Honeywell

Experion PKS
Hardware and Point Build Reference

EPDOC-XX50-en-431A
February 2015

Release 431

Honeywell

Document	Release	Issue	Date
EPDOC-XX50-en-431A	431	0	February 2015

Disclaimer

Contents

About this reference

This reference describes the contents of the *hardware definition* and *point definition* files that are created by Quick Builder when you download a project. It is intended for engineers who want to understand the syntax and structure of the files for troubleshooting purposes.

This reference contains the following chapters:

- Hardware build reference, which describes the syntax of the hardware definition files used in server to define Stations, printers, and controllers (other than Experion Process Controllers).
- Point build reference, which describes the syntax of the point definition files used in the server to define points.
- Point algorithms reference, which describes the syntax of algorithm definitions attached to points.

For details about using Quick Builder, see Quick Builder's help.

Revision history

Revision	Date	Description
A	February 2015	Initial release of document.

Hardware build reference

This chapter describes the syntax of the *hardware definition files* that are created by Quick Builder when you download a project. These files are used by the server to define Stations, printers, and controllers.

(Hardware definition files are also known as *hardware build files*, *hardware configuration files* and *hdwbld files*.)

This chapter also describes how to use the **hdwbld** utility.

CAUTION

If you edit a hardware definition file created by Quick Builder, the Quick Builder project and the hardware definition file will be out of sync. Honeywell recommends that, after updating the file, you import it into Quick Builder to synchronize the project.

Related topics

"About creating a hardware definition file" on page 10
"Station connections" on page 11
"Printer connections" on page 13
"Controller connections" on page 14
"hdwbld" on page 16
"hdwbckbld" on page 17

About creating a hardware definition file

Notes

- You can use any text editor used to create a hardware definition file.
- Hardware definition files can reside in any folder but they are usually placed under *<install folder>\Honeywell\Experion PKS\server\user*.
- You can have either:
  - A single hardware definition file for all devices
  - Separate files for different devices (For example, you might have one file for Stations, one for printers, and one for controllers.)
- The three basic entries for defining a device connection are:
  - A *DEL* entry to delete any existing definition for that device.
  - An *ADD* entry to define the device's communications characteristics.
  - A *DEF* entry to define the device's characteristics.
- For details about the entries required to define specific types of channel and controller, see the associated *Interface Reference*.

Rules and guidelines

- Start all entries in column number 1.
- Separate fields in an entry with one or more spaces (they are not column-sensitive).
- Start comment lines with an “&” (ampersand) character.
- Blank lines are ignored.

Related topics

“Printer connections” on page 13
“Station connections” on page 11
“Controller connections” on page 14

Station connections

The following topics describe the hardware definition entries for:

- Local/Remote LAN connection
- Station options (Also called a *local Station connection*.)

CAUTION

When you delete a Station connection, any user-defined function key definition for that Station is also deleted.

Attention

Serial connections for Stations are no longer supported. If a serial link is required, an RAS service should be set up (see Microsoft Windows documentation) and Stations built to operate as LAN connections.

Related topics

"Local Remote LAN connection" on page 11
"Station options" on page 12
"About creating a hardware definition file" on page 10

Local Remote LAN connection

The following hardware definition file entries define the connection of remote Stations to the server using a LAN. The order of the entries is important.

```
DEL STNnn
ADD STNnn [connection_type]
DEF STNnn [station_options]
```

Part	Description
nn	The Station number from *01* to the maximum number of licensed Stations. Note that 2 digits are required. (*01* is normally used for the server Station).
connection_type	The connection type: • *LAN_ROTARY* Used as a connection type to permit this Station number to be used for rotary connections. The Station can be configured (at the remote Station end) to connect as a rotary Station. In this case, the first available rotary Station number is used for the connection. • *LAN_STATIC* Used as a connection type to permit this Station number to only be used for a static connection. The Station can be configured (at the remote Station end) to connect as a particular Station number. In this case the connection can only be made if the connection has been defined as *LAN_STATIC*.
station_options	See the topic "Station options".

Example

The following example defines Station number 2 as a Station that can accept rotary connections from a remote LAN Station. Defaults are used for all other define Station options.

```
DEL STN02
ADD STN02 LAN_ROTARY
DEF STN02 IPS
```

Station options

The order of the entries is important.

```
DEF STNnn IPS [SIGNON][NAME=n][IDLE=i][UPDATE=u][FAST=f]
```

Part	Description
nn	The Station number from *01* to the maximum number of allowable Stations. Note that 2 digits are required. (*01* is normally used for the server Station.)
SIGNON	Enables operator-based security for the Station. When this is enabled, users require an operator ID and password in order to sign on and use the Station. For more information about operator-based security, see "About Station-based security" in the *Configuration Guide*.
NAME=n	Specifies the name of the Station. The maximum length of the name is 16 characters. Use the underscore character "_" to specify spaces in the name.
IDLE=i	Specifies the page number of the display [*i*] to be called up on the Station when the idle timeout timer for the Station expires. For more information about idle timeouts, see "Server wide settings" in the *Configuration Guide*.
UPDATE=u	Specifies the rate in seconds [*u*] at which the Experion server subscribes to data from devices such as controllers. This parameter setting equates to the Quick Builder **Update Rate** setting on the Main tab for a static Station. Care should be taken when setting the update rate to ensure that you are not placing an unnecessary load on your controllers. Note that: • This update rate does not apply to rotary Stations. You configure update rates for rotary Stations on the Connection tab of the **Connection Properties** dialog box in Station. • Custom displays and individual parameter values on custom displays can be configured to update at a different rate to the rates specified here.
FAST=f	Specifies the rate in seconds [*f*] at which Station subscribes to updates from the Experion server. This parameter setting equates to the Quick Builder **Fast Update Rate** setting on the Main tab for a static Station. Note that: • Dynamic values from controllers and other devices are updated in the server at the fast update rate when the FAST key is pressed on the IKB/OEP keyboard. • This rate does not apply to rotary Stations. You configure update rates for rotary Stations on the Connection tab of the **Connection Properties** dialog box in Station. • Custom displays and individual parameter values on custom displays can be configured to update at a different rate to the rate specified here.

Printer connections

This section shows you how printer connections are defined.

The following hardware definition file entries define the connection of a printer to the server using a serial line. The order of the entries is important.

```
DEL LPTnn
ADD LPTnn VIRTUAL
DEF LPTnn NT NAME=[printer_name]
```

Part	Description
nn	The printer number from *01* to the maximum number of printers that can be enabled. Note that 2 digits are required.
printer_name	The name of the printer, as defined under Windows.

Related topics

"About creating a hardware definition file" on page 10

Controller connections

The following sections describe the basic syntax of channel and controller definition entries.

Details about hardware definition file entries for defining specific channel and controller connections are described in the relevant *Controller Reference.*

Related topics

"Syntax for deleting a channel and a controller" on page 14
"Syntax for adding a channel" on page 14
"Syntax for adding channels with redundant communication links" on page 15
"Syntax for adding controllers" on page 15
"About creating a hardware definition file" on page 10

Syntax for deleting a channel and a controller

To delete a channel or controller, use the following two entries respectively:

```
DEL CHNcc
DEL RTUrrrrr
```

Part	Description
cc	The channel number from *01* up to the maximum allowed on your system. For more information about setting the maximum number of channels, see the topic titled "Adjusting sizing of non-licensed items" in the *Supplementary Installation Tasks Guide*.
rrrrr	The controller number from *00001* up to the maximum allowed on your system. For more information about setting the maximum number of controllers, see the topic titled "Adjusting sizing of non-licensed items" in the *Supplementary Installation Tasks Guide*.

Remarks

- Before a channel can be deleted, all controllers configured on the channel need to be deleted first. A *DEL RTU* entry should always precede a *DEL CHN* entry.
- Before a controller can be deleted, all points configured on the controller need to be deleted.

Syntax for adding a channel

To add a channel the following entries are used:

```
ADD CHNcc ...
DEF CHNcc ...
```

The *ADD CHN* entry defines the details about the physical characteristics of the communications port such as connection type (serial, parallel, LAN), port name, baud, and parity.

The *DEF CHN* entry defines the details about the channel used internally by the server such as channel name and marginal and fail barometer limits.

For example, to define a channel for an LCS 620 controller connected to the first COM port, the following entries would be used:

```
ADD CHN02  SERIAL PORT=COM1 BAUD=19200
DEF CHN02  IPC NAME=plcs MARG=25 FAIL=50
```

Alarm priority for marginal alarms and fail alarms

You can change the priority of marginal alarms and fail alarms, for an individual channel or for all channels system wide.

To change the priority of the alarm for one channel, see the topic titled "About configuring custom system alarm priorities for an individual channel or controller" in the *Server and Client Configuration Guide*.

To change the priority of the alarm for all channels system wide, see the topic titled "Configuring system alarm priorities" in the *Server and Client Configuration Guide*.

Syntax for adding channels with redundant communication links

Some devices support redundant communication links. In this case the following entry is used in addition to an *ADD CHNcc* entry to define the physical characteristics for the second communication link:

```
ADD CHNccD ...
```

For example, to define a redundant communication link to an LCS 620 controller where the two links are connected to the first and second COM ports, the following entries would be used:

```
ADD CHN02 SERIAL PORT=COM1 BAUD=19200
ADD CHN02D SERIAL PORT=COM2 BAUD=19200
DEF CHN02 IPC NAME=plcs MARG=25 FAIL=50
```

For more details about setting up redundant communications links, see the topic "Communications redundancy" in the *Server and Client Configuration Guide*.

Syntax for adding controllers

To add a controller, the following entry is used:

```
DEF RTUcc.rrrrr
```

This entry defines the details about the controller such as controller name, marginal and fail barometer limits for the controller and controller ID (address of the controller on the communications link).

Alarm priority for marginal alarms and fail alarms

You can change the priority of marginal alarms and fail alarms, for an individual controller or for all controller system wide.

To change the priority of the alarm for one controller, see the topic titled "About configuring custom system alarm priorities for an individual channel or controller" in the *Server and Client Configuration Guide*.

To change the priority of the alarm for all controllers system wide, see the topic titled "Configuring system alarm priorities" in the *Server and Client Configuration Guide*.

hdwbld

Description

hdwbld is a command-line utility that can be run while the server software is running. However, any serial ports that are being used must be present in order for **hdwbld** to work. (This is because **hdwbld** checks their existence at build time.)

Syntax

```
hdwbld file_name [-out path_name] [-nl] [-le] [-ns] [-DEL] [-FORCE] [-DIAG]
```

Part	Description
file_name	The name of the hardware definition file. If the full path is not specified, the current folder is assumed.
path_name	The name of the file in which the output is stored. By default the output goes to Command Prompt window. This option is useful when there is too much output to be viewed in the window. After **hdwbld** has finished running, you can view the contents of the output file. If the full path is not specified, the current folder is assumed.
-nl	Suppresses all **hdwbld** output.
-le	Only directs error messages to the output file. (The default is to direct all output to the output file, that is, the hardware definition file lines and also error messages for incorrect lines.)
-ns	Suppresses the building of scanning tables when defining channel and controller connections, which saves execution time. After you are satisfied with the controller and channel definitions, you can re-run **hdwbld** to build scanning tables. (The server performs diagnostic scanning of a controller after the scanning tables are built.)
-DEL	Only executes DEL commands. This option is useful for deleting items.
-FORCE	Forces deletes and adds.
-DIAG	Lists diagnostic messages.

Example

If your hardware definition file, *stn.hdw*, is located in the same folder as the **hdwbld** utility and you want to direct the output to the screen, you would use the following command:

```
hdwbld stn.hdw
```

If your hardware definition file, *stn.hdw*, is located in *c:\<install folder>\Honeywell\Experion PKS\server \user* and you want to direct the output to a file called *hdw.out* (to be created in the same folder as the hardware definition file), you would use the following command:

```
hdwbld c:\<install folder>\Honeywell\Experion PKS\server\user\stn.hdw
-out c:\<install folder>\Honeywell\Experion PKS\server\user\hdw.out
```

hdwbckbld

Description

hdwbckbld is a command-line utility that is used to create a hardware definition file that reflects the current hardware configuration details in the server database. It is used, for example, when you have made changes to hardware and hardware connections via Station, rather than via Quick Builder or **hdwbld**.

After creating a hardware definition file, you can upload it into a Quick Builder project. For details, see the topic titled "Uploading items" in Quick Builder's help.

Syntax

`hdwbckbld [-out` *`file_name`*`] [`*`options`*`]`

Part	Description
file_name	The name of the file to which the hardware definitions are written. By default the output goes to the screen.
options	*-all* Backbuilds all channels, controllers, printers, and stations defined in the server. *-chn chn_num* Backbuilds the specified channel. *-cnt cnt_num* Backbuilds the specified controller. *-prt ptr_num* Backbuilds the specified printer. *-stn stn_num* Backbuilds the specified Station. *-chncnt chn_num* Backbuilds the specified channel and controllers associated with that channel. *-cntchn cnt_num* Backbuilds the specified controller and the channel to which the controller belongs.

Remarks

- To run *hdwbckbld*, the database must be loaded, but the server software does not need to be running.

Point build reference

This chapter describes the syntax of the *point definition files* that are created by Quick Builder when you download a project. These files are used by the server to define points.

(Point definition files are also known as *point build files*, *point configuration files*, and *pntbld files*.)

This chapter also describes how to use the **pntbld** utility.

CAUTION

If you edit a point definition file created by Quick Builder, the Quick Builder project and the point definition file will be out of sync. Honeywell recommends that, after updating the file, you import it into Quick Builder to synchronize the project.

If you add any new points to a point definition file, you need to set the point reference number to `0` for the new points. (This allows **pntbld** to allocate the reference number as described in the "ADD" topic.)

This chapter does not include controller-specific point definition information. For controller-specific details, see the associated *Interface Reference*.

For details of algorithm configuration, see the "Point algorithms reference" section.

Related topics

“About creating a point definition file” on page 20

“Point definition entries reference” on page 21

“pntbld” on page 79

“Point-management utilities” on page 81

“Example point definitions” on page 82

“ADD” on page 26

“Point algorithms reference” on page 85

About creating a point definition file

Notes

- You can use any text editor to create a point definition file.
- Point definition files can reside in any folder but they are usually placed under *\<install folder>\Honeywell\Experion PKS\server\user*.
- You can create either:
  - A single point definition file for all the points in your server database
  - Multiple point definition files (For example, you might have one file for status points, one for analog, and so on.)
- For details about the syntax of **pntbld** entries, see the “Point definition entries reference” topic.

Rules and guidelines

- The following entries are required, in the following order, for each point: DEL and ADD.
- Start all entries in column number 1.
- Separate fields in an entry with one or more spaces (they are not column-sensitive).
- If a space is required in a point ID, use an “_” (underscore)—this is replaced with a space by **pntbld**. (This is only applicable to legacy systems. The current point naming rules do not allow spaces—see the “Naming rules for points” topic.)
- Start comment lines with an “&” (ampersand) character.
- Blank lines are ignored.
- Hex values are represented using `Z'xxxx'` notation, and character values are represented using `A'xx'` notation.

Point definition entries reference

This section describes the entries of a point definition file.

Related topics

"Interpreting pntbld error and warning messages" on page 80

Required entries

This topic summarizes the required entries, in the following order, for each point:

Entry	Description
DEL	Deletes any existing definition for that point
ADD	Adds a new point definition to the database

Related topics

"ADD" on page 26
"DEL" on page 37

General entries

This topic summarizes the general entries for a point definition file.

Entry	Description
&	Comment.
AREA	For backward compatibility only. Assign a point to an asset. See "PARENT" on page 55.
DISPLAY	Associate a display with a point.
END	The last entry.
ENTNAM	The entity name of a point
GROUP	Assign a point to a group.
PARAM	Add a child point to a container point. Add a user-defined parameter to a point.
PARENT	Assign a point to an asset.
PNTDTLPG	Assign a point to a point detail display.
RANGE	Specifies the range of a point (in engineering units)
SUBTYPE	Determines which icon to show in the Location Pane of the System Status Display
TREND	Assign a point to a trend.

Related topics

"AREA" on page 33
"DISPLAY" on page 37
"END" on page 39
"ENTNAM" on page 39
"GROUP" on page 41
"PARAM" on page 53
"PARENT" on page 55

"PNTDTLPG" on page 56
"RANGE" on page 58
"SUBTYPE" on page 63
"TREND" on page 65

Scanning entries

This topic summarizes the scanning entries for a point definition file.

Entry	Description
DRIFTDB	Drift deadband (analog point only).
MDREVERS	Reverses the mode parameter value for the MAN state. Applies to analog points and status points only. Do not enable MD reverse on points connected to Bristol Babcock controllers and Bristol Babcock OpenBSI controllers.
METER	Meter Factor (accumulator point only).
ONSCAN	Build the point "off" scan (not normally used).
OPREVERS	Reverses an entire OP parameter or individual bits of an OP parameter. Applies to status points only.
OPWIDTH	Number of output bits for a status point.
PVCLAMP	Clamp the PV of an analog point to 0% and 100%.
PVREVERS	Reverses an entire PV parameter or individual bits of a PV parameter. Applies to status points only.
ROLOVR	Counter rollover value for an accumulator point.
SCALE	Scale Factor for an accumulator point.
STATEDES	Names (*State Descriptors*) for status point states.
xxDESTIN	Destination addresses for I/O point parameters.
xxDYNSCN	Enables and disables dynamic scanning for the point parameter.
xxNAME	Associate descriptors with analog point parameters.
xxPERIOD	Scan period for I/O point parameters.
xxSOURCE	Source addresses for I/O point parameters.

Related topics

"DRIFTDB" on page 38
"METER" on page 50
"MDREVERS" on page 49
"ONSCAN" on page 50
"OPREVERS" on page 52
"OPWIDTH" on page 53
"PVCLAMP" on page 57
"PVREVERS" on page 58
"ROLOVR" on page 60
"SCALE" on page 60
"STATEDES" on page 62
"xxDESTIN" on page 65
"xxNAME" on page 67
"xxPERIOD" on page 69
"xxSOURCE" on page 70

“xxDYNSCN” on page 66

Alarm entries

This topic summarizes the alarm entries for a point definition file.

Entry	Description
AKDESTIN	Alarm point parameter for acknowledging alarms.
ALARM	Alarm priority and alarm states for status points. Also the unreasonable value analog point priority unreasonable value analog point priority.
ALARMDB	Alarm deadband for analog points.
ALMLIMn	Alarm types, priorities and limits for analog and accumulator points.
ALMMSG	Message text index.
ALMXCHG	External change alarm for analog and status points.
JNLONLY	Journal Only for alarms on status, analog, and accumulator points.

Related topics

“AKDESTIN” on page 26
“ALARM” on page 27
“ALARMDB” on page 29
“ALMLIMn” on page 30
“ALMMSG” on page 32
“ALMXCHG” on page 32
“JNLONLY” on page 47

Control entries

This topic summarizes the control entries for a point definition file.

Entry	Description
CCONFIRM	Control confirmation for control actions.
CNTINH	Control inhibit for status and analog points.
CNTRLDB	Control deadband for analog points.
CNTRLLVL	Control level for status and analog points.
CNTRLTO	Control timeout for analog points.
ESIGPRIM	Specifies that a primary electronic signature is required for point control for status and analog points.
ESIGSEC	Specifies that a secondary electronic signature is required for point control for status and analog points.
LPPERIOD	Assigns the scan period to loop point parameters PV, MD, OP and SP.
LPSOURCE	Specifies the source and destination addresses for loop point parameters PV, MD, OP, and SP.
MDDISABL	Determine whether mode control is enabled or disabled.
MDNORMAL	Normal mode for status and analog points.
OPLIMIT	Output limit for analog points.
OPPULSE	Output pulse width for status points.

Entry	Description
REVERSE	Reverse output indication for analog and status points.
SCRIPT	Assigns a server script to a point/point parameter.
SPLIMIT	Set point limit for analog points.
TARGET	Target input states for status point outputs.
TIMESYNC	Specifies that a point is the time synchronization point for a TDC 3000 DHP.

Related topics

"CCONFIRM" on page 34
"CNTINH" on page 34
"CNTRLDB" on page 34
"CNTRLLVL" on page 35
"CNTRLTO" on page 36
"ESIGPRIM" on page 39
"ESIGSEC" on page 40
"LPPERIOD" on page 47
"LPSOURCE" on page 48
"MDDISABL" on page 48
"MDNORMAL" on page 49
"OPLIMIT" on page 51
"OPPULSE" on page 51
"REVERSE" on page 59
"SCRIPT" on page 60
"SPLIMIT" on page 62
"TARGET" on page 63
"TIMESYNC" on page 64

History entries

This topic summarizes the history entries for a point definition file.

Entry	Description
HISDEL	Stop history collection for a point.
HISGATE	Control history collection using another status point (only included for backward compatibility).
HISTEXCP	Assign a point parameter for exception history collection.
HISTEXTD	Assign a point parameter for extended history collection.
HISTFAST	Assign a point parameter for fast history collection.
HISTORY	Assign a point's PV for history collection (only included for backward compatibility).
HISTSLOW	Assign a point parameter for standard history collection.

Related topics

"HISDEL" on page 41
"HISGATE" on page 42
"HISTEXTD" on page 43
"HISTFAST" on page 44
"HISTORY" on page 45

"HISTSLOW" on page 46

Algorithm entries

This topic summarizes the algorithm entries for a point definition file.

Entry	Description
ACTALGO	Attach an Action algorithm to the point.
ALG(xx)	Define the details of an algorithm block.
PVALGO	Attach a PV algorithm to the point.

Related topics

"ACTALGO" on page 25
"ALG(xx)" on page 29
"PVALGO" on page 56

ACTALGO

Description

Attaches an action algorithm to the point and assigns an algorithm data block.

Syntax

```
ACTALGO Point_ID Num Block
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Num	The number of the action algorithm. Must be a valid algorithm number.
Block	The number of the algorithm data block used by this algorithm for this point. You must specify a different block for each algorithm/point combination. (Use the *alglst* utility to find a free block. For details, see the *Server and Client Configuration Guide*.) A warning message is issued by **pntbld** if a block is used by more than one algorithm/point combination.

Remarks

- Action algorithms run only when there is a change in the PV.
- See the topic "Configuring action algorithms using pntbld" for a description of each algorithm.
- *Action Algo 11 Composite Alarm* requires the multiple usage of algorithm blocks. In this case, the warning should be ignored.
- When used with analog points, the DRIFTDB must be greater than zero on the point with the algorithm attached.
- Some algorithms do not require a block, and therefore 0 (zero) is used.

Related topics

"Algorithm entries" on page 25
"Configuring action algorithms using pntbld" on page 106
"DRIFTDB" on page 38
"PV Algo 12: Composite Alarm Processing" on page 93

ADD

Description

Adds a new point to the database.

Syntax

ADD *Point_ID Point_No Description*

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points."
Point_No	This consists of the point *Type* and *Ref*. *Type*, which is one of the following point types: • *ANA* (analog point) • *STA* (status point) • *ACC* (accumulator point) • *PSA* (flexible point) This field defines the point type. Certain entries following this one become type-specific. *Ref*: You can let **pntbld** automatically allocate a reference number, by specifying *0* as the reference number. Otherwise, you must allocate a unique reference number as follows. The reference number, which is between 1 and 65000, must be unique. That is, the same number cannot be used for two different point types. The number of points can you build depends on how many you are licensed to have.
Description	This consists of up to 132 alphanumeric characters. This is shown in full at the top of the Point Detail display. You can use uppercase and lowercase characters.

Related topics

"Point build reference" on page 19
"Required entries" on page 21
"Naming rules for points" on page 72

AKDESTIN

Description

Provides an alarm acknowledge parameter for a status point without having to build a separate point for acknowledging alarms. This parameter is used by an operator to acknowledge a critical alarm so that it can be reset in the controller.

Syntax

AKDESTIN *Point_ID RTU Address*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.

Part	Description
RTU	The controller number of the output device as configured in the hardware definition file.
Address	The address within the controller. (The same syntax as for the *xxSOURCE* entry.)

Example

This example shows how AKDESTIN is typically used. The controller is programmed to have:

- A field value
- An alarm bit that is latched on a rising edge of the field value as shown below:

```
     Field              Alarm
-----[/\]----------------(L)
```

The alarm bit is unlatched if either of two conditions is satisfied. First, the alarm bit is unlatched if a rising edge of the alarm acknowledge bit that is also called the Ack bit occurs after the field value has returned to normal. Second, it will be unlatched if a lowering edge of the field value occurs while the Ack bit is set:

```
 Ack Field  Alarm
--+--[/\]----[/]----+----------(U)---
|           |
| Field Ack |
--+--[\/]----[]-----+
```

Related topics

“Alarm entries” on page 23
“Naming rules for points” on page 72

ALARM

Description

Used to:

- Set the alarm priority and sub-priority level for each state of a status point.
- Define the alarm states.
- Specify unreasonable value alarm priorities and sub-priorities for analog points.

To define limit alarms for analog and accumulator points, use the ALMLIMn entry.

Syntax

ALARM *Point_ID P*[.*S*] *C*[.*S*] *T SSSSSSSS* [*p*[.*s*] *p*[.*s*]....]

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.

Part	Description
P[.*S*]	The alarm priority level and optional sub-priority for status points, and unreasonable values for analog points. This parameter defines the importance of the alarm: the default is 3 (urgent). The available priorities are: *0* - A journal event. Journal events are entered into the alarm/event file and optionally printed on the alarm/event printer. *1* - A low priority alarm. Low priority alarms are entered into the alarm/event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the alarm list and the audible alarm may be activated. *2* - A high priority alarm. High priority alarms are entered into the alarm/ event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the high alarm list and the audible alarm may be activated. *3* - An urgent alarm. Urgent alarms are entered into the alarm/event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the urgent alarm list and the audible alarm may be activated. The allowable values for alarm sub-priorities range from 0 (lowest, default) to 15 (highest). For example, an entry of 3.12 would set an urgent alarm priority with sub-priority 12. (An alarm subpriority of 0 is used whenever a subpriority field is omitted)
C[.*S*]	The control alarm priority level and optional sub-priority. This defines the importance of the alarm which occurs when a controlled point fails to reach a requested state within a defined period of time. The same level assignment is used here as in alarm priority levels. The default is urgent, sub-priority 0.
T	This enables status point alarm transition (re-alarming): *Y* = allows alarming on transition between alarm states *N* = prevents re-alarming
SSSSSSSS	This specifies a status point alarm mask. S is either Y or N depending on whether the state is an alarm state. State 0 is represented by the left-most character and State 7 is the right-most character: Example: *YNNNNNNN* is for single-bit point alarming in State 0 only *YNYNNNNN* is for dual-bit point alarming in States 0 and 2 *NNNNNNNY* is for three-bit point alarming in State 7 only
p[.*s*]	This is used to specify up to eight alarm priorities and sub-priorities for the 8 input states. The default for all states is the P.S entry immediately following `Point_ID`. Priorities are 0 to 3, as above and sub-priorities are 0 to 15.

Remarks

- If no alarm states are specified for status points, the event file entry and the printed line will indicate CHANGE (status change) rather than ALARM or NORMAL.
- The audible alarm can be configured for each priority level (1 to 3) on Station-by-Station basis, by selecting the **Alarm Audible** check box for that level on the Station Configuration display.
- If no *ALARM* entry is used, then *P* defaults to 0 (alarm priority defaults to journal) and C defaults to 3 (control alarm priority defaults to urgent).
- For analog points, T SSSSSSSS and trailing `p.s` entries are not processed.
- P, C, and T fields should not be left blank. Use *x* for default values.
- An alarm subpriority of 0 is used whenever a subpriority field is omitted.

Related topics

“Alarm entries” on page 23
“ALMLIMn” on page 30
“Naming rules for points” on page 72

ALARMDB

Description

Specifies how far inside the alarm limit the PV must be before the alarm condition will be cleared. This deadband is to reduce the effect of a point bouncing in and out of alarm due to a small change in its input value.

Syntax

```
ALARMDB Point_ID DB
```

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
DB	Analog alarm deadband index (0-15). Index deadbands are defined as: 0 = 0.000% 1 = 0.001% 2 = 0.002% 3 = 0.005% 4 = 0.010% 5 = 0.020% 6 = 0.050% 7 = 0.100% 8 = 0.200% 9 = 0.500% 10 = 1.000% (the default) 11 = 2.000% 12 = 5.000% 13 = 10.000% 14 = 20.000% 15 = 50.000%

Related topics

“Alarm entries” on page 23
“Naming rules for points” on page 72

ALG(xx)

Description

Allows various words within the point algorithm data block to be assigned values or addresses as required.

Syntax

```
ALG(xx) Point_ID Pntname Parname
ALG(xx) Point_ID Pntname f word bit width
ALG(xx) Point_ID Value
```

Part	Description
xx	Algorithm data block word number (01 - 20).
Point_ID	The point's name. See the topic “Naming rules for points.”
Pntname	The name of the point being referenced.
Parname	The name of the point parameter being referenced.
f	Point file type 0-DAT, 1-EXT, 2-CNT, 3-DES. The *f word bit width* syntax allows any bit field to be read from the point record (even if it is not defined as a standard parameter, such as PV). Each point record is made up of information from four separate database files.
word	Word in point record to be referenced.
bit	Bit in word to be referenced (0-15).
width	Width of field -1 (0-15).
Value	Integer, real, hex, or character value. Hex is represented by Z'xxxx'. Character is represented by A'xx'.

Remarks

- When *pntname* is specified, data block word ALG(xx+1) will be used to store the parameter information. Items *f*, *word*, *bit*, and *width* are free-form and require a space as a field separator.
- Definition of the data within the algo block must follow the definition of the algo itself. A *PVALGO* entry, for example, must be followed by a number of *ALG(xx)* entries (if applicable) before definition of an *ACTALGO*. In this way data within the algo block is linked to the correct algo.
- The record layout of these files are described in *dat000_def*, *ext000_def*, *cnt000_def,* and *des000_def* which are in *\<install folder>\Honeywell\Experion PKS\server\def\src*.

Related topics

“Algorithm entries” on page 25
“Naming rules for points” on page 72
“PVALGO” on page 56

ALMLIMn

Description

Allows alarm types, individual alarm priorities and corresponding alarm limits to be set for analog and accumulator points. Alarm sub-priorities may also be set if desired.

Syntax

```
ALMLIMn Point_ID Limit T P[.S]
```

Part	Description
n	The alarm limit number: With analog points you can have: 1 to 8 alarm limits With accumulator points you can have 1 to 4 alarm limits
Point_ID	The point's name. See the topic "Naming rules for points."
Limit	Alarm limit value of the analog or accumulator point.
T	Alarm type: 0 - no alarm type 1 - rate of change (engineering units/sec) 2 - deviation low (engineering units) 3 - deviation high (engineering units) 4 - transmitter low (engineering units) 5 - transmitter high (engineering units) 6 - PV low (engineering units) 7 - PV high (engineering units) 8 - PV low low (engineering units) 9 - PV high high (engineering units)
P[.S]	The alarm priority level and optional sub-priority for status points and unreasonable values for analog points. This parameter defines the importance of the alarm: the default is *URGENT*. The available priorities are: *X* - Inhibits alarms *0* - A journal event. Journal events are entered into the alarm/event file and optionally printed on the alarm/event printer. *1* - A low priority alarm. Low priority alarms are entered into the alarm/event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the alarm list and the audible alarm may be activated. *2* - A high priority alarm. High priority alarms are entered into the alarm/event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the urgent and high alarm list and the audible alarm may be activated. *3* - An urgent alarm. Urgent alarms are entered into the alarm/event file and optionally printed on the alarm/event printer. In addition, the alarm is placed in the Urgent alarm list and the audible alarm may be activated. The allowable values for alarm sub-priorities range from 0 (lowest, default) to 15 (highest). For example, an entry of 3.12 would set an urgent alarm priority with sub-priority 12. If you do not specify a sub-priority, the sub-priority defaults to 0, the lowest sub-priority.

Remarks

- The first low alarm limit and the first high alarm limit are used for alarm limit red shading on analog PV Bar/Pointer and Trend displays.
- For accumulator points, the only valid alarm types are 1, 7, and 9.
- A system-wide reasonable high and low alarm limit for analog points is also available on the Point System-Wide Configuration display. The priority is specified by the *ALARM* entry.
- The audible alarm may be configured for each priority level (1 to 3) on a Station-by-Station basis, by selecting the **Alarm Audible** check box for that level on the Station Configuration display.

- `Limit`, `T`, and `P` must not be blank.
- An accumulator point can only be configured to have PV High, PV High High, Rate of Change.

Related topics

“Alarm entries” on page 23
“ALARM” on page 27
“Naming rules for points” on page 72

ALMMSG

Description

Identifies the index number of the message associated with the point. When the point goes into alarm the message appears in the Message Summary display, and is also printed with the point alarm. For details about defining a message, see the *Configuration Guide*.

Syntax

```
ALMMSG Point_ID xxxx
```

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
xxxx	The Message Index 1 to 1000 (or 0 for no message). The default is 0.

Related topics

“Alarm entries” on page 23
“Naming rules for points” on page 72

ALMXCHG

Description

Allows alarming when a point parameter value changes in the controller, and that change in value was not requested from the server.

Syntax

```
ALMXCHG Point_ID ABCDEFGH
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
A	Alarm external PV changes (Y/N)
B	Alarm external MD changes (Y/N)
C	Alarm external OP changes (Y/N)
D	Alarm external SP changes (Y/N)
E	Alarm external A1 changes (Y/N)
F	Alarm external A2 changes (Y/N)
G	Alarm external A3 changes (Y/N)

Part	Description
H	Alarm external A4 changes (Y/N)

Remarks

- The default is no external change alarming.
- Typically, PV and OP values change frequently. It is unwise to create external change alarms on these parameters because they would result in a continuous flood of alarms.

  In comparison, MD and SP values typically change infrequently. Most operators would like to know when these values have been changed by another person external to the Experion server; for example, by a technician at a remote control panel. The ALMXCHG parameter does this. To configure ALMXCHG to raise an alarm on MD and SP parameters only you would set *ABCD* = NYNY.

  `ALMXCHG` *Point_IDNYNY*.
- For a pump, all changes of state are generally initiated from the server. In that case you would set *A* = Y.

  `ALMXCHG` *Point_IDY*.

Related topics

"Alarm entries" on page 23
"Naming rules for points" on page 72

AREA

Description

Defines the asset for the point. If no *AREA* entry is specified, the asset is either taken from the first two characters of the point ID or the point becomes an unassigned item. If the first two characters of the point ID do not match an asset, the point becomes an unassigned item.

Syntax

`AREA` *Point_ID Area*

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points."
Area	The tag name or full item name of the asset to which the point belongs. See the topic "Naming rules for assets within an Asset Model."

Remarks

- The AREA entry is supported for backward compatibility only. It is replaced by the PARENT entry.

Related topics

"General entries" on page 21
"Naming rules for assets within an Asset Model" on page 75
"Naming rules for points" on page 72
"PARENT" on page 55

CCONFIRM

Description

Allows points to be built that require control confirmation. When a control action is performed on a point with control confirmation the prompt, *Please confirm control request*, is displayed. The operator must press Y to confirm the control action.

Syntax

`CCONFIRM Point_ID A`

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
A	Use: *Y* - to build the point with control confirmation *N* - to build the point without control confirmation (This is the default.)

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

CNTINH

Description

Inhibits control of the point. Operators can view, but not control, the point.

Syntax

`CNTINH Point_ID Y/N`

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Y/N	Y = Control inhibit is enabled. Operators cannot control the point. N = Control inhibit is disabled. Operators can control the point.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

CNTRLDB

Description

Lets you specify, for analog points, what constitutes a good control. If the PV signal, read back after an SP control is issued, does not reach the {new SP value + this deadband} within the control timeout period, then this results in a control fail alarm being generated.

Syntax

`CNTRLDB Point_ID DB`

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
DB	Control deadband index (0–15). The options are: 0 = 0.000% 1 = 0.001% 2 = 0.002% 3 = 0.005% 4 = 0.010% 5 = 0.020% 6 = 0.050% 7 = 0.100% 8 = 0.200% 9 = 0.500% 10 = 1.000% (the default) 11 = 2.000% 12 = 5.000% 13 = 10.000% 14 = 20.000% 15 = 50.000%

Remarks

- This check is performed every 10 seconds until good control has been achieved or the control timeout has elapsed; whichever happens first.
- This deadband is also used during analog SP and OP read after write confirmation, to avoid errors in precision. This will occur if there is a SP or OP source and destination address.
- See also *CNTRLTO*.
- Percentages refer to percentage of full range (see *RANGE*).

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72
“CNTRLTO” on page 36
“RANGE” on page 58

CNTRLLVL

Description

Specifies the control security level for analog and status points. This command is applicable only when operator-based security is enabled. For details, see the topic “Control level” in the Configuration Guide.

Syntax

`CNTRLLVL` *`Point_ID xxx`*

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
xxx	The control level for controllable analog and status points. Choose a number between 0 and 255. Operators are allowed to control the point only if they have a control level higher or equal to the point's control level. The default is *0*.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

CNTRLTO

Description

Allows specification of the Control timeout. This timeout is used in the determination of Command Fail alarms.

Syntax

CNTRLTO *Point_ID Timeout*

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
Timeout	Control timeout period index (0-15). Specifies the maximum allowable time for the PV of a point to reach the required State without alarming. 0 = none 1 = 2 s 2 = 5 s 3 = 10 s 4 = 20 s 5 = 30 s 6 = 40 s 7 = 50 s 8 = 60 s 9 = 90 s 10 = 120 s 11 = 180 s 12 = 240 s 13 = 300 s 14 = 600 s 15 = 1200 s

Remarks

- Default is 0.

- Specification of a timeout implies that there is PV confirmation on status OP controls and analog SP controls.
- For status points the PV must reach the *TARGET* state for the OP action within the timeout period. For analog points the PV must reach a value within the control deadband of the new SP, and within the timeout period.
- Also see *CNTRLDB* and *TARGET*.

Related topics

"Control entries" on page 23
"CNTRLDB" on page 34
"Naming rules for points" on page 72
"TARGET" on page 63

DEL

Description

Deletes a point from the database, thus releasing its reference number.

Syntax

```
DEL Point_ID
```

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points."

Remarks

- DEL does not delete any reference to this point in algorithms, displays, or reports.

Related topics

"Required entries" on page 21
"Naming rules for points" on page 72

DISPLAY

Description

Associates a particular display to a point. From any display, selecting the point, then pressing the ASSOC DISP function key, or clicking the **Associated Page** button on the toolbar, will force the configured associated display to be called up at that Station.

If there is no point selected, pressing the ASSOC DISP function key will force the associated display configured for the point in the Alarm Zone of the Station to be called up.

Syntax

```
DISPLAY Point_ID PageName
```

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points."
PageName	The number or name of the display to be used as the Associated Display.

Related topics

"General entries" on page 21
"Naming rules for points" on page 72

DRIFTDB

Description

Allows specification of what constitutes a significant change of the Input analog parameter signal. If the change is less than this drift amount, the new value will not be processed.

Syntax

```
DRIFTDB Point_ID DB
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
DB	Analog drift deadband index: 0 = 0.000% 1 = 0.001% 2 = 0.002% 3 = 0.005% 4 = 0.010% 5 = 0.020% 6 = 0.050% 7 = 0.100% 8 = 0.200% 9 = 0.500% 10 = 1.000% (the default) 11 = 2.000% 12 = 5.000% 13 = 10.000% 14 = 20.000% 15 = 50.000%

Remarks

- Percentages refer to percentage of full range (see RANGE).
- If the value of a hardware source address has changed, a PV algorithm will be executed regardless of the DRIFTDB value.
- An action algorithm will run only if the DRIFTDB value is exceeded.
- DRIFTDB applies to all analog parameters including auxiliary.

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72
"RANGE" on page 58

“ACTALGO” on page 25

END

Description

Terminates **pntbld** at the completion of a series of other entries.

Syntax

```
END
```

Remarks

- If this command is not specified, **pntbld** will terminate at the end of the point definition file.

Related topics

“General entries” on page 21

ENTNAM

Description

Defines the item name for the point.

Syntax

`ENTNAM` *`Point_ID ItemName`*

Part	Description
Point_ID	The point's name. See the topic ”Naming rules for points.”
ItemName	An intuitive name given to a point, which can be used as an alternative to the point ID. The ItemName is used within the full item name of a point, which shows the point's location with the location hierarchy.

Related topics

“General entries” on page 21
“Naming rules for points” on page 72

ESIGPRIM

Description

Allows points to be built that require an electronic signature. When a control action is performed on a point with electronic signatures, the Electronic Signatures dialog box is displayed. The operator must provide an authorized user ID and password to be able to control the point.

Syntax

`ESIGPRIM` *`Point_ID x Meaning`*

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
x	The number of the reason set assigned to the point. (The operator must select the appropriate reason from this set when controlling the point.)
Meaning	An optional property that describes the meaning of entering the primary signature, for example: 'Issued', 'Implemented'. If used, it should be an approved term for your industry/work practices. Consists of 24 characters. This is displayed in the Electronic Signature dialog box.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

ESIGSEC

Description

Allows points to be built that require a second electronic signature. When a control action is performed on a point with electronic signatures, the Electronic Signatures dialog box is displayed. The operator must provide an authorized user ID and password and a second operator must also provide an authorized user ID and password.

Syntax

```
ESIGSEC Point_ID x Meaning
```

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
x	The security level of the second signer. 2 = Oper 3 = Supv 4 = Engr 5 = Mngr
Meaning	An optional property that describes the meaning of entering the secondary signature, for example: 'Confirmed', 'Authorized'. If used, it should be an approved term for your industry/work practices. Consists of 24 characters. This is displayed in the Electronic Signature dialog box.

Remarks

- If you have an ESIGSEC entry you must also have an ESIGPRIM entry.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

GROUP

Description

Assigns a point to a group. Usually points are assigned using the Group Configuration Display. For details, see the “Groups and trends” topic in the Configuration Guide.

Syntax

`GROUP` *`Point_ID GGGG P`* `[`*`Param`*`]`

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
GGGG	Group number
P	Position in the group, 1 to 8
Param	The parameter to view on the Group Trend display and the Group Numeric History display. If this is left blank, the PV is assumed.

Related topics

“General entries” on page 21
“Naming rules for points” on page 72

HISDEL

This entry is only included for backward compatibility. See the HISTFAST, HISTSLOW, HISTEXTD, and HISTEXCP entries.

Description

Stops history collection for previously selected HISTFAST, HISTSLOW, HISTEXTD and HISTEXCP points.

> **Attention**
> HISDEL will only delete SCADA PV points and not CDA type points. Those have to be deleted with the HISTFAST / del, HISTSLOW /del, HISTEXTD /del, or HISTEXCP options instead.

Syntax

`HISDEL` *`Point_ID Type Type Type`*

Part	Description
Point_ID	The point's name. See the topic “Naming rules for points.”
Type	The options are: *FAST* = Fast history *SLOW* = Standard history *EXTD* = Extended history *EXCP* = Exception history Up to three assignments may be made.

Related topics

“History entries” on page 24

“HISTFAST” on page 44
“HISTSLOW” on page 46
“HISTEXTD” on page 43

HISGATE

This entry is only included for backward compatibility. See the HISTFAST, HISTSLOW, HISTEXTD, and HISTEXCP entries.

Description

Controls history collection of an analog or status point, based on the state of a “gate point.” (History is only collected when the gate point—which must be a status point—is in the nominated state.)

For example, you may want to stop keeping history of motor current if the motor has stopped.

Syntax

```
HISGATE Point_ID Gate_ID State
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
Gate_ID	The point ID of the gate point.
State	The state (0-7) of the gate point that allows history to be collected.

Related topics

“History entries” on page 24
“HISTFAST” on page 44
“HISTSLOW” on page 46
“HISTEXTD” on page 43
“Naming rules for points” on page 72

HISTEXCP

Description

Enables exception history collection for a point parameter. The collection of history can optionally be controlled by a “gate point.” (History is only collected when the gate point—which must be a status point—is in the nominated state.) In addition, you can specify to send history to the PHD server.

Syntax

```
HISTEXCP Point_ID parameter [gate_Point gate_param gate_state] [/del] [/group=n] [/period=n] [/
PHDCOLLECT=Value]
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
parameter	The point parameter for which history is to be collected. Note that only input/output parameters can be trended. For details of available fixed point parameters for SCADA points, see the "Summary of internal point parameters" topic in the *Server and Client Configuration Guide*.

Part	Description
gate_Point	The point ID of the gate point.
gate_param	The gate point parameter that controls history gating.
gate_state	The state (0–7) of the gate point parameter that allows history to be collected.
/del	Removes the history assignment.
/group	The offset group to which this point is to be assigned.
/period	The collection rate (in seconds).
/PHDCOLLECT	Exception history is collected to the PHD server, according to the *PHD collection rule*, where *Value* is: • *DEFAULT* • *DISABLE* **Attention** • See the topic titled “PHD collection rule” to determine the collection of history to the PHD server. • Having no */PHDCOLLECT* switch has the same effect as */PHDCOLLECT=DEFAULT*.

Remarks

- Point parameters cannot be assigned to both exception and fast/standard/extended history.
- Exception history collection is only available for parameters that have string values.

Related topics

“PHD collection rule” on page 76

HISTEXTD

Description

Enables extended history collection for a point parameter. The collection of history can optionally be controlled by a “gate point.” (History is only collected when the gate point—which must be a status point—is in the nominated state.) In addition, you can specify to send history to the PHD server.

Syntax

```
HISTEXTD Point_ID parameter [gate_Point gate_param gate_state] [/del] [/PHDCOLLECT=Value]
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
parameter	The point parameter for which history is to be collected. Note that only input/output parameters can be trended. For details of available fixed point parameters for SCADA points, see the "Summary of internal point parameters" topic in the *Server and Client Configuration Guide*.
gate_Point	The point ID of the gate point.
gate_param	The gate point parameter that controls history gating.
gate_state	The state (0–7) of the gate point parameter that allows history to be collected.
/del	Removes the history assignment.

<table>
<tr><th>Part</th><th>Description</th></tr>
<tr><td>/PHDCOLLECT</td><td>Extended history is collected to the PHD server, according to the PHD collection rule, where Value is:
• DEFAULT
• OVERRIDE
• DISABLE
Attention
• See the topic titled “PHD collection rule” to determine the collection of history to the PHD server.
• Having no /PHDCOLLECT switch has the same effect as /PHDCOLLECT=DEFAULT.</td></tr>
</table>

Remarks

- Point parameters cannot be assigned to both exception and fast/standard/extended history.
- History collection is only available for parameters that have numeric or time values.

Related topics

“History entries” on page 24
“HISGATE” on page 42
“HISTFAST” on page 44
“HISTSLOW” on page 46
“Naming rules for points” on page 72
“HISTORY” on page 45
“HISDEL” on page 41
“PHD collection rule” on page 76

HISTFAST

Description

Enables fast history collection for a point parameter. The collection of history can optionally be controlled by a “gate point.” (History is only collected when the gate point—which must be a status point—is in the nominated state.) In addition, you can specify to send history to the PHD server.

Syntax

```
HISTFAST Point_ID parameter [gate_Point gate_param gate_state ] [/del] [/period=n] [/
PHDCOLLECT=Value]
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
parameter	The point parameter for which history is to be collected. Note that only input/output parameters can be trended. For details of available fixed point parameters for SCADA points, see the "Summary of internal point parameters" topic in the *Server and Client Configuration Guide*.
gate_Point	Point ID of the gate point.
gate_param	The gate point parameter that controls history gating.
gate_state	State (0–7) of the gate point parameter that allows history to be collected.

Part	Description
/del	Removes the history assignment.
/period	The collection rate (in seconds).
/PHDCOLLECT	Fast history is collected to the PHD server, according to the *PHD collection rule*, where *Value* is: • *DEFAULT* • *OVERRIDE* • *DISABLE* **Attention** • See the topic titled “PHD collection rule” to determine the collection of history to the PHD server. • Having no */PHDCOLLECT* switch has the same effect as */PHDCOLLECT=DEFAULT*.

Remarks

- Point parameters cannot be assigned to both exception and fast/standard/extended history.
- History collection is only available for parameters that have numeric or time values.

Related topics

“History entries” on page 24
“HISDEL” on page 41
“HISGATE” on page 42
“HISTSLOW” on page 46
“HISTEXTD” on page 43
“Naming rules for points” on page 72
“PHD collection rule” on page 76

HISTORY

This entry is only included for backward compatibility. See the HISTFAST, HISTSLOW, HISTEXTD and HISTEXCP entries.

Description

The *HISTORY* entry assigns the specified point's PV to be selected for *SLOW*, *FAST*, or *EXTD* history collection.

Syntax

```
HISTORY Point_ID Type Type Type
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
Type	Selection type: *SLOW* = Standard history *EXTD* = Extended history *FAST* = Fast history *EXCP* = Exception history Up to three assignments may be made.

Remarks

- History may be collected for the PV of analog or status points. When collected for status points, the values will be:

  0 or 1 for single-bit

  0, 1, 2 or 3 for dual-bit

  0, 1, 2, 3, 4, 5, 6 or 7 for 3-bit

Related topics

"History entries" on page 24
"Naming rules for points" on page 72
"HISTEXTD" on page 43
"HISTSLOW" on page 46

HISTSLOW

Description

Enables standard history collection for a point parameter. The collection of history can optionally be controlled by a "gate point." (History is only collected when the gate point—which must be a status point—is in the nominated state.) In addition, you can specify to send history to the PHD server.

Syntax

```
HISTSLOW Point_ID parameter [gate_Point gate_param gate_state] [/del] [/group=n] [/period=n] [/
PHDCOLLECT=Value]
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
parameter	The point parameter for which history is to be collected. Note that only input/output parameters can be trended. For details of available fixed point parameters for SCADA points, see the "Summary of internal point parameters" topic in the *Server and Client Configuration Guide*.
gate_Point	Point ID of the gate point.
gate_param	The gate point parameter that controls history gating.
gate_state	The state (0–7) of the gate point parameter that allows history to be collected.
/del	Removes the history assignment.
/group	The offset group to which this point is to be assigned.
/period	The collection rate (in seconds).
/PHDCOLLECT	Standard history is collected to the PHD server, according to the *PHD collection rule*, where *Value* is: • *DEFAULT* • *OVERRIDE* • *DISABLE* **Attention** • See the topic titled "PHD collection rule" to determine the collection of history to the PHD server. • Having no */PHDCOLLECT* switch has the same effect as */PHDCOLLECT=DEFAULT*.

Remarks

- Point parameters cannot be assigned to both exception and fast/standard/extended history.
- History collection is only available for parameters that have numeric or time values.

Related topics

"History entries" on page 24
"HISGATE" on page 42
"HISTFAST" on page 44
"HISTEXTD" on page 43
"Naming rules for points" on page 72
"HISTORY" on page 45
"HISDEL" on page 41
"PHD collection rule" on page 76

JNLONLY

Description

Journal Only. When enabled, handles all alarms for the point as journaled events; alarms do not appear in the alarm summary. Note that alarms must be enabled for this option to work.

Syntax

JNLONLY *Point_ID Y/N*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Y/N	Y = Journal only is enabled. Alarms are handled as journaled events. N = Journal only is disabled. Alarms appear in the alarm summary..

Related topics

"Alarm entries" on page 23
"Naming rules for points" on page 72

LPPERIOD

Description

Assigns the scan period to loop point parameters PV, MD, OP and SP.

Syntax

LPPERIOD *Point_ID I*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
I	Scan period in seconds. See the xxPERIOD entry for valid scan periods. Default is none. It should be less than 60 seconds.

Remarks

- This entry is used for TDC 3000 modulating slots only. See the Release 525 *TDC 3000 Data Hiway Reference Module* for details of limitations concerning this entry.

Related topics

"Control entries" on page 23
"Naming rules for points" on page 72
"xxPERIOD" on page 69

LPSOURCE

Description

Specifies the source and destination addresses for loop point parameters PV, MD, OP, and SP.

Syntax

```
LPSOURCE Point_ID RTU Address
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
RTU	Controller number of the device as configured by the hdwbld utility.
Address	Loop address of the TDC 3000 Modulating Slot.

Remarks

- This entry is used for TDC 3000 Modulating Slots only. For example, for MC Modulating Slot 3:

  ```
  LPSOURCE ---------- --- LP03
  ```
- See the Release 525 *TDC 3000 Data Hiway Reference Module* for details of limitations concerning this entry.

Related topics

"Control entries" on page 23
"Naming rules for points" on page 72

MDDISABL

CAUTION
This point definition entry should not be used unless the operator understands the behavior of the receiving controller. Performing OP changes to control loops in AUTO, or SP changes to loops in CASCADE, can cause unexpected results.

Description

Specifies whether mode control for a point is enabled or disabled. If the mode check is disabled (MDDISABL=Y) when the operator issues a control to the OP or SP, Experion ignores the current value of the mode of the point.

Syntax

```
MDDISABL Point_ID A
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
A	Is the disable indicator. Y= mode control is off (mode is ignored) N = mode control is on

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

MDNORMAL

Description

Allows specification of the normal mode of the point. By selecting a point, from any display, then pressing the NORM function key (followed by the ENTER key), the mode will be forced to change to the specified normal mode.

Syntax

`MDNORMAL` *`Point_ID nnnn`*

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
nnnn	A 4-character mode mnemonic: MAN = Manual AUTO = Automatic (the default) CASC = Cascade COMP = Composite

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

MDREVERS

Description

Reverses the mode parameter value for the lowest bit (Bit 0) reverse only. Do not enable MD reverse on points connected to Bristol Babcock controllers and Bristol Babcock OpenBSI controllers.

Syntax

`MDREVERS` *`Point_ID Flag`*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.

Part	Description
Flag	Defines whether the reverse should apply to the parameter. Valid values are *Y* (reverse) or *N* (do not reverse).

Remarks

Applies to analog points and status points only.

Related topics

“Scanning entries” on page 22

METER

Description

Allows a meter factor to be entered for accumulator points. This value is a multiplier and is usually close to 1.0. It is typically used to compensate for calibration errors.

Syntax

```
METER Point_ID M
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
M	Meter factor, floating-point (default is *1.0*).

Related topics

“Scanning entries” on page 22
“Naming rules for points” on page 72

ONSCAN

Description

Allows the building of points “off-scan”.

Syntax

```
ONSCAN Point_ID A
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
A	*Y* = Point is built “on-scan”. This is the default. *N* = Point is built “off-scan”.

Remarks

- Do not include this entry if point is to be built “on-scan”. The majority of points will be built like this.
- This entry should appear immediately after the *ADD* entry.

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72

OPLIMIT

Description

Allows specification of high and low limits for an output point. If an output higher or lower than the corresponding limit is requested, then the output will be clamped at the limit and the control executed.

Syntax

```
OPLIMIT Point_ID Lo-limit Hi-limit
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Lo-limit	Low limit, floating-point (default is 0).
Hi-limit	High limit, floating-point (default is 100.0).

Remarks

- This limit only applies to controls from within the server. It does not limit the value in the controller from internal or other external changes.

Related topics

"Control entries" on page 23
"Naming rules for points" on page 72

OPPULSE

Description

Allows specification of the output pulse width for status output points.

Syntax

```
OPPULSE Point_ID PW
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.

Part	Description
PW	Pulse width code: 0 = Latched (Default) 1 = 200 ms 2 = 300 ms 3 = 400 ms 4 = 500 ms 5 = 800 ms 6 = 1 s 7 = 2 s 8 = 3 s 9 = 4 s 10 = 5 s 11 = 10 s 12 = 20 s 13 = 30 s 14 = 60 s 15 = 120 s

Remarks

- This pulsing is performed by the server. If the server issues a control, a pulse width later, the server will issue the reverse control.
- Where feasible, the controller should perform pulsing. In the case of TDC 3000 PIU, MC, and A-MC, the hardware pulse width may be set using the Box Configuration Summary display.
- Critical applications should not use server pulsing.
- It is not recommended to define an OPSOURCE (see xxSOURCE) address when using a pulse.
- This entry must appear after the OPDESTIN (see xxDESTIN) entry.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72
“xxSOURCE” on page 70
“xxDESTIN” on page 65

OPREVERS

Description

Reverses an entire OP parameter or individual bits of an OP parameter.

Syntax

```
OPREVERS Point_ID Flag Mask
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.

Part	Description
Flag	Defines whether the reverse should apply to the whole parameter. Valid values are *Y* to apply the reverse to the whole parameter, or *N* to apply the reverse to individual bits according to the mask. The flag takes precedence over the mask.
Mask	Defines the bits to be reversed if *Flag* = *N*. There is one mask for each bit to be reversed. Valid values are *Y* (reverse) or *N* (do not reverse). For a 2–bit status point OP, values could be *YY*, *YN*, *NY*, or *NN*.

Remarks

Applies to status points only.

The reverse applies independently of whether non-consecutive bit addresses are defined.

Related topics

“Scanning entries” on page 22

OPWIDTH

Description

Allows specification of the number of output bits for a status point (that is, single or dual output bit).

Syntax

```
OPWIDTH Point_ID A
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
A	1 = single (the default) 2 = dual

Remarks

- The output action (that is, latched or pulsed) is configured via the *OPPULSE* entry.
- This entry should appear before all other *OP...* entries for a given point.

Related topics

“Scanning entries” on page 22
“Naming rules for points” on page 72

PARAM

Description

Adds a:

- Child point to a container point
- User-defined parameter to a point

Syntax

To add a child point to a container point:

`PARAM` *`ParentPoint_ID Alias`* `PNT` *`ChildPoint_ID`*

To add a user-defined parameter to a point:

`PARAM` *`Point_ID ParamName, 0,Flags,ENTITY, {ParamDefinition`*`}`

Additional properties of the user-defined parameter are set using keywords.

Part	Description
ParentPoint_ID	The parent point's name. See the topic titled "Naming rules for points."
Alias	The alias (user-defined name) for the contained point.
ChildPoint_ID	The name of the contained point.
Point_ID	The point's name.
ParamName	The name of the user-defined parameter.
Flags	Optional list of comma-separate flags chosen from the following sets. Write access: • FL_RD = Read only • FL_RW = Read-write (default — can be omitted) Retain parameter value when downloaded: • FL_KOV= Keep old value
ParamDefinition	The parameter's definition.

Parameter definition for user-defined parameters

The syntax of the parameter definition for a user-defined point is:

`{`*`ValueType`*`, NUM|CHAR,` *`LinkType`*`,` *`Value`*`}`

Part	Description
ValueType	The parameter's value type: *INT2* *INT4* *DBLE* *REAL* *String*
NUM|CHAR	Set to *CHAR* if *ValueType* is set to *String*. Otherwise, set to *NUM*.
LinkType	The type of data the parameter represents: *PARAM* (point parameter) *DATAIO* (user file) *VARIABLE* *CONSTANT*

Part	Description
Value	Represents the: • Point parameter address if *LinkType* is set to *PARAM* (The address syntax is: *Point_ID*, *ParamName*, *Offset*) • File/record/word address if *LinkType* is set to *DATAIO* (The address syntax is: *File*, *Record*, *Field*) • Initial value if *LinkType* is set to *VARIABLE* (The value is enclosed in double quotes, for example "255") • Value if *LinkType* is set to *CONSTANT* (The value is enclosed in double quotes, for example "21Mar2002")

Examples

The following example creates a "tank" container point.

```
DEL T1TANK
ADD T1TANK CON00000 Tank One
PARAM T1TANK PUMP PNT T1PUMP
PARAM T1TANK LEVEL PNT T1LEVEL
PARAM T1TANK VALVE PNT T1VALVE
DISPLAY T1TANK TANKFARM
PNTDTLPG T1TANK TANK
```

The following example creates four user-defined parameters (one for each type of data) for point "POIACC1."

```
PARAM  POIACC1   UdefConstant,0,ENTITY,{CHAR,CHAR,CONSTANT,"21March2002"}
PARAM  POIACC1   UdefDatabaseRef,0,ENTITY,{INT2,NUM,DATAIO,251,1,1}
PARAM  POIACC1   UdefParamRef,0,ENTITY,{REAL,NUM,PARAM,POISTA27,PV,0}
PARAM  POIACC1   UdefVariable,0,ENTITY,{REAL,NUM,VARIABLE,100}
```

Related topics

"General entries" on page 21
"Naming rules for points" on page 72

PARENT

Description

Defines the parent asset for the point. If no *PARENT* entry is specified, the point becomes an unassigned item.

Syntax

```
PARENT Point_ID ParentAssetName
```

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points".
ParentAssetName	The tag name or full item name of the asset to which the point belongs. See "Naming rules for assets within an Asset Model".

Related topics

"General entries" on page 21
"Naming rules for assets within an Asset Model" on page 75
"AREA" on page 33
"Naming rules for points" on page 72

PNTDTLPG

Description

Defines the Point Details display for a point.

Syntax

```
PNTDTLPG Point_ID PageName
```

Part	Description
Point_ID	The point's name. See the topic "Naming rules for points".
PageName	The name or number of the display to be used as the Point Details display. The name or number can have up to 255 characters, but must not start with a blank character.

Related topics

"General entries" on page 21
"Naming rules for points" on page 72

PNTSRVTP

Description

Defines the type of communications interface used to read to and write from the point.

Syntax

```
PNTSRVTP Point_ID Interface Instance_Name
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Interface	The name of the communications interface.
Instance_Name	The unique identifier for the point server. Also known as the alias of the point server.

Related topics

"Naming rules for points" on page 72

PVALGO

Description

Attaches an algorithm to a point and assigns an algorithm data block.

Syntax

```
PVALGO Point_ID Num Block
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Num	The number of the PV algorithm. Must be a valid algorithm number.
Block	The number of the algorithm data block used by this algorithm for this point. You must specify a different block for each algorithm/point combination. (Use the *algl st* utility to find a free block. For details, see the *Server and Client Configuration Guide*.) A warning message is issued by **pntbld** if a block is used by more than one algorithm/point combination.

Remarks

- PV algorithms run periodically, specified by the PVPERIOD entry. (See the "xxPERIOD" topic.)
- See the "Configuring PV algorithms using pntbld" topic for a description of each algorithm.
- *PV Algo 12 Composite Alarm Processing* requires the multiple usage of algorithm blocks. In this case, the warning should be ignored.

Related topics

"Algorithm entries" on page 25
"Naming rules for points" on page 72
"Configuring PV algorithms using pntbld" on page 86
"ALG(xx)" on page 29
"xxPERIOD" on page 69
"PV Algo 12: Composite Alarm Processing" on page 93

PVCLAMP

Description

Causes the PV of an analog point to be clamped at 0% if it is less than the PV clamp low limit. Similarly, the PV will be clamped at 100% if it is greater than the PV clamp high limit.

Syntax

```
PVCLAMP Point_ID A
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
A	Y = activate clamping N = deactivate clamping

Remarks

- Clamping is performed before any PV algorithms.
- PV clamp low and high limits are set on the Point System-Wide Configuration display.

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72

PVREVERS

Description

Reverses an entire PV parameter or individual bits of a PV parameter.

Syntax

`PVREVERS` *`Point_ID Flag Mask`*

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
Flag	Defines whether the reverse should apply to the whole parameter. Valid values are *Y* to apply the reverse to the whole parameter, or *N* to apply the reverse to individual bits according to the mask. The flag takes precedence over the mask.
Mask	Defines the bits to be reversed if *Flag* = *N*. There is one mask for each bit to be reversed. Valid values are *Y* (reverse) or *N* (do not reverse). For a 2–bit status point PV, values could be *YY*, *YN*, *NY*, or *NN*. For a 3–bit status point PV, values could be *YYY*, *YYN*, *YNY*, *YNN*, *NYY*, *NYN*, *NNY*, or *NNN*.

Remarks

Applies to status points only.

The reverse applies independently of whether non-consecutive bit addresses are defined.

Related topics

"Scanning entries" on page 22

RANGE

Description

Specifies the 0% and 100% values for a point.

For analog points and accumulator points, the engineering units (EU) are also specified. For status points, this entry specifies the number of input bits.

For accumulator points this entry does not specify a percentage but rather the minimum and maximum values for the point. Above this maximum value, an asterisk (*) will be displayed until the point is reset by an operator.

Syntax

`RANGE` *`Point_ID 0% 100% Enunit`*

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
0%	Zero percent represents the value (in engineering units) of the bias of the Input signal for analog points, and the starting value for accumulator points. For status and accumulator points, the value must be 0.0 (default)

Part	Description
100%	One hundred percent represents the value (in engineering units) of the full scale value of the input signal for analog points, and the maximum value for accumulator points. For status points the value must be 1.0, 3.0, or 7.0 representing 1–bit, 2–bit, or 3–bit digital values. Default = 100 for analog points and 1,000,000 for accumulator points.
Enunit	Engineering units (for analog and accumulator points only). Can consist of up to 8 alphanumeric characters.

Related topics

"General entries" on page 21
"DRIFTDB" on page 38
"Naming rules for points" on page 72
"CNTRLDB" on page 34

REVERSE

Description

Specifies whether the output of a status or analog point is reversed. For analog output signals this facility is needed when the device to be controlled "closes" on a low (0%) signal and "opens" on a high (100%) signal.

Syntax

```
REVERSE Point_ID A
```

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
A	Reverse output indicator: N = "Normal", default Y = "Reverse"

Remarks

- For analog output:

	OP+Value	Signal to/from device
Normal	0%	0%
	100%	100%
Reverse	0%	100%
	100%	0%

- For digital output:

	Command Pushbutton	Single Output Action	Dual Output Action
Normal	F9 (Raise function key)	1	10
	F10 (Lower function key)	0	01
Reverse	F9 (Raise function key)	0	01
	F10 (Lower function key)	1	10

In addition to reversing the control action this entry will provide reverse output indication on the *OPSOURCE*.

Related topics

"Control entries" on page 23
"Naming rules for points" on page 72
"TARGET" on page 63

ROLOVR

Description

Allows a meter rollover value to be entered for accumulator points. This value corresponds to the value at which the meter reading rolls over to zero.

Syntax

`ROLOVR` *`Point_ID Value`*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Value	Meter rollover value (default = *4095*).

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72

SCALE

Description

Specifies the scale factor for accumulator points. This factor is a multiplier and is used to convert counts to engineering units.

Syntax

`SCALE` *`Point_ID S`*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
S	Scale factor, floating-point number (default = 1.0).

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72

SCRIPT

Description

Defines one or more server scripts for a point.

For details about server scripts, see the *Server Scripting Reference*.

Syntax

```
SCRIPT   Point_ID BEGINSCRIPT
SCRIPT   Point_ID <HWSCRIPTCOLLECTION><HWSCRIPTS>
SCRIPT   Point_ID <HWSCRIPT ObjectType="ObjectType" ObjectID="ObjectName"
EventType="EventName1" EngineID="">FirstLineOfCode
SCRIPT   Point_ID LineOfCode.
.
SCRIPT   Point_ID LineOfCode
SCRIPT   Point_ID LastLineOfCode</HWSCRIPT>
SCRIPT   Point_ID <HWSCRIPT ObjectType="ObjectType" ObjectID="ObjectName"
EventType="EventName2" EngineID="">FirsLineOfCode
SCRIPT   Point_ID LineOfCode
.
.
SCRIPT   Point_ID LineOfCode
SCRIPT   Point_ID LastLineOfCode</HWSCRIPT></HWSCRIPTS><HWPERIODICEVENTS/></HWSCRIPTCOLLECTION>
SCRIPT   Point_ID ENDSCRIPT
```

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
ObjectType	The type of object to which the following script applies. Either *Point* or *Param*.
ObjectName	The name of the object to which the following script applies. If the script (event) is associated with the point, it is point's name (Point_ID). If the script is associated with one of the point's parameters, the name is of the form: *Point_ID.ParameterName*.
EventName	The name of the event to which the following script applies, such as OnAlarm or OnChange.
FirstLineOfCode *LineOfCode* *LastLineOfCode*	The lines of script code.

Remarks

- Scripts are not checked for validity when they are uploaded. It is up to you to test the operation of each script.

Example

This example for point 'POINTACC1' contains scripts for two events: *OnAlarm* (which is associated with the point) and *OnOperatorChange* (which is associated with the point's PV).

```
SCRIPT   POIACC1 BEGINSCRIPT
SCRIPT   POIACC1 <HWSCRIPTCOLLECTION><HWSCRIPTS>
SCRIPT   POIACC1 <HWSCRIPT ObjectType="Point" ObjectID="POIACC1"
EventType="OnAlarm" EngineID="">Dim AlmDtls
SCRIPT   POIACC1
SCRIPT   POIACC1   If (ParamValue("PUMP1.UNACKALARMEXISTS")) Then
SCRIPT   POIACC1     Set AlmDtls = Server.CreateAlarmDetails
SCRIPT   POIACC1     AlmDtls.Description = "TANK1 and PUMP1 both
have problems."
SCRIPT   POIACC1     AlmDtls.Priority = hscUrgent
SCRIPT   POIACC1     AlmDtls.Area = ""
SCRIPT   POIACC1     Server.GenerateAlarm AlmDtls
SCRIPT   POIACC1   Else
SCRIPT   POIACC1     ParamValue(".SP") = ParamValue(".SP") / 2
SCRIPT   POIACC1   End If</HWSCRIPT>
SCRIPT   POIACC1 <HWSCRIPT ObjectType="Param" ObjectID="POIACC1.PV"
EventType="OnOperChange" EngineID="">Dim RMS
SCRIPT   POIACC1
SCRIPT   POIACC1   RMS = ParamValue(".A1")
SCRIPT   POIACC1   RMS = (RMS * RMS + EventInfo.value * EventInfo.value) /2
SCRIPT   POIACC1   ParamValue(".A1") = Sqr(RMS)</HWSCRIPT></HWSCRIPTS><HWPERIODICEVENTS/></
```

```
HWSCRIPTCOLLECTION>
SCRIPT   POIACC1 ENDSCRIPT
```

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

SPLIMIT

Description

Allows specification of high and low limits for the set point. If a set point higher or lower than the corresponding limit is requested, then the set point will be clamped at the limit.

Syntax

`SPLIMIT` *`Point_ID Lo-limit Hi-limit`*

Part	Description
Point_ID	The point's name. See the "Naming rules for points" topic.
Lo-limit	Low limit, floating-point (default = 0% or range)
Hi-limit	High limit, floating-point (default = 100.0% of range)

Remarks

- This limit only applies to controls from within the server application. It does not limit the value in the controller from internal or other external changes.

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

STATEDES

Description

Defines the State descriptor for each state of a status point.

Syntax

`STATEDES` *`Point_ID Descr1 Descr2... Descr8`*

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
Descrx	State descriptor for the specified state of a status point. State descriptors may be composed of any 8 alphanumeric characters. For single-bit status points, two state descriptions must be specified; for dual-bit status points four state descriptions, and for 3-bit points all eight state descriptions are required. If states are not required, you can use hyphens to 'pad out' the states, as shown in the example.

Example

This example shows how to pad out states that are not required.

```
       ON  -  OFF FAILED  -  -  -  -
States  0  1   2    3     4  5  6  7
```

Related topics

“Scanning entries” on page 22
“Naming rules for points” on page 72

SUBTYPE

Description

Defines the subtype of a point that is built in the Network or System Components tree. The subtype is primarily used to determine what icon to show in the Location Pane of the System Status Display. This keyword is not applicable to points assigned to Assets.

Syntax

`SUBTYPE Point_ID ST R`

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
ST	The point's subtype.
R	Redundant Flag. Use: *Y* = redundant and primary, or redundant and only have one redundant icon *N* = not redundant *B* = redundant and backup

Related topics

“General entries” on page 21
“Naming rules for points” on page 72

TARGET

Description

Assigns target states to single-, dual-, or 3-bit status points for CLOSED/STOP/OFF and OPEN/START/ON output states.

Syntax

`TARGET Point_ID A B C D`

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
A	Input (PV) state that corresponds to output (OP) state 0.

Part	Description
B	Input (PV) state that corresponds to output (OP) state 1.
C	Input (PV) state that corresponds to output (OP) state 2. Specify *F* for single-bit point.
D	Input (PV) state that corresponds to output (OP) state 3. Specify *F* for single-bit point.

Remarks

- Single-bit outputs only have states 0 and 1. This does not affect the values sent by a control Station to the *OPDESTIN* address (see *xxDESTIN*).
- Control actions using the up and down arrow keys will display the corresponding requested *TARGET* state in the Message Zone on the Station display:

Command Pushbutton	Single	Dual
F9 (Raise function key)	B	C (ON)
F10 (Lower function key)	A	B (OFF)

These control indications can be reversed using the REVERSE entry.

Default values are:

Single: A=0, B=1

Dual: B=1, C=2

- Specify *F* for A, B, C, or D if you do not wish to assign a state to that OP state (see the following example).

Example

```
TARGET statuspoint1 F 1 2 F
```

This entry assigns the following state descriptors for the OP.

OP = 0 (no state descriptor)

OP = 1 (same state descriptor as PV = 1)

OP = 2 (same state descriptor as PV = 2)

OP = 3 (no state descriptor)

Related topics

“Control entries” on page 23
“REVERSE” on page 59
“Naming rules for points” on page 72
“xxDESTIN” on page 65
“CNTRLTO” on page 36

TIMESYNC

Description

Specifies that a point is the time synchronization point for a TDC 3000 DHP.

Syntax

`TIMESYNC Point_ID`

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.

Remarks

- Time synchronization will take place at midnight, at system time change, and at controller initialization.
- Time data is transferred into the following addresses:

  PVSOURCE address-minutes this hour

  OPSOURCE address-minutes since midnight

  SPSOURCE address-hours this day

  A1SOURCE address-year

  A2SOURCE address-month

  A3SOURCE address-date

  A4SOURCE address-day of week

Related topics

“Control entries” on page 23
“Naming rules for points” on page 72

TREND

Description

Assigns a point to a trend. For details about trends, see the *Configuration Guide*.

Syntax

TREND *Point_ID TTTT P* [*Param*]

Part	Description
Point_ID	The point's name. See the “Naming rules for points” topic.
TTTT	Trend number
P	Position 1 to 32
Param	The parameter to display on the Trend. If this is left blank, the PV is assumed.

Related topics

“General entries” on page 21
“Naming rules for points” on page 72

xxDESTIN

Description

Allows specification of the parameter destination (or output) address.

Syntax

*xx*DESTIN *Point_ID RTU Address*

Part	Description
xx	Parameter (MD, OP, SP, A1, A2, A3, A4). All are valid for analog points. Parameter (A1, A2, A3, A4). All are valid for status points.
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
RTU	Controller number of the output device as configured by *hdwbld*.
Address	Address within the controller—the same syntax as for *xxSOURCE*.

Addressing non-consecutive bit addresses

For status points with a 4–state output, you can address two non-consecutive bits as the OP Destination Address.

- Use *OPDESTIN* to address the first bit (Bit 0)
- Use *A3DESTIN* to address the second bit (Bit 1)

The following example shows how to address a 4-state OP using the *OP* and *A3* parameters.

```
OPSOURCE OPC_4StateOP   018 opc_item_name_for_op_bit0
OPDESTIN OPC_4StateOP   018 opc_item_name_for_op_bit0
A3SOURCE OPC_4StateOP   018 opc_item_name_for_op_bit1
A3DESTIN OPC_4StateOP   018 opc_item_name_for_op_bit1
A3NAME   OPC_4StateOP   OP_BIT1
```

Remarks

- This entry is not required for database points because a destination address is automatically configured when there is a source address.
- If an *xxSOURCE* address is specified for the same parameter, the controller numbers must be identical.
- When writing to a Computer Output (CO) slot of a TDC Hiway Extended Controller, specify the value as a percentage (%) when using OPDESTIN, or in engineering units (EU) when using A1DESTIN–4DESTIN.

Related topics

"Scanning entries" on page 22
"OPPULSE" on page 51
"Naming rules for points" on page 72
"xxSOURCE" on page 70
"TARGET" on page 63

xxDYNSCN

Description

Enables and disables dynamic scanning of accumulator, analog, and status points.

Syntax

*xx*DYNSCN *Point_ID Y/N*

Part	Description
xx	Parameter (MD, PV). All are valid for accumulator points. Parameter (MD, OP, PV, SP, A1, A2, A3, A4). All are valid for analog points. Parameter (MD, OP, PV). All are valid for status points.
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
Y/N	*Y* = Enables dynamic scanning. *N* = Disables dynamic scanning

Remarks

If *xx*DYNSCN does not exist for a point parameter, dynamic scanning is enabled by default.

Example

This entry disables dynamic scanning on the PV parameter of *FIC123*.

```
PVDYNSCN   FIC123    N
```

Related topics

“Naming rules for points” on page 72
“Scanning entries” on page 22

xxNAME

Description

Allows descriptors to be associated with the parameters of analog points and status points.

Syntax

*xx*NAME *Point_ID Name*

Part	Description
xx	Parameter (MD,OP,SP, A1, A2, A3, A4). All are valid for analog points. Parameter (A1, A2, A3, A4). All are valid for status points.
Point_ID	The point's name. See the "Naming rules for points" topic.
Name	An 8-character descriptor for the parameter. If the descriptor for the auxiliary variables (A1–A4) is the name of a server internal point parameter, then the auxiliary variable value is pushed through to that parameter. Use this mechanism to address non-consecutive bits for status point parameters.

Example

This entry specifies that the value scanned by A1 can be referenced as *FIC101.K*.

```
A1NAME   FIC101     K
```

Addressing non-consecutive bit addresses

The following example shows how an 8–state PV is addressed using the *PV*, *A1*, and *A2* parameters.

```
PVSOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit0
A1SOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit1
A2SOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit2
A1NAME   OPC_8StatePV   PV_BIT1
A2NAME   OPC_8StatePV   PV_BIT2
```

A 4–state OP with individual bit addresses is handled in a similar way. For a 4–state OP, the second bit is addressed by the *A3* parameter, as shown in the following example.

```
OPSOURCE OPC_4StateOP   018 opc_item_name_for_op_bit0
OPDESTIN OPC_4StateOP   018 opc_item_name_for_op_bit0
A3SOURCE OPC_4StateOP   018 opc_item_name_for_op_bit1
A3DESTIN OPC_4StateOP   018 opc_item_name_for_op_bit1
A3NAME   OPC_4StateOP   OP_BIT1
```

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72

xxOFFDLY

Description

Defines the Off Delay period for an alarm on an accumulator point, analog point, and status point.

Syntax

```
xxOFFDLY Point_ID sec
```

Part	Description
xx	For accumulator points, *xx* is: • *An*, where *n* specifies the alarm limit number (1 to 4). For analog points, *xx* is either: • *An*, where *n* specifies the alarm limit number (1 to 8). • *GN* specifies the Unreasonable alarm. For status points, *xx* is: • *Sn*, where *n* specifies the state alarm number (0 to 7).
Point_ID	The point's name. See the "Naming rules for points" topic.
sec	The number of seconds (0–9999) to delay the return to normal (RTN).

Example

This example defines an Off Delay period of 45 seconds for the S2 state of the FIC001 status point.

```
S2OFFDLY FIC001 45
```

Related topics

"Naming rules for points" on page 72

xxONDLY

Description

Defines the On Delay period for an alarm on an accumulator point, analog point, and status point.

Syntax

xx`ONDLY` *`Point_ID sec`*

Part	Description
`xx`	For accumulator points, *`xx`* is: • *`An`*, where *`n`* specifies the alarm limit number (1 to 4). For analog points, *`xx`* is either: • *`An`*, where *`n`* specifies the alarm limit number (1 to 8). • *`GN`* specifies the Unreasonable alarm. For status points, *`xx`* is: • *`Sn`*, where *`n`* specifies the state alarm number (0 to 7).
`Point_ID`	The point's name. See the “Naming rules for points” topic.
`sec`	The number of seconds (0–9999) to delay the alarm.

Example

This example defines an On Delay period of 30 seconds for alarm limit 7 of the FIC123 analog point.

```
A7ONDLY FIC123 30
```

Related topics

“Naming rules for points” on page 72

xxPERIOD

Description

Assigns a scan period to the parameter.

Syntax

xx`PERIOD` *`Point_ID P`*

Part	Description
`xx`	Parameter (MD, PV). All are valid for accumulator points. Parameter (MD, OP, PV, SP, A1, A2, A3, A4). All are valid for analog points. Parameter (MD, OP, PV). All are valid for status points.
`Point_ID`	The point's name. See the topic titled "Naming rules for points" for information about naming rules.

Part	Description
P	Scan period in seconds. The valid periods are: 1, 2, 5, 10, 15, 30, 60, 120, 300, 900, 1800, 3600, and DEMAND (for dynamic scanning). If no scan period is specified, it defaults to no scan period.

Remarks

- For maximum scanning efficiency it is important that addresses that may be grouped into a single block or list by the scanning system be scheduled at the same frequency.
- The period should be less than or equal to the history interval if the point is assigned for history collection.
- Use the *DEMAND* period when you configure dynamic scanning.

Related topics

"Scanning entries" on page 22
"Naming rules for points" on page 72
"LPPERIOD" on page 47
"PVALGO" on page 56
"Interpreting pntbld error and warning messages" on page 80
"PV Algo 12: Composite Alarm Processing" on page 93

xxSOURCE

Description

Defines the address of the point parameter's input and its characteristics.

Syntax

*xx*SOURCE *Point_ID RTU Address*

Part	Description
xx	Parameter (MD, OP, PV, SP, A1, A2, A3, A4). All are valid for analog points. Parameter (A1, A2, A3, A4). All are valid for status points.
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
RTU	Controller number of the output device as configured by *hdwbld*.
Address	There are three classes of address syntax: • Hardware addresses which are addresses of values within controllers. These are controller-specific and are described in the associated *Controller Reference*. • If you leave out the address, this specifies that there is to be no acquisition. • Database addresses which are addresses of other information in the server database. See "Addressing another point parameter" and "Addressing a user file." **Attention** If a SP (set point) address references either a database file or a point parameter, you must only specify the SPSOURCE address. (This is because the SPSOURCE and SPDESTIN fields are the same for database and point addresses.)

Addressing non-consecutive bit addresses

For 4–state and 8–state status points, you can address up to three non-consecutive bits as the PV Source Address. A 4–state point will have two addresses and an 8–state point will have three addresses.

- Use *PVSOURCE* to address the first bit (Bit 0)
- Use *A1SOURCE* to address the second bit (Bit 1)
- Use *A2SOURCE* to address the third bit (Bit 2)

The following example shows how to address an 8–state PV using the *PV*, *A1*, and *A2* parameters.

```
PVSOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit0
A1SOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit1
A2SOURCE OPC_8StatePV   018 opc_item_name_for_pv_bit2
A1NAME   OPC_8StatePV   PV_BIT1
A2NAME   OPC_8StatePV   PV_BIT2
```

For status points with a 4–state output, you can address up to two non-consecutive bits as the OP Source Address.

- Use *OPSOURCE* to address the first bit (Bit 0)
- Use *A3SOURCE* to address the second bit (Bit 1)

The following example shows how to address an 4–state OP using the *OP* and *A3* parameters.

```
OPSOURCE OPC_4StateOP   018 opc_item_name_for_op_bit0
OPDESTIN OPC_4StateOP   018 opc_item_name_for_op_bit0
A3SOURCE OPC_4StateOP   018 opc_item_name_for_op_bit1
A3DESTIN OPC_4StateOP   018 opc_item_name_for_op_bit1
A3NAME   OPC_4StateOP   OP_BIT1
```

Addressing another point parameter

The following syntax is used to acquire a value from another point parameter:

```
P: Point_id parameter
```

Part	Description
Point_ID	The point's name. See the topic titled "Naming rules for points" for information about naming rules.
parameter	Is the point parameter to acquire data from.

Remarks

- Normally only the point parameters SP, PV, OP, MD, A1, A2, A3, and A4 are scanned.
- To acquire data from a history parameter, the point must be built with history collected for that history type.

Addressing a user file

Use the following syntax to read a value in a user file. (For details about user files, see the "Accessing user-defined data" topic in the *Application Development Guide*.)

```
F:ff R:rr W:ww B:bb W:wd fmt
```

Part	Description
ff	The file number.
rr	The record number.
ww	The word number.
bb	The base bit (0-15). Only required for a bit field.
wd	The field width (1-16). Only required for a bit field.

Part	Description
fmt	The data format name, where: *INT2* for 2-byte and for partial integers *INT4* for a 4-byte integer *REAL* for real data *DBLE* for double precision real If no format is specified, INT2, the default, is assumed.

Related topics

"Scanning entries" on page 22
"OPPULSE" on page 51
"Naming rules for points" on page 72
"xxDESTIN" on page 65
"Interpreting pntbld error and warning messages" on page 80
"PV Algo 12: Composite Alarm Processing" on page 93

Naming rules for points

All points within your system have a *tag name* (also called a *point ID* or *point name*) and an *item name*. Tag names must be unique, whereas item names can be duplicated as long as the resulting full item name is unique.

When a point is created, it is given a unique tag name, for example, POINT01 or POINT02. This identifier is used in Experion whenever it is necessary to refer to a point in the server (for example, on a custom display or in a report).

Point names must follow certain naming rules:

- An item name cannot match the item name of any other point belonging to the same parent asset.
- Tag names must be unique within the cluster server.
- Tag names and item names can contain up to 40 single-byte or 20 double-byte alphanumeric characters, with at least one alpha character.
- Tag names and item names are not case-sensitive: `POINT01` and `Point01` represent the same asset.
- The first character of a tag name and an item must not be any of the following characters:
  - At sign (@)
  - Dollar sign ($)
  - Space
- Tag names and item names cannot contain any of the following characters:
  - Ampersand (&)
  - Asterisk (*)
  - Backslash (\)
  - Braces { } (rule applies to item names only)
  - Brackets [ ]
  - Caret (^)
  - Colon (:)
  - Comma (,)
  - Double quote (")
  - Equals (=)
  - Forward slash (/)

  - Greater than (>)
  - Less than (<)
  - Number sign (#)
  - Parentheses ( )
  - Percent (%)
  - Period (.)
  - Question mark (?)
  - Semi colon (;)
  - Single quote (')
  - Space (rule applies to tag names only)
  - Tabs
  - Vertical bar (|)
- The last character of a tag name and an item must not be a space.
- A *full item name*:
  - Must not be longer than 200 characters
  - Must be unique

It is also important for both engineers and operators that points are named in a consistent and ‘user-friendly’ manner. You might, for example, consider:

- Basing the names on existing documentation, such as schematics and wiring diagrams, so that users can easily switch between documents and displays.
- Using the same prefix for related points, so that users can easily find related points.
- Starting each part of a name with a capital, to improve readability. For example: `Boiler1Temp`.

Related topics

“ADD” on page 26
“AKDESTIN” on page 26
“ALARM” on page 27
“ALMLIMn” on page 30
“ALARMDB” on page 29
“ALG(xx)” on page 29
“ALMMSG” on page 32
“ALMXCHG” on page 32
“CCONFIRM” on page 34
“CNTINH” on page 34
“CNTRLDB” on page 34
“CNTRLTO” on page 36
“CNTRLLVL” on page 35
“DEL” on page 37
“DISPLAY” on page 37
“DRIFTDB” on page 38
“RANGE” on page 58
“ENTNAM” on page 39
“ESIGPRIM” on page 39
“ESIGSEC” on page 40
“GROUP” on page 41
“HISGATE” on page 42
“HISTFAST” on page 44
“HISTSLOW” on page 46

"HISTEXTD" on page 43
"HISTORY" on page 45
"JNLONLY" on page 47
"LPPERIOD" on page 47
"LPSOURCE" on page 48
"MDNORMAL" on page 49
"METER" on page 50
"ONSCAN" on page 50
"OPLIMIT" on page 51
"OPPULSE" on page 51
"xxSOURCE" on page 70
"xxDESTIN" on page 65
"OPWIDTH" on page 53
"PARAM" on page 53
"PNTDTLPG" on page 56
"PNTSRVTP" on page 56
"PVALGO" on page 56
"PVCLAMP" on page 57
"REVERSE" on page 59
"TARGET" on page 63
"ROLOVR" on page 60
"SCALE" on page 60
"SCRIPT" on page 60
"SPLIMIT" on page 62
"STATEDES" on page 62
"SUBTYPE" on page 63
"TIMESYNC" on page 64
"TREND" on page 65
"xxDYNSCN" on page 66
"xxNAME" on page 67
"xxOFFDLY" on page 68
"xxONDLY" on page 69
"AREA" on page 33
"MDDISABL" on page 48
"PARENT" on page 55
"xxPERIOD" on page 69

Naming rules for user-defined parameters

User-defined parameter names must follow certain naming rules:

- Parameter names must be unique.
- Parameter name can contain up to 255 alphanumeric characters, with at least one alpha character.
- Parameter names can contain periods (.).
- Parameter names are not case-sensitive.
- Parameter names cannot contain any of the following characters:
  - Ampersand (&)
  - Asterisk (*)
  - Backslash (\)

  – Caret (^)
  – Colon (:)
  – Comma (,)
  – Double quote (")
  – Forward slash (/)
  – Greater than (>)
  – Less than (<)
  – Number sign (#)
  – Percent (%)
  – Question mark (?)
  – Semi-colon (;)
  – Single quote (')
  – Space
  – Tab
  – Vertical bar (|)

Naming rules for assets within an Asset Model

Each asset has three names:

- A *tag name* (also called a *point ID* or *point name*). A unique name, which is used by the system to identify the asset. The tag name must be unique across the DSA system.
- An *item name*. A descriptive (user friendly) name for the asset. Unlike the point name, the item name only has to be unique with respect to its siblings (other items that share the same parent item).

  For example, you can give many assets the item name of 'Mainvalve' provided each asset has a different parent.
- A *full item name* (also called an *enterprise model name*). A unique name, which consists of the tag name and the names of all its ancestors within the asset model.

  A full item name has the same structure as the full path name of a file on a network. The following example is the full item name of an asset named ‘Agitator’:

  ```
  /Assets/Precipitation/Train1/Precipitator1/Agitator
  ```

Asset names must follow certain naming rules:

- Tag names must be unique.
- Tag names and item names can contain up to 40 single-byte or 20 double-byte alphanumeric characters, with at least one alpha character.
- Tag names and item names are not case-sensitive: *`Cooling Tower1`* and *`cooling tower1`* represent the same asset.
- The first character of a tag name and an item must not be any of the following characters:
  – At sign (@)
  – Dollar sign ($)
  – Space
- Tag names and item names cannot contain any of the following characters:
  – Asterisk (*)
  – Backslash (\)
  – Braces { } (rule applies to item names only)
  – Brackets [ ]
  – Caret (^)

  - Colon (:)
  - Comma (,)
  - Double quote (")
  - Equals (=)
  - Forward slash (/)
  - Greater than (>)
  - Less than (<)
  - Parentheses ( )
  - Period (.)
  - Question mark (?)
  - Semi colon (;)
  - Single quote (')
  - Space (rule applies to tag names only)
  - Tabs
  - Vertical bar (|)
- The last character of a tag name and an item must not be a space.
- A *full item name*:
  - Must not be longer than 200 characters
  - Must be unique

Related topics

"AREA" on page 33
"PARENT" on page 55

PHD collection rule

When configuring your system for PHD collection, the default collection method collects history for *all* point parameters at the same rate, which by default is the fastest history type assigned. However, it is possible (using the `/PHDCOLLECT` switch) to configure PHD history collection for individual point parameters either at a different (slower) rate, or to exclude the point parameter from PHD collection.

PHD collection hierarchy

Experion collects history to the PHD server at the fastest history type configured for the point parameter (e.g., `HISTFAST`, `HISTSLOW`). However, to prevent overloading your system with too much data, a global setting on the **Configuration** tab of the **PHD Server Configuration** display can prevent Experion sending fast history for all point parameters to the PHD server. When a single point parameter is assigned to multiple history types, the hierarchy for collecting history to PHD (assuming no overrides) is:

1. Fast history
2. Standard history
3. Extended history

/PHDCOLLECT rules

When determining whether to collect history to PHD for a point parameter, Experion uses the following rules:

1. If `/PHDCOLLECT=DEFAULT` (or is not defined), Experion collects history to PHD according to the PHD collection hierarchy above.
2. If `/PHDCOLLECT=OVERRIDE`, Experion collects history to PHD at the rate configured for the history type where the switch is defined (e.g., `HISTFAST`, `HISTSLOW`).

> **Attention**
> If a point parameter has multiple history assignments with */PHDCOLLECT=OVERRIDE*, Experion uses the PHD collection hierarchy above to choose the rate to send to PHD.

3. If */PHDCOLLECT=DISABLE*, Experion does not collect history to PHD for the history type where the switch is defined (e.g., *HISTFAST*, *HISTSLOW*).

> **Attention**
> To disable collecting all history to the PHD server for a particular point parameter, *all* defined history assignments on the point parameter must be set to */PHDCOLLECT=DISABLE*.

Examples

This point has its PV assigned to fast history (*HISTFAST*) and standard history (*HISTSLOW*). The two */PHDCOLLECT=DEFAULT* switches on the *HISTFAST* and *HISTSLOW* lines below mean that fast history will be used by PHD collection (based on the PHD collection hierarchy above).

```
&
&Item:      PLANT_DB_ANA202
DEL         PLANT_DB_ANA202
ADD         PLANT_DB_ANA202     ANA00000 Analog 202 Point
ENTNAM      PLANT_DB_ANA202     PLANT_DB_ANA202_Item
RANGE       PLANT_DB_ANA202     10 730 EngUnits
AREA        PLANT_DB_ANA202     PLANT_DB_Asset
HISTFAST    PLANT_DB_ANA202     PV /Period=5 /PHDCOLLECT=DEFAULT
HISTSLOW    PLANT_DB_ANA202     PV /Period=60 /PHDCOLLECT=DEFAULT
PVSOURCE    PLANT_DB_ANA202     010 114
PVPERIOD    PLANT_DB_ANA202     1
ALARM       PLANT_DB_ANA202     3 3 N NNNNNNNN
ALMXCHG     PLANT_DB_ANA202     NNNNNNNN
DRIFTDB     PLANT_DB_ANA202     0
ALARMDB     PLANT_DB_ANA202     10
CNTRLDB     PLANT_DB_ANA202     10
PVCLAMP     PLANT_DB_ANA202     N
OPLIMIT     PLANT_DB_ANA202     0 100
SPLIMIT     PLANT_DB_ANA202     0 100
ALMINH      PLANT_DB_ANA202     N
```

The same point, with the same history assignments, now has the */PHDCOLLECT=OVERRIDE* switch defined on the *HISTSLOW* line. This means that standard history will be used by PHD collection, rather than fast history.

```
&
&Item:      PLANT_DB_ANA202
DEL         PLANT_DB_ANA202
ADD         PLANT_DB_ANA202     ANA00000 Analog 202 Point
ENTNAM      PLANT_DB_ANA202     PLANT_DB_ANA202_Item
RANGE       PLANT_DB_ANA202     10 730 EngUnits
AREA        PLANT_DB_ANA202     PLANT_DB_Asset
HISTFAST    PLANT_DB_ANA202     PV /Period=5 /PHDCOLLECT=DEFAULT
HISTSLOW    PLANT_DB_ANA202     PV /Period=60 /PHDCOLLECT=OVERRIDE
PVSOURCE    PLANT_DB_ANA202     010 114
PVPERIOD    PLANT_DB_ANA202     1
ALARM       PLANT_DB_ANA202     3 3 N NNNNNNNN
ALMXCHG     PLANT_DB_ANA202     NNNNNNNN
DRIFTDB     PLANT_DB_ANA202     0
ALARMDB     PLANT_DB_ANA202     10
CNTRLDB     PLANT_DB_ANA202     10
PVCLAMP     PLANT_DB_ANA202     N
OPLIMIT     PLANT_DB_ANA202     0 100
SPLIMIT     PLANT_DB_ANA202     0 100
ALMINH      PLANT_DB_ANA202     N
```

In this example, the */PHDCOLLECT=DEFAULT* switch has been removed from the *HISTFAST* line because it is not necessary; having no */PHDCOLLECT* switch has the same effect as */PHDCOLLECT=DEFAULT*.

```
&
&Item:      PLANT_DB_ANA202
DEL         PLANT_DB_ANA202
ADD         PLANT_DB_ANA202     ANA00000 Analog 202 Point
ENTNAM      PLANT_DB_ANA202     PLANT_DB_ANA202_Item
RANGE       PLANT_DB_ANA202     10 730 EngUnits
AREA        PLANT_DB_ANA202     PLANT_DB_Asset
```

```
HISTFAST    PLANT_DB_ANA202     PV /Period=5
HISTSLOW    PLANT_DB_ANA202     PV /Period=60 /PHDCOLLECT=OVERRIDE
PVSOURCE    PLANT_DB_ANA202     010 114
PVPERIOD    PLANT_DB_ANA202     1
ALARM       PLANT_DB_ANA202     3 3 N NNNNNNNN
ALMXCHG     PLANT_DB_ANA202     NNNNNNNN
DRIFTDB     PLANT_DB_ANA202     0
ALARMDB     PLANT_DB_ANA202     10
CNTRLDB     PLANT_DB_ANA202     10
PVCLAMP     PLANT_DB_ANA202     N
OPLIMIT     PLANT_DB_ANA202     0 100
SPLIMIT     PLANT_DB_ANA202     0 100
ALMINH      PLANT_DB_ANA202     N
```

In this example, the */PHDCOLLECT=DISABLE* switch on the *HISTFAST* line disables fast history from PHD collection. Standard history will be used by PHD collection, rather than fast history.

Note that this example has the same result as the previous two examples in that standard history—not fast history—is used for PHD collection. The difference is that this example disables fast history from PHD collection, whereas the previous two examples use the override switch on standard history.

```
&
&Item:      PLANT_DB_ANA202
DEL         PLANT_DB_ANA202
ADD         PLANT_DB_ANA202     ANA00000 Analog 202 Point
ENTNAM      PLANT_DB_ANA202     PLANT_DB_ANA202_Item
RANGE       PLANT_DB_ANA202     10 730 EngUnits
AREA        PLANT_DB_ANA202     PLANT_DB_Asset
HISTFAST    PLANT_DB_ANA202     PV /Period=5 /PHDCOLLECT=DISABLE
HISTSLOW    PLANT_DB_ANA202     PV /Period=60
PVSOURCE    PLANT_DB_ANA202     010 114
PVPERIOD    PLANT_DB_ANA202     1
ALARM       PLANT_DB_ANA202     3 3 N NNNNNNNN
ALMXCHG     PLANT_DB_ANA202     NNNNNNNN
DRIFTDB     PLANT_DB_ANA202     0
ALARMDB     PLANT_DB_ANA202     10
CNTRLDB     PLANT_DB_ANA202     10
PVCLAMP     PLANT_DB_ANA202     N
OPLIMIT     PLANT_DB_ANA202     0 100
SPLIMIT     PLANT_DB_ANA202     0 100
ALMINH      PLANT_DB_ANA202     N
```

In this example, the */PHDCOLLECT=DISABLE* switches on both the *HISTFAST* and the *HISTSLOW* lines disable all history from PHD collection. However, Experion server is still collecting fast history and standard history at the defined periods.

```
&
&Item:      PLANT_DB_ANA202
DEL         PLANT_DB_ANA202
ADD         PLANT_DB_ANA202     ANA00000 Analog 202 Point
ENTNAM      PLANT_DB_ANA202     PLANT_DB_ANA202_Item
RANGE       PLANT_DB_ANA202     10 730 EngUnits
AREA        PLANT_DB_ANA202     PLANT_DB_Asset
HISTFAST    PLANT_DB_ANA202     PV /Period=5 /PHDCOLLECT=DISABLE
HISTSLOW    PLANT_DB_ANA202     PV /Period=60 /PHDCOLLECT=DISABLE
PVSOURCE    PLANT_DB_ANA202     010 114
PVPERIOD    PLANT_DB_ANA202     1
ALARM       PLANT_DB_ANA202     3 3 N NNNNNNNN
ALMXCHG     PLANT_DB_ANA202     NNNNNNNN
DRIFTDB     PLANT_DB_ANA202     0
ALARMDB     PLANT_DB_ANA202     10
CNTRLDB     PLANT_DB_ANA202     10
PVCLAMP     PLANT_DB_ANA202     N
OPLIMIT     PLANT_DB_ANA202     0 100
SPLIMIT     PLANT_DB_ANA202     0 100
ALMINH      PLANT_DB_ANA202     N
```

Related topics

"HISTEXCP" on page 42
"HISTSLOW" on page 46
"HISTFAST" on page 44
"HISTEXTD" on page 43

pntbld

Description

pntbld is a command-line utility that downloads point definition files to the server. (It can download while the server software is running.)

Syntax

```
pntbld [file_name] [-out file_name] [-nl] [-le] [-ns] [-offscan] [-r nnn] [-ph] [-del] [-diag] [-dt] [-dg]
```

Part	Description
file_name	The file name of the point definition file. A full or relative path can be specified.
-out file_name	The path-name of the file in which to store the output. By default, the output goes to the Command Prompt window. This option is useful when there is too much output to be viewed in the window. After **pntbld** has finished running, you can peruse the output file.
-nl	Suppresses all output.
-le	Only outputs errors. By default, the point definition file lines are output together with error messages for incorrect lines.
-ns	Suppresses the building of scan tables. This is useful when you are still developing point definition files. After development has finished, you then run **pntbld** without any arguments to build the scan tables. Note that points are not be scanned until scan tables have been built.
-offscan	Only deletes points that are off scan.
-r nnn	Specifies the release number of the point definition file. This option is used for backward compatibility for point definition files created on earlier releases of SCAN to translate them to the new syntax. It is recommended that all earlier releases point build source files are initially imported into Quick Builder.
-ph	Prevents history configuration from being deleted for a point when a *DEL*/*ADD* pair is encountered.
-del	Only processes lines in the point definition file beginning with DEL. This option is useful for deleting points.
-diag	Lists diagnostic messages.
-dt	Removes points from trend assignments.
-dg	Removes points from group assignments.

Remarks

- Quick Builder creates a point definition file when you download a project (or part of a project) to the server.
- For help with error messages, see the topic “Interpreting pntbld error and warning messages”.

Example

This example runs **pntbld** with the point definition file *test.pnt*, which is in the current folder, and directs the output to *Point.out* which is also in the current folder.

```
pntbld test.pnt -out Point.out
```

Related topics

“Interpreting pntbld error and warning messages” on page 80

Interpreting pntbld error and warning messages

pntbld checks the syntax and semantics of point definition files and generates error messages for incorrect entries. The error messages are generally self-explanatory, however, this section gives extra help with interpreting the most common types of error messages.

Syntax errors

Invalid data has been found in one of the fields in the entry. Ensure that the correct data is in the right column for the entry. See the “Point definition entries reference” topic for details about the **pntbld** entries.

There is no source address

The entry requires a source address to be defined first. Add an xxSOURCE entry.

Invalid scan period

The specified scan period is not one of the allowed scan periods. See the xxPERIOD entry for the allowable scan periods. With assistance from Honeywell, these scan periods are sometimes changed using display number 10. In this case, the scan period specified must be one of the changed scan periods.

Point is implemented as …

The same point reference number is already being used by another point. Either delete the other point first or change the reference number to a spare one. You can use the **listag** utility to list the currently used point reference numbers (For details, see the *Configuration Guide*.)

Related topics

“pntbld” on page 79
“Point definition entries reference” on page 21
“xxSOURCE” on page 70
“xxPERIOD” on page 69

Point-management utilities

The following utilities can be used to manage points after they have been downloaded to the server.

Related topics

"bckbld" on page 81
"listag" on page 81
"lisscn" on page 81
"pntdel" on page 81

bckbld

You use **bckbld** (the point backbuilding utility) to create a point definition file.

Instead of uploading from Quick Builder, you can use **bckbld** to create a point definition file from the current point database contents.

After creating a point definition file with **bckbld**, you can view and update it with a text editor.

For details about the bckbld utility, see the "bckbld" topic in the *Server and Client Configuration Guide*.

listag

You use the **listag** utility to find out which point reference numbers have been used.

When you use **pntbld** to create point definitions, you may need to assign a point reference number to each point. If you need to find out which point reference numbers have already been assigned, you can use **listag** to list all the points currently configured in the database.

For details about the listag utility, see the "listag" topic in the *Configuration Guide*.

lisscn

You use the **lisscn** utility to analyze scanning load.

After defining points, you might also want to fine-tune the scanning packets used in your server database.

You can use **lisscn** to list details about all scan packets configured in the database. This utility lists the scan packets in each scan period (interval) and a summary showing the number of scan packets per period and the scan packets per period per second.

For details about the lisscn utility, see the topic titled "lisscn" in the *Server and Client Configuration Guide*.

pntdel

You use the **pntdel** utility to delete all channel, controller, and point configuration details from the database.

CAUTION

Use **pntdel** with caution. Deleted configuration data cannot be recovered without rebuilding the database using hardware and point definition files. History data is also lost.

For details about the pntdel utility, see the "pntdel" topic in the *Configuration Guide*.

Example point definitions

This section provide example point definitions for the following point types:

Related topics

"Accumulator point" on page 82
"Analog points" on page 82
"Status point" on page 83

Accumulator point

```
&
DEL      FCE-KWH
ADD      FCE-KWH  ACC00001 FURNACE KWH
RANGE    FCE-KWH  0.0    1000.0000 KWH
PVSOURCE FCE-KWH  002 5001
PVPERIOD FCE-KWH  15
SCALE    FCE-KWH  2.0000
METER    FCE-KWH  1.1000
ROLOVR   FCE-KWH  300
PVALGO   FCE-KWH  5   56
ALG(01)  FCE-KWH  Z'0000'
ALG(02)  FCE-KWH  FCE-TOTAL&
```

Analog points

```
DEL      LC131
ADD      LC131    ANA00001 INPUT TEMP CONTROLLER
RANGE    LC131    O   3600.0000 DEG.C
PVSOURCE LC131    001 PV02
PVPERIOD LC131    5
CNTRLLVL LC131    50
GROUP    LC131    12   1
DISPLAY  LC131    310
OPSOURCE LC131    001 OP02
OPDESTIN LC131    001 OP02
OPPERIOD LC131    300
MDSOURCE LC131    001 MD02
MDDESTIN LC131    OO1 MD02
MDPERIOD LC131    300
MDNORMAL LC131    AUTO
CNTRLTO  LC131    03
REVERSE  LC131    N
ALMXCHG  LC131    YNY
HISTORY  LC131    SLOW
ALMLIMI  LC131    1000.0000 06 1
ALMLIM2  LC131    800.0000 08 3
ALMLIM3  LC131    3000.0000 07 1
ALMLIM4  LC131    3200.0000 09 3
SPSOURCE LC131    001 SP02
SPDESTIN LC131    001 SP02
SPPERIOD LC131    30
DRIFTDB  LC131    06
ALARMDB  LC131    13
CNTRLDB  LC131    06
PVCLAMP  LC131    Y
SPLIMIT  LC131    1000.0000 3000.0000
OPLIMIT  LC131    0.0   100.0
&
DEL      SINE
ADD      SINE     ANA00002  TEST SINEWAVE
PARENT   SINE     ASSET1
RANGE    SINE     0   100.0
PVSOURCE SINE     001 F:8 R:1 W:114 B:0 W:15
PVPERIOD SINE     5
```

```
ENTNAM   SINE     SINEWAVE
DRIFTDB  SINE     00
&
DEL      FCE-HRS
ADD      FCE-HRS  ANA00003  FCE PRSR HRS RUN
RANGE    FCE-HRS  0  100.0  HOURS
PVSOURCE FCE-HRS  001
PVPERIOD FCE-HRS  15
DRIGFTDB FCE-HRS  00
PVALGO   FCE-HRS  7  199
ALG(01)  FCE-HRS  Z'000C'
ALG(02)  FCE-HRS  FCE-PRESS
ALG(03)  FCE-HRS  FCE-SHIFT
ALG(O4)  FCE-HRS  1
```

Status point

```
&
DEL      FCE-PRESS
ADD      FCE-PRESS  STA00001  FCE OUTPUT PRSR
RANGE    FCE-PRESS  0.0  1.0
STATEDES FCE-PRESS  NORMAL HIGH
PVSOURCE FCE-PRESS  002  202
PVPERIOD FCE-PRESS  15
HISTORY  FCE-PRESS  SLOW
HISGATE  FCE-PRESS  DIGHIS  0
ALARM    FCE-PRESS  2  Y  Y
```

Point algorithms reference

This chapter contains reference information on the point algorithms available in Experion. It provides general information on the functions of each algorithm, and details of the **pntbld** structures that underlie each point algorithm.

The information in this chapter is primarily intended for those who:

- Are familiar with **pntbld** and algorithm parameters
- Need to understand the point definition file entries for troubleshooting purposes

Attention

Algorithms can only refer to numbered displays, not named displays. For further details, see the topic titled "Named versus numbered displays” in the *Display Building Guide*.

The values contained in this section cannot be substituted for *Word* values in File 31. If you need assistance while working with algorithms, contact your Honeywell Service Representative.

Related topics

“Configuring PV algorithms using pntbld” on page 86
“Configuring action algorithms using pntbld” on page 106
“About composite alarms” on page 123
“Handling errors in PV algorithms” on page 125
“Point build reference” on page 19

Configuring PV algorithms using pntbld

The following PV algorithms are available using **pntbld**.

Related topics

"PV Algo 4: General Arithmetic" on page 86
"PV Algo 5: Production" on page 87
"PV Algo 7: Run Hours" on page 89
"PV Algo 10: General Logic" on page 91
"PV Algo 12: Composite Alarm Processing" on page 93
"PV Algo 15: Integration" on page 97
"PV Algo 16: Cyclic Task Request" on page 99
"PV Algo 20: Advanced Arithmetic" on page 100
"PV Algo 21: Advanced Logic" on page 101
"PV Algo 22: Piecewise Linearization" on page 101
"PV Algo 64: Maximum/Minimum" on page 103
"PV Algo 68: Value Transportation" on page 104
"PVALGO" on page 56
"Handling errors in PV algorithms" on page 125

PV Algo 4: General Arithmetic

Description

Performs an arithmetic calculation using seven input point parameters and six constants. The result of the calculation is stored in the PV of the point to which this algorithm is attached. This algorithm is used to perform derived calculations based on analog or status points.

The calculation is as follows:

$$\text{Result} = \frac{(F1 + F2 + F3 + F4)}{(F5 + F6)} \times F7$$

Where:

$$Fn = Constant\_n \times IP\_Point\_ID\_n.Param\_n$$

Syntax

```
ALG(01) Point_ID Constant_1
ALG(03) Point_ID IP_Point_ID_1 Param_1
ALG(04) Point_ID Constant_2
ALG(06) Point_ID IP_Point_ID_2 Param_2
ALG(07) Point_ID Constant_3
ALG(09) Point_ID IP_Point_ID_3 Param_3
ALG(10) Point_ID Constant_4
ALG(12) Point_ID IP_Point_ID_4 Param_4
ALG(13) Point_ID Constant_5
ALG(15) Point_ID IP_Point_ID_5 Param_5
ALG(16) Point_ID Constant_6
ALG(18) Point_ID IP_Point_ID_6 Param_6
ALG(19) Point_ID IP_Point_ID_7 Param_7
```

Remarks

- If `ALG(03)`, `ALG(06)`, `ALG(09)`, `ALG(12)`, `ALG(15)`, or `ALG(18)` is not specified (left blank), then F*n* = `Constant_n`.

- If F5 + F6 = zero, then the divisor is automatically set to 0.000001 to avoid a divide by zero.
- If no division is required, then *Constant_5* should be set to 0.0, and *Constant_6* should be set to 1.0. *IP_Point_ID_5* and *IP_Point_ID_6* should not be defined.
- The *PVSOURCE* entry for a point to which this algorithm is attached should contain *only* an RTU (controller) number.
- *Constant_7* is not user-definable and is always equal to 1.

Example

An example of a point definition file entry using *PV Algo 4: General Arithmetic*.

```
&
&  GENERAL ARITHMETIC ALGO
&
DEL      TestPoint
ADD      TestPoint   ANA00049  GEN ARITH ALGO
RANGE    TestPoint   0.0       500.0     %
PVSOURCE TestPoint   001
PVPERIOD TestPoint   015
PVALGO   TestPoint   004              200
&
&                algorithm Block
&
ALG(01)  TestPoint   2.5
ALG(03)  TestPoint   AlgoInputPt_1   PV
ALG(04)  TestPoint   1.2
ALG(06)  TestPoint   AlgoInputPt_2   PV
ALG(07)  TestPoint   -0.9
ALG(09)  TestPoint   AlgoInputPt_3   PV
ALG(10)  TestPoint   101.5
ALG(12)  TestPoint   AlgoInputPt_4   PV
ALG(13)  TestPoint   3.4
ALG(15)  TestPoint   AlgoInputPt_5   PV
ALG(16)  TestPoint   6.5
ALG(18)  TestPoint   AlgoInputPt_6   PV
ALG(19)  TestPoint   AlgoInputPt_7   PV
&
END
```

PV Algo 5: Production

Description

This algorithm stores the shift, daily, or monthly total of an accumulator point PV to the nominated parameter of the destination point. After the total is stored, the accumulator point PV can optionally be reset to zero.

Syntax

`ALG(01)` *`Point_ID`* `Z'`*`nnnn`*`'`

Part	Description
Point_ID	The name of the point to which the algorithm is attached.
Z'*nnnn*'	The *storage period* and *reset value*, where *nnnn* is a value determined from “Table 1: Defining storage period and reset value”.

Table 1: Defining storage period and reset value

nnnn	Reset value	Storage period
0000	Enabled	Shift
0001	Enabled	Day

nnnn	Reset value	Storage period
0002	Enabled	Month
0004	Disabled	Shift
0005	Disabled	Day
0006	Disabled	Month

```
ALG(02) Point_ID Destination_Point_ID Destination_Parameter
```

Part	Description
Point_ID	The name of the point to which the algorithm is attached.
Destination_Point_ID Destination_Parameter	The point and parameter ID that is used to store the value at the end of the storage period. This must be an analog parameter.

Remarks

- The point to which this algorithm is attached *must* be an accumulator point.
- The accumulator point for which this algorithm is being defined must have either a database or controller address, and a PV scan period 60 seconds or less.
- When **Reset PV** is enabled, the value stored at the end of the shift, day, or month is the value of the accumulator PV, which is then reset to zero.

  If you specify either `Day` or `Month`, the storage will happen at the commencement of the first shift of that day or month (not at midnight).

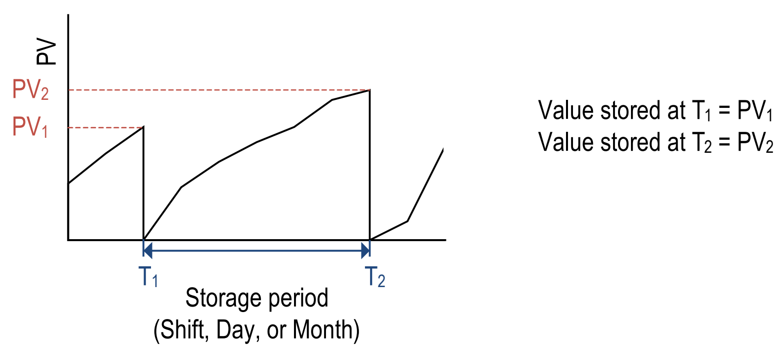

Figure 1: Reset PV enabled

- When **Reset PV** is not enabled, the value stored at the end of the shift, day, or month is equal to the current value minus the value from the previous period. That is, the delta value is stored.

  If you specify either `Day` or `Month`, the storage will happen at the commencement of the first shift of that day or month (not at midnight).

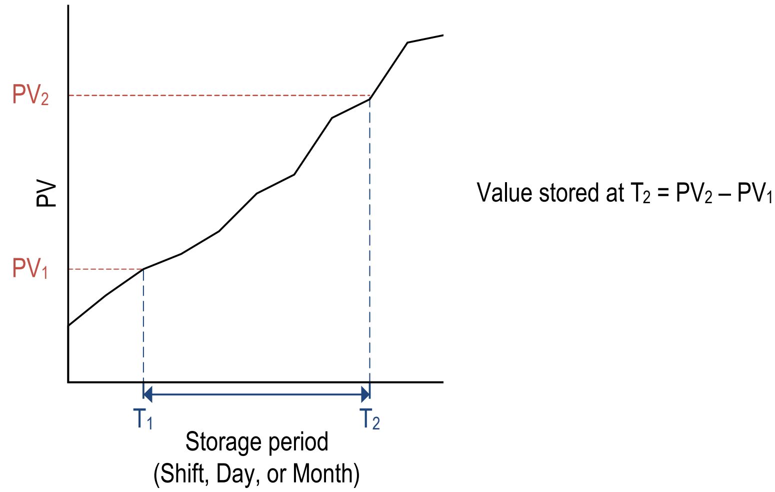

Figure 2: Reset PV not enabled

CAUTION

If a user changes either the **Storage Period** value or the **Reset PV at end of period** check box from the Algorithm's detail display on a Station, the value stored to the destination point parameter at the start of the next Storage Period will be incorrect. The algorithm will perform as expected from the start of the following Storage Period.

Example

An example of a point definition file entry using *PV Algo 5: Production*.

```
&
DEL      TestPoint
ADD      TestPoint     ACC00005     PRODUCTION ALGO POINT
RANGE    TestPoint     0.0          500.0     %
PVSOURCE TestPoint     002 CI0104
PVPERIOD TestPoint     015
PVALGO   TestPoint     005  057
&
    Algorithm Block
&
ALG(01)  TestPoint     Z'0001'
&        save at start of day to PV of TestPointDay
ALG(02)  TestPoint     TestPointDay  PV
&
END
```

PV Algo 7: Run Hours

Description

Performs run hours accumulation when the gating point parameter is in the nominated state.

At the end of a shift, the current value of run hours is output to the destination point parameter. If store shift total is enabled, the incremental run hours since the last shift boundary will be stored. The run hours can be manually reset at any time. The timestamp of the reset may be optionally written into the descriptor of the destination point. An optional associated point can be attached to record its PV at the last run hours reset.

The run hours determination is made as follows:

$$HRUN = HRUN + (TIME - OLDTIM) \times RUNID$$

Part	Description
HRUN	Run hours total
TIME	Current Time (hours)
OLDTIM	Time of last processing (hours)
RUNIND	Run indicator status point value (= 0 or 1)

This algorithm accumulates run hours according to a given run indicator status point. The accumulated run hours are stored in the PV of the point the algorithm is attached to. It will continue to accumulate until it is manually reset from the Algorithm Detail display.

On a shift boundary, the algorithm will move either the total accumulated run hours or the run hours accumulated in the last shift to the nominated destination point.

Algorithm details

- ALG(01)

BIT 2	If set to 1, then the reset date/time (ASCII) is stored in the descriptor of the destination point.
BIT 3	If set to 1, at the end of a shift, the shift total is stored in the destination point parameter instead of the current value of run hours.

- ALG(02)

  Gating point ID and parameter.
- ALG(04)

  Run state for input point.
- ALG(12)

  Associated point ID (for extraction of production tonnages, for example).
- ALG(21)

  Destination point ID and parameter into which the shift value will be stored.
- Algo Block

  The results of the calculations are stored in the algorithm block, as follows:
- ALG(5, 6)

  System time of last reset (real, seconds since midnight)
- ALG(7, 8)

  Value (real) in hours since last reset
- ALG(9, 10)

  Value (integer*4) in seconds since last reset
- ALG(11)

  System date of last reset (integer*2, Julian)
- ALG(13, 14)

  System time of last scan (real, seconds since midnight)
- ALG(15, 16)

  Total at last reset
- ALG(17, 18)

  PV of associated point at last reset of run hours (real)
- ALG(19, 20)

  Current PV of associated point (real)

Remarks

- This algorithm must be attached to an analog point that has no PVSOURCE address (for example, controller number only).
- Start and stop states may be single-, dual-, or triple-bit.
- The point parameter that is to contain the result of the time accumulation, must be built with a zero drift deadband to ensure processing each scan.
- If required, the ASCII date and time (20 characters) of last reset can be written into the descriptor of the destination point.
- The run hours can be reset at any time from the algorithm detail display.
- The point to which this algorithm is attached should have a scan period *no longer than 60 seconds*.
- Changing the system time will affect the values calculated by this algorithm.
- The Z'*xxxx*' notation is used to represent hexadecimal values.

Example

An example of a point definition file entry using *PV Algo 7: Run Hours*.

```
& Hours Run algorithm
&
DEL       ALGO7P1
ADD       ALGO7P1  ANA00071  HOURS  RUN  ALGO
RANGE     ALGO7P1  0.0       10000.0
PVSOURCE  ALGO7P1  001
PVPERIOD  ALGO7P1  15
DRIFTDB   ALGO7P1  0
&
&  algorithm Block
&
&  switch Point - ALGO7P2
&  shift total  - ALGO7P3
&
PVALGO    ALGO7P1  007 199
& set up for PV destination
& gating Point parameters and
& store shift total
ALG(01)  ALGO7P1   Z'0008'
ALG(02)  ALGO7P1   ALGO7P2  PV
ALG(04)  ALGO7P1   1
ALG(21)  ALGO7P1   ALGO7P3  PV
& END
```

PV Algo 10: General Logic

Description

Performs a logical combination of up to five single-bit inputs through four logic gates. Output is optionally sent to the output destination of the point to which the algorithm is attached and/or the specified parameter of a database point.

Optionally, a delay may be applied to the output after a transition to the target delay state. The specified delay can be relative to the current time or it can be absolute compared to the system time.

Up to five inputs are passed through logic functions. The six function types supported are:

- 0 (OR)
- 1 (NOR)
- 2 (AND)
- 3 (NAND)
- 4 (XOR)
- 5 (NXOR)

If less than five inputs are requested, the unused inputs default to a value of zero and the unused functions default to logical OR. In this way, they have no effect on the output (note that input A must be assigned). The result of the final logic function F4 is passed to the delay function.

If the delay is configured, then a transition from non-target state to target state causes the time delay function to be initiated. If the target state is held until the delay time has expired, then the delay output becomes the target state. Note that transition to a non-target state has no effect. The time delay may be relative (to current time) or absolute (seconds after midnight).

Algorithm details

- ALG(01)

  Four logic functions to be performed, one defined by each hex digit. For example HEX '0123' (entry Z'0123') gives F1=3, F2=2, F3=1, F4=0.
- ALG(02)

  Algorithm control flags as follows:

BIT 11		target state for delay
BIT 12		output to database
BIT 13		output to this point's OP destination
BIT 14	1	delay time is absolute
	0	delay time is relative
BIT 15		Enable delay function

- ALG(03)

  Database point and parameter
- ALG(05)

  Input A point ID and parameter
- ALG(07)

  Input B point ID and parameter
- ALG(09)

  Input C point ID and parameter
- ALG(11)

  Input D point ID and parameter
- ALG(13)

  Input E point ID and parameter
- ALG(15)

  Relative or Absolute Delay time in seconds (real)

Remarks

- The point to which this algorithm is attached must be a single-bit status point with only a controller number in the PVSOURCE entry.

Example

An example of a point definition file entry using *PV Algo 10: General Logic*.

```
&  General Logic algorithm
&
DEL      ALGO10
ADD      ALGO10     STA00616  GENERAL LOGIC ALGO
RANGE    ALGO10     0         7.0
```

```
STATEDES ALGO10     STAT00 STAT01
PVSOURCE ALGO10     002
PVPERIOD ALGO10     15
OPWIDTH  ALGO10     1
OPDESTIN ALGO10     002 19
&
&         algorithm Block
PVALGO   ALGO10     010 098
& F1 = OR  F2 = NOR  F3 = AND  F4 = NAND
ALG(01)  ALGO10     Z'3210'
& disable delays, output OPDESTIN and output to database
ALG(02)  ALGO10     Z'3000'
ALG(03)  ALGO10     ALGO10P6   OP
ALG(05)  ALGO10     ALGO10P1   PV
ALG(07)  ALGO10     ALGO10P2   PV
ALG(09)  ALGO10     ALGO10P3   PV
ALG(11)  ALGO10     ALGO10P4   PV
ALG(13)  ALGO10     ALGO10P5   PV
```

PV Algo 12: Composite Alarm Processing

Monitors and reports alarm conditions in several diverse plant locations through a central alarm reporting structure.

This algorithm is used in conjunction with *Action Algo 11: Composite Alarm* to complete the composite alarm structure. For an overview, see the topic titled "About composite alarms."

Attention
Composite alarming is only supported for legacy systems. Newer systems should use Alarm Groups instead.

After initiation of processing, a point to which this algorithm is attached checks the alarm condition of all subordinate points configured on the level immediately below it and changes its own state to be equivalent to the highest alarm state encountered.

The master point reflects the most severe (highest) alarm state found in any of its subordinate points. The possible master point states are:

- State 0 = Normal
- State 1 = Normal with previously acknowledged alarm
- State 2 = In alarm and acknowledged
- State 3 = In alarm and unacknowledged

Algorithm details

- ALG(01)

  This point's ID.
- ALG(02)-(19)

  All subordinate points (up to 18) on the level immediately below. The last entry terminating the list (02-20) must have the entry Z'0000'. (The Z'xxxx' notation is used to represent hexadecimal values.)

Remarks

- The point this algorithm is attached to must have the following attributes:
  - Dual-bit status point
  - No PVPERIOD (xxPERIOD) entry
  - Alarming inhibited
  - No hardware source address (controller number only in PVSOURCE (xxSOURCE) entry)
- This algorithm may not be attached to points at the lowest level of the hierarchy (Field points).
- The algorithm block number must be the same as that of each algorithm 11 used by all points listed in ALG(02)-(19). No other points may use this block number.

- This is a PV algorithm. Only the PVALGO entry may be used. The ACTALGO entry must not be used.

Example

An example of a point definition file entry using PV Algo 12: Composite Alarm Processing.

```
&
D&
DEL       PNT2-2-2-1
ADD       PNT2-2-2-1     STA00234
RANGE     PNT2-2-2-1     0.0       3.0
PVSOURCE  PNT2-2-2-1     001 P:PNT2-2-2-1 OP
PVPERIOD  PNT2-2-2-1     5
STATEDES  PNT2-2-2-1     ZERO   ONE    TWO    THREE
&
ACTALGO   PNT2-2-2-1     11   106
ALG(01)   PNT2-2-2-1     SECT2-2-1
ALG(02)   PNT2-2-2-1     PNT1-2-2-1
ALG(03)   PNT2-2-2-1     PNT2-2-2-1
ALG(04)   PNT2-2-2-1     Z'0000'
&
ALARM     PNT2-2-2-1     3 X N NNYYNNNN
&
DEL       PNT1-2-2-1
ADD       PNT1-2-2-1     STA00233
RANGE     PNT1-2-2-1     0.0        3.0
PVSOURCE  PNT1-2-2-1     001 P:PNT1-2-2-1 OP
PVPERIOD  PNT1-2-2-1     5
STATEDES  PNT1-2-2-1     ZERO   ONE    TWO    THREE
&
ACTALGO   PNT1-2-2-1     11  106
ALG(01)   PNT1-2-2-1     SECT2-2-1
ALG(02)   PNT1-2-2-1     PNT1-2-2-1
ALG(03)   PNT1-2-2-1     PNT2-2-2-1
ALG(04)   PNT1-2-2-1     Z'0000'
&
ALARM     PNT1-2-2-1     3 X N NNYYNNNN
&
DEL       SECT2-2-1
ADD       SECT2-2-1   STA00232
RANGE     SECT2-2-1   0.0       3.0
PVSOURCE  SECT2-2-1   001
STATEDES  SECT2-2-1   NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO   SECT2-2-1   11  102
ALG(01)   SECT2-2-1   FLOOR2-1
ALG(02)   SECT2-2-1   SECT1-2-1
ALG(03)   SECT2-2-1   SECT2-2-1
ALG(04)   SECT2-2-1   Z'0000'
&
PVALGO    SECT2-2-1   12  106
ALG(01)   SECT2-2-1   SECT2-2-1
ALG(02)   SECT2-2-1   PNT1-2-2-1
ALG(03)   SECT2-2-1   PNT2-2-2-1
ALG(04)   SECT2-2-1   Z'0000'
&
DEL       PNT2-1-2-1
ADD       PNT2-1-2-1     STA00231
RANGE     PNT2-1-2-1     0.0  3.0
PVSOURCE  PNT2-1-2-1     001  P:PNT2-1-2-1  OP
PVPERIOD  PNT2-1-2-1     5
STATEDES  PNT2-1-2-1     ZERO  ONE   TWO  THREE
&
ACTALGO   PNT2-1-2-1     11   105
ALG(01)   PNT2-1-2-1     SECT1-2-1
ALG(02)   PNT2-1-2-1     PNT1-1-2-1
ALG(03)   PNT2-1-2-1     PNT2-1-2-1
ALG(04)   PNT2-1-2-1     Z'0000'
&
ALARM     PNT2-1-2-1     3 X N NNYYNNNN
&
DEL       PNT1-1-2-1
ADD       PNT1-1-2-1     STA00230
RANGE     PNT1-1-2-1     0.0       3.0
PVSOURCE  PNT1-1-2-1     001 P:PNT1-1-2-1 OP
PVPERIOD  PNT1-1-2-1     5
STATEDES  PNT1-1-2-1     ZERO  ONE   TWO   THREE
&
ACTALGO   PNT1-1-2-1     11  105
ALG(01)   PNT1-1-2-1     SECT1-2-1
```

```
ALG(02)    PNT1-1-2-1     PNT1-1-2-1
ALG(03)    PNT1-1-2-1     PNT2-1-2-1
ALG(04)    PNT1-1-2-1     Z'0000'
&
ALARM      PNT1-1-2-1     3 X N NNYYNNNN
&
DEL        SECT1-2-1
ADD        SECT1-2-1   STA00229
RANGE      SECT1-2-1   0.0       3.0
PVSOURCE   SECT1-2-1   001
STATEDES   SECT1-2-1   NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO    SECT1-2-1   11  102
ALG(01)    SECT1-2-1   FLOOR2-1
ALG(02)    SECT1-2-1   SECT1-2-1
ALG(03)    SECT1-2-1   SECT2-2-1
ALG(04)    SECT1-2-1   Z'0000'
&
PVALGO     SECT1-2-1   12  105
ALG(01)    SECT1-2-1   SECT1-2-1
ALG(02)    SECT1-2-1   PNT1-1-2-1
ALG(03)    SECT1-2-1   PNT2-1-2-1
ALG(04)    SECT1-2-1   Z'0000'
&
DEL        FLOOR2-1
ADD        FLOOR2-1    STA00228
RANGE      FLOOR2-1    0.0    3.0
PVSOURCE   FLOOR2-1    001
STATEDES   FLOOR2-1    NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO    FLOOR2-1    11  100
ALG(01)    FLOOR2-1    BUILDING1
ALG(02)    FLOOR2-1    FLOOR1-1
ALG(03)    FLOOR2-1    FLOOR2-1
ALG(04)    FLOOR2-1    Z'0000'
&
PVALGO     FLOOR2-1    12  102
ALG(01)    FLOOR2-1    FLOOR2-1
ALG(02)    FLOOR2-1    SECT1-2-1
ALG(03)    FLOOR2-1    SECT2-2-1
ALG(04)    FLOOR2-1    Z'0000'
&
DEL        PNT2-2-1-1
ADD        PNT2-2-1-1     STA00227
RANGE      PNT2-2-1-1     0.0       3.0
PVSOURCE   PNT2-2-1-1     001 P:PNT2-2-1-1 OP
PVPERIOD   PNT2-2-1-1     5
STATEDES   PNT2-2-1-1     ZERO   ONE    TWO    THREE
&
ACTALGO    PNT2-2-1-1     11  104
ALG(01)    PNT2-2-1-1     SECT2-1-1
ALG(02)    PNT2-2-1-1     PNT1-2-1-1
ALG(03)    PNT2-2-1-1     PNT2-2-1-1
ALG(04)    PNT2-2-1-1     Z'0000'
&
ALARM      PNT2-2-1-1     3 X N NNYYNNNN
&
DEL        PNT1-2-1-1
ADD        PNT1-2-1-1     STA00226
RANGE      PNT1-2-1-1     0.0         3.0
PVSOURCE   PNT1-2-1-1     001 P:PNT1-2-1-1 OP
PVPERIOD   PNT1-2-1-1     5
STATEDES   PNT1-2-1-1     ZERO   ONE    TWO    THREE
&
ACTALGO    PNT1-2-1-1     11  104
ALG(01)    PNT1-2-1-1     SECT2-1-1
ALG(02)    PNT1-2-1-1     PNT1-2-1-1
ALG(03)    PNT1-2-1-1     PNT2-2-1-1
ALG(04)    PNT1-2-1-1     Z'0000'
&
ALARM      PNT1-2-1-1     3 X N NNYYNNNN
&
DEL        SECT2-1-1
ADD        SECT2-1-1   STA00225
RANGE      SECT2-1-1   0.0       3.0
PVSOURCE   SECT2-1-1   001
STATEDES   SECT2-1-1   NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO    SECT2-1-1   11  101
ALG(01)    SECT2-1-1   FLOOR1-1
ALG(02)    SECT2-1-1   SECT1-1-1
ALG(03)    SECT2-1-1   SECT2-1-1
```

```
ALG(04)   SECT2-1-1   Z'0000'
&
PVALGO    SECT2-1-1   12  104
ALG(01)   SECT2-1-1   SECT2-1-1
ALG(02)   SECT2-1-1   DR1-2-1-1
ALG(03)   SECT2-1-1   DR2-2-1-1
ALG(04)   SECT2-1-1   Z'0000'
&
DEL       PNT2-1-1-1
ADD       PNT2-1-1-1     STA00224
RANGE     PNT2-1-1-1     0.0       3.0
PVSOURCE  PNT2-1-1-1     001 P:PNT2-1-1-1 OP
PVPERIOD  PNT2-1-1-1     5
STATEDES  PNT2-1-1-1     ZERO   ONE    TWO    THREE
&
ACTALGO   PNT2-1-1-1     11  103
ALG(01)   PNT2-1-1-1     SECT1-1-1
ALG(02)   PNT2-1-1-1     PNT1-1-1-1
ALG(03)   PNT2-1-1-1     PNT2-1-1-1
ALG(04)   PNT2-1-1-1     Z'0000'
&
ALARM     PNT2-1-1-1     3   N NNYYNNNN
&
DEL       PNT1-1-1-1
ADD       PNT1-1-1-1     STA00223
RANGE     PNT1-1-1-1     0.0       3.0
PVSOURCE  PNT1-1-1-1     001 P:PNT1-1-1-1 OP
PVPERIOD  PNT1-1-1-1     5
STATEDES  PNT1-1-1-1     ZERO   ONE    TWO    THREE
&
ACTALGO   PNT1-1-1-1     11  103
ALG(01)   PNT1-1-1-1     SECT1-1-1
ALG(02)   PNT1-1-1-1     PNT1-1-1-1
ALG(03)   PNT1-1-1-1     PNT2-1-1-1
ALG(04)   PNT1-1-1-1     Z'0000'
&
ALARM     PNT1-1-1-1     3 X N NNYYNNNN
&
DEL       SECT1-1-1
ADD       SECT1-1-1   STA00222
RANGE     SECT1-1-1   0.0       3.0
PVSOURCE  SECT1-1-1   001
STATEDES  SECT1-1-1   NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO   SECT1-1-1   11  101
ALG(01)   SECT1-1-1   FLOOR1-1
ALG(02)   SECT1-1-1   SECT1-1-1
ALG(03)   SECT1-1-1   SECT2-1-1
ALG(04)   SECT1-1-1   Z'0000'
&
PVALGO    SECT1-1-1   12  103
ALG(01)   SECT1-1-1   SECT1-1-1
ALG(02)   SECT1-1-1   PNT1-1-1-1
ALG(03)   SECT1-1-1   PNT2-1-1-1
ALG(04)   SECT1-1-1   Z'0000'
&
DEL       FLOOR1-1
ADD       FLOOR1-1    STA00221
RANGE     FLOOR1-1    0.0       3.0
PVSOURCE  FLOOR1-1    001
STATEDES  FLOOR1-1    NORMAL UNACK  ALACK  ALUNAK
&
ACTALGO   FLOOR1-1    11  100
ALG(01)   FLOOR1-1    BUILDING1
ALG(02)   FLOOR1-1    FLOOR1-1
ALG(03)   FLOOR1-1    FLOOR2-1
ALG(04)   FLOOR1-1    Z'0000'
&
PVALGO    FLOOR1-1    12  101
ALG(01)   FLOOR1-1    FLOOR1-1
ALG(02)   FLOOR1-1    SECT1-1-1
ALG(03)   FLOOR1-1    SECT2-1-1
ALG(04)   FLOOR1-1    Z'0000'
&
DEL       BUILDING1
ADD       BUILDING1   STA00220
RANGE     BUILDING1   0.0       3.0
PVSOURCE  BUILDING1   001
STATEDES  BUILDING1   NORMAL UNACK  ALACK  ALUNAK
&
PVALGO    BUILDING1   12  100
```

```
ALG(01)   BUILDING1   BUILDING1
ALG(02)   BUILDING1   FLOOR1-1
```

Related topics

“Action Algo 11: Composite Alarm” on page 106
“About composite alarms” on page 123
“xxSOURCE” on page 70
“xxPERIOD” on page 69
“PVALGO” on page 56
“ACTALGO” on page 25

PV Algo 15: Integration

Description

Provides integration that is periodically performed on the nominated parameter of an input point providing totals and predicted totals. Period totals may be configured to reset at the end of the period and output to the defined destinations. One of the period totals may be optionally stored in the PV of the point to which this algorithm is attached. Used for integration of rates to obtain totals and the calculation of predicted totals.

The calculations are performed as follows:

```
NEW TOTAL = VALUE x F x T + OLD TOTAL
PREDICTED = NEW TOTAL + (VALUE x F x TIME REMAINING)
```

Part	Description
NEW TOTAL	The new running total
VALUE	The most recent value of point being integrated (units/time)
F	The scale factor (to convert rate to units)
T	The time between scans (in seconds)
OLD TOTAL	The last running total
PREDICTED	The predicted total
TIME REMAINING	The remaining time in interval (seconds)

Integration is performed on a shift, day, and month basis. The PV of the point to which the algorithm is attached may or may not contain any of these values.

Predicted values and shift, day, and month values are available in the algorithm block. The shift, day, and month values can be downloaded to other point parameters and can be reset as defined below.

The point to which this algorithm is attached should have a scan period *no longer than 60 seconds*.

Algorithm details

- ALG(01)

BIT 0	Reset shift value	0	No
		1	Yes
BIT 1	Reset day value	0	No
		1	Yes
BIT 2	Reset month value	0	No

		1	Yes
BIT 4,5	Total is not stored in PV	0	
	Shift value is stored in PV	1	
	Day value is stored in PV	2	
	Month value is stored in PV	3	

- ALG(03)

  Input point ID and parameter to be integrated (0 for this point PV)
- ALG(05)

  Destination point ID and parameter for shift value
- ALG(07)

  Destination point ID and parameter for day value
- ALG(09)

  Destination point ID and parameter for month value
- ALG(11)

  Scale factor (real)
- Algo Block

  The results of the calculations are stored in the algorithm block as follows:
- ALG(13)

  Shift value (real)
- ALG(15)

  Day value (real)
- ALG(17)

  Month value (real)
- ALG(19)

  Predicted shift value (real)
- ALG(21)

  Predicted day value (real)
- ALG(23)

  Predicted month value (real)

Remarks

- The point parameter that is to contain the result of the time accumulation, must be built with a zero drift deadband to ensure processing at each scan.
- The point to which the algorithm is attached must be an analog point with no database or hardware source address (controller number only).
- The Z'xxxx' notation is used to represent hexadecimal values.

Example

An example of a point definition file entry using PV Algo 15: Integration.

```
& Integration algorithm
&
DEL       ALGO15P1
ADD       ALGO15P1    ANA00052  INTEGRATION ALGO
RANGE     ALGO15P1    0.0       60000.0 KG
PVSOURCE  ALGO15P1    001
PVPERIOD  ALGO15P1    15
```

```
DRIFTDB   ALGO15P1    0
&
&    algorithm Block
&     reset shift, day and month values
&     store all values in PV
&
PVALGO    ALGO15P1    015  197
& reset shift, day and month values and
& store shift value in PV
ALG(01)   ALGO15P1    Z'0017'
ALG(03)   ALGO15P1    ALGO15P2  PV
ALG(05)   ALGO15P1    ALGO15P3  PV
ALG(07)   ALGO15P1    ALGO15P4  PV
ALG(09)   ALGO15P1    ALGO15P5  PV
& use PV parameters of input and
destination Points
ALG(11)   ALGO15P1    2.5
END
```

PV Algo 16: Cyclic Task Request

Description

Enables an operator to request a task on a regular basis.

While a point with this algorithm is *on scan*, the application task with the specified LRN is activated on a regular basis.

For details about using this algorithm to request a custom task, see the “Activating a task while a point is on scan” topic in the *Application Development Guide*.

Algorithm details

To define the algorithm in a point definition file, include the following entries:

```
PVALGO    pointid     016         block
ALG(01)   pointid     lrn
ALG(02)   pointid     reqrate
ALG(06)   pointid     param1
ALG(07)   pointid     param2
. . .
ALG(15)   pointid     param10
```

Part	Description
pointid	The point ID of the point to which the algorithm is to be attached.
block	The algorithm data block number.
lrn	The logical resource number of the task to be requested.
param1	Must be a non-zero number.
param2-10	Are numerical parameters that are to be passed to the task.
reqrate	The task request rate in seconds. This value must be a multiple of the point scan rate.

Remarks

- The algorithm block can also be configured from the Cyclic Task Request Algorithm display. Using Station, select the Algorithm number of the point's detail display, then select View, Detail from the pulldown menu.
- This algorithm must be attached to either a status or analog point with no database or hardware address (that is, controller number only).
- Time of the last request (in seconds) is stored by the system in ALG(04).

Example

An example of a point definition file entry using PV Algo 16: Cyclic Task Request.

```
DEL       ALGO16P1
ADD       ALGO16P1  STA0001  CYCL  TASK  REQ  TEST  PT
RANGE     ALGO16P1  0        1.0
STATEDES  ALGO16P1  STAT00   STAT01
PVSOURCE  ALGO16P1  001
PVPERIOD  ALGO16P1  15
PVALGO    ALGO16P1  016  26
ALG(01)   ALGO16P1  113
ALG(02)   ALGO16P1  30.0
ALG(06)   ALGO16P1  1
```

When activated using this method, your task can call GETREQ to obtain the following information in the parameter block.

Words 1-10

PV Algo 20: Advanced Arithmetic

Description

Performs arithmetic calculation of multiple input point parameters and constants. The result of the calculation is stored in the PV of the point for which this algorithm is being defined.

Syntax

"*Equation*"

where *Equation* is the calculation performed using the arithmetic operators listed in the remarks section.

Remarks

- Enclose the equation with quotation marks (" ").
- Maximum 1,000 characters.
- Arithmetic Operators are:
  - + (Plus Sign) Add
  - - (Minus Sign) Subtract
  - * (Asterisk) Multiply
  - / (Slash mark) Divide
  - ^ (Caret) Power
  - *sqrt()* Square root
- If part of a point name contains an arithmetic operator, use a backslash (\) to escape the character. For example, *fic-123.pv* can be used if entered as *fic\-123.pv*.

Example

An example of a point definition file entry for the point *FIC001* using *PV Algo 20: Advanced Arithmetic*.

```
PVALGO    FIC001 20 0
ALG(01)   FIC001 "(Sqrt(FIC004.DACA.PV))*10"
```

PV Algo 21: Advanced Logic

Description

Performs logical combination of multiple single-bit inputs. The result of the calculation is stored in the PV of the point for which this algorithm is being defined. If the result is more than 3 bits, then attach the algorithm to an Analog point.

Syntax

"`Equation`"

where `Equation` is the calculation performed using the logic operators listed in the remarks section.

Remarks

- Enclose the equation with quotation marks (" ").
- Maximum 1,000 characters.
- Logic Operators are:
  - `+` (Plus Sign) Concatenation (shift left)
  - `&` (Ampersand) And
  - `|` (Pipe) Or
  - `^` (Caret) Exclusive or (Xor)
  - `~` (Tilde) Not
- If part of a point name contains a logic operator, use a backslash (\) to escape the character. For example, `di+123.pv` can be used if entered as `di\+123.pv`.

Example

An example of a point definition file entry for the point `FIC001` using *PV Algo 21: Advanced Logic*.

```
PVALGO    FIC001 21 0
ALG(01)   FIC001 "FIC004.Event_01.PVfl & FIC050.PV"
```

PV Algo 22: Piecewise Linearization

Description

Linearizes the PV and SP of the associated point using piecewise linearization of up to six segments. The segments are defined by breakpoints as assigned in the following chart. It is often used for linearization of thermocouple readings to produce true temperature.

The raw value (passed from the point being linearized) is converted by the use of linear interpolation between (up to) seven coordinates.

The end points of the graph are defined implicitly at the 0% and 100% of range values, with corresponding engineering unit (EU) values being the 0% of range and 100% of range values defined for the point.

Up to five intermediate breakpoints along the graph are defined by placing coordinate pairs in the algorithm block. Should less than five breakpoints be required, the coordinates of the unused breakpoints must be set to 0, 0.

A negative gradient on the linearized graph can be obtained by defining appropriate values for the EU range and percentage range. If, for example, a graph with a positive gradient needed to be 'reversed' to have a corresponding negative gradient, this could be achieved by subtracting all EU values from the 100% EU value. Generally, a graph with a negative gradient will have EU values decreasing as percentage values increase.

Algorithm details

- ALG(01)

  Percentage of range value for first breakpoint
- ALG(03)

  EU value for first breakpoint
- ALG(05)

  Percentage of range value for second breakpoint
- ALG(07)

  EU value for second breakpoint
- ALG(09)

  Percentage of range value for third breakpoint
- ALG(11)

  EU value for third breakpoint
- ALG(13)

  Percentage of range value for fourth breakpoint
- ALG(15)

  EU value for fourth breakpoint
- ALG(17)

  Percentage of range value for fifth breakpoint
- ALG(19)

  EU value for fifth breakpoint

Remarks

- To ensure meaningful results, the graph must have either a positive gradient or a negative gradient but not a combination of both.
- Percentage coordinates must be in the range 0 to 100%. EU coordinates must be in the range of the point.
- A percentage coordinate with value 0% indicates the end of breakpoint data.
- This algorithm is also applied to the SP (Set point) for SP source input and a reverse calculation is performed on SP destination controls.
- This algorithm must be attached to an analog point with a hardware or database source address.

Example

An example of a point definition file entry using PV Algo 22: Piecewise Linearization.

```
&  PIECEWISE LINEARIZATION ALGORITHM
&
DEL        ALGO22
ADD        ALGO22      ANA00022    P-L POINT
RANGE      ALGO22      0.0         500.0   %
PVSOURCE   ALGO22      005 PV0201
PVPERIOD   ALGO22      900
PVALGO     ALGO22      022  056
&
&  algorithm Block
&
ALG(01)    ALGO22         0.8
ALG(03)    ALGO22         9.2
ALG(05)    ALGO22      10.0
ALG(07)    ALGO22      15.5
```

```
ALG(09)   ALGO22      11.0
ALG(11)   ALGO22      25.6
ALG(13)   ALGO22      21.2
ALG(15)   ALGO22      30.9
ALG(17)   ALGO22      26.7
ALG(19)   ALGO22      35.6
&
END
```

PV Algo 64: Maximum/Minimum

Description

Records the maximum and minimum values of the PV of a point and the times at which they occurred over a period of a shift or a day. These values can be stored at the nominated destinations and reset at the beginning of the selected period.

The current PV value of a point with this algorithm attached is compared with the recorded maximum and minimum values and a new maximum or minimum is set. The values can be reset at the beginning of a selected period. At the beginning of each period, the maximum and minimum values of the previous period are downloaded to the defined destination points.

Recording and downloading can be disabled by turning OFF the status of that period.

Algorithm details

The following parameters are set by the user.

- ALG(13)

  Destination point ID and parameter for shift maximum
- ALG(15)

  Destination point ID and parameter for shift minimum
- ALG(17)

  Destination point ID and parameter for daily maximum
- ALG(19)

  Destination point ID and parameter for daily minimum
- ALG(21)

BIT 0	Shift status	0	OFF
		1	ON
BIT 1	Day status	0	OFF
		1	ON
BIT 2	Reset after each shift	0	NO
		1	YES
BIT 3	Reset after each day	0	NO
		1	YES

Algo Block

The record number is the algo block assigned to the point, and the results of the calculations are stored in the algorithm block as follows. The algo blocks are stored in File 31.

The following parameters are stored in **real** format:

- WORD(01) - Maximum value in current shift

- WORD(03) - Minimum value in current shift
- WORD(05) - Maximum value in current day
- WORD(07) - Minimum value in current day

The following parameters are stored in **integer** format:

- WORD(09) - Time at which shift maximum recorded
- WORD(10) - Time at which shift minimum recorded
- WORD(11) - Time at which day maximum recorded
- WORD(12) - Time at which day minimum recorded

Remarks

- The algorithm parameters may be configured from the Max/Min Algorithm display.
- The algorithm is attached to an analog point.
- The Z'xxxx' notation is used to represent hex values.

Example

An example of a point definition file entry using PV Algo 64: Maximum/Minimum.

```
&
&  MAXIMUM / MINIMUM ALGORITHM
&
DEL       ALGO64
ADD       ALGO64      ANA00064    MAX/MIN POINT
RANGE     ALGO64      0.0         500.0   %
PVSOURCE  ALGO64      005  PV0202
PVPERIOD  ALGO64      900
PVALGO    ALGO64      064  097
&
&  algorithm Block
&
ALG(01)   ALGO64      27.8
ALG(03)   ALGO64      19.2
ALG(05)   ALGO64      75.0
ALG(07)   ALGO64      63.5
&           store to ALGO64OUT A1,A2,A3,A4
ALG(13)   ALGO64      ALGO64OUT  A1
ALG(15)   ALGO64      ALGO64OUT  A2
ALG(17)   ALGO64      ALGO64OUT  A3
ALG(19)   ALGO64      ALGO64OUT  A4
&           shift, day and reset after day
ALG(21)   ALGO64      Z'000B'
&
END
```

PV Algo 68: Value Transportation

Used to move a value from the PV of the point, to which the algorithm is attached, to the hardware address defined by the:

- OP destination (for a status point)
- SP destination (for an analog point)

CAUTION

Do not use this algorithm to transfer safety or mission critical information between controllers. To transfer this type of information use a method to transfer information directly between controllers.

Remarks

- If the algorithm is attached to a status point, the number of input states must match the number of output states. If this isn't the case, the algorithm won't be able to transfer data correctly and the following error will occur: `Data not convertible`.
- Use the algorithm with care—assigning it to many points may result in a significant load on the server. (As an alternative, you should consider the equivalent *Action Algo 68: Value Transportation*.)
- Set the drift deadband to a reasonable value so that the algorithm does not execute for inconsequential changes.
- Using this as a PV algorithm degrades system performance. Using OP/SP source for confirmation in conjunction with using this as a PV algorithm severely degrades system performance and is not recommended.
- Do not use control timeouts in conjunction with this algorithm.

Example

An example of a point definition file entry using PV Algo 68: Value Transportation.

```
&
& Value Transportation algorithm
&     Status Point
&     - change of PVSOURCE sent to OPDESTIN
&
DEL       ALGO68P1
ADD       ALGO68P1    STA00063    ALGO 68
RANGE     ALGO68P1    0.0  1.0
STATEDES  ALGO68P1    OFF  ON
PVSOURCE  ALGO68P1    079  001
PVPERIOD  ALGO68P1    15
OPDESTIN  ALGO68P1    002  19
PVALGO    ALGO68P1    068  0
&
&  Analog Point
&  - change of PVSOURCE sent to SPDESTIN
&
DEL       ALGO68P2
ADD       ALGO68P2    ANA00064  ALGO 68
RANGE     ALGO68P2    0.0       100.0
PVSOURCE  ALGO68P2    079  5001 U4095
PVPERIOD  ALGO68P2    15
SPDESTIN  ALGO68P2    002  18
PVALGO    ALGO68P2    068  0
&
END
```

Configuring action algorithms using pntbld

The following action algorithms are available using **pntbld**.

Related topics

"Action Algo 11: Composite Alarm" on page 106
"Action Algo 68: Value Transportation" on page 107
"Action Algo 69: Status Change Task Request" on page 108
"Action Algo 70: Status Change Report Request" on page 109
"Action Algo 71: Queued Task Request" on page 110
"Action Algo 72: Status Value Transportation with Mapping" on page 110
"Action Algo 74: Status Change USKB LED Request" on page 112
"Action Algo 75: Status Point Notification" on page 113
"Action Algo 76: Analog Point Notification" on page 115
"Action Algo 77: Status Change Display Request" on page 116
"Action Algo 78: Group Control of Points" on page 117
"Action Algo 79: Status Change Alarm Group Inhibit" on page 118
"Action Algo 80: Status Change Alarm Area Inhibit" on page 119
"Action Algo 92: Queued Task Request" on page 120
"ACTALGO" on page 25

Action Algo 11: Composite Alarm

Initiates the monitoring and reporting of alarm conditions in several diverse plant locations through a central alarm reporting structure.

This algorithm is used in conjunction with *PV Algo 12: Composite Alarm Processing* to complete the composite alarm structure. For an overview, see the topic titled "About composite alarms."

Attention
Composite alarming is only supported for legacy systems. Newer systems should use Alarm Groups instead.

It is used when a change of PV, a point to which this algorithm is attached, initiates processing of the master point.

The master point reflects the most severe (highest) alarm state found in any of its subordinate points. The possible master point states are:

- State 0 = Normal
- State 1 = Normal with previously unacknowledged alarm
- State 2 = In alarm and acknowledged
- State 3 = In alarm and unacknowledged

Algorithm details

- ALG(01)

  Master point ID
- ALG(02)-(19)

  List of all points (up to 18) that are on the level below the master. Include this point. The first entry terminating the list (02-20) must have the entry Z'0000'. (The format Z'xxxx' is used to represent hexadecimal values.)

Remarks

- This algorithm *must not* be attached to the highest point in the hierarchy.
- If attached to a field point, the point must be at the lowest level of the hierarchy and may be of any point type. Alarming should be permitted for the point.
- If attached to a point above the lowest level, the point must have the following attributes:
  - Dual bit status point
  - No PVPERIOD entry
  - Alarming inhibited

    No hardware source address (controller number only in PVSOURCE entry)
- The algorithm block number must be the same as that used with *PV Algo 12: Composite Alarm Processing* attached to the point above. The block number must be unique to this section of the hierarchy.
- This is an action algorithm. Only the ACTALGO entry may be used. The PVALGO entry *must not* be used.

Related topics

"PV Algo 12: Composite Alarm Processing" on page 93
"About composite alarms" on page 123

Action Algo 68: Value Transportation

This algorithm is used to move a value from the PV of the point, to which the algorithm is attached, to the hardware address defined by the:

- OP destination (for a status point)
- SP destination (for an analog point)

The algorithm is used to transport field and database inputs to output locations.

The drift deadband (DRIFTDB) should be set at a reasonable value so that the algorithm does not execute for inconsequential changes.

CAUTION

Do not use this algorithm to transfer safety or mission critical information between controllers. To transfer this type of information use a method to transfer information directly between controllers.

Remarks

- Always build with algorithm block 0.
- Using as a PV algorithm degrades system performance. Using OP/SP source for confirmation in conjunction with use as a PV algorithm severely degrades system performance and is not recommended.
- Control timeouts (CNTRLTO) should not be used in conjunction with this algorithm.
- This algorithm is attached to either an analog or status point.

Example

An example of a point definition file entry using Action Algo 68: Value Transportation.

```
&
& Value Transportation algorithm
&     Status Point
&     - change of PVSOURCE sent to OPDESTIN
&
DEL      ALGO68P1
ADD      ALGO68P1    STA00063   ALGO 68
RANGE    ALGO68P1    0.0  1.0
STATEDES ALGO68P1    OFF  ON
PVSOURCE ALGO68P1    079  001
PVPERIOD ALGO68P1    15
OPDESTIN ALGO68P1    002  19
```

```
ACTALGO  ALGO68P1    068  0
&
&  Analog Point
&  - change of PVSOURCE sent to SPDESTIN
&
DEL      ALGO68P2
ADD      ALGO68P2    ANA00064   ALGO 68
RANGE    ALGO68P2    0.0        100.0
PVSOURCE ALGO68P2    079  5001  U4095
PVPERIOD ALGO68P2    15
SPDESTIN ALGO68P2    002  18
ACTALGO  ALGO68P2    068  0
&
END
```

Action Algo 69: Status Change Task Request

This is an exception-based program request that is used to run a program under certain conditions.

The algorithm will make a single task request each time a status point makes the transition from non-nominated state to the nominated state.

Algorithm details

- ALG(01)

  Logical Resource Number (LRN) of the task to be requested
- ALG(02)

  Nominated state (0-7) or -1 for all state transitions
- ALG(03)

  Task request error status (information only)
- ALG(04-13)

  10-word request block (optional)

  Word 4 = 0 for no request block

  This block contains data in a format meaningful to the requested task. It has no fixed format.

  If the tasks are requested from the Server Display program (LRN 21), these fields represent the following values:

  word 4 = Station connection number

  word 5 = parameter 1

  word 6 = parameter 2

  word 7 = object ID

  word 8 = X coordinate

  word 9 = Y coordinate

  word 10 = X width

  word 11 = Y width

  word 12 = Reserved for Honeywell only

  word 13 = Reserved for Honeywell only

Remarks

- For details about using the algorithm to request a custom task, see the topic titled "Activating a task when a status point changes state" in the *Application Development Guide*.
- The parameters can also be configured, along with the other algorithm parameters, from the Status Change Task Request algorithm display.

- This algorithm is attached to a status point.
- For all task request algorithms, the following restrictions apply:
  - The request rate must be some multiple of the point scan rate.
  - The values specified in the request block must be numeric.
- The Z'xxxx' notation is used to represent hexadecimal values.

Example

An example of a point definition file entry using Action Algo 69: Status Change Task Request.

```
&
Status change task request algorithm to call p301 to Station 1 on
change of state to state 1
&
DEL      ALGO16P1
ADD      ALGO16P1  STA00069 STAT CHG TASK REQUEST
RANGE    ALGO16P1  0    1.0
STATEDES ALGO16P1
PVSOURCE ALGO16P1  001
PVPERIOD ALGO16P1  15
&
algorithm block
&
ACTALGO  ALGO16P1  069 193
ALG(01)  ALGO16P1  111
ALG(02)  ALGO16P1  -1
ALG(04)  ALGO16P1  20
ALG(05)  ALGO16P1  100
ALG(06)  ALGO16P1  1000
END
```

Action Algo 70: Status Change Report Request

This algorithm will request the specified report to be produced when the status point changes to the report request state.

It is used to request a report under certain conditions.

This algorithm will make a single report request each time a status point makes the transition from any non-nominated state to the nominated state.

Algorithm details

- ALG(01)

  Report Number to be requested
- ALG(02)

  Report request state (0-7) for triple bit input, (0-3) for dual bit input, (0-1) for single bit input.

Attention

This algorithm is attached to a status point.

Remarks

- In order for the report to print, when you configure the report definition details, under Periodic Reporting, you must specify a report printer as the destination.
- To limit the asset from which data is reported, specify the ID of an operator (under Periodic Reporting) who is assigned the assets you want to include.

Example

An example of a point definition file entry using Action Algo 70: Status Change Report Request.

The following example shows the use of the algorithm to demand report 2 on change of state to state 1.

The Z'xxxx' notation is used to represent hexadecimal values.

```
DEL       TEST1
ADD       TEST1  STA00000  TEST POINT
RANGE     0      1
STATEDES  OFF    ON
PVSOURCE  001    SI16
ACTALGO   70     3
ALG(01)   Z'0002'
ALG(02)   Z'0001'
```

Action Algo 71: Queued Task Request

This algorithm is deprecated, and is replaced by *Action Algo 92: Queued Task Request*.

Action Algo 72: Status Value Transportation with Mapping

This algorithm is attached to a status point, and will transport four specified values to (a maximum of) four separate points when the status point changes value.

When transport occurs, the states are set in four bit masks. Each bit mask has a value to be transported: a target point and a target parameter. The algorithm can write to status, analog, and accumulator points. This algorithm will also perform limited PV control.

Definition: F is the value of the attached status point.

For each bit mask defined in the algorithm block, if bit F is set, then the value associated with that bit mask is transported to the corresponding point and parameter.

Building the point record

ALG (Number)	Purpose	Data Type
1	Mask 1	Int*2
2-3	Value 1	Real
4	Point 1	Tag
5	Parameter 1	Int*2
6	Mask 2	Int*2
7-8	Value 2	Real
9	Point 2	Tag
10	Parameter 2	Int*2
11	Mask 3	Int*2
12-13	Value 3	Real
14	Point 3	Tag
15	Parameter 3	Int*2
16	Mask 4	Int*2
17-18	Value 4	Real
19	Point 4	Tag
20	Parameter 4	Int*2

ALG (Number)	Purpose	Data Type
Value is a real number to be transported *Point* is a database point tag (stored internally as an Int*2) *Parameter* is an integer whose values are defined “Parameter definitions.” *Mask* is an integer (or Hex) number whose binary representation is the bit mask		

Parameter definitions

The Z'xxxx' notation is used to represent hexadecimal values.

Parameter	Definition
Z'0000'	PV
Z'0001'	MD
Z'0002'	OP
Z'0003'	SP
Z'0004'	A1
Z'0005'	A2
Z'0006'	A3
Z'0007'	A4

PV control definitions

Value	Definition
0	Reset
1	On Line/In Service
2	Off Line/Out of Service

Mode definitions

Value	Definition
0	Man
1	Auto
2	Casc
3	Comp
4	S-Man
5	S-Auto
6	S-Casc
7	S-Comp

Example

An example of a point definition file entry using Action Algo 72: Status Value Transportation with Mapping.

```
DEL      ALGO72P1
ADD      ALGO72P1  STA00000  SMP B Point 1
RANGE    ALGO72P1  0        7.0
STATEDES ALGO72P1 ACSEAL - SCSEAL ISOLTD ACALRM ACTAMP
         SCALRM   SCTAMP
PVSOURCE ALGO72P1  001 1
```

```
PVPERIOD ALGO72P1  15
OPWIDTH  ALGO72P1  1
OPSOURCE ALGO72P1  001 5
OPDESTIN ALGO72P1  001 5
TARGET   ALGO72P1  0 2 0 0
ALARM    ALGO72P1  2 0 N NNNNYNYY
ACTALGO  ALGO72P1  72 100
&
ALG(01)  ALGO72P1  Z'0001'
&     When in State 0 (See Bit Mask Table)
ALG(02)  ALGO72P1  1.0
&     Transport a value of 1.0
ALG(04)  ALGO72P1  ALGO72P2
&     To the point ALGO72P2
ALG(05)  ALGO72P1  Z'0002'
&  and Parameter OP (See Parameter Table)
&
ALG(06)  ALGO72P1  Z'0008'
&     When in State 3 (See Bit Mask Table)
ALG(07)  ALGO72P1  2.0
&     Transport a value of 2.0
ALG(09)  ALGO72P1  ALGO72P3
&     To the point ALGO72P3
ALG(10)  ALGO72P1  Z'0000'
&     and Parameter PV (See Parameter Table)
&
ALG(11)  ALGO72P1  Z'0040'
&     When in State 6 (See Bit Mask Table)
ALG(12)  ALGO72P1  34.0
&     Transport a value of 34.0
ALG(14)  ALGO72P1  ALGO72P4
&     To the point ALGO72P4
ALG(15)  ALGO72P1  Z'0003'
&     and Parameter SP (See Parameter Table)
&
ALG(16)  ALGO72P1  Z'00A0'
&     When in State 5 or 7 (See Bit Mask Table)
ALG(17)  ALGO72P1  0.0
&     Transport a value of 0.0
ALG(19)  ALGO72P1  ALGO72P5
&     To the point ALGO72P5
ALG(20)  ALGO72P1  Z'0001'
&     and Parameter MD (See Parameter Table)
&
```

Example Bitmask

State	7	6	5	4	3	2	1	0
Example			x				x	
Binary			1				1	
Hex	2				2			
Example		x	x		x	x		x
Binary		1	1		1	1		1
Hex	6				D			

Action Algo 74: Status Change USKB LED Request

This algorithm is supported for backward compatibility only. For the current method for controlling LEDs, see the "Controlling LEDs on a specialized keyboard" section of the *Server and Client Configuration Guide*.

On status point change, the LED associated with a nominated push button on a Universal Station keyboard (USKB) is controlled with the characteristics specified for the defined location or workstation.

Used on status point change controls, the USKB LED associated with a push button (for example, turn on LED on alarm condition).

When a status point reaches one of up to eight states, the LEDs on a nominated button on a nominated USKB are controlled. The following LED characteristics are available: red, yellow, off, slow blink, fast blink, on.

Using pntbld to Define Algorithm Details

- ALG(01)

BITS 15-12	LED color and operation	0	no action
		4	red off
		5	red slow blink
		6	red fast blink
		7	red on
		C	yellow off
		D	yellow slow blink
		E	yellow fast blink
		F	yellow on
BITS 11-8	Point state at which action takes place		
BITS 7-0	USKB LED number		

- ALG(02)

BITS 15-12	not used		
BITS 11-8	Station affected	0	act on all screens assigned to location
		1	act on specified screen only
BITS 7-0	location or Station number	0	all screens

- Additional status actions may be specified by using the pairs:

  ALG(03), ALG(04)

  ALG(05), ALG(06)

  ALG(07), ALG(08)

  ALG(09), ALG(10)

  ALG(11), ALG(12)

  ALG(13), ALG(14)

  ALG(15), ALG(16)

Remarks

- The USKB is a console-mounted membrane keyboard.
- If the nominated Station does not have a USKB the LED control is ignored.
- This algorithm is attached to a status point.

Action Algo 75: Status Point Notification

This algorithm is included for backward compatibility. This algorithm will queue a message to be used by a user-written application program to notify an external computer of a change of point status to the nominated state, provided the gating point is in the permit state.

Attention
This algorithm is made available for customers upgrading to Experion from SCAN 3000 on VMS.

Algorithm details

- ALG(01)

  States to request task

  Bit 0 State 0

  Bit 1 State 1

  Bit 2 State 2

  Bit 3 State 3

  Bit 4 State 4

  Bit 5 State 5

  Bit 6 State 6

  Bit 7 State 7
- ALG(02)

  State 1 action
- ALG(03) to ALG(07)

  State n action
- ALG(08)

  State 7 action
- ALG(09)

  Gating point tag name

  Optional. If ALG(09) is not configured, then the gating point state is not checked however the task requests will always go through if the other conditions are correct.
- ALG(10)

  Gating point notification non-inhibit state
- ALG(11)

  LRN number to be notified
- ALG (12) to ALG (2)

  Optional algo date to pass om task

Remarks

- The change of state and application notification are recorded in the event file.
- Use of this algorithm requires a user-written application program to accept change of state notifications.
- The type and number of the gating point is found using TAGLOG.

Example

An example of a point definition file entry using Action Algo 75: Status Point Notification.

```
&
& STATUS POINT NOTIFICATION ALGORITHM
&
DEL       ALGO75
ADD       ALGO75    STA00075    PUMP
RANGE     ALGO75    0            1.0
STATEDES  ALGO75    OOF     ON
PVSOURCE  ALGO75    005 DI0102
```

```
ALARM     ALGO75    3  3  Y  NYN
&
& algorithm Block
&
ACTALGO   ALGO75    075 111
ALG(01)   ALGO75    0
ALG(02)   ALGO75    1
ALG(03)   ALGO75    0
ALG(04)   ALGO75    0
ALG(05)   ALGO75    0
ALG(06)   ALGO75    0
ALG(07)   ALGO75    0
ALG(08)   ALGO75    0
ALG(09)   ALGO75    GATEPOINT
ALG(10)   ALGO75    1
ALG(11)   ALGO75    2
END
```

Action Algo 76: Analog Point Notification

This algorithm is included for backward compatibility, when upgrading to Experion from SCAN 3000 on VMS.

A message is queued to a user-written application to notify an external computer:

- When the analog point changes by a specified percentage from the last value reported, or
- When the maximum time between notifications is exceeded (providing the gating point is in the permit state).

Use this algorithm to inform external computers of significant change to analog point PVs.

Algorithm details

- ALG(01)

  Percentage change (real)

- ALG(07)

  Maximum time (seconds) between notifications (0 = no notification)

- ALG(09)

  Gating point tag name

  Optional. If ALG(09) is not configured, then the gating point state is not checked, however the task requests will always go through if the other conditions are correct.

- ALG(10)

  Gating point notification non-inhibit state

- ALG(11)

  Point parameter in destination computer to be notified:

  0 = PV

  1 = MD

  2 = OP

  3 = SP

  4 = A1

  5 = A2

  6 = A3

  7 = A4

Algo Block

Results are stored in the algorithm block as follows:

- ALG(03)

  Last value reported (real)
- ALG(05)

  Time last value was reported (real)

Remarks

- The change of value and application notification are recorded in the Event file.
- Use of this algorithm requires a user-written application program to accept change of value notifications.

Example

An example of a point definition file entry using Action Algo 76: Analog Point Notification.

```
&
& ANALOG POINT NOTIFICATION ALGORITHM
&
DEL       ALGO76
ADD       ALGO76    ANA00076    PRESSURE
RANGE     ALGO76    0            100.0  PSI
PVSOURCE  ALGO76    005 AI0202
&
& algorithm Block
&
ACTALGO   ALGO76    076 112
ALG(01)   ALGO76    5.0
ALG(07)   ALGO76    300
ALG(09)   ALGO76    ALGO76GATE
ALG(10)   ALGO76    1
ALG(11)   ALGO76    0
END
```

Action Algo 77: Status Change Display Request

This algorithm will cause a display to appear either on a specified Station or all Stations assigned the specified assignable asset when the status point changes to a nominated state. A maximum of six display request states can be nominated.

Upon status point change, this algorithm will call up specified displays at specified Stations or locations.

Up to six (state, display, Station) triplets can be defined.

Algorithm details

- ALG(01)

  Bits 2-0 nominated state at which action takes place
- ALG(02)

  Bits 15-0 page number of the display to call up
- ALG(03)

Bits 8	Stations affected	0	act in all Stations assigned to location
		1	act on specified Stations only
Bits 7-0	Asset or Station number	0	all Stations

- Additional status actions may be specified by using the triplets:

Action 2	ALG(04), ALG(05), ALG(06)
Action 3	ALG(07), ALG(08), ALG(09)
Action 4	ALG(10), ALG(11), ALG(12)
Action 5	ALG(13), ALG(14), ALG(15)
Action 6	ALG(16), ALG(17), ALG(18)

Remarks

- It is recommended that for a particular state and Station, only one display is requested.
- To configure this algorithm, you need to know the number of the required assignable asset.

To obtain the asset number

- In the Station command zone, type the tag name of the asset and press F12.

Example

An example of a point definition file entry using Action Algo 77: Status Change Display Request.

The following example shows the use of the algorithm to demand page 302 to Station 1 on change to state 1.

The format Z 'xxxx' is used to represent hexadecimal values.

```
DEL       TEST1
ADD       TEST1   STA00000  TEST POINT
RANGE     0    7
STATEDES  OFF  ON
PVSOURCE  001  CS01
ACTALGO   77   10
ALG(01)   Z'0001'
ALG(02)   Z'012E'
ALG(03)   Z'0101'
```

Action Algo 78: Group Control of Points

This algorithm controls a group of analog and status points through a group control point. It does this by sending its PV to the nominated point/parameters in the control list.

You can optionally define a gating point (status) that prevents the PV being sent to the control points if it is in the nominated state.

Algorithm details

- ALG(01)

  Point ID of gating point
- ALG(02)

  Gating state
- ALG(03)

  Point ID and parameter of point 1 in group
- ALG(05)

  Point ID and parameter of point 2 in group
- ALG(n-2)

  The last point and parameter in the group (up to 15)
- ALG(n)

  -1

Remarks

- This algorithm is attached to a point with either a hardware or database address.
- The issuing of a large number of point controls will produce increased processor loading. The effects of daisy-chaining this algorithm should be carefully considered.

Example

An example of a point definition file entry using Action Algo 78: Group Control of Points.

```
&
& Group  Control algorithm
&
DEL       ALGO78P1
ADD       ALGO78P1  STA00010  Algo 78 Example
RANGE     ALGO78P1  0.0   1.0
PVSOURCE  ALGO78P1  002   CS01
&
& algorithm Block - to control Floor 2, Floor 3 if Floor 1 is not
in state 1 when CS01 changes state
&
ACTALGO   ALGO78P1  078 090
ALG(01)   ALGO78P1  XLP2.1
ALG(02)   ALGO78P1  1
ALG(03)   ALGO78P1  XLP2.2  OP
ALG(05)   ALGO78P1  XLP2.3  OP
ALG(07)   ALGO78P1  XLP2.4  OP
ALG(09)   ALGO78P1  -1
&
DEL       ALGO78P2
ADD       ALGO78P2  ANA00082   ALGO78P2
RANGE     ALGO78P2  0.0   4095.0
PVSOURCE  ALGO78P2  003   5001    U4095
PVPERIOD  ALGO78P2  15
&
& algorithm Block
&
ACTALGO   ALGO78P2  078 091
ALG(01)   ALGO78P2  FICVL1  SP
ALG(02)   ALGO78P2  2
ALG(03)   ALGO78P2  FICVL2  SP
ALG(05)   ALGO78P2  FICVL3  SP
ALG(07)   ALGO78P2  FICVL4  SP
ALG(09)   ALGO78P2  -1
END
```

Action Algo 79: Status Change Alarm Group Inhibit

Alarm reporting will be inhibited for the nominated group of points when the status point is in an alarm inhibit state.

This algorithm can be used to inhibit alarm reporting on sensor points if, for example, the unit has been shut down. It is used to inhibit alarm reporting for a group of points specified in the parameter block, depending on the nominated states.

Algorithm details

- ALG(01)

  Bits 0 to 7 of the 16-bit word represents a state of the point in which the alarms will be inhibited. For example, (using hexadecimal numbers):

  - Z'0001' will inhibit alarms for the state 0
  - Z'000A' will inhibit alarms for the states 1 and 3

  The Z'xxxx' notation is used to represent hexadecimal values.
- ALG(02)-ALG(20)

  List of up to 19 point IDs to be included in the group.

Remarks

- This algorithm does not affect the scanning of points. It simply ignores the alarm status being reported by the controller. If a point is in the alarm condition at the transition from inhibit to active, the point will not report the alarm state until either a background scan or the point restores to normal and then alarm again.

Example

An example of a point definition file entry using Action Algo 79: Status Change Alarm Group Inhibit.

```
&
& Alarm Group Inhibit algorithm
&
DEL       ALGO79P1
ADD       ALGO79P1    STA00100  STAT CHNG ALRM
RANGE     ALGO79P1    0    7.0
STATEDES  ALGO79P1    OFF  ON
PVSOURCE  ALGO79P1    001  CS01
PVPERIOD  ALGO79P1    015
&
& algorithm Block
&
& - inhibit alarming in states 1 and 3
&
ACTALGO   ALGO79P1    079 031
ALG(01)   ALGO79P1    Z'000A'
ALG(02)   ALGO79P1    FF79TST01
ALG(03)   ALGO79P1    FF79TST02
ALG(04)   ALGO79P1    FF79TST03
ALG(05)   ALGO79P1    FF79TST04
ALG(06)   ALGO79P1    FF79TST05
ALG(07)   ALGO79P1    FF79TST06
ALG(08)   ALGO79P1    FF79TST07
ALG(09)   ALGO79P1    FF79TST08
ALG(10)   ALGO79P1    FF79TST09
ALG(11)   ALGO79P1    FF79TST10
ALG(12)   ALGO79P1    FF79TST11
ALG(13)   ALGO79P1    FF79TST12
ALG(14)   ALGO79P1    FF79TST13
ALG(15)   ALGO79P1    FF79TST14
ALG(16)   ALGO79P1    FF79TST15
ALG(17)   ALGO79P1    FF79TST16
ALG(18)   ALGO79P1    FF79TST17
ALG(19)   ALGO79P1    FF79TST18
ALG(20)   ALGO79P1    FF79TST19
&
```

Action Algo 80: Status Change Alarm Area Inhibit

Alarm reporting will be inhibited for the nominated assignable assets when the status point is in an alarm inhibit state.

This algorithm is used to inhibit alarm reporting for sensor points in given assets if the unit has been shutdown.

This algorithm will inhibit alarm reporting for a group of assignable assets specified in the parameter block depending on the nominated states.

Algorithm details

- ALG(01)

  Bits 0 to 7 of the 16-bit word represents a state of the point in which the alarms will be inhibited. For example, (using hexadecimal numbers):

  - Z'0001' will inhibit alarms for the state 0
  - Z'000A' will inhibit alarms for the states 1 and 3

  The Z'xxxx' notation is used to represent hexadecimal values.

- ALG(02)-(20)

List of up to 19 assignable assets to be included in the group.

For example:

```
ALG(02) A'F1'     ALG(03) A'F2'
```

Where `F1` and `F2` are tag names of assets, and the format A'*xx*' is used to represent character values.

Remarks

- This algorithm does not affect the scanning of points. It simply ignores the alarm status being reported by the controller. If a point is in the alarm condition at the transition from inhibit to active, the point will not report the alarm state until either a background scan or the point restores to normal and then alarm again.

Example

An example of a point definition file entry using Action Algo 80: Status Change Alarm Area Inhibit.

```
&
& Area Group Inhibit algorithm
&
DEL      ALGO80P1
ADD      ALGO80P1   STA00010  STATUS CHANGE
RANGE    ALGO80P1   0         1.0
STATEDES ALGO80P1   OFF  ON
PVSOURCE ALGO80P1   006 CS01
PVPERIOD ALGO80P1   015
&
& algorithm Block
&
& - inhibit alarming in state 1
&
ACTALGO   ALGO80P1   80  30
ALG(01)   ALGO80P1   Z'0002'
ALG(02)   ALGO80P1   A'1A'
ALG(03)   ALGO80P1   A'2A'
ALG(04)   ALGO80P1   A'3A'
ALG(05)   ALGO80P1   A'1C'
ALG(06)   ALGO80P1   A'2C'
ALG(07)   ALGO80P1   A'3C'
&
END
```

Action Algo 92: Queued Task Request

Requests a specific task when a status point changes value. Up to seven optional parameters can be passed to the task. The point number is also passed to the task.

The algorithm makes single task request each time any status point changes. Because requests are queued, the risk of losing a request is reduced. The requested task uses GETPRM to process the request block. The task request uses a 10-word parameter block defined in the algorithm block, but words 3, 4, and 5 are reserved for use by the algorithm.

Algorithm details

- ALG(01)

  The Logical Resource Number (LRN) of the task to be requested.

- ALG(02)

  States (0-5) to request task.

  Bit 0 = State 0

  Bit 1 = State 1

  Bit 7 = State 7

- ALG(03) - (05)

  Reserved.
- ALG(06)

  Optional Parameter 1.
- ALG(07)

  Optional Parameter 2.
- ALG(08)

  Optional Parameter 3.
- ALG(09)

  Reserved Parameter 4. Passes the point reference number of the status point to which the algorithm is attached.
- ALG(10)

  Reserved Parameter 5. Passes the state of the status point to which the algorithm is attached.
- ALG(11)

  Optional Parameter 6.
- ALG(12)

  Optional Parameter 7.
- ALG(13)

  Optional Parameter 8.
- ALG(14)

  Optional Parameter 9.
- ALG(15)

  Optional Parameter 10.
- ALG(16)

  States (64-79) to request task.
- ALG(17)

  States (80-95) to request task.
- ALG(18)

  States (96-111) to request task.
- ALG(19)

  States (112-127) to request task.
- ALG(20)

  Optional Parameter 11.
- ALG(22)

  Optional Parameter 13.
- ALG(24)

  Optional Parameter 15.
- ALG(26)

  Optional Parameter 17.

Remarks

- Parameters 4 and 5 [ALG(09) and ALG(10)] are reserved for use by the algorithm process.
- This algorithm cannot be used in conjunction with *Action Algo 69: Status Change Task Request* to request the same task.

- To request multiple tasks on the same state change(s), configure multiple points on the same controller address. The algorithm can then be configured for each task using the different points that access the same information.

Example

An example of a point definition file entry using Action Algo 92: Queued Task Request.

The Z'xxxx' notation is used to represent hexadecimal values.

```
&
& Queued task request algorithm to request
& task with LRN 111 when the status point
& changes to any value.  Pass value of 3 to
& Parameter One, 34 to Parameter Two and 4 to
& Parameter Seven.
&
DEL          ALGO92P1
ADD          ALGO92P1  STA00000  QUEUED TASK REQUEST
RANGE        ALGO92P1  0.0  1.0
STATEDES     ALGO92P1  OFF  ON
PVSOURCE     ALGO92P1  010  P:ALGO92P1 OP
PVPERIOD     ALGO92P1  2
&
& Algorithm block
&
ACTALGO      ALGO92P1  092  100
ALG(01)      ALGO92P1  111
ALG(02)      ALGO92P1  Z'0002'
ALG(06)      ALGO92P1  3
ALG(07)      ALGO92P1  34
ALG(12)      ALGO92P1  4
&
END
&
```

About composite alarms

Attention

Composite alarming is only supported for legacy systems. Newer systems should use Alarm Groups instead.

The alarm icon does not support composite alarming.

Composite alarming does not work over DSA or on console Stations.

The composite alarm algorithm structure enables a master point to report on the alarm condition of a series of field points.

One master monitors up to 18 subordinate points. The master, in turn, can be one of up to 18 subordinates of a higher master. This enables one point to monitor the alarm condition of the whole plant and subordinate points to define the alarm conditions in areas of the plant.

By using custom graphics, operators are able to identify a relationship between points in alarm state. Plant section failures can be seen and operators can quickly zoom in on trouble spots.

The state of the master point will reflect the most severe alarm condition that exists in any of its subordinate points. The alarm severity states from highest to lowest are:

- Alarm and unacknowledged
- Alarm and acknowledged
- Normal and unacknowledged (previous alarm)
- Normal

There are two algorithms required to implement the composite alarm structure:

- Composite Alarm Initiation (Algo#11). When a point with this algorithm attached changes state, it causes the master point to process.
- Composite Alarm Processing (Algo#12). When a point with this algorithm attached processes, it checks the state of all its subordinate points and changes it own state accordingly.

A typical scenario would be a field point entering an alarm state. As a change of state has occurred, algorithm 11 causes the master point to process. Algorithm 12, of the master point, looks at the state of the points below and changes its own state to unacknowledged alarm. As the state of the master point has changed, its algorithm 11 causes its master point to process. Thus, a ripple of change propagates through the hierarchy. The ultimate master point now reflects the condition of the severest field alarm condition.

To reflect the four possible alarm conditions, each master point (all points except the lowest level) must be a dual-bit status point. These points must have alarming inhibited, no PVPERIOD and must have algorithm 12 attached. All points, bar the ultimate master, must have algorithm 11 attached. Field points (the lowest level) can be of any point type and must have alarming permitted, and alarm states defined. The triggering alarm must be higher than journal priority, however, composite alarming makes no further differentiation between urgent, high, and low alarm priorities.

Each algorithm 12 must use the same algorithm block number as its subordinate point's algorithm 11 block number. No other points may use this block number. Incorrect usage of these block numbers may result in severely degraded server system performance.

The following figure shows a simple implementation of the composite alarm structure. Numbers refer to algorithms and letters to algorithm block numbers. Where *x*, *y*, *z* are algorithm block numbers, and *11*, *12* are the algorithm numbers.

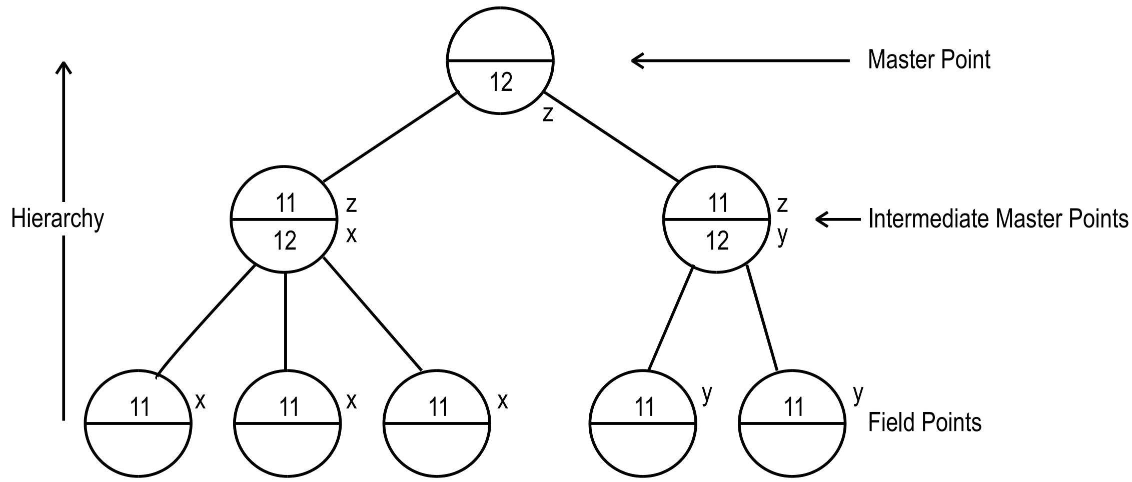

Figure 3: Composite alarm algorithm example

Related topics

"Action Algo 11: Composite Alarm" on page 106
"PV Algo 12: Composite Alarm Processing" on page 93

Handling errors in PV algorithms

If any inputs to a PV algorithm contain bad values or cannot be found, the PV will be set to '*bad*.'

Related topics

"Configuring PV algorithms using pntbld" on page 86

Notices

Trademarks

Experion®, PlantScape®, SafeBrowse®, TotalPlant®, and TDC 3000® are registered trademarks of Honeywell International, Inc.

OneWireless™ is a trademark of Honeywell International, Inc.

Other trademarks

Microsoft and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Trademarks that appear in this document are used only to the benefit of the trademark owner, with no intention of trademark infringement.

Third-party licenses

This product may contain or be derived from materials, including software, of third parties. The third party materials may be subject to licenses, notices, restrictions and obligations imposed by the licensor. The licenses, notices, restrictions and obligations, if any, may be found in the materials accompanying the product, in the documents or files accompanying such third party materials, in a file named third_party_licenses on the media containing the product, or at http://www.honeywell.com/ps/thirdpartylicenses.

Documentation feedback

You can find the most up-to-date documents on the Honeywell Process Solutions support website at:

http://www.honeywellprocess.com/support

If you have comments about Honeywell Process Solutions documentation, send your feedback to:

hpsdocs@honeywell.com

Use this email address to provide feedback, or to report errors and omissions in the documentation. For immediate help with a technical problem, contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

How to report a security vulnerability

For the purpose of submission, a security vulnerability is defined as a software defect or weakness that can be exploited to reduce the operational or security capabilities of the software.

Honeywell investigates all reports of security vulnerabilities affecting Honeywell products and services.

To report a potential security vulnerability against any Honeywell product, please follow the instructions at:

https://honeywell.com/pages/vulnerabilityreporting.aspx

Submit the requested information to Honeywell using one of the following methods:

- Send an email to security@honeywell.com.

  or

- Contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

Support

For support, contact your local Honeywell Process Solutions Customer Contact Center (CCC). To find your local CCC visit the website, https://www.honeywellprocess.com/en-US/contact-us/customer-support-contacts/Pages/default.aspx.

Training classes

Honeywell holds technical training classes on Experion PKS. These classes are taught by experts in the field of process control systems. For more information about these classes, contact your Honeywell representative, or see http://www.automationcollege.com.

Index

A

B

C

D

E

G