LOW • If QASMX <= QAS <= 10*QASMX, then QAST = MEDIUM • If 10*QASMX < QAS, then QAST = HIGH
EXCL	Yes	none	YES | NO switch to determine whether to exclude <node>:DQ from QS and QAS calculations. Default is NO.
QS	No	FLOW	Sum of all <monitor>:Q and, if EXCL=NO, all <node>:DQ flows
QAS	No	FLOW	Sum of the absolute values of each <monitor>:Q and, if EXCL=NO, the absolute value of each <node>:DQ flows
QASMX	Yes	FLOW	User-entered flow rate used to determine :QAST

Monitor (E) attributes

Attribute	Pokable	Units	Description
P	Yes	PRESSURE	Model pressure at the monitor external.
SP	Yes	PRESSURE	Set point pressure at the monitor external.
NP	Yes	PRESSURE	Node pressure at the monitor external.
ST	No	TEMP	Temperature set point.
Q	No	FLOW	Flow at monitor external at custody transfer conditions.
CF	Yes	VOLUME	Cumulative flow at monitor external.
F	No	AFLOW	Actual flow at monitor external at pipeline conditions.
DF(i)	No	FLOW	Diagnostic flows associated with the monitor.
NN+	No	none	Node where external is connected.
P0	No	PRESSURE	Fluid reference pressure.
SP0	No	PRESSURE	Set point fluid reference pressure.
NP0	No	PRESSURE	Node fluid reference pressure.
T0	No	TEMP	Fluid reference temperature.
ST0	No	TEMP	Set point fluid reference temperature.
NT0	No	TEMP	Node fluid reference temperature.
D0	No	DENSITY	Reference fluid density at P0 and T0.
SD0	No	DENSITY	Set point reference fluid density at P0 and T0.

Attribute	Pokable	Units	Description
ND0	No	DENSITY	Node reference fluid density at P0 and T0.
PM0	No	BULK.MOD	Reference fluid bulk modulus at P0 and T0.
SPM0	No	BULK.MOD	Set point reference fluid bulk modulus at P0 and T0.
NPM0	No	BULK.MOD	Node reference fluid bulk modulus at P0 and T0.
TM0	No	ΔTEMP	Reference fluid temperature modulus.
STM0	No	ΔTEMP	Set point reference fluid temperature modulus.
NTM0	No	ΔTEMP	Node reference fluid temperature modulus.
PTM	No	none	Multiplier for the ΔPΔT term.
SPTM	No	none	Set point multiplier for the ΔPΔT term.
NPTM	No	none	Node multiplier for the ΔPΔT term.
PPM	No	none	Multiplier for the $(\Delta P)^2$ term.
SPPM	No	none	Set point multiplier for the $(\Delta P)^2$ term.
NPPM	No	none	Node multiplier for the $(\Delta P)^2$ term.
V0	No	VISCOSITY	Reference viscosity at P0 and T0.
SV0	No	VISCOSITY	Set point reference viscosity at P0 and T0.
NV0	No	VISCOSITY	Node reference viscosity at P0 and T0.
VPMI	No	1/ΔP	Pressure coefficient of viscosity.
VTMI	No	1/ΔT	Temperature coefficient of viscosity.
HC	No	HEAT.CAP	Heat capacity.
SHC	No	HEAT.CAP	Set point heat capacity.
NHC	No	HEAT.CAP	Node heat capacity.
FLU	No	none	Fluid name.
SFLU	Yes	none	Set point fluid name.
NFLU	No	none	Node fluid Name.
CB	No	none	Name of SCADA element controlling monitor pressure.
NQ	No	FLOW	Nominal flow; same as Q.
NF	No	AFLOW	Nominal flow; same as F.
SP	Yes	PRESSURE	Set point pressure.
DP	No	ΔPRESSURE	SP minus :P+.
PMAX	Yes	PRESSURE	Highest allowable pressure.
PMIN	Yes	PRESSURE	Lowest allowable pressure.
QMAX	Yes	FLOW	Highest allowable flow.
QMIN	Yes	FLOW	Lowest allowable flow.
STYP	No	none	External type.

Attribute	Pokable	Units	Description
WOB	No	none	Wobbe number.
SWOB	No	none	Set point Wobbe number.
NWOB	No	none	Node Wobbe number.
SG	No	none	Gas specific gravity.
SSG	No	none	Set point for gas gravity.
NSG	No	none	Node gas specific gravity.
LHV	No	HEAT.VAL	Lower heating value.
SLHV	No	HEAT.VAL	Set point lower heating value.
NLHV	No	HEAT.VAL	Node lower heating value.
HHV	No	HEAT.VAL	Higher heating value.
SHHV	No	HEAT.VAL	Set point higher heating value.
NHHV	No	HEAT.VAL	Node higher heating value.
T	No	TEMP	Flowing temperature.
NT	No	TEMP	Node temperature.
VP	No	PRESSURE	Vapor pressure.
SVP	No	PRESSURE	Set point vapor pressure.
NVP	No	PRESSURE	Node vapor pressure.
COMP	No	none	Name of BWRS components and associated percentage.
MST	Yes	none	Monitor status. :MST may be poked to OPEN or to CLOSE. The monitor status descriptions are: {Corresponding integer value} • OPENED—Monitor is opened and allows diagnostic flows. {1} • CLOSED—Monitor is closed and prevents diagnostic flows. {2} • CLOSING—Monitor is transitioning to CLOSED. {3} • OPENING—Monitor is transitioning to OPENED. {4}
RMUL	Yes	none	Not used.
DQWT	Yes	none	Not used.
DPWT	Yes	none	Not used.
AGE	Yes	TIME	Monitor age. The amount of time that the monitor has been open. :AGE is set to 0 at startup of the model and is incremented by DT every time step taken. :AGE will be automatically reset to zero when a “LOAD.STATUS, SET.TIME=” command is given, or the monitor is poked to open.

Attribute	Pokable	Units	Description
MAGE	Yes	TIME	Maximum age in minutes. When :AGE becomes greater than :MAGE, the associated monitor will begin to close. {1E20}
H	No	TFLOW	Thermal flow rate.
NH	No	TFLOW	Nominal thermal flow rate.
NCMC	Yes	none	Number of control mode changes.
CO2	No	none	CO2 fraction.
LABEL(i)	No	none	User-defined gas properties for State AGA.

Online (ON) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
DT.ALARM	Yes	TIME	Parameter that determines how often the alarm or file is written. {1E10}
DT.COMPUTED	Yes	TIME	Obsolete.
DT.FLUIDS	Yes	TIME	Parameter that determines how often the fluids.inc file is overwritten with online calculated values from an EOS loop.
FRIC.ACC	Yes	none	Friction accuracy multiplier used to set bound on pipe friction pressure drop.
PE.MUL	Yes	none	The tuning parameters PE.MUL, EE.MUL, and QE.MUL are used to vary the magnitude of LDT curves, thereby making leak detection more conservative or less conservative, without affecting State Estimation. You should enter repeatabilities appropriate for State Estimation then use these multipliers to tune the LDT curves. {1} Pressure Monitor Repeatability Multiplier used in the calculation of Leak Detection Thresholds. The repeatability is multiplied by PE.MUL before being used in subsequent calculations. This may be used to globally change the pressure monitor repeatability uncertainties used in the calculation for LDTs. {1} A value of 2.00 would increase the thresholds and make it more difficult to announce a leak while a value of .5 would have the opposite effect.

Attribute	Pokable	Units	Description
QE.MUL	Yes	none	Flow Meter Repeatability Multiplier used in the calculation of Leak Detection Thresholds. The repeatability is multiplied by QE.MUL before being used in subsequent calculations. This may be used to globally change the flow meter repeatability uncertainties. {1.00.}
EE.MUL	Yes	none	Elevation Error Multiplier used in the calculation of Leak Detection Thresholds. The repeatability is multiplied by EE.MUL before being used in subsequent calculations. This may be used to globally change the elevation uncertainties used in the calculation for LDTs. {1.00} A value of 2.00 would increase the thresholds and make it more difficult to announce a leak while a value of .5 would have the opposite effect.
DRHO.MIN	Yes	DENSITY	The minimum difference in density expected in a pipe. DHRO.MIN is used in determining the bounds on Pressure Drop Forces (PDFs) in online models.
LF.DATAOUT	No		Not used.
MAXSCANS	Yes	none	The maximum number of scans for each RTU data point saved in memory that will be accumulated before State Estimation extrapolates. Used as follows: Extrapolate if: number points in memory > MAXSCANS * number_of_scada_elements (non - fake) Default: 60.
PFC.BOUND	Yes	none	Pipe friction correction bound. The effective friction factor is :FFE * (1 + PIPE:FC) * (1 + BATCH:FC).
PSC.BOUND	Yes	none	Pipe slope correction bound. Pipe slope corrections are made to correct for the gravitational part of PDFs.
ACP.PRES	Yes	TIME	Accuracy correction period for pressure.
ACP.PFRIC	Yes	none	Autocalibration period for friction correction.
ACP.SLOPE	Yes	none	Autocalibration period for pipe slope correction.
ACP.BFRIC	Yes	TIME	Autocalibration period for batch friction correction.
BFC.BOUND	Yes	none	Batch friction correction bound. Batch friction corrections are made to correct for the frictional part of PDFs.
LF.MAXWAIT	Yes	TIME	Parameter used in determining whether State Estimation extrapolates or interpolates. The model uses the following relationship to determine whether it can take a step: (tnewpt - LF.MAXWAIT > MODEL.TIME + dtmin).
ACP.FLUIDS	Yes	TIME	Autocalibration period for adjusting fluids database to agree with fluids measurement. {2880 mins}

Attribute	Pokable	Units	Description
RTU.FILE	No	none	The name of the RTU data file the model is currently accessing for data. Includes path.
OL.MINITR	Yes	none	Minimum number of iterations used in solving JTS equations.
OL.MAXITR	Yes	none	Maximum number of iterations used in solving JTS equations.
OL.NUMITR	No	none	Number of iterations performed in solving JTS equations.
OL.PART	No	none	Number of partitions that comprise JTS matrix. The JTS matrix is a sparse matrix with dimensions OL.NUNK by OL.NUNK. If OL.NUNK is 85, this means that the matrix must be solved 85 times to get a complete set of partial derivatives for a given time step. In order to speed things up, the matrix is partitioned into independent sections whose resulting partial derivatives are only a function of the elements within that partition. This allows the matrix to be solved fewer times.
OL.NUNK	No	none	Number of unknowns in JTS matrix.
OL.TOL1	Yes	none	Not used.
OL.TOL2	Yes	none	Not used.
JTSWT(n)	Yes	none	JTS weight for the individual equations.
BMC.BOUND	Yes	none	The upper and lower bound for bulk modulus correction.
BMC.BOUND	Yes	none	The upper and lower bound for bulk modulus correction.
RTU.TMAX	No	TIME	Largest time in RTU data file.
RTU.TIME	No	TIME	Time associated with RTU.RECNO.
RTU.RECNO	No	none	Point number being read by the model in the RTU data file
JTS.PRINT	Yes	none	Number of JTS equations to print to OUTTRN when debugger flag is on.
JTS.SORTED	Yes	none	Used to output JTS equations to OUTTRN; sorted by residual.
PDF.GFRAC	Yes	none	Not used.
PDF.FFRAC	Yes	none	Not used.
PDFB.MUL	Yes	none	Not used.
TAP.AUTOCAL	Yes	TIME	Time averaging period for autocalibration.
DAT.LBOUND	Yes	none	Lower bound on the data quality value for autocalibration.
DIF.LBOUND	Yes	none	Lower bound on the difference value for autocalibration.

Attribute	Pokable	Units	Description
LM.WT	Yes	none	Leakless monitor weight.
DF.TOL	Yes	none	Diagnostic flow tolerance (Fraction).
S.PWT	Yes	none	Small weight on JTS pressure equation (debug).
M.PWT	Yes	none	Medium weight on JTS pressure equation (debug).

Property (PR) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
STYP	No	none	Subtype.
CNC	No	none	Property element connection point.
P	No	PRESSURE	Pressure at CNC.
T	No	TEMP	Temperature at CNC.
MT	Yes	TEMP	Measured Temperature.
CBT	No	none	Controlled by for Temperature.
MP	Yes	PRESSURE	Measured Pressure.
CBP	No	none	Controlled by for Pressure.
MD	Yes	DENSITY	Measured Density.
CBD	No	none	Controlled by for Density.
MVP	Yes	PRESSURE	Measured Vapor Pressure.
CBVP	No	none	Controlled by for Vapor Pressure.
MDE	Yes	DENSITY	Measured Density Error.
CBDE	No	none	Controlled by for Density Error.
MBM	Yes	BULK.MOD	Measured Bulk Modulus.
CBBM	No	none	Controlled by for Bulk Modulus.
MBME	Yes	PERCENT	Measured Bulk Modulus Error.
CBBE	No	none	Controlled by For Bulk Modulus Error.
MV	Yes	VISCOSITY	Measured Viscosity.
CBV	No	none	Controlled by for Viscosity.
MCO2	Yes	COMP	Measured CO2.
CBCO2	No	none	Controlled by for CO2.
MHHV	Ye	HEAT.VAL	Measured HHV.
CBHHV	No	none	Controlled by for HHV.
MSG	Yes	none	Measured specific gravity.
CBSG	No	none	Controlled by for specific gravity.

Attribute	Pokable	Units	Description
M<LABEL>	Yes	none	Measured <label>.
CV<LABEL>	No	none	Controlled by for <label>.

RTUFILT attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ID	No	none	Corresponding name in input RTU data file.

SCADA (SC) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
ID	No	n/a	Name of the SCADA element in the RTU data file when the name does not match the name in the RTU data file.
RAW	Yes	user units	The interpolated or extrapolated value from the RTU data file.
FILT	Yes	user units	The filtered value from the RTU data file. (:RAW after Limit Checks and Time Averaging)
VAL	Yes	user units	SCADA point value after passing, :IMIN, :IMAX, and rate of change tests, time averaging, function and interpolation/ extrapolation.
USE	No	user units	Value used by model in determining the hydraulic state of the pipeline (:VAL after State Estimation)
EREP	No	user units	Effective repeatability takes into consideration the repeatability of the measuring process (:REP), interpolation/ extrapolation and errors, time tag error, and quality flags.
RC	No	user units	Repeatability correction is determined for SCADA points driving monitor pressure or external flows. State Estimation will adjust their values relative to one another to best fit the current pipeline state. :RC is constrained to fall between +/- :EREP.
AC	Yes	user units	Current accuracy correction. (See "Autocalibration of pressure drop errors" on page 752.)
ACC	Yes	user units	Maximum limits for accuracy correction.

Attribute	Pokable	Units	Description
ACP	Yes	TIME	Autocalibration period defines the period of time over which a constant error between actual and reported SCADA points due to inaccuracy of the reporting instrument may be corrected with :AC via the autocalibration process. (See "Autocalibration of pressure drop errors" on page 752.)
DR	No	none	Poke name of model element parameter that is driven by the SCADA element.
RDR	Yes	user units /TIME	Repeatability decay rate defines the greatest rate at which the measured quantity may change from the closest reported value.
UDT	No	TIME	The difference in time between simulation time and the time stamp for the closest point in the RTU data file that has the same quality as :ST. A :UDT is calculated for each quality type. The lowest nominated :EREP determines which :UDT is displayed.
SP	Yes	TIME	Expected scan period. It is used to calculate :EREP.
SDB	Yes	user units	Bound on the second difference filter.
TEB	Yes	TIME	Time tag error bound quantifies the maximum difference between the time stamp associated with a point in the RTU data file and the time that the measurement was actually sampled.
FILP	No	user units /TIME	The maximum of the absolute value of the slope of interpolation between pairs of RTU data points. The RTU data points that are included in the calculation of :FILP include the RTU data points between the current and previous time steps, and the RTU data point before the previous time step and the points after the current time step if available. Used in the calculation of :EREP.
REP	Yes	user units	Repeatability value for use when :ST is GOOD.
BREP	Yes	user units	Bad repeatability is used in lieu of :REP when :ST is BAD.
MREP	Yes	user units	Repeatability associated with MANUAL points. Used in calculating manrep.
NREP	Yes	user units	Repeatability associated with NOCHANGE points. Used in calculating nochrep.
OREP	Yes	user units	Repeatability used when MODE is OFF.
RREM	No	user units	Repeatability correction that is available without reaching the correction limit. If RREM is negative, the value has been adjusted beyond its limit.

Attribute	Pokable	Units	Description
ST	No	none	Status of the RAW value being used at this point based on the lowest EREP value. Options include: {Corresponding integer values} • GOOD—RTU data OK {1} • BAD—SCADA system doubts the point is GOOD or :UDT > :TOUT {2} • NOCHANGE—RTU data has not changed outside dead band {3} • MANUAL—RTU data is manually entered {4} **Note:** ST will be BAD if no points of the same quality are available for interpolation/extrapolation.
LIM	No	none	Limit status of the RAW value being used. Status descriptions are: {Corresponding integer values} • GOOD—RTU data OK {1} • HIGH—:FILT > :IMAX {2} • LOW—:FILT < :IMIN {3} • TOOFAST—The change with respect to time between the two real points used for interpolation was greater than :RB. {4} **Note:** If :STR is BAD, then BMIN, BMAX and BRB are used.
STR	No	none	Status of the nearest SCADA data point to the model time. Status descriptions include: {Corresponding integer values} • GOOD—RTU data OK {1} • BAD—SCADA system doubts the point is GOOD or :UDT > :TOUT {2} • NOCHANGE—RTU data has not changed outside dead band {3} • MANUAL—RTU data is manually entered {4}

Attribute	Pokable	Units	Description
LIMR	No	none	Limit status of the nearest SCADA data point to the model time. Status descriptions include: {Corresponding integer values} • GOOD – RTUA data OK {1} • HIGH – :RAW > :IMAX {2} • LOW – :RAW < :IMIN {3} • SDB – Second difference (|SD|) exceeds bound (:SDB) **Note:** See the "Box C" description in "SCADA element flow chart" on page 794 for more information. • TOOFAST – The change between the two real points used for interpolation was greater than :RB. {4} **Note:** If :STR is BAD, then BMIN, BMAX, and BRB are used.
MODE	Yes	none	Indicates mode. (See the *Off* description below.)
OFF	Yes	user units	:OFF is set to :USE when mode is ON and :ST is not BAD. Otherwise :VAL is set to :OFF.
IMIN	Yes	user units	Minimum acceptable value of :RAW. Short-term violations of this test result in :LIMR set to LOW.
IMAX	Yes	user units	Maximum acceptable value of :RAW. Short-term violations of this test result in :LIMR set to HIGH.
OMIN	Yes	user units	Minimum acceptable :USE after :VAL is modified by State Estimation.
OMAX	Yes	user units	Maximum acceptable :USE after :VAL is modified by State Estimation.
RB	Yes	user units /TIME	Maximum acceptable rate of change of the SCADA point. Compares previous GOOD raw values with the current raw values with the current raw value.
BMIN	Yes	user units	Minimum value accepted when ST is BAD.
BMAX	Yes	user units	Maximum value accepted when :ST is BAD.
BRB	Yes	user units	Maximum rate of change of :VAL when :ST is BAD. If :VAL changes at a rate greater than +/-:BRB, then :VAL is changed to PREV(:VAL).
TAVE	Yes	TIME	Time averaging period used for input.

Attribute	Pokable	Units	Description
TOUT	Yes	TIME	Time out period. Defines the maximum period of time that can elapse between current simulation time and the closest RTU data file time stamp before the value's status is considered BAD.
FORE	Yes	none	This item is simply a peek/poke for you to set in any way. The name, FORE, anticipates that the you defined it to a forecasted value.
NDUP	Yes	none	Number of duplicate updates.
FLAT	No	TIME	Time that the SCADA point has been flat (constant).
RULE	Yes	none	Interpolation rule used to interpret RTU data points. Valid values include: LINEAR, EARLY, MIDDLE, and LATE.

Span (SP) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
NDEV	No	none	Number of elements.
DEV	No	none	Name of the elements.
LEN	No	LENGTH.PIPE	Total length of the span.
P-	No	PRESSURE	Pressure at from-end of span.
P+	No	PRESSURE	Pressure at to-end of span.
T-	No	TEMP	TEMP at from-end of span.
T+	No	TEMP	TEMP at to-end of span.
Q-	No	FLOW	Flow into from-end, standard conditions.
Q+	No	FLOW	Flow out of to-end, standard conditions.
F-	No	AFLOW	Flow into from-end, pipeline conditions.
F+	No	AFLOW	Flow out of to-end, pipeline conditions.
FPD	No	ΔPRESSURE	Total span frictional pressure drop.
PDF	No	PRESSURE	Total span pressure drop force.
FC	Yes	none	Friction correction factor.
FCR	No	none	Friction correction ratio.
PD	No	ΔPRESSURE	Pressure drop of span: (P-) - (P+).
PK	No	FLOW	Pack rate of span: (Q-) - (Q+), Standard Conditions.
PKF	No	AFLOW	Pack rate of span: (F-) - (F+), pipeline conditions.
INV	No	VOLUME	Span inventory.
NBAT	No	none	Number of batch interfaces in span.

Attribute	Pokable	Units	Description
PKM	No	MFLOW	Pack rate of span.
FFE	No	none	Effective friction factor.
LPDT	No	TIME	Low pressure delta time.
NINT	No	none	Number of intervals of the (# pipe knots - 1).
FLU(i)	No	none	Name of fluid(s) in span.
ID1(i)	No	none	Additional fluid identifier.
ID2(i)	No	none	Additional fluid identifier.
DT	No	TIME	Nominated time step.
PDFH	No	PRESSURE	PDF High Limit.
PDFL	No	PRESSURE	PDF Low Limit.
FPDC	No	none	Friction pressure drop correction coefficient.
GPDC	No	none	Gravity pressure drop correction coefficient.
GPD	No	PRESSURE	Total gravitation pressure drop.
GPE	No	none	Gravitational pressure drop error.
DIF	No	none	Measure of the suitability of the observed difference in flow during the preceding TAP.AUTOCAL period.
DAT	No	none	Measure of the quality of the data during the preceding TAP.AUTOCAL period.
DATI	No	none	Value that is integrated to get :DAT.
DIF	No	none	Measure of the suitability of the observed difference in flow during the preceding TAP.AUTOCAL period.
MF	No	none	Integral of the modeled frictional pressure drop over the previous TAP.AUTOCAL.
MFMF	No	none	Integral of the square of the model frictional pressure drop over the previous TAP.AUTOCAL.
SF	No	none	Integral of the SCADA measured frictional pressure drop over the previous TAP.AUTOCAL.
MFSF	No	none	Integral of the SCADA measured frictional pressure drop multiplied by the modeled measured frictional pressure drop over the previous TAP.AUTOCAL.
MFMF	No	none	Integral of the square of the model frictional pressure drop over the previous TAP.AUTOCAL.
MON-	No	none	From-end controlling monitor.
MON+	No	none	To-end controlling monitor.
CP-	No	none	From-end controlling path.
CP+	No	none	To-end controlling path.

Transfer lines (T) attributes

Also see "Common peek and poke attributes" on page 641.

Attribute	Pokable	Units	Description
P-	Yes	PRESSURE	Pressure at from-end of pipe.
P+	Yes	PRESSURE	Pressure at to-end of pipe.
T-	No	TEMP	Temperature at from-end of pipe.
T+	No	TEMP	Temperature at to-end of pipe.
Q-	No	FLOW	Flow into from-end, Standard Conditions.
Q+	No	FLOW	Flow out of to-end, Standard Conditions.
PD	No	ΔPRESSURE	Pressure drop of pipe: (P-) - (P+).
PK	No	FLOW	Pack rate of pipe: (Q-) - (Q+).
LEN	Yes	LENGTH.PIPE	Length of pipe.
OD	Yes	DIAM	Outside diameter of pipe.
WT	Yes	WALL	Wall thickness of pipe.
V-	No	VEL	Fluid velocity into from-end of pipe.
V+	No	VEL	Fluid velocity out of to-end of pipe.
F-	No	AFLOW	Flow into from-end, pipeline conditions.
F+	No	AFLOW	Flow out of to-end, pipeline conditions.
PKF	No	AFLOW	Pack rate of pipe: (F-) - (F+).
FLU	Yes	NONE	Name of the fluid.
ETA (±n)	No	TIME	Estimated time of arrival for ±n batch interface.
POS (±n)	Yes	LENGTH.PIPE	Location of ±n batch interface.
INV	No	none	Pipe inventory.
IC(0)	No	none	Incremental conductivity.
IC(1)	No	none	Incremental conductivity.
IC(2)	No	none	Incremental conductivity.
IC(3)	No	none	Incremental conductivity.
IC(4)	No	none	Incremental conductivity.
AMP	Yes	none	Flow area multiplier.
FF	Yes	none	Interval-by-interval average of the effective friction factor.
RUF	Yes	WALL	Pipe roughness.
FPD	No	ΔPRESSURE	Frictional pressure drop.
GPD	No	ΔPRESSURE	Gravitational pressure drop.
FC	Yes	none	Friction correction factor.

Attribute	Pokable	Units	Description
SC	Yes	none	Slope correction factor.
LPDT	Yes	TIME	Low pressure delta time.
SVB	Yes	VOLUME	Slack volume bound.
PDF	Yes	PRESSURE	Pressure drop force (liquid only).
BMC	Yes	none	Bulk modulus correction coefficient (liquid only).
ID1(N)	Yes	none	Fluid identifier 1.
ID2(N)	Yes	none	Fluid identifier 2.
VPR	No	PRESSURE	Percentage of pipe volume that is vapor.
NBAT	No	none	Number of fluid mixture vectors in pipe.
BPP(N)	No	none	Batch pipe position.
HMUL	Yes	none	Hold-up mult.
HMIN	Yes	none	Hold-up min.
HMIN	Yes	none	Hold-up min.
EE	No	ELEV	Elevation error.
MERR	No	none	Bound on mass imbalance due to slack line flow modeling error (numerical truncation).
PKM	No	MFLOW	Pack rate of pipe.
FFE	No	none	Effective friction factor.
NINT	No	none	Number of intervals of the pipe (# pipe knots - 1).
TG-	Yes	TEMP	Temperature of ground layer.
TG+	Yes	TEMP	Temperature of ground layer.
VC-	No	VELOCITY	From-end velocity correction.
VC+	No	VELOCITY	To-end velocity correction.
VP-	No	PRESSURE	From-end vapor pressure.
VP+	No	PRESSURE	To-end vapor pressure.
DT	No	TIME	Nominated time step.
DTM	No	TIME	Minimum DT within isolation distribution.
PDFH	No	PRESSURE	Upper bound of PDF.
PDFL	No	PRESSURE	Lower bound of PDF.
E-	No	ELEVATION	From-end elevation.
E+	B	ELEVATION	To-end elevation.
GPE	No	PRESSURE	Gravitational pressure drop error.

Utilities for online modeling

The following utilities are provided to facilitate tasks related to online modeling:

- *COMPRESS_RTU.* Reduces the storage required for initialization by continuously reading the SCADA system-generated RTU data file, time-averaging the data, and then writing a compressed RTU data file. This compressed file is then used as the dynamic data to drive the Statefinder or Leakfinder analysis. COMPRESS_RTU is also useful in an off-line test environment for Statefinder or Leakfinder models.
- *CTOREVIEW.* Converts an RTU data file to a REVIEW file, which can be accessed by TPORT or SimPlot to view time plots of the SCADA data.
- *DRTU.* Allows you to view the contents of a binary RTU data file produced by the SCADA interface or the COMPRESS_RTU utility. This utility may be run through the command line or through a Windows user interface. See "DRTU (Windows)" on page 872 for more information.
- *RTUMERGE.* Combines the content of multiple RTU data files, generated form different sources or SCADA systems, by continuously reading the content of the files and merging them into a single file.

COMPRESS_RTU

```
COMPRESS_RTU [input-file-name] [output-file-name]
[-MATCH=match-list]     /* (s1,s2,...,sn)
[-INTERVAL=averaging-interval]     /* real
[-MAXRECS=max-points-in-output-file]     /* integer
[-START.RTUDATA=first-rec-to-read]     /* integer
[-START.COMPRESS=first-rec-to-write]     /* integer
[-TIMEOUT=time-out-period(seconds)]     /* integer
[-ECHO]     /* echo-reads-and-writes
[-SPEED=times-real-time]     /* real
[-h]
```

Field	Description
input-file-name	Name of RTU data file to be read for input. {rtudata.dt}
output-file-name	Name of new RTU data file to be opened for output. {compress.dt}
match	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be passed through to the output file. Wild cards * and ? are permitted. Match arguments are converted to uppercase; therefore, names in the RTU data file with lowercase characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 52. The first record in the file is always displayed first, whether or not it meets the match criteria {*}.
interval	For each SCADA point ID, this specifies the interval of time in minutes over which data from the RTU data file will be averaged.
maxrecs	Maximum number of points to allocate space for in the output file. After writing this many points, the output will wrap around and begin again, overwriting point number 1. Applicable only when the file is created. {100000}
start.rtudata	The point number where COMPRESS.RTU starts reading the input data file. The default is to start at the time corresponding to the output data file's newest points. (This default means that COMPRESS.RTU can be restarted after any amount of down-time, provided no input data has been lost.) The SCADA point numbers can be determined using the DRTU utility.
start.compress	Represents the starting point number to start writing to the output data file. The default is to start at the point just after the newest point in the output data file. The SCADA record numbers can be determined using the DRTU utility.
timeout	Period of time (seconds) the utility will wait for new data when the end of the RTU data file is encountered. After this amount of time, with no new data, SPS will exit. {172800 seconds}
echo	Echoes input points and output points to the display. Default is no echo. This option produces a lot of output and is not typically used.

Field	Description
speed	This option is used to set up a test environment in which the output file is being written as if the RTU data file is being written by a real SCADA system. SPEED=1 (and INTERVAL=0.001) will delay its output so that it copies records from its input file to its output file in real time. In an online environment, COMPRESS_RTU automatically waits for its input file to be updated, so the SPEED option is not necessary. The default is to write the data as quickly as possible.
h	Print the options this utility supports.

Description

COMPRESS_RTU is a utility used to average data contained in an RTUDATA file. It creates a compressed RTUDATA file.

In operational environments, such as Statefinder models or Leakfinder models of batched pipeline systems, the amount of historical recorded RTU data required to initialize a simulation is substantial. To reduce the storage required by this “cold start” process, COMPRESS_RTU is run on a continuous basis, reading the SCADA system generated RTU data file, time averaging the data, and writing a compressed RTU data file. This compressed file is then used as the dynamic data to drive the Statefinder or Leakfinder.

COMPRESS_RTU is also useful in an off-line test environment for Statefinder or Leakfinder models. With the various options, you can

- Average an RTU data file
- Create another RTUDATA file with only selected data points
- Start at a specified input file record
- Specify the time averaging period *and/or* specify the speed at which the output data is written

Take note

Output

Typically no output is displayed. The ECHO option enables various analytical output messages. Also, diagnostic messages are displayed.

INTERPOLATION.RULE

COMPRESS_RTU is incompatible with the INTERPOLATION.RULE option on the SCADA device. COMPRESS_RTU always uses simple averaging.

Quitting

.COMPRESS_RTU will run until no data remains for the timeout period. You can also stop the process using a control C.

Example input

Using a compression interval of 10 minutes, COMPRESS_RTU would write the following to its output file:

```
Example 1
REC#   DATE/TIME STAMP     SCADA ID   VALUE
```

```
100    94/05/01 00:00:05   Q1          250
101    94/05/01 00:00:05   Q2          250
102    94/05/01 00:00:15   Q1          350
103    94/05/01 00:00:15   Q2          350
Examples
compress_rtu rtu.dt -match="(s1,..,sn)"
Examples
compress_rtu rtu.dt -interval=10
Examples
compress_rtu rtu.dt -maxrecs=2500000
Examples
compress_rtu rtu.dt -st.rtu=125350
Examples
compress_rtu rtu.dt -st.comp=125350
Examples
compress_rtu rtu.dt -timeout=600
Examples
compress_rtu rtu.dt compress.dt -echo
Examples
compress_rtu rtu.dt compr.dt -speed=2
```

CTOREVIEW

```
CTOREVIEW [input-file-name] [output-file-name]
[-TBEGIN= begin-time]    /*"92/01/01 11:57:00"
[-TEND= end-time]    /*"91/01/01 11:57:00"
[-MATCH= match-list]    /*(s1,s2,...,sn)
[-MAX.NAMES= number-of-names]     /*integer
```

Field	Description
input-file-name	Name of RTU data file to convert. {rtudata.dt}
output-file-name	Name of REVIEW file to create must start with an alpha character. {bc.review}
begin-time	The beginning time in the input file to be converted. The utility will look for a data point with a time stamp close to the TBEGIN date/time. Data will be converted and written to the output file from that point until program termination is caused by TEND (or the end of the data).
end-time	This is the end time for data to be converted. CTOREVIEW will quit when it finds a point with time greater than TEND (or the end of the data).
match-list	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be displayed. Wild Cards * and ? are allowed, where * matches a string and ? matches a character. Match arguments are converted to uppercase; therefore, names in the RTU data file with lower case characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 52.
number-of-names	A guess at the number of identifiers that will be in the REVIEW file. The REVIEW file has a dictionary at the front; space for this dictionary is pre-allocated (the dictionary automatically grows, if required, overwriting early data records). The CTOREVIEW program allocates space for MAX.NAMES, assuming the names average 12 characters each {1000}. Do not use a very large number for MAX.NAMES, because disk space will be allocated to hold that many names.

Description

CTOREVIEW is a utility designed to convert an RTU data file to a REVIEW file. The REVIEW file can be accessed by TPORT or SimPlot to view time plots of the SCADA data.

Take note

Output

Progress is displayed in the following format:

```
Outputting a record for <SCADA name>, time <time>
Outputting a record for SS.102, time 92/10/25 22:00:00
```

Example input

```
Examples
ctoreview rtu.dt -tbegin=92/01/01_11:57:00
```

```
Examples
ctoreview rtu.dt -tend=92/01/01_11:57:00
Examples
ctoreview rtu.dt -match="(s1,s2,s3,...,sn)"
Examples
ctoreview rtu.dt -max.names=1000
```

DRTU

```
DRTU [input-file-name]
[-MATCH=match-list]   /* (s1,s2,...,sn)
[-TBEGIN=begin-time]   /* "YY/MM/DD_hh:mm:ss"
[-TEND=end-time]   /* "YY/MM/DD_hh:mm:ss"
[-IDWIDTH=width]
[-MAXTRYS=number-of-retries]   /* integer
[-PBEGIN=begin-record-number]   /* integer
[-PEND=end-record-number]   /* integer
[-WAIT=milliseconds-between-retries]   /* integer
[-QUALITY=quality-flag]   /* (s1,s2,...sn)
[-CUSPS=<cusp-seconds>]   /* integer
[-GAPS=<gap-seconds>]   /* integer
[-GAPANY]
[-PROGRESS=records]   /* integer
```

Field	Description
input-file-name	The name of the RTU data file to read. {rtudata.dt}
match-list	Allows you to specify one or more text strings to match with SCADA point IDs within the RTU data file. Only those that match the MATCH arguments will be displayed. Wild cards "*" and "?" are allowed. Match arguments are converted to uppercase; therefore, names in the RTU data file with lower case characters will not be matched if the MATCH argument is used, unless the lower case characters match a wildcard. See "Wildcards" on page 52.
begin-time	This option is used to specify the beginning time of data to be viewed. The utility will look for a data point with a time stamp close to the TBEGIN date/time. Data will be displayed from that point on until program termination is caused by TEND, REND, or the end of the data. DRTU will skip data with bad time tags in exactly the same manner as the TRANS program.
end-time	This is the end time for data to be viewed. When DRTU reads a point ID with a time stamp after the TEND date/time, DRTU terminates.
width	This specifies the width of the SCADA point name to be listed. The default width is 12 characters.
number-of-retries	Number of times DRTU will retry reading each data point. DRTU will exit if new data is not available after this many retries of any point. {100} See the description of WAIT *milliseconds-between-retries*.
begin-record-number	This is the first point number to be viewed. The utility will display data beginning with this point number until terminated by a TEND, REND, or end of data. Use RBEGIN to view points suspected of having bad time tags. When RBEGIN is specified, DRTU prints out all records, including those with bad time tags. Also, DRTU will read past the cusp in this case.
end-record-number	This is the last point number to be viewed. The utility will terminate when this record is encountered.
milliseconds-between-retries	Wait time between retries, in milliseconds. {1000} See the description of MAXTRYS *number-of-retries*.

Field	Description
quality-flag	Allows you to specify that only points for which quality matches the specified quality flags be displayed.
cusp-seconds	Ignores MATCH. Will print only the points adjacent to a cusp of the specified size or larger. If used without RBEGIN, it implies RBEGIN = 1.
gap-seconds	Based on the current MATCH settings, prints only lines that would have been printed, with adjacent lines having a time-stamp separated by at least the specified amount.
GAPANY	Modifies GAPS, so that any time any single ID has the specified time gap, the points bounding the gap are printed.
records	Outputs a record, regardless of MATCH, CUSP, etc., after every specified number of records. This option makes it easier to see that DRTU is functioning when the output would otherwise be very sparse.

Description

DRTU is a utility designed to allow you to view the contents of a binary RTU data file produced by the SCADA interface or the COMPRESS_RTU utility. The RTU data file is a circular data file that contains SCADA point data for the input of dynamic data to the Statefinder or Leakfinder models. The file has a finite length; new data points are written over the oldest points once the size of the file reaches its defined size limit. The boundary between the newest point and the oldest point is referred to as the *cusp*.

Take note

Output

DRTU will display the contents of the RTU data file in the following manner:

```
Point No.    Date     Time      SCADA Point Name     Value       Quality

125000        91/07/21 00:42:44  STA535.PRES.DISC     757.89      good
125001        91/07/21 00:42:44  STA535.PRES.SUCT     590.45      good
125002        91/07/21 00:43:04  STA520.FLOW.STD      1987.12     good
```

Example input

The following examples should help illustrate the DRTU utility:

```
drtu rtudata.dt
Point No.    Date     Time       SCADA Point Name     Value     Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC      757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT      590.45     good
125002     91/07/21 00:43:04     STA520.FLOW.STD      1987.12     good
125003     91/07/21 00:42:44     STA060.PRES.DISC      799.03     good
125004     91/07/21 00:42:44     STA060.PRES.SUCT      653.87     good
125005     91/07/21 00:43:04     STA520.PRES.DISC      749.33     good
125006     91/07/21 00:43:04     STA520.PRES.SUCT      624.29     good
125007     91/07/21 00:43:04     STA520.FLOW.ACT      1103.65     good
```

Match is not case sensitive; however, use of match excludes any names with lowercase characters:

```
drtu rtu.dt -match=sta060.pres.disc     <---WINDOWS/DOS
Point No.     Date     Time      SCADA Point Name     Value     Quality
125003     91/07/21 00:42:44     STA060.PRES.DISC     799.03     good
125354     91/07/21 00:43:26     STA060.PRES.DISC     799.67     good
125702     91/07/21 00:43:59     STA060.PRES.DISC     799.92     good
drtu rtu.dt -match=sta520.flow.act    <---WINDOWS
Point No.     Date     Time     SCADA Point Name       Value    Quality
 < no points match - point name is lowercase in RTUDATA.DT>
```

Wildcards may be used. See "Wildcards" on page 52.

```
drtu rtu.dt "-match=sta5??.pres.*"     <---WINDOW
Point No.    Date     Time       SCADA Point Name    Value  Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC     757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT     590.45     good
125005     91/07/21 00:43:04     STA520.PRES.DISC     749.33     good
125006     91/07/21 00:43:04     STA520.PRES.SUCT     624.29     good
```

Time specification:

```
drtu rtu.dt -tbegin=91/07/21 -tend=91/07/21_00:43:00   <---WINDOWS
Point No.    Date     Time       SCADA Point Name     Value    Quality
125000     91/07/21 00:42:44     STA535.PRES.DISC     757.89     good
125001     91/07/21 00:42:44     STA535.PRES.SUCT     590.45     good
125003     91/07/21 00:42:44     STA060.PRES.DISC     799.03     good
125004     91/07/21 00:42:44     STA060.PRES.SUCT     653.87     good
```

DRTU (Windows)

DRTUW allows you to view the contents of a binary RTU file produced by a SCADA interface or the COMPRESS_RTU utility. It is a Windows version of the DOS-run predecessor utility, DRTU.

DRTUW lets you:

- Open and view an existing RTU file
- Set various identifier matches, ranges, and qualities
- Adjust the display speed of the data
- Save the text to a text read file
- Pause, resume, stop, or restart the reading while viewing the RTU data

To start DTRUW

From a command line, type `drtuw`.

To use DRTUW

To use DTRUW, you should perform the following general steps:

1. Start DRTUW.
2. Set your reading options, which include:
   - Identifier matches, ranges, and qualities
   - Name width
   - Number of lines to keep
   - Speed for displaying data
3. Open an RTU file, which begins the reading process.
4. Save the text read, if desired.

To set reading options

1. From the DRTUW main menu, select **View > Options**.
2. Use the **Options** dialog box to perform the following tasks:

To	Do this
Specify a match	Next to Match, select the text strings to match with SCADA IDs within the RTU file. You can use wildcards to broaden your search. For more information, see "Wildcards" on page 52.
Specify a range	In the Select Range area, choose either the **By Time** or **By Point Number** option. Next to **Begin** and **End**, type in a beginning and/or ending range as needed.
Specify qualities	In the Qualities area, check the qualities you want to display. Available choices are **Good**, **Bad**, **Manual**, and **NoChange**.

To	Do this
Specify the wait time between retries	In the Retries area, type a value for the desired **Wait** time (ms).
Specify the number of times DRTUW should try reading a point	In the Retries area, in the **Max trys** text box, type the number of tries.
Locate timestamp anomalies	In the Locate Timestamp Anomalies area, select from the following options: • **Cusps(s)** – DRTUW will print only the points that are adjacent to a cusp of the specified size or larger. Selecting this option will ignore the MATCH setting. • **Gaps(s)** – Based on the current MATCH settings, DRTUW print only lines that would have been printed, with adjacent lines having a time-stamp separated by at least the specified amount. • **Any tag** – Select this option to modify Gaps so that any time any single ID has the specified time gap, the points bounding the gap are printed.
Specify the number of SCADA point name characters to display for the name width	Next to **Name Width**, click the arrow buttons to increase or decrease the number, as appropriate.
Auto Scroll	Clear this check box if you have a slow connection and want to speed up reads. Use the scroll bar to move through the file.
Save Options	If selected, saves the current options when you exit DRTUW.
Specify the number of times slower to display the data	Next to **Slow Down**, type the desired integer value.
Specify the number of lines to maintain while DRTUW reads the RTU file	Next to **Lines to Keep**, type the number of desired lines.

3 Click **OK**.

To start an RTU file read

Select **File > Open**.

Note: RTU files use a DT extension.

Once an RTU file is opened, the reading process is automatically initiated.

To control the reading process

Select **View > {command}**.

Available commands include:

- *Pause*—Pauses the reading, such that it can be resumed from the same spot.

- *Continue*—Resumes a paused reading.
- *Stop*—Stops the reading. A stopped reading cannot be resumed from the stop point.
- *Restart*—Restarts the reading from the beginning.

To save a text read

Select **File > Save As**.

Note: Text read files are saved with a TXT extension.

RTUMERGE

```
RTUMERGE
-READ=input-file-list    /* (s1,s2,...,sn)
-WRITE=output-file       /* s1
[-MAXRECS=max-points-in-output-file]     /* integer
[-TIMEOUT=time-out-period-list (minutes)]    /* (r1,r2,...rn)
[-ECHO]
[-MEMSUM=memsum-after-these-points]    /* integer
[-QUIT]     /* quit when all timed out
[-H]
```

Field	Description
input-file-list	The list of the RTU data files to be read.
output-file	The name of the output file.
max-points-in-output-file	Maximum number of data points stored in the output file. RTUMERGE will write MAXRECS points to the output file and will then begin to write over the oldest existing data {1000000}. Once the number of data points has been selected for a given output file, it cannot be changed.
time-out-period-list	Defines the amount of time RTUMERGE will wait for additional data before it goes on without the data from the file that has timed out. There may be a different timeout period associated with each input file. {300 seconds}
ECHO	Print out each point as it is read and/or written. The name of the input file is written in the last column as the point is read. A W is written in the last column if the point is being written to the output file.
memsum-after-these-points	Every memsum number of points give a summary. {no summary}
QUIT	Quit when all the read files have timed out.

Description

RTUMERGE is a utility that combines the contents of two or more RTU data files into a single RTU data file. This utility is used with either a Statefinder or Leakfinder model, both of which may require input data from two or more sources. For example, this utility can be used with a single Statefinder model that requires dynamic online data, which is maintained by two separate SCADA systems. Each SCADA system is configured to generate an RTU data file recording the data from its database, which is required for the model. RTUMERGE runs continuously, reading the two SCADA generated RTU data files, merging the contents of the files into single file. RTUMERGE works by having a table of "current" points from each input file. RTUMERGE copies the point with the smallest time tag to the output file, and then gets a new point from the corresponding input file.

Note: With the RPC version of the API, RTUMERGE is no longer needed.

Take note

Restarts

RTUMERGE does a good job of restart if it is shutdown. RTUMERGE opens the output and finds the newest time. It then finds the time in the input and starts reading from there.

Output

RTUMERGE displays the contents of the RTU data file in the following format when the ECHO option is specified:

```
(Point No.   Date     Time     SCADA Point Name   Value  )

1231        91/07/21  00:42:44  STA535.PRES.DISC  757.89  W
125000      91/07/21  00:42:44  STA535.PRES.DISC  757.89  RTUDATA1.DAT
125001      91/07/21  00:42:44  STA535.PRES.SUCT  590.45  RTUDATA1.DAT
1232        91/07/21  00:42:44  STA535.PRES.SUCT  590.45  W
125002      91/07/21  00:43:04  STA520.FLOW.STD  1987.12  RTUDATA2.DAT
1233        91/07/21  00:43:04  STA520.FLOW.STD  1987.12  W
```

CHAPTER

18

Trainer

Trainer combines the same advanced transient modeling features found in SPS with the ability to run in real-time. It can also be interfaced to SCADA display software to provide a realistic environment for training new or experienced operators. A software interface must be developed when simulation results are sent to SCADA display consoles.

This document supplements the main *SPS Help and Reference* and provides necessary information to build training models and use Trainer. For more information on interfacing to SCADA display software, see "Interfacing SPS to External Applications" on page 925.

Overview of Trainer features and commands

Trainer is a proven product built on the modeling technology used in the Stoner Pipeline Simulator (SPS), a general-purpose off-line engineering simulator for either liquid or gas networks.

Trainer can be used in a stand-alone, off-the-shelf environment or can be interfaced to a pipeline SCADA console to achieve the most realistic training experience possible.

Trainer provides high-fidelity simulation of pipeline operations. It realistically simulates fluid hydraulics and equipment performance, as well as the operator control interface for the actual pipeline. The operator may not be able to determine whether he is operating the pipeline or the simulator.

Trainer uses the same approach to modeling pipeline equipment and hydraulics as SPS, but adds special operator interface features and capabilities to enhance the training environment.

Trainer simulates all the control interactions that an operator makes in day-to-day operations: start or stop compressor or pump units, change pressure and flow set points, and open or close valves. Furthermore, Trainer simulates the in-sequence controls performed by SCADA Remote Terminal Units (RTUs) during compressor or pump start and stop operations, such as timing of valve activation.

Trainer may be used as a 1-on-1 teaching tool, individual study aid, tool to instruct many operators, or performance evaluation aid. An evaluator may work with an operator, giving hints, answering questions, and interacting with the simulator to challenge the operator.

Pre-programmed scenarios may be used to pace new operators through a controlled curriculum, providing the equivalent of years of pipeline operations experience in the shortest possible time. Experienced personnel can develop techniques for pipeline control during emergency conditions that may only occur once every several years. Such contingency operations can be simulated without risk to the actual pipeline.

These scenarios can be rapidly developed. Starting with an interactive session, the evaluator enters commands to define the scenario. These commands are logged to a REPLAY file in a clear text format. This enables the REPLAY file

to be edited to change any of the logged commands. To further develop the scenario, the evaluator simply replays the previous (or edited) scenario and adds to the scenario interactively. Both the replayed commands and the interactive commands are logged to a new REPLAY file, so that the training scenario can be progressively and simply developed.

The operator may interact with the simulation through the SCADA console to respond to the scenario in progress. All operator interactions with the training simulator are logged for review and analysis. The logs are recorded in a compatible format with the pre-programmed scenario files so that a training session can be replayed in entirety during a counseling session with the operator.

The evaluator can control the simulation speed, speeding up the pipeline response during quiet periods and slowing down the pipeline during periods of heavy activity, during which the operator needs the maximum time to recognize and react to pipeline conditions, or where the evaluator wishes to elaborate on particular operator responses or detailed hydraulic behavior.

Trainer may also be used to test the implementation of new SCADA display software or displays before they are used.

Features specific to Trainer

The following features are specific to Trainer:

- Runs in real-time (clock time) or multiples of real-time.
- The simulation does not automatically pause after commands are entered.

Trainer attempts to match the time step it is taking to the elapsed time for the time step, e.g. a 6 second time step would take 6 seconds of real-time. During times where significant activities and transient are occurring, the model time step may fall behind (lag) real-time. When conditions permit, the model will take large time steps to catch up to real-time. The evaluator also has the option of changing the speed of the simulation. For more information, see SPEED and LAG in "GLOBALS (GB) attributes" on page 665. If the evaluator sets a minimum time step that is greater than the time it takes Trainer to perform a step, Trainer will *sleep* between time steps. This will keep Trainer in real-time.

A Trainer model has *sticky mode* turned off by default. When sticky mode is turned off, Trainer does not pause after a command is given but rather continues to run. This allows the model to run in real-time. If the model is paused, it may need to take large time steps to catch up with real-time.

Stand-alone and SCADA-interfaced Trainers

An operator can interact with Trainer through:

- SCADA console
- A terminal using displays to interact with the model and display results

Using a SCADA console, the operator can practice training scenarios and use the same SCADA displays and keyboard used in daily operations. The Trainer model provides the equivalent information that the RTUs provide to the SCADA system, thus allowing the operator to see the same information that would be available during actual operations. This method requires a software program to interface Trainer with the SCADA system.

An operator can interact with Trainer through displays that are similar to or mimic the actual SCADA displays. This method requires interfacing the Trainer model to a graphical user interface (GUI). The modeler or evaluator may create

custom displays to display model information and facilitate model interaction. The use of custom displays eliminates the need for the operator to learn SPS commands.

For more information, see "Interfacing SPS to External Applications" on page 925.

Using Trainer

Trainer may be used to train on general hydraulic principles, the specifics of operating a given system, or the operation of a SCADA system. The amount of detail and the display of information varies, depending on the type of training to be done. This section will discuss the amount of detail required for the models and how to activate a training session.

Level of detail for Trainer models

The level of model detail depends on the type of training. Training on general hydraulic principles does not require the level of detail that training on a specific system or in the use of a SCADA system requires. A model intended to train on a specific piping network requires accurate representation and excellent tuning to mimic the actual piping network. The same is true for training to use the SCADA system. If the model does not behave in the same manner as the actual network, the operators have confidence in neither the modeling process nor the training process.

Therefore, the input for a Trainer model is usually more detailed than for a standard SPS model. The basic principal is to develop a model that mimics the actual pipeline network. This usually entails:

- Developing DEFINEs for non-hydraulic SCADA information.
- Developing model logic to mimic field control logic.
- Defining ALARMS that mimic SCADA alarms.
- Developing scenarios for the training sessions.
- Developing displays for the evaluator and/or operator use.

Each of these points is described in more detail in this section. All of the detail described here does not need to be in the initial model, but may be added in layers until the model is complete.

Non-hydraulic information

In general, there is non-hydraulic information (information that is not pressure-flow related) that affects the operation of the pipeline and the duties of the operators. This information varies depending on the network being modeled. It may include such information as intrusion alarms, fire alarms, battery charge status, RTU status and ambient temperature. This type of information may be modeled using DEFINEs. The DEFINEs may be ramped or poked to change their value during each scenario. This feature allows the evaluator to observe the operator's reaction to non-hydraulic events.

Mimic field control logic

Trainer has the capability of mimicking field control logic. In general, the RTUs and/or PLCs at a site control a majority of the site events. For instance a start unit command from the SCADA system may be broken into several commands at the RTU level. For example, the RTU may check the status of the unit suction valve and open it if needed, check the status of the unit discharge valve and close it if necessary, start the unit, and wait before opening the discharge valve. INTRAN logic may be used to mimic such events as unit starts, unit stops, emergency shutdown (ESD), high pressure

shutdowns, and low pressure failovers. The more realistically the control logic is modeled, the more faith the operators will have in the training sessions. This can be done using control elements, WHENEVERs and DEFINE.SEQUENCEs.

Alarms

SCADA system alarms may need to be modeled if Trainer is not interfaced to a SCADA system. This would provide standard alarms for the training sessions. Alarms may also be modeled to inform the evaluator that an alarm has occurred. Also, model generated alarms are logged to the ALMLOG file and may be kept as part of the training record. The modeling can be done using the ALARM and ALARM.CATEGORY commands.

Scenario definition

Probably the most difficult part of the modeling process is to define realistic modeling scenarios. The catastrophic scenarios, such as line breaks, power failures or loss of supply are easy to define. The difficult scenarios are those that mimic daily operation and the normal operator functions. If the operator functions vary depending on the shift being worked, then this variance needs to be included in the scenarios. If pipeline operation is temperature dependent, the training scenarios should incorporate the temperature dependencies. It may be necessary to have several base models of the network that account for different operation depending on flowing temperature and seasonal network configuration. There is no absolute correct method for setting up the training scenarios. Each network is unique and the scenarios must reflect that uniqueness.

Training scenarios may be predefined and require no action from the evaluator, the evaluator may interactively initiate each event, or a combination of the two approaches may be used.

Displays

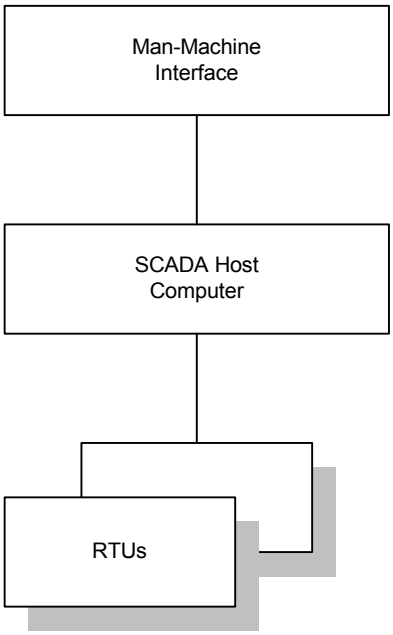
It is useful to develop evaluator displays that may be used to monitor the simulation and to initiate scenarios. At a minimum, text displays may be used by the evaluator. For Trainer systems that do not use the SCADA consoles for training, displays will need to be developed for the operator also. The displays can be simple text displays similar to the evaluator displays, or graphic displays that mimic the actual SCADA displays. The graphic display may be created with a GUI.

Running a training session

Trainer uses the same commands used to run SPS. However, the commands to start a training session will vary depending on the operator user interface. For systems using SCADA consoles for training, the exact sequence of commands will depend on the manner that the system is interfaced. In all cases, a TRANS session must be active for the training session.

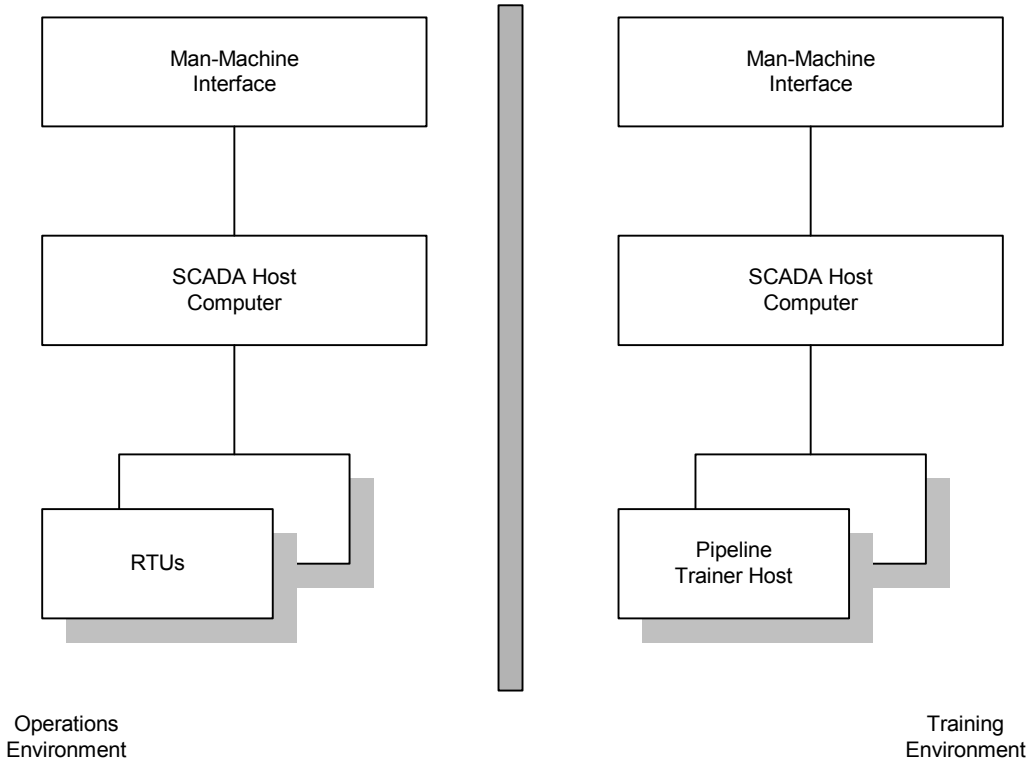
SCADA interface

Pipeline operators communicate with the pipeline through a Man-Machine Interface (MMI) that is connected to a SCADA host computer. The SCADA host system typically communicates with field-installed equipment through Remote Terminal Units (RTUs) and does so over a communications link. This relationship is shown as follows:



With a SCADA-interfaced Trainer installation, the SCADA host environment is duplicated; there is a copy of the SCADA host hardware, the SCADA software, and the SCADA Man-Machine Interface.

Trainer configuration differs from the actual SCADA system in that the connection between the SCADA system and the RTUs is interrupted, generally at the data acquisition level of the SCADA host system. Trainer is connected to the communications port of the duplicate SCADA host system and mimics the RTU side of the communications protocol in force. This relationship is shown in the following diagram:



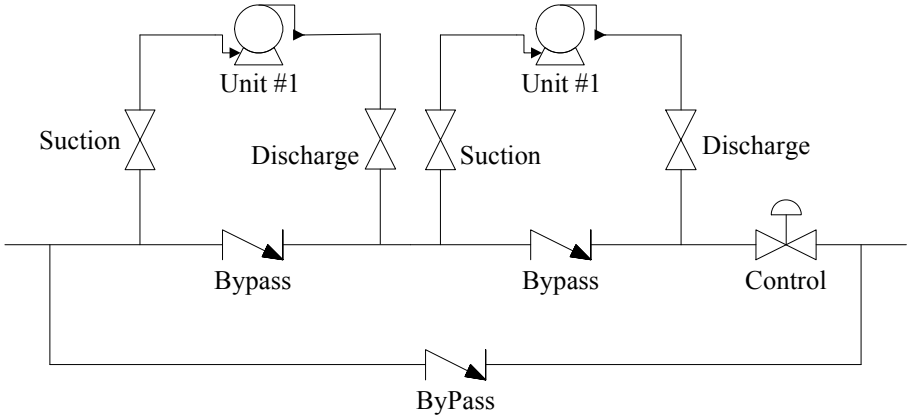
Notice that the SCADA host hardware and software used for the training environment are *completely separate* from the SCADA host used for controlling the pipeline. While the SCADA host computer and Trainer host are shown as separate computers (this is the most common configuration), it is not mandatory that they be separate. Typically, the SCADA software and Trainer software can be run as separate SPS instances on the same computer.

If the operational SCADA system includes a standby computer, available for use as an online SCADA system if the primary computer fails, then this standby computer could be used as the SCADA host for the training environment. The physical connection between the SCADA host and the Trainer host can be a simple RS-232-C serial line interface, or it could be a high-speed Ethernet (IEEE 802.3) or token ring interface. Data throughput requirements will typically determine the mode of physical connection.

Background information on SCADA systems

The equipment in the field

Consider the perspective of a real world pump/compressor station. This station has 2 in-series units that both have a suction valve, a discharge block valve, and a bypass check valve. In addition, the station has a bypass check valve and a station control valve. This configuration is shown as follows:



Measurements and controls

Various measurements and controls, interfaced to the SCADA host computer via a communications link and an RTU, are available at this station. On typical SCADA systems, these measurement and control points are typically identified using a naming convention such as:

```
<stn_name>,<pnt_name>
```

where <stn_name> refers to a specific geographic collection of measurement and control points or to a set accessed through a specific RTU, and <pnt_name> describes a specific measurement or control point at this station or RTU. Note that most operating companies adopt a further convention with respect to how <pnt_name> actually describes the measurement or control. For example, the <pnt_name> for all unit status indicators might be formulated by appending the characters STS to the unit name. In this example, AA is the station name. The SCADA measurements are defined as follows:

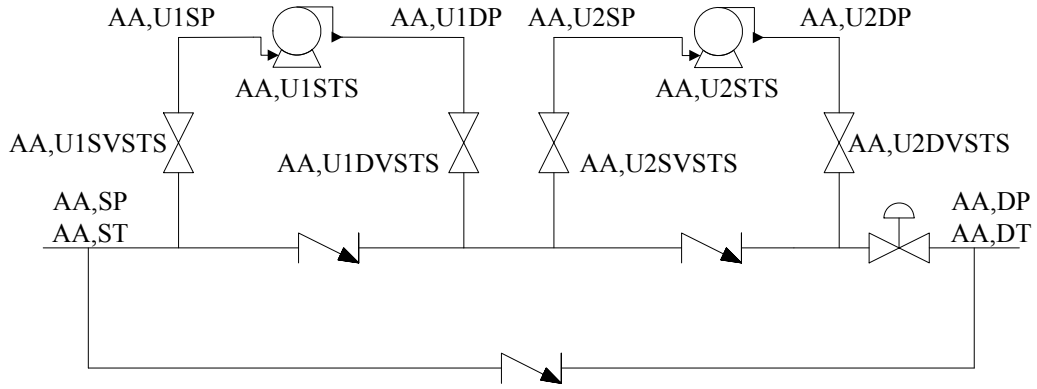
Analog Measurements

```
AA,SP          station suction pressure
AA,DP          station discharge pressure
AA,ST          station suction temperature
AA,DT          station discharge temperature
AA,U1SP        unit 1 suction pressure
AA,U1DP        unit 1 discharge pressure
AA,U2SP        unit 2 suction pressure
AA,U2DP        unit 2 discharge pressure
```

Status Measurements

```
AA,U1STS       unit 1 status
AA,U1SVSTS     unit 1 suction valve status
AA,U1DVSTS     unit 1 discharge valve status
AA,U2STS       unit 2 status
AA,U2SVSTS     unit 2 suction valve status
AA,U2DVSTS     unit 2 discharge valve status
```

The station schematic with the SCADA measurements follows:



The controls that are available through the SCADA system follow:

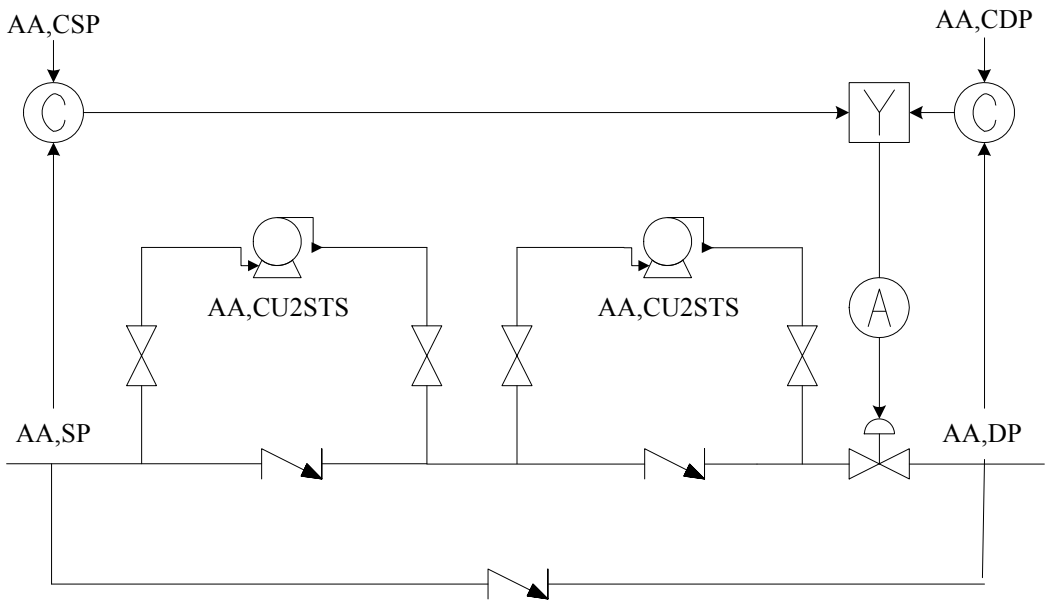
Analog Controls

```
AA,CSP          station minimum suction pressure
AA,CDP          station maximum discharge pressure
```

Status Controls

```
AA,CU1STS     unit 1 start/stop
AA,CU2STS     unit 2 start/stop
```

The station diagram with controls is:



The suction and discharge pressure control system elements shown here are the controllers (C) and the select relay (S). In practice, a real station has more elements, but the diagram has been simplified for the purposes of this discussion.

Notice that on this system the status of the suction and discharge valves are sent back to the central SCADA system, but the SCADA host system cannot control these valves. Instead, the local RTU can control these elements in association with commands to start and stop the units.

The data acquisition process

The SCADA system communicates with this station by sending out a request for data (poll) to the communication line associated with this RTU. Between the SCADA system and the RTU, communication is not typically by names, but rather by numbers, the <stn_id>. The request for data message is typically issued for each RTU in turn, and each RTU responds with all of its analog and status measurements in a predetermined order in the reply message.

The data format for the poll command to request a complete scan of the RTU might be of the form:

```
<stn_id>          16 bits
<chk_sum>         16 bits
<scan_stn>        16 bits
```

where <chk_sum> is a field used for verifying correct transmission of the data message (e.g. CRC, BCH, and so forth), and <scan_stn> is the command to this RTU to send all of its available data.

The RTU would then respond with a message in the following form (note the SCADA point names are used to describe the values in this message):

```
<stn_id>          16 bits
<chk_sum>         16 bits
AA,SP             16 bit integer, psia
AA,DP             16 bit integer, psia
<2 spare bytes>   16 bits
AA,ST             16 bit integer, 10*(° F)
AA,DT             16 bit integer, 10*(° F)
```

```
<2 spare bytes>    16 bits
AA,U1SP            16 bit integer, psia
AA,U1DP            16 bit integer, psia
AA,U2SP            16 bit integer, psia
AA,U2DP            16 bit integer, psia
<16 spare bytes>   128 bits
AA,U1STS           4 bits
AA,U1SVSTS         2 bits
AA,U1DVSTS         2 bits
AA,U2STS           4 bits
AA,U2SVSTS         2 bits
AA,U2DVSTS         2 bits
<6 spare bytes>    48 bits
```

Set point control

For the pipeline operator to control equipment in the field, the SCADA to RTU communications need to support set point control. A set point control action may require additional security and confirmation, so a 2-step *handshake* procedure is used. In a handshake procedure, the central control SCADA sends a message to the RTU. This message indicates that the SCADA will send a command to the RTU. In turn, the RTU responds (handshakes) and then the SCADA sends the command. The following example considers this case.

In response to an operator action, a message is first sent from the SCADA host to the RTU to *prime* the set point. This may be in the form:

```
<stn_id>              16 bits
<chk_sum>             16 bits
<prime_set point>     16 bits
<set point_id>        16 bits
<set point_value>     16 bits
```

To verify that the RTU has received the message correctly, the RTU sends a message back to the SCADA system in the form:

```
<stn_id>              16 bits
<chk_sum>             16 bits
<set point_primed>    16 bits
<set point_id>        16 bits
<set point_value>     16 bits
```

The SCADA host system checks that the <prime_set point> command has been received at the RTU correctly, and typically provides the pipeline operator the option of executing the set point command or canceling the set point request without affecting the current settings. In either case, the SCADA host issues a command back to the RTU to execute the set point command or to cancel it:

```
<stn_id>              16 bits
<chk_sum>             16 bits
<execute_set point>   16 bits          <cancel_set point>
<set point_id>        16 bits
<set point_value>     16 bits
```

Finally, the RTU acknowledges the <execute_set point> or the <cancel_set point> command and returns a confirmation back to the SCADA host that the appropriate action has been taken:

```
<stn_id>               16 bits
<chk_sum>              16 bits
<set point_confirm>    16 bits          <cancel_confirm>
<set point_id>         16 bits
<set point_value>      16 bits
```

Equipment status change

The equipment status change commands (unit start, unit stop) have a similar handshaking communications procedure as set point control requests. Quite often, these commands have a distinct set of message identifiers from those for a set point change.

```
<prime_stchange>      <prime_set point>
<stchange_primed>     <set point_primed>
<execute_stchange>    <execute_set point>
<cancel_stchange>     <cancel_set point>
<stchange_confirm>    <set point_confirm>
<cancel_confirm>      <cancel_confirm>
```

However, while these communications are transparently similar, they do have the additional complexity that a single start/stop command from the SCADA host will actually translate into a fairly complex sequence of operations at the RTU level.

For example, take a command to start unit #1 in the example station. This may cause execution of the following sequence of events at the RTU:

1. If the sequence is being executed, ignore the command.
2. If unit #1 is running, ignore the command.
3. Open unit #1 suction valve if not already open.
4. Close unit #1 discharge valve if not already closed.
5. Wait 3 minutes for the valves to complete movement.
6. Start unit #1.
7. Open unit #1 discharge valve.

The RTU controls the suction and discharge valves even though the specific control information does not originate from the SCADA host system; the *start* command indirectly requests this.

Background information on the Trainer

Building the Trainer model

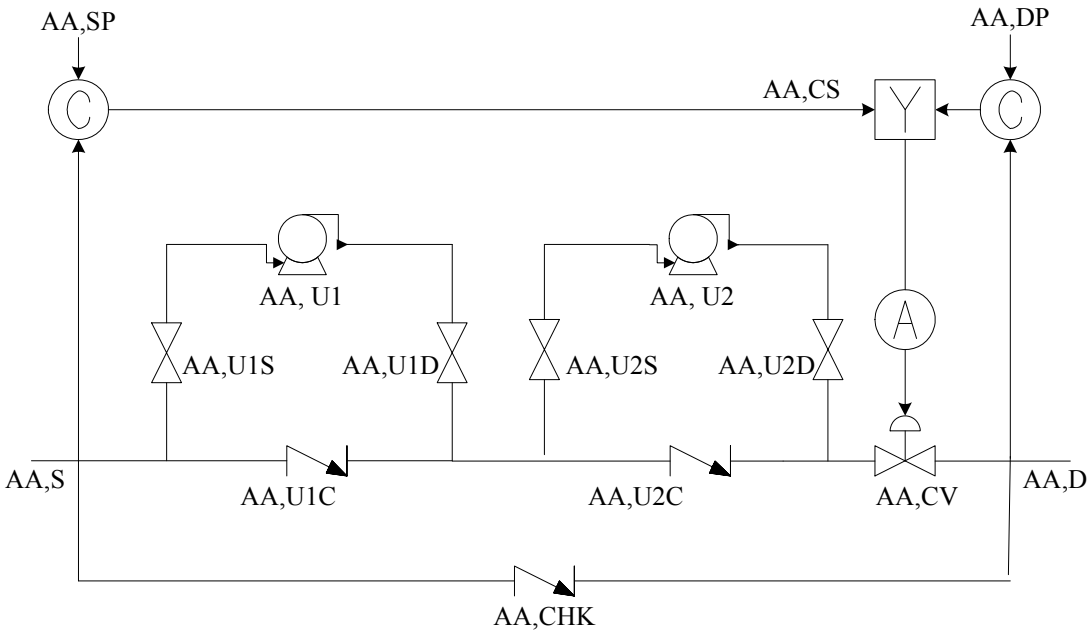
For Trainer, the element and connectivity definitions are maintained in the INPREP file. In the INPREP file, elements of each type automatically have an assigned set of variable names. For example, a pump/compressor unit automatically receives suction and discharge pressure variables and a status control. Consequently, the construction of a variable name in the Trainer requires an element name and a variable identifier, rather than just the discrete measurement name that is entirely adequate for the requirements of the SCADA system.

Because the station is named AA on the SCADA system, the Trainer data sets use AA as the identifying first characters of the elements associated with this station. In addition, the period (.) character aids legibility of the element names.

The 2 units associated with the station are called U1 and U2, respectively. Each unit has a suction and discharge valve (UnS, and UnD) and a bypass check valve (UnC). The station has a bypass check valve (CHK) and a control valve (CV). This discussion also requires the following control system elements: the suction pressure controller (SP), the discharge pressure controller (DP), and the select relay (CS). Finally, the connection points (nodes) at the suction and discharge sides of the station are named S and D, respectively.

Even though the SCADA system and the RTUs are completely unaware of the bypass check valves and the station control valve, the Trainer model needs to have these elements included to properly simulate the effect on the fluid hydraulics induced by these elements.

The following diagram shows the station:



For the purpose of this example, only the pertinent hydraulic variables generated in the RESTRT file are used (note the Trainer convention of referencing hydraulic variables as <device name>:<variable name>):

For a pump/compressor unit:

P- is the suction pressure (e.g. AA.U1:P-)

P+ is the discharge pressure (e.g. AA.U2:P+)

ST is the unit status (e.g. AA.U1:ST)

For a node:

P is the pressure (e.g. AA.S:P)

T is the temperature (e.g. AA.D:T)

For a valve:

ST is the valve position (e.g. AA.U1S:ST)

For a controller:

SP is the current set point (e.g. AA.DP:SP)

Command sequences

The pipeline model has been configured. However, before discussing the SCADA system interface, consider the implications of the RTU logic associated with starting and stopping units. Recall the start unit command discussed earlier for this illustration.

The SCADA system issues only the start command. The SCADA system has no knowledge of the logic (for checking valve positions, issuing commands to open or close the valves, and so forth) embedded in the RTU. However, for a realistic simulation, Trainer must mimic these details. This is done using a construct known as a DEFINE.SEQUENCE.

You can mimic the RTU logic defined for the starting of unit #1 by using a DEFINE.SEQUENCE command. The sequence of events to be mimicked, from the above example, is:

1 If the sequence is being executed, ignore the command.

2 If unit #1 is running, ignore the command.

3 Open unit #1 suction valve if not already open.

4 Close unit #1 discharge valve if not already closed.

5 Wait 3 minutes for the valves to swing.

6 Start unit #1.

7 Open unit #1 discharge valve.

This can be readily mimicked as follows. First DEFINE the simulation variable AA.U1.INSEQ which is given the value 0 if the sequence is inactive, or 1 if the sequence is already active. The value of this variable is checked in the DEFINE.SEQUENCE to determine if the sequence is in progress so that multiple start/stop commands are not issued. The start sequence is defined as follows:

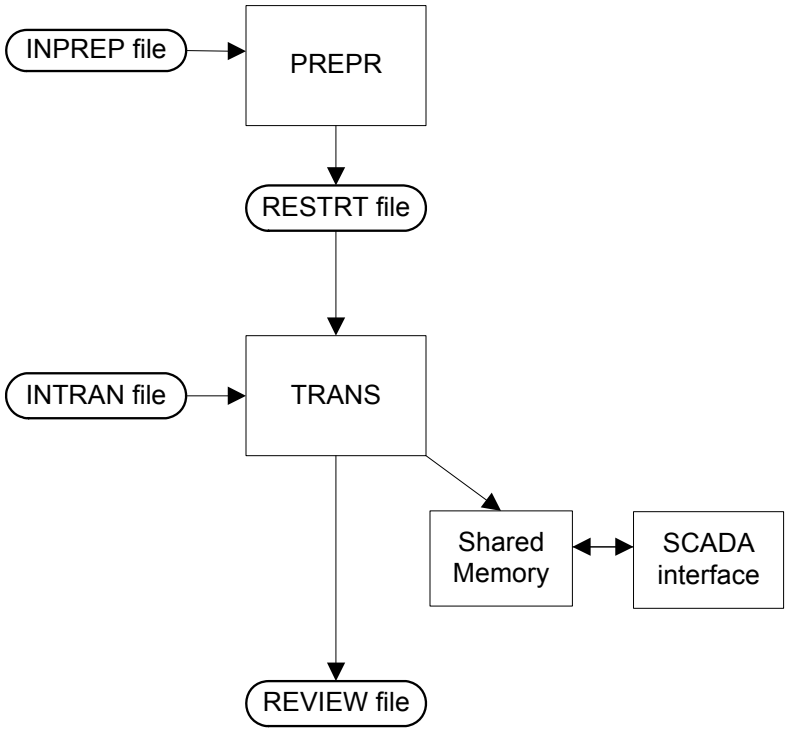
```
DEFINE AA.U1.INSEQ = 0                  /* sequence inactive
DEFINE.SEQUENCE START_AA.U1 {
     IF(AA.U1.INSEQ = 0)   {
     IF(AA.U1:ST != STARTING |  AA.U1:ST != RUNNING) {
          SET AA.U1.INSEQ = 1
          IF (AA.U1D:ST != CLOSED & AA.U1D.ST != CLOSING ) {
                CLOSE AA.U1D }
          IF (AA.U1S:ST != OPENED & AA.U1D.ST != OPENING ) {
               OPEN AA.U1S }
          WAIT 3
          START AA.U1
          OPEN AA.U1D
          SET AA.U1.INSEQ = 0
               } }
          }
```

During the simulation, if this sequence is to be executed interactively, it can be done by typing the following command at the > prompt:

```
> START_AA.U1
```

System flow for Trainer

Trainer communicates bi-directionally with the SCADA interface software through shared memory. For more information, see "Shared memory" on page 57 and "The SCADA interface for Trainer" on page 889. The following diagram shows shared memory and the SCADA interface in the context of the SPS system flow.

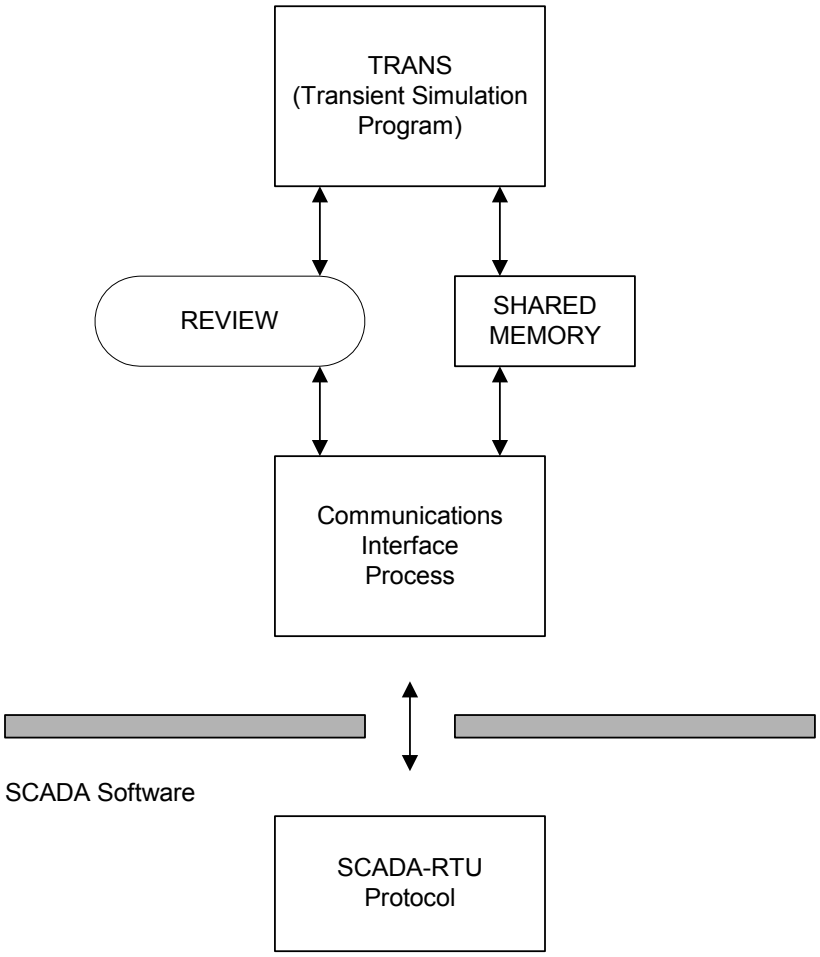


Trainer System Flow Diagram

For descriptions of the other programs and files in the diagram, see the "Glossary" on page 1053.

The SCADA interface for Trainer

Now that the two sides of the interface have been established, namely the typical structure of messages between the SCADA host and the RTUs, and the files used by Trainer, the discussion now turns to the interface itself. From the prior descriptions, the data flows available to the interface software have been shown to be the following:



Pipeline Trainer Software

In summary:

- The Communications Interface Process (CIP) is required to support the RTU side of the SCADA-RTU protocol.
- The data available to the interface software on the Trainer side is a read-only access to the REVIEW file.
- A read-write access to the data structures shared with the TRANS simulation program.

RTU-SCADA messages

In the SCADA host computer, data tables typically define for each <stn_id> the available control points and the predetermined format of the reply to a <scan_stn> command. These tables are generally maintained in a form that refers to SCADA point names so that they are readily understood by the system support staff.

This information needs to be made available to the CIP so that it can properly format messages both to Trainer and to the SCADA host. From the example used earlier, the SCADA table for the 2-unit station might be:

```
<scan_stn>
AA,SP                    16 bit integer, psia
AA,DP                    16 bit integer, psia
```

```
<2 spare bytes>          16 bits
AA,ST                    16 bit integer, 10*(° F)
AA,DT                    16 bit integer, 10*(° F)
<2 spare bytes>          16 bits
AA,U1SP                  16 bit integer, psia
AA,U1DP                  16 bit integer, psia
AA,U2SP                  16 bit integer, psia
AA,U2DP                  16 bit integer, psia
<16 spare bytes>         128 bits
AA,U1STS                 4 bits
AA,U1SVSTS               2 bits
AA,U1DVSTS               2 bits
AA,U2STS                 4 bits
AA,U2SVSTS               2 bits
AA,U2DVSTS               2 bits
<6 spare bytes>          48 bits

<execute_setpoint>

AA,CSP
AA,CDP

<execute_stchange>

AA,CU1STS
AA,CU2STS
```

This table needs to incorporate any validation or scaling parameters that are available at the RTU. For example, the analog set points AA,CSP and AA,DSP may be checked for permissible range at the RTU, and the temperature measurements AA,ST and AA,DT have a scaling factor applied.

Cross reference table

Perhaps the most obvious element that is still missing from this environment is the mechanism for the CIP to be able to match a SCADA point reference to a Trainer model variable reference. For example, using the above tables, the CIP needs to be able to interpret these as:

```
<scan_stn>
AA.S:P               16 bit integer, psia
AA.D:P               16 bit integer, psia
<2 spare bytes>      16 bits
AA.S:T               16 bit integer, 10*(° F)
AA.D:T               16 bit integer, 10*(° F)
<2 spare bytes>      16 bits
AA.U1:P-             16 bit integer, psia
AA.U1:P+             16 bit integer, psia
AA.U2:P-             16 bit integer, psia
AA.U2:P+             16 bit integer, psia
<16 spare bytes>     128 bits
AA.U1:ST             4 bits
AA.U1S:ST            2 bits
AA.U1D:ST            2 bits
AA.U2:ST             4 bits
```

```
AA.U2S:ST            2 bits
AA.U2D:ST            2 bits
<6 spare bytes>      48 bits

<execute_setpoint>

AA.SP:SP
AA.DP:SP

<execute_stchange>

AA.U1          "START_AA.U1"     "STOP_AA.U1"
AA.U2          "START_AA.U2"     "STOP_AA.U2"
```

Note that for the <execute_stchange> entries, the CIP requires a DEFINE.SEQUENCE name for each of the states that can be commanded for this control point.

The definitions of the SCADA-RTU message formats and the correspondence between SCADA point names and Trainer model variable names are stored in the INXREF file.

The actual correspondence between the SCADA point names and the Trainer model variable names need to be manually defined, unless you have a rigorous and unambiguous convention used for the SCADA point names that is carried over in some way to the names assigned to the modeling elements. If the latter is the case, the INXREF file can be generated by a program described in “Trainer example - Automated INXREF file generation” on page 918.

INTRAN file input for Trainer models

This information supplements the main documentation for SPS. This section describes changes to the INTRAN file necessary for running Trainer.

SELECT TRAINER

INTRAN

```
SELECT TRAINER
```

Description

The SELECT TRAINER command is used to select Trainer. This command turns *sticky mode* off (the simulation automatically continues running after any command is given) and attempts to match clock time, i.e., an hour of simulation takes an hour of actual time. This command must be entered in the INTRAN and INPREP file. A SHARE command and a TRENDLIST command must also be included in the INTRAN file to enable the shared memory interface. For more information, see "SHARE" on page 524 and "TRENDLIST" on page 537.

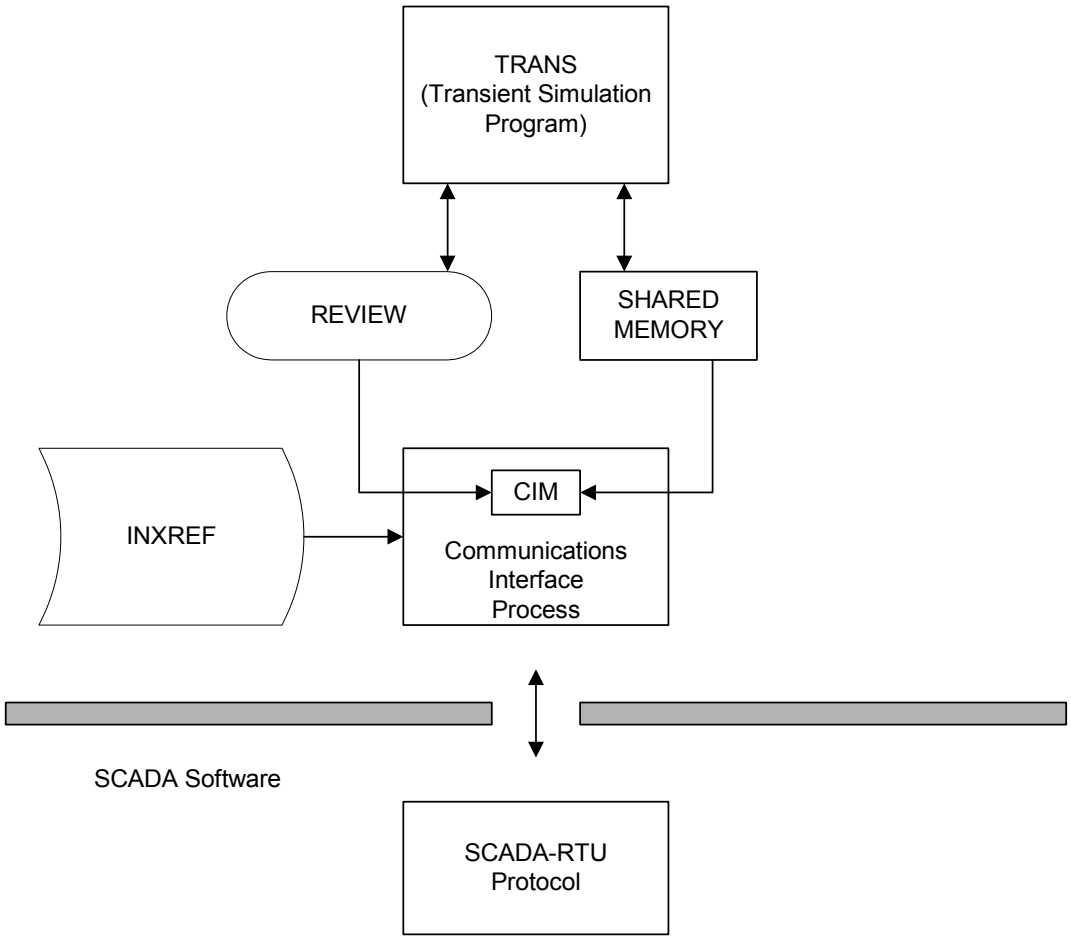
Example input

To run a Trainer scenario.

```
SELECT TRAINER
```

INXREF file input for Trainer models

This section describes the syntax of the INXREF file. A system flow diagram is included here for reference:



Pipeline Trainer Software

This interface has been developed with emphasis on ease of use and maintainability. The interface is designed to build on functionality that is a standard part of GL's modeling software. Because GL is committed to upward compatibility, the interface presented here will continue to work with later releases of the modeling software.

The CIM routine must perform three separate tasks for each message from the SCADA software. First, CIM must recognize the received command. Second, it must pass the corresponding action (if any) to TRANS for execution. Third, CIM must respond to the command with the appropriate reply. This functionality is data-driven so that you can change the behavior of the software as the pipeline configuration changes.

To achieve the required functionality, the CIM routine reads the INXREF file on startup. The INXREF file is message oriented; each COMMAND statement in the INXREF file contains a description of a particular type of message as well as the action(s) for this command.

The various statements in the INXREF file can appear in any order, provided the items are defined before they are used. Also, MACROs and INCLUDEs may be used.

Data types used in COMMAND and REPLY statement messages

The COMMAND and REPLY statements contain descriptions of messages; the messages are sequences of fields, with each field having (possibly) a different data type.

The following data types are supported:

BIT	Single bit, 0 or 1
UBYTE	Unsigned 8-bit integer, 0 ... 255
SBYTE	Signed 8-bit integer, -128 ... 127
ULINT2	Unsigned 2-byte integer, least significant byte first, 0 ... 65535
UMINT2	Unsigned 2-byte integer, most significant byte first, 0 ... 65535
SLINT2	Signed 2-byte integer, least significant byte first, -32768 ... 32767
SMINT2	Signed 2-byte integer, most significant byte first, -32768 ... 32767
SLINT4	Signed 4-byte integer, least significant byte first, -2147483648 ... 2147483647
SMINT4	Signed 4-byte integer, most significant byte first, -2147483648 ... 2147483647
REAL4	Single precision floating point value, using the floating point format of the machine that is running the interface
REAL8	Double precision floating point value, using the floating point format of the machine that is running the interface
STRING(len)	ASCII string of length, len. If a string is specified and it is not of length, len, it is truncated, or padded from the left to the length. The string is not null terminated and is case and space sensitive. The string must be enclosed in quotes (" ").
LIEREAL4	4 byte, least significant byte first, IEEE single precision floating point.
LIEREAL8	8 byte, least significant byte first, IEEE double precision floating point.
MIEREAL4	4 byte, most significant byte first, IEEE single precision floating point.
MIEREAL8	8 byte, most significant byte first, IEEE double precision floating point.
VAXREAL4	4 byte, vax single precision floating point.
VAXREAL8	8 byte, vax double precision floating point.
MVXREAL4	4 byte, most significant byte first, vax single precision floating point.

Note that the data type determines its size (1 bit; 1, 2, 4, or 8 bytes). All data types except BIT are on a byte boundary, with any required padding done automatically by setting the least significant bits to 0. This means that a byte that is defined with the BIT 1 statement has the same value as a byte that is defined with the UBYTE 128 statement.

INXREF commands

The following INXREF command are discussed in this section:

ARM	Sets a timer.
COMMAND	Identifies a SCADA message and defines the action(s) to be taken.
COMMENT	Comments may be added to the file.
DEFINE.FUNCTION	Used to set up schedules for varying information.
DEFINE	Create a user-defined variable.
EXPRESSION	Mathematical operators.
IF, ELSE IF, ELSE	Controls execution of various statements.
ISARMED	Checks to see if a timer is armed.
ISSUE	Sends a command to TRANS.
Peek fields	Generates strings that vary at run time with the value of an expression.
POKE	Used to change a local user variable.
REPLY	Sends a message to the SCADA software.

Variable substitution in INXREF file

The use of the value of INXREF variables in the body of INXREF statements is useful both in the control INXREF operation, such as in the use of if-conditionals, and in the formulation of commands to the TRANS program (via the ISSUE statement). The use of the [variable] construct permits the variable to be substituted by its value in text form, for DEFINEd variables, for variables identified as named arguments the COMMAND statement, and for DEFINE.FUNCTIONs.

For further illustration, the following examples show the command passed to TRANS both with, and without the use of the [variable] construct.

```
ISSUE Statement                        <trans-command>
ISSUE(POKE AA.DP:SP=DP_SETPOINT)       POKE AA.DP:SP=DP_SETPOINT
ISSUE(POKE AA.DP:SP=[DP_SETPOINT])           POKE AA.DP:SP=500
```

The first example would be a valid command if DP_SETPOINT were present in a DEFINE statement in the TRANS environment. The second example is valid if DP_SETPOINT is present in the INXREF environment, and assumes that DP_SETPOINT has a current value of 500.

Example

```
[DP_SETPOINT]           /* DEFINEd variable
[STXT(STA)]      /* DEFINE.FUNCTION variable
```

ARM

INXREF

```
ARM ( "name", duration )
```

Field	Description
name	Name of the timer
duration	Clock time, for the timer to be armed (minutes)

Description

The ARM statement is used inside a COMMAND statement to set a timer in the interface.

name is a string expression that specifies the name of the timer. It may contain peek attributes, enclosed in square brackets [ ].

duration specifies the clock time (not simulation time), in minutes, that the timer remains armed.

If the time is already armed, it will be reset to the new duration.

Example input

Arm a timer STA1 for a duration of 5.2 minutes.

```
ARM("STA1", 5.2 )
```

Use a variable to identify the station.

```
ARM("STA[STATION_NUMBER]", 5.2 )
```

COMMAND

INXREF

```
COMMAND (rtu-command) {
action | if-stmt
      ...
action | if-stmt
}
```

Field	Description
rtu-command	Command from the SCADA system to the RTU

Note: Each action contains at least 1 REPLY, POKE, ARM, or ISSUE statement.

Description

Each COMMAND statement identifies a message from the SCADA system and defines the actions to be taken whenever a message from the SCADA system matches rtu-command. IF, ELSE IF, and ELSE statements are used to specify different actions based on different conditions.

When the interface finds a message pattern that matches rtu-command, it executes the associated actions and if-statements. The interface responds with the (possibly concatenated) REPLYs, issues the command(s) from the appropriate ISSUE statements, executes the associated POKE statements, and arms timers, as specified in the selected actions for the particular rtu-command.

rtu-command permits wild carding, meaning that INXREF can be configured so that one particular rtu-command matches all commands of a specified pattern. Parameter substitution, which provides a convenient way for the interface to use elements of rtu-command in the various actions and if-statements, is also permitted.

The syntax of rtu-command is:

```
type_1 element_11, element_12, ...;
type_2 element_21, element_22, ...;
                ...
type_n element_n1, element_n2, ...
```

where each type_i is a data type (see "Data types used in COMMAND and REPLY statement messages" on page 895) and each element_ij has the following syntax:

```
[ name = ] { constant | ? | % }
```

where:

name	=	An optional element name, such as SOURCE, which identifies the element. If name is specified, it must be followed by an equal sign (=).
constant	=	The required value of the element.
?	=	Any value may appear.
%	=	Any value may appear and is optional.

Either constant, ? , or % should be used.

Examples of valid elements are:

23	Unnamed element; value is decimal 23
“XYZ”	Unnamed element; value is the character string XYZ
?	Unnamed element; any value
%	Unnamed optional element; any value
14x	Unnamed element, value is hexadecimal 14
SOURCE = 16	Element named SOURCE; value is decimal 16
SOURCE = “XYZ”	Element named SOURCE; value is character string XYZ
DEST = ?	Element named DEST; any value
DEST2 = %	Optional element named DEST2; any value (All optional elements must be input after all required elements; their values default to 0).

Take note

Named elements, such as SOURCE, may not be modified by POKE statements.

Example input

Define a command that matches a value of 2, a value of 3, an element named X with any value, and an element named Y with any value. Based on the value X and Y, change the value of XY; change the set point of AA to the new value, and reply with a value of 2, a value of 3, X and Y.

```
COMMAND(UBYTE 2, 3, X=?, Y=?){
     IF(X > 0){
          POKE XY = X * Y
           }
     ELSE IF(Y > 0){
          POKE XY = X + Y
           }
     ELSE{
          POKE XY = 0
          ARM("VALVE1",1)
           }
     ISSUE(POKE AAA:SP = [XY])
     REPLY(UBYTE 3, 2, X, Y)
     }
```

DEFINE.FUNCTION

INXREF

```
DEFINE.FUNCTION name ( arg_1, ... arg_n ) = expression
/*or
DEFINE.FUNCTION name, Y = y1 y2 ... yn, X = x1 x2 ... xn
```

Field	Description
name	Name of the defined function
arg_j	Arguments to be passed to the expression
expression	Valid expressions
$y_1 \ldots y_n$	Value associated with the corresponding y
$x_1 \ldots x_n$	Value associated with the corresponding x

Description

In a DEFINE.FUNCTION, one or more arguments are given between the parentheses to calculate expression. The alternate syntax defines a lookup table in which Y is linearly interpolated as a function of X.

A DEFINE.FUNCTION statement can be referenced in an expression or an if-conditional. The DEFINE.FUNCTION statement is not permitted inside a COMMAND statement.

A DEFINE.FUNCTION in the INXREF file is local to the INXREF file. Other processes such as TRANS cannot see it. A name that is defined in the INXREF file that conflicts with a name known to TRANS is resolved in favor of the INXREF name.

Example input

Define a function SQUARE that will calculate the square of a value passed to it:

```
DEFINE.FUNCTION SQUARE(X) = X ** 2
```

Define a function UPDOWN which has a value of 0 for an input of 1, a value of 1 for an input of 2, and a value of 0 for an input of 3:

```
DEFINE.FUNCTION UPDOWN, Y = 0 1 0, X = 1 2 3
```

DEFINE

INXREF

```
DEFINE name = expression
```

Field	Description
name	User-defined variable name
expression	Value to which name is set initially

Description

A DEFINE statement can be referenced in an expression or an if-conditional. It can also be reset with a POKE statement. The DEFINE statement is not permitted inside a COMMAND statement.

A DEFINE that is defined as equal to an expression takes the value of the expression at the time the definition occurs. DEFINE does not track the value of the expression like DEFINEs in the INTRAN file do. A DEFINE that is defined in the INXREF file changes only when a POKE statement resets it.

A DEFINE that is defined in the INXREF file is local to the INXREF file. Other processes such as TRANS cannot see it. A name that is defined in the INXREF file that conflicts with a name known to the TRANS program is resolved in favor of the INXREF name.

Example input

Define a variable AA.U1.INSEQ that has a value of 0:

```
DEFINE AA.U1.INSEQ = 0
```

Expressions

INXREF

```
expression
```

Field	Description
expression	Mathematical expression for use in DEFINE, POKE and DEFINE.FUNCTION statements

Description

Expressions are used in various places in the INXREF file. A brief description of expressions is given here. Expressions may consist of:

- Decimal and/or hexadecimal integer constants, such as 3, 24x, etc.
- Floating point constants, such as 16., etc.
- Peek names from the simulation, such as UNIT1:ST
- Named elements of rtu-command, such as DEST
- Mathematical operators, such as +, -, *, etc.
- Variable names or function names created using the DEFINE or DEFINE.FUNCTION statements

Decimal is the default for integer constants, hexadecimal numbers must start with a digit and end with suffix x or X. 90x and 0abcx are treated as hexadecimal numbers; abcx is not.

Expressions are evaluated using double precision floating point, and if necessary, converted to integer.

The INXREF language supports all of the usual operators and functions that are supported by SPS, such as +, -, *, /, MAX, MIN, etc. In addition to the standard operators and functions, the INXREF language also supports the ISARMED function.

IF, ELSE IF, ELSE

INXREF

```
IF (condition) { action }
ELSE IF (condition) { action }
ELSE IF (condition) { action }
   ...
ELSE { action }
```

Field	Description
condition	Relational test on numerical or status values
action	List of REPLY, POKE, ARM, IF or ISSUE statements

Description

IF, ELSE IF, and ELSE statements are used inside a COMMAND statement to control the execution of the various actions. Any number of IF and ELSE IF statements may be used inside a COMMAND statement; nesting of IF statements is permitted.

Example input

Within a command, check if PIPE1:P+ is greater than 300. If the condition is true, reply with values of 1, 0 and 0. Otherwise, check if X is greater than Y.

If it X is greater than Y, arm the timer XY.TIMER for 5 minutes and reply with values of 1, 0, and 0. Otherwise, check if the timer XY.TIMER is armed.

- If XY.TIMER is armed, reply with a value of 1. Reply with values of 0, 0, and 0; and issue a poke to STA1 set pressure.
- If XY.TIMER is not armed, reply with a value of 2. Reply with values of 0, 0, and 0; and issue a poke to STA1 set pressure.

```
IF(PIPE1:P+ > 300){
     REPLY(UBYTE 1, 0, 0)
     }
ELSE IF(X > Y){
     ARM("XY.TIMER", 5)
     REPLY(UBYTE 1, 0, 0)
     }
ELSE {
      IF(ISARMED("XY.TIMER")){
           REPLY(UBYTE 1)
           }
     ELSE{
           REPLY(UBYTE 2)
     }
     REPLY(UBYTE 0, 0, 0)
     ISSUE(POKE STA1:SP = [MAX(200, PIPE1:P+)])
     }
```

ISARMED

INXREF

```
ISARMED ("name")
```

Field	Description
name	Name of the timer. May contain peek attributes enclosed in square brackets, [ ].

Description

Within expressions, the ISARMED function tells whether or not a timer is currently armed. ISARMED(name) will return a value of 1 (true) if name is a currently armed timer or a value of 0 (false) otherwise.

Example input

Check to see if STA1 is armed prior to issuing additional commands:

```
IF(ISARMED("STA1")){...}
```

Use a variable to identify the station:

```
IF(ISARMED("STA[STATION_NUMBER]")){...}
```

ISSUE

INXREF

```
ISSUE ( trans-command )
```

Field	Description
trans-command	Any valid TRANS command

Description

The ISSUE statement is used inside a COMMAND statement to send a command to TRANS. Commands include Poking set points, submitting sequences, etc. The command is in Interactive format.

Example input

Change STA1 set pressure to 500:

```
ISSUE (POKE STA1:SP = 500)
```

Change STA1 set pressure to a value contained in an INXREF variable:

ISSUE (POKE STA1:SP = [NEW_SETPOINT])

Start unit A1, using the DEFINE.SEQUENCE START_A1:

```
ISSUE (START_A1)
```

Peek fields

INXREF

```
[expression]
/*or
[expression,width.decimal]
```

Field	Description
expression	Mathematical expression to be evaluated and converted to text
width	Width of the resulting text. If zero, or not specified, the text will be just large enough to hold the value of the expression, with no leading or trailing blanks.
decimal	Number of decimal places to be displayed for numeric results. If zero, or not specified, the number of decimal places will be controlled by FORMAT statements in INTRAN, or SPS default formatting rules.

Description

Within the INXREF file, peek attributes may be used to generate strings that vary at run time with the value of an expression. Peek attributes may be used within the name of ARM statements and ISARMED expressions. Peek attributes may also be used with the TRANS-command portion of ISSUE statements.

Use the second form of Peek Field to ensure that sufficient precision is used when changing set points.

Example input

Determine if a particular setpoint is armed and ready to be changed:

```
IF (ISARMED("ST[STA].[SETPOINT_ID]_PRIME") )
```

Issue a change to a set point, including four digits past the decimal point:

```
ISSUE(POKE ST[STA].[SETPOINT_ID]:SP = [SETPOINT_VAL,0.4])
```

POKE

INXREF

```
POKE name = expression
```

Field	Description
name	Value to be change
expression	A name, peek name, or arithmetic expression of numbers and peek names

Description

The POKE statement is used inside the COMMAND statement to set the value of an item in the interface that was declared with a DEFINE or the DEFINE.FUNCTION statement. Only local variables are set in this way. Use the ISSUE(POKE ...) statement to set a simulation variable.

Take note

POKE may be applied to elements created with a DEFINE statement. POKE may not be applied to named RTU-command elements in a COMMAND statement.

Example input

```
DEFINE STA1 = 0
COMMAND(UBYTE 2, 3, X=?){  /* Change the value of define STA1 to 1:
  POKE STA1 = 1

  /* Not valid.  Can't POKE named COMMAND elements
  POKE X = 2
  }
```

REPLY

INXREF

```
REPLY ( rtu-reply )
```

Field	Description
rtu-reply	Reply to be sent to the SCADA system

Description

The REPLY statement is used inside the COMMAND statement to send a message to the SCADA software. Multiple rtu-replies are concatenated, producing one longer reply.

The rtu-reply is the list of elements (i.e., data items) to reply to the sender. The syntax of rtu-reply is:

```
type_1 expression_11, expression_12, ...;
type_2 expression_21, expression_22, ...;
          ...
type_n expression_n1, expression_n2, ...
```

STRING values matched in a COMMAND cannot be converted to another data type in a REPLY. The conversion will result in a value of 0, no matter what the STRING value was.

Example input

Imitate the reply from an RTU to the SCADA system that is 3 UBYTEs with values of 0, 0, and 0.

```
REPLY(UBYTE 0, 0, 0)
```

Trainer examples

Trainer example - Scan station

Recall that in the example, the scan station command has the format:

```
<stn_id>              16 bits
<chk_sum>             16 bits
<scan_stn>            16 bits
```

The stn_id is AA, and for the sake of this example, the scan_stn command value is assumed to be 1. The following command recognizes the scan station command:

```
COMMAND(UBYTE 41x, 41x, ?, ?, 1, 0)
```

Note that 41x is hexadecimal for A. In addition, this example assumes that integers are transmitted with the least significant byte first. Equivalently, the following commands also recognize the scan station command:

```
COMMAND(UBYTE 41x, 41x; ULINT2 ?, 1)

COMMAND(ULINT2 4141x, ?, 1)
```

The first form treats the command as a stream of 6 bytes. The second form treats the command as 2 bytes followed by 2 unsigned 2-byte integers; and the third form treats the command as a stream of three 2-byte integers. Notice that in the second example the semicolon is used to separate lists of elements, while the comma is used to separate elements within each list.

After recognizing the command, a reply command must be generated. The reply has the format:

```
<stn_id>                16 bits
<chk_sum>               16 bits
AA,SP                   16 bit integer, psia
AA,DP                   16 bit integer, psia
<2 spare bytes>         16 bits
AA,ST                   16 bit integer, 10*(° F)
AA,DT                   16 bit integer, 10*(° F)
<2 spare bytes>         16 bits
AA,U1SP                 16 bit integer, psia
AA,U1DP                 16 bit integer, psia
AA,U2SP                 16 bit integer, psia
AA,U2DP                 16 bit integer, psia
<16 spare bytes>       128 bits
AA,U1STS                 4 bits
AA,U1SVSTS               2 bits
AA,U1DVSTS               2 bits
AA,U2STS                 4 bits
AA,U2SVSTS               2 bits
AA,U2DVSTS               2 bits
<6 spare bytes>         48 bits
```

An INXREF that will recognize the scan station command for station AA and generate an appropriate reply follows:

```
COMMAND(ULINT2 4141x, ?, 1) {
     REPLY(
          ULINT2     4141x,               /* station id
                     0,          /* check sum
                     AA.SP:SP     /* suction pressure setpt
                     AA.DP:SP     /* discharge pressure setpt
                     0,          /* spare
                     10*AA.S:T,     /* 10*(suction temperature)
                     10*AA.D:T,     /* 10*(discharge temperature)
                     0,          /* spare
                     AA.U1:P-,     /* unit 1 suction pressure
                     AA.U1:P+,     /* unit 1 discharge pressure
                     AA.U2:P-,     /* unit 2 suction pressure
                     AA.U2:P+,     /* unit 2 discharge pressure
                     0,0,0,0,     /* spare
                     0,0,0,0;     /* spare
          UBYTE      PVSTAT(AA.U1:ST, AA.U1S:ST, AA.U1D:ST),
                     PVSTAT(AA.U2:ST, AA.U2S:ST ,AA.U2D:ST);
          ULINT2      0,0,0          /* spare
          )
     }
```

Notice that the comma (,) is used to separate data elements and that the semicolon (;) is used to separate data types. This command assumes that the INPREP file has specified that the units for pressure are psia and that the units for temperature are degrees Fahrenheit (°F). This command also assumes that the PVSTAT function (which combines a unit status and 2 valve statuses into 1 byte) has been defined:

```
DEFINE.FUNCTION PVSTAT(U, V1, V2) =
+     16*UNIT.STAT(U) + 4*VALVE.STAT(V1) + VALVE.STAT(V2)
```

The PVSTAT function in turn uses the related functions UNIT.STAT and VALVE.STAT. Each of these functions maps the value of the unit or valve status in text form to an appropriate integer value. These functions should be defined prior to PVSTAT.

```
/* maps (STOPPED,STARTING,RUNNING,STOPPING) to (1,2,4,8)
DEFINE.FUNCTION UNIT.STAT(U) =
+     U == "STOPPED"  ? 1 :
+     U == "STARTING" ? 2 :
+     U == "RUNNING"  ? 4 : 8
/* maps (CLOSED,OPENING,OPEN,CLOSING) to (0,1,2,3)
DEFINE.FUNCTION VALVE.STAT(V) =
+     V == "CLOSED"   ? 0 :
+     V == "OPENING"  ? 1 :
+     V == "OPEN"  ? 2 : 3
```

The defined function UNIT.STAT is used to assign a numerical value for each of the 4 possible statuses. UNIT.STAT is the name given to the defined function and U is the name used for the argument that represents the status of a particular unit.

The ? , : syntax is interpreted as in the C programming language; UNIT.STAT is evaluated in the following manner:

1 If U == “STOPPED” is true (when U is replaced with the status of the unit) UNIT.STAT(U) evaluates to 1; otherwise the next line in the defined function is evaluated,

U == “STARTING” ? 2 :

2 If U == “STARTING” is true, UNIT.STAT(U) evaluates to 2; otherwise, the next line in the function is evaluated,

U == “RUNNING” ? 4 : 8

3 If U == “RUNNING” is true, UNIT.STAT(U) is returned as 4; otherwise, what comes after the colon (:), is evaluated, which is the value 8, so UNIT.STAT(U) is returned as 8.

The function VALVE.STAT is evaluated in a similar manner.

The function PVSTAT, which in turn uses function UNIT.STAT and VALVE.STAT, will have values between 16 and 143 depending on the statuses of the units and the 2 valves. For example, if unit AA.U1 is running and its suction and discharge valves, AA.U1S and AA.U1D respectively, are open, then the value of the function PVSTAT will be 74. This value is calculated as follows:

```
AA.U1:ST     == "RUNNING"     ==> UNIT.STAT(AA.U1:ST)     = 4
AA.U1S:ST      == "OPEN"     ==> VALVE.STAT(AA.U1S:ST)     = 2
AA.U1D:ST      == "OPEN"     ==> VALVE.STAT(AA.U1D:ST)     = 2
PVSTAT(AA.U1:ST, AA.U1S:ST, AA.U1D:ST) = 16*4 + 4*2 + 2 = 74
```

Note that the function PVSTAT produces a unique value for each of the possible combinations of unit and valve statuses. This allows the statuses of 3 pieces of equipment to be described in 1 byte.

Trainer example - Station set points

Trainer example - Prime station discharge pressure set point

For the example station, a set point change is a 2-step operation: the set point is primed, then it is changed. In this example, a set point will stay primed for 1 minute. The execute set point must come while the set point is primed or it will be ignored.

Recall that the command sent from the SCADA host to the RTU to prime the set point is of the form:

```
<stn_id>              16 bits
<chk_sum>             16 bits
<prime_setpoint>      16 bits
<setpoint_id>         16 bits
<setpoint_value>      16 bits
```

Again, assume stn_id is “AA”. Also assume that the prime_setpoint command takes the value “5” and that the setpoint_id field is “10”. Since the value the operator may request for the discharge pressure set point is not known in advance, this set point should be represented as a named element that may have any value (DP_SETPOINT = ?). The command to prime the set point would then be recognized by the statement:

```
COMMAND(ULINT2 4141x, ?, 5, 10, DP_SETPOINT = ?)
```

After the command is recognized, an appropriate reply must be generated. Recall that the reply to a prime set point command is of the form:

```
<stn_id>               16 bits
<chk_sum>              16 bits
<setpoint_primed>      16 bits
<setpoint_id>           16 bits
<setpoint_value>        16 bits
```

If the setpoint_primed> field is “20”, then an INXREF command that will recognize the prime set point command and generate a reply would be:

```
COMMAND(ULINT2 4141x, ?, 5, 10, DP_SETPOINT = ?) {
     REPLY(ULINT2     4141x,          /* station id
                    0,     /* check sum
                    20,     /* set point primed
                    10,     /* set point id
                    DP_SETPOINT)/* requested set point value
                ARM("AA.DP_PRIME", 1)/* set discharge pressure
                               /* set point timer to 1 min.
     }
```

Note that the previous command sets a timer, named AA.DP_PRIME, which will count for 1 minute. The status of this timer (whether 1 minute has elapsed or not) will be used in executing the set point.

Trainer example - Change discharge pressure set point

Previously, the discharge pressure set point of the example station is primed. Recall that for the station, the command to change a set point is of the form:

```
<stn_id>               16 bits
<chk_sum>              16 bits
<execute_setpoint>     16 bits
<setpoint_id>          16 bits
<setpoint_value>       16 bits
```

If the execute_setpoint field is “30” (and using the setpoint_id of “10” from the previous example), then an INXREF command that will recognize the change set point command will be:

```
COMMAND(ULINT2 4141x, ?, 30, 10, DP_SETPOINT = ?)
```

After the change set point command has been recognized, a reply must be generated. The reply will be either a confirmation that the set point change has been made, or a reply that the change was not made. The set point change will not be made if the operator waited too long to issue the change command and, consequently, the set point arming timer ran out. The reply to the execute set point command is of the form:

```
<stn_id>                16 bits
<chk_sum>               16 bits
<setpoint_confirm>      16 bits
<setpoint_id>           16 bits
<setpoint_value>        16 bits
```

The setpoint_confirm field is “40” if the set point has been changed, and is “45” if the execute_setpoint command has been rejected. The INXREF command that will recognize the change set point, generate a reply, and either execute the TRANS command to cause the set point to change in the model, or ignore the command, would be:

```
COMMAND(ULINT2 4141x, ?, 30, 10, DP_SETPOINT = ?) {
IF ( ISARMED("AA_DP_PRIME") ) {
     REPLY(ULINT2     4141x,          /* station id
                    0,     /* check sum
                    40,     /* set point change confirmed
                    10,     /* set point id
                    DP_SETPOINT)/* set point value
     ISSUE( POKE AA.DP:SP = [DP_SETPOINT] )
     }
ELSE {
     REPLY(ULINT2     4141x,          /* station id
                    0,     /* check sum
                    45,     /* set point change rejected
                    10,     /* set point id
                    DP_SETPOINT)/* set point value
     }
}
```

Notice that the INXREF command recognizes the change set point command then, based on the status of the set point arming timer, either changes the set point in the model (using the POKE command) and produces an affirmative reply, or simply produces a reply that states that the set point has been rejected.

Trainer example - Change any set point at any station

The previous examples presented were specific to a particular station, AA, and to a particular type of set point. These examples illustrate the basic ideas involved in the interface's interpretation of, and response to, messages from the SCADA host. However, the INXREF commands used in an actual interface can and should take advantage of features that allow for more general purpose interpretation of messages from the SCADA host.

Priming any set point

Previously, the INXREF command for the priming command was defined as:

```
COMMAND(ULINT2 4141x, ?, 5, 10, DP_SETPOINT = ?)
```

To make this command usable for stations other than "AA" and for set points other than discharge pressure, "4141x" (which is specific to station "AA") and both "10" and "DP_SETPOINT" (which are specific to a discharge pressure set point) must be replaced with elements that are general in nature. An example would be:

```
COMMAND(ULINT2     STA = ?,
                    ?,
                    5,
                SETPOINT_ID = ?,
                SETPOINT_VAL = ?)
```

Here, STA is a named element that may be used to describe any station. SETPOINT_ID and SETPOINT_VAL are now two named elements that may be used to describe a variety of set points. Assume that the setpoint_primed field is "5" regardless of the station or the type of set point.

After the command from the SCADA host is recognized an appropriate reply must be generated and the set point must be primed. Recall the portion of the INXREF command from the previous example that accomplished this:

```
REPLY(ULINT2     4141x,          /* station id
               0,                /* check sum
               20,               /* set point primed
               10,               /* set point id
               DP_SETPOINT)      /* requested set point value
ARM("AA.DP_PRIME", 1)            /* set discharge pressure
                                  /* set point timer to 1 min.
```

To make this portion of the INXREF command more general purpose, “4141x”, “10”, “DP_SETPOINT” and “AA.DP_PRIME” must be replaced with elements that will allow for other stations and set points. Using the same element names as in the message recognition part, an example would be:

```
REPLY(ULINT2     STA,            /* station id
               0,                /* check sum
               20,               /* set point primed
               SETPOINT_ID,      /* set point id
               SETPOINT_VAL)     /* requested set point value
ARM("[STXT(STA)].[SETPOINT_ID]_PRIME", 1)
```

The function STXT has been previously defined to convert the numeric form of a station to the text form:

```
DEFINE.FUNCTION STXT(STA) =
+     STA == 4141x ? "AA" :
+     STA == 4242x ? "BB" :
+     STA == 4343x ? "CC" : "??"
```

If the STXT function were not used, then the field [STA] would be replaced with the base 10 form of “4141x”, as text ([STA] would be replaced with “16705”). Note also that a timer is armed with a name that is created from STA and SETPOINT_ID. For the discharge pressure set point example, where SETPOINT_ID is “10”, and STA is “AA”, the name of the timer would be “AA.10_PRIME”. If the suction pressure SETPOINT_ID is “11”, then that timer would be named “AA.11_PRIME”.

The complete INXREF command to recognize a prime set point command and react to it would be:

```
COMMAND(
     ULINT2          STA=?,
               ?,
               5,
               SETPOINT_ID = ?,
               SETPOINT_VAL = ?) {
REPLY(
     ULINT2          STA,              /* station id
               0,                      /* check sum
               20,                     /* set point primed
               SETPOINT_ID,            /* set point id
               SETPOINT_VAL)           /* requested value
ARM("[STXT(STA)].[SETPOINT_ID]_PRIME", 1)}
```

Changing any set point

In the previous subsection, any set point at any station is primed. For the station “AA” example, an INXREF command that recognizes the change set point command and generates a reply was:

```
COMMAND(ULINT2 4141x, ?, 30, 10, DP_SETPOINT = ?) {
     IF ( ISARMED("AA.DP_PRIME") ) {
          REPLY(ULINT2     4141x,     /* station id
               0,                     /* check sum
               40,                    /* set point change confirmed
               10,                    /* set point id
               DP_SETPOINT     )      /* set point value
          ISSUE( POKE AA.DP:SP = [DP_SETPOINT] )
          }
     ELSE {
          REPLY(ULINT2     4141x,     /* station id
               0,                     /* check sum
               45,                    /* set point change rejected
               10,                    /* set point id
               DP_SETPOINT)           /* set point value
          }
}
```

This INXREF command can be made general purpose by replacing the station AA and discharge pressure set point specific elements in a manner similar to that done for the set point prime command. Because the change set point command may result in a change to a model variable, namely the appropriate control system set point, a way to issue the correct POKE based on the SETPOINT_ID (i.e., discharge pressure or suction pressure) must be included in the INXREF command.

An example of a general purpose command for changing either the discharge or the suction pressure set point at any station would be:

```
COMMAND(ULINT2     STA = ?,
               ?,
               30,
               SETPOINT_ID = ?,
               SETPOINT_VAL = ?) {
IF ( ISARMED("[STXT(STA)].[SETPOINT_ID]_PRIME") ) {
     REPLY(ULINT2          STA,          /* station id
               0,                        /* check sum
               40,                       /* set point change confirm
               SETPOINT_ID,              /* set point id
               SETPOINT_VAL)             /* set point value
     IF ( SETPOINT_ID = 10 )  {
          ISSUE(POKE [STXT(STA)].DP:SP = [SETPOINT_VAL])
          }
     ELSE IF ( SETPOINT_ID = 11 ) {
          ISSUE(POKE [STXT(STA)].SP:SP = [SETPOINT_VAL])
          }
     }
ELSE {
     REPLY(ULINT2          STA,          /* station id
               0,                        /* check sum
               45,                       /* set point change rejected
               SETPOINT_ID,              /* set point id
               SETPOINT_VAL)             /* set point value
     }
}
```

Trainer example - Start a unit

Similar to the set point change in the previous examples, a unit status change is also a 2-step operation: the status change is primed, then it is executed. This example will assume that after the status change has been primed, it will remain primed for 1 minute. The change status command must come while the status change is primed or it will be ignored.

Using the same station example as before, the command sent from the SCADA host to the RTU to prime the status change is of the form:

```
<stn_id>                16 bits
<chk_sum>               16 bits
<prime_stchange>        16 bits
<unit_id>               16 bits
```

To make the unit status change command general purpose, assume that the prime_stchange field is 6. The command to prime the status change is then recognized by the command:

```
COMMAND(ULINT2 STA = ?, ?, 6, UNIT_ID = ?)
```

After the command is recognized, an appropriate reply must be generated. The reply to a prime status change command is of the form:

```
<stn_id>                16 bits
<chk_sum>               16 bits
<stchange_primed>       16 bits
<unit_id>               16 bits
```

Assuming that the stchange_primed field is 21, then an INXREF command that will recognize the prime status change command and generate a reply would be:

```
COMMAND(ULINT2 STA = ?, ?, 6, UNIT_ID = ?) {
     REPLY(ULINT2          STA,                /* station id
               0,                              /* check sum
               21,                             /* status change primed
    ARM("[STXT(STA)].[UNIT_ID]_PRIME", 1)    /* arm timer for 1 min.
UNIT_ID)                        /* unit id
     }
```

Note that this command will set a timer, named AA_1_PRIME for station AA, unit #1.

Now that the status change has been primed, the command needed to actually start the unit will be defined. The command to start a unit is of the form:

```
<stn_id>                    16 bits
<chk_sum>                   16 bits
<execute_stchange>          16 bits
<unit_id>                   16 bits
```

Assuming that the stchange_primed field is 31, then an INXREF command that will recognize the change status command is:

```
COMMAND(ULINT2 STA = ?, ?, 31, UNIT_ID = ?)
```

After the change status command has been recognized, a reply must be generated. The reply will be either a confirmation that the status change has been made, or a reply that the change was not made. The status change will not be made if the operator waited too long to issue the change command, and consequently the arming timer ran out. The reply to the execute status change command is of the form:

```
<stn_id>                    16 bits
<chk_sum>                   16 bits
<stchange_confirm>          16 bits
<unit_id>                   16 bits
```

Assume that the stchange_confirm field is 41 if the status change has been executed. The following is the resulting INXREF statement:

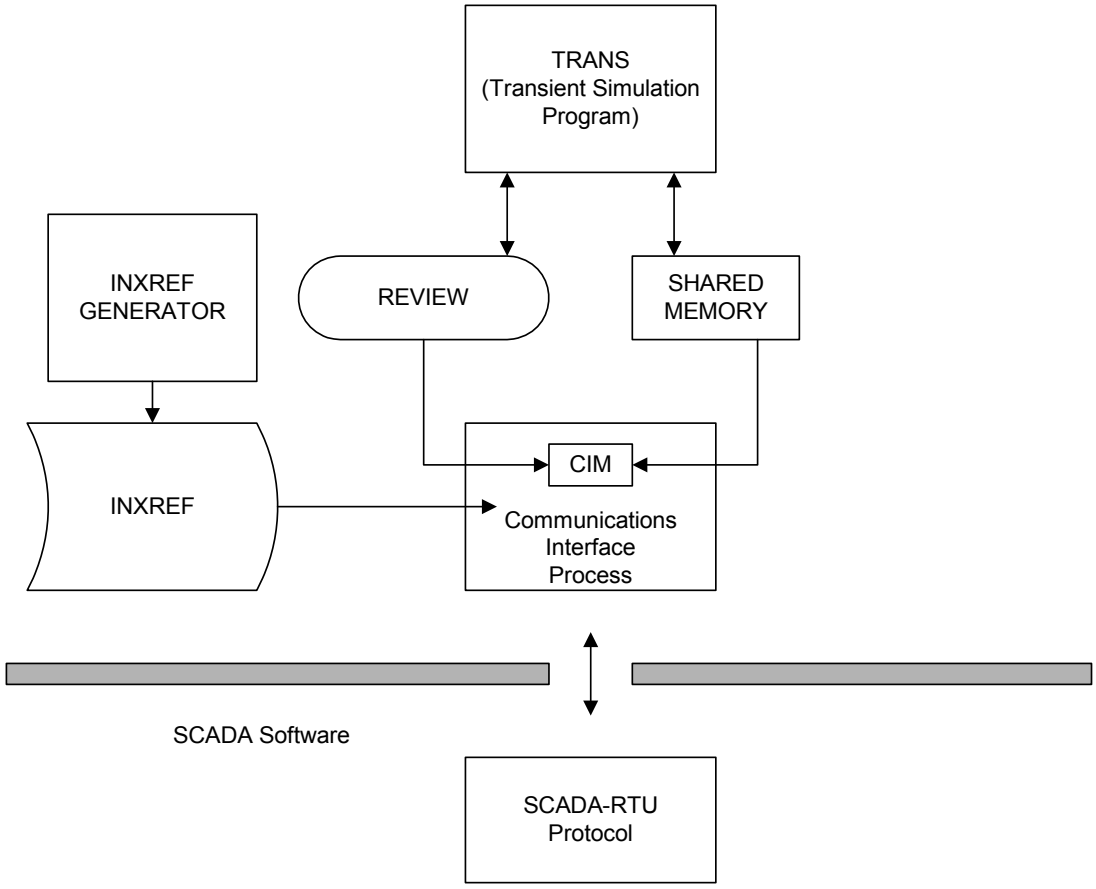
```
COMMAND(ULINT2 STA = ?, ?, 31, UNIT_ID = ?)  {
     IF ( ISARMED("[STXT(STA)].[UNIT_ID]_PRIME") ) {
          REPLY(ULINT2     STA,          /* station id
                0,                      /* check sum
                41,                     /* status change confirmed
                UNIT_ID)                /* unit id
     ISSUE(START_UNIT([STXT(STA)].U[UNIT_ID])
     }
ELSE {
     REPLY(ULINT2     STA,              /* station id
                0,                      /* check sum
                46,                     /* status change rejected
                UNIT_ID)                /* unit id
     }
}
```

Notice that in the previous example, the INXREF command recognizes the change status command then, based on the status of the arming timer, either starts the unit in the model and produces an affirmative reply, or simply produces a reply that states that the status of the unit has not been changed.

The TRANS command that actually starts the unit in the model submits a DEFINE.SEQUENCE. To use this command, there must already exist (in the INTRAN file) a DEFINE.SEQUENCE with the name START_UNIT. Generally, a command to start or stop a unit actually results in a sequence of events being initiated. This sequence may check on the status of the unit and its associated suction and discharge valves, then open the valves in sequence or aborts.

Trainer example - Automated INXREF file generation

In general, the INXREF file for a specific CIM interface to a SCADA system can be quite lengthy. The longer the INXREF file, the more time-consuming it is to maintain manually. Depending on the SCADA system, and the presence of naming conventions for points in the SCADA database, the automatic generation of an INXREF file is feasible. This subsection describes a method to automate INXREF file generation, if the SCADA software allows easy manipulation of internal SCADA values and relationships. The following flow diagram shows the relationship of the INXREF file generator to the Trainer system.



Pipeline Trainer Software

Philosophy

The INXREF file is a set of relations between a SCADA database and a Trainer model. Creation of the INXREF file will depend on how the Trainer model and the SCADA database view data.

An abstract SCADA database contains a name, a value, and a method to update this value. The name for the value could be as descriptive as AA_Unit1_Status, AA,U1STS, or as cryptic as 040503, (with an implied mapping of: 04 to the AA Remote Telemetry Unit (RTU), 05 to unit #1, and 03 for the status). The value would be the important data related to the name, (e.g. a status). An update method would be a mapping of the database name to telemetry values, and a request of that value from the telemetry system. (The string AA_Unit1_Status could be tied to the command to poll RTU 04 for the point 03 (status) of the unit 05, or the 040503 could be sent directly to the RTU polling routines if the cryptic name were used.)

The Trainer model contains a descriptive name for each element and the associated values, chosen by the Trainer model builder.

The relation between the SCADA database and the Trainer model values associates a name in the SCADA database with a name in the Trainer model. This relationship can be made in several locations. The INXREF file could be built to associate Trainer element names with RTU points or to directly relate a SCADA database name to a Trainer model value. In the case where the SCADA database has the ability to produce tables of the database names and a method

with which it retrieves that value, then an automated INXREF file generator is feasible. GL does not supply an INXREF file generator.

Input files required

The relations for the INXREF file are in two directions, inputs (set points to be changed, valves to open and close, etc.) and outputs (return temperatures and pressures, etc.). An INXREF file generator will require two lists, commands (SCADA database to Trainer model) and polls (Trainer model to SCADA database). The most trivial of these files would list a SCADA database name, update ID, and a model name. A pair of sample input files that use variables from the example follow:

Poll data table:

```
/* SCADA Database Name          Update ID          Model Name
AA_Unit1_Status                   01x               AA.U1:ST
AA_Unit1_Inlet_P                  02x               AA.U1:P-
```

Command data table:

```
AA_Suction_Setpoint               03x               AA.SP:SP
AA_Unit1_Status                   04x               AA.U1:ST
```

These files could be used to automatically generate the following INXREF file.

```
COMMAND (UBYTE 01x) { REPLY (ULINT2 AA.U1:ST) }
COMMAND (UBYTE 02x) { REPLY (ULINT2 AA.U1:P-) }
COMMAND (UBYTE 03x; ULINT2 VALUE = ?)
     {     ISSUE (POKE AA.SP:SP=[VALUE]) }
COMMAND (UBYTE 04x; ULINT2 VALUE = ?)
     {     ISSUE (POKE AA.U1:ST=[VALUE])}
```

The previous example has some limits. All read values read are returned as unsigned long integers (ULINT2), and all set values are set by unsigned long integers (ULINT2). The update IDs must be unique, even if both points are the same for input and output, (AA_Unit1_Status and AA.U1:ST). Part of the relation is lost in that the specific RTU is not mentioned, so the INXREF file processing has no concept of the RTU grouping of data points.

In an ideal example, the Trainer model should mimic an RTU in the INXREF file, pass different storage types for values, read different storage types for different values, and use a value for the update ID that is useful to the SCADA software. Because all of this data exists in the SCADA database, it is better to extract that information from the SCADA database rather than maintaining a second copy of that information. Therefore the initial input files should appear similar to the following:

Poll data table:

```
/*SCADA Database Name          Model Name
AA_Unit1_Status                AA.U1:ST
AA_Unit1_Inlet_P               AA.U1:P-
```

Command data table:

```
AA_Suction_Setpoint            AA.SP:SP
AA_Unit1_Status                AA.U1:ST
```

These tables could be processed by the SCADA vendor to add other useful information, such as an RTU number, a SCADA database index, and the type of this variable.

Polling list:

```
/*SCADA Database Name     Index     RTU     Type          Model Name
AA_Unit1_Status             1        4      UBYTE           AA.U1:ST
AA_Unit1_Inlet_P            2        4      ULINT2          AA.U1:P-
```

Command list:

```
AA_Suction_Setpoint          3       4      ULINT2          AA.SP:SP
AA_Unit1_Status              1       4      UBYTE           AA.U1:ST
```

The INXREF generating program operates on these files and generates the following output:

```
INCLUDE LOCALDEFINES.INC
COMMAND (POLL; UBYTE 01)
     {
     IF (RTU_4_STATUS == OK)
          { REPLY (ACK; UBYTE AA.U1:ST) }
     ELSE
          { REPLY (NACK) }
     }
COMMAND (POLL; UBYTE 02)
     {
     IF (RTU_4_STATUS == OK)
          { REPLY (ACK; ULINT2 AA.U1:P-) }
     ELSE
          { REPLY (NACK) }
     }
COMMAND (CONTROL; UBYTE 03; ULINT2 VALUE = ?)
     {
     IF (RTU_4_STATUS == OK)
     {
          REPLY (ACK; ULINT2 VALUE)
          ISSUE (POKE AA.SP:SP=[VALUE])
     }
     ELSE
          { REPLY (NACK) }
     }
COMMAND (CONTROL; UBYTE 01; UBYTE VALUE = ?)
     {
     IF (RTU_4_STATUS == OK)
          {
          REPLY (ACK; UBYTE VALUE)
          ISSUE (POKE AA.U1:ST=[VALUE])
          }
     ELSE
          { REPLY (NACK) }
     }
```

The include file, local DEFINES.INC, allows site configuration of various constants. The sample MACROs that must be defined for the example to work are:

```
MACRO(OK, 1)                /* the Value for a functioning
                            /* RTU_#_STATUS variable
MACRO(CONTROL, UBYTE 05)    /* the constant used to represent
                            /* a control request from the
                            /* SCADA system
MACRO(POLL, UBYTE 06)       /* the constant used to represent a
                            /* poll request from the
                            /* SCADA system
MACRO(ACK, UBYTE 1)         /* the acknowledgement constant
                            /* returned to the SCADA system
                            /* when a request is accepted
MACRO(NACK, UBYTE 0)        /* the non acknowledgement constant
                            /* returned when a SCADA system
                            /* request is rejected
```

The RTU_#_STATUS variables should be declared in the Trainer INTRAN file and initially set to whatever the OK constant is MACROed to expand to. This will allow the evaluator of the Trainer software to simulate an RTU outage by poking the variable RTU_#_STATUS to a value other than the OK constant. When the RTU_#_STATUS variable is no longer set to OK, values will not be returned to SCADA interface software requests for that RTU, so emulating an RTU failure.

Hexidecimal/decimal conversions

Commands and Replies often contain data that is defined in hexadecimal format. While numbers may be entered in decimal format, the use of hexadecimal will offer the ability to reduce the length of numbers as well as remain consistent with the format of messages exchanged between SCADA and the real pipeline.

This appendix will define the conversion procedures between decimal and hexadecimal, as well as provide a table displaying the equivalents of various ranges of decimal/hexadecimal numbers.

Column definitions:

HEX placeholders: 65536 4096 256 16 0

DEC placeholders: 10000 1000 100 10 0

Rules for conversion to hexadecimal

1 Numbering takes place right to left as the decimal equivalent is increased. That is, the hexadecimal columns increase beginning with the zero's column, followed by the 16's column, etc.

2 The hexadecimal numbers in each column range from 0x to 0fx. The decimal equivalent to these values are defined in the following table:

Decimal	Hexadecimal
0	0x
1	1x
2	2x
3	3x
4	4x
5	5x
6	6x
7	7x
8	8x
9	9x
10	10ax
11	0bx
12	0cx
13	0dx
14	0ex
15	0fx

The numbers included under the hexadecimal column could have been written without the x appended to the end. These are included only because hexadecimal numbers in the INXREF file require the x on the end.

3 The conversion of hexadecimal to decimal is demonstrated by the following equation. Note that this equation assumes no numbers larger than decimal 65535 will be converted.

DEC = (4096*d)+(256*c)+(16*b)+a

where:

a = equivalent number in 0's column

b = equivalent number in 16's column

c = equivalent number in 256's column

d = equivalent number in 4096's column

Example of hexidecimal/decimal conversions

Convert 23ex to decimal.

```
DEC = (4096 * 0) + (256 * 2) + (16 * 3) + 14
DEC = 0 + 512 + 48 + 14
DEC = 574
```

The conversion from decimal to hexadecimal is not obvious. The following guidelines illustrate the appropriate steps.

Notice the magnitude of the decimal number and follow the appropriate step that defines the procedure for that number in the input example below. (This conversion process assumes no numbers larger than decimal 65535 will be converted.) *DEC* refers to the value of the decimal number.

Input example 1

```
IF (0 <= DEC <= 15)
```

Then the hexadecimal equivalent will be 0x - 0fx. See "Rules for conversion to hexadecimal" on page 922.

Convert 12 to hexadecimal.

```
HEX = 0cx
  IF (16 <= DEC <= 255)
```

Then perform the following calculation: DEC/16. The resulting integer will be placed in the 16's column. The remaining difference will be placed in the 0's column.)

Convert 123 to hexadecimal

```
123/16 = 7
(7 * 16) = 112
```

The remaining difference is 123 - 112 = 11 (= bx). Therefore, HEX = 7bx

Input example 2

```
IF (256 <= DEC <= 4095)
```

Then perform the following calculation: DEC/256. The resulting integer will be placed in the 256's column

Perform the following calculation with the remaining difference:

- REMAINDER/16
- The resulting integer will be placed in the 16's column.
- The remaining difference will be placed in the 0's column.)

Convert 574 to hexadecimal

```
574/256 = 2
(2 * 256) = 512
```

The remaining difference is 574 - 512 = 62

```
2. 62/16 = 3
(3 * 16) = 48
```

The remaining difference is 62 - 48 = 14 (= ex). Therefore, HEX = 23ex

Sample hexidecimal/decimal table

Decimal	Hexadecimal
0	0x
1	1x
9	9x
10	0ax
15	0fx
16	10x
25	19x
26	1ax
32	20x
48	30x
64	40x
80	50x
96	60x
112	70x
128	80x
144	90x
160	a0x
176	b0x
192	c0x
208	d0x
224	e0x
240	f0x
256	100x
257	101x
265	109x
266	10ax
271	10fx
272	110x
288	120x
4096	1000x
65536	10000x

CHAPTER

19

Interfacing SPS to External Applications

This section provides detailed descriptions of the various application program interfaces (APIs) that are supported by the SPS family of products as well as information on the various servers used to communicate among SPS-related software products and other external applications. This section contains the following:

Conventions for this programming documentation

- Program names and file names are capitalized in this document, but are not capitalized in practice.
- Environment variables are capitalized in this document and are also capitalized in practice. Environment variables are preceded by a dollar sign, $, in this document.
- Function names are case correct and are shown with trailing parenthesis, e.g., rtuReadNextPt().
- Function prototypes (calling sequences) are shown using C coding style.
- Function call arguments are documented using arrows to indicate the direction of data flow. An arrow that points away from an argument means that the argument is input to the subroutine. An arrow that points towards an argument means that the argument is an output of the subroutine. A double arrow that points both away from and towards an argument means that the argument is both an input and an output of the subroutine.
- Within the Usage Considerations paragraph of the Function Prototypes section, function call arguments are case correct and are shown in double quotes.

Common uses for interfacing with SPS

This section describes some of the common uses of the interfaces and how to build the interfaces. The interfaces may be used for the following purposes:

Interfacing a SCADA system to a Trainer

Trainer simulates the pipeline and the SCADA control system(s). For simulation purposes, Trainer acts as the pipeline and the Remote Terminal Units (RTU) that are physically in the field. The trainee using the Trainer issues commands through the standard SCADA interface. To interface a SCADA system to a Trainer, you may use one of the following interfaces:

Use of Time

The TRANS program normally attempts to run in real time. However, there are features of the system which permit it to run either slower or faster than real time. These features may be used during a training session or during a review by the instructor.

If these features are used during a training session, it is best if the SCADA system has some way of using the simulation time that is available from TRANS rather than its own system clock time. The trainee can then observe the advancing training simulation time through the SCADA displays and the time-dependent activities in the SCADA system can be triggered from the simulation time (e.g. periodic trending of data to a disk file).

The TIME variable contains the simulation time in minutes since 00:00:00 December 31, 1967.

Example:

Use the one byte command 4ex to return integer seconds since 00:00:00 January 1, 1970:

```
COMMAND (UBYTE 4ex) {
REPLY (ULINT4 (TIME - 60 * 24 (1 + 366 + 365) ) * 60 )
}
```

Sending simulation results from the Statefinder/ Leakfinder model to the SCADA system

You may use one of the following interfaces to send simulation results from the Statefinder/Leakfinder model to the SCADA system:

- "TRANSEXPORT interface" on page 991
- "StOPC" on page 980
- "WinCip" on page 1001
- "dr* interface" on page 934

Providing SCADA information to a Statefinder/ Leakfinder model

The SCADA system will create and then periodically update a file of SCADA data, called the RTU data file. The Statefinder/Leakfinder model continually reads the RTU data file and produces a model state that is a close approximation of the actual state of the pipeline system. This process is described in the "Statefinder and Leakfinder" on page 741. The RTU data file is a binary file and is typically a few hundred megabytes in size. The file is organized in a circular fashion, with typically a few days or weeks of historical data in addition to current data.

Statefinder

Statefinder acts as an extension to the SCADA system by constantly providing available distance plots of pressure, flow, density and other parameters along the pipeline. These variables can be passed back to the SCADA system just as if they came from installed field equipment. This enables the operators to observe system performance even where there is no installed telemetry.

Periodically, the SCADA system scans the remote telemetry equipment and receives updated data from the field measurements and status indicators. Not all SCADA points are sampled at exactly the same time. For Statefinder, both the SCADA point value and the time that the value represents are important. At each SCADA scan, data (name, value, time, and quality) for each scanned SCADA point is passed to the model environment even if the point's value is exactly the same as that from the previous scan. Assigning a time to a SCADA point at the RTU in the field is the most accurate means for time-tagging a SCADA point. Typical installations assign time-tags to the SCADA points as they are received by the SCADA host computer or as the data are passed to the modeling environment.

Statefinder automatically adjusts and calibrates incoming SCADA data to compensate for errors associated with meter drift or changing pipeline roughness. Alarms can be configured to alert problems such as an instrument reaching its maximum calibration limit. This assists the user in determining maintenance schedules.

Leakfinder

Leakfinder is a tool for pipeline operations surveillance and leak detection.

Leakfinder combines Statefinder with the Leakanalyzer to form an effective leak detection system. Statefinder uses real time SCADA data to track pipeline hydraulics and to signal atypical measurements. Leakanalyzer examines those measurements and looks for possible leak signatures. When a leak is detected, Leakfinder raises an alarm that shows

the leak location, the onset time, the leak rate, and the total volume released. Leakfinder also assigns a confidence level to the presence and location of the leak.

Statefinder can operate with degraded or limited SCADA measurement data. When this occurs, as when a liquid pipeline runs slack, Leakanalyzer automatically adjusts its leak detection thresholds (LDTs). This is adjusted in order to maintain the best possible system performance while also avoiding false alarms. An LDT is a distance plot along the pipeline that represents the smallest leak that is not masked by many factors. These factors include the aggregate uncertainties in the measurement data, current operating conditions, fluid properties, and elevation data. These LDTs are profiled graphically. They also provide a dynamic and constantly available measure of how well the pipeline is protected.

Interfaces

The following interfaces may be used to provide SCADA information to Statefinder/Leakfinder/Predictor:

- "todrem interface" on page 980
- "OpcToSps" on page 958

Reading RTU data files

You may use the rtu* interface to read RTU data files. For more information, see "rtu* interface" on page 958.

Sending Predictor data to SCADA

You may use the following interfaces to send Predictor data to SCADA:

- "TRANSEXPORT interface" on page 991
- "dr* interface" on page 934

Checking the status of the real time model

Use the TRANSHEARTBEAT program to determine whether the real time model is still running, and if running, to confirm that it is not very far behind real time. For more information, see "TRANSHEARTBEAT" on page 987.

Interfacing to another GUI

Sometimes users prefer to use a GUI other than the SPS GUI. GL provides interfaces that allow information from TRANS to be sent to the GUI, and commands issued through the GUI to be sent to TRANS. You may accomplish through the following interfaces:

- "dr* interface" on page 934
- "StOPC" on page 980

Interfacing to another GUI using low-level API calls

GL provides low level function prototypes which the interface programmer can use to communicate with a TRANS run. This interface displays TRANS data in the local GUI and sends commands issued through the GUI to TRANS. This interface retrieves information from shared memory and from the review file. For more information, see "Shared memory" on page 57 and "REVIEW file" on page 1058.

The types of interaction include:

- Connecting to a running TRANS and initializing the interface using cdrini()
- Retrieving data from a running TRANS using cdrgta(), cdrgtn(), cdrgts(), drgetv(), and drgtvs()
- Retrieving plots and data related to plots using drGetDistPlot(), drGetTimePlot(), and drGetDistPlotTimes()
- Sending commands to a running model and change simulation variables using cdrptc() and drputv()

For more information, see "dr* interface" on page 934.

Interfacing to Microsoft applications

You may interface SPS to Microsoft applications using one of the following COM servers:

- "StOPC" on page 980

Interfaces

This section describes the available interfaces to the SPS family of products. The interface you choose to use likely depends on the platforms you are interfacing to or from and the purpose of your interface. For more information on choosing an interface by task, see "Common uses for interfacing with SPS" on page 926. The following interfaces to SPS are available:

Interface	Description	Interface technology
"cim interface" on page 930	Used to interface a SCADA system to a Trainer. Process RTU queries. Available on all supported platforms.	C-language API
"dr* interface" on page 934	Used to interface a Statefinder/Leakfinder model to a SCADA system or interface SPS to another GUI. Low-level access to SPS devices and attributes. Available on all supported platforms.	C-language API
"OpcToSps" on page 958	Used to interface a SCADA system to a Statefinder/ Leakfinder model. Available for supported versions of Windows only. Contact GL for more information.	OPC-Data-Access Client
"StOPC" on page 980	Used to interface a SCADA system to a Trainer, interface Statefinder, Leakfinder, or Predictor to a SCADA system, interface SPS to another GUI or to a Microsoft application. Includes a user-interface for configuring and monitoring the server. Available for supported versions of Windows only. Contact GL for more information.	OPC Data-Access server (industry-standard COM interface).

Interface	Description	Interface technology
"rtu* interface" on page 958	Used to read RTU data files. Available on all supported platforms	C-language API
"todrem interface" on page 980	Used to get SCADA data into a real-time model.	C-language API
"TRANSHEARTBEAT" on page 987	Used to determine whether the real time model is still running, and if running, to confirm that it is not very far behind real time.	C-language API
"TRANSEXPORT interface" on page 991	Used to push data from a Statefinder/Leakfinder/Predictor model to a SCADA system. Available on all supported platforms.	C-language API
"WinCip" on page 1001	Used to interface Trainer/Statefinder/Leakfinder to SCADA. Makes SPS appears to be one or more RTU devices. For monitoring the connection, WinCip includes a user interface. Available for supported versions of Windows only. For more information on using WinCip, see "WinCip" on page 1001.	TCP/IP server

Several utility routines are also available that do not communicate with SPS directly but may help you as you work with some of the other interfaces. See "Utility routines" on page 996.

cim interface

The cim interface is the communications interface module routine. This interface is used to interface a SCADA system to a Trainer and is available on all supported platforms.

WinCip is built around cim and provides a TCP/IP interface to it. For more information, see "WinCip" on page 1001.

cim interface initialization

To initialize the interface, the cim() on page 933 routine is called and automatically reads the configuration file, INXREF, into its internal memory and connects to the TRANS process. The case name of the INXREF file must be provided to the `cim()` routine via the environment variable $DREMCASENAME. The name of the file that `cim()` opens and reads is $DREMCASENAME.inxref.

The $DREMCASENAME environment variable is supported so that the user can run and/or maintain multiple cases. GL recommends that the user be allowed to specify the case name (either via command line or prompt) while starting CIP, the communications interface process.

CIP is a main program that passes messages to and from the SCADA software with little or no interpretation. CIP may be written by GL, a SCADA vendor, or the pipeline operating company.

The `cim()` routine opens and then reads from the INXREF file. The `cim()` routine also opens and the writes to the OUTXRF file, an echo of the INXREF file plus any diagnostics. The name of the OUTXRF file is $DREMCASENAME.outxrf. The `cim()` routine expects that the CIP is running in the same directory as the TRANS program so that the `cim()` routine can find the necessary files.

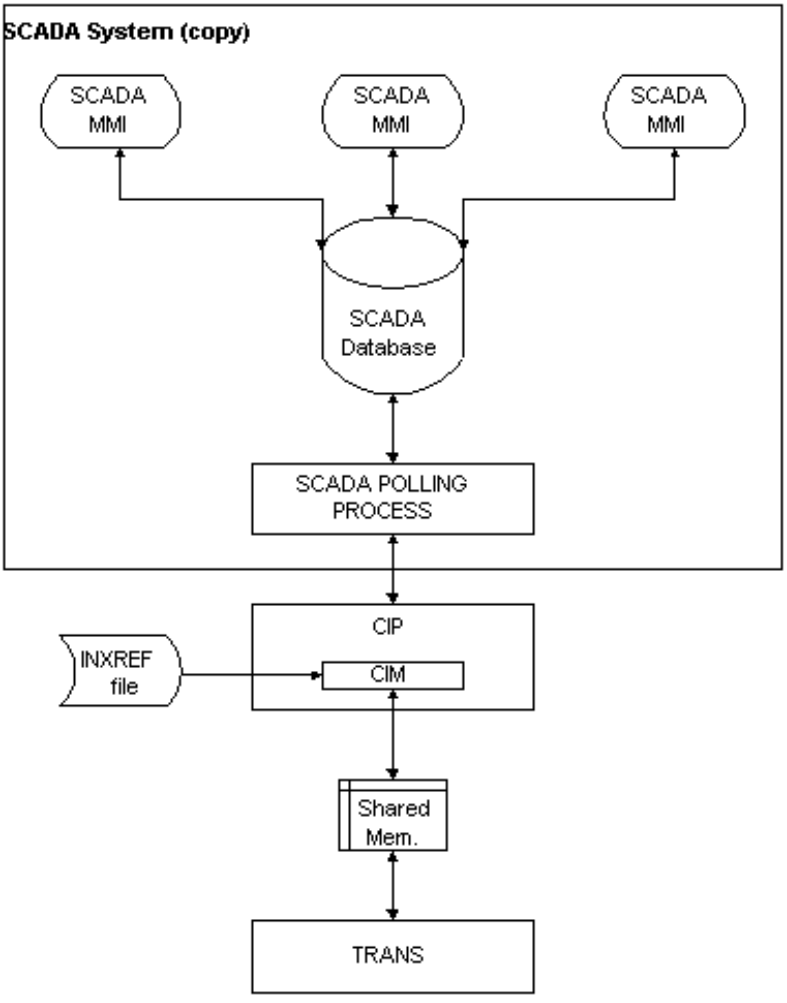
cim data acquisition

The TRANS program updates the shared memory array after each simulation time step. For more information, see "Shared memory" on page 57. As part of the command processing, the `cim()` routine checks to see if TRANS has completed a time step. If so, the `cim()` routine copies the shared array into local storage. This copy is used for all subsequent processing of the <rtu-command>. Therefore, several calls to `cim()` will operate on the same simulated data. The simulated data does not change until TRANS completes a step. (A step is usually five to ten seconds.)

cim program flow diagram

The `cim()` routine accepts RTU commands from the SCADA system and matches those commands to the INXREF file. Then it processes the commands using Trainer commands.

The basic flow of information between the SCADA system and Trainer is shown in the figure below.



Use of Checksums

This interface is designed assuming that the required checksum handling is done by the CIP, outside of the `cim()` routine. Because Trainer is normally on the same computer as the SCADA software or is connected using a reliable communication link, it is usually not necessary to validate the checksum field of the commands that are incoming to CIP.

The checksum field in the replies may either be inserted into the `cim()` routine reply by the CIP or simply ignored by the SCADA software.

Restarting

If the initial connection attempt to TRANS fails, the CIP program must be restarted.

Time Synchronization

There is no time synchronization between `cim()` and TRANS.

SPSCIMECHO

If the environment variable $SPSCIMECHO is set (to anything), this routine will print out to the standard output the integer values of every byte in `cmdin` and `reply`, in both decimal and hexadecimal formats.

Functions

See the following function for more information.

Function	Description
cim()	Exports data

cim

This routine is the communication interface module (CIM). See "Trainer" on page 877 for more information on implementing a Trainer interface.

Declaration

```
void cim(const char *cmdin, const int *clen, char *reply, int *rlen, int
*ier)
```

Arguments

`*cmdin`	>>	SCADA command to the RTU
`*clen`	>>	length in bytes of the RTU command
`*reply`	<<	reply from the RTU to the SCADA system, as returned by `cim()`
`*rlen`	<<	length in bytes of the reply, as returned by `cim()`
`*ier`	<<	error value, as returned by `cim()`. If an error occurs, a non-zero value is returned.

Usage Considerations

This routine retrieves the model case name *case* from the environment variable DREMCASENAME. The first time cim() is called, it parses the *case*.inxref file into internal memory and connects to the *case* TRANS process. `cim()` will not parse *case*.inxref again unless CIP is restarted.

This routine executes the following steps:

1. Check if TRANS has taken a step. If TRANS has taken a step, lock the shared memory array, copy the memory to local storage, and unlock the shared memory.
2. Search for `cmdin` in the data structure containing the INXREF directives.
3. Interpret the if-logic in the data structure using the current value of the simulation variables.
4. Generate the `reply` message based on the current value of the simulation variables. Format the reply as specified by the INXREF file.
5. Send commands to TRANS in order, as specified in the INXREF file.

This routine has no knowledge of a lost connection to TRANS once it connects to TRANS. This routine will hold onto shared memory after TRANS has exited.

Possible return values for `*ier` are as follows:

Value	Meaning
0	command found
1	command not found
2	cannot parse INXREF file
3	cannot connect to TRANS; restart CIP

See Also

cdrini() on page 939

"Interfacing a SCADA system to a Trainer" on page 926

Library: spsclient stclib

Licensing: DRINIT CIM

dr* interface

The `dr*` interface includes most `dr*` and `cdr*` API routines. This collection of routines is useful for interfacing a Statefinder/Leakfinder model to a SCADA system or for interfacing SPS to another GUI. These routines are available on all supported platforms and are considered low-level API calls, which may be called directly or through a wrapper class.

Function	Description
cdrini()	Connects to a simulation model
drgetm()	Gets the current model's connection ID
drsetm()	Sets the current model's connection ID
drdetm()	Detaches from the current model
drupd()	Copies shared memory into the local process memory
cdrgta()	Retrieves the address for a given expression string
cdrgtn()	Retrieves the peek name of a given address
cdrgts()	Retrieves the current string value
drgetv()	Retrieves the current scalar value
drgtvs()	Retrieves an array of values from a review file
cdrv2s()	Converts a numerical value to a status string
drfrea()	Frees resources associated with a `cdrgta()` call
cdrptc()	Sends a command string to a simulation
drputv()	Changes a simulation variable
DrGetStringList()	Gets a list of strings at given address
drGetDistPlot()	Gets distance plot data
drGetTimePlot()	Gets trend data
drGetDistPlotTimes()	Gets all available time stamps for which historical distance plot data may be retrieved
drGetDistPlotHist()	Gets historical distance plot data
drFree()	Frees resources associated with plot and trend data

cdrgta

Creates a “getval” address for an expression stored in character form, for use by other `dr-` routines as a handle when retrieving or modifying the value of the expression. The expression may be as simple as a peek name or as complex as any valid INTRAN expression. The “getval” address of the expression is valid while TRANS is running.

Declaration

```
void cdrgta(const char *str, int *addr, int *ier)
```

Arguments

`*str`	>>	string for which to create the “getval” address
`*addr`	<<	“getval” address, as returned by `cdrgta()`
`*ier`	<<	error value, as returned by `cdrgta()`. If an error occurs, a non-zero value is returned.

Usage Considerations

A “getval” address is an encoded integer containing various flags and offsets into data structures. A “getval” address is not a pointer in the usual C sense.

A “getval” address uses both local and shared memory. Once the application is finished with the “getval” address, you should use `drfrea()` to free the memory. If this is not done, repeated calls to `cdrgta()` will eventually fill up available memory.

See Also

cdrini() on page 939

drfrea() on page 943

“Interfacing to another GUI” on page 928

Library: spsclient stclib

Licensing: DRINIT

cdrgtn

"De-compiles" the expression associated with the "getval" address `*addr`.

Declaration

```
void cdrgtn(const int *addr, char *str, int *ier, int *size)
```

Arguments

`*addr`	>>	"getval" address, as returned by `cdrgta()`
`*str`	<<	resulting string, as returned by `cdrgtn()`
`*ier`	<<	error value, as returned by `cdrgtn()`. If an error occurs, a non-zero value is returned.
`*size`	<< >>	size for the allocation of `str`. `cdrgtn()` returns the actual string size.

Usage Considerations

The resulting string will not necessarily be identical (in a string sense) to the original string passed to `cdrgta()` because spacing and/or parenthesis nesting may be different.

You must allocate enough space for `str` to hold the resulting string. At the most, `size` number of bytes of the expression will be copied into `str`. Memory corruption can take place if `size` is not the allocated size of `str`.

This routine is useful for generating error messages.

See Also

cdrini() on page 939

cdrgta() on page 936

"Interfacing to another GUI" on page 928

Library: spsclient stclib

Licensing: DRINIT

cdrgts

Retrieves the current value of a string-valued expression associated with the “getval” address `*addr`.

Declaration

```
void cdrgts(const int *addr, char *str, int *ier, int *size)
```

Arguments

`*addr`	>>	“getval” address, as returned by `cdrgta()`
`*str`	<<	resulting string, as returned by `cdrgts`
`*ier`	<<	error value, as returned by `cdrgts()`. If no error occurs, zero is returned.
`*size`	<<	size for the allocation of `str`. `cdrgts()` returns the actual string size.

Usage Considerations

You must allocate enough space for `str` to hold the resulting string. At the most, `size` number of bytes of the expression will be copied into `str`. Memory corruption can take place if `size` is not the allocated size of `str`.

`*ier` returned from this routine is as follows:

- Zero if the retrieval was successful
- `DR_NOT_AVAILABLE` if the requested data is not known to the client (see the file saiInterface.h)
- Other values if the value of the expression did not evaluate to a string

If `*ier` is returned as `DR_NOT_AVAILABLE`, `str` is returned as “Not Available”. In this case, a later call to this function may return the actual value. This routine returns a date/time value as a string representation in the form YY/MM/DD HH:MM:SS. Use `drgetv()` to retrieve scalar values.

See Also

cdrini() on page 939]

cdrgta() on page 936

drgetv() on page 954

“Interfacing to another GUI” on page 928

saiInterface.h

Library: spsclient stclib

Licensing: DRINIT

cdrini

Initializes the `dr-` and `cdr-` API routines, otherwise known as `dr-` routines, for TRANS case name `case`. This routine reads the dictionary in the *case*.review file and performs other miscellaneous initialization.

Declaration

```
void cdrini(const char *case, int *ier)
```

Arguments

`*case`	>>	TRANS case name
`*ier`	<<	error value, as returned by `cdrini()`. For information on possible return values, refer to saiInterface.h.

Usage Considerations

For flexibility, consider allowing the end user to specify `case` via an environment variable and/or command line option.

There are two versions of the dr- routines, the drvydef version and the drshare version. The drvydef version of these routines runs on the same CPU that is running TRANS and uses shared memory to communicate with that TRANS. The drshare version uses RPC's to put a client/server "wrapper" around the dr- routines to communicate to TRANS through TRANSSHARE server process over the network.

A successful call to cdrini establishes a trans session. Multiple TRANS sessions may be managed by calling `drgetm()`, `drsetm()`, and `drdetm()` routines.

The drshare version of this routine establishes a new connection by reading the RPC network information in the case.share file created by TRANSSHARE server process.

See Also

drgetm() on page 950

drsetm() on page 957

drdetm() on page 942

saiInterface.h

"Interfacing to another GUI" on page 928

Library: (drshareclient or drvydefclient) spsclient stclib

Licensing: DRINIT

cdrptc

Sends a command string to TRANS.

Declaration

```
void cdrptc(const char *str, int *ier)
```

Arguments

`*str`	>>	command to send to TRANS
`*ier`	<<	error value, as returned by `cdrptc()`. If an error occurs, a non-zero value is returned.

Usage Considerations

The routine returns a non-zero `*ier` if it is unable to send the command. The command string `str` should contain a valid interactive TRANS command. This routine has no knowledge of the eventual success or failure of the command.

See Also

cdrini() on page 939

drputv() on page 956

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

cdrv2s

Converts a numeric value to the corresponding status string.

Declaration

```
void cdrv2s(const double *val, char *str, int *len, int *ier)
```

Arguments

`*val`	>>	value to convert
`*str`	<<	resulting string, as returned by `cdrv2s()`
`*len`	<< >>	size for the allocation of `str`. `cdrv2s()` returns the actual string size.
`*ier`	<<	error value, as returned by `cdrv2s()`. If an error occurs, a non-zero value is returned.

Usage Considerations

You must allocate enough space for `str` to hold the resulting string. At the most, `*len` number of bytes of the expression will be copied into `str`. Memory corruption can take place if `*len` is not the allocated size of `str`.

When `drgtvs()` returns a value with `flag` set to `DR_STRING` flag, use `cdrv2s()` to get the corresponding status string.

This routine returns a non-zero `*ier` if the conversion fails.

See Also

cdrini() on page 939

drgtvs() on page 955

"Interfacing to another GUI" on page 928

Library: psiml tsclient (spsclient on Windows)

Licensing: DRINIT

drdetm

Detaches and shuts down the connection to the current model.

Declaration

```
void drdetm(int *ier)
```

Arguments

*ier	<<	error value, as returned by drdetm(). If an error occurs, a non-zero value is returned.

Usage Considerations

The tport version of this routine is a no-op.

The tsclient (spsclient on Windows) version of the dr- routines supports multiple connections. This version of drdetm() detaches from the current model and frees associated RPC resources.

To avoid filling all available memory, you should free all of the "getval" addresses acquired by cdrgta() calls during the current session with matching drfrea() calls.

See Also

cdrini() on page 939

drsetm() on page 957

drgetm() on page 950

cdrgta() on page 936

drfrea() on page 943

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drfrea

Frees resources associated with a “getval” address returned by `cdrgta()`.

Declaration

```
void drfrea(const int *addr, int *ier)
```

Arguments

`*addr`	>>	“getval” address, as returned by `cdrgta()`
`*ier`	<<	error value, as returned by `drfrea()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Do not use `drfrea()` with a previously freed “getval” address, or anything other than a “getval” address that may have been returned by `cdrgta()`. Doing so will likely result in an access violation error.

Always use `drfrea()` to free a “getval” address before generating another with `cdrgta()`. Failure to do so could eventually cause all available memory to be filled.

See Also

cdrini() on page 939

cdrgta() on page 936

“Interfacing to another GUI” on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drFree

Frees resources associated with `drPointXY` data allocated in routine `drGetDistPlot()`, `drGetTimePlot()`, and `drGetDistPlotHist()`, or the double array allocated in `drGetDistPlotTimes()`.

Note: Failure to use drFree before generating new resources could eventually cause all available memory to be filled. Also, use of drFree to free any other type of resource will likely cause an access violation error.

Declaration

```
void drFree(void *ptr, int *status)
```

Arguments

`*ptr`	>>	pointer to the memory block to be freed
`*status`	<<	error value, as returned by `drFree()`. If no error occurs, zero is returned.

Usage Considerations

Possible return values for `*ier` are as follows:

Value	Meaning
0	operation succeeded
1	operation failed

See Also

cdrini() on page 939

drGetDistPlot()

drGetTimePlot() on page 952

drGetDistPlotHist() on page 947

drGetDistPlotTimes() on page 949

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drGetDistPlot

Returns the distance plot data of a simulation value for a specified item along a given path.

Declaration

```
void drGetDistPlot(char const* const* path, const char *item, int *points,
drPointXY **curvedata, int *ier)
```

Arguments

`path`	>>	path defined using a list of device names
`*item`	>>	name of item to get data for, e.g., PRESSURE
`*points`	<<	number of data points, as returned by `drGetDistPlot`
`**curvedata`	<<	pointer to an array of drPointXY data points, as returned by `drGetDistPlot`
`*ier`	<<	error value, as returned by `drGetDistPlot()`. If no error occurs, zero is returned.

Usage Considerations

After using or copying the retrieved data, the caller must call `drFree()` to free the memory allocated for the `*curvedata` array. Failure to do this may eventually cause all available memory to be filled up.

The input argument `path` should be a NULL-terminated list of null-terminated C strings, with each string containing a valid path type device name. Valid path device types include:

- transfer line
- header
- external
- valve
- pump
- node
- define path
- interface alignment.

The devices in `path` do not have to be connected together. `drGetDistPlot` searches for the shortest interconnected paths between each of the two consecutive devices in the list, and then concatenates them in the listing order to form the full path. For interface alignment, `path` must consist of a single interface alignment device name, and nothing else. See DEFINE.PATH on page 571 for a discussion of the `path` argument.

Possible return values for `*ier` are as follows:

Value	Meaning
0	operation succeeded
1	data not yet available

Value	Meaning
-1	unable to communicate with the model
-2	plot item is invalid
-3	`path` is invalid

An `*ier` value of 1 generally indicates that the requested distance plot data is not in the shared memory yet. To help avoid this situation, you can use `drupd()` beforehand to update the shared memory.

If `*ier` is returned negative, the following values were also set:

- `*points` as 0
- `*curvedata` as NULL

See Also

cdrini() on page 939

drGetTimePlot() on page 952

drGetDistPlotHist() on page 947

drFree() on page 944

drupd() on page 958

"Distance plot items" on page 172 for a list of supported distance plot items

"Distance plot items for online modeling" on page 815 for additional distance plot items related to interface alignment device

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drGetDistPlotHist

Returns the distance plot data of a simulation value for a specified item along a given path at a given historical time.

Declaration

```
void drGetDistPlotHist(char const* const* path, const char *item, double
time, int *points, drPointXY **curvedata, double *atime, int *ier)
```

Arguments

`path`	>>	path defined using a list of device names
`*item`	>>	name of item to get data, for example, PRESSURE
`time`	>>	time for the requested profile (in SPS unit)
`*points`	<<	number of data points, as returned by `drGetDistPlotHist()`
`**curvedata`	<<	pointer to an array of drPointXY data points, as returned by `drGetDistPlotHist()`
`*atime`	<<	actual time for the returning data (in SPS unit), as returned by `drGetDistPlotHist()`
`*ier`	<<	error value, as returned by `drGetDistPlotHist()`. If no error occurs, zero is returned.

Usage Considerations

Routine `drGetDistPlotTimes()` can be called before this routine to obtain a list of historical timestamps at which distance plot data are available. `drGetDistPlotTimes()` and this routine are not atomic, which means that for a circular review file, distance plot data with old timestamps could be over-written by TRANS in the duration between the calls.

The closest timestamp is used if `time` does not match exactly with any of the available timestamps. The actual time for the returning data is returned in `*atime`.

Note: Live distance plots may have a higher resolution than a historical curve because of the knot spacing stored in memory. Differences can be expected.

See DISTPLOT on page 548 for a list of available historical distance plot items.

See drGetDistPlot() on page 945 for more information.

Possible return values for `*ier` are as follows:

Value	Meaning
0	operation succeeded
-1	unable to communicate with the model
-2	=PROFILE is not specified in the INTRAN file
-3	the plot item is invalid
-4	`path` is invalid
-5	the plot item is not available from the model

If `*ier` is returned negative, the following values were also set:

- `*points` as 0
- `*curvedata` as NULL
- `*atime` as 0

See Also

cdrini() on page 939

drGetDistPlot() on page 945

drGetDistPlotTimes() on page 949

drFree() on page 944

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drGetDistPlotTimes

Returns all of the available time stamps for which historical distance plot data may be retrieved.

Declaration

```
void int *points, double **x, int *ier)
```

Arguments

`*points`	<<	number of available time stamps (size of x[]), as returned by `drGetDistPlotTimes()`
`**x`	<<	pointer to an array of doubles holding the returning values of available time (SPS time unit), as returned by `drGetDistPlotTimes()`
`*ier`	<<	error value, as returned by `drGetDistPlotTimes()`. If no error occurs, zero is returned.

Usage Considerations

This routine should be called before `drGetDistPlotHist()`. You should use the information returned from this routine to determine the time for historical profile data retrieval.

After using or copying the retrieved time values, you should use `drFree()` to free the memory allocated for the double array `*x`. Failure to do this may eventually result in filling up all available memory.

Possible return values for `*ier` are as follows:

Value	Meaning
0	operation succeeded
-1	unable to communicate with the model
-2	=PROFILE is not specified in the INTRAN file

If `*ier` is returned negative, the following values were also set:

- `*points` as 0
- `*x` as NULL

See Also

cdrini() on page 939

drGetDistPlotHist() on page 947

drFree() on page 944

“Interfacing to another GUI” on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drgetm

Returns a connection identifier `model` for the current model connection.

Declaration

```
void drgetm(int *model, int *ier)
```

Arguments

`*model`	<<	connection identifier, as returned by `drgetm()`
`*ier`	<<	error value, as returned by `drgetm()`. If an error occurs, a non-zero value is returned.

Usage Considerations

The tport version of this routine is a no-op.

The tsclient (spsclient on Windows) version supports multiple simultaneous connections with different TRANS processes. The routines do not take a connection argument, instead they act on the current connection. Use `drgetm()` to get the current connection identifier before changing it. Use `drsetm()` to restore the current connection to this stored value.

See Also

cdrini() on page 939

drdetm() on page 942

drsetm() on page 957

“Interfacing to another GUI” on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

DrGetStringList

Returns the list of strings associated with a “getval” address of a peeklist.

Declaration

```
char const* const* DrGetStringList(int addr)
```

Arguments

`addr`	>>	“getval” address, as returned by `cdrgta()`

Usage Considerations

The returning strings are pointed to by a NULL-terminated array of null-terminated C-string pointers. In the case of an error, NULL is returned.

The memory for the strings and the array of pointers can be over written without notifying the caller. Therefore, the caller should use or copy these results immediately.

The argument `addr` is obtained from a call to `cdrgta()`. For example, in the following sample, `DrGetStringList()` returns all the peek names for all of the transfer lines in the model.

```
int addr, ier;
cdrgta("PEEKLIST(\"*, K.L = T\")", &addr, &ier);
char** pipepeeks = (char **)DrGetStringList(addr);
```

Note that this code sample, however, might consume an undesirable amount of system resources if you are using a large model, perhaps with hundreds of transfer lines.

See Also

cdrini() on page 939

cdrgta() on page 936

PEEKLIST on page 621

“Interfacing to another GUI” on page 928

Library: psiml tsclient (spsclient on Windows)

Licensing: DRINIT

drGetTimePlot

Returns the history of a simulation value for a user specified time interval.

Declaration

```
void drGetTimePlot(double t_begin, double t_end, const char *item, int
*points, drPointXY **curvedata, int *ier)
```

Arguments

`t_begin`	>>	begin time (SPS time units)
`t_end`	>>	end time (SPS time units)
`*item`	>>	name of item to for which to get data, for example, PIPE1:P-
`*points`	<<	number of data points, as returned by `drGetTimePlot()`
`**curvedata`	<<	pointer to an array of drPointXY data points, as returned by `drGetTimePlot()`
`*ier`	<<	error value, as returned by `drGetTimePlot()`. If no error occurs, zero is returned.

Usage Considerations

After using or copying the retrieved data, you should use `drFree()` to free the memory allocated for the `*curvedata` array. Failure to do this may eventually result in filling up available memory.

If `t_begin` is smaller than the smallest available simulation time, the smallest available simulation time is used instead of `t_begin`. Likewise, if `t_end` is larger than the largest available simulation time, the largest available simulation time is used instead of `t_end`.

Possible return values for `*ier` are as follows:

Value	Meaning
0	operation succeeded
-1	unable to communicate with the model
-2	the plot item is not available
-3	there is no data available for any of the specified time interval
-4	`t_end`

If `*ier` is returned negative, the following values were also set:

- `*points` as 0
- `*curvedata` as NULL

See Also

cdrini() on page 939

drGetDistPlot() on page 945

drFree() on page 944

"Distance plot items" on page 172 for a list of supported distance plot items

"Distance plot items for online modeling" on page 815 for additional distance plot items related to interface alignment device

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drgetv

Retrieves the current value of a scalar-valued expression associated with the “getval” address `*addr`.

Declaration

```
void drgetv(const int *addr, float *value, int *ier
```

Arguments

`*addr`	>>	“getval” address, as returned by `cdrgta()`
`*value`	<<	returned value in user units
`*ier`	<<	error value, as returned by `drgetv()`. If no error occurs, zero is returned.

Usage Considerations

The format for a date/time value is the number of accumulated minutes since Dec 31, 1967 00:00:00.

The following values for `*ier` may be returned by `drgetv()`:

Value	Meaning
Zero	operation succeeded
`DR_NOT_AVAILABLE`	requested data not known to the client
`DR_ERROR`	value of the expression did not evaluate to a scalar value

If `*ier` is returned as `DR_NOT_AVAILABLE`, `*value` is returned as zero. In this case, a later call to this function may return the actual value. Use `cdrgts()` to retrieve string values. For more information on `DR_NOT_AVAILABLE`, see the file saiInterface.h.

See Also

cdrini() on page 939

cdrgta() on page 936

cdrgts() on page 938

saiInterface.h

“Interfacing to another GUI” on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drgtvs

Retrieves an array of values at time `*time` associated with an array of “getval” addresses [addr].

Declaration

```
void drgtvs(const double *time, const int *n, const int addr[n], double
val[n], int flag[n], int *ier)
```

Arguments

`*time`	>>	simulation time for requested information (SPS time unit)
`*n`	>>	number of values
`addr[n]`	>>	array of “getval” addresses, as returned by `cdrgta()`
`val[n]`	<<	array of scalar peek values, as returned by `drgtvs()`
`flag[n]`	<<	flags for special information about peeks, as returned by `drgtvs()`
`*ier`	<<	error value, as returned by `drgtvs()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Even though `cdrgta()` handles expressions, this routine handles only simple peeks. `flag` is set based on the method of calculation. Values flagged with `DR_EXT_FTR` or `DR_EXT_PST` have been extrapolated; otherwise intrapolation was used. Use `cdrv2s()` to convert values with `DR_STRING` flags to a status string. For more information on possible `flag` values, see the file saiInterface.h.)

See Also

cdrini() on page 939

cdrgta() on page 936

cdrv2s() on page 941

saiInterface.h

“Interfacing to another GUI” on page 928

Library: psiml tsclient (spsclient on Windows)

Licensing: DRINIT

drputv

Changes the value of a simulation variable, such as a set point or controller tuning parameter, by generating a POKE name:sffx = value command and then passing it to TRANS.

Declaration

```
void drputv(const int *addr, const float *value, int *ier)
```

Arguments

`*addr`	>>	"getval" address, as returned by `cdrgta()`
`*value`	>>	new value in user units
`*ier`	<<	error value, as returned by `drputv()`. If an error occurs, a non-zero value is returned.

Usage Considerations

The routine returns a non-zero `*ier` if it is unable to send the command. The routine has no knowledge of the eventual success or failure of the command.

Set `*value` to zero to close a valve or stop a pump and/or compressor. Set `*value` to 1 to open a valve or start a pump and/or compressor.

See Also

cdrini() on page 939

cdrgta() on page 936

cdrptc() on page 940

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drsetm

Sets a connection identifier for the current connection.

Declaration

```
void drsetm(const int *model, int *ier)
```

Arguments

`*model`	>>	connection identifier
`*ier`	<<	error value, as returned by `drsetm()`. If an error occurs, a non-zero value is returned.

Usage Considerations

The tport version of this routine is a no-op.

The tsclient (spsclient on Windows) version supports multiple simultaneous connections with different TRANS processes. The routines do not take a connection argument, instead they act on the current connection. Use `drgetm()` to get the current connection identifier before changing it. Use `drsetm()` to restore the current connection to a remembered connection identifier.

The routine will abort if the connection identifier `model` is invalid.

See Also

cdrini() on page 939, drgetm() drdetm()

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

drupd

Copies the shared memory into the local process memory for the other `dr-` routines to use.

Declaration

```
void drupd(int *update, int *ier)
```

Arguments

`*update`	<<	TRUE if memory update was necessary, as returned by `drupd()`
`*ier`	<<	error value, as returned by `drupd()`. If an error occurs, a non-zero value is returned.

Usage Considerations

To keep the local process copy of the memory current, call this routine every second or so depending on model performance. The copy takes place only if TRANS has updated shared memory since the last `drupd()` call.

See Also

cdrini() on page 939

"Interfacing to another GUI" on page 928

Library: tport tsclient (spsclient on Windows)

Licensing: DRINIT

OpcToSps

OpcToSps is OPC client software that passes live pipeline information from any OPC compliant SCADA system to GL's SPS/Statefinder product.

Statefinder acts as an extension to the SCADA system by constantly providing available pressure, flow, density, and other parameters along the pipeline. These variables can be passed back to the SCADA system just as if they came from installed field equipment. This enables the operators to observe system performance even where there is no installed telemetry.

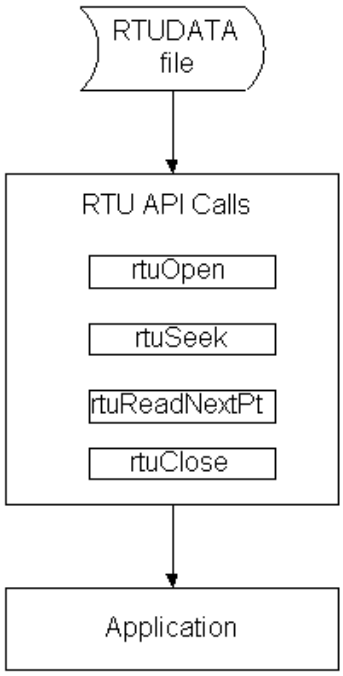
rtu* interface

Reading an RTU data file and interpreting the data in the file can assist a user to:

- Display the information on a GUI of the user's choice
- Send the information to an expert system
- Send the information to a statistical package to analyze specific data

The RTU data file is produced as described in "Providing SCADA information to a Statefinder/Leakfinder model" on page 927. The GUI reads the information in the RTU data file by using functions provided by GL. Each point in an RTU data file contains a SCADA point name, a time stamp, a value, and a quality flag. The functions can scan all the SCADA points in an RTU data file or only specified SCADA points.

The basic flow of information between the RTU data file and the receiver of information is shown below.



See these functions for implementation details:

Function	Description
rtuOpen() on page 961	Establishes a session with an RTU data file
rtuSeek() on page 963	Positions the session pointer to a time stamp in an RTU data file
rtuReadNextPt() on page 962	Reads the next data point for a given session
rtuClose() on page 960	Closes an RTU data file

rtuClose

Closes `session` with an rtu data file.

Declaration

```
voidrtuClose(int session, int *ier)
```

Arguments

`session`	>>	session handle, as returned by `rtuOpen()`
`*ier`	<<	error value, as returned by `rtuClose()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Consider logging a non-zero value of `*ier` to a file and allowing the client to continue processing.

This routine releases data structures allocated by `rtuOpen()`.

See Also

rtuOpen() on page 961

Library: dreml

Licensing: TODREM

rtuOpen

Establishes a session with the rtu data file `filename`.

Declaration

```
void rtuOpen(const char *filename, openKeys openkey, int mxpts, int
*session, int *ier)
```

Arguments

`*filename`	>>	rtu data filename
`openkey`	>>	file open mode
`mxpts`	<<	not used
`*session`	<<	session handle, as returned by `rtuOpen()`
`*ier`	<<	error value, as returned by `rtuOpen()`. If an error occurs, a non-zero value is returned.

Usage Considerations

`READ_ONLY` is the only valid value for `openkey`. In addition, since `mxpts` is not used with the `READ_ONLY` key, `mxpts` is currently unused.

`rtuClose()` releases data structures allocated by `rtuOpen()`.

See Also

rtuClose() on page 960

Library: dreml

Licensing: TODREM

rtuReadNextPt

Reads the next data point for the given `session`.

Declaration

```
void rtuReadNextPt(int session, int *ptnum, int *rtime, char *ident, float
*value, int *qual, int *ier)
```

Arguments

`session`	>>	session handle, as returned by `rtuOpen()`
`*ptnum`	<<	point number, as returned by `rtuReadNextPt()`
`*rtime`	<<	point time, as returned by `rtuReadNextPt()`
`*ident`	<<	point name, as returned by `rtuReadNextPt()`. The storage must be provided by the caller.
`*value`	<<	point value, as returned by `rtuReadNextPt()`
`*qual`	<<	point quality, as returned by `rtuReadNextPt()`
`*ier`	<<	error value, as returned by `rtuReadNextPt()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Possible return values for `*ier` are as follows:

Value	Meaning
zero	operation succeeded
2	invalid `session`
3	no data available

See Also

rtuOpen() on page 961

rtuSeek() on page 963

Library: dreml

Licensing: TODREM

rtuSeek

Positions the session pointer to `rtime`.

Declaration

```
void rtuSeek(int session, int rtime, int *ier)
```

Arguments

`session`	>>	session handle, as returned by `rtuOpen()`
`rtime`	>>	time to seek to
`*ier`	<<	error value, as returned by `rtuSeek()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Set `rtime` to `BIGTIME` to position to the cusp and read entire file. Set `rtime` to `SMALLTIME` to position to the cusp and wait for new data. `BIGTIME` and `SMALLTIME` are defined in saiInterface.h)

This routine returns a non-zero value for `*ier` if `session` is invalid.

See Also

rtuOpen() on page 961

saiInterface.h

Library: dreml

Licensing: TODREM

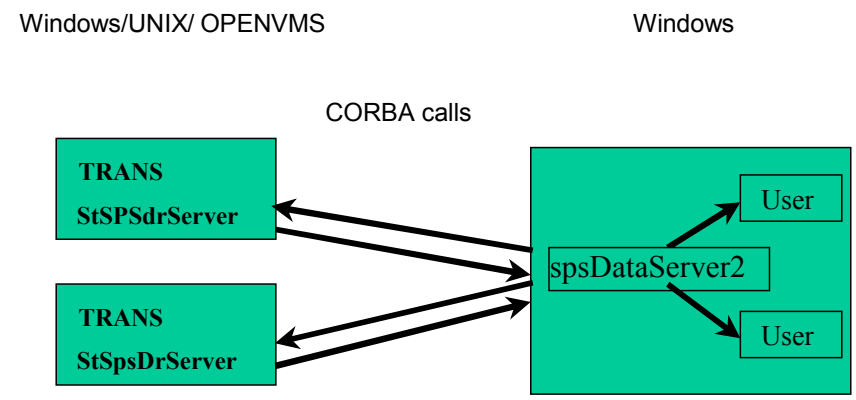
spsDataServer2

The spsDataServer2 is an in-process, automation server that allows user interaction with SPS through any application that supports automation, such as Microsoft Visual Basic®, Access, Excel, or Internet Explorer.

In conjunction with the SPS RPC server process TRANSSHARE, the spsDataServer2 allows applications on a Windows® system to communicate through Remote Procedure Call (RPC) calls with one or more SPS models running in TRANS anywhere on the same local area network.

The spsDataServer2 is a DLL file-based server named spsServer2.dll, and is registered as an object in the Windows system registry where it can be referenced by other applications. While the spsDataServer2 must be registered on a PC running Windows, SPS models must run on Windows.

The spsDataServer2 was implemented using the client/server version of the documented low-level SPS API.



The sps DataServer2 API allows you to perform the following tasks:

- Connect or disconnect with an SPS model. (See AttachWithRetry on page 967, Detach on page 968, and Update on page 980.)
- Send a command to an SPS model. (See SendCommand on page 979.)
- Get data from and SPS model (See GetValue on page 978, GetNumValue on page 975, GetStringValue on page 976, GetNumValues on page 974, GetStringValues on page 976, and GetHistValue on page 973.)
- Get distance and time plots (See GetDistancePlot on page 970, GetTimePlot on page 977, GetHistDistancePlotTimes on page 972, and GetHistDistancePlot on page 972.)
- Get device and attribute information (GetDeviceNames on page 969, GetDeviceTypes on page 970, and GetAttributes on page 968.)
- Debug the spsDataServer2 (See DebugLevel on page 967 and LogFileName on page 979.)

This section describes:

Connecting spsDataServer2 Objects to an SPS model

The following Visual Basic code creates an instance of the spsDataServer2 object:

```
Dim sps As spsDataServer2
Set sps = New spsDataServer2
```

After an instance of the spsDataServer2 object has been created, its properties and methods may be used.

To connect to an active SPS model

Connect to the SPS model using the `Attach` method:

```
Dim Error As Long
Error = sps.Attach("spslx::/st/tatooine/c:\sps\v9.30\models\trainer")

If (Error) Then 'try again
```

Note: The `Attach` method should return "0" as long as the TRANSSHARE process is ready to accept requests.

Note: The spsDataServer2 maintains the connection based on the *connection string* passed to the `Attach` method. This connection string consists of the data source url specifier, and the registered server name from TRANSSHARE. The current available data sources are spslx and gasSolver.

To disconnect from the current SPS model

Disconnect from the SPS model using the `Detach` method:

```
Dim Error As Long
Error = sps.Detach
```

Tip: To prevent wasting resources, the `Detach` method should be called at the end of program execution if an Attach method has been called, and any time the current connection is no longer needed. The Attach method will automatically detach from an existing connection before attaching to the new connection.

Connection method example

The following example demonstrates how the connection methods work.

```
Dim sps As spsDataServer2
Set sps = New spsDataServer2
Dim Error As Long

' Attach to an Sps model named Model1
Error = sps.Attach("spslx::Model1")
If (Error = 0) Then
  … ` Get some data from Model1
End If

' Detach from Model1 and attach to a Gas Solver model Model2
Error = sps.Attach("gasSolver::Model2")
If (Error = 0) Then
  … ` Get some data from Model2
End If

Error = sps.Detach
If (Error = 0) Then
  … ` Now you are left with no more connections
End If
```

Installation and setup of spsDataServer2

Depending on your installation of SPS or other GL products, files required to run the spsDataServer2 are typically installed and registered, as needed, on your PC and *require no manual registry entries*. However, if for some reason you do need to copy and register some of these files manually, instructions have been provided below.

To register spsDataServer2 and related files

Note: You should be familiar with MS-DOS commands.

1 Open an MS-DOS command prompt.
2 Change the directory to the bin of latest version of SPS installed.
3 Execute the command: Regsvr32 spsServer2.dll.

To unregister spsDataServer2 and related files

Note: You should be familiar with MS-DOS commands.

1 Open an MS-DOS command prompt.
2 Change the directory to the bin of latest version of SPS installed.
3 Execute the command: Regsvr32 /u spsServer2.dll.

spsDataServer2 functions

Attach

This method sets the current connection of the spsDataServer2 object to the TRANSSHARE server identified by strTransshareName in the SPS Lexicon Service.

Parameters

```
[IN] strTransshareName As String
```

`strTransshareName` is the registered name of the TRANSSHARE server in the SPS Lexicon Service.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrini or drsetm

Example 1

```
Dim sps As spsDataServer2
Set sps = New spsDataServer2
Dim Error As Long
Error = sps.Attach("Model")
```

Example 2

```
Dim sps As spsDataServer2
Set sps = New spsDataServer2
Dim Error As Long
```

```
Error = sps.Attach("spslx::/st/tatooine/c:\sps\v9.30\models\trainer")
```

AttachWithRetry

This method sets the current connection of the spsDataServer2 object to the TRANSSHARE server identified by strTransshareName in the SPS Lexicon Service. This method allows for multiple retries if the previous attempts fail.

Parameters

```
[IN] strTransshareName As String
```

`strTransshareName` is the registered name of the TRANSSHARE server in the SPS Lexicon Service.

```
[IN] INumOfRetry As Long
```

`INumOfRetry` is the number of the connection attempts to make before failing the attach.

```
[IN] ISleepInterval As Long
```

`ISleepInterval` is the sleep interval in milliseconds between connection retries.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrini or drsetm

Example

```
Dim sps As spsDataServer2
Set sps = New spsDataServer2
Dim Error As Long
Error = sps.Attach("Model")
```

DebugLevel

This long property obtains or changes the debug level for logging spsDataServer2 messages. You *must* set a `LogFileName` property before spsDataServer2 messages are written to a file.

`DebugLevel` can have the following values:

- 0 when no message is logged. (This is the default value.)
- 1 when any change(s) in connection, log file name, debug level, or any system error is logged.

- 2 when any error occurred in the methods (such as an invalid model data name passed to the GetValue method) is also logged.
- 3 when the arguments passed to all of the methods are also logged.

Example

```
Dim DebugLevel As Long
DebugLevel = sps.DebugLevel
If (Not DebugLevel) Then
  sps.DebugLevel = 1
End If
```

Detach

This method detaches from the current model.

Parameters

none

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drdetm

Example

```
Dim Error As Long
Error = sps.Detach
```

GetAttributes

This method returns a list of strings for simulation devices attributes that are associated with the device name specified.

Parameters

```
[IN]  strDeviceName As String
[OUT] pvarAttributes As Variant
[IN]  IExclAttribs As Constant
```

`strDeviceName` is the string to specify the device name.

`pvarAttributes` is an Array of Strings stored in a Variant. This array contains the list of attribute names matching the device name specified.

`IExcAttribs` is a constant specifying whether to exclude device attributes that are not trendable. The default is “0”. To retrieve trendable attributes, set this parameter to 1 or “EANonTrending”.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetStringList

Example 1

```
Dim Attributes As Variant
Dim Error As Long

Error = sps.GetAttributes(“PIPE1”, Attributes)
```

Example 2

```
Dim Attributes As Variant
Dim Error As Long

Error = sps.GetAttributes(“PUMP1”, Attributes, 1)
```

GetDeviceNames

This method returns a list of strings for simulation devices names that match the device type specified.

Parameters

```
[IN}  strDeviceType As String
[OUT] pvarDeviceNames As Variant
```

`strDeviceType` is the string to specify the device type.

`pvarDeviceNames` is an Array of Strings stored in a Variant. This array contains the list of device names matching the device type specified.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetStringList

Example

```
Dim DeviceNames As Variant
Dim Error As Long

Error = sps.GetDeviceNames("T", DeviceNames)
```

GetDeviceTypes

This method returns a list of strings for simulation devices types that exist in the attached model.

Parameters

```
[OUT] pvarDeviceTypes As Variant
```

`pvarDeviceTypes` is an Array of Strings stored in a Variant. This array contains the list of device types in the attached model.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetStringList

Example

```
Dim DeviceTypes As Variant
Dim Error As Long

Error = sps.GetDeviceTypes(DeviceTypes)
```

GetDistancePlot

This method returns the profile of a simulation value for a specified item along a given path.

Parameters

```
[IN}  Path As Variant
[IN]  PlotItem As String
[OUT] NumPoints As Long
[OUT} XData As Variant
[OUT] YData As Variant
```

`Path` is an Array of Strings stored in a Variant. This array contains the list of device names that make up the distance plot.

`PlotItem` is the name of the item to get data for, such as pressure.

`NumPoints` is the number of points in each of the returned arrays (XData and YData).

`XData` is the array of X (distance) values stored in a Variant.

`YData` is the array of Y values stored in a Variant.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetDistPlot

Example

```
Dim Path As Variant
Dim PlotItem As String
Dim NumPoints As Long
Dim Xdata As Variant
Dim Ydata As Variant
Dim Error As Long

ReDim Path(2) As String
Path(0) = "Pipe1"
Path(1) = "B1"
Path(2) = "Pipe2"
PlotItem = “Pressure”

Error = sps.GetDistancePlot(Path, PlotItem, NumPoints, Xdata, Ydata)
```

GetHistDistancePlotTimes

Description

This method returns an array of times when profiles are available.

Parameters

```
[OUT] NumPoints As Long
[OUT] TimeArray As Variant
```

`NumPoints` is the number of available time stamps.

`TimeArray` is the array of times at which profile data was saved (SPS time unit)

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetDistPlotTimes

Example

```
Dim NumPoints As Long
Dim TimeArray As Variant
Dim Error As Long

Error = sps.GetDistPlotTimes(NumPoints, TimeArray)
```

GetHistDistancePlot

This method returns the distance plot data of a simulation value for a specified item along a given path at some historical simulation time.

Note: Live distance plots may have a higher resolution than a historical curve because of the knot spacing stored in memory. Differences can be expected.

Parameters

```
[IN]  Path As Variant
[IN]  PlotItem As String
[IN]  ReqTime as Double
[OUT] NumPoints As Long
[OUT] XData As Variant
```

```
[OUT] YData As Variant
[OUT] ActualTime As Double
```

`ReqTime` is the time (in SPS time units) at which the distance plot is desired.

`ActualTime` is the actual time (in SPS time units) corresponding to the returned distance plot values.

For other parameters, see GetDistancePlot on page 970.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetDistPlotHist

Example

```
Dim Path As Variant
Dim PlotItem As String
Dim ReqTime As Double
Dim NumPoints As Long
Dim Xdata As Variant
Dim Ydata As Variant
Dim ActualTime As Double
Dim Error As Long

ReDim Path(2) As String
Path(0) = "Pipe1"
Path(1) = "B1"
Path(2) = "Pipe2"
PlotItem = “Pressure”
ReqTime = 100

Error = sps.GetHistDistancePlot(Path, PlotItem, ReqTime, NumPoints, Xdata,
Ydata, ActualTime)
```

GetHistValue

This method retrieves an array of scalar values associated with the input array of names at the specified time.

Parameters

```
[IN]  Time As Double
[IN]  N As Long
[IN]  PeekName As Variant
```

```
[OUT] Value As Variant
[OUT] Flag As Variant
```

`Time` is the model simulation time for requested information in SPS time units.

`N` is the size of the array, meaning the number of data points requested.

`PeekName` is an array of strings stored in a Variant. This array contains the list of names of the data points.

`Value` is the array of Variants stored in a Variant. This array contains the list of return values and data types (String or Double) associated with the list of names for the requested data points.

`Flag` is the array of Long stored in a Variant. This array contains the list of return statuses associated with the corresponding list of names for the requested data points.

Return value

```
Error as Long
```

The `Error` is 0 if the historical values are retrieved successfully.

Low-level API equivalent

cdrgta, drgtvs, and cdrv2s

Example

```
Dim Time as Double
Dim N as Long
Dim PeekName As Variant
Dim Value As Variant
Dim Flag As Variant
Dim Error As Long

ReDim PeekName(2) As String
PeekName(0) = "Pipe1:P-"
PeekName(1) = "Pipe2:Q-"
PeekName(2) = "B1:ST"
N = 3
Time = 0 'Minutes since 00:00:00 Dec 31, 1967

Error = sps.GetHistValue(Time, N, PeekName, Value, Flag)
```

Usage considerations

Consider caching the “Getval” addresses internally.

GetNumValues

This method retrieves an array of current numeric values associated with an array of input expressions from the current SPS model.

Parameters

```
[IN]  Expr As Variant
[OUT] Value As Variant
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrgta, drgetv, cdrgts, and DrGetStringList

Example

```
Dim Error As Long
Dim varValue As Variant
Dim varName As Variant
ReDim varName(2)
VarName(0) = "PIPE1:P-
VarName(1) = "PIPE2:P-
VarName(2) = "PIPE3:P-
Error = sps.GetValue(varName, varValue)
```

GetNumValue

This method retrieves a current numeric value associated with the input expression from the current SPS model.

Parameters

```
[IN]  Expr As String
[OUT] Value As Variant
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrgta, drgetv, cdrgts, and DrGetStringList

Example

```
Dim Error As Long
Dim Value As Variant
Error = sps.GetNumValue("PIPE1:P-", Value)
```

GetStringValues

This method retrieves an array of current string values associated with an array of input expressions from the current SPS model.

Parameters

```
[IN]  Expr As Variant
[OUT] Value As Variant
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrgta, drgetv, cdrgts, and DrGetStringList

Example

```
Dim Error As Long
Dim varValue As Variant
Dim varName As Variant
ReDim varName(2)
VarName(0) = "PUMP1:ST
VarName(1) = "PUMP2:ST
VarName(2) = "PUMP3:ST
Error = sps.GetValue(varName, varValue)
```

GetStringValue

This method retrieves a current string value associated with the input expression from the current SPS model.

Parameters

```
[IN]  Expr As String
[OUT] Value As Variant
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrgta, drgetv, cdrgts, and DrGetStringList

Example

```
Dim Error As Long
Dim Value As Variant
Error = sps.GetStringValue("PUMP1:ST", Value)
```

GetTimePlot

This method returns the history of a specific simulation value.

Parameters

```
[IN]  BeginTime As Double
[IN]  EndTime As Double
[IN]  PeekName As String
[OUT] NumPoints As Long
[OUT] XData As Variant
[OUT] YData As Variant
```

`BeginTime` is the beginning time (SPS time units).

`EndTime` is the end time (SPS time units).

`PeekName` is the name of the item to get data for, e.g., “Pipe1:P-”.

`NumPoints` is the number of points in each of the returned arrays (`XData` and `YData`).

`XData` is the array of X (time) values.

`YData` is the array of `PeekName` values.

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drGetTimePlot

Example

```
Dim BeginTime As Double
Dim EndTime As Double
Dim PeekName As String
Dim NumPoints As Long
Dim Xdata As Variant
Dim Ydata As Variant
Dim Error As Long

PeekName = "PIPE1:P-"
BeginTime = 0
EndTime = 100

Error = sps.GetTimePlot(BeginTime, EndTime, PeekName, NumPoints, Xdata,
Ydata)
```

GetValue

This method retrieves a current value associated with the input expression from the current SPS model.

Parameters

```
[IN]  Expr As String
[OUT] Value As Variant
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrgta, drgetv, cdrgts, and DrGetStringList

Example

```
Dim Error As Long
Dim Value As Variant
Error = sps.GetValue("PIPE1:P-", Value)
If (Error = 0) Then
  If VarType(Value) = vbString then
    ... ` You got a String value
  ElseIf VarType(Value) = vbSingle then
```

```
    ... ` You got a scalar value
  ElseIf VarType(Value) = vbArray + vbString then
    ... `You got an array of String value
  Else
    ... ` Something must be wrong
  End If
ElseIf (Error = 1) Then
  ... ` Expression is valid, but data is not available now, try later
Else
  ... ` Expression is not known to the model
End If
```

LogFileName

This string property specifies the file name in which the spsDataServer debug messages are logged. If you do not specify this property, no debug messages are logged, regardless of the debug level.

Example

```
If (varType(sps.LogFileName) = vbEmpty) Then
  sps.LogFileName = "spsServer2.log"
End If
```

SendCommand

This method sends a TRANS interactive command to the currently connected SPS model.

Parameters

```
[IN] strCommand As String
```

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

cdrptc

Example

```
Error = sps.SendCommand("archive model1.ark")
Error = sps.SendCommand("Poke Pipe1:P- = 500")
```

Update

This method forces the local memory of the RPC server process (which the spsDataServer2 object currently connects to) to be synchronized with TRANS.

Parameters

none

Return value

```
Error as Long
```

`Error` can have the following values:

- 0 when successful
- -1 when unsuccessful

Low-level API equivalent

drupd

Example

```
Dim Error As Long
Error = sps.Update
```

StOPC

StOPC, OPC server software developed by GL, allows OPC Data Access clients, such as SCADA systems, to communicate with GL's simulation software, SPS.

StOPC provides the communications link between OPC clients and SPS. It provides an OPC server interface (for talking to OPC clients), an SPS client interface (for talking to SPS models), and a user interface (for editing settings and viewing activity).

For more information on StOPC, contact GL.

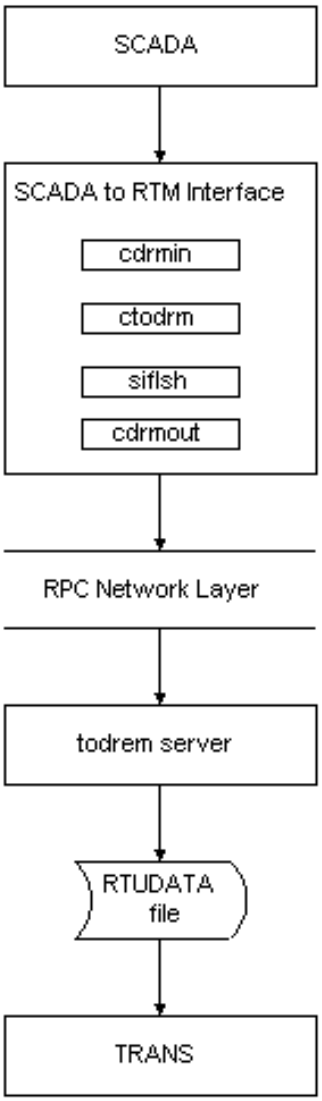
todrem interface

The todrem interface is used to send SCADA data to a real time model.

During start-up, the interface program calls a GL provided function `cdrmin()`. This function is responsible for the initialization of the SCADA to Statefinder/Leakfinder communications.

Upon the completion of each SCADA scan, the interface program processes each of the points. For each point that has been polled, a GL-provided function `ctodrm()`, is called to pass the data to the modeling environment. The `ctodrm()` routine is used for recording both analog and discrete values.

The basic flow of information between the SCADA system and the Statefinder/Leakfinder model is shown in the figure below.



Note: The process is exactly the same with or without the RPC layer.

The `ctodrm()` function stores the SCADA point data in memory. When a sufficient amount of data has been received the data are output to the RTU data file shared with the Statefinder/Leakfinder model. This file is circular thereby saving a specified amount of data. As new data are received the oldest data are overwritten.

After processing all of the data for a given scan, a GL-provided function, `siflsh()`, is called one time. This function ensures that any data still in the program memory at the end of a scan update are written to the RTU data file.

The SCADA to Statefinder/Leakfinder data interface program performs the following tasks:

1 Establishing and maintaining communications with the SCADA to Statefinder/Leakfinder interface software,

2 Upon start-up, calling the GL initialization function, `cdrmin()`, with the output file name, the logical unit number for the output file, and the maximum size of the output file (number of point-value records to store). If

the output file does not already exist, the file is created and initialized. If the output file is found on disk, `cdrmin()` opens the file and locates the position of the last written record.

3 Applying a bit mask to the SCADA point status word to assign a model data quality indicator. The data quality indicator is assigned one of the following values:

Value	Meaning
1	GOOD
2	Manual (indicating that a SCADA-scanned point value has been overwritten in the SCADA system's database with a manually entered value)
3	Nochange1 (passes old value, with an indication that the value did not change)
4	Nochange2 (indicates that the value did not change, value not available)
Anything else	BAD (scan inhibited, bad communications, etc.)

4 Upon each SCADA update for each SCADA point that has been scanned, calling `ctodrm()`, passing the point ID, time-tag, value, and model data quality indicator.

See these functions for implementation details:

Function	Description
`cdrmin()` on page 983	Opens an rtu data file
`cdrmout()` on page 984	Closes an rtu data file
`ctodrm()` on page 985	Writes SCADA data to a RTU data file
`siflsh()` on page 987	Flushes stream buffer to an RTU data file

cdrmin

Connects to an rtu data file for writing, either by connecting to an existing todremserve process, or by creating a todremserve process on the local machine. The connection is suitable for subsequent writing by `ctodrm()`. If the rtu data file does not exist, it is created. Otherwise, this routine positions the file pointer just past the most recently written SCADA point.

Declaration

```
void cdrmin(const char *filename, int *ichan, const int *mxpts, int *ier)
```

Arguments

`*filename`	>>	filename or connection information: "hostname:portnumber:protocol:version"n
`*ichan`	<< >>	channel number used to open file
`*mxpts`	>>	number of ident/tstamp/value/qualit updates the rtu data file will hold before wrapping around
`*ier`	<<	error value, as returned by `cdrmin()`. If no error occurs, zero is returned.

Usage Considerations

For flexibility, consider allowing the user to specify `filename` and `*mxpts` via an environment variable and/or command line option.

Only one rtu data file can be open for writing by any particular process.

The `*mxpts` argument is used only when the rtu data file is being created. Use 0 for `*mxpts` to get a default of 1,000,000.

Use 0 for `*ichan` to have this routine select and return a channel not currently in use. Channel conflicts can occur in code that is processed after this routine.

You should use `cdrmout()` to close the rtu data file and release resources allocated by this routine.

See Also

ctodrm() on page 985

siflsh() on page 987

cdrmout() on page 984

"Providing SCADA information to a Statefinder/Leakfinder model" on page 927

Library: spsclient stclib

Licensing: TODREM

cdrmout

Closes an rtu data file and releases resources associated with the file.

Declaration

```
void cdrmout(int ichan, int *ier)
```

Arguments

`ichan`	>>	channel number, as used and returned by `cdrmin()`
`*ier`	<<	error value, as returned by `cdrmout()`. If an error occurs, a non-zero value is returned.

Usage Considerations

Use this routine instead of closing the file directly with a system call.

See Also

cdrmin() on page 983

siflsh() on page 987

ctodrm() on page 985

"Providing SCADA information to a Statefinder/Leakfinder model" on page 927

Library: spsclient stclib

Licensing: TODREM

ctodrm

Writes data (usually SCADA data) to an rtu data file for subsequent reading by Statefinder.

Declaration

```
void ctodrm(const int *ichan, const char *ident, const int *tstamp, const
float *value, const int *qualit, int *ier)
```

Arguments

*ichan	>>	channel number, as returned by cdrmin()
*ident	>>	SCADA point name
*tstamp	>>	date/time stamp
*value	>>	process value in engineering units
*qualit	>>	data quality indicator. For valid values, see "Usage Considerations" below.
*ier	<<	error value, as returned by ctodrm(). If an error occurs, a non-zero value is returned.

Usage Considerations

This routine should be called every SCADA scan for every SCADA point. This routine records the point name, value, time, and quality to the rtu data file. For efficiency, this routine stores several ident/value/tstamp/qualit updates in a memory buffer then writes them all at once to the circular rtu data file when the memory buffer fills up.

When writing the buffer, if the next file record is past the logical end of the file as defined by cdrmin(), ctodrm() will "wrap around" to overwrite the first data record in the file.

The value of *ichan passed into ctodrm() must be the same as the *ichan argument to cdrmin(). The time for the *tstamp argument is accumulated seconds since December 31, 1967 00:00:00.

A non-zero value for *ier should be logged to a file by the caller, along with the other arguments, but the caller should continue processing subsequent points.

Valid values for *qualit are as follows:

Value	Meaning
1	GOOD
2	MANUAL
3	NOCHANGE1
4	NOCHANGE2
(other)	BAD

See Also

cdrmin() on page 983

siflsh() on page 987

cdrmout() on page 984

“Providing SCADA information to a Statefinder/Leakfinder model” on page 927

Library: spsclient stclib

Licensing: TODREM

siflsh

Flushes the memory buffer that is maintained by `ctodrm()` by writing the memory buffer to channel `ichan`.

Declaration

```
void siflsh(const int *ichan, int *ier)
```

Arguments

`*ichan`	>>	channel number to flush, as returned by `cdrmin()`
`*ier`	<<	error value, as returned by `siflsh()`. If an error occurs, a non-zero value is returned.

Usage Considerations

If there can be a relatively long delay between calls to `ctodrm()`, then `siflsh()` should be called before the delay begins. On the other hand, if each call to `ctodrm()` is followed by another call to `ctodrm()` at more or less the same delay between them, then it is not necessary to call `siflsh()` at all.

If a non-zero value for `*ier` is returned, consider having it logged to a file and allowing the client to continue processing.

See Also

cdrmin() on page 983

ctodrm() on page 985

cdrmout() on page 984

"Providing SCADA information to a Statefinder/Leakfinder model" on page 927

Library: spsclient stclib

Licensing: TODREM

TRANSHEARTBEAT

TRANSHEARTBEAT is used to determine whether the real time model is still running, and if running, to confirm that it is not very far behind real time.

The `drbeat()` routine attempts to connect to the Statefinder/Leakfinder model. Then, if successful, `drbeat()` returns the shared memory age (how long since shared memory has been updated), real time model lag time (how far behind real time the model is), and the real time model status. See "Source code for drbeat()" on page 988.

TRANSHEARTBEAT program

TRANSHEARTBEAT is a program that uses the drbeat() routine to check the status of the Statefinder/Leakfinder model. TRANSHEARTBEAT supports the following command line options:

```
TRANSHEARTBEAT casename
```

```
-DIR=model-directory
-FROZEN=maximum-frozen-seconds
-MAXLAG=maximum-lag-seconds
-TRIES=number-of-tries
-WAIT=seconds-between-tries
```

TRANSHEARTBEAT is basically a loop that calls drbeat() “number-of-tries” times and waits “seconds-between-tries” between calls. TRANSHEARTBEAT prints out the results of the drbeat() call after every call.

Source code for drbeat()

The drbeat() routine is implemented using the low-level API. The source code for drbeat() is included here as an example of how to use the low-level API routines.

```
//drbeat.cpp

//  drbeatStatus drbeat
//   (
//     char* case_name  --> trans case name to connect to
//     char* model_dir  --> directory that trans is running in
//     int max_age      --> max age of data before called "frozen" (sec)
//     int max_lag      --> max lag before being called "far behind" (sec)
//     int& age        <--  age of shared memory (sec)
//     int& lag        <--  how far behind model is (sec)
//   )
//
//  Purpose
//  -------
//  This routine tries to connect to and fetch data from a running
//  trans session to see if it is still alive.
//
//  Returns
//  -------
//  An indication of the trans status
//   COULD_NOT_CONNECT = 1    // connection, update, or lookup failed
//   FROZEN            = 2    // model TIME has not changed in a while
//   FAR_BEHIND,       = 3    // model is far behind real time
//   RUNNING           = 4    // model seems okay
//
//  Usage Considerations
//  --------------------
//  transshare program must be running for the connection to
//  work.
//
//  This routine uses model TIME and model SYSTIME; these
//  values are accurate only to the nearest minute, due to
//  rounding error.  Hence, age and lag are accurate only to
//  the nearest minute.
//
//  Time Zone Treatment
//  -------------------
//  This routine, drbeat(), is not cognizant of time zones.  This
//  fact means that the system time and model time must be consistent
```

```
//  for the model lag time to be calculated properly.  In practice,
//  this requires that DREMTIMEZONE be set to LOCAL
//  Note that
//  DREMTIMEZONE of STANDARD will not work because in this case the
//  system time and model time are out of synchronization by whatever
//  time zone the system is in.
//
//  See Also
//  -------
//  cdrgta() cdrini() drdetm() drgetv() drupd()
//  saiInterface.h
//
//  Library
//  -------
//  spsclient.lib
//
//  Licensing
//  ---------
//  DRINIT
//
#include "saiInterface.h"
#include <time.h>
#if unix
#  include <unistd.h>
#else
#  include <direct.h>   //chdir, getcwd
#endif
#include <stdlib.h>
enum drbeatStatus
{
  COULD_NOT_CONNECT = 1,   // connection, update, or lookup failed
  FROZEN,                  // model TIME has not changed in a while
  FAR_BEHIND,              // model is far behind real time
  RUNNING                  // model seems okay
};
EXTERNC int drbeat
(
  char* case_name,  //  --> trans case name to connect to
  char* model_dir,  //  --> directory that trans is running in
  int max_age,      //  --> maximum frozen time (sec)
  int max_lag,      //  --> maximum lag time (sec)
  int& age,         // <--  delta real time since last shmem update (sec)
  int& lag          // <--  how far behind model is (sec)
)
{
  int connected = 0;
  int ier = 0;
  int systime_addr = 0;
  int time_addr = 0;
  const char* thecase = case_name ? case_name : "bc";
  char* thedir  = model_dir;
  char* home_dir;
  // initializations for early exit
  drbeatStatus trans_status = COULD_NOT_CONNECT;
```

```
  age = 1000000000;
  lag = age;
  while(1)  // bogus loop to avoid deeply nested if's
    {
      home_dir = getcwd(NULL, 80);// need to chdir back here

      if(! home_dir) break;
      if(chdir(thedir)) break;                  // attach to directory

      cdrini(thecase, &ier);                    // attach to model
      if(ier) break;

      connected = 1;
      logical update_flag;
      drupd(&update_flag, &ier);                // update from shmem
      if(ier) break;

      float mod_systime_min;
      cdrgta("SYSTIME", &systime_addr, &ier);  // look up "SYSTIME"; get
value
      if(ier) break;

      drgetv(&systime_addr, &mod_systime_min, &ier);
      if(ier) break;

      float mod_time_min;
      cdrgta("TIME", &time_addr, &ier);         // look up "TIME"; get value
      if(ier) break;

      drgetv(&time_addr, &mod_time_min, &ier);
      if(ier) break;

      // trans units are minutes since 31 Dec 1967 00:00:00 GMT
      // unix  units are seconds since 01 Jan 1970 00:00:00 GMT
      int base = (1 + 366 + 365)*24*60;
      //           \     \     \
      //           1968  1969  1970
      // convert SYSTIME and TIME to unix base
      int mod_systime_unix = int( 60*(mod_systime_min - base) );
      int mod_time_unix    = int( 60*(mod_time_min - base) );
      int my_systime = int(time(NULL));
      age = my_systime - mod_systime_unix; //how much realtime since update
      lag = my_systime - mod_time_unix;      // how far behind model is
      if(age > max_age)
           trans_status = FROZEN;
      else if(lag > max_lag)
           trans_status = FAR_BEHIND;
      else
           trans_status = RUNNING;
      break;
    }
  if(time_addr)
    drfrea(&time_addr, &ier);
  if(systime_addr)
```

```
    drfrea(&systime_addr, &ier);
  if(connected)  // don't detach unless connection worked
    drdetm(&ier);
  if(home_dir)
    {
      chdir(home_dir);
      free(home_dir);
    }
  return trans_status;
}
```

TRANSEXPORT interface

The Statefinder/Leakfinder/Predictor model communicates with the SCADA system using the TRANSEXPORT program, the data and alarm export program. This communication is necessary to update the SCADA information from the running model and to update alarm messages in SCADA.

The model data and alarm export interface is implemented within the online modeling system environment. For each model in the online modeling system environment there are individual model data and alarm export processes.

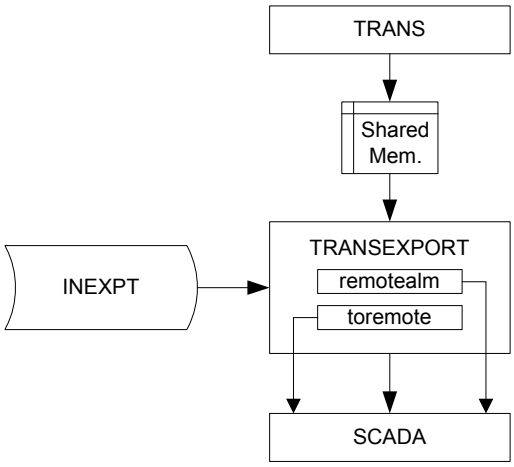
For each product (Statefinder, Predictor, etc.) from which data is to be extracted, TRANSEXPORT is run in parallel with TRANS. TRANSEXPORT reads an ASCII model export configuration file (INEXPT) upon start-up. The INEXPT file defines the model variable names for which the values are to be exported, and the corresponding "foreign" (i.e., non-model) database variable names. The configuration file also identifies which alarms are to be exported. The actual alarm definition, including alarm name, alarm condition, and text messages, are defined in the model's simulation input file (INTRAN). A third-party (typically the SCADA vendor) provides two functions, toremote() on page 996 and remotealm() on page 995. These functions perform the actual transfer of data when called by TRANSEXPORT.

Upon startup, a model spawns a separate TRANSEXPORT process if it contains data or alarms that need to be exported. This process communicates with its parent TRANS model process via shared memory. For more information, see "Shared memory" on page 57.

When starting up, TRANSEXPORT reads the INEXPT file. The INEXPT file describes the content of the desired transfer.

The actual export of data and alarms from TRANSEXPORT is performed by third-party software. TRANSEXPORT's interface to this software is via two functions, `toremote()` and `remotealm()`. At each data update interval, TRANSEXPORT calls `toremote()` and passes the most recently calculated model data. The `toremote()` function is responsible for any subsequent data transfer. Each time a selected model alarm message is updated, TRANSEXPORT calls `remotealm()`. The `remotealm()` function is responsible for the transfer of each of the new alarm messages to its desired destination.

When configuring the modeling computers during final installation, it is necessary to confirm that appropriate execution priorities are assigned to the TRANS, TRANSEXPORT, and other modeling computer-resident processes. This is intended to ensure that all desired model-generated data and alarms are exported. This is an important system configuration consideration. For example, if TRANS were assigned an inappropriately high priority relative to TRANSEXPORT, alarms and data updates could be missed.



TRANSEXPORT performs the following tasks:

1 Reads the INEXPT file upon start-up. (The INEXPT file typically contains between 200 and 2000 model variables with a corresponding number of SCADA point names and approximately 200 to 2000 alarm names.)

2 Attaches to the parent model's shared memory and validates the availability of the selected model variables and alarms upon startup,

3 Calls the third-party function, `toremote()`, for all of the variables specified with the EXPORTDATA command, according to the export data update period (specified in the INEXPT file).

4 Evaluates all of the model alarms upon the completion of each time step,

5 Calls the third-party subroutine, `remotealm()` for each alarm with a new message. A "new message" will result when an alarm condition is first generated, each time the repeat message is repeated (only if a repeat period is specified and the model time is included in the alarm text), and when the alarm condition is cleared (if a clear message is specified).

See these functions for implementation details:

Function	Description
transexport()	Parses INEXPT file. Periodically calls toremote and remotealm.
toremote()	Exports data
remotealm()	Exports alarm

transexport

This is the control routine for transexport applications. It establishes the connection with SPS, parses the INXREF file, and calls remotealm() and toremote() as appropriate. For more information, see "Statefinder and Leakfinder" on page 741.

Declaration

```
int transexport(int argc, char const*const* argv,
   void(*toremote)(const char*pvname[], const double dval[], int n, int*
        err),
   void(*remotealm)(const char*alname, const char* almsg, int* err))
```

Arguments

`argc`	>>	number of command-line arguments
`*argv[]`	>>	Command line arguments
`*toremote`	>>	function to export data values
`*remotealm`	>>	function to export alarm messages

Usage Considerations

`transexport()` returns zero (0) if transexport times out. Otherwise, a non-zero value is returned, indicating an error.

Your program calls `transexport()` to parse the INEXPT file and periodically call `toremote()` and `remotealm()`. `Transexport()` does not return until a time-out or error occurs.

Previous to version 9.4, SPS supplied a transexport `main()` program that contained this functionality. For increased flexibility, you may now provide your own main routine. You can get old behavior with the following main routine:

```
#include "saiInterface.h" /* transexport */
void main(int argc, char const*const*argv)
{
  transexport(argc, argv, toremote, remotealm);
}
```

For each EXPORTALARM configured in the INEXPT file, `transexport()` does the following:

1. Maintains the old status and export message
2. Gets the current status and alarm message from the simulation
3. If the status or message has changed, calls `remotealm()`, unless the message is either zero length or all blanks.

`transexport()` calls `toremote()` periodically with all EXPORTDATA simulation variables that are configured in the INEXPT file.

The transexport `main()` ignores the return value of `*ier`.

See Also

toremotealm(), toremote()

"Sending simulation results from the Statefinder/Leakfinder model to the SCADA system" on page 927

"Statefinder and Leakfinder" on page 741

Library: spsclient stclib

Licensing: DRINIT TRANSEXPORT

remotealm

This is the alarm export routine. See "Statefinder and Leakfinder" on page 741 for more information.

Declaration

```
void remotealm(const char *alname, const char *almsg, int *ier)
```

Arguments

`*alname`	>>	alarm name string
`*almsg`	>>	alarm message string
`*ier`	<<	error value, as returned by `remotealm()`. If an error occurs, a non-zero value is returned.

Usage Considerations

You should use a version of this routine to export simulation alarms. It can perform a variety of functions, including:

- Print a report
- Load alarms into a database
- Pass alarms to another application

GL provides a version of this routine in the sample program for the transexport interface (transexporttest). The GL version simply echoes the input parameters to `stdout`.

`transexport()` calls this routine periodically to export any alarm that has changed status or message text.

For each EXPORTALARM configured in the INEXPT file, `transexport()` does the following:

1. Maintains the old status and export message
2. Gets the current status and alarm message from the simulation
3. If the status or message has changed, calls `remotealm()`, unless the message is either zero length or all blanks.

`transexport()` ignores the return value of `*ier`.

See Also

toremote() on page 996

"Sending simulation results from the Statefinder/Leakfinder model to the SCADA system" on page 927

"Statefinder and Leakfinder" on page 741

Library: spsclient stclib

Licensing: DRINIT TRANSEXPORT

toremote

This is the data export routine. See "Statefinder and Leakfinder" on page 741 for more information.

Declaration

```
void toremote(const char **pvname, const double dval[], const int n, int
*ier)
```

Arguments

`**pvname`	>>	array of pointers to variable names
`dval[]`	>>	array of model variable values
`n`	>>	number of data points
`*ier`	<<	error value, as returned by `toremote()`. If an error occurs, a non-zero value is returned.

Usage Considerations

You should use a version of this routine to export simulation data. It can perform a variety of functions, including:

- Print a report
- Load data into a database
- Pass data to another application

GL provides a version of this routine in the sample program for the transexport interface (transexporttest). The GL version simply echoes the input parameters to stdout.

`transexport()` calls this routine periodically with all EXPORTDATA simulation variables that are configured in the INEXPT file.

`transexport()` ignores the return value of `*ier`.

See Also

remotealm() on page 995

"Sending simulation results from the Statefinder/Leakfinder model to the SCADA system" on page 927

"Statefinder and Leakfinder" on page 741

Library: spsclient stclib

Licensing: DRINIT TRANSEXPORT

Utility routines

The following routines do not directly communicate with SPS, but rather they make it easier to use some of the other interfaces, in particular dr* and todrem.

spsSleep

Provides a system independent function to suspend execution for a specified time interval.

Declaration

```
void spsSleep(int wait, int *ier)
```

Arguments

`wait`	>>	time to wait, in milliseconds
`*ier`	<<	error value, as returned by `spsSleep()`. If no error occurs, zero is returned.

Usage Considerations

You may call the system function on your operating system to provide the same functionality.

See Also

n/a

Library: dreml

Licensing: n/a

Servers

Various server technologies extend the capabilities of SPS and allow for communication and interaction of SPS data with other applications. Your system may include one or more of the following servers.

Server	File name	Type	Description
"TRANSSHARE program" on page 997	Transshare.exe	RPC	Remote Procedure Call (RPC) server that allows for communication between TRANS and dr* clients over a computer network.
"WinCip" on page 1001	CIPLib.dll	TCP/IP	TCP/IP server that allows SCADA to communicate with a running SPS model using RTU protocols. WinCip includes a user interface, Wincip.exe.

Each server must be installed and configured to allow communication between a running SPS model and another server or application. Some components are installed and configured through a standard SPS installation while others may require custom installation and setup.

TRANSSHARE program

This section contains a description of the TRANSSHARE command and procedures for setting up the portmap required to run TRANSSHARE.

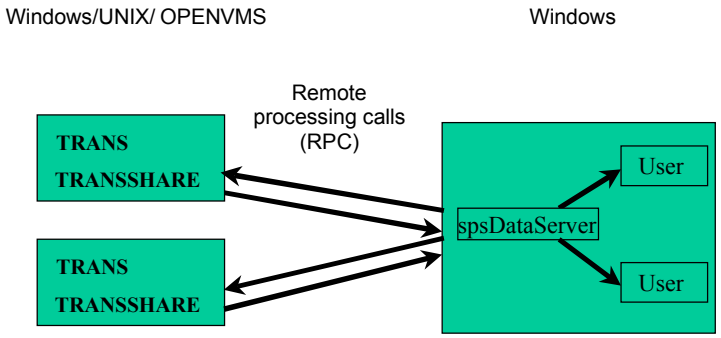
TRANSSHARE

```
TRANSSHARE [-r programnumber -z rname -x] {-c model | model}
```

Field	Meaning
programnumber	Specifies a specific RPC program number, only if needed. TRANSSHARE computes a program number that will be used under most circumstances and does not typically require specification. When choosing a program number, be careful to avoid conflicts with other software.
rname	Specifies that the server will register itself with the lexicon, using the specified name (rname). This name must be placed in Schematic's Control Manager to connect properly.
-x	Specifies that the server will register itself with the lexicon. If used without the -z option, the registration will use the default name. The default name will conform to the following format: `/st/machine_name/current_dir_name/model_name` The above name must be placed in Schematic's control manager to connect properly.
model	Specifies the model running in TRANS to which TRANSHARE should be connected. {bc}. If used without "-c", this must be the last entry on the command line.

Description

TRANSSHARE is a program that permits TRANS to communicate with dr* clients over a computer network. In some instances, TRANS and TRANSSHARE will be run on one computer, while dr* clients will run on another computer. However, they can all be run on the same computer if desired. TRANSSHARE can be run from a command line or through the BACKGROUND mode.



Take note

- Whenever a dr* client is started on a different computer than the one running TRANSSHARE, the file created by TRANSSHARE, *model*.SHARE, must first be copied to the computer running the dr* client.

- Before TRANSSHARE can be executed on a model, the TRANS process associated with the model must have taken at least one (1) step. Two ways of ensuring that TRANS is executing before starting TRANSSHARE are:
- Wait until the *model*.vydef file has been created, then start TRANSSHARE.
- Add instructions to start TRANSSHARE using the background command within the INTRAN file.

Example input

The following example executes TRANSSHARE and connects to the TRANS process that is running with a model name of “predict”:

```
transshare predict
```

The following example executes TRANSSHARE and connects to the TRANS process that is running with a model name of predict and forces the rpc socket number to be 123456789:

```
transshare predict -r 123456789
```

The following example for starting TRANSSHARE on a TRANS process that is running with a model name of predict is located within the predict.intran file. This example uses the DO.INTERACTIVE and BACKGROUND commands:

```
DO.INTERACTIVE "BACKGROUND transshare predict"
```

Note: TRANSSHARE creates a file called model.share in the directory from which it is executed. The contents of the file will look like the following:

```
laforge 1073759520      1       tcp
```

where:

laforge	=	host name of the computer that is executing TRANSSHARE
107375920	=	230 + PID of the executing TRANS for model
1	=	Current revision of the TRANSSHARE executable
tcp	=	Communications protocol over the network

Portmap

To run successfully, TRANSSHARE requires the portmap service to run simultaneously. By default, the port number changes for each TRANSSHARE run, even for the same model.

Usually, you should set the port number for TRANSSHARE for a given model by using the TRANSSHARE command line when the process is invoked. However, this depends on your operating system.

- On Windows, SPS provides two executables in the ...\sps\bin directory called portmap.exe. Portmap.exe is the portmap service program. The portmap service program must run continuously as long as TRANSSHARE runs. You must also install the portmap.srg file at the same time.

Note: Avoid using the same port number for multiple TRANSSHARE processes at the same host machine

To install portmap permanently on Windows

1 Select **Start > Run**.

2 In the **Run** dialog box, type:

```
portmap -registerservice
```

3 Select **Start > Settings > Control Panel**.

4 Double-click **Administrative Tools**.

5 Double-click **Services**.

6 Locate and select **portmap**.

7 From the main menu in the Services dialog box, select **Actions > Start**.

8 Close the **Services** dialog box.

To uninstall portmap on Windows

1 Select **Start > Run**.

2 In the **Run** dialog box, type:

```
portmap -unregisterservice
```

3 Click **OK**.

TRANSSHARE as a CORBA replacement

Prior to SPS 7.0, both CORBA and TRANSSHARE were available for use as SPS servers. In SPS 7.0, the CORBA server was removed. The following table is provided as a reference for CORBA users who need to update to TRANSSHARE.

Descripiton	CORBA Version	Lexicon Replacement
Environment variable	CORBA_NAMESERVICE= host:1525	SPS_LX_HOST=lx:host:10403
Registry Entry	HKEY_LOCAL_MACHINE SOFTWARE ACE TAO TaoNamingServiceOptions = -ORBEndPoint iiop://host:1525	None
Directory Service	TAO NT Naming Service	SPS Network Lexicon
Model server	StSpsDrServer -casename cname -registerName rname	Transshare -c cname -z rname
Model server	StSpsDrServer -casename cname	Transshare -c cname -x
Network Tport	Tport spscorba::rname	Tport lx::rname
Network Tport with copy of cname.share in local directory	Not supported	Tport share::cname
SpsServer2 Attach() argument	“rname”	“rname”
SpsServer2 Attach() argument	“spscorba::rname”	“spslx::rname” or just “rname”
List contents of Directory Service	Various custom programs	Lxutil list

WinCip

The WinCip program is used to connect a running SPS model to a SCADA system using a TCP/IP connection. The SPS model might be a Statefinder/Leakfinder model or a training model. The WinCIP program makes the model appear to be a set of RTUs attached to one or more TCP/IP ports.

This section describes:

- WinCip Interfaces
- Running WinCip
- Using the WinCip user interface
- Notes for INXREF authors
- Protocol descriptions

WinCip interfaces

The following SCADA interfaces allow SPS/Statefinder and SPS/Trainer to communicate with SCADA systems:

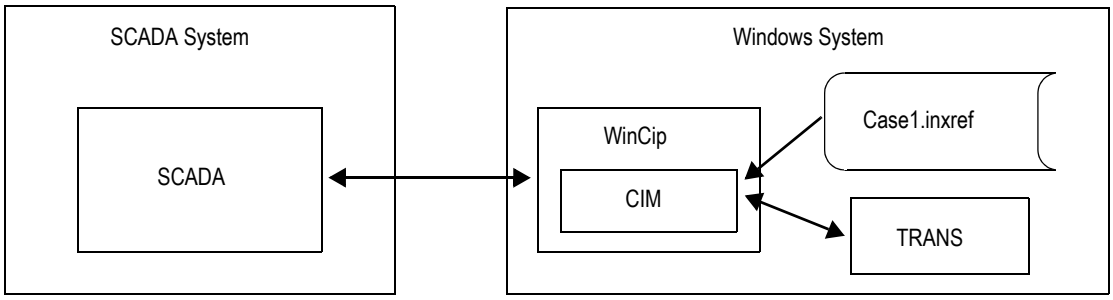
- SPS/Statefinder and Predictor to SCADA
- SCADA/Trainer Interface

Both interfaces use the same interface program, WinCip, but with different configuration files.

SPS/Statefinder and Predictor to SCADA

The SPS/Statefinder/Predictor to SCADA program, WinCip, is built around the CIM interface to SPS/Statefinder/Predictor (TRANS). The following diagram shows how WinCip communicates with the rest of the system.

SPS to SCADA using WinCip



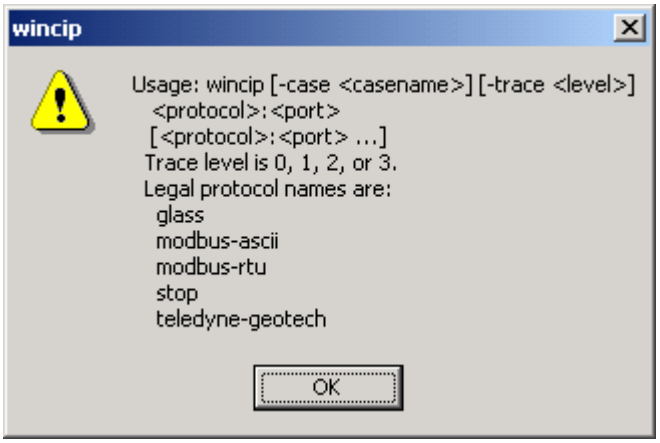
The SCADA system in this diagram is the primary SCADA system, although it could also be a backup system operating as primary due to fail over. The Windows system is the primary simulation system, the one where the SPS/Statefinder/Predictor (TRANS) is running. SCADA can be configured to communicate with an RTU server running on a TCP/IP socket on a remote system. WinCip reads an input file, case1.inxref in the diagram above, which is maintained in conjunction with the model. The input file describes the messages sent from SCADA to WinCip and the associated actions and replies.

SCADA/Trainer interface

The operating environment for the SCADA/Trainer interface is very similar to the environment for the SPS/Statefinder/Predictor to SCADA interface. The same interface program, WinCip, is used, except with a different INXREF file.

Running WinCIP

If you run WinCIP with no command line arguments, or give it invalid command line arguments, you will receive a message box that shows the command line arguments that may be used. Here is an example:



On your system, the list of legal protocol names may be different. This section contains a description of the use of these command line arguments.

Protocol and port for WinCIP

The only required command line argument is a <protocol>:<port> argument that tells WinCIP what TCP/IP port to use and what RTU protocol will be expected on that port. An example command line is:

```
wincip teledyne-geotech:1000
```

This will start WinCIP using TCP/IP port 1000 and tell it to use the Teledyne-Geotech RTU protocol. You may include several <protocol>:<port> arguments to have a single running WinCIP program support communications on multiple ports. If you do this, you must be sure that each port number used in the argument list is unique. You can use the same protocol on more than one port, if you wish. Here is another example:

```
wincip teledyne-geotech:1000 modbus-ascii:1001 modbus-ascii:1002
```

This starts WinCIP using the Teledyne-Geotech protocol on port 1000 and the Modbus ASCII protocol on ports 1001 and 1002.

Legal ports are numbers between 1 and 65535, but usually low numbered ports are already in use for well-known TCP/IP services. *If you select a port that is already in use on your system, WinCIP will not be able to bind to that port.*

Case name for WinCIP

You may use the “-case” command line argument to specify which running SPS case WinCIP should try to connect to. Each running WinCIP program can only connect to a single case. You may, however, run several WinCIP programs at the same time, each connecting to a different case.

If you don’t supply the “-case” argument, the environment variable DREMCASENAME will be used to determine the case that should be used.

Here is an example of starting WinCIP using the Modbus RTU protocol on port 5000 and connecting to case “basecase”:

```
wincip –case basecase modbus-rtu:5000
```

WinCIP will look for files named <casename>.vydef and <casename>.inxref in the current directory. You will probably want to create a shortcut that starts WinCIP with the arguments you want and sets the “Start in” directory to the directory containing the data files for your case.

Since WinCIP will try to transfer data from a running SPS model with the specified case name, you should be sure to start SPS running on this case before you start WinCIP. SPS must also be sharing the information that the <casename>.inxref file tells WinCIP to retrieve from the running model. To share information, make sure you have a “SHARE” directive in the <casename>.intran file. Since “SHARE” also requires “TRENDLIST” to be effective, you may need to add it also. Here is a portion of a <casename>.intran file illustrating how these directives might be used:

```
=SHARE *
=TRENDLIST *, PEEK.MATCH = *:P* *:Q*
```

These two directives tell SPS to share all peeks that match *:P* or *:Q*.

WinCIP will try to connect to the model and parse the <casename>.inxref file when the first incoming packet arrives. If WinCIP is started with the wrong working directory, or if SPS is not running on the model, you will receive an error message in the log and WinCIP will be shut down. Likewise, if there is an error parsing the <casename>.inxref file. To recover from either of these errors, WinCIP must be restarted after the problem is corrected.

You will probably want to start WinCIP each time you run SPS. You can do this by adding the following lines to your <casename>.intran file.

```
DEFINE.SEQUENCE BO { DO.INTERACTIVE "SPAWN runwincip.bat" }
SUBMIT.SEQUENCE BO
```

When SPS processes these lines in the <casename>.intran file, it will start a new subprocess to run the runwincip.bat batch file. Since SPS will wait for this process to finish before continuing, the batch file needs to start WinCIP and return right away. You can do this using the command processor “start” command. Here is an example runwincip.bat:

```
set DREMCASENAME=casename
start /min wincip -trace 1 teledyne-geotech:1000
```

Tracing and Logging for WinCIP

Diagnostic tracing is turned off by default. You can start WinCIP with tracing turned on by using the “-trace” command line argument. You can also adjust the tracing level using the user interface after WinCIP is started. There are four

different tracing levels: 0, 1, 2, and 3. Higher levels include more detail. Here is a table showing the tracing messages that you get with each level.

Trace Level	Trace Log Messages
0	None (Tracing off) [default]
1	Client connect and disconnect
2	All level 1 trace messages Trace of all data going in and out on the ports
3	All level 2 trace messages Trace of all data going to and from the simulator

Here is an example of a command line that starts WinCIP on case “bc” using modbus-rtu protocol on port 5001 with trace level 2:

```
wincip –case bc –trace 2 modbus-rtu:5001
```

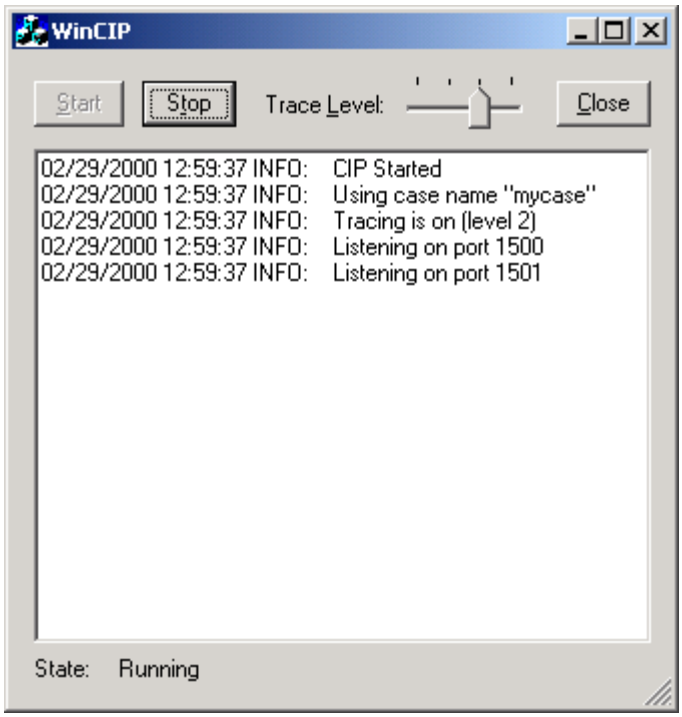
This will result in trace messages for client connect and disconnect and a trace of all data going in and out of port 5001.

There are other types of logging messages that WinCIP produces: informational messages, warnings, and errors. These messages are always displayed, regardless of the setting of the trace level.

Log messages all include the date, time, and type of message (INFO, WARNING, ERROR, TRACE). In addition to being displayed in the user interface, WinCIP appends these messages to the end of a file named “cip.log” in the installation directory. WinCIP never deletes or clears this log file, so you should arrange to archive and delete this log file periodically. WinCIP only opens the log file long enough to append each message before closing the file again.

Using the WinCip user interface

Once WinCIP is running, its user interface displays all log messages and allows the trace level to be changed. You may also use the interface to start or stop the communication between the running simulation and the SCADA system.



The WinCip user interface

When WinCIP is started, it should be in the running state. If you press the Stop button, all current TCP/IP connections will be closed and new connections will not be accepted.

When WinCIP is in the stopped state, pressing the Start button will cause WinCIP to begin accepting connections on all the ports specified on the command line again.

If the simulator is stopped for any reason, the running WinCIP will not be able to communicate with the simulator, even if it is restarted on the same case. To reestablish communications, close WinCIP and restart it.

The Trace Level slider may be used at any time to change the trace level. The leftmost position is level 0 and the right-most is position 3. See "Tracing and Logging for WinCIP" on page 1003 for a description of the messages displayed at the different trace levels.

Clicking and dragging the gripper in the lower right corner will resize the user interface window. The log window will scroll when necessary. If the last line of the text is visible, WinCIP will automatically scroll the text as each new log message is added so that the last message is always visible.

The number of log messages that may be displayed in the log window is limited by the operating system. When the limit is reached, lines from the beginning of the log are removed to make room for the new log messages. This has no effect on the log file, which will continue to grow until it is deleted.

Notes for INXREF file authors

The messages that WinCIP returns to the SCADA system are determined by the rules given in a file named <casename>.inxref. This file contains instructions for constructing replies based on the commands that come from the SCADA system. The syntax and rules for constructing the .inxref file are documented elsewhere, but this section describes how WinCIP prepares packets that come from the SCADA system before it hands them down to the .inxref processor and how it prepares replies from the .inxref processor before sending them back out on a TCP/IP port.

In general, one WinCIP program may be communicating on several TCP/IP ports, but the .inxref file used depends only on the case name. To make it possible to distinguish these connections in the .inxref file, WinCIP will generally prepend the port number (low byte then high byte) to the packet before sending it to the .inxref processor. This behavior depends on the protocol used.

The “glass” and “stop” protocols are used for testing and these do not prepend the port number. The “glass” protocol sends and receives all bytes without any additions or deletions. The “stop” protocol shuts down the connection (like pressing “Stop”) when the first packet arrives.

Some protocols have an “envelope” which may consist of standard header bytes and/or trailer bytes and possibly a checksum or CRC used to verify that the packet arrived intact. The WinCIP protocols will generally remove any envelope characters before sending the packet down to the .inxref file processor. The WinCIP protocols generally expect to receive a reply from the .inxref file processor that does not include the envelope characters. The protocol processor in WinCIP will add back the envelope before transmitting the packet on the TCP/IP port. In particular, WinCIP protocols are responsible for checking and generating any required checksum or CRC bytes.

WinCip protocol descriptions

This section describes in detail how WinCIP supports each of its protocols. This includes information about whether the port number is passed down to the .inxref processor, what “envelope” is removed before sending the packet on to the .inxref processor and added back before sending the reply on the TCP/IP port, and how any checksum or CRC is calculated.

Modbus RTU protocol

Modbus RTU is a binary protocol that has packets that include two CRC bytes. In general, a packet looks like this:

[Addr] [Func] [Data Count] [Data] [Data] [Data] [Data] [CRC low byte] [CRC high byte]

When WinCIP receives a packet, it prepends the port number and strips the CRC bytes before handing it down to the .inxref processor. The CRC is not computed and the CRC bytes are not checked. The .inxref processor receives a packet that looks like this:

[Port low byte] [Port high byte] [Addr] [Func] [Data Count] [Data] [Data] [Data] [Data]

WinCIP expects the .inxref processor to return a packet without the port number or CRC. The packet should look like this:

[Addr] [Func] [Data Count] [Data] [Data] [Data] [Data]

WinCIP will then compute the CRC and add it to the end of the packet before transmitting it out on the TCP/IP port. Here is the C++ code used to calculate and append the CRC bytes:

```
protocolInterface::protocol_status protocolModbusRTU::PostCim(
  const vector<char>& replyin, vector<char>& replyout) const
{
  unsigned short crc = 0xFFFF;
  const unsigned short magic = 0xA001;
  int lsb;

  replyout.resize(0);
  for (int index = 0; index < replyin.size(); index++)
```

```
    {
      replyout.push_back(replyin[index]);
      crc ^= (replyin[index] & 0xFF);
      for (int shift = 0; shift < 8; shift++)
      {
        lsb = crc & 1;
        crc >>= 1; // crc is unsigned so high bit is zero-filled.
        if (lsb)
          crc ^= magic;
      }
    }

    replyout.push_back(crc & 0xFF);
    replyout.push_back((crc >> 8) & 0xFF);

    return protocolInterface::PROTOCOL_STATUS_OK;
  }
```

Modbus ASCII protocol

Modbus ASCII is an ASCII character protocol that includes a leading colon character, a trailing carriage return/line feed (CR/LF) combination, and a one byte “Longitudinal Redundancy Check” (LRC). The LRC is inserted just before the trailing carriage return/line feed. The LRC is transmitted as two ASCII hex digits. Each byte between the colon character and the LRC is also transmitted as two ASCII hex digits. Here is an example packet:

[Colon] [Addr] [Func] [Data Count] [Data] [Data] [Data] [Data] [LRC] [CR] [LF]

When WinCIP receives a packet it prepends the port number. Then it strips the colon character, the LRC bytes, and the CR/LF before sending the packet down to the .inxref processor. It also converts each packet byte from two ASCII hex digits into one byte. The LRC is not computed or checked. Here is a Modbus ASCII packet as it will be sent to the .inxref processor:

[Port low byte] [Port high byte] [Addr] [Func] [Data Count] [Data] [Data] [Data] [Data]

WinCIP expects to receive a packet from the .inxref processor that does not include the port number, colon, CR/LF, or LRC bytes. Here is an example:

[Addr] [Func] [Data Count] [Data] [Data] [Data] [Data]

WinCIP will then prepend the colon, compute the LRC and append it and the CR/LF before sending the packet back out on the TCP/IP port. Each of the data bytes is converted into two ASCII hex digits. Here is the code used for computing the LRC and assembling the outgoing packet:

```
protocolInterface::protocol_status protocolModbusASCII::PostCim(
  const vector<char>& replyin, vector<char>& replyout) const
{
  const char hex_digits[] = "0123456789ABCDEF";
  unsigned char lrc = 0;

  replyout.resize(0);
  replyout.push_back(':');  // add back the protocol "envelope" header

  unsigned char reply_byte;
```

```
  for (int index = 0; index < replyin.size(); index++)
  {
    reply_byte = (unsigned char)(replyin[index]);
    lrc += reply_byte;

    // reply_byte is unsigned, high bits are zero filled in the shift
    replyout.push_back(hex_digits[reply_byte >> 4]);
    replyout.push_back(hex_digits[reply_byte & 0x0F]);
  }

  // Now take the twos complement
  lrc = (unsigned char)(-(char)lrc);

  // convert to two hex digits and append, high byte first
  // lrc is unsigned, so high bits are zero filled in the shift right
  replyout.push_back(hex_digits[lrc >> 4]);
  replyout.push_back(hex_digits[lrc & 0x0F]);

  // add back the protocol "envelope" trailer
  replyout.push_back('\r');
  replyout.push_back('\n');

  return protocolInterface::PROTOCOL_STATUS_OK;
}
```

Teledyne-Geotech protocol

The Teledyne-Geotech protocol consists entirely of 8 byte packets. Each packet has a one byte CRC as the final byte. We expect all requests coming to WinCIP to be one packet long. Here is an example request packet:

[Byte 1] [Byte 2] [Byte 3] [Byte 4] [Byte 5] [Byte 6] [Byte 7] [CRC]

WinCIP prepends the port number and removes the CRC byte before passing the packet down to the .inxref processor. The CRC is not computed or checked. Here is a packet as it will be passed down to the .inxref processor:

[Port low byte] [Port high byte] [Byte 1] [Byte 2] [Byte 3] [Byte 4] [Byte 5] [Byte 6] [Byte 7]

Replies may be one or more packets long. WinCIP expects the .inxref processor to return the data for all of the reply packets with no CRC bytes included. The number of bytes returned should be some multiple of seven. Here is a two packet reply as it would be sent back by the .inxref file processor:

[Byte 1] [Byte 2] [Byte 3] [Byte 4] [Byte 5] [Byte 6] [Byte 7] [Byte 8] [Byte 9]

[Byte 10] [Byte 11] [Byte 12] [Byte 13] [Byte 14]

WinCIP computes the CRC for each seven bytes and inserts it immediately after these bytes. All the Teledyne-Geotech packets in a single reply are sent out on the TCP/IP port in one TCP/IP packet. Here is the TCP/IP packet that would be returned for the two packet reply above:

[Byte 1] [Byte 2] [Byte 3] [Byte 4] [Byte 5] [Byte 6] [Byte 7] [CRC for bytes 1 to 7] [Byte 8] [Byte 9] [Byte 10] [Byte 11] [Byte 12] [Byte 13] [Byte 14]

[CRC for bytes 8 to 14]

Here is the code for calculating and inserting the CRC bytes:

```
protocolInterface::protocol_status protocolTeledyneGeotech::PostCim(
  const vector<char>& replyin, vector<char>& replyout) const
{
  protocolInterface::protocol_status return_status =
protocolInterface::PROTOCOL_STATUS_OK;

  if (replyin.size() == 0 || replyin.size() % 7 != 0)
    return_status = protocolInterface::PROTOCOL_STATUS_WARNING;

  replyout.resize(0);

  unsigned char crc = 0;
  unsigned char carry;
  unsigned char magic = 0x07;

  int index = 0;
  while (index < replyin.size())
  {
    unsigned char byte = replyin[index];
    replyout.push_back(byte);
    for (int shift = 0; shift < 8; shift++)
    {
      crc ^= (byte & 0x80);
      carry = crc & 0x80;
      crc <<= 1;
      if (carry)
       crc ^= magic;
      byte <<= 1;
    }
    index++;
    if (index % 7 == 0)
    {
      replyout.push_back(~crc);  // ones complement, for remote unit
      crc = 0;// start over with a new crc value for next packet
    }
  }

  return return_status;
}
```

Glass protocol

The glass protocol passes all incoming packet bytes down to the .inxref processor exactly as they are received. The port number is not added. The glass protocol passes all bytes received from the .inxref processor directly out on the TCP/IP port.

Although the glass protocol is intended primarily for debugging, it will be useful in cases where the .inxref processor is sufficient to do all the work of producing reply packets based on incoming request packets. In general this requires that there is either no CRC or checksum or that the SCADA system pays no attention to any CRC or checksum that is present.

Since the port number is not prepended to the packet before it is sent to the .inxref processor, the .inxref file processor will not be able to distinguish requests coming in on different ports. Generally this will mean that a single running WinCIP

program can only service the glass protocol on a single port, and that no other protocol (except Stop) can be used along with glass.

Stop protocol

The stop protocol exists solely to make WinCIP more testable. The first packet that arrives causes WinCIP to stop processing packets and stop accepting connections on all TCP/IP ports. The data of the packet is ignored and no reply packet is ever sent. The WinCIP application then shuts down.

Function prototypes

This section provides an alphabetical quick reference of functions to the various interfaces.

- rtuOpen() on page 961
- rtuReadNextPt() on page 962
- rtuSeek() on page 963
- siflsh() on page 987
- spsSleep() on page 997
- toremote() on page 996

API Examples

The following examples have been provided for your reference. You may also find these examples in the source files in the SPS custsrc installation directory.

cimtest

```
/* cimtest.c */
#include <stdio.h>;/* fprintf, sprintf, etc. */
#include "saiInterface.h" /* GL cim function */
/* Purpose
 * -------
 * Sample program for SPS' trainer interface.
 *
 * Description
 * -----------
 * Loop through various patterns of bytes and report
 * the cim replies.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
*/
static void getit(const unsigned char*, const int*, unsigned char*, int*);
   int main()
{
   unsigned char in[20], out[2000];
   int ilen, olen;
   int c1, c2;
   /* backward loop over some values */
   ilen = 2;
   for (c2 = 127; c2 > 0; c2 -=7)
   {
      in[1] = c2;
      for (c1 = 7; c1 > 0; c1--)
      {
         in[0] = c1;
         getit(in, &ilen, out, &olen);
      }
```

```
}
/* test an optional argument */
ilen = 3;
in[0] = 3;
in[1] = 5;
in[2] = 7;
getit(in, &ilen, out, &olen);
/* forward loop over some values */
ilen = 2;
for (c2 = 0; c2 < 128; c2 += 7)
{
   in[1] = c2;
   for (c1 = 0; c1 < 8; c1++)
   {
      &#9; in[0] = c1;
      &#9; getit(in, &ilen, out, &olen);
   }
}
/* do some string operations */
ilen = 2;
in[0] = 0xd0;
in[1] = 'Y';
getit(in, &ilen, out, &olen);
fprintf(stdout,"String : %7s-\n", &out[2]);
ilen = 8;
in[0] = 0xd1;
sprintf((char*)&in[1], "%7s", " OPEN");
getit(in, &ilen, out, &olen);
fprintf(stdout,"String : %16s-\n", &out[2]);
ilen = 17;
in[0] = 0xd2;
sprintf((char*)&in[1], "%16s", "8.187");
getit(in, &ilen, out, &olen);
fprintf(stdout,"String : %12s-\n", &out[2]);
ilen = 2;
in[0] = 0xd0;
in[1] = 'N';
getit(in, &ilen, out, &olen);
ilen = 8;
in[0] = 0xd1;
sprintf((char*)&in[1], "%7s", " CLOSE");
getit(in, &ilen, out, &olen);
fprintf(stdout,"String : %12s-\n", &out[2]);
ilen = 17;
in[0] = 0xd2;
sprintf((char*)&in[1], "%16s", "9.187");
getit(in, &ilen, out, &olen);
fprintf(stdout,"String : %12s-\n", &out[2]);
/* forward loop over some values */
ilen = 2;
```

```
   for (c2 = 0; c2 < 128; c2 += 7)
   {
      in[1] = c2;
      for (c1 = 0; c1 < 8; c1++)
      {
         &#9; in[0] = c1;
         &#9; getit(in,&ilen,out,&olen);
      }
   }
   /* send the magic quit command */
   ilen = 3;
   in[0] = 9;
   in[1] = 9;
   in[2] = 9;
   getit(in,&ilen,out,&olen);
   return 0;
}
/* call cim and print the input and output bytes
 * in ---> command to send to cim
 * ilen ---> length of in
 * out <--- reply returned by cim
 * olen <--- length of out
*/
static void getit(const unsigned char* in, const int* ilen, unsigned char*
out, int* olen)
{
   int k;
   int ier = 0;
   fprintf(stdout,"Input : ");
   for (k = 0; k < *ilen; ++k)
   fprintf(stdout, "%d ", in[k]);
   fprintf(stdout,"\n");
   cim((char*)in, ilen, (char*)out, olen, &ier);
   if ( ier )
   {
      fprintf (stdout,"Ier is %d\n", ier);
   }
   else
   {
      fprintf(stdout,"Output : ");
      for (k = 0; k < *olen; ++k)
      fprintf(stdout, "%d ", out[k]);
      fprintf(stdout,"\n");
   }
}
```

drtest

```
/* drtest.c
```

```
 *
 * Purpose
 * -------
 * Sample program for SPS's "Data Retrieval" interface.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
 * Gotchas
 * -------
 * This code references some devices in the test model provided by
 * GL (drtest). This code needs to be modified accordingly when
 * the model is changed.
 *
 * This test program must be invoked in the same directory where the
 * simulation is running.
 *
 */
#include <stdio.h> /* printf, sprintf */
#include <stdlib.h> /* exit */
#include <time.h> /* time */
#include "saiInterface.h" /* GL drxxxx/cdrxxx functions */
static void passed(char* name, int ier);
static void sai_sync(char* name);
#define ARRAYSIZE 4 /* dimension of the array below */
char* ARRAYPEEK[] = {"BV1:P+", "BV1:ST", "TIME", "DT"};
char* PATH[] = {"dp1", "out1", 0};
/* difference between UNIX GMT time and DREM time */
const int TIMEDIFF = (366+365+1)*24*60;
   int main(int argc, char** argv)
{
     int ier = 0;
     int numtries = 0;
     char* casename;
     char* peekname = "TIME";
     char* pokename = "MAXPRES";
     int addr;
     char name[81];
     char str[41];
     int name_length;
     float value;
     float newval;
     int points, time_points;
     drPointXY *curvedata, *the_point;
     double *x, *ta;
     int flag[ARRAYSIZE];
     int address[ARRAYSIZE];
     double val[ARRAYSIZE];
     char strval[ARRAYSIZE][80];
     const int array_size = ARRAYSIZE;
```

```
time_t timevar;
double t;
int i, j;
char** peeks;
switch(argc)
{
  case 1:
    casename = "bc";
    break;
  case 2:
    casename = argv[1];
    break;
  default:
    printf("Usage: %s casename\n", argv[0]);
    exit(-1);
}
/* get the connection
cdrini should only be called once in the program */
cdrini(casename, &ier);
/* ier = 0 means connected to a running model
ier = 2 means connected to review file only, no shared
memory communication (usually means model is not running) */
passed("cdrini", ier);

/* retrieve information associated with a string-valued peekable */
/* first need to get the "getval addr" for the peekable */
cdrgta(peekname, &addr, &ier);
passed("1st cdrgta", ier);

/* retrieve the current value for the string-valued peekable */
name_length = sizeof(str);
cdrgts(&addr, str, &ier, &name_length);
passed("1st cdrgts", ier);
printf("String value for %s is %s\n", peekname, str);
/* get the name associated with the "getval addr"
look redundant here, well, just for a demonstration */
name_length = sizeof(name);
cdrgtn(&addr, name, &ier, &name_length);
passed("cdrgtn", ier);
printf("cdrgtn: expected %s, got %s\n", peekname, name);
/* send a command string to the simulation. you can put
 * any one of the valid interactive commands in the first
 * argument */
cdrptc("run for 1 step", &ier);
/* ier = 0 means the command has been successfully sent to
 * the simulation, it does not mean the command has been
 * successfully executed in the simulation */
 passed("1st cdrptc", ier);
/* for some commands, you can check whether or not the command
 * has been executed successfully in the simulation. let us
```

```
 * try it for the above command */
sai_sync("first"); /* get local memory updated */
/* retrieve again the current value for the string-valued peekable */
name_length = sizeof(str);
cdrgts(&addr, str, &ier, &name_length);
passed("2st cdrgts", ier);
printf("String value for %s is %s\n", peekname, str);
/* should always call drfrea to free the memory allocated for the
 * "getval addr" after it is no longer needed */
drfrea(&addr, &ier);
passed("1st drfrea", ier);

/* retrieve information associated with a scalar-valued peekable */
/* get the "getval addr" for the peekable */
cdrgta(pokename, &addr, &ier);
passed("2nd cdrgta", ier);

/* retrieve the current value */
drgetv(&addr, &value, &ier);
passed("1st drgetv", ier);
printf("Analog value for %s is %.2f\n", pokename, value);
/* change the value of a pokable in the simulation */
newval = value * 0.9f;
printf("Set %s to %.2f\n", pokename, newval);
drputv(&addr, &newval, &ier);
passed("drputv", ier);

/* the above change will take place only after the simulation
 * takes a step, so let us send another command to the simulation */
cdrptc("run for 1 step", &ier);
passed("2rd cdrptc", ier);
sai_sync("second");

/* check if the change has been made */
drgetv(&addr, &value, &ier);
passed("2nd drgetv", ier);
printf("Analog value for %s is %.2f\n", pokename, value);
/* should always call drfrea to free the memory allocated for the
 * "getval addr" after it is no longer needed */
drfrea(&addr, &ier);
passed("1st drfrea", ier);
/* get distance profile data from the simulation */

/* distance plot data is not in the shared memory when first
 * requested. it will take one or more steps for TRANS to honor
 * the request */
numtries = 0;
do {
  cdrptc("run for 1 step", &ier);
  sai_sync("drGetDistPlot");
```

```
    drGetDistPlot(PATH, "temperature", &points, &curvedata, &ier);
  } while ( ier && ++numtries < 10);
  the_point = curvedata;
  for ( i = 0; i < points; i++ )
  {
    printf("distance = %3.0f, temperature = %3.0f\n", curvedata->x,
       curvedata->y);
    curvedata++;
  }
  drFree(the_point, &ier);
  /* let the simulation run for a few steps */
  cdrgta(peekname, &addr, &ier);
  passed("third cdrgta", ier);
  do {
    cdrptc("run for 1 step", &ier);
    sai_sync("get time loop");
    drgetv(&addr, &value, &ier);
  } while ( value < 10.0 );
  drfrea(&addr, &ier);
  /* get trend data from the simulation */
  printf("current time is %4.1f\n", value);
  /* get the trend data from the beginning of simulation upto current */
  drGetTimePlot(0.0, 10.0, "PIPE1:P-", &points, &curvedata, &ier);
  passed("drGetTimePlot", ier);
  the_point = curvedata;
  printf("points = %d\n", points);
  for ( i = 0; i < points; i++ )
  {
    printf("time = %4.1f, PIPE1:P- = %3.0f\n", curvedata->x, curvedata-
>y);
    curvedata++;
  }
  drFree(the_point, &ier);
  /* get historical profile data from the simulation */
  /* get all available time */
  drGetDistPlotTimes(&time_points, &x, &ier);
  passed("drGetDistPlotTimes", ier);
  ta = x;
  for ( i = 0; i < time_points; i++ )
  {
    printf("Request time = %3.1f\n", *ta);
    'drGetDistPlotHist(PATH, "elevation", *ta, &points, &curvedata, &t,
        &ier);
    passed("drGetDistPlotHist", ier);
    the_point = curvedata;
    printf("Actual time = %3.1f\n", t);
    for ( j = 0; j < points; j++ )
    {
      printf("distance = %3.0f, elevation = %3.0f\n",
        &#9;curvedata->x, curvedata->y);
```

```
      curvedata++;
    }
    drFree(the_point, &ier);
    ta++;
  }
  drFree(x, &ier);
  /* get an array of peekable values from the simulation */
  t = time(&timevar)/60.0 + TIMEDIFF;
  for ( i = 0; i < ARRAYSIZE; i++ )
  {
    cdrgta(ARRAYPEEK[i], &address[i], &ier);
    passed("Array Peek", ier);
  }
  drgtvs(&t, &array_size, address, val, flag, &ier);
  passed("drgtvs", ier);
  for ( i = 0; i < ARRAYSIZE; i++ )
  {
    if ( flag[i] & DR_STRING )
    { /* val[i] can be converted to a status string */
      cdrv2s(&val[i], strval[i], &name_length, &ier);
      passed("cdrv2s", ier);
      printf("Flag[%d] = %d %s: %s\n", i, flag[i], ARRAYPEEK[i],
strval[i]);
    }
    else
    {
      printf("Flag[%d] = %d %s: %4.1f\n", i, flag[i], ARRAYPEEK[i],
val[i]);
    }
  }

  /* get a list of peek names associated with the transfer
  * lines in the model. Note: could be very expensive in
  * your real time model */
  cdrgta("PEEKLIST(\"*, K.L = T\")", &addr, &ier);
  passed("PeekList", ier);
  peeks = (char **)DrGetStringList(addr);
  ier = peeks == 0;
  passed("DrGetStringList", ier);
  while ( *peeks )
  {
    printf("%s\n", *peeks);
    peeks++;
  }
  drfrea(&addr, &ier);

  /* let the model finish */
  cdrptc("run", &ier);
  passed("3rd cdrptc", ier);
  return 0;
```

```
   }
   static void passed(char *name, int ier)
   {
     if (ier == 0 )
     {
       printf("%s ok\n", name);
     }
     else
     {
       printf("%s failed\n", name);
       exit(ier);
     }
}
static void sai_sync(char *name)
{
  int ier = 0;
  int iier = 0;
  int update= 0;
  int loop = 0;
  do
  {
    /* update shared memory */
    drupd(&update, &ier);
    /* ier = 0 when drupd is successful
    update = 1 when shared memory is updated
    ier = 0 and update = 0 means drupd call is ok, but TRANS has
    not taken a step since we last updated shared memory */
    if ( !update )
    {
      /* now wait for a second before the next try */
      spsSleep(1000, &iier);
      loop++;
    }
  } while ((loop < 120) && (ier == 0) && !update);
  if ( !update ) printf("%s update failed \n", name);
}
```

rpcdrtest

```
/*rpcdrtest

 * Purpose
 * -------
 * Sample program for the Client/Server version of the SPS
 * "Data Retrieval" interface.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
```

```
 * Gotchas
 * -------
 * This code references some devices in the test model provided by
 * GL (rpcdrtest). This code needs to be modified accordingly when
 * the model is changed.
 *
 * In order for this client program to run successfully, the server
 * process TRANSSHARE must be run along with the model.
 *
 */
#include <stdio.h> /* printf, sprintf */
#include <stdlib.h> /* exit */
#include <time.h> /* time */
#include "spsclient.h" /* GL drxxxx/cdrxxx functions */
static void passed(char* name, int ier);
static void sai_sync(char* name);
#define ARRAYSIZE 4 /* dimension of the array below */
char* ARRAYPEEK[] = {"BV1:P+", "BV1:ST", "TIME", "DT"};
char* PATH[] = {"dp1", "out1", 0};
/* difference between UNIX GMT time and DREM time */
const int TIMEDIFF = (366+365+1)*24*60;
int main(int argc, char** argv)
{
   int ier = 0;
   int numtries = 0;
   char* casename;
   char* peekname = "TIME";
   char* pokename = "MAXPRES";
   int addr;
   char name[81];
   char str[41];
   int name_length;
   float value;
   float newval;
   int points, time_points;
   drPointXY *curvedata, *the_point;
   double *x, *ta;
   int flag[ARRAYSIZE];
   int address[ARRAYSIZE];
   double val[ARRAYSIZE];
   char strval[ARRAYSIZE][80];
   const int array_size = ARRAYSIZE;
   time_t timevar;
   double t;
   int i, j;
   char** peeks;
   switch(argc)
   {
      case 1:
         casename = "bc";
```

```
      break;
   case 2:
      casename = argv[1];
      break;
   default:
      printf("Usage: %s casename\n", argv[0]);
      exit(-1);
}
/* get the connection. the client/server version of
 * cdrini can be called multiple times in the program */
cdrini(casename, &ier);
while ( ier && ++numtries < 10 )
{ /* try a few more times before quitting */
   spsSleep(1000, &ier);
   cdrini(casename, &ier);
}
passed("cdrini", ier);

/* retrieve information associated with a string-valued peekable */
/* first need to get the "getval addr" for the peekable */
cdrgta(peekname, &addr, &ier);
passed("1st cdrgta", ier);

/* retrieve the current value for the string-valued peekable */
name_length = sizeof(str);
cdrgts(&addr, str, &ier, &name_length);
passed("1st cdrgts", ier);
printf("String value for %s is %s\n", peekname, str);
/* get the name associated with the "getval addr"
 * look redundant here, well, just for a demonstration */
name_length = sizeof(name);
cdrgtn(&addr, name, &ier, &name_length);
passed("cdrgtn", ier);
printf("cdrgtn: expected %s, got %s\n", peekname, name);
/* send a command string to the simulation. you can put
 * any one of the valid interactive commands in the first
 * argument */
cdrptc("run for 1 step", &ier);
/* ier = 0 means the command has been successfully sent to
 * the simulation, it does not mean the command has been
 * successfully executed in the simulation */
passed("1st cdrptc", ier);
/* for some commands, you can check whether or not the command
 * has been executed successfully in the simulation. let us
 * try it for the above command */
sai_sync("first"); /* get local memory updated */
/* retrieve again the current value for the string-valued peekable */
name_length = sizeof(str);
cdrgts(&addr, str, &ier, &name_length);
passed("2st cdrgts", ier);
```

```
printf("String value for %s is %s\n", peekname, str);
/* should always call drfrea to free the memory allocated for the
 * "getval addr" after it is no longer needed */
drfrea(&addr, &ier);
passed("1st drfrea", ier);

/* retrieve information associated with a scalar-valued peekable */
/* get the "getval addr" for the peekable */
cdrgta(pokename, &addr, &ier);
passed("2nd cdrgta", ier);

/* retrieve the current value */
drgetv(&addr, &value, &ier);
passed("1st drgetv", ier);
printf("Analog value for %s is %.2f\n", pokename, value);
/* change the value of a pokable in the simulation */
newval = value * 0.9f;
printf("Set %s to %.2f\n", pokename, newval);
drputv(&addr, &newval, &ier);
passed("drputv", ier);

/* the above change will take place only after the simulation
 * takes a step, so let us send another command to the simulation */
cdrptc("run for 1 step", &ier);
passed("2rd cdrptc", ier);
sai_sync("second");

/* check if the change has been made */
drgetv(&addr, &value, &ier);
passed("2nd drgetv", ier);
printf("Analog value for %s is %.2f\n", pokename, value);
/* should always call drfrea to free the memory allocated for the
"getval addr" after it is no longer needed */
drfrea(&addr, &ier);
passed("1st drfrea", ier);
/* get distance profile data from the simulation */
/* distance plot data is not in the shared memory when first
 * requested. it will take one or more steps for TRANS to honor
 * the request */
numtries = 0;
do {
   cdrptc("run for 1 step", &ier);
   sai_sync("drGetDistPlot");
   drGetDistPlot(PATH, "temperature", &points, &curvedata, &ier);
} while ( ier && ++numtries < 10 );
the_point = curvedata;
for ( i = 0; i < points; i++ )
{
   printf("distance = %3.0f, temperature = %3.0f\n", curvedata->x,
   curvedata->y);
```

```
   curvedata++;
}
drFree(the_point, &ier);
/* let the simulation run for a few steps */
cdrgta(peekname, &addr, &ier);
passed("third cdrgta", ier);
do {
   cdrptc("run for 1 step", &ier);
   sai_sync("get time loop");
   drgetv(&addr, &value, &ier);
} while ( value < 10.0 );
drfrea(&addr, &ier);
/* get trend data from the simulation */
printf("current time is %4.1f\n", value);
/* get the trend data from the beginning of simulation upto current */
drGetTimePlot(0.0, 10.0, "PIPE1:P-", &points, &curvedata, &ier);
passed("drGetTimePlot", ier);
the_point = curvedata;
printf("points = %d\n", points);
for ( i = 0; i < points; i++ )
{
   printf("time = %4.1f, PIPE1:P- = %3.0f\n", curvedata->x, curvedata-
   >y);
   curvedata++;
}
drFree(the_point, &ier);
/* get historical profile data from the simulation */
/* get all available time */
drGetDistPlotTimes(&time_points, &x, &ier);
passed("drGetDistPlotTimes", ier);
ta = x;
for ( i = 0; i < time_points; i++ )
{
   printf("Request time = %3.1f\n", *ta);
   drGetDistPlotHist(PATH, "elevation", *ta, &points, &curvedata, &t,
   &ier);
   passed("drGetDistPlotHist", ier);
   the_point = curvedata;
   printf("Actual time = %3.1f\n", t);
   for ( j = 0; j < points; j++ )
   {
      printf("distance = %3.0f, elevation = %3.0f\n", curvedata->x,
      curvedata->y);
      curvedata++;
   }
   drFree(the_point, &ier);
   ta++;
}
drFree(x, &ier);
/* get an array of peekable values from the simulation */
```

```
   t = time(&timevar)/60.0 + TIMEDIFF;
   for ( i = 0; i < ARRAYSIZE; i++ )
   {
      cdrgta(ARRAYPEEK[i], &address[i], &ier);
      passed("Array Peek", ier);
   }
   drgtvs(&t, &array_size, address, val, flag, &ier);
   passed("drgtvs", ier);
   for ( i = 0; i < ARRAYSIZE; i++ )
   {
      if ( flag[i] & DR_STRING )
      { /* val[i] can be converted to a status string */
         cdrv2s(&val[i], strval[i], &name_length, &ier);
         passed("cdrv2s", ier);
         printf("Flag[%d] = %d %s: %s\n", i, flag[i], ARRAYPEEK[i],
         strval[i]);
      }
      else
      {
         printf("Flag[%d] = %d %s: %4.1f\n", i, flag[i], ARRAYPEEK[i],
         val[i]);
      }
   }
   /* get a list of peek names associated with the transfer
    * lines in the model. Note: could be very expensive in
    * your real time model */
   cdrgta("PEEKLIST(\"*, K.L = T\")", &addr, &ier);
   passed("PeekList", ier);
   peeks = (char **)DrGetStringList(addr);
   ier = peeks == 0;
   passed("DrGetStringList", ier);
   while ( *peeks )
   {
      printf("%s\n", *peeks);
      peeks++;
   }
   drfrea(&addr, &ier);

   /* let the model finish */
   cdrptc("run", &ier);
   passed("3rd cdrptc", ier);
   return 0;
}
static void passed(char *name, int ier)
{
   if (ier == 0 )
   {
      printf("%s ok\n", name);
   }
   else
```

```
   {
      printf("%s failed\n", name);
      exit(ier);
   }
}
static void sai_sync(char *name)
{
   int ier = 0;
   int iier = 0;
   int update= 0;
   int loop = 0;
   do
   {
      /* update shared memory */
      drupd(&update, &ier);
      /* ier = 0 when drupd call is successful
       * update = 1 when shared memory is updated&#9;(copied)
       * ier = 0 and update = 0 means drupd call is ok, but TRANS has
       * not taken a step since we last updated shared memory */
      if ( !update )
      {
         /* now wait for a second before the next try */
         spsSleep(1000, &iier);
         loop++;
      }
   } while ((loop < 120) && (ier == 0) && !update);
   if ( !update ) printf("%s update failed \n", name);
}
```

rpctodremtest

```
/* rpctodremtest.c
 *
 * Purpose
 * -------
 * Sample program for the Client/Server version of SPS
 * "SCADA to model" interface.
 *
 * Description
 * -----------
 * This program must be run with the server program todremserve.
 * This program connects to the remote todremserve process,
 * writes some rtudata to the local process buffer, sends the
 * buffered data to the server, and writes some more. This
 * program disconnects from the remote todremserve process before
 * exits.
 * The remote todremserve process opens the rtudata file, and
 * writes the data to the file. You can view the rtudata file
 * using the DRTU utility.
```

```
 *
 * The connection argument is a character string of the following
 * format:
 * hostname:portnumber:protocol:version
 * Todremserve process prints out this string to standard output
 * at startup.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include "spsclient.h"
#define QUALITY_GOOD 1
int main(int argc, char** argv)
{
   int chan=0;
   int numtries = 0;
   int ier;
   int qual = QUALITY_GOOD;
   int size = 513;
   int time = 1340;
   float value = 19.f;
   static char *point[] =
   {
      "PRESS01", "STATUS.09", "036_ANALOG", "099RATE33",
      "RTU007.PT023", "FLOWRATEONE", "TEMP0977"
   };
   if ( argc < 2 )
   {
      printf("Usage: %s <connection>\n", argv[0]);
      exit(-1);
   }
   /* get connection to the remote server process todremserve */
   cdrmin(argv[1], &chan, &size, &ier);
   while ( ier && numtries++ < 10 )
   { /* try a few more times for the connection */
      spsSleep(1000, &ier);
      cdrmin(argv[1], &chan, &size, &ier);
   }
   if ( ier )
   {
      printf("cdrmin failed, ier = %s.\n", ier);
      exit(-1);
   }
   while ( time < 1000000 )
   {
      value += 7;
      ctodrm(&chan, point[((time)%(7))], &time, &value, &qual, &ier);
```

```
      if ( ier ) printf("ctodrm failed.\n");
      if( ((time)%(67)) == 0)
      {
         siflsh(&chan, &ier);
         if ( ier ) printf("siflsh failed.\n");
      }
      time += 120;
   }
   cdrmout(chan, &ier);
   if (ier) printf("cdrmout failed.\n");
   /* give todremserve a chance to write the last data to the file */
   spsSleep(5000, &ier);
   return 0;
}
```

rtuapitest

```
/* rtuapitest.c
 *
 * Purpose
 * -------
 * Sample program for SPS's RTUDATA to GUI interface.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
#include <stdio.h> /* setbuf printf */
#include <stdlib.h> /* system */
#include "saiInterface.h" /* rtuOpen() rtuReadNextPt() rtuSeek() rtuClose()
*/
#define NUMFILES 3
static void PrintMsg(int ier, char* action, char* filename);
static void CreateRTUFiles(void);
   int main(void)
{
   const int MXSTR = 256;
   int i, ses, ier = 0;
   int session[NUMFILES+1];
   int mxpts=0;
   int ptnum, rtime, qual;
   char ident[81];
   char *qualityStrings[]={ "BAD", "GOOD", "MANUAL", "NOCHANGE1",
   "NOCHANGE2" };
   float value;
   int nfail = 0;
   char* filename[NUMFILES]={"rtu1.dt","rtu2.dt","rtu3.dt"};
   char* fname = filename[2];
   int rtimein[]={3000, SMALLTIME, BIGTIME};
   setbuf(stdout, 0);
```

```
/* Here, we submit jobs that create three rtudata files
 * Note: this interface can be run against "hot" rtudata
 * files. For the purpose of result comparison, we use
 * "cold" rtudata file here */
CreateRTUFiles();
for(i = 0; i < NUMFILES; i++)
{
   rtuOpen(filename[i], READ_ONLY, mxpts, &session[i], &ier);
   PrintMsg(ier, "Open", filename[i]);
}
/* Try to open a non-existant file, should fail */
rtuOpen("junkfile", READ_ONLY, mxpts, &session[3], &ier);
PrintMsg(ier, "Open non-existant file", "junkfile");
/* Position to the end of the files */
for(i = 0; i < NUMFILES; i++)
{
   rtuSeek(session[i], BIGTIME, &ier);
   PrintMsg(ier, "Seek", filename[i]);
}
/* Read from the rtudata files at the same time */
i = 0;
printf("%8s %6s %8s %5s %4s %10s %5s\n",
"filename", "ptnum", "rtime", "ident", "value", "quality", "ier");
while (nfail < 10)
{
   rtuReadNextPt(session[i], &ptnum, &rtime, ident, &value, &qual, &ier);
   if(ier)
   { /* no new data */
      printf("Read failed for file %s", filename[i]);
      i = (i + 1) % 3; /* next file */
      nfail++;
      spsSleep(1000, &ier);
      printf(", switching to %s, fail count %d\n", filename[i], nfail);
   }
   else
   {
      printf("%-8s %6d %8d %-5s %4f %-10s %d\n",
      filename[i], ptnum, rtime, ident, value, qualityStrings[qual],
      ier);
   }
}
/* Close the files */
for(i = 0; i < 2; i++)
{
   rtuClose(session[i],&ier);
   PrintMsg(ier, "Close", filename[i]);
}
/* Check seeking from beginning of the file and a specific
 *place to the end.
 */
```

```
   rtuOpen(fname, READ_ONLY, mxpts, &ses, &ier);
   PrintMsg(ier, "Open", fname);

   for(i=0; i<3; i++)
   {
      printf("\nSeeking for time %d\n", rtimein[i]);
      rtuSeek(ses, rtimein[i], &ier);
      PrintMsg(ier, "Seek", fname);
      printf("%8s %6s %8s %5s %4s %10s %5s\n",
      "filename", "ptnum", "rtime", "ident", "value", "quality", "ier");
      while(1)
      {
         rtuReadNextPt(ses, &ptnum, &rtime, ident, &value, &qual, &ier);
         if(ier) break;
         printf("%-8s %6d %8d %-5s %4f %-10s %d\n", filename[2], ptnum,
         rtime, ident, value, qualityStrings[qual], ier);
      }
   }
   rtuClose(ses,&ier);
   PrintMsg(ier, "Close", fname);
   return 0;
}
/* This function prints results of an action */
static void PrintMsg(int ier, char* action, char* filename)
{
   printf("%s %s %s %s\n", action, (ier ? "failed" : "okay"), "for file",
   filename);
}
/* This function submitts three commands to create three rtudata files
 * using the SPS's utility executable rtugen. Note that rtugen must
 * be in your path
*/
static void CreateRTUFiles(void)
{
   int i;
   /* Parameters for rpcgen:
    * 1 pts per second (copies=1)
    * saving 20 minutes of data (maxpts=copies*20*60=1200)
    * 1 hour (npts=10*60*60=3600)
    * at 100 times real time (speed=100)
   */
#if unix
   char* commands[NUMFILES] =
   {
      "rtugen rtu1.dt -id=id1 -maxpt=1200 -npts=3600 -speed=100",
      "rtugen rtu2.dt -id=id2 -maxpt=1200 -npts=3600 -speed=100",
      "rtugen rtu3.dt -id=id3 -maxpt=1200 -npts=3600 -speed=100"
   };
#elif _MSC_VER
   char* commands[NUMFILES] =
```

```
   {
      "rtugen rtu1.dt -id=id1 -maxpt=1200 -npts=3600 -speed=100",
      "rtugen rtu2.dt -id=id2 -maxpt=1200 -npts=3600 -speed=100",
      "rtugen rtu3.dt -id=id3 -maxpt=1200 -npts=3600 -speed=100"
   };
#else
   char* commands[NUMFILES] = {"", "", ""};
#endif
   for(i = 0; i < NUMFILES; i++)
   {
#if unix || _MSC_VER
   system(commands[i]);
#else
   ;
#endif
   }
}
extern "C" int rtuapitest(void);
int main(void)
{
   return rtuapitest();
}
```

todremtest

```
 /* todremtest.c
 *
 * Purpose
 * Sample program for the SPS "SCADA to model" interface.
 *
 * Description
 * -----------
 * This program creates an rtudata.dt file from fake data.
 * You can view the rtudata.dt file using the DRTU utility.
 *
 * This file is delivered to our clients. Any modification to
 * this file must be fully tested on all supported platforms.
 *
 */
#include <stdio.h> /* fprintf sprintf setbuf */
#include <stdlib.h> /* exit */
#include "saiInterface.h" /* cdrmin ctodrm siflsh sclose */
const int MAX_SCAN_NUM = 10;
const int MAX_SCAN_DAT = 20;
/* difference between UNIX GMT time and DREM time (in seconds) */
const int TIMEDIFF = (366+365+1)*24*60*60;
/* fake starting time stamp and update rate */
const int STARTTIME = 820454400; /* 00:00:00, Jan 1, 1996, in seconds since
00:00:00 GMT, Jan. 1, 1970 */
```

```
const int UPDATETIME = 60; /* in seconds */
typedef struct
{
   char ident[80];
   int timeStamp;
   float value;
   int quality;
} ScadaPoint;
enum
{
   QUALITY_BAD = 0,
   QUALITY_GOOD,
   QUALITY_MANUAL,
   QUALITY_NOCHANGE1,
   QUALITY_NOCHANGE2
};
   int main(void)
{
   char* filename = "rtudata.dt"; /* rtudata file name */
   int ichan = 0; /* input channel to rtudata file */
   int mxpts = 0; /* cdrmin sets mxpts to default 1000000 */
   int status = 0; /* return value from GL supplied functions */
   char* basename = "SCADA.DATA";
   ScadaPoint scada_data;
   int scan, data;

   setbuf(stdout, 0);
   /* open rtudata file */
   cdrmin(filename, &ichan, &mxpts, &status);

   if ( status )
   {
      fprintf(stdout, "Failed to open file %s\n", filename);
      exit(-1);
   }

   fprintf(stdout, "File %s opened on channel %d, status %d\n",
   filename, ichan, status);
   /* UNIX GMT time needs to be converted to DREM time */
   /* What you really need to use is the timestamp from your SCADA
    * database converted to DREM time. If your timestamp is in
    * UNIX GMT, add the TIMEDIFF.
    * scada_data.timeStamp = STARTTIME + TIMEDIFF;
   for (scan = 0; scan < MAX_SCAN_NUM; scan++)
   {
      fprintf(stdout, "\nFake SCADA scan %d, RTU time stamp: %d\n", scan,
      scada_data.timeStamp);
      for (data = 0; data < MAX_SCAN_DAT; data++)
      {
```

```
            /* make up some scada data point names, values, and quality flags
            */
            /* You should use whatever protocol you have to get these
            information
             * from your SCADA database */
             sprintf(scada_data.ident, "%s%.2d", basename, data);
             scada_data.value = (float)scan + data;
             scada_data.quality = QUALITY_GOOD;

             /* When you are debugging/testing, it is always a good idea to
            print
             out the arguments passed to ctodrm right before it is called */
             fprintf(stdout, "SCADA point name: %s Value: %12.6f Quality:
            %d\n", scada_data.ident, scada_data.value, scada_data.quality);

             ctodrm(&ichan, scada_data.ident, &scada_data.timeStamp,
            &scada_data.value, &scada_data.quality, &status);
             if ( status )
             {
             fprintf(stdout, "Failed to write to file %s\n", filename);
             exit(-1);
          }

      }
      siflsh(&ichan, &status);
      if ( status )
      {
         fprintf(stdout, "Failed to flush file %s\n", filename);
         exit(-1);
      }
      /* Now wait for the next scan */
      /* You may want to use "sleep" to reduce CPU usage */
      scada_data.timeStamp += UPDATETIME;
   }

      cdrmout(ichan, &status);
      if ( status )
      {
      fprintf(stdout, "Failed to close file %s\n", filename);
      exit(-1);
      }

      return 0;
   }
```

transexporttest

```
/* transexporttest.c */
#include "saiInterface.h"
int main(int argc, char const*const* argv)
```

```
   {
      return transexport(argc, argv, toremote, remotealm);
   }
/* remotealm.c
// void remotealm
// (
// const char *alname --> alarm name string
// const char *almsg --> alarm message string
// int *ier <-- returned non-zero if error
// )
//
// This routine is part of the documented API; any functionality
// changes must be upwardly compatible. These header comments are
// inserted verbatim into the API Reference Manual and must be kept
// user-quality.
//
// Purpose
// -------
// This is the data export routine. See the Statefinder and Leakfinder
// documentation for full details.
//
// Usage Considerations
// --------------------
// The applications programmer needs to supply a version of this routine
// to export simulation alarms. The applications programmer can
// implement this routine to do a wide variety of things. For example,
// this routine can print a report, load alarms into a database, and/or
// pass alarms to another application.
//
// GL provides a version of this routine in the sample program for the
// transexport interface (transexporttest). The GL version simply echoes
// the input parameters to stdout.
//
// The transexport main() calls this routine periodically to export any
// alarm that has changed status or message text.
//
// For each EXPORTALARM configured in the INEXPT file, transexport main()
// does the following:
//
// 1) maintain old status and export message,
// 2) get the current status and alarm message from the simulation, and
// 3) if the status or message has changed, call remotealm()
// (but not if the message is either zero length or all blanks).
//
// The transexport main() ignores the return value of "*ier".
//
// SeeAlso
// -------
// toremote()
// Section "Statefinder to SCADA"
```

```
// Staefinder and Leakfinder documentation.
//
// Library
// -------
// n/a
//
// Licensing
// ---------
// DRINIT TRANSEXPORT
//
*/
#include <stdio.h>
#include "saiInterface.h"
void remotealm(const char *alname, const char *almsg, int *ier)
{
   printf(" %s %s \n", alname, almsg);
   fflush(stdout);
   *ier = 0;
}
/* toremote.c
// void toremote
// (
// const char **pvname --> array of pointers to variable names
// const double dval[] --> array of model variable values
// const int n --> number of data points
// int *ier <-- returned non-zero if error
// )
//
// This routine is part of the documented API; any functionality
// changes must be upwardly compatible. These header comments are
// inserted verbatim into the API Reference Manual and must be kept
// user-quality.
//
// Purpose
// -------
// This is the data export routine. See the Statefinder and Leakfinder
// documentation for full details.
//
// Usage Considerations
// --------------------
// The applications programmer needs to supply a version of this routine
// to export simulation data. The applications programmer can implement
// this routine to do a wide variety of things. For example, this
// routine can print a report, load data into a database, and/or pass
// data to another application.
//
// GL provides a version of this routine in the sample program for the
// transexport interface (transexporttest). The GL version simply echoes
// the input parameters to stdout.
//
```

```
// The transexport main() calls this routine periodically. The transexport
// main() calls this routine with all EXPORTDATA simulation variables that
// are configured in the INEXPT file.
//
// The transexport main() ignores the return value of "*ier".
//
// SeeAlso
// -------
// remotealm()
// Section "Statefinder to SCADA"
// Statefinder and Leakfinder documentation
//
// Library
// -------
// n/a
//
// Licensing
// ---------
// DRINIT TRANSEXPORT
//
*/
#include <stdio.h>
#include "saiInterface.h"
void toremote(const char **pvname, const double dval[], const int n, int
*ier)
{
   int i;
   for ( i = 0; i < n; i++ )
      printf(" %s %f \n", pvname[i], dval[i]);
   fflush(stdout);
   *ier = 0;
}
```

Compiling

When a new version of the SPS is received, normally you are required to re-link your interface program, in order to make your interface programs work with the new SPS executables The link procedure may vary as the SPS version changes. It certainly differs with different interfaces and different operating systems. Before you link, please find the link procedure relating to your corresponding operating system and interface. Your calling routines do not need to be recompiled unless they are modified, or unless the compiler or the operating system on your machine is upgraded.

In the link commands and examples, `YOUR_CALLING_ROUTINES` refers to the routines you implemented to call the API functions, including your main routine. `OTHER_LIBS_PATH` and `OTHER_LIBS` stand for the path and the name of other libraries you may need to link with (such as third party software libraries, system libraries, etc.). Note that the order of the libraries for linking with is important in most operating systems.

The various C-language API's for SPS are written in C and can typically be called from any code capable of C-language calls. The header file, saiInterface.h, has prototypes for these API's and is usable from C or C++.

In the compiling and linking commands provided, it is assumed you compile and link your interface programs in debug mode. It is recommended that you use the debug option in the test stage.

Compiling and linking on Solaris

See the system requirements documentation provided with your installation package for information on the specific platforms supported by SPS.

The SPS libraries are written in C, and should work with most C or C++ products on Solaris.

In the commands below, we assume you have the following three environment variables defined to point to the correct version of the SPS installation: DREMHOME, DREMARCH, DREMVERSION. For example, if SPS is installed under /usr/local/sai/solaris/v940, then these three environment variables should be defined as:

Under C Shell, use

```
> setenv DREMHOME "/usr/local/sai"
> setenv DREMARCH "solaris"
> setenv DREMVERSION "v940"
```

Under Bourne Shell, use

```
> DREMHOME = "/usr/local/sai"; export DREMHOME
> DREMARCH = "solaris"; export DREMARCH
> DREMVERSION = "v940"; export DREMVERSION
```

Compiling and linking on Solaris - Trainer interface

The following information is to aid in linking the user developed Trainer Interface program (CIP) with SPS libraries. These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>cc -g -o cip YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on Solaris - SCADA to Statefinder/ Leakfinder interface

The following information is to aid in linking the user developed SCADA to Statefinder/Leakfinder Interface program (TODREM). These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>cc -g -o todrem YOUR_CALLING_ROUTINES \
```

```
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lspsclient -lstclib \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on Solaris - Statefinder/Leakfinder to SCADA interface

The following information is to aid in linking the user developed Statefinder/Leakfinder to SCADA Interface program (TRANSEXPORT). These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_main.c YOUR_toremote.c YOUR_remotealm.c \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>cc -g -o transexport YOUR_main.o \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> YOUR_toremote.o YOUR_remotealm.o \
> -lspsclient -lstclib \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on Solaris - Reading RTU data files

The following information is to aid in linking the user developed RTU data to GUI Interface program (RTUREAD). These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>cc -g -o rturead YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lspsclient -lstclib \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on Solaris - Low-level API calls (shared memory version)

The following information is to aid in linking the user developed programs using the SPS low-level API calls. With this version of the link, calls to `cdrini()` will be resolved through the shared-memory interface, using information in the CASENAME.VYDEF file. This requires SPS to be running on the same machine as the client. It does not require any particular server to be running when `cdrini()` is called. These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
```

```
>cc -g -o md2GUI YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -ldrvydefclient -lspsclient -lstclib \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on Solaris - Low-level API calls (client/server version)

The following information is to aid in linking the user developed programs using the SPS low-level API calls. With this version of the link, calls to `cdrini()` will be resolved through the transshare interface, using information in the CASENAME.SHARE file. This allows SPS to be running on any machine accessible on the local network. It requires that the transshare program be running before `cdrini()` is called. These are general guidelines.

Typical commands to compile and link the interface are:

```
>cc -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>cc -g -o md2GUICLNT YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -ldrshareclient -lspsclient -lstclib \
> -lOTHER_LIBS \
> -lm -lsunmath /usr/ucblib/libucb.a -lnsl -lintl -lmalloc
```

Compiling and linking on HPUX

See the system requirements documentation provided with your installation package for information on the specific platforms supported by SPS.

The SPS libraries are written in C, and should work with most C or C++ products on HPUX.

In the following commands, it is assumed assume you have the following three environment variables defined to point to the correct version of the SPS installation: DREMHOME, DREMARCH, DREMVERSION. For example, if SPS is installed under /usr/local/sai/hpux/v940, then these three environment variables should be defined as:

Under C Shell, use

```
> setenv DREMHOME "/usr/local/sai"
> setenv DREMARCH "hpux"
> setenv DREMVERSION "v940"
```

Under Bourne Shell, use

```
> DREMHOME = "/usr/local/sai"; export DREMHOME
> DREMARCH = "hpux"; export DREMARCH
> DREMVERSION = "v940"; export DREMVERSION
```

Compiling and linking on HPUX - Trainer interface

The following information is to aid in linking the user developed Trainer Interface program (CIP) with SPS libraries. These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o cip YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lspsclient -lstclib \
> -lOTHER_LIBS -lm
```

Compiling and linking on HPUX - SCADA to Statefinder/ Leakfinder interface

The following information is to aid in linking the user developed SCADA to Statefinder/Leakfinder Interface program (TODREM). These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o todrem YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lspsclient -lstclib\
> -lOTHER_LIBS -lm
```

Compiling and linking on HPUX - Statefinder/Leakfinder to SCADA interface

The following information is to aid in linking the user developed Statefinder/Leakfinder to SCADA Interface program (TRANSEXPORT). These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_main.c YOUR_toremote.c YOUR_remotealm.c \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o transexport YOUR_main.o \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> YOUR_toremote.o YOUR_remotealm.o \
> -lspsclient -lstclib \
> -lOTHER_LIBS -lm
```

Compiling and linking on HPUX - Reading RTU data files

The following information is to aid in linking the user developed RTU data to GUI Interface program (RTUREAD). These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o rturead YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -lspsclient -lstclib \
> -lOTHER_LIBS -lm
```

Compiling and linking on HPUX - Low-level API calls (shared memory version)

The following information is to aid in linking the user developed programs using the SPS low-level API calls. With this version of the link, calls to `cdrini()` will be resolved through the shared-memory interface, using information in the CASENAME.VYDEF file. This requires SPS to be running on the same machine as the client. It does not require any particular server to be running when `cdrini()` is called. These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o md2GUI YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -ldrvydefclient -lspsclient -lstclib \
> -lOTHER_LIBS -lm
```

Compiling and linking on HPUX - Low-level API calls (client/server version)

The following information is to aid in linking the user developed programs using the SPS low-level API calls. With this version of the link, calls to `cdrini()` will be resolved through the transshare interface, using information in the CASENAME.SHARE file. This allows SPS to be running on any machine accessible on the local network. It requires that the transshare program be running before `cdrini()` is called. These are general guidelines.

Typical commands to compile and link the interface are:

```
>c89 -g -c YOUR_CALLING_ROUTINES \
> -I$DREMHOME/$DREMARCH/$DREMVERSION/custsrc/include
>c89 -g -o md2GUICLNT YOUR_CALLING_ROUTINES \
> -L$DREMHOME/$DREMARCH/$DREMVERSION/lib \
> -LOTHER_LIBS_PATH \
> -ldrshareclient -lspsclient -lstclib \
> -lOTHER_LIBS -lm
```

Compiling and linking on Windows

See the system requirements documentation provided with your installation package for information on the specific platforms supported by SPS.

On Windows, we provide project files (.dsp) for all of the above interfaces. These project files are located at SPS's custsrc directory, under the sample program subdirectory corresponding to the specific interface. You can load the project file into the Microsoft Development Studio.

CHAPTER

20

Utilities

The following utilities may be used with SPS:

- *DEMAC*—Processes any of the various SPS input files, such as INPREP or INTRAN, to expand MACRO, INCLUDE, IFELSE, and other data processing commands to their full text equivalents.
- *TELLTRANS*—Allows you to issue commands from the system prompt to an executing TRANS process, regardless of whether TRANS is being run interactively or not.

DEMAC

Note: If you are a Windows user, you will likely want to use the DEMAC window provided from SPS. For more information, see "Expanding files that contain macros and include statements" on page 585.

```
DEMAC INPUT-FILE OUTPUT-FILE
[-MEMSUM]
[-UPPERCASE]
[-NOBLANKLINES]
[-NOREPEATEDBLANKLINES]
```

Field	Description
INPUT-FILE	Name of the existing INPREP or INTRAN file to be processed.
OUTPUT-FILE	Name of output file that is created and contains the results of the DEMAC processing.
MEMSUM	Memory usage summary.
UPPERCASE	Converts the output file to all uppercase.
NOBLANKLINES	Removes all blank lines from the output file.
NOREPEATEDBLANKLINES	Removes duplicate blank lines from the output file.

Description

DEMAC is a utility program that processes any of the various SPS input files, such as INPREP or INTRAN, to expand MACRO, INCLUDE, IFELSE, and other data processing commands to their full text equivalents.

The created OUTPUT-FILE is a fully-functional input file that can be used as an input file to its corresponding program, such as PREPR or TRANS.

See the "User-defined Variables, Macros, and Include Files" on page 567 for a further description of these commands.

Take note

DEMAC is used primarily as a debugging tool. For example, PREPR and TRANS assist in finding errors in input data files by writing error messages with a line number to the OUTPRP or OUTTRN file. If SPS finds an error in an include file, the error message in OUTPRP or OUTTRN shows the line number of the INCLUDE command itself. In this case, it is often useful to pre-process the INPREP or INTRAN file with DEMAC and run PREPR or TRANS on the output file from DEMAC. In this way, the INCLUDE command is expanded and the error is more easily located.

Any comments in the INPUT-FILE are copied to the OUTPUT-FILE.

Example input

The following examples show how the DEMAC utility is used:

```
demac /disk/e/models/50hdd_run.intran debug.intran
```

TELLTRANS

```
TELLTRANS MODEL COMMAND
```

Field	Description
MODEL	Name of executing TRANS process with which TELLTRANS is to communicate commands.
COMMAND	Any valid interactive TRANS command.

Description

TELLTRANS is a utility program that allows you to issue commands from the system prompt to an executing TRANS process. This program allows interactive TRANS commands to be sent to a TRANS process whether it is being run interactively or not. TELLTRANS can be thought of as a remote command line for the TRANS process.

TELLTRANS is run from the same directory (and same computer) in which TRANS is running. The MODEL parameter is the same case name as the executing TRANS process. The COMMAND parameter can be any valid interactive TRANS command. TELLTRANS does not validate the TRANS COMMAND parameter; it merely passes the command verbatim to TRANS, as if it were typed on the TRANS command line.

The INTRAN file of the TRANS process must contain the SHARE command.

Take note

TELLTRANS is primarily used as a way to communicate with TRANS processes that are run in a background or detached mode. Some of the more typical uses of TELLTRANS is to communicate to a simulation interactive commands such as archiving a state, pausing the simulation, loading a previously archived state, changing equipment statuses, halting a simulation or issuing a submit sequence.

Example input

The following examples show how to use the TELLTRANS utility:

```
telltrans max_flow pause
telltrans testing "stop pump_1; run for 1 hour"
```

Glossary

This glossary provides terms and definitions for Stoner Pipeline Simulator (SPS) and related products.

activity

Action performed by SPS at a specified time, such as opening or closing a valve.

ALMLOG file

A text file created by TRANS that contains a list of alarm messages for which the conditions have been met. Associated times are not stored in this file. Also see “EVENTS file”.

archive file (ARK)

A binary file that contains a “snapshot” of the system at a given point in the simulation. may be used as the initial status for a subsequent simulation or to re-initialize the current (ongoing) simulation using the LOAD.STATUS command. Structurally, these files are similar to RESTRT files and may be used as RESTRT files to initialize TRANS. When loaded using the LOAD.STATUS command, an archive file may be used across platforms and architectures. However, when used as a RESTRT file, the archive must be read using the same computer architecture with which it was written.

batch run

A simulation in which all simulation control data is entered into the INTRAN file and involves no user interaction or control during the simulation. The simulation runs for the entire length specified in the INTRAN file.

built-in units

Units that are pre-programmed into SPS and can be selected individually using the USEUNITS command.

case

A collection of input and output files associated with a particular pipeline model. All files share the same model name and have different file extensions.

chart

A generic term used to describe a graph and its associated labels, axes, etc.

cold review file

A review file that TRANS is no longer updating.

command

Control that applies or accesses a Stoner Software application function. You apply commands by selecting menu items or icons, pressing toolbar buttons or dialog box buttons, or entering text in a text box or data file.

connection string

A specified syntax that determines the required server and data source for an enterprise application.

control element

Actuators, sensors, monitors, etc. used in controls systems.

Control Manager

The Control Manager, an ActiveX control provided with Schematic, serves as a central processor for the communication of data and displays connection status.

data curve

A set of reusable curve data defined in the INPREP file.

data source

Path to the server and/or the model data.

DEMAC

A utility that processes SPS input files, such as INPREP or INTRAN, to expand MACRO, INCLUDE, IFELSE, and other data processing commands to their full text equivalents.

device

Collective term for nodes, elements, externals, and other conceptual groupings, such as a defined path.

diagnostic flows (DF)

.

display (DSP) file

Text files that store formats for the display of data, including text, reports, and plots, during an interactive simulation. Text displays are formats for display of text and other printable characters, and are created by using a text editor. Reports and plots are created interactively during a simulation.

display

The window or terminal display, as defined in a display (DSP) file.

distance plot

A graph of a variable over a distance or defined path. Also called a “profile”.

DSP file

See "display (DSP) file".

echo

A repeat of the input data in an output or report file.

element

A feature in the model that represents a component in a piping system—such as a pipe, valve, or pump—through which fluid flows. Also see "control element".

EVENTS file

A text file created by TRANS that contains all of the events (triggered alarms and error messages) and the times at which they occur during the simulation. This file is appended as events occur during the simulation. When an event occurs, an En appears in the upper-left corner of the current display. The E indicates an event has occurred and the n is the number of events since the event summary was last viewed.

external

A device used to introduce flow into or take flow out of a system. More than one external can be attached to a single node.

fluid mixture vector (FMV)

The internal point added to a batch whenever the trivial concentration limit is violated. It moves the batch down the pipeline so TRANS can locate changes in fluid properties. Also referred to as fluid movement vector.

from-node

The node that defines the start point of an element.

include file

A user-created text file that contains data or commands and may be called by another file for processing.

initialization

The initial state of the pipeline model, including pressure and flow values and the current status of operational equipment.

INPREP file

An input text file that describes the physical system being modeled.

interactive job

A simulation in which instructions from the user may be issued during the simulation in addition to simulation control data already defined in the INTRAN file.

internal units

Units that SPS uses for its internal calculations. In most circumstances, the SPS internal units are not relevant to the user.

INTRAN file

A required text file that establishes simulation control parameters (such as the time interval to be simulated) and may contain a pipeline operating scenario for a simulation. All INTRAN commands must be entered in capital letters.

keyletter

Similar to a keyword, a keyletter signals a certain type of data entry in an input file or dialog box. Keyletters are usually abbreviations.

keyword

A keyword signals a certain type of data entry in an input file or dialog box.

knots

Standard distance increment at which SPS calculates pressure, flow, etc. SPS determines the knot spacing automatically but it can be manually set as well.

leak detection threshold (LDT)

Dynamically calculated threshold determining the minimum leak which can be alarmed based on data uncertainty and transients in the system.

Leakfinder

SPS module for pipeline operations surveillance and leak detection. Leakfinder combines Statefinder, which uses real-time SCADA data to track pipeline hydraulics and to signal measurement anomalies, and Leakanalyzer, which scrutinizes those anomalies looking for possible leak signatures. When a leak is detected, SPS Leakfinder raises an alarm that shows the leak location, the onset time, the leak rate, and the total volume released.

model

A mathematical, and potentially graphical, representation of a hydraulic piping network used to analyze engineering problems. Consists of at least one element and two nodes, but can accommodate operational elements, control elements, and operational characteristics. A model is defined in various input files, such as INPREP and INTRAN files. Input and output files for a single model may be referred to as a “case”.

.NET

.NET Framework. Application/Software framework for use with contemporary Microsoft Windows Operating Systems. Programs within the .NET execute in a managed runtime environment.

node

In a piping system model, a point that represents a juncture or endpoint of a pipe or operational element, a demand or supply point in the system, or a pressure calculation point. Nodes have attributes such as name, pressure, and flow.

Online suite

Statefinder, Leakfinder, and Predictor are built upon SPS technology and are used as aids by pipeline operators in performing their daily tasks. Unlike SPS, Statefinder and Leakfinder simulate the behavior of a pipeline using real data from a SCADA system. Predictor then predicts the future state of the pipeline by extending the current state of the pipeline, obtained from Statefinder, forward in time.

OUTPRP file

A legacy text file created by PREPR that contains summary information on the INPREP file and any error messages or warning messages. An upgraded OUTPRP report is available. See "OUTPRP report".

OUTPRP report

An HTML report, available on all platforms, that provides summary information on the INPREP file, error messages or warning messages, and the ability to browse the report through hyperlinks.

OUTTRN file

A text file created by TRANS that contains simulation summary information, including a history log of equipment changes and actions taken, tabular reports, error messages and warning messages, etc.

peek

A command to display or output a simulation variable, which cannot be changed.

plot

A distance plot (profile) or time plot (trend) that may be displayed through or any other display or output format available through Stoner Software applications. May also refer to a compressor/pump map plot.

poke

A command to set an attribute to a certain value or setting.

Predictor

Part of the "Online suite", Predictor runs "look ahead" or "what-if" hydraulic simulations, depending on whether the application is being used in automatic mode or as a planning tool, respectively.

PREPR

An SPS program that preprocesses data before running the simulation. PREPR reads the INPREP file and evaluates it for input errors and connectivity problems.

Pressure Drop Forces (PDF)

PDFs are modeling forces that increase or decrease the pressure difference across a span.

profile

See "distance plot".

ramp

A set of curve data defined in the INTRAN file that describes the relationship between two variables. A ramp may be applied in SPS and related applications to change the value of a variable over time.

REPLAY file

A text file created when TRANS is run that contains all the commands issued during the simulation, both from the INTRAN file and interactively. The REPLAY file is an echo of the INTRAN file along with the interactive commands that were made during the simulation appended to the file. A simulation can be re-created by using the REPLAY file from a previous run as the INTRAN file for a new run.

RESTRT file

A binary file that contains information on the physical system that is used by TRANS for initialization. A RESTRT file is created when PREPR runs successfully, and this file may be updated and/or added to by TRANS. An archive file may be substituted for the RESTRT file. See “archive file (ARK)”.

REVIEW file

A binary file created by TRANS that contains data for user-defined values for each time step taken. Specifically, the REVIEW file stores time plot (TRENDLIST) and/or distance plot (PROFILE) data during the simulation. This data is used by TRANS to produce time plots, and by to produce time and/or distance plots and reports.

RTU data

Data received from a remote terminal unit.

SCADA

Supervisory control and data acquisition system

set point

A specified value used to regulate a system, set of alarms, etc.

shared memory

Portion of CPU memory that TRANS allows TPORT and SPS modules to access while running.

Show

The Show command displays data associated with a specific device or defined variable on a Show window.

simulation

The iterative process of solving pressure and flow equations to determine how a modeled network responds to the conditions specified in the model data.

simulation time

Time used by SPS during a simulation, which may be relative to the start of the simulation or relative to a specific clock time.

span

A series of elements, including pipes, headers, block valves, check valves, control valves, and externals. Each element connects to only two elements, one on each end, except for an external connection. Spans are automatically generated and named. The name is the first element name, an underscore (_), and the last element name. Each span has a Show window.

status file (STA)

A binary file that contains a “snapshot” of the system at a given point in the simulation and may be used as the initial status for a subsequent simulation or to re-initialize the current (ongoing) simulation using the LOAD.STATUS command. When loaded using the LOAD.STATUS command, a status file may be used across platforms and architectures. Unlike an archive file (ARK), a status file may not be used as a RESTRT file.

Statefinder

Using real-time measurement data from a Supervisory Control and Data Acquisition (SCADA) system, Statefinder maintains a constantly available picture of the current operation of the pipeline. It provides detailed information the ongoing operation of the pipeline even when operating with degraded or limited SCADA data.

steady state

A condition in which all variables are assumed to remain constant with time. An SPS simulation may be initialized at steady state through an STS file.

steady state file (STS)

A file, generated either in SPS or SynerGEE, that contains steady-state results that may be imported into SPS for initialization.

STS file

See “steady state file (STS)”.

TELLTRANS

A utility that allows interactive commands to be issued from the operating system’s system prompt to an executing TRANS process.

text display

A custom display that may combine text and peek attributes, and poke attributes to display rudimentary schematics with static or dynamic text as labels and may also offer editable fields.

time plot

A graph of a variable at a specified location over a time. Also called a “trend”.

time step

Time interval at which SPS calculates pressure, flow, etc. SPS determines the time step automatically but it can be manually set as well.

to-node

A node that defines the end point of an element.

TPORT

An SPS transport program that connects to TRANS for viewing multiple displays for a model. TPORT may also connect to a “cold” “REVIEW file” to review the results of a previous simulation.

TRANS

An SPS program that runs a transient analysis. TRANS uses the “RESTRT file” and the “INTRAN file” to calculate the time-varying effects of operating changes on pressure and flow in the pipeline system.

TRANSEXPORT

The data and alarm export utility, TRANSEXPORT, is used to send data and/or alarms back to the SCADA system or to any other process or data store. TRANSEXPORT reads the *model*.INEXPT file for a list of model variables and alarms that are to be exported. Upon startup, the list is validated against TRANS shared memory. Any errors are written to the *model*.OUTXPT file. TRANSEXPORT monitors the status of TRANS, if TRANS is exited TRANSEXPORT exits. If TRANS exits without a user-defined HALT or QUIT, TRANSEXPORT exits after the time-out period.

TRANSSHARE

Remote Procedure Call (RPC) server that allows for communication between TRANS and dr* clients over a computer network.

trend

See “time plot”.

user units

Units that you use to input data (like pressure, flow, length, diameter, etc.) in the INPREP file, INTRAN file, or interactively. These units are used by SPS for both input and output.

user-defined units

Additional units that the user has defined using the DEFUNITS command. In order to use a unit that is not built-in to SPS, the unit must be defined using DEFUNITS and then selected using USEUNITS.

VYDEF file

A text file created when TRANS is first run. TPORT reads this file upon startup for information on the REVIEW file and shared memory.

WCF

Windows Communication Foundation. Part of the .NET Framework, this is a framework that allows applications to communicate with one another, usualy using WSDL. A WCF service consists of the service class, a hosting environment and one or more endpoints (through which all communication is accomplished).

Index

B

C

E

F

G

H

I

P

S

T

Z

www.gl-group.com