Eurotherm®

by Schneider Electric

Eurotherm PAC

Application & Control Module Blocks

Reference Manual

Issue 9

January 2018
HA084012U003

Contents

CHAPTER 1 INTRODUCTION

This manual describes the Application Modules and Control Modules available within the LIN environment. These control modules can be used in conjunction with existing LIN blocks to perform a wide range of control tasks. This manual consists of,

- Application Module details
- Control Module details

Application modules can be utilised for specific activities such as combustion control, load management and pump set control.

Control Modules consist of common devices such as Controls, Motors and Valves.

1.1 WHAT ARE APPLICATION MODULES AND CONTROL MODULES ?

The term ‘control module’ comes from the ISA-S88, Batch Control Part 1 ‘Models and terminology’, seven-layer physical model. Although the ISA-S88 is aimed at batch control systems, the control module concept works well in all control applications.

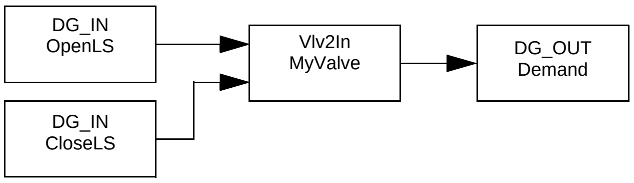
An Application/ Control Module consists of a number of elements that can be used either separately or in conjunction with one another. The core element is a pre-defined block type that runs in a LIN database. To support the control module within a system requiring an MMI interface, graphic icons and parameter pages are supplied for use at the supervisory computer.

The LIN block contains any logic required, e.g. discrepancy checking, as well as providing the links to the supervisory computer via icons and faceplates. A system implementing dedicated control modules is more straightforward as a minimum amount of blocks are required, tagging is preserved through the configuration and function specific naming is used making documentation and Structured Text (ST) in Sequential Function Charts, SFC’s, clearer.

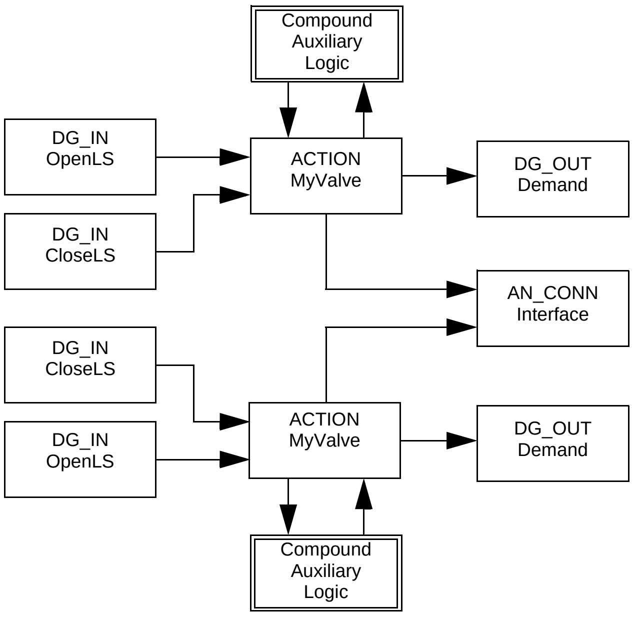
These modules include resource management logic to ease implementation of complex Sequence and batching applications.

1.2 WHAT ARE THE BENEFITS OF APPLICATION MODULES AND CONTROL MODULES ?

Within a LIN database, an Application Module or Control Module is usually connected to I/O blocks, although it may be linked in other ways, e.g. MODBUS.



The valve can be represented as a logical object, with a common tag, and a single block interface throughout the system.



This replaces the traditional approach where each Control Module is implemented as a collection of LIN blocks, e.g. an ACTION block and auxiliary logic, with a shared LIN block interface to the supervisory computer.

The following table details some of the advantages these module have to offer.

Feature	Control Module block	ACTION block
LIN alarms generation	Yes	No
Built-in timers	Yes	No
Optimised LIN performance	Yes	No
Meaningful field names	Yes	No
Fully integrated	Yes	No
Pre-defined and tested logic	Yes	No

Table 1-1: Application Module/Control Module features

1.2.1 Faceplates, icons and engineer pages

In order to allow operator interfacing to the Application Modules or Control Modules, a selection of icons, faceplates and engineer pages are available.

These are collected within a library of supervisory computer graphic objects that map to each control module. Some of the graphic objects can be used with more than one LIN block type. Tables are provided to show the valid permutations.

Icons are display-only representations of a module. Each icon provides the facility to activate the faceplate and the engineer page of the module.

The faceplates provide the operator interaction, e.g. open and close a valve. Each faceplate is split up into a display area and an interactive area. The display area shows modes, alarms etc. The interactive area allows mode changes and commands to be sent to the module.

The engineer page maps to all the LIN block parameters available and is designed for use by engineers and technicians.

1.2.2 Faceplates

Faceplates provide an operational summary of the Application Module or Control Module in the form of an instrument faceplate and may allow some operator interaction with the device. Faceplates may be displayed on the supervisory computer as a sub-window or be placed directly onto a mimic graphic at configuration time.

All faceplates are based on a generic display.

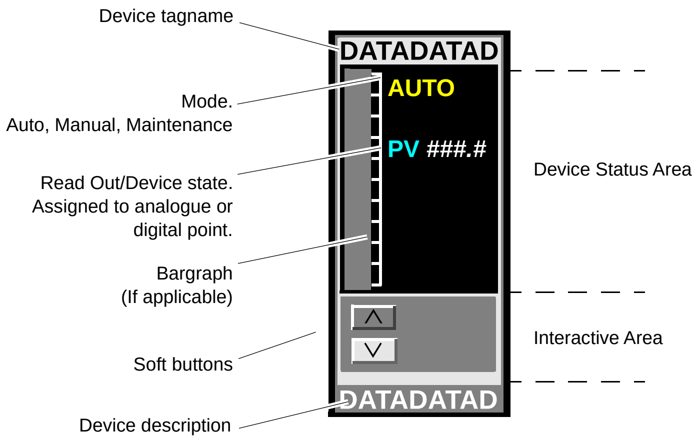


Figure 1-1: Faceplates

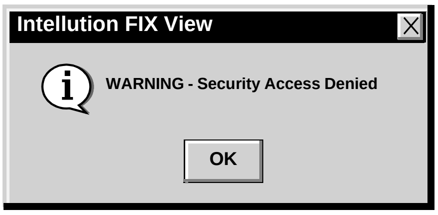
1.2.3 Device status area

The device status area displays a summary of the module current status. Text messages are used to display the device's state; read outs and bar graphs are used to display analogue values. The operator may be able to interact with some of the displays in the device status area by double clicking on an entry using the pointing device, e.g. to adjust a setpoint on a PID controller faceplate.

The device status area displays a flashing yellow 'ALM' message if the device is indicating any unacknowledged alarms. The device status area displays a steady yellow 'ALM' message if the device is indicating only acknowledged alarms.

1.2.4 Interactive area

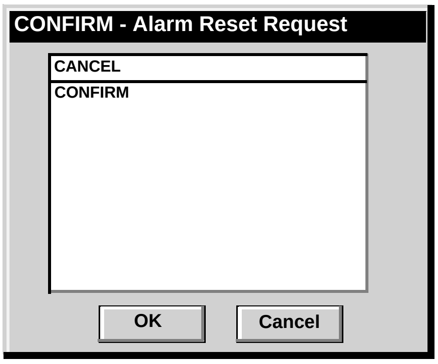
The interactive area may contain soft buttons which allow the operator to interact with the module. Each faceplate is assigned to a single security area where enabled. A single click to a soft button causes the supervisory computer to check if the user has access to the security area. If the check fails to authenticate the user, a warning message is displayed.


Intellution FIX View

WARNING - Security Access Denied

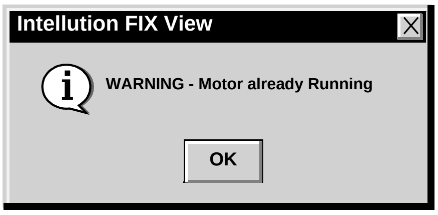
If the current user is logged into a sufficiently privileged account and the button is confirmation enabled, then a confirmation message dialog is displayed. Each soft button may be individually configured to display a confirmation prompt or take immediate action.

The operation is only performed if the user first selects CONFIRM and then single clicks the Enter soft button. Selection of the CANCEL option, no option selection or a single click to the Cancel soft button results in no operation being performed.


CONFIRM - Alarm Reset Request

CANCEL
CONFIRM

Error or warning messages are displayed, as appropriate, for each soft button. Where an error or warning message is displayed, no operation is performed. The user clicks on an ‘OK’ button to clear the message.


Intellution FIX View

WARNING - Motor already Running

The soft buttons follow the standard Windows convention of being ‘Greyed Out’ when not available. A single click to a ‘Greyed Out’ soft button has no effect.

Active action soft button

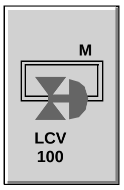
Inactive (Greyed Out) action soft button

1.2.5 Icons

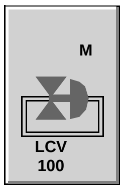
An icon is a graphical object which is configured on a supervisory computer mimic and linked to a single module. The icon displays key information about the current status of a device.

The icon tags are configured as two lines of text. The tags may be hidden on a per mimic basis to reduce complexity.

Each icon provides a mechanism to display the control module faceplate and engineer page.



Selection of the upper part of the icon activates the appropriate engineer page of the device



Selection of the lower part of the icon activates the appropriate faceplate of the device

1.2.6 Engineer pages

Engineer pages are used to provide information and interaction at an engineering level for an Application Module or Control Module. Each control module type has a single engineer page associated with it. The engineer page associated with a control module can be displayed from an icon, or by using the **point** soft button.

All the control module engineer pages are based on a generic display.

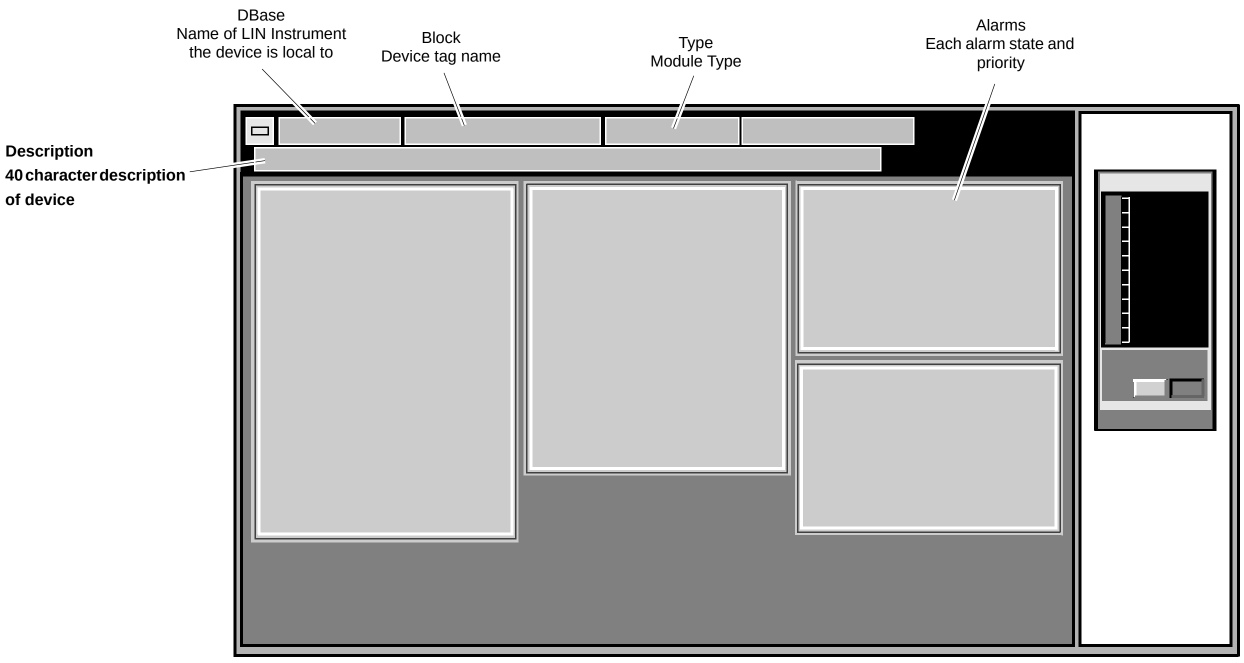

DBase
Name of LIN Instrument the device is local to
Block
Device tag name
Type
Module Type
Alarms
Each alarm state and priority
Description
40 character description of device

Figure 1-2: Engineer pages

Other fields contained in engineer page are dependent upon the module type.

Each field in a module engineer page is assigned to one of the three security levels, which define the level of operator interaction, if any, of the field. This assignment is fixed for all instances of the module,

- Read only. These fields cannot be altered from the engineer page.
- LIN Operator Access. These fields may be written to by users whose accounts contain either the ‘Eurotherm LIN Operator Access’, ‘Eurotherm LIN Supervisor Access’, or ‘Eurotherm LIN Engineer Access’ application features. The fields are read only to all other users.
- LIN Supervisor Access. These fields may be written to only by users whose accounts contain ‘LIN Supervisor Access’ application feature. The fields are read only to all other users.

The engineer page also contains a faceplate for the device. The faceplate is loaded dynamically from the file whose name corresponds to the module ‘FpltType’ field. If this file is not present the default faceplate is loaded.

1.2.7 Compatibility

Configuration of the Application Module or Control Module into a LIN instrument database requires LINtools Engineering Studio. This is because this version of LINtools includes a MAKE dialog that can include new groupings of templates, e.g. Valves, Motors etc.

To place a module LIN block using the LINtools,.

- Make an appropriate header block for the target instrument as usual e.g. ‘Tactician’, ‘T800’, or ‘Eycon-10’
- Select MAKE from the LINtools menu bar
- Select CtrlModules as the Target
- Select the required Category (e.g. Valves)
- Select the appropriate Template (e.g. Vlv1In)
- Place, drag and drop, on the work sheet

Full support for Application Module and Control Module LIN blocks is available in the following instruments.

Instrument	Version
T920	4.1 and above
T921	4.1 and above
T932	4.1 and above
T640	4.1 and above

Table 1-2: Instrument Support

1.2.8 System design constraints

The use of the Application Module and Control Module can increase the number of blocks cached compared to a system based on DG_CONN and AN_CONN blocks. When planning the system, care must be taken to verify that sufficient database and licence resources exist. The additional cached block count may cause increased LIN traffic and so the system loading should be verified at the design stage.

Application Module and Control Module LIN blocks are implemented as foreign templates. This means that the description of how the block looks and behaves is not known by the instrument firmware but is downloaded within the LIN database file. Consequently, database file space and database memory is used for each module in each database. The use of memory to hold the module definition is usually offset by the decrease in the number of wires and LIN blocks used in a database.

The Application Module and Control Module documentation gives two figures for memory use.

- Template size. This is an overhead that is used only once in a database.
- Instance size. This is the use for each block built in the database.

Example

If a database were built with five ‘BigValve’ blocks and seven “BigMotor” blocks, the memory consumed would be 2245 + 1896 + 5x97 + 7x113 = 5417 bytes. To get a total database size the instance sizes of the header and any other blocks used would have to be added to the total figure.

Template Name	Template Size	Instance Size
BigValve	2245	97
BigMotor	1896	113

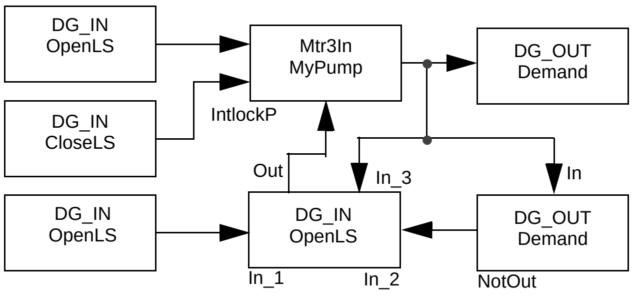
Table 1-3: Template Constraints

1.2.9 Hints and tips

It is recognised that customisation may be needed for individual systems. To this end, the standard module components are designed to include the generic functionality whilst allowing customisation by the addition of external logic, module options, parameter configuration, etc. It is recommended that any such customisation be reviewed and tested before system configuration to avoid the possibility of extensive rework.

Example 1

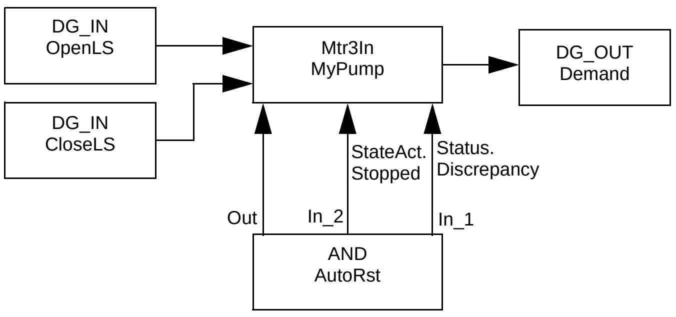
It is required to force a pump to stop on detection of a low flow. This could be achieved using a control module interlock. However, further measures must be taken to allow the pump to restart following a low flow detection. This is necessary because during starting there will be a low flow detected which would then activate the interlock causing the pump to stop. The desired functionality is achieved using LINtools.



- Add a DG_IN, an AND block and a PULSE block and name them ‘LowFlow’, ‘SetInlk’ and ‘DsblInlk’ respectively..
- Connect the ‘*Out*’ of ‘Low Flow’ to the ‘*In_1*’ of ‘SetInlk’. This asserts ‘SetInlk’ when low flow is detected provided the other conditions are conducive.
- Connect the ‘*Out*’ of ‘SetInlk’ to the ‘*IntlockP*’ of ‘MyPump’. This asserts the primary interlock when ‘*SetInlk*’ is asserted. If the primary interlock is set FALSE, this causes the control module demand to be set FALSE and, hence, the pump to be stopped.
- Connect the ‘*Demand*’ of ‘MyPump’ to the ‘*In*’ of ‘DsblInlk’ and the ‘*NotOut*’ of ‘DsblInlk’ to the ‘*In_2*’ of SetInlk. Set the ‘*Sense*’ parameter of ‘DsblInlk’ to ‘Rising’. This stops ‘SetInlk’ being asserted for the period defined in ‘DsblInlk’ following a start command.
- Connect the ‘*Demand*’ of ‘MyPump’ to the ‘In_3’ of ‘SetInlk’. This stops ‘*SetInlk*’ being asserted when the pump has being asked to stop. This ensures that the pump may be restarted after a low flow has caused the pump to be stopped.

Example 2

It is required that a discrepancy alarm be self-resetting such that no operator reset is required before being permitted to re-start a pump following a trip. The desired functionality is achieved using LINtools.

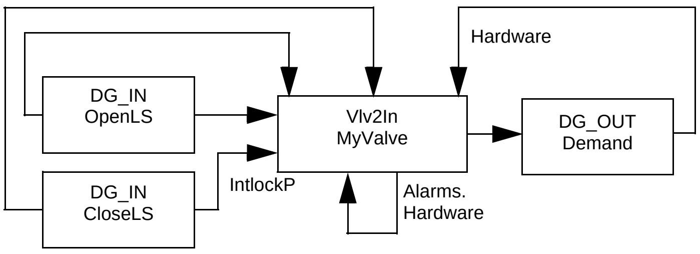


- Connect from the ‘*Out*’ field of ‘AutoRst’ to the ‘*Reset*’ field of ‘MyPump’. This resets the ‘*Discrepancy*’ alarm when ‘AutoRst’ is asserted. This removes the need for an operator to reset the module.
- Connect from the ‘*Status.Discrepancy*’ field of ‘MyPump’ to the ‘*In_1*’ field of ‘AutoRst’ and from the ‘*StateAct.Stopped*’ field of ‘DsblInlk’ to the ‘In_2’ field of ‘AutoRst’. This causes ‘AutoRst’ to be asserted when the discrepancy alarm is raised and the pump has come to a stopped state, i.e. the normal operator reset action is performed automatically.

Hints and tips (Cont.)

Example 3

It may be required to force a valve closed on a hardware fault detection. This is not an inherent function of any of the standard control modules. However, the Vlv2In block does include a hardware alarm that is raised if any bit in the ‘Hardware’ bitfield is set and also includes interlocks to force the demand to a given position. The desired functionality is achieved using LINtools.



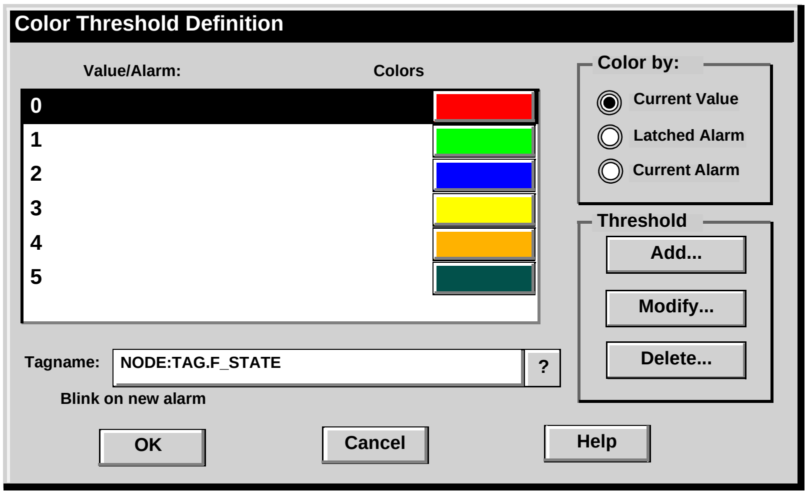
- Connect each Hardware alarm in the DG_IN and DG_OUT blocks to the ‘Hardware’ bitfield in the module. This ensures that if a DG_IN block detects a hardware fault it will cause the Hardware alarm to be raised in ‘MyValve’.
- Set the ‘Hardware’ alarm fields to the desired priority. An alarm priority of zero disables the alarm and, hence, prevents this configuration from functioning correctly.
- Connect from the ‘Hardware’ alarm of ‘MyValve’ back in to the ‘IntlockP’ field of ‘MyValve’. This causes the primary interlock to be asserted when the ‘Hardware’ alarm is raised.
- Set the ‘InlkValP’ parameter of ‘MyValve’ to ‘FALSE’. This ensures that, when the primary interlock is asserted, the control module demand is set FALSE and, hence, the valve is closed.

Hints and tips (Cont.)

Example 4

It is required to change the Valve Icon dynamo such that the colour is black for Closed, red for Open and grey at other times. The desired functionality is achieved using the T3500 Draw package.

- Paste down the required dynamo.
- Using the ‘Dynamo’ menu, select ‘Convert to Objects’.
- Double click the icon and change the foreground colours as desired. The presented dialog lists the colour for each of the enumerations of the ‘*State*’ field, e.g. for a valve the values 0, 1, 2, 3, 4 and 5 correspond to the states Open, Closed, Opening, Closing, LSFault and Unknown respectively.



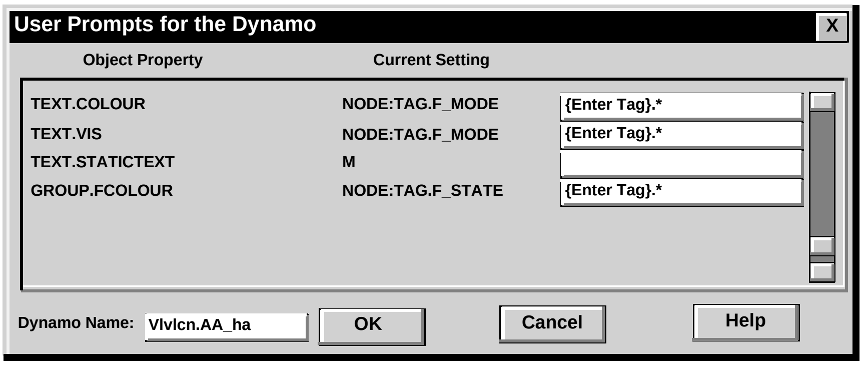
Value/Alarm:	Colors
0	🟥
1	🟩
2	🟦
3	🟨
4	🟧
5	⬛

Color by: [✓ Current Value] Latched Alarm Current Alarm

Tagname: NODE:TAG.F_STATE

Blink on new alarm

- Use the select tool to select all items of the dynamo.
- Using the ‘Dynamo’ menu, select ‘Create Dynamo’.
- Locate the modified dynamo property, ‘*NODE:TAG.F_STATE*’, and re-enter the prompt ‘{Enter Tag}’.



Object Property	Current Setting	
TEXT.COLOUR	NODE:TAG.F_MODE	{Enter Tag}.*
TEXT.VIS	NODE:TAG.F_MODE	{Enter Tag}.*
TEXT.STATICTEXT	M	
GROUP.FCOLOUR	NODE:TAG.F_STATE	{Enter Tag}.*

Dynamo Name: VlvIcn.AA_ha

Hints and tips (Cont.)

Example 5

When using control modules to interface with plant devices requiring pulsed outputs, it is sometimes desirable after a re-download of the database to have the device state match the demand.

This is achieved by setting the *Options.FrcPulse*' bit to True in LINtools Engineering Studio. When the database is re-downloaded and during its first scan, a pulsed output is transmitted to the plant device making its state match the control module demand.

Note: This implementation is not recommended for systems requiring plant devices to stay put after a database re-download. When using the RaiseLwr control module in a T640, bear in mind that if the travel time for the control device is, for example, equal to 12 seconds and the loop scan time is 420 milliseconds, the maximum achievable accuracy can be slightly greater than 0.42* 100/12, i.e. 3.5% at best. Better resolution can be achieved with slower control devices. The RaiseLwr application module includes an *ErrLim* field (deadband) which can be set to a value such that hunting is stopped.

1.3 BLOCK SPECIFICATION MENU

This section, specifically the Tagname, Type, Task, LIN Name, DBase, Rate fields apply to all blocks except as otherwise stated.

Tagname, Type, Task: These are block related parameters used to control the operation of the block within the LIN Database.

LIN Name, DBase, Rate: These are LIN Database related parameters used to control communications between LIN Instruments.

1.3.1 TagName

This is the user-entered (16-character max.) block Tagname (default 'NoName'), which identifies and distinguishes the block from other blocks of the same Type. Tagname appears as a label on the block icon, below its Type identifier. The Tagname is usually related to the LIN Name unless it exceeds 8 characters, but can differ entirely by changing the Tag settings configuration using View > Options > Settings > Tags, in LINtools Engineering Studio.

Note: A Strategy cannot be saved unless the TagName has been entered for all blocks in the LIN Database.

1.3.2 Type

This read-only field shows the Template Function block category mnemonic, e.g. AN_IP, PID, SIM, etc. which also appears as a label on the block icon, above its Tagname. The remaining chapters in this manual are related to block type.

1.3.3 Task

This defines the Task in which this block will be run. Each task number corresponds to a specific repeat rate so all blocks assigned to this Task number are updated at the interval defined by the relevant 'Period ms' parameter.

1.3.4 LIN Name

This is the (maximum 8-character) unique network LIN Name of the block. This Name can be derived from TagName or can be edited by changing the Tag settings configuration using View > Options > Settings > Tags, in LINtools Engineering Studio.

1.3.5 DBase (Local/Layer/Remote)

This is used to define the location of the LIN Database that any block, other than the Header block, will run, local or cached. A cached block is a local image of a remote block, i.e. a block running in another instrument on the LIN, that allows interaction with the remote block.

Note: In a cached block, Dbase specifies the name of the remote LIN Database containing the 'real' block.

'Local' means that this block operates in the LIN Database in this Instrument.

'Layer' indicates that the block is local to the base.dbf in this LIN Database, but is on another layer and is referenced only to make connections.

'Remote' means that this block operates in a LIN Database in a LIN Instrument at a different LIN Node address. This remote database is referenced to gain access to field values and to make connections. When selected, Dbase and Node Address parameters become available.

1.3.6 Rate

This is used to defined the minimum update rate (cycle time in milliseconds) of either a single or a group of remote (cached) function blocks, at which an individual cached block is transmitted across the LIN. The default is 10ms minimum, i.e. 100Hz maximum. Rate can be set between 10ms and 64s. These rate values are minimum update times only, and heavily loaded networks may not be able to attain the faster update rates.

Note: For the system to run correctly, a database of the selected name must reside at the specified node, and must contain a corresponding real block of the same type as the local cached block.

1.4 ALARMS

This field displays the name of the most significant alarm in the block (most significant/Combined Alarm is defined below). Alarms can also be subject to a suppression condition, refer to the Alarm Suppression User Guide, HA030272. The Alarms field also accesses a window with three columns - Name, Value, and Priority. The Name column lists the names of all the block alarms, e.g. Software, Hardware, etc. The Value column displays for each current alarm condition a message dependant upon 'Alarm Priority Number' setting as follows, 'In Ack', 'In Unack' or 'Out UnAck' where 'In' indicates an active Alarm condition and 'Out UnAck' indicates an Alarm that has cleared awaiting Acknowledgement. By 'right-clicking' in this field, unacknowledged Alarms can be Acknowledged. The Priority column shows the user-specified priority number for each alarm and is the only read/write field in the Alarms window.

NOTE: In instruments without integral displays, the Alarms window can be seen using the Terminal Configurator attached to the instrument via Ethernet (ELIN), via a block cached in a display-type instrument (e.g. Eycon™ 10/20 Visual Supervisor) or via ELIN on a PC running LINtools' 'Connect' facility.

Alarm Priority Number. There are four types of alarm priority.

1.4.1 Priority 0 (Lowest Priority)

Switches the alarm off.

1.4.2 Priorities 1 to 5

Specifies 'self-acknowledging' alarms. For these, the 'In Ack' message appears when the alarm condition occurs, and disappears when the alarm has cleared without the need to acknowledge the alarm.

1.4.3 Priorities 6 to 15

Specifies 'acknowledging' alarms. For these the 'In UnAck' message appears when the alarm condition occurs. To acknowledge the alarm, 'right-click' in the field and select 'Acknowledge' from the drop down menu and the field now displays 'In Ack'. When the alarm condition clears the 'In Ack' message disappears. However, if the alarm condition clears and the Alarm has not been acknowledged ('In UnAck'), the 'Out UnAck' message appears indicating that the Alarm has cleared but is awaiting acknowledgement.

Note: For certain legacy instruments, priorities 11 to 15 set a special bit in the Configuration (Header) block and activate a hardware alarm relay. (Not supported in current instruments.)

1.4.4 Software.

All LIN function blocks have a Software alarm (and a Combined) alarm, as a minimum and cannot be suppressed. A Software alarm (default priority 1) is generated upon sumcheck failure in a block, i.e. corruption of its database. For cached, S6000, and TAN blocks, failure of communications to the principal block also activates the alarm

1.4.5 Hardware.

A hardware alarm is generated when any of the bits in the Status parameter are set, e.g. in the event of a power interruption, incorrect module type, hardware fault, etc.

A hardware alarm also causes the respective I/O module LED to turn off, located at the top of each module. Also any other relevant LEDs fitted to a module will also indicate a Hardware fault condition, e.g. AI2 module.

1.4.6 Alarm Suppresion.

All Alarms (except software and combined) can be subjected to Alarm Suppression functionality as appropriate. For further information see the Alarm Suppression User Guide, HA030272.

1.4.7 Combined.

All blocks have a Combined alarm (and a Software) alarm, as a minimum. A Combined alarm is generated as a copy of the most significant alarm in the block and inherits the same Value message and Priority number. Consequently the name of this most significant alarm is displayed in the Alarms field of the Specification menu.

When several alarms are active, the most significant alarm is chosen in accordance to the following list of attributes:

1. Any unacknowledged Alarm is chosen in preference to any Acknowledged Alarm.
2. If there is more than one Alarm as described in step 1 above, then the Alarm with the highest priority number is chosen.
3. For Unacknowledged Alarms only; If there is more than one, then alarms that are currently active are chosen in preference to those Alarms currently not active.
4. If there is more than one Alarm (Acknowledged or Unacknowledged) with the same priority number, then the lowest numbered alarm (highest/first position in the alarms field list of names) is chosen.

CHAPTER 2 BATCH APPLICATION BLOCKS

The BATCH category of Function Block Templates provides the control strategy with functions for batch and sequence programming.

Intentionally left blank

VLV3WAY: INBATCH PHASE LOGIC INTERFACE BLOCK

Block function

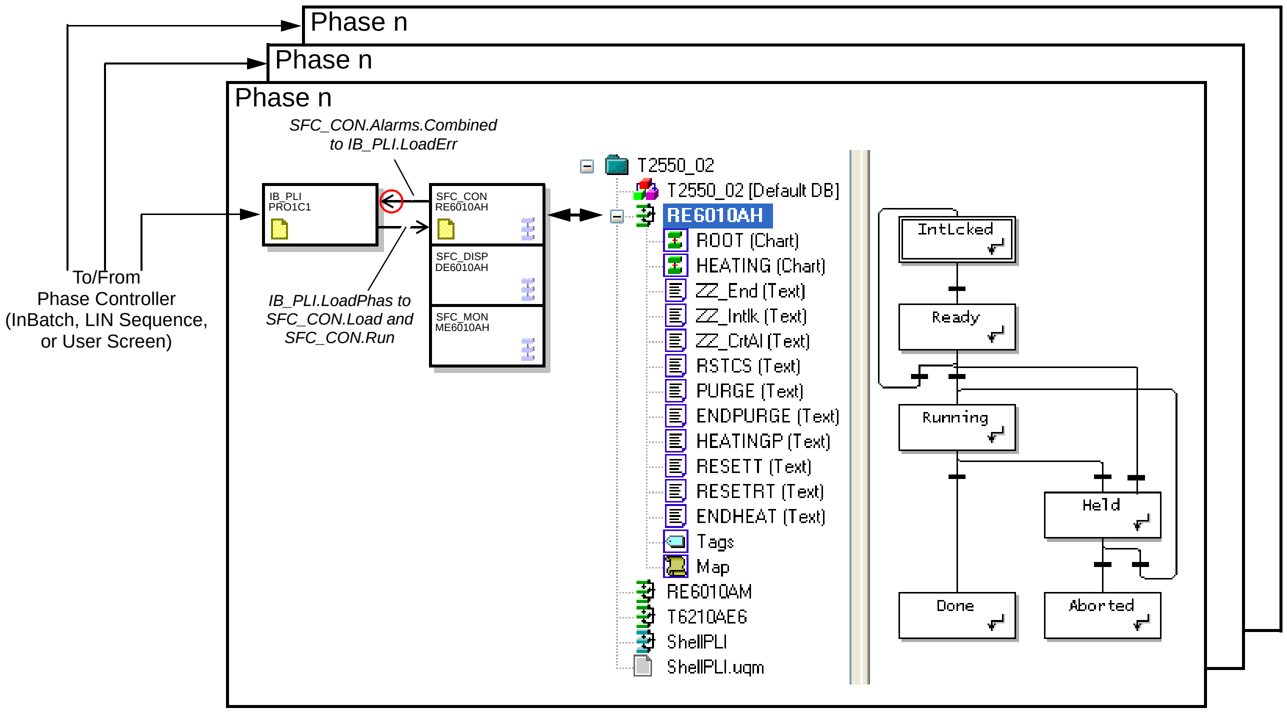


Figure 2-1: Block Interface

The block is designed to operate with the Wonderware InBatch software, but can also be executed from any other Phase Controller, e.g. an SFC or User Screen. InBatch software is used to control and manage the Batch process (Recipe).

This block is an interface between any Phase Controller, i.e. InBatch software on the Supervisory Control And Data Acquisition, SCADA system, and the Sequence (SFC) of the strategy in the instrument. It is connected to and from specific fields in the SFC_CON block to provide seamless control and management of the Batch in progress.

Note	An SFC is linked to a FBD via the SFC_CON block. The SFC_MON block and SFC_DISP block are optional but provide online monitoring and runtime control, display and monitoring of a remote SFC respectively.

A Batch is a product of a system derived from an individual Phase or a combination of Phases produced from a recipe running on a unit. A recipe is a list of ingredients or steps used to create the product. For example, a bakery may have a basic cookie dough recipe that lists all of the ingredients required to make plain cookies and all of the optional ingredients such as nuts, fruit, and chocolate chips, that can be added to the basic recipe to make various kinds of cookies. In a steel mill, a recipe might be a collection of machine setup parameters. For batch processors, a recipe can be used to describe the various steps in the batch process. Recipe values can be configured using the *A0* to *A15* (floating-point variables), *I0* and *I1* (Signed long integer variables) and the *Word0* to *Word5* (16-bit digital variables) fields.

Each IB_PLI block used in the database provides an interface to each Phase relating to a specific SFC generated from the Shell PLI (Phase Logic Interface) SFC. The Shell PLI SFC is supplied as a Generic SFC (described in the *LINtools help file*) and is used in the generic form. This provides a template for the repeated use of a configured Phase but can be copied and edited to suit each individual Phase.

IMPORTANT ***Only add the logic required to provide Interlock, Critical Alarm, Running, and Held control to the SFC as new Actions, but do not edit any Step name in the Sequence. Edit Transitions to provide an indication that all configured states have been obtained and the Step is complete.***

Block Function (Cont.)

Examples

Interlock Logic

```
(*
Action:  ZZ_IntLk

This action should have any code required to derive interlocks that are not
derived in the LIN FBD and wired to the PLI Block Interlock inputs.  The
result, or results should be assigned into free bits within the PLI block
IntLocks field.

If there are no interlocks then state that and assign false.
ZZ_PLI.Intlocks.Ilk0 := 0;
*)

Cleaning.CurrStep:=65535;
```

Critical Alarm Logic

```
(*
Action:  ZZ_CrtAl

This action should have any code required to derive Critical Alarms that
are not derived in the  LIN FBD and wired to the PLI Block Critical Alarm
inputs.  The result, or results should be assigned into free bits within
the PLI block CritAlms field.

If there are no Critical Alarms then state that and assign 0 (false).
ZZ_PLI.CritAlrm.Bit0 := 0;

*)
```

Running Logic

```
(*Set the Set Point for the temperature control in the kitchen tank*)
TIC102.SL:=TempKitch.A0;

(*Kitchen recirculation lane*)
K200.AutoDmnd:=1;
PMP2.AutoDmnd:=1;
```

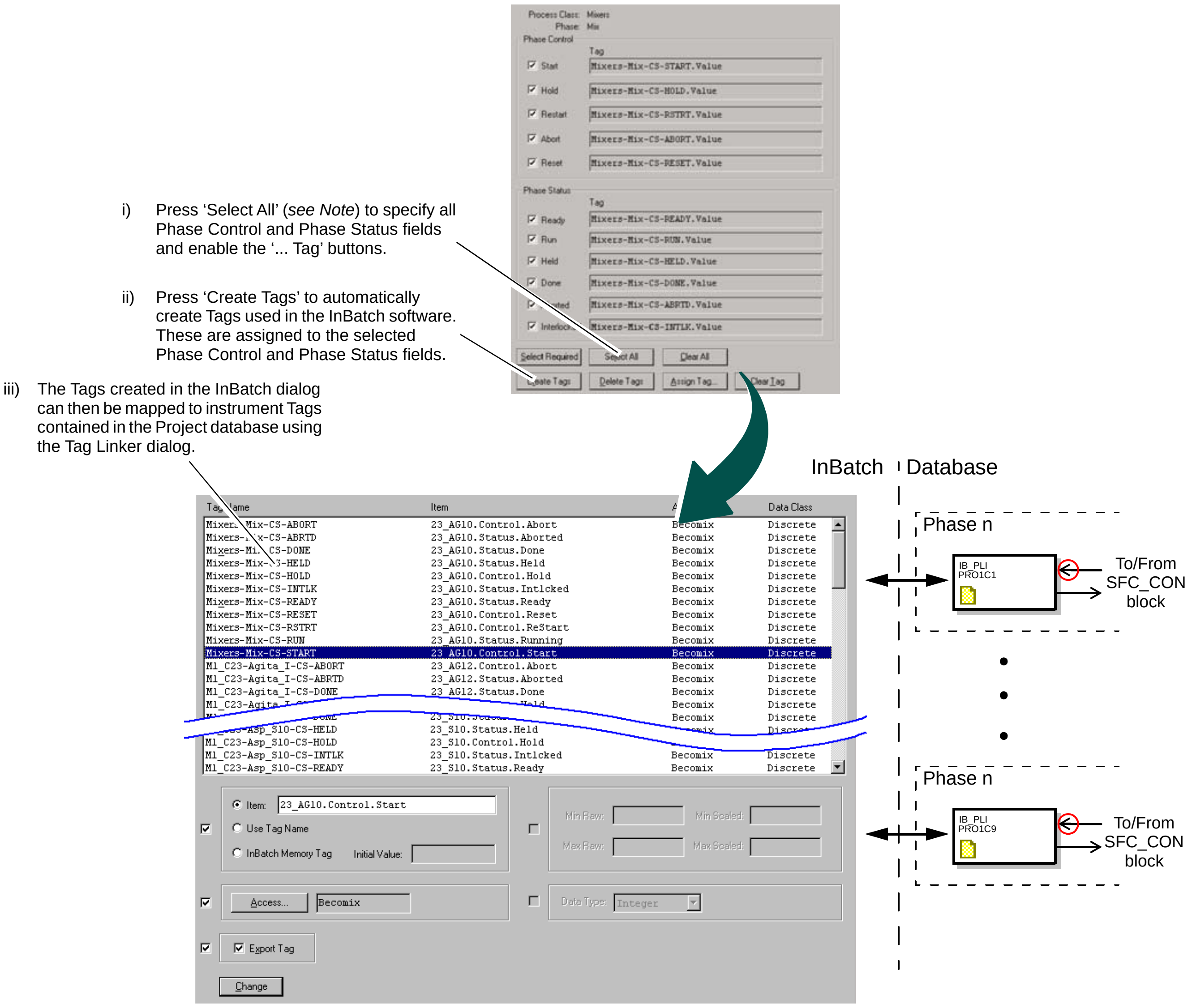
Block Function (Cont.)

Held Logic

```
LeakTest.Control.Hold:=1;
Desinf.Control.Hold:=1;
Dosing.Control.Hold:=1;
TempKitch.Control.Hold:=1;
Transf.Control.Hold:=1;
Cleaning.Control.Hold:=1;
Ste.Control.Hold:=1;
CIPBATCH.Command:="HOLD";
```

IB_PLI WITH INBATCH PHASE CONTROLLER

This diagram shows the relationship between this block, the InBatch software and the database.



i) Press ‘Select All’ (*see Note*) to specify all Phase Control and Phase Status fields and enable the ‘... Tag’ buttons.

ii) Press ‘Create Tags’ to automatically create Tags used in the InBatch software. These are assigned to the selected Phase Control and Phase Status fields.

iii) The Tags created in the InBatch dialog can then be mapped to instrument Tags contained in the Project database using the Tag Linker dialog.

Process Class: Mixers
Phase: Mix

Phase Control

	Tag
✓ Start	Mixers-Mix-CS-START.Value
✓ Hold	Mixers-Mix-CS-HOLD.Value
✓ Restart	Mixers-Mix-CS-RSTRT.Value
✓ Abort	Mixers-Mix-CS-ABORT.Value
✓ Reset	Mixers-Mix-CS-RESET.Value

Phase Status

	Tag
✓ Ready	Mixers-Mix-CS-READY.Value
✓ Run	Mixers-Mix-CS-RUN.Value
✓ Held	Mixers-Mix-CS-HELD.Value
✓ Done	Mixers-Mix-CS-DONE.Value
✓ [illegible]ted	Mixers-Mix-CS-ABRTD.Value
✓ Interlock[illegible]	Mixers-Mix-CS-INTLK.Value

Tag Name	Item	A	Data Class
Mixers-Mix-CS-ABORT	23_AG10.Control.Abort	Becomix	Discrete
Mixers-[illegible]x-CS-ABRTD	23_AG10.Status.Aborted	Becomix	Discrete
Mixers-Mi[illegible]CS-DONE	23_AG10.Status.Done	Becomix	Discrete
Mixers-Mix-[illegible]S-HELD	23_AG10.Status.Held	Becomix	Discrete
Mixers-Mix-CS-HOLD	23_AG10.Control.Hold	Becomix	Discrete
Mixers-Mix-CS-INTLK	23_AG10.Status.Intlcked	Becomix	Discrete
Mixers-Mix-CS-READY	23_AG10.Status.Ready	Becomix	Discrete
Mixers-Mix-CS-RESET	23_AG10.Control.Reset	Becomix	Discrete
Mixers-Mix-CS-RSTRT	23_AG10.Control.ReStart	Becomix	Discrete
Mixers-Mix-CS-RUN	23_AG10.Status.Running	Becomix	Discrete
Mixers-Mix-CS-START	23_AG10.Control.Start	Becomix	Discrete
M1_C23-Agita_I-CS-ABORT	23_AG12.Control.Abort	Becomix	Discrete
M1_C23-Agita_I-CS-ABRTD	23_AG12.Status.Aborted	Becomix	Discrete
M1_C23-Agita_I-CS-DONE	23_AG12.Status.Done	Becomix	Discrete
M1_C23-Agita_I-[illegible]	[illegible]Held	Becomix	Discrete
[illegible]DONE	23_S10.[illegible]	Becomix	Discrete
[illegible]-Asp_S10-CS-HELD	23_S10.Status.Held	[illegible]mix	Discr[illegible]
M1_C23-Asp_S10-CS-HOLD	23_S10.Control.Hold		
M1_C23-Asp_S10-CS-INTLK	23_S10.Status.Intlcked	Becomix	Discrete
M1_C23-Asp_S10-CS-READY	23_S10.Status.Ready	Becomix	Discrete

✓ (•) Item: 23_AG10.Control.Start
( ) Use Tag Name
( ) InBatch Memory Tag Initial Value:

☐ Min Raw: Min Scaled: Max Raw: Max Scaled:

✓ Access... Becomix

☐ Data Type: Integer

✓ ✓ Export Tag

Note	The Select Required button is used to indicate that Tags are required for only the manually selected Phase Control and Phase Status fields.

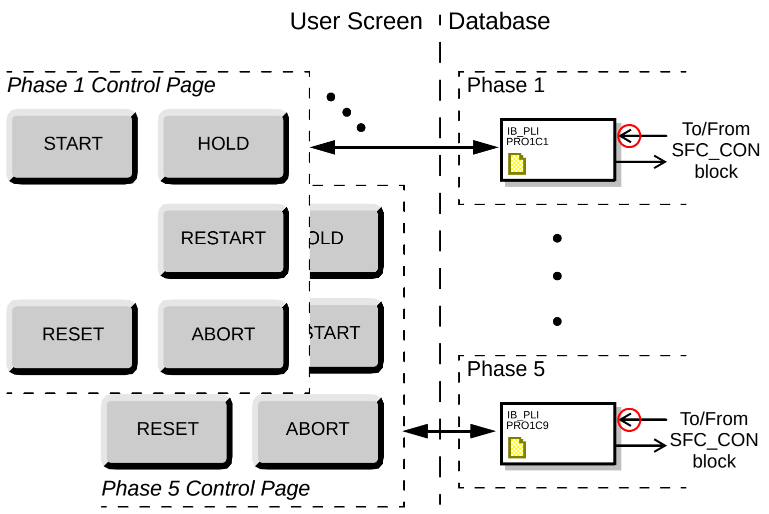
Figure 2-2:

Block Function (Cont.)

IB_PLI WITH OTHER PHASE CONTROLLERS

This shows the Phase Control derived from a User Screen.

Each User Screen button is linked via Structured Text to an individual IB_PLI block that controls a particular Phase.



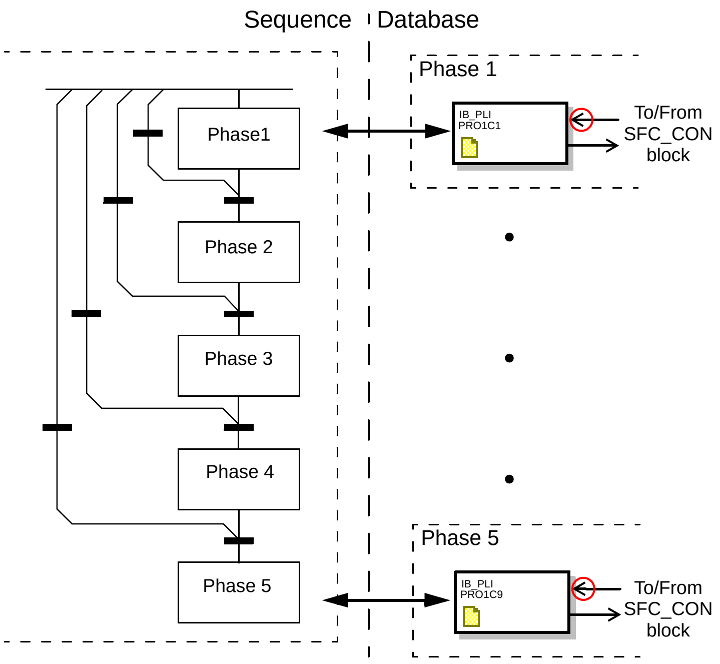
Example

The START button should read,

```
ENABLE:"[LeakTest.Status.Ready]",ST:"[LeakTest.Control.Start]:=1;",NOTE:"START Leak Test Phase"
```

This shows the Phase Control derived from an SFC.

Each Step in a SFC is linked via Structured Text to an individual IB_PLI block that controls a particular Phase. The configured Transition of one Phase initiates the start of the next Phase.



Block operation

This block operates by receiving a Start request (*Control.Start* set TRUE). This loads and starts the SFC specified in the SFC_CON block in the Interlock state, *Status.Intlcked* set TRUE after checking all interlocks and writing the values back to this block. When no interlocks exist (*Status.Intlcked* set FALSE), the SFC continues to the Ready State (*Status.Ready* set TRUE) and on to the Running state (*Status.Running* set TRUE). If interlocks have not cleared, the SFC remains in the Interlocked state, allowing the SFC to progress to the Ready state and wait for this block to indicate that another Start Request (*Control.Start* set TRUE) is received.

If Hold logic is required, e.g. to stop a product setting in a reactor (some heat and mix) actions can be added to the Held state. In these cases, the Running logic in this block may have to record what Step was occurring so the Held logic can determine the next function. Similarly, the Hold logic may have to record what Step was occurring to provide a position to re-start as determined by *CurrStep*, *NextStep* and *ReStStep* (re-start step), allowing the Phase to continue until it is complete, or Aborted, *Control.Abort* set TRUE.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Control	8-bit digital inputs used to control the Phase	CD hex	□➧
Start	Starts Phase derived from Phase Controller	T/F	
Hold	Hold Phase derived from Phase Controller	T/F	
ReStart	Restart Phase derived from Phase Controller	T/F	
Abort	Abort Phase derived from Phase Controller	T/F	
Reset	Reset Phase derived from Phase Controller	T/F	
Buttons	Defines active buttons	CD hex	□➧
Btn1	User button 1 active		
Btn2	User button 2 active		
CritAlrm		ABCD hex	
Alm0 to Alm15	15 Digital critical alarm indicator fields	T/F	➧□➧
Intlocks			
Ilk0 to Ilk15	15 Digital interlock indicator fields	ABCD hex	➧□➧
Status	16-bit status field, indicating Phase status	(AB)CD hex	□➧
Ready	Phase is ready for use	T/F	D 1
Running	Phase is currently in use	T/F	D 2
Held	Phase is paused	T/F	D 4
Done	Phase is complete	T/F	D 8
Aborted	Phase has been manually cancelled	T/F	C 1
Intlcked	Phase is currently interlocked	T/F	C 2
			C 4
			C 8
eStatus	Enumerated display of Status bitfields	Enum	□➧

Table 2-1: Block parameters

Parameter	Function	Units		Status
UnitCtrl	Controls the unit	CD hex		➧□➧
Hold	Pauses the unit identified by the recipe	T/F	1	
ReStart	Restarts the unit identified by the recipe	T/F	2 D	
Abort	Terminates the unit identified by the recipe	T/F	4	
			8	
LoadPhse	Phase loading control	T/F		□➧
LoadTim	Maximum time permitted to load Phase	Secs		□➧
LoadTmr	Time remaining until the Phase is loaded	Secs		□➧
LoadErr	Phase failed to load	T/F		➧□➧
CurrStep	Current Step derived from Phase specific SFC	Integer		□➧
NextStep	Defined next step derived from Phase specific SFC	Integer		□➧
ReStStep	Restart defined step following HOLD	Integer		□➧
LastFail	Last step to fail	Integer		□➧
CurrFail	Current step failure	Integer		□➧
Options	Block options	(ABC)D hex		➧□➧
SetU1Alm	TRUE asserts User1 alarm	T/F	1	
SetU2Alm	TRUE asserts User2 alarm	T/F	2 D	
			4	
			8	
Alarms				□➧ 🕮 🔊
Software	Block RAM data sumcheck error/network failure	T/F		
EvalFail	Evaluation failure in block's internal logic	T/F		
LoadFile	File loading error, hardware or software	T/F		
CritAlrm	TRUE, if any CritAlrm bitfield is TRUE	T/F		
Intlcked	TRUE, if any Intlock bitfield is TRUE	T/F		
User1	User alarm 1 asserted from SetU1Alm	T/F		
User2	User alarm 2 asserted from SetU2Alm	T/F		
Combined	OR-ing of all alarm bits	T/F		
A0 to A15	16 Floating-point variables	Integer		□➧
I0, I1	Signed long integer variables	Integer		□➧
Word0 to Word15	16-bit digital variables	ABCD hex		➧□➧
Bit0 to Bit15	Digital 0 to Digital 15	T/F		

Table 2-1: Block parameters

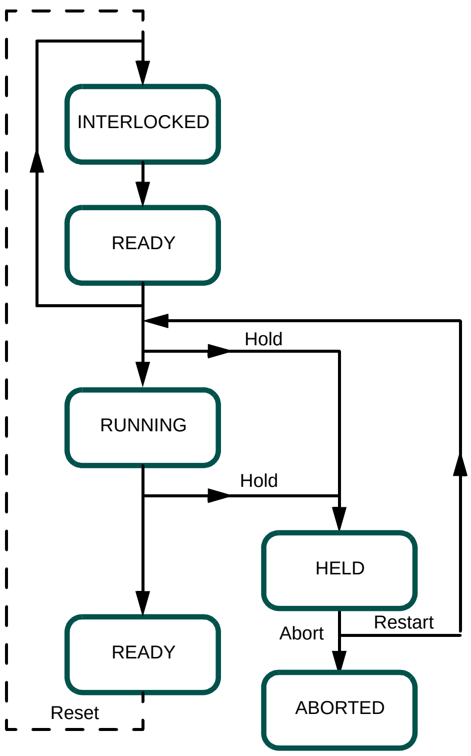
Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Method. (Template/Block/Native). Indicates the location of the block’s ST update routine. By default this is embedded in the template, but the template update routine may be overridden by a different method located in the block pool. (*Currently this field can be set only to ‘Template’.*)

Control. Bitfields providing instrument control derived from selected Phase Controller..

- **Start.** Set TRUE to start the execution of the Phase via the selected Phase controller. This automatically returns to FALSE immediately
- **Hold.** Set TRUE to hold the execution of the current Phase via the selected Phase controller. This automatically returns to FALSE immediately.
- **ReStart.** Set TRUE to remove the Hold control and restart the current Phase via the selected Phase controller. This automatically returns to FALSE immediately.
- **Abort.** Set TRUE to remove the Hold control and terminate the Phase currently in a the Hold control via the selected Phase controller. This automatically returns to FALSE immediately.
- **Reset.** Set TRUE to release the current Phase, ready to repeat the execution. This automatically returns to FALSE when Phase SFC has unloaded.



Buttons. Bitfields providing control of the current Phase via a user interface.

- **Btn1, Btn2.** These bitfields can be linked to User Screens in a visualisation instrument or a SCADA system to provide additional button control of the Phase.

CritAlrm. Bitfields that can be assigned to up to 16 variables that are external to the database and critical to the Phase.

- **Alm0 to Alm15.** If any bitfield is set TRUE the current Phase enters HOLD mode.

Intlocks. Bitfields that can be assigned to up to 16 variables that are external to the database and disable the Phase operation.

- **Ilk0 to Ilk15.** TRUE, if set by the assigned variable. If any bitfield is set TRUE the Phase will not start.

Status. Bitfield used to indicate the Phase status.

- **Ready.** TRUE, shows the Phase is ready to run. This will set FALSE if any external variables assigned to the *Intlocks* bitfields are TRUE.
- **Running.** TRUE, shows the Phase is in operation.
- **Held.** TRUE, shows the Phase is frozen at the current position.
- **Done.** TRUE, shows the Phase has completed successfully.
- **Aborted.** TRUE, shows the Phase was manually terminated.
- **Intlcked.** TRUE, shows the Phase could not start because variable external to the database have set at least one *Intlocks* bitfield TRUE.

eStatus. (Ready/Running/Held/Done/Aborted/Intlckd). Enumerated version of the *Status* bitfields. These can be used to display the current state on a visualisation instrument or SCADA system, see *Status*.

UnitCtrl. Bitfield used to control a unit connected to this block. The unit is specified by the recipe. A sequence of Phases create a recipe that is run in a Batch on a Unit.

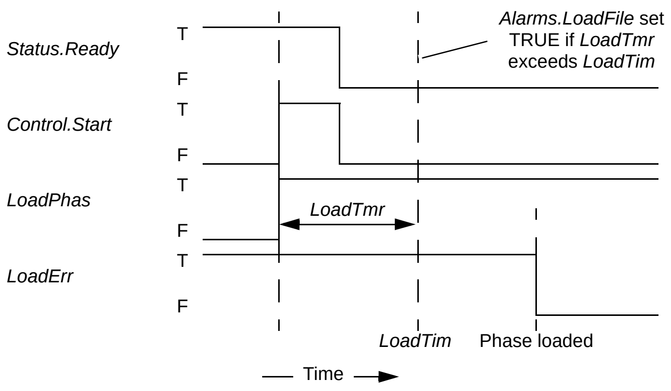
- **Hold.** Set TRUE to freeze the unit at its current position.
- **ReStart.** Set TRUE to restart the unit from the Hold mode
- **Abort.** Set TRUE to terminate the Batch running on the unit.

LoadPhse. Set TRUE to load the Phase SFC. Wire this to *Load* and *Run* of the SFC_CON block in the database.

LoadTim. Used to configure the time limit allowed to load the Phase SFC called by the SFC_CON block. If this period is exceeded *Alarms.LoadFile* is set TRUE.

LoadTmr. Shows the time remaining until the Phase has been loaded. If this count exceeds the time configured in *LoadTim*, *Alarms.LoadFile* is set TRUE.

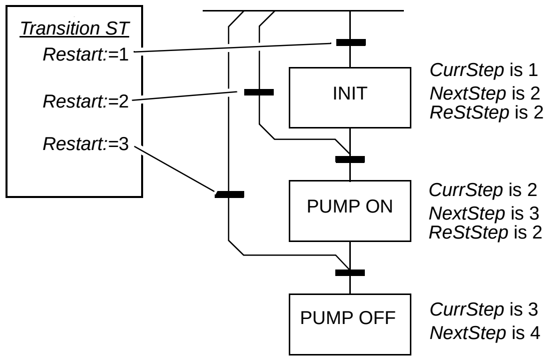
LoadErr. Shows the status of the Phase called by the SFC_CON block. TRUE shows the Phase is not loaded, set FALSE when the Phase has loaded. Normally this will be wired from the SFC_CON *Alarms.Combined* bitfield to assert an alarm in this block.



CurrStep. Shows the step currently active in the Phase when the block is updated by the Phase SFC.

NextStep. Shows the next step in the Phase when the block is updated by the Phase SFC. This is used to indicate which step should be used when the Phase is restarted, *UnitCtrl.ReStart* is set TRUE.

ReStStep. Shows the step that the Phase will restart at when the block is updated by the Phase SFC.



LastFail. Last Phase failure. Shows the previous failure derived from *CurrFail* when updated by the Phase SFC. The cause of the failure can be used to determine the control of the connected node.

CurrFail. Current Phase failure. Shows a failure in the step currently active in the Phase was detected when updated by the Phase SFC.

Options. Bitfield enabling User alarms configured in ST.

- **SetU1Alm and SetU2Alm.** Enables User configured Alarm.

Alarms. See page 11-6 of the *LIN Block Reference Manual* (Part no. HA 375 U003) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **LoadFile.** Asserted if file-loading error caused by either hardware, e.g. faulty drive, or software, e.g non-existent filename or path, faults are present, *LoadErr* value is set.
- **CritAlrm.** Asserted if any subfield in *CritAlrm* is TRUE.
- **Intlcked.** Asserted if any subfield in *Intlocks* is TRUE.
- **User_1 and User_2.** Asserted and cleared according to *Options.SetU1Alm* and *Options.SetU2Alm*, respectively.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

A0 to A15. 16 (sixteen) floating-point variables that can be used as recipe values for the Phase. These can be inputs or outputs via the strategy.

I0 to I1. 2 (two) signed long integer variables that can be used as recipe values for the Phase. These can be inputs or outputs via the strategy.

Word0 to Word5. 6 (six) Word fields containing 16-bit digital variables, *Bit0* to *Bit15*, that can be used as recipe values for the Phase. These can be inputs or outputs via the strategy.

- **Bit0 to Bit15.** Individual 16-bit digital variables per Word field.

Intentionally left blank

CHAPTER 3 CONDITION APPLICATION BLOCKS

The CONDITION category of Application Function Block Templates provides the control strategy with functions for signal conditioning, linearisation, filtering, etc.

Intentionally left blank

VLV3WAY: ANALOGUE ALARM TWO BLOCK

Block function

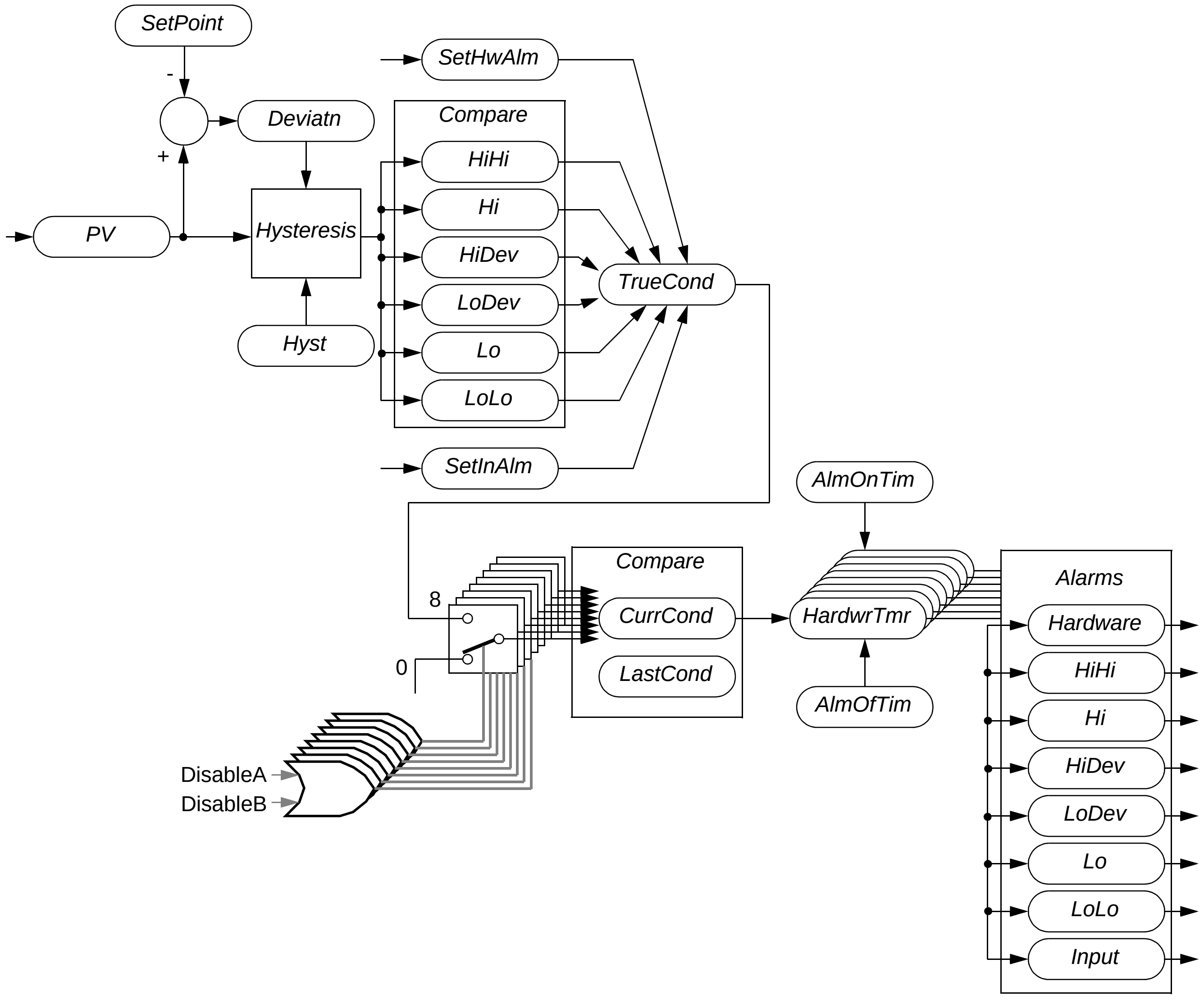


Figure 3-1: Block schematic

The AN_ALM_2 block generates *HiHi*, *Hi*, *Lo* and *LoLo* Absolute alarms, *Hi* and *Lo* Deviation alarms, a *Hardware* alarm which could be used for sensor health and finally, a general purpose *Input* alarm. An adjustable hysteresis band is provided common to the absolute and deviation alarms. All alarms have On and Off separately adjustable delays. Each type of delay and therefore its associated delay value is common to all alarms. Each alarm can be masked (turned off) by either one of two available *Disable* bits.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units			Status		
Method	Reserved for future use block's ST update routine (default = Template)	Enum				📖	
PV	Process variable (Block Input)	Eng			➡☐➡		
HR,LR	PV High & Low graphics range	Eng			➡☐➡		
SetPoint	Setpoint for deviation alarms	Eng			➡☐➡		
Hyst	Hysteresis bandwidth	Eng			➡☐➡		
HiHi	High high alarm level	Eng			➡☐➡	&	
Hi	High alarm level	Eng			➡☐➡	&	
HiDev	High deviation alarm level	Eng			➡☐➡		
LoDev	Low deviation alarm level	Eng			➡☐➡		
LoLo	Low low alarm level	Eng			➡☐➡		
Deviatn	Calculated deviation between PV and SetPoint	Eng			☐➡		
DisableA, DisableB	Alarm disabling fields	(A)BCD hex			➡☐➡		
Hardware	Disables Hardware alarm	T/F	1	D			
HiHi	Disables HiHi alarm	T/F	2	D			
Hi	Disables Hi alarm	T/F	4	D			
HiDev	Disables HiDev alarm	T/F	8	D			
LoDev	Disables LoDev alarm	T/F	1	C			
Lo	Disables Lo alarm	T/F	2	C			
LoLo	Disables LoLo alarm	T/F	4	C			
Input	Disables Input alarm	T/F	8	C			
All	Disables all alarms	T/F	1	B			
			2	B			
			4	B			
			8	B			
Param1	User parameter # 1	Eng			➡☐➡		
Param2	User parameter # 2	Eng			➡☐➡		
Alarms		ABCD hex			☐➡	📖	🔈
Software	Block RAM data sumcheck error/network failure	T/F					
EvalFail	Evaluation failure in block's internal logic	T/F					
Hardware	Hardware alarm	T/F					
HiHi	High high absolute alarm	T/F					
Hi	High absolute alarm	T/F					
HiDev	High deviation alarm	T/F					
LoDev	Low deviation alarm	T/F					
Lo	Low absolute alarm	T/F					
LoLo	Low low absolute alarm	T/F					
Input	Input alarm	T/F					
Combined	OR-ing of all alarm bits	T/F					

Table 3-1: Block parameters

Parameter	Function	Units		Status
HardwTmr to InputTmr	Alarm delay timers	Secs		□▶ 📖
AlmOnTim	On timer delay setting	Secs		▶□▶
AlmOfTim	Off timer delay setting	Secs		▶□▶
TrueCond	True condition flags	CD hex		
Hardware	Hardware alarm	T/F	1 D	
HiHi	High high absolute alarm	T/F	2	
Hi	High absolute alarm	T/F	4	
HiDev	High deviation alarm	T/F	8	
LoDev	Low deviation alarm	T/F	1 C	
Lo	Low absolute alarm	T/F	2	
LoLo	Low low absolute alarm	T/F	4	
Input	Input alarm	T/F	8	
CurrCond	Current condition flags	CD hex		□▶ 📖
LastCond	Last condition flags	CD hex		□▶ 📖
LastAlrm	Last alarm status flags	CD hex		□▶ 📖
Status		(ABC)D hex		□▶ 📖
NewAlarm	New alarm raised	T/F	1 D	
Combined	OR-ing of all LastAlarm bits	T/F	2	
			4	
			8	
Options		(C)D hex		▶□▶
AckNewAl	Resets Status.NewAlarm	T/F	1 D	
SetHwAlm	Sets the Hardware alarm	T/F	2	
SetInAlm	Sets the Input alarm	T/F	4	
InitLast	Forces LastCond to be the inverse of CurrCond	T/F	8	
FpltType	Supervisory faceplate type	String		📖

Table 3-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

PV. Analogue value representing the Process Variable in engineering units.

HR, LR. High and low range for graphic objects (bar, trend) linked to *PV*. They define the 100% and 0% displays, respectively.

SetPoint. Analogue signal representing the Setpoint used for deviation alarms.

Hyst. Hysteresis bandwidth value, applied in engineering units to the absolute and deviation alarms. All alarms are raised immediately after they exceed the limit set, but will not clear until they are within the limit by more than the hysteresis value.

HiHi, Hi, HiDev, LoDev, Lo, LoLo. Alarm levels defining when the block alarms trip, see *Alarms*.

Deviatn. Calculated deviation (error) between the process variable and the Setpoint (*Deviatn* = *PV* - *SetPoint*). This is used to generate the deviation alarms.

DisableA, DisableB. Alarm disabling (masking) fields. A TRUE on any input turns off the named alarm. If unacknowledged, the alarm will still require acknowledging after it has been disabled. *DisableA* and *DisableB* operate in an 'OR' fashion such that a TRUE in either disables the corresponding alarm.

Param1, Param2. Optional parameters. Associate additional floating-point data with the block.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Raised when the field *Options.SetHwAlm* is TRUE. This alarm is intended to allow the associated I/O point failure to be flagged. Typically connected from the I/O block's *Combined* alarm with the I/O block's *Hardware* and *OutRange* alarms enabled.
- **HiHi.** Raised when *PV* is greater than *HiHi*. It will not clear until *PV* has fallen below *HiHi* by more than the hysteresis *Hyst*.
- **Hi.** Raised when *PV* is greater than *Hi*. It will not clear until *PV* has fallen below *Hi* by more than the hysteresis *Hyst*.
- **HiDev.** Raised when *Deviatn* is greater than *HiDev*. It will not clear until *Deviatn* has fallen below *HiDev* by more than the hysteresis *Hyst*.
- **LoDev.** Raised when *Deviatn* is less than *LoDev*. It will not clear until *Deviatn* has risen above *LoDev* by more than the hysteresis *Hyst*.
- **Lo.** Raised when *PV* is less than *Lo*. It will not clear until *PV* has risen above *Lo* by more than the hysteresis *Hyst*.
- **LoLo.** Raised when *PV* is less than *LoLo*. It will not clear until *PV* has risen above *LoLo* by more than the hysteresis *Hyst*.
- **Input.** Raised when the field *Options.SetInAlm* is TRUE.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

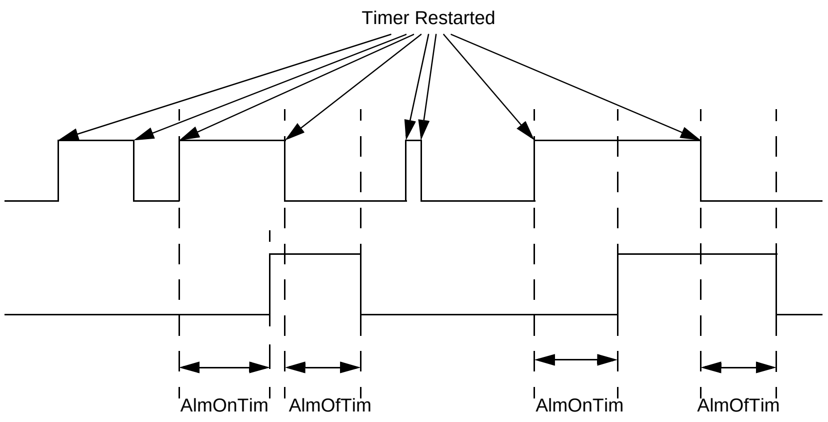
HardwTmr, HiHiTmr, HiTmr, HiDevTmr, LoDevTmr, LoTmr, LoLoTmr, InTmr. Alarm delay timers. Every time the block is processed *CurrCond* is updated with the current alarm condition based upon *PV*, *SetPoint*, limits and disables. If *CurrCond* is different from *LastCond*, the appropriate *xxxxxTmr* field is loaded with the value of *AlmOnTim* (if *CurrCond* is TRUE) or *AlmOfTim* (if *CurrCond* is FALSE). When the timer has reached zero, the alarm state is updated from *CurrCond*.

TrueCond. Subfields corresponding to the *Alarms* subfields show which condition is actually occurring, regardless of timers and disables.

CurrCond. Subfields corresponding to the *Alarms* subfields show condition is set after all disables have been taken into consideration.

LastCond. Subfields corresponding to the *Alarms* subfields show the status of *CurrCond* at the previous iteration.

LastAlrm. This subfield shows the latest status the alarms were updated to after the time delays and the disables have been applied.



Note	Also applies to *LastAlrm.Hardware*, *LastAlrm.HiHi*, *LastAlrm.Hi*, *LastAlrm.HiDev* *LastAlrm.LoDev*, *LastAlrm.Lo*, *LastAlrm.LoLo*, and *LastAlrm.Input*.

Figure 3-2: Derivation of an output

Status. Subfield reflecting the status of the alarm processing

- **NewAlarm.** Whenever a new alarm is raised (after time delay), this bit is set TRUE. This could be used in an alarm management scheme to generate an external horn signal when a new alarm occurs. *Status.NewAlarm* can be reset by setting *Options.AckNewAl* TRUE. To get automatic reset after one database scan, wire *Status.NewAlarm* to *Options.AckNewAl.*
- **Combined.** 'OR' of all *LastAlrm* bits. Provides a replacement for the field *Alarms.Combined* when the alarm priorities are set to zero.

Options. Bitfield for selecting the different operational options of the block.

- **AckNewAl.** Resets *Status.NewAlarm.*
- **SetHwAlm.** The *Hardware* alarm is raised when this bit is TRUE.
- **SetInAlm.** The *Input* alarm is raised when this bit is TRUE.
- **InitLast.** Setting this bit TRUE forces *LasCond* to be the inverse of *CurrCond*. This feature could be useful at start-up.

FpltType. Faceplate type. Associates a faceplate type or other alphanumeric string with the block.

Implementation notes

If AN_ALM_2 control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2694 bytes and by 226 bytes for each instance of the control module.

Note	For this application module please refer to factory.

Intentionally left blank

VLV3WAY: GAS DENSITY BLOCK

Block function

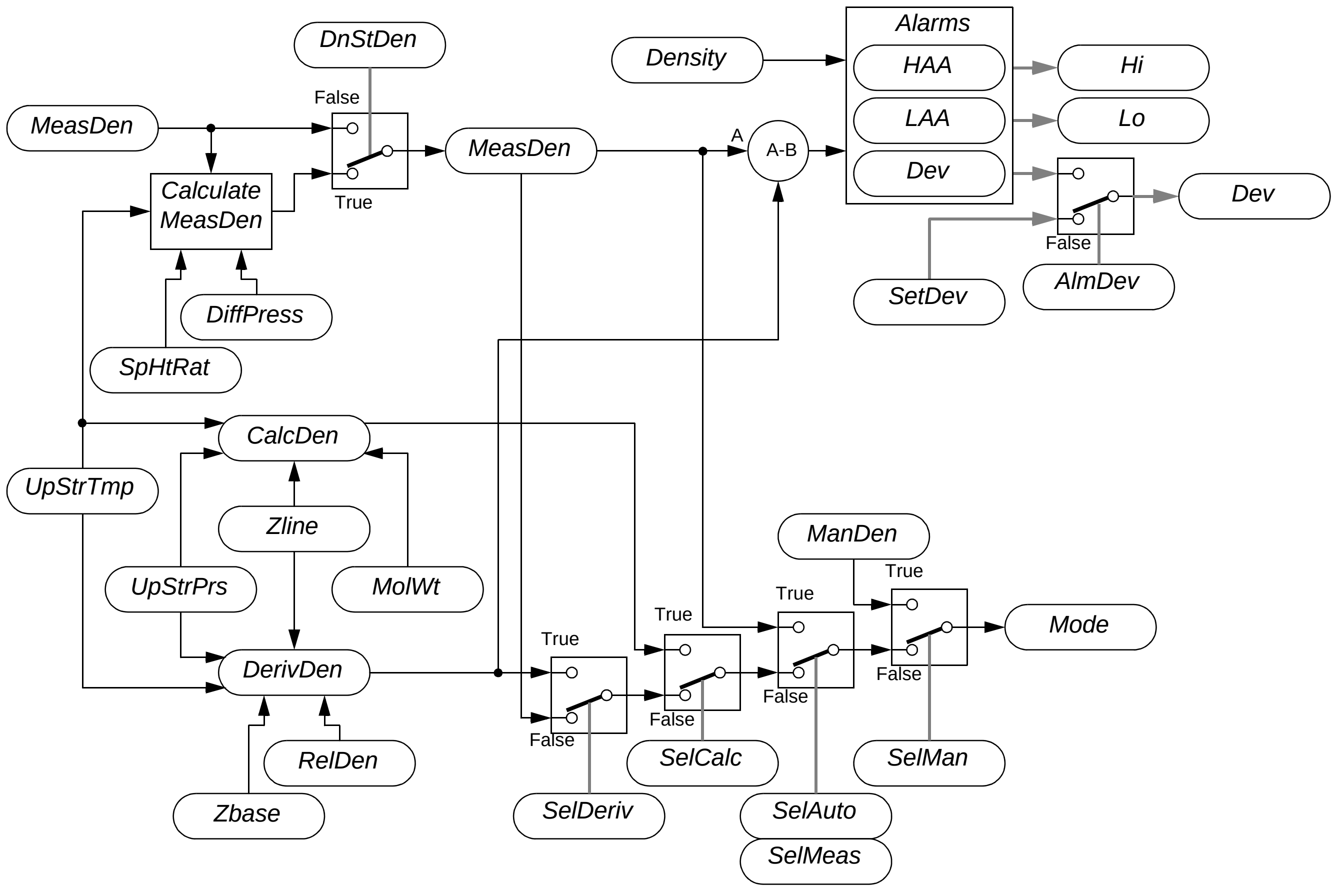


Figure 3-1: Block schematic

This block generates an up-stream density for use with the ISO5167O block or the ISO5167V block. It generates density figures by calculation, allows a live density to be wired in and an overall switch is provided to select the desired density.

The block assumes that pressure and temperature inputs are in barG and degC, respectively. These are then converted to barA and K for use in the appropriate calculations.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block’s ST update routine (default = Template)	Enum	📖
DiffPress	Differential pressure	mBar	▶☐▶
UpStrPrs	Upstream static line pressure	barG	▶☐▶
UpStrTmp	Upstream line temperature	degC	▶☐▶
LiveDen	Live density	kg/m^3	▶☐▶
SpHtRat	Specific heat ratio	ABCD hex	▶☐▶ &
RelDen	Relative density		▶☐▶ &
MolWt	Molecular weight	g/mol	▶☐▶
Zbase	Base compressibility factor		▶☐▶
Zline	Line compressibility factor		▶☐▶
HR	Graphical high range	kg/m^3	▶☐▶
LR	Graphical low range	kg/m^3	▶☐▶
HAA	In use density high alarm limit	kg/m^3	▶☐▶
LAA	In use density low alarm limit	kg/m^3	▶☐▶
Dev	In use density deviation alarm limit	kg/m^3	▶☐▶
Hyst	Alarm hysteresis		▶☐▶
Alarms		ABCD hex	☐▶ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block’s internal logic	T/F	
Override	Block in manual mode	T/F	
Hi	High absolute alarm	T/F	
Lo	Low absolute alarm	T/F	
Dev	Deviation alarm	T/F	
Combined	OR-ing of all alarm bits	T/F	
Mode	Current operating mode	Enum	
SelMode	Mode selector	CD hex	▶☐▶ 📖
SelMan	Orifice diameter corrected to line conditions	T/F	1 D
SelMeas	Pipe diameter corrected to line conditions	T/F	2 D
SelAuto	Diameter ratio	T/F	4 D
SelCalc	Expansibility (expansion) factor	T/F	8 D
SelDeriv	Velocity of approach factor	T/F	1 C
			2 C
			4 C
			8 C

Table 3-1: Block parameters

Parameter	Function	Units		Status
ModeAct	Mode active	CD hex		□➡ 📖
ManAct	Manual mode active	T/F	1 D	
MeasAct	Measured mode active	T/F	2 D	
AutoAct	Auto mode active	T/F	4 D	
CalcAct	Calculated mode active	T/F	8 D	
DerivAct	Derived mode active	T/F	1 C	
			2 C	
			4 C	
			8 C	
Density	Upstream density	kg/m^3		□➡ 📖
ManDen	Manual input density	kg/m^3		➡□➡
MeasDen	Measured density	kg/m^3		□➡ 📖
CalcDen	Calculated density	kg/m^3		□➡ 📖
DerivDen	Derived density	kg/m^3		□➡ 📖
Status		(ABC)D hex		📖
Hi	Status of the Hi alarm	T/F		
Lo	Status of the Lo alarm	T/F		
Dev	Status of the Dev alarm	T/F		
Options		(ABC)D hex		➡□➡
DnStrDen	Used in the derivation of MeasDen	T/F		
AlmDev	Used in the derivation of the Dev alarm	T/F		
SetDev	Used in the derivation of the Dev alarm	T/F		
Param1	User parameter # 1	Eng		➡□➡
Param2	User parameter # 2	Eng		➡□➡
FpltType	Supervisory faceplate type	String		

Table 3-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

DiffPress. Differential pressure measurement representing the raw flowrate in mbar.

UpStrPrs. Measurement representing the upstream static line pressure in barG.

UpStrTmp. Measurement representing the upstream line temperature in degC. This could be derived from the downstream temperature in the ISO5167O block.

LiveDen. Measurement representing the live density, usually from downstream of the orifice plate or Venturi tube. May be corrected to upstream conditions by setting *Options.DnStrDen* TRUE. The resulting measured density is stored in *MeasDen*.

SpHtRat. Specific heat ratio used to calculate the pressure loss.

RelDen. Gas relative density. Normally wired in from the ISO5167O block.

MolWt. Gas molecular weight. Can be an operator entry or wired in from the AGA8DATA type block.

Zbase. Base compressibility factor for the gas. Can be an operator entry or wired in from the AGA8DATA type block.

Zline. Compressibility factor for the gas at current stream conditions. Can be wired in from the AGA8DATA block.

HR, LR. Graphical high and low ranges in kg/m$_3$ used to bound the density fields such that a proportional signal may be generated to allow trending, bar graph fill etc.

HAA, LAA. In use density high and low alarm limits in kg/m^3 see Alarms.

Dev. In use density deviation alarm limit in kg/m^3, see Alarms.

Hyst. Alarm hysteresis in kg/m^3, see Alarms.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Override.** Raised when mode is Manual. Cleared when mode is anything other than Manual.
- **Hi.** Raised when the in use density (field *Density*) is greater than *HAA*. It will not clear until the density has fallen below *HAA* by more than the hysteresis *Hyst*.
- **Lo.** Raised when the in use density (field *Density*) is lower than *LAA*. It will not clear until the density has risen above *LAA* by more than the hysteresis *Hyst*.
- **Dev.** The derivation of this alarm is dependent upon the option bit *Options.AlmDev*. If set to 'Int', the alarm is raised when the *MeasDens* deviates from the density by more than the deviation limit *Dev*. It will not clear until the deviation is less than the limit by more than the hysteresis. If *Options.AlmDev* is set to 'Ext', the alarm follows the status of *Options.SetDev*.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Mode. (Manual/Meas/Auto/Calc/Deriv). Controls which of the density figures is routed through as the output density figure *Density*. In Auto (default mode), *MeasDen* is routed to *Density*.

SelMode. (SelMan/SelMeas/SelAuto/SelCalc/SelDeriv). Bitfield for selecting the block's *Mode*. These bits are prioritised such that if more than one bit is true then the selected mode can be predicted. The priority order is shown above (highest first).

ModeAct. (ManAct/MeasAct/AutoAct/CalcAct/DerivAct). Bitfield indicating the block's active mode, i.e. the value of the *Mode* parameter. Only one *ModeAct* bit can be active at one time.

Density. Upstream density in kg/m^3 derived from any of the four density fields below depending on the block's mode. This is intended to be passed to the ISO5167O block type.

ManDen. Upstream density in kg/m^3. Can be an Operator entry used during calibration to set the density output of the block.

CalcDen. Calculated upstream density in kg/m^3. This is the result of the calculation below

CalcDen = (*UpStrPrs*+1.01325)**MolWt* / (*Zline**R*(*UpStrTmp*+273.15))

Where R is the universal gas constant (0.0831434 bar.m$_3$/kmol/degK).

DerivDen. Calculated upstream density in kg/m^3. This is the result of the calculation below

DerivDen = (*RelDen**1.22550)*288.15*(*UpStrPrs*+1.01325)**Zbase* /

((*UpStrTmp*+273.15)*1.01325**Zline*)

MeasDen. This value is used as the measured density in kg/m^3. If *Options.DnStrDen* is FALSE, *MeasDen* is equal to *LiveDen*. Otherwise *MeasDen* is obtained from the calculation below

MeasDen = *LiveDen**((UpStrPrs+1.01325)/

(*UpStrPrs*+1.01325-*DiffPrs*/1000))^(1/*SpHtRat*)

Status. Bitfield showing status of the calculations.

- **Hi.** Follows the status of the *Hi* alarm. This bit is set and cleared irrespective of alarm priority.
- **Lo.** Follows the status of the *Lo* alarm. This bit is set and cleared irrespective of alarm priority.
- **Dev.** Follows the status of the *Dev* alarm. This bit is set and cleared irrespective of alarm priority.

Options. Bitfield for selecting the different operational options of the block.

- **DnStrDen.** Used in the derivation of the measured density, see *MeasDen*.
- **AlmDev.** Used in the derivation of the deviation alarm, see Alarms.
- **SetDev.** Used in the derivation of the deviation alarm when *Options.AlmDev* = Ext, see Alarms.

Param1, Param2. Optional parameters. Associate additional floating-point data with the block.

FpltType. Faceplate type. Associates a faceplate type or other alphanumeric string with the block.

Implementation notes

If GasDens control module blocks are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 1574 bytes and by 180 bytes for each instance of the control module.

Note	For this application module please refer to factory.

Intentionally left blank

VLV3WAY: FLOW CALCULATION FOR ORIFICE PLATE BLOCK

Block function

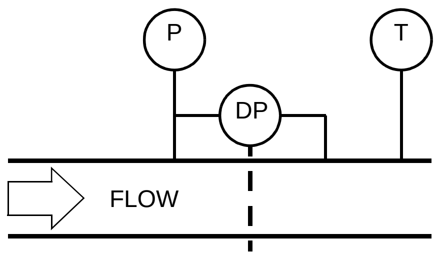


Figure 3-1: Typical orifice plate metering system

The ISO5167O block is implemented against the ISO 5167-1:1991(E) Standard.

The block calculates and displays the mass flowrate, Reynolds number, etc. but does not perform any uncertainty calculations. The block assumes that pressure and temperature inputs are in barG and degC, respectively. These are then internally converted to barA and K for use in the appropriate calculations.

The use of upstream and downstream values is critical in obtaining the correct results; the block clarifies this using specific field names within the block template. The block does not include any means of overriding any real I/O, either manually or automatically (e.g. in the event of I/O hardware failure).

A real gas or liquid orifice plate metering system is normally set up as shown schematically below, with a differential pressure measurement together with an upstream pressure and a downstream temperature measurement.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
DiffPress	Differential pressure	mBar	►□►
UpStrPrs	Upstream static line pressure	barG	►□►
DnStrTmp	Downstream line temperature	degC	►□►
UpStrDen	Upstream density	kg/m3	►□►
OrifCalD	Orifice plate calibration diameter	mm	►□►
OrifCalT	Orifice plate calibration temperature	degC	►□►
OrifExpF	Orifice plate expansion factor	mm/degC	►□►
PipeCalD	Pipe calibration diameter	mm	►□►
PipeCalT	Pipe calibration temperature	degC	►□►
PipeExpF	Pipe expansion factor	mm/degC	►□►
IsentExp	Isentropic exponent		►□►
DynaVisc	Dynamic viscosity	cP	►□►
RelDen	Relative density		►□► &

Table 3-1: Block parameters

Parameter	Function	Units	Status
SpHtRat	Specific heat ratio	ABCD hex	➡➡ &
Tapping	Arrangement of the dp tapping on the pipe	Enum	
Alarms		ABCD hex	➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
BetaOOR	Beta ratio out of range	T/F	
ReyNoOOR	Reynolds number out of range	T/F	
Converge	Convergence error	T/F	
Combined	OR-ing of all alarm bits	T/F	
MsFlRate	Mass flowrate	kg/hr	➡ 📖
VsFlRate	Standard volumetric flowrateStandard volumetric flowrate	Sm3/hr	➡ 📖
OrifDiam	Orifice diameter corrected to line conditions	mm	➡ 📖
PipeDiam	Pipe diameter corrected to line conditions	mm	➡ 📖
Beta	Diameter ratio		➡ 📖
ExpFact	Expansibility (expansion) factor		➡ 📖
VelAppr	Velocity of approach factor		➡ 📖
C_invar	Intermediate value for calculation of C		➡ 📖
C	Discharge coefficient		➡ 📖
Alpha	Flow coefficient		➡ 📖
ReyNo	Reynolds number		➡ 📖
DnStrPrs	Downstream static line pressure	barG	➡ 📖
UpStrTmp	Upstream line temperature	degC	➡ 📖
Options	Block options	ABCD hex	➡ 📖
Fluid	Fluid type (Gas/Liquid)	T/F	1
AlmBeta	Method for derivation of the BetaOOR alarm	T/F	2 D
SetBeta	External BetaOOR alarm	T/F	4
AlmReyNo	Method for derivation of the ReyNoOOR alarm	T/F	8
SetReyNo	External ReyNoOOR alarm	T/F	1
Bit5	User option #5	T/F	2 C
Bit6	User option #6	T/F	4
Bit7	User option #7	T/F	8
Bit8	User option #8	T/F	1
Bit9	User option #9	T/F	2 B
Bit10	User option #10	T/F	4
Bit11	User option #11	T/F	8
Bit12	User option #12	T/F	1
Bit13	User option #13	T/F	2 A
Bit14	User option #14	T/F	4
Bit15	User option #15	T/F	8

Table 3-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

DiffPress. Differential pressure measurement representing the raw flowrate in mBar.

UpStrPrs. Measurement representing the upstream static line pressure in barG.

DnStrTmp. Measurement representing the downstream line temperature in degC. This is used to correct the pipe and orifice diameters.

Tapping (Flange,D_or_D/2,Corner). Selects the current arrangement of the differential pressure tapping on the pipe. The selection automatically calls up the correct constants used in the calculation of the coefficient of discharge (*C*) as defined in the ISO5167 Standard.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **BetaOOR.** Beta ratio is out of range. This alarm may be derived internally or externally as defined by the option bit *AlmBeta*. When this option bit is set to Int, the *BetaOOR* alarm is derived internally and is raised if *Beta* < 0.2 or Beta > 0.6. When *AlmBeta* = Ext, the *BetaOOR* alarm follows the *Option.SetBeta* bit (True = alarm, False = no alarm).
- **ReyNoOOR.** Reynolds number is out of range. This alarm may be derived internally or externally as defined by the option bit *AlmReyNo*. When this option bit is set to Int, the *ReyNoOOR* alarm is derived internally and is raised if *ReyNo* (-3150). When *AlmReyNo* = Ext, the *ReyNoOOR* alarm follows the *Option.SetReyNo* bit (True = alarm, False = no alarm).
- **Converge.** Iteration to find the coefficient of discharge (*C*) has not converged. The alarm clears following the next successful convergence. If the convergence fails, the flowrate is left at its last calculated value.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

MsFlRate. Calculated mass flowrate in kg/hr.

VsFlRate. Calculated standard volumetric flowrate in Sm3/hr. This calculation depends upon the setting of *Options.Fluid*.

VsFlRate = *MsFlRate*/*RelDen* * 1.22550 for gas, where 1.22550 is the air density determined from the British Gas Instrumentation and Control Guide (reference temperature of 15 degC and reference pressure of 1.01325 bar).

VsFlRate = *MsFlRate*/*RelDen* * 999.0121 for liquid, where 999.0121 is the water density determined from API2540 (reference temperature of 15.6 degC and reference pressure of 1.01325 bar).

ExpFact. Calculated expansibility (expansion) factor. This value is dependent upon the type of process fluid defined by the setting of the bitfield *Options.Fluid*.

C_invar. Internal field used during the iteration to find the coefficient of discharge.

C. Calculated discharge coefficient given by the Stoltz equation.

ReyNo. Calculated Reynolds number as referred to the upstream conditions of the fluid and the upstream diameter of the pipe.

DnStrPrs. Calculated value reflecting the static line pressure at the downstream conditions in barG. For use by external density calculations.

Intentionally left blank

VLV3WAY: FLOW CALCULATION FOR VENTURI TUBES BLOCK

Block function

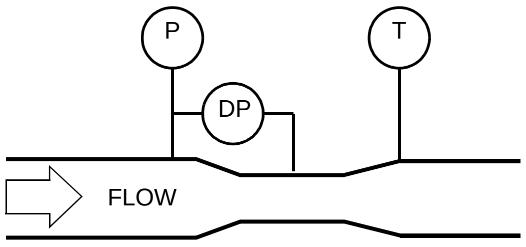


Figure 3-1: Typical Venturi tube metering system

The ISO5167V block is implemented against the ISO 5167-1:1991(E) Standard.

The block calculates and displays the mass flowrate, Reynolds number, etc. but does not perform any uncertainty calculations. The block assumes that pressure and temperature inputs are in barG and degC, respectively. These are then internally converted to barA and K for use in the appropriate calculations.

The use of upstream and downstream values is critical in obtaining the correct results; the block clarifies this using specific field names within the block template. The block does not include any means of overriding any real I/O, either manually or automatically (e.g. in the event of I/O hardware failure).

A real gas or liquid venturi metering system is normally set up as shown schematically below, with a differential pressure measurement together with an upstream pressure and a downstream temperature measurement.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
DiffPress	Differential pressure	mBar	⮞☐⮞
UpStrPrs	Upstream static line pressure	barG	⮞☐⮞
DnStrTmp	Downstream line temperature	degC	⮞☐⮞
UpStrDen	Upstream density	kg/m3	⮞☐⮞
ThrtCalD	Throat calibration diameter	mm	⮞☐⮞
ThrtCalT	Throat calibration temperature	degC	⮞☐⮞
ThrtExpF	Thraot expansion factor	mm/degC	⮞☐⮞
PipeCalD	Pipe calibration diameter	mm	⮞☐⮞
PipeCalT	Pipe calibration temperature	degC	⮞☐⮞
PipeExpF	Pipe expansion factor	mm/degC	⮞☐⮞
IsentExp	Isentropic exponent		⮞☐⮞
DynaVisc	Dynamic viscosity	cP	⮞☐⮞
RelDen	Relative density		⮞☐⮞ &
SpHtRat	Specific heat ratio	ABCD hex	⮞☐⮞ &
Venturi	Venturi type	Enum	

Table 3-1: Block parameters

Parameter	Function	Units	Status
Alarms		ABCD hex	➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block’s internal logic	T/F	
BetaOOR	Beta ratio out of range	T/F	
ReyNoOOR	Reynolds number out of range	T/F	
Combined	OR-ing of all alarm bits	T/F	
MsFlRate	Mass flowrate	kg/hr	➡ 📖
VsFlRate	Standard volumetric flowrateStandard volumetric flowrate	Sm3/hr	➡ 📖
ThrtDiam	Throat diameter corrected to line conditions	mm	➡ 📖
PipeDiam	Pipe diameter corrected to line conditions	mm	➡ 📖
Beta	Diameter ratio		➡ 📖
ExpFact	Expansibility (expansion) factor		➡ 📖
VelAppr	Velocity of approach factor		➡ 📖
Tau	Pressure ratio		➡ 📖
C	Discharge coefficient		➡ 📖
Alpha	Flow coefficient		➡ 📖
ReyNo	Reynolds number		➡ 📖
DnStrPrs	Downstream static line pressure	barG	➡ 📖
UpStrTmp	Upstream line temperature	degC	➡ 📖
Options	Block options	ABCD hex	➡ 📖
Fluid	Fluid type (Gas/Liquid)	T/F	1 D
AlmBeta	Method for derivation of the BetaOOR alarm	T/F	2
SetBeta	External BetaOOR alarm	T/F	4
AlmReyNo	Method for derivation of the ReyNoOOR alarm	T/F	8
SetReyNo	External ReyNoOOR alarm	T/F	1 C
Bit5	User option #5	T/F	2
Bit6	User option #6	T/F	4
Bit7	User option #7	T/F	8
Bit8	User option #8	T/F	1 B
Bit9	User option #9	T/F	2
Bit10	User option #10	T/F	4
Bit11	User option #11	T/F	8
Bit12	User option #12	T/F	1 A
Bit13	User option #13	T/F	2
Bit14	User option #14	T/F	4
Bit15	User option #15	T/F	8

Table 3-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

DiffPress. Differential pressure measurement representing the raw flowrate in mBar.

UpStrPrs. Measurement representing the upstream static line pressure in barG.

DnStrTmp. Measurement representing the downstream line temperature in degC. This is used to correct the pipe and orifice diameters.

Venturi. (AsCast, Machined, RuffWeld, Nozzle). Selects the current Venturi type. The selection automatically calls up the correct constants as defined in the ISO5167 Standard.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **BetaOOR.** Beta ratio is out of range. This alarm may be derived internally or externally as defined by the option bit AlmBeta. When this option bit is set to Int, the *BetaOOR* alarm is derived internally and is raised if:

  Beta < 0.3 or *Beta* > 0.75 for *AsCast*

  Beta < 0.4 or *Beta* > 0.75 for *Machined*

  Beta < 0.4 or *Beta* > 0.75 for *RuffWeld*

  Beta < 0.316 or *Beta* > 0.75 for *Nozzle*

  When *AlmBeta* = Ext, the *BetaOOR* alarm follows the *Option.SetBeta* bit (True = alarm, False = no alarm).
- **ReyNoOOR.** Reynolds number is out of range. This alarm may be derived internally or externally as defined by the option bit *AlmReyNo*. When this option bit is set to Int, the *ReyNoOOR* alarm is derived internally and is raised if *ReyNo* (-3150). When *AlmReyNo* = Ext, the *ReyNoOOR* alarm follows the *Option.SetReyNo* bit (True = alarm, False = no alarm).
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

MsFlRate. Calculated mass flowrate in kg/hr.

VsFlRate. Calculated standard volumetric flowrate in Sm3/hr. This calculation depends upon the setting of *Options.Fluid*.

> *VsFlRate* = *MsFlRate*/*RelDen* * 1.22550 for gas, where 1.22550 is the air density determined from the British Gas Instrumentation and Control Guide (reference temperature of 15 degC and reference pressure of 1.01325 bar)
>
> *VsFlRate* = *MsFlRate*/*RelDen* * 999.0121 for liquid, where 999.0121 is the water density determined from API2540 (reference temperature of 15.6 degC and reference pressure of 1.01325 bar).

ExpFact. Calculated expansibility (expansion) factor. This value is dependent upon the type of process fluid defined by the setting of the bitfield *Options.Fluid*.

Tau. Calculated pressure ratio.

C. Calculated discharge coefficient.

ReyNo. Calculated Reynolds number as referred to the upstream conditions of the fluid and the upstream diameter of the pipe.

DnStrPrs. Calculated value reflecting the static line pressure at the downstream conditions in barG. For use by external density calculations.

UpStrTmp. Calculated value reflecting the line temperature at the upstream conditions. For use by external density calculations.

Options. Bitfield for selecting different operational options of the block.

- **Fluid (Gas/Liquid).** Used in the derivation of the term *ExpFact* and *VsFlRate*.
- **AlmBeta (Int/Ext).** Used in the derivation of the *BetaOOR* alarm, see Alarms.
- **SetBeta.** Used in the derivation of the *BetaOOR* alarm, see Alarms.
- **AlmReyNo (Int/Ext).** Used in the derivation of the *ReyNoOOR* alarm, see Alarms.
- **SetReyNo.** Used in the derivation of the *ReyNoOOR* alarm, see Alarms.

Implementation notes

If ISO5167V custom LIN blocks are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 1812 bytes and by 186 bytes for each instance of the custom LIN blocks.

Note For this application module please refer to factory.

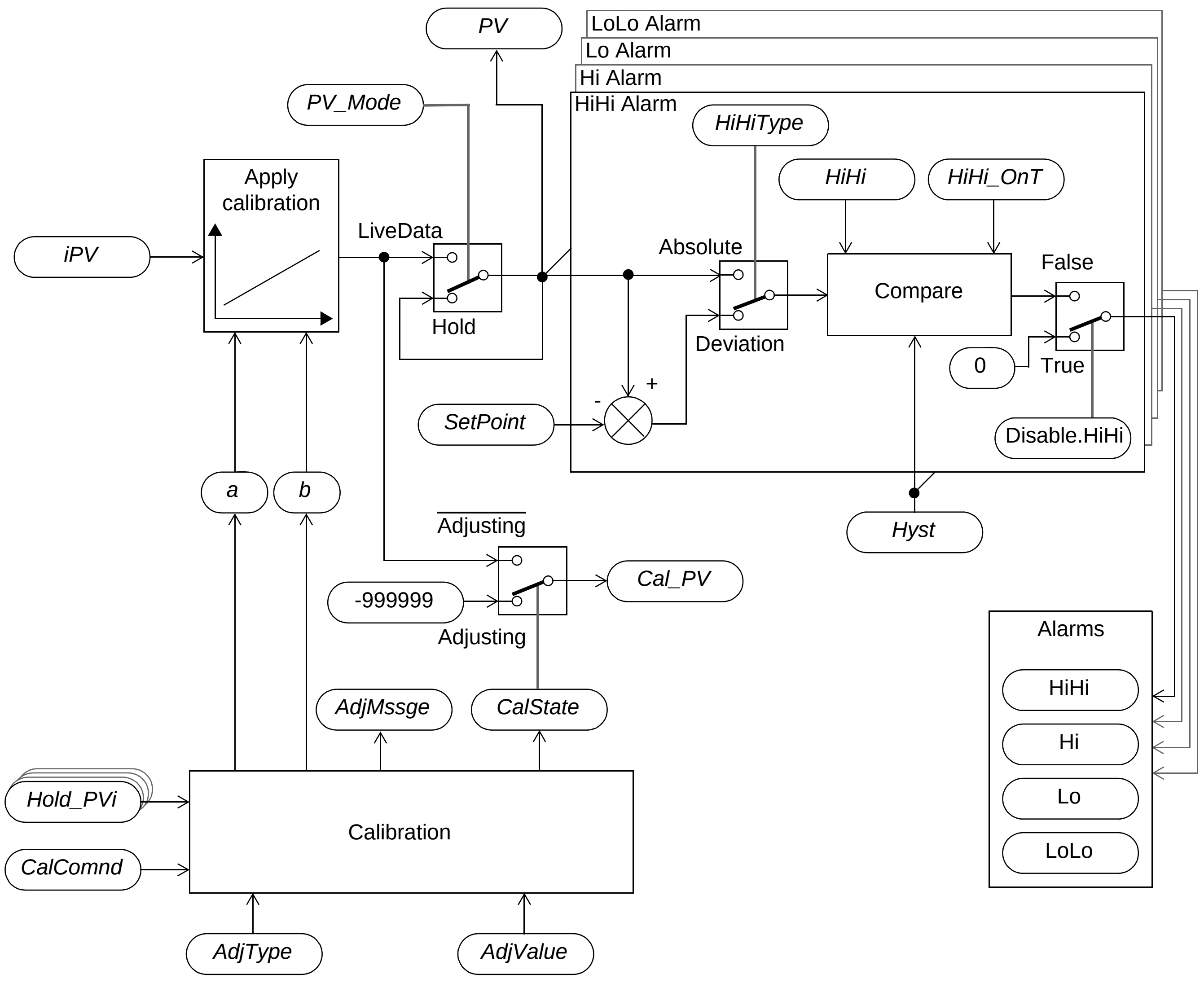
VLV3WAY: EMS ANALOGUE INPUT ALARM BLOCK

Block function

The VLV3WAY block is a point interface with alarm functions and a calibration facility designed specifically to work closely with the universal analogue input channel function block type, AI_UIO. The VLV3WAY block takes the real process inputs, provides user calibration of the engineering value, and generates alarms and grouped status signals for “critical alarm”, “warning alarm”, and three separate “new alarm” flags (with manual reset). The alarms include system style and process alarms as shown below:

- The system alarms include a calibration warning alarm, *CalWarng* (based upon a last calibration date and calibration interval), and a bad calibration alarm, *Bad_Cal* for use in complementing the site’s calibration management system. In addition, a *Hardware* alarm signals hardware-related issues (such as missing I/O modules), and a *PV Held* alarm signals when the *PV* value is being held (normally raised during calibration checks or adjustments).
- Four process alarms: *HiHi*, *Hi*, *Lo*, *LoLo*, each of which are selectable as absolute or deviation alarms – each with its own independent on-delay and a common setpoint (for deviation calculation) and common hysteresis. A digital *Input* alarm with on-delay is provided, driven by an option for use as a door alarm or communications failure to a sensor device (a humidity sensor, for example).

To ensure that data recording and online status displays can correctly reflect the state of the analogue value (*PV*), a mode indicator is provided (*PV_Mode*) to clearly identify the process variable as being derived from *LiveData* or *Hold* (*PV* maintains the last value during calibration checking or adjustment).



Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following the table.

Parameter		Function	Units	Status
Alarms				□➧ 🕮 🔈
	Software	Block RAM data sumcheck error/network failure	T/F	
	EvalFail	Evaluation failure in block's internal logic	T/F	
	Hardware	Hardware alarm	T/F	
	Bad_Cal	Bad calibration alarm	T/F	
	HiHi	High high alarm (absolute or deviation)	T/F	
	Hi	High alarm (absolute or deviation)	T/F	
	Lo	Low alarm (absolute or deviation)	T/F	
	LoLo	Low low alarm (absolute or deviation)	T/F	
	Input	Input alarm	T/F	
	PV_Held	Process value held alarm	T/F	
	CalWarng	Calibration warning alarm	T/F	
	Combined	OR-ing of all alarm bits	T/F	
Method		Reserved for future use. Block's ST update routine (default = Template)	Menu	🕮
PV_Mode		The PV update source (LiveData/Hold)	Menu	🕮
PV		Process variable	EngA	□➧ 🕮
HR		Engineering high range	EngA	➧□➧
LR		Engineering low range	EngA	➧□➧
ChainIn		Chain input	(A)BCD hex	➧□
	Init	An initialise input	T/F	1 D
	CritAlm	Chained input critical alarm status	T/F	2 D
	WarnAlm	Chained input warning alarm status	T/F	4 D
	NewAlmA	Chained input new alarm A status	T/F	8 D
	NewAlmB	Chained input new alarm B status	T/F	1 C
	NewAlmC	Chained input new alarm C status	T/F	2 C
	RsetNewA	Chained input reset new alarm A status	T/F	4 C
	RsetNewB	Chained input reset new alarm B status	T/F	8 C
	RsetNewC	Chained input reset new alarm C status	T/F	1 B
				2 B
				4 B
				8 B

Table 3-1: Block parameters

Parameter	Function	Units			Status
ChainOut	Chain output	(A)BCD hex			□▶ 📖
Init	A copy of chain input init (*ChainIn.Init*)	T/F	1	D	
CritAlm	Chained output critical alarm status	T/F	2		
WarnAlm	Chained output warning alarm status	T/F	4		
NewAlmA	Chained output new alarm A status	T/F	8		
NewAlmB	Chained output new alarm B status	T/F	1	C	
NewAlmC	Chained output new alarm C status	T/F	2		
RsetNewA	Chained output reset new alarm A status	T/F	4		
RsetNewB	Chained output reset new alarm B status	T/F	8		
RsetNewC	Chained output reset new alarm C status	T/F	1	B	
			2		
			4		
			8		
ThisAlm	Alarm state	(A)BCD hex			▶□▶
Hardware	Hardware “in alarm” status after alarm suppression	T/F	1	D	
Bad_Cal	Bad calibration “in alarm” status after alarm suppression	T/F	2		
HiHi	High high “in alarm” status after alarm suppression	T/F	4		
Hi	High “in alarm” status after alarm suppression	T/F	8		
Lo	Low “in alarm” status after alarm suppression	T/F	1	C	
LoLo	Low low “in alarm” status after alarm suppression	T/F	2		
Input	input “in alarm” status after alarm suppression	T/F	4		
PV_Held	PV Held “in alarm” status after alarm suppression	T/F	8		
CalWarng	Calibration warning “in alarm” status after alarm suppression	T/F	1	B	
			2		
			4		
			8		
CritAlmM*	Critical alarm mask to determine the alarms included in the critical alarm status (*Status.CritAlm*). Bit field names are the same as for *ThisAlm*.	(A)BCD hex			▶□▶
WarnAlmM*	Warning alarm mask to determine the alarms included in the warning alarm status (*Status.WarnAlm*)	(A)BCD hex			▶□▶
NewAlmMA*	New alarm mask A, similar to *CritAlmM*, but for the New Alarm A status	(A)BCD hex			▶□▶
NewAlmMB*	New alarm mask B, similar to *NewAlmMA*, but for the New Alarm B status	(A)BCD hex			▶□▶
NewAlmMC*	New alarm mask C, similar to NewAlmMA, but for the New Alarm C status	(A)BCD hex			▶□▶
Options	Sets the options for the block	(ABC)D hex			▶□▶
Init	Sets internal flags used in the derivation of the time delay alarms		1	D	
SetInAlm	Sets the *Input* alarm subject to the on-delay timer		2		
			4		
			8		

Table 3-1: Block parameters

Parameter	Function	Units			Status
Status	The block’s status	(AB)CD hex			□➧ 📖
CritAlm	Critical alarm status, subject to the *CritAlmM* mask	T/F	1	D	
WarnAlm	Warning alarm status, subject to the *WarnAlmM* mask	T/F	2	D	
NewAlmA	New alarm status A, subject to the *NewAlmMA* mask	T/F	4	D	
NewAlmB	New alarm status B, subject to the *NewAlmMB* mask	T/F	8	D	
NewAlmC	New alarm status C, subject to the *NewAlmMC* mask	T/F	1	C	
LiveData	Set TRUE when *PV_Mode* is indicating “LiveData”	T/F	2	C	
PV_Held	Set TRUE when *PV_Mode* is indicating “Hold”	T/F	4	C	
			8	C	
SetPoint	The Setpoint for the process deviation alarms	EngA			➧□➧
Hyst	Hysteresis for the process alarms (absolute or deviation)	EngA			➧□➧
Deviatn	Calculated setpoint deviation	EngA			□➧ 📖
Disable	Alarm disables	(A)BCD hex			➧□➧
Hardware	If set to TRUE, disables the hardware alarm	T/F	1	D	
Bad_Cal	If set to TRUE, disables the bad calibration alarm	T/F	2	D	
HiHi	If set to TRUE, disables the high high alarm	T/F	4	D	
Hi	If set to TRUE, disables the high alarm	T/F	8	D	
Lo	If set to TRUE, disables the low alarm	T/F	1	C	
LoLo	If set to TRUE, disables the low low alarm	T/F	2	C	
Input	If set to TRUE, disables the input alarm	T/F	4	C	
PV_Held	If set to TRUE, disables the PV_Held alarm	T/F	8	C	
CalWarng	If set to TRUE, disables the calibration warning alarm	T/F	1	B	
All	If set to TRUE, disables all the above alarms	T/F	2	B	
			4	B	
			8	B	
HiHiType	High High alarm type (Absolute / Deviation)	Menu			
HiHi	High High alarm threshold	EngA			➧□➧
HiHi_OnT	High High alarm on-delay time in seconds	Secs			➧□➧
HiType	High alarm type (Absolute / Deviation)	Menu			
Hi	High alarm threshold	EngA			➧□➧
Hi_OnT	High alarm on-delay time in seconds	Secs			➧□➧
LoType	Low alarm type (Absolute / Deviation)	Menu			
Lo	Low alarm threshold	EngA			➧□➧
Lo_OnT	Low alarm on-delay time in seconds	Secs			➧□➧
LoLoType	Low Low alarm type (Absolute / Deviation)	Menu			
LoLo	Low Low alarm threshold	EngA			➧□➧
LoLo_OnT	Low Low alarm on-delay time in seconds	Secs			➧□➧
In_OnT	Input alarm on-delay time in seconds	Secs			➧□➧
iPV	Input PV, normally wired from the associated AI_UIO block	EngA			➧□➧
iStatus	Input PV status, normally bus wired from the associated AI_UIO block	ABCD hex			➧□➧

Table 3-1: Block parameters

Parameter	Function	Units			Status
iStatusM	Input PV status mask, with bit names matching *iStatus*	ABCD hex			➧▭➧
iAlarms	Input PV alarms, normally wired from the associated AI_UIO block *Alarms* field	(AB)CD hex			➧▭➧
OutRange	I/O block Out of Range alarm In Alarm status	T/F	1	D	
PVError	I/O block PV Error alarm In Alarm status	T/F	2	D	
OCctDel	I/O block Open Circuit Delay In Alarm status	T/F	4	D	
CharErr	I/O block Characterisation Error alarm In Alarm status	T/F	8	D	
NotAuto	I/O block Not Automatic alarm In Alarm status	T/F	1	C	
ModBlock	I/O block Module Block alarm In Alarm status	T/F	2	C	
			4	C	
			8	C	
iAlarmsM	Input PV alarms mask, with bit names matching *iAlarms*.	(AB)CD hex			➧▭➧
iOptions	Provides options for the processing of the input PV and associated alarms	(AB)CD hex			➧▭➧
SetHard1	TRUE sets the Hardware alarm	T/F	1	D	
SetHard2		T/F	2	D	
SetHard3		T/F	4	D	
SetHard4		T/F	8	D	
HoldPV_1	A TRUE on any *HoldPV_n* bit indicates that an external calibration related to the Input PV, or some other action that invalidates the PV is in progress and will cause *PV_Mode* to be “Hold”	T/F	1	C	
HoldPV_2		T/F	2	C	
HoldPV_3		T/F	4	C	
HoldPV_4		T/F	8	C	
CalState	Calibration state (OK/Warning/Bad/Checking/Adjusting)	Menu			📖
CalComnd	Commands to set the calibration state to control the calibration process	(AB)CD hex			➧▭➧
StartAdj	Start adjusting	T/F	1	D	
StartChk	Start checking	T/F	2	D	
SetOk	Set state Ok	T/F	4	D	
SetBad	Set state Bad	T/F	8	D	
Restore	Restore factory calibration	T/F	1	C	
Confirm	User confirmation for “2 point” or “3 point” calibration adjustment	T/F	2	C	
Abort	Used to cancel/abort the calibration adjustment process	T/F	4	C	
AuditChk	Used by the HMI to trigger the logging of an audit check	T/F	8	C	
Cal_PV	Calibration process variable	EngA			📖
Cal_AI	Calibration Analogue Input, typically the electrical input	EngB			➧▭
AdjType	Calibration adjustment type (Disabled/User/2 Point/3 Point)	Menu			
AdjMssge	Adjustment message to guide the user through the adjustment process				📖
AdjValue	Adjustment value, entered by the user during calibration	EngA			
ExtEnabl	External adjustment value enable				➧▭➧
ExtValue	External adjustment value				➧▭
a	Calibration slope				
b	Calibration offset				
CalDate	Calibration date	DD/MM/YY			
Days	Calibration warning interval				
DaysLeft	Calibration warning days left count down				📖
iPV_Lo	Calibration input reading low value	EngA			📖
Val_Lo	Calibration source low value	EngA			📖
iPV_MdUp	Calibration input reading mid value (going up, low to mid)	EngA			📖

Table 3-1: Block parameters

Parameter	Function	Units	Status
Val_MdUp	Calibration source mid value (going up, low to mid)	EngA	📖
iPV_Hi	Calibration input reading high value	EngA	📖
Val_Hi	Calibration source high value	EngA	📖
iPV_MdDn	Calibration input reading mid value (going down, high to mid)	EngA	📖
Val_MdDn	Calibration source mid value (going down, high to mid)	EngA	📖

Table 3-1: Block parameters

Note: * Bit-fields marked with an asterisk share the same field names as *ThisAlm*.

Block specification menu

Dbase, Block, Type. See section 1.3 of this document (Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Method (Template/Block/Native). The location of the function block’s update routine (reserved for future use).

PV_Mode. The PVs update source. When indicating **LiveData**, *PV* is updating from the field *iPV* subject to any calibration that has been applied. When indicating **Hold**, *PV* is not updating and maintains its last good value. **Hold** is set when the calibration state (*CalState*) is **Checking** or **Adjusting** or any of the input options are asserted to hold *PV* (any of *iOptions.HoldPV_1* to *iOptions.HoldPV_4* are **TRUE**).

PV. The process variable. When *PV_Mode* indicates **LiveData**, the value updates according to the prevailing calibration as below, otherwise *PV* holds the last good value.

HR. The engineering High Range. Normally wired to the associated AI_UIO block type.

LR. The engineering Low Range. Normally wired to the associated AI_UIO block type.

ChainIn. Chain Input. This is used to cascade EMS blocks together to form overall sets of resulting chained status bits for driving alarm beacons, pagers and other types of grouped logic outputs (eg for EMS rooms). This input is normally wired from previous EMS block’s *ChainOut* field through subfield bussed wiring (one wire for all bits).

- **Init**. An initialise input, this is ORed with *Options.Init* internally and used as per *Options.Init*.
- **CritAlm**. Chained Input critical Alarm status. Used in the derivation of *ChainOut.CritAlm*.
- **WarnAlm**. Chained Input Warning Alarm status. Used in the derivation of *ChainOut.WarnAlm*.
- **NewAlmA, NewAlmB, NewAlmC**. Chained Input New Alarm status. Used in the derivation of *ChainOut.NewAlmA*, *ChainOut.NewAlmB,* and *ChainOut.NewAlmC.*
- **RsetNewA, RsetNewB, RsetNewC**. Chained Input Reset New Alarm status. Used to reset *ChainOut.NewAlmA/B/C*. This is level dependant and so is usually pulsed TRUE to cause a reset.

ChainOut. Chain Output. This is used to cascade EMS blocks together to form overall sets of resulting chained status bits for driving alarm beacons, pagers, and other types of grouped logic outputs (eg for EMS rooms). This is normally wired to the next VLV3WAY block’s *ChainIn* field through subfield bussed wiring (one wire for all bits). For the last block in the chain, this is normally wired to real digital outputs or other logic.

- **Init**. A copy of chain input init (*ChainIn.Init*).
- **CritAlm**. Chain Output Critical Alarm status. This is TRUE if this block’s critical alarm status (*Status.CritAlm*) or the chained input critical alarm status (*ChainIn.CritAlm*) is TRUE.
- **WarnAlm**. Chain Output Warning Alarm status. This is TRUE if this block’s warning alarm status (*Status.WarnAlm*) or the chained input warning alarm status (*ChainIn.WarnAlm*) is TRUE.
- **NewAlmA, NewAlmB, NewAlmC**. Chained Output New Alarm status. This bit is latched TRUE when either *ChainIn.NewAlmA/B/C* or *Status.NewAlarmA/B/C* become TRUE. This latched bit can be held reset (FALSE) by setting *ChainIn.ResetNewA/B/C* TRUE.

- **RsetNewA, RsetNewB, RsetNewC**. Chained Output Reset New Alarm status. A copy of *ChainIn.ResetNewA/B/C*.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail**. Evaluation failure. Raised if there is an internal evaluation error in the function block update method.
- **Hardware**. Raised if any input options bit *iOptions.SetHard1* to *iOptions.SetHard4* is TRUE or any bit within *iStatus* or *iAlarms* is TRUE and the corresponding include mask bit is TRUE in *iStatusM* or *iAlarmsM* respectively.
- **Bad_Cal**. Bad Calibration. Raised if the user determines there is doubt with the calibration (eg a replacement transmitter has been installed) and opts to 'set bad' calibration. See "Calibration" on page 33 for information on the Calibration State Machine.
- **HiHi.** The High High process alarm. Raised according to the High High alarm type (*HiHiType*) of absolute or deviation when the input is compared to limit *HiHi* with configurable on-delay time (*HiHi_OnT*). The alarm is raised as soon as the limit is breached, but only clears when the alarm value recovers by at least the hysteresis margin (*Hyst*). For absolute alarms, the input is *PV*; for deviation alarms, the input is the calculated deviation (*Deviatn*).
- **Hi.** The High process alarm. See *HiHi* description.
- **Lo.** The Low process alarm. See *HiHi* description.
- **LoLo.** The Low Low process alarm. See *HiHi* description.
- **Input.** Follows the state of *Options.SetInAlm* subject to configurable on-delay time (*In_OnT*).
- **PV_Held.** Raised when *PV_Mode* is **Hold**.
- **CalWarng.** Calibration Warning alarm. Raised when calibration due days (*Days*) reaches zero to flag a calibration is due. This does not necessarily mean the calibration is bad, but merely that a calibration check is due. Note that if the real time clock can't be read or the calibration date (*CalDate*) is set to the invalid date setting (??/??/??), then *DaysLeft* will be zero and the alarm will be raised.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

ThisAlm. This Alarm State. A 16-bit subfield with bit names reflecting the state of the like-named alarms as listed below. These are used in the derivation of critical alarm, warning alarm and new alarm flags. If used, these bits are usually wired from the block's own *Alarms* field (this then takes account of alarm suppression).

- **Hardware**. Hardware "In Alarm" status after alarm suppression.
- **Bad_Cal**. Bad Calibration "In Alarm" status after alarm suppression.
- **HiHi**. High High "In Alarm" status after alarm suppression.
- **Hi**. High "In Alarm" status after alarm suppression.
- **Lo**. Low "In Alarm" status after alarm suppression.
- **LoLo**. Low Low "In Alarm" status after alarm suppression.
- **Input**. Input "In Alarm" status after alarm suppression.
- **PV_Held**. PV Held "In Alarm" status after alarm suppression.
- **CalWarng**. Calibration Warning "In Alarm" status after alarm suppression.

Note: **Software**, **EvalFail** and **Combined** alarms are not included in this list. If Software or EvalFail alarms are raised then the behaviour of these bits and the associated functionality cannot be defined so their inclusion is not relevant.

CritAlmM. Critical Alarm Mask. This has one subfield for each alarm that can be included into the critical alarm status (*Status.CritAlm*). If the mask bit is TRUE and the alarm is raised (as flagged in *ThisAlm*) the critical alarm status is set TRUE.

WarnAlmM. Warning Alarm Mask. As Critical Alarm Mask (*CritAlmM*), but for the Warning alarm status (*Status.WarnAlm*).

NewAlmMA, NewAlmMB, NewAlmMC. New Alarm Mask A/B/C. As Critical Alarm Mask (*CritAlmM*), but for the New Alarm A/B/C status (*Status.NewAlmA/B/C*). Note, however, that the New Alarm Status is based upon a change from FALSE to TRUE in This Alarm status, and so pulses TRUE as new alarms occur.

Note: Normally only one alarm enabled by the mask changes FALSE to TRUE in any database (task) scan, and so there is a pulse for each FALSE to TRUE new alarm transition and its width is set by the database execution time. If new alarms included in the mask occur on consecutive scans of the database, then the bit will remain TRUE for more than one database scan. Similarly, if two new alarms occur on the same database scan, then only a single New Alarm pulse is generated (ie, output pulses do not represent a “new alarm count”).

Options. This sets options for the block.

- **Init**. When TRUE, this sets internal flags used in the derivation of the time delay alarms. If used, it is intended to be wired from the database header block temporary power fail flag (*Header.Status.TmpPFail*) and causes alarm states to be re-evaluated and alarm timers to be re-loaded after a power fail.
- **SetInAlm**. When TRUE, this sets the *Input* alarm subject to the on-delay timer.

Status. This 16-bit subfield shows the block’s status.

- **CritAlm**. Critical Alarm Status. See *CritAlmM*.
- **WarnAlm**. Warning Alarm status. See *WarnAlmM*.
- **NewAlmA, NewAlmB, NewAlmC**. New Alarm status A/B/C. See *NewAlmMA*.
- **LiveData**. Set TRUE when *PV_Mode* is indicating **LiveData**.
- **PV_Held**. Set TRUE when *PV_Mode* is indicating **Hold**.

SetPoint. The Setpoint for process deviation alarms.

Hyst. The hysteresis for the process alarms (absolute or deviation) in engineering units.

Deviatn. The calculated Setpoint deviation. `Deviatn = PV - SetPoint`

Disable. Alarm Disables. One bit per individual alarm (except Software, EvalFail and Combined) and one bit to disable “All”. The alarm is disabled immediately the bit is set TRUE. When set FALSE, the alarm may be delayed if an on-delay time is set to a non-zero value.

- **Hardware**. If set TRUE, disables the Hardware alarm.
- **Bad_Cal**. If set TRUE, disables the Bad Calibration alarm.
- **HiHi**. If set TRUE, disables the High High alarm.
- **Hi**. If set TRUE, disables the High alarm.
- **Lo**. If set TRUE, disables the Low alarm.
- **LoLo**. If set TRUE, disables the Low Low alarm.
- **Input**. If set TRUE, disables the Input alarm.
- **PV_Held**. If set TRUE, disables the PV Held alarm.
- **CalWarng**. If set TRUE, disables the Calibration Warning alarm.
- **All**. If set TRUE, disables all of the alarms listed above (the equivalent of setting all bits above TRUE).

iPV. The Input PV. Normally wired from the associated AI_UIO block.

iStatus. The Input PV Status. Normally wired from the associated AI_UIO block. This can be wired as a single bussed connection from the AI_UIO block as subfield names exactly match those of *AI_UIO.Status*.

iStatusM. Input PV Status Mask. Bit names matching *iStatus*. When a mask bit is TRUE the corresponding status bit being set will raise the *Hardware* alarm. This allows selective I/O status bits to be included or excluded from the *Hardware* alarm.

iAlarms. Input PV Alarms. Normally wired from the associated AI_UIO block *Alarms* field with bit names as below.

- **OutRange**. I/O block Out of Range alarm In Alarm status.
- **PVError**. I/O block PV Error alarm In Alarm status.
- **OCctDel**. I/O block Open Circuit Delay alarm In Alarm status.
- **CharErr**. I/O block Characterisation Error alarm In Alarm status.
- **NotAuto**. I/O block Not Automatic alarm In Alarm status.
- **ModBlock**. I/O block Module Block alarm In Alarm status.

iAlarmsM. Input PV Alarms Mask. A 16-bit subfield with bit names matching *iAlarms*. When a mask bit is TRUE, the corresponding input alarm bit being set will raise the *Hardware* alarm. This allows selective I/O alarms to be included or excluded from the *Hardware* alarm.

iOptions. This 16-bit subfield provides options for the processing of the input PV and associated alarms.

- **SetHard1, SetHard2, SetHard3, SetHard4**. TRUE sets the Hardware alarm. If *SetHard1, SetHard2*, *SetHard3* and *SetHard4* are all FALSE, *iStatus*, *iStatusM, iAlarms*, and *iAlarmM* determine the *Hardware* alarm state. These could be used with I/O being gathered by other means (for example, communications) rather than AI_UIO blocks to trigger a hardware alarm if the input fails in some way.
- **HoldPV_1, Hold_PV2, Hold_PV3, Hold_PV4**. A TRUE on any *HoldPV_n* bit indicates that an external calibration related to the Input PV, or some other action that invalidates the PV is in progress and will cause the *PV_Mode* to be **Hold**. This might be used if two real inputs are being used to form a differential pressure input and one or both of them are being calibrated.

CalState (Ok/Warning/Bad/Checking/Adjusting). This enumeration defines the calibration state of the function block. See “Calibration” on page 33 for details of the calibration state machines.

- **Ok**. The last calibration check performed resulted in the user defining the calibration to be “**OK**” and no subsequent expiry of the Calibration Warning timer has occurred.
- **Warning**. The calibration state has been set to **OK**, but the Calibration Warning timer has expired and the *CalWarng* alarm has been raised. This state is driven from the “In Alarm” status and so the *CalWarng* alarm must be enabled to get a warning indication. This ensures only users who want a warning alarm will receive the warning state.
- **Bad**. The user has chosen to set the calibration state to **Bad** in order to flag that equipment should not be used until a satisfactory calibration check is performed following a transmitter change, for example.
- **Checking**. The user has started a routine calibration check or has completed a calibration adjustment and is now checking the calibration before choosing to set the stater to **Ok** or **Bad**.
- **Adjusting**. The user has started an adjustment. The adjustment process follows the specified adjustment type, *AdjType* setting, and automatically moves the user to **Checking** upon completion.

CalComnd. Provides commands to set the calibration state of the block and so control the calibration process. Commands are set TRUE by the user, processed if relevant at this time, and cleared FALSE automatically by the function block. Only one command should be set TRUE at any time; if more than one command it set TRUE, then all commands are cleared.

- **StartAdj**. Start Adjusting.
- **StartChk**. Start Checking.
- **SetOk**. Set State Ok.
- **SetBad**. Set State Bad.
- **Restore**. Restore factory calibration *a*=1.0, *b*=0.0 and clear adjustment settings (*iPV_** and *iVal_** set to 0.0).
- **Confirm**. Used to advance the “2 Point” or “3 Point” calibration adjustment process as each step is acknowledged as complete by the user.

- **Abort**. Used to cancel/abort the calibration adjustment process.
- **AuditChk**. Used by the HMI to trigger the logging of an audit check message, typically as part of the calibration checking process. This is not used internally by the function block, but is included to provide a complete 21CFR part 11 audit trail against this block's tagname.

Cal_PV. Calibration Process Variable. This PV updates from *iPV* and the calibration constants (*a* and *b*) during calibration checking to assist with the checking process (the regular *PV* is held during calibration checking). In the Adjusting state, this is forced to -999999 to show it can't be updated while the calibration constants are being changed.

Cal_AI. Calibration Analogue Input. Usually wired from the AI_UIO block *AI* parameter to show the electrical input reading for the sensor during calibration.

AdjType (Disabled/User/2 Point/3 Point). Defines the calibration Adjustment Type to be used.

- **Disabled**. The calibration adjustment state is disabled and commands to start adjustment are ignored. This is typically used when calibration is performed external to this block, for example, on differential pressure points when each input is calibrated separately or when the input is read over comms and the input device incorporates its own calibration process.
- **User**. The calibration adjustment process is user define and the user manually determines the required calibration factors for slope (*a*) and offset (*b*). This might be used if more than 3 points are required or an alternative algorithm is used to calculate the slope and offset.
- **2 Point**. The calibration adjustment process is based around a user applying two input points; one for a "low" and then one for a "high" input reading, and entering the corresponding actual values from the calibration equipment.
- **3 Point**. As for the 2 Point adjustment, but with three nominal points. The additional centre (mid) point reading is applied with the mid value entered twice, once going up (Mid Up) from a "low" to a "high", and once from a "high" to a "low" (Mid Down) to take account of any hysteresis in the measurement system

AdjMssge. Adjustment Message. A string message to guide the user through the adjustment process (it is expected that the adjustment message would be expanded and internationalised at any user HMI). When the Calibration State (*CalState*) is not **Adjusting** or the Adjustment Type (*AdjType*) is **User**, this message is blank.

AdjValue. Adjustment Value. When prompted (*AdjMssge*), the user enters the adjustment value from the calibration source as part of the calibration adjustment process. An external value (*ExtValue*) can be applied rather than a manual input if enabled (*ExtEnabl*). The value is forced to -999999 when not relevant or awaiting input.

ExtEnabl. External Adjustment Value Enable. When TRUE, the external calibration source value (*ExtValue*) is enabled to update the adjustment value (*AdjValue*) when a value is required.

ExtValue. External Adjustment Value. If enabled (*ExtEnabl* is TRUE), an external calibration source can be read via a comms link to update this value and use it as the adjustment value (*AdjValue*) when prompted, rather than manually entering the value. It is important that the accuracy of this input is considered as if wired to real I/O, it becomes subject of calibration itself, hence, the use of a value read on comms should be considered.

CalDate. Calibration Date. A manual entry of the date that the calibration process is deemed to have been completed. This is not automatically updated as part of the adjusting state because it may need linking to other external documentation processes, for example.

Days. Calibration Warning Interval. This user entry defines how many days after the Calibration Date (*CalDate*) a calibration warning alarm is raised.

DaysLeft. Calibration Warning Days Left Count Down. This value counts down as the days pass since the Calibration Date (*CalDate*) for *Days* number of days. When zero is reached, the calibration warning alarm (*CalWarng*) is raised. Note that no calibration time is stored, and so timing runs from 00:01 of the calibration date.

iPV_Lo. Calibration Input Reading Low value.

iPV_Hi. Calibration Input Reading High value.

Val_Lo. Calibration Source Low value.

Val_Hi. Calibration Source High value.

Note: The above *iPV_Lo*, *iPV_Hi*, *Val_Lo*, and *Val_Hi* fields apply to both **2 Point** and **3 Point** adjustment types; they are forced to zero for the **User** adjustment type.

iPV_MdUp. Calibration Input Reading Mid value (going up - low to high).

Val_MdUp. Calibration Source Mid value (going up - low to high).

iPV_MdDn. Calibration Input Reading Mid value (going down - high to low).

Val_MdDn. Calibration Source Mid value (going down - high to low).

Note The above *iPV_MdUp*,*Val_MdUp*, *iPV_MdDn*, and *Val_MdDn* fields apply only to the **3 Point** adjustment type; they are forced to zero for the **2 Point** and **User** adjustment types.

Calibration

The VLV3WAY function block provides services to facilitate calibration checking and adjustment. Throughout this section, the term "calibration checking" refers to the process of verifying the calibration, and "calibration adjustment" refers to the process of adjusting the calibration constants to comply with calibration standards.

CALIBRATION STATE MACHINE

In a typical system, the function block calibration state machine normally resides in the **OK** state. The state changes automatically to **Warning** when a calibration warning is flagged (if the *CalWarng* alarm is enabled).

The **Adjusting**, **Checking** and **Bad** states are all selected by the user as part of the calibration process. The user interacts with preset commands from the subfield bit list of the *CalComnd* field to drive the calibration process. The following figure shows an overview of the calibration state machine (the *CalComnd* subfield name is omitted from the figure for clarity).

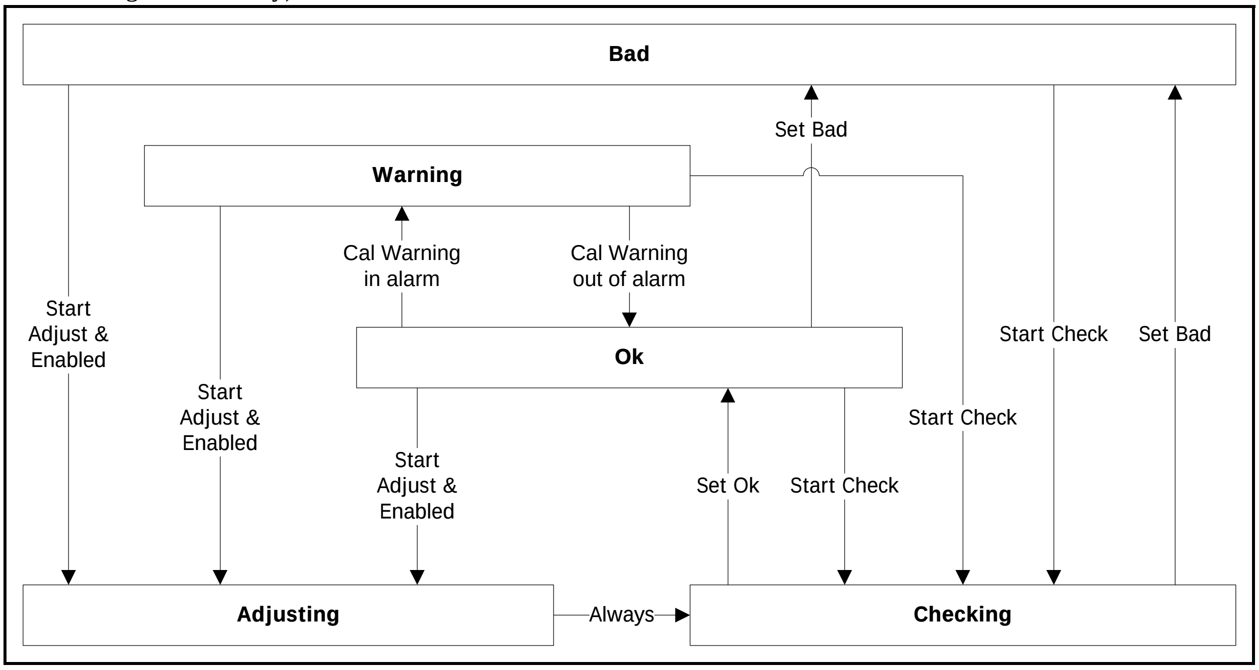


Figure 3-1: Calibration State Machine

When in **Adjusting** or **Checking**, the value of *PV* is not updated and the *PV_Mode* indicates **Hold** (the actual input would be disconnected from the real process during these states). For any other state (**Ok**, **Warning**, and **Bad**), the *PV_Mode* indicates **LiveData** (the actual input would be connected to the real process during these states). This allows use of the *PV_Mode* field to remove data from long-term data processing functions at the HMI (mean kinetic temperature, and averages, for example), when *PV_Mode* is not set to **LiveData**.

When the state is **Bad**, *Alarms.Bad_Cal* is continuously raised. This alarm only clears under user control on the transition from **Checking** to **Ok** (**Set Ok** command).

When the state is **Checking**, no specific calibration actions are undertaken by the function block. Instead, the **Checking** state can be reported to the HMI for user information whilst the user performs their site-specific calibration checks. The reading of calibration PV (*Cal_PV*) and calibration input (*Cal_AI*) are useful in this state because the main block *PV* is held. In this checking mode, the user can make as many checks as they like, logging some or all points to the HMI audit trail as desired.

When an adjustment is enabled (*AdjType* is not **Disabled**), and a Start adjustment command is issued (**StartAdj**) has been requested, the adjustment is handled by the state machine (refer to the next section). No matter how the adjustment process completes (normal end, or an abort), the state moves automatically to **Checking** and the user must follow their checking procedures and then make the choice to "Set **Bad**", or "Set **Ok**".

CALIBRATION ADJUSTING STATE MACHINE

Calibration adjustment is usually the least used of the calibration states because most calibration checks pass with the result **Ok** and adjustment is not required. The calibration adjustment process is used to explicitly flag that changes are planned and in-progress to the calibration constants (*a* and *b*). Calibration adjustment cannot be started if adjustment is disabled (*AdjType* is **Disabled**).

During calibration adjustment, the calibration PV (*Cal_PV*) is forced to -999999 to indicate it is invalid (it cannot be sensibly updated when the calibration contents are being changed).

The following state transition figure shows an outline of the calibration adjustment procedure.

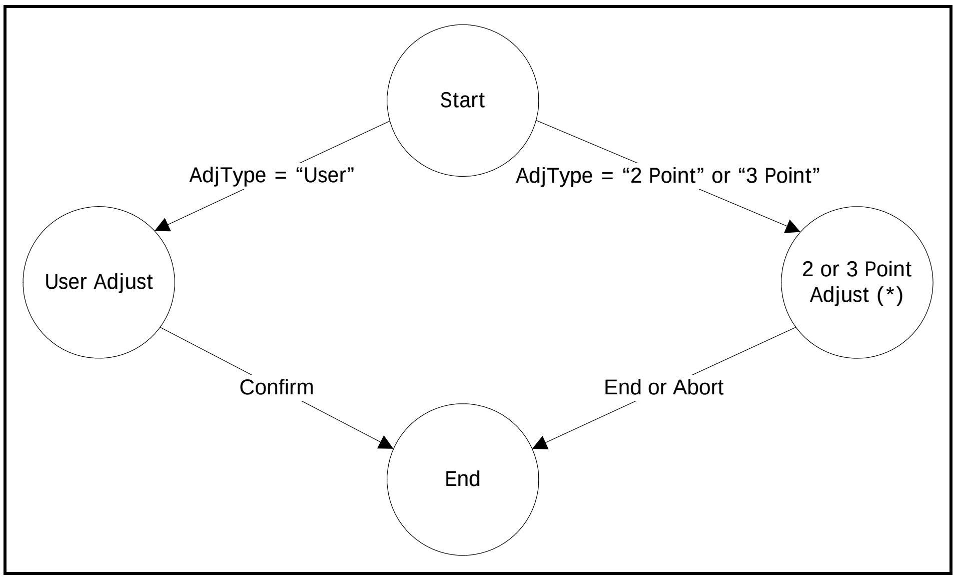
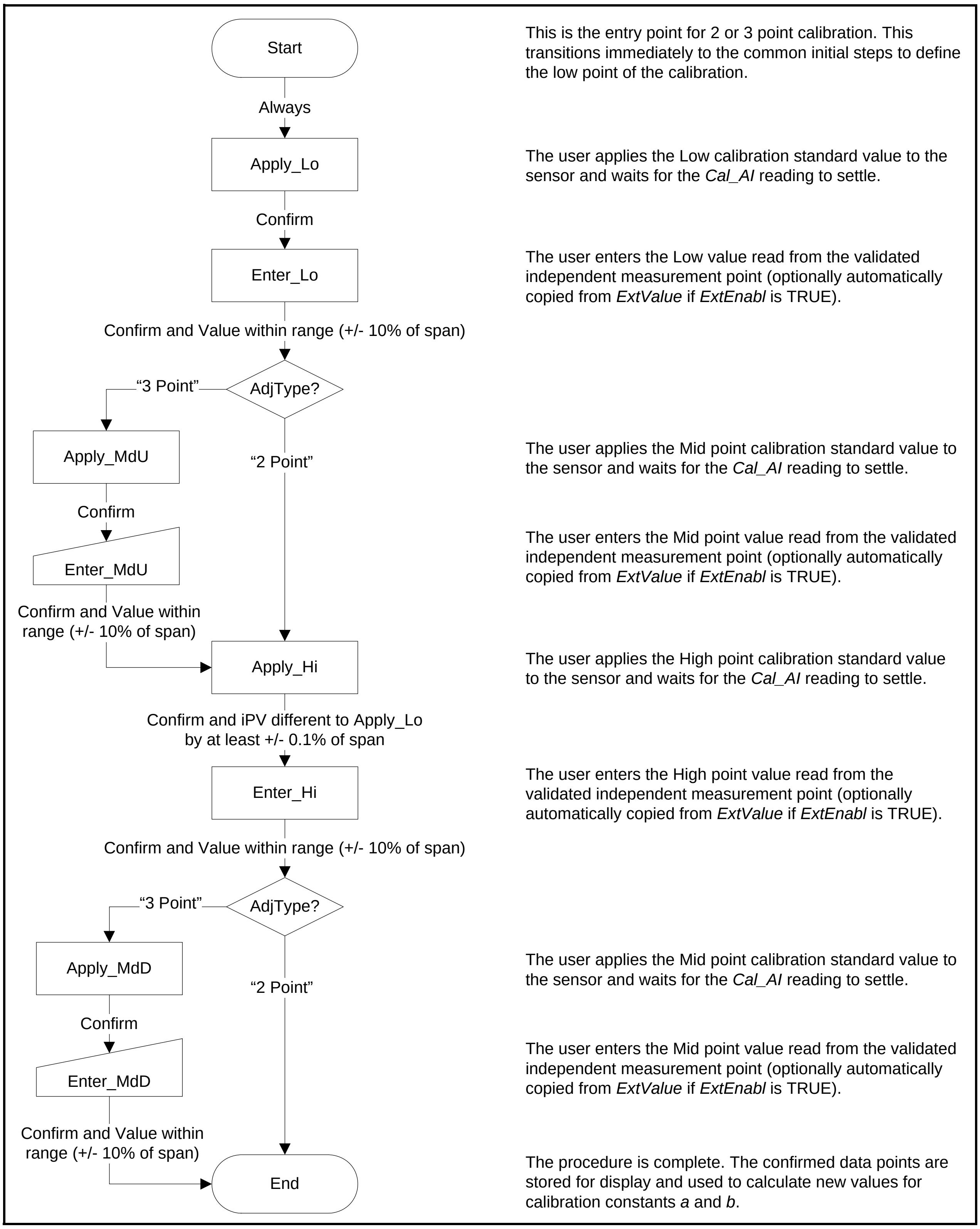


Figure 3-2: Calibration Adjustment State Machine Overview

The following figure shows the Adjustment State Machine, which is implemented when AdjType is **2 Point** or **3 Point** (* in the previous figure).


Start
Always
Apply_Lo
Confirm
Enter_Lo
Confirm and Value within range (+/- 10% of span)
AdjType?
"3 Point"
"2 Point"
Apply_MdU
Confirm
Enter_MdU
Confirm and Value within range (+/- 10% of span)
Apply_Hi
Confirm and iPV different to Apply_Lo by at least +/- 0.1% of span
Enter_Hi
Confirm and Value within range (+/- 10% of span)
AdjType?
"3 Point"
"2 Point"
Apply_MdD
Confirm
Enter_MdD
Confirm and Value within range (+/- 10% of span)
End

This is the entry point for 2 or 3 point calibration. This transitions immediately to the common initial steps to define the low point of the calibration.

The user applies the Low calibration standard value to the sensor and waits for the *Cal_AI* reading to settle.

The user enters the Low value read from the validated independent measurement point (optionally automatically copied from *ExtValue* if *ExtEnabl* is TRUE).

The user applies the Mid point calibration standard value to the sensor and waits for the *Cal_AI* reading to settle.

The user enters the Mid point value read from the validated independent measurement point (optionally automatically copied from *ExtValue* if *ExtEnabl* is TRUE).

The user applies the High point calibration standard value to the sensor and waits for the *Cal_AI* reading to settle.

The user enters the High point value read from the validated independent measurement point (optionally automatically copied from *ExtValue* if *ExtEnabl* is TRUE).

The user applies the Mid point calibration standard value to the sensor and waits for the *Cal_AI* reading to settle.

The user enters the Mid point value read from the validated independent measurement point (optionally automatically copied from *ExtValue* if *ExtEnabl* is TRUE).

The procedure is complete. The confirmed data points are stored for display and used to calculate new values for calibration constants *a* and *b*.

Figure 3-3: Adjustment State Machine

The calculation of the calibration constants depend upon the adjustment type of **2 Point** or **3 Point**. The block's *PV* is related to the input PV (*iPV*) and the calibration constants (*a* and *b*) by definition by the following equation.

$PV = a * iPV + b$

CALCULATION OF CALIBRATION CONSTANTS — 2 POINT ADJUSTMENT

At the end of the two point adjustment process, there is a low point-pair and a high point-pair of numbers. Each point-pair consists of one actual input value reading (*iPV*) and one expected value (*AdjValue*) from the moment the entered point data was confirmed.

The low and high point-pair are ideally selected to reflect the actual range that the sensor will be used over (that is, positioned about the deviation alarm setpoint) and this would normally be inside the overall range of the sensor. The following figure shows *iPV* plotted against *AdjValue*.

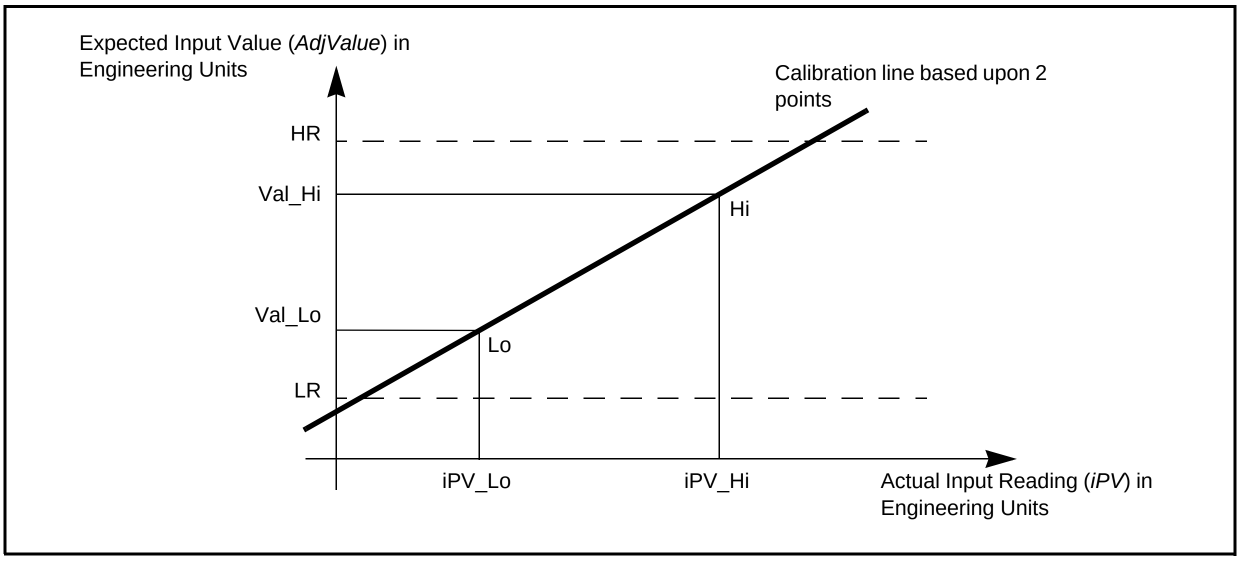


Figure 3-4: 2 Point Adjustment Graph: *iPV* plotted against *AdjValue*

From the above graph, the gradient (*a*), or slope, is given by:

$$\boldsymbol{a = \frac{(Val\_Hi - Val\_Lo)}{(iPV\_Hi - iPV\_Lo)}}$$

Substitution of a point-pair (for example, the low point-pair as used within the block) into the main equation for PV and rearranging yields the offset (*b*):

$$\boldsymbol{b = Val\_Lo - a * iPV\_Lo}$$

CALCULATION OF CALIBRATION CONSTANTS — 3 POINT ADJUSTMENT

At the end of a 3 point adjustment, there are a total of four pair-points recorded: a low point-pair, a high point-pair, and two mid point-pairs. These four point-pairs need to be processed to form a straight line fit and hence derive the calibration constants (*a* and *b*) from the calibration data. The following example shows the median-fit method used by the block.

The median-fit is calculated as follows:

1. Derive the constants *a* and *b* for the line passing through the low and high point-pair exactly as for the two point case.
2. Form the average of the two mid point values (reducing them to a one point-pair).
3. Calculate the difference between the actual average mid point value found in step 2 and the predicted mid point value using the equation found in step 1.
4. Adjust the offset (*b*) to slide the “best fit line” on third of the way towards the average mid point value found in step 2.

The following figure shows the three *iPV*s plotted against the *AdjValue*s.

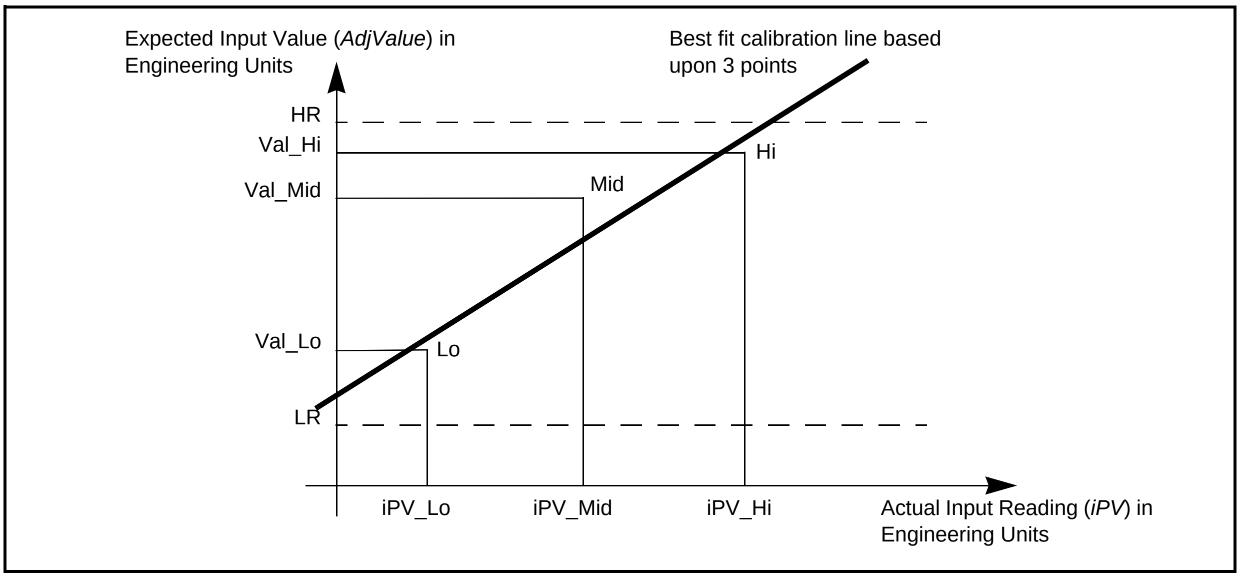


Figure 3-5: 3 Point Adjustment Graph: *iPV* plotted against *AdjValue*

The point **Mid** on the graph is the average of Mid Up and Mid Down:

$$\mathit{Val\_Mid} = \frac{\mathit{Val\_MdUp} + \mathit{Val\_MdDn}}{2} \quad \text{and} \quad \mathit{iPV\_Mid} = \frac{\mathit{iPV\_MdUp} + \mathit{iPV\_MdDn}}{2}$$

and:

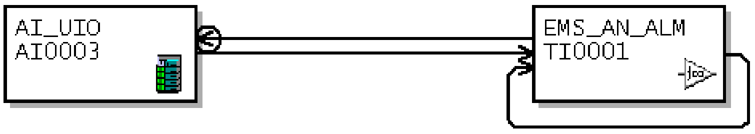
$$a = \frac{\mathit{Val\_Hi} - \mathit{Val\_Lo}}{\mathit{iPV\_Hi} - \mathit{iPV\_Lo}} \quad \text{and} \quad b1 = \mathit{Val\_Lo} - a * \mathit{iPV\_Lo}$$

Finally:

$$b = \frac{b1 + (\mathit{Val\_Mid} - a * \mathit{iPV\_Mid} + b1)}{3}$$

EXAMPLE BLOCK USAGE

When an EMS_AN_ALM block is paired with an AI_UIO block, for a sensor input, the typical connections between the two is as shown below. The basic electrical input characteristics (mA, linearisation, etc) are set in the AI_UIO block as usual for all analogue inputs.



Typical wiring connections are shown in the following two tables. The header block's *Status.TmpPFail* is typically wired to the EMS_AN_ALM block's *Options.Init* field, as shown in the first table. This would ensure the *Options.Init* field is pulsed once at database start to initialise the alarm on-delay timers at start up.

Header Block	Direction	EMS_AN_ALM Block
Status.TmpPFail	--->	Options.Init

Table 3-2: Header block wiring to an EMS_AN_ALM block

The Warning Alarm, Critical Alarm and New Alarm (x3) status bits (and hence the status chaining - *ChainIn* and *ChainOut*) rely upon the current alarm status being wired back from the block into the *ThisAlm* field. This arrangement ensures that any applied alarm suppression is taken into account when these status bits are derived (if an alarm is suppressed, then it will not be able to set the related status bit). The typical wiring is shown in the following table.

EMS_AN_ALM block	Direction	EMS_AN_ALM Block
Alarms.Hardware	--->	ThisAlm.Hardware
Alarms.Bad_Cal	--->	ThisAlm.Bad_Cal
Alarms.HiHi	--->	ThisAlm.HiHi
Alarms.Hi	--->	ThisAlm.Hi
Alarms.Lo	--->	ThisAlm.Lo
Alarms.LoLo	--->	ThisAlm.LoLo
Alarms.Input	--->	ThisAlm.Input
Alarms.PV_Held	--->	ThisAlm.PV_Held
Alarms.CalWarng	--->	ThisAlm.CalWarng

Table 3-3: Typical wiring for status bits and alarm suppression

Typical wiring between the AI_UIO block and the EMS_AN_ALM block is shown in the following table. There are connections in both directions between the two blocks which should both be assigned to the same task (normally task 3, in a T2550 instrument).

AI_UIO Block	Direction	EMS_AN_ALM Block
PV	--->	iPV
AI	--->	CalAI
Status	--->	iStatus
Alarms.OutRange	--->	iAlarms.OutRange
Alarms.PVError	--->	iAlarms.PVError
Alarms.OcctDel	--->	iAlarms.OcctDel
Alarms.NotAuto	--->	iAlarms.NotAuto
Alarms.CharErr	--->	iAlarms.CharErr
Alarms.ModBlock	--->	iAlarms.ModBlock
HR	<---	HR
LR	<---	LR

Table 3-4: AI_UIO block connection to/from an EMS_AN_ALM block

This arrangement of wires allows the EMS_AN_ALM block to act as the main interface block for the sensor point. Hardware-related alarms and status are copied from the AI_UIO block, and ranging data is passed from the EMS_AN_ALM block to the AI_UIO block, so there is often no requirement to directly visualise the AI_UIO block to the operator HMI. The AI_UIO block would typically only be used for diagnostics via LINtools.

CHAPTER 4   CONTROL APPLICATION BLOCKS

The CONTROL category of Application Function Block Templates provides the control strategy with functions for controlling the control loop operations, PID, load simulation, etc. of an instrument.

Intentionally left blank

VLV3WAY: COMBUSTION CONTROL BLOCK

Block function

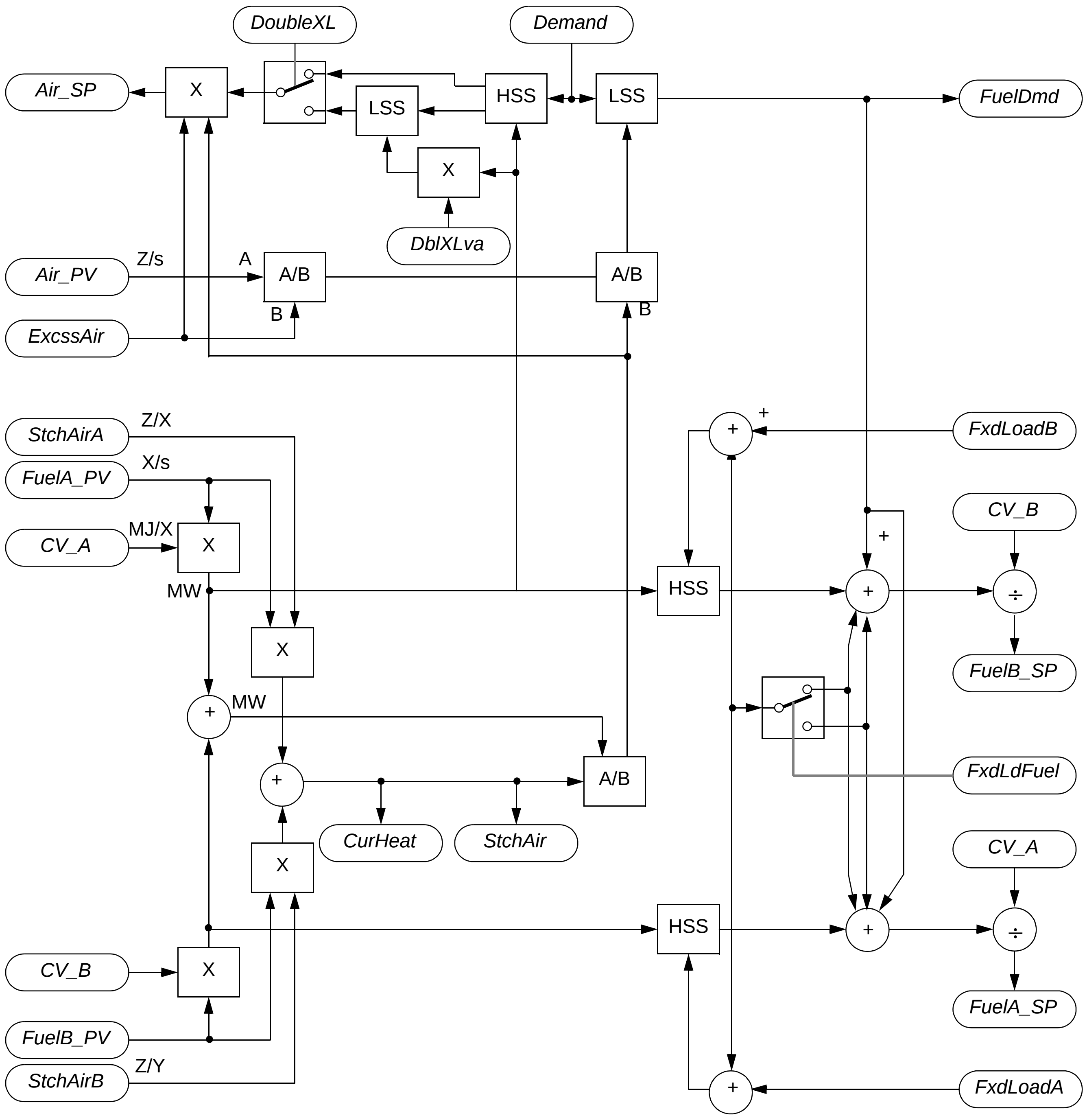


Figure 4-1: Block schematic

This block uses a heat demand to generate remote setpoints, for up to two fuel flow controllers and a combustion air controller, based on a cross-limiting combustion algorithm.

In manual mode, the heat demand may be set directly by an operator. In automatic mode, the heat demand may be set by a profiling sequence or a master controller.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units		Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum		📖
Mode	Current operating mode	Enum		
ModeAct	Mode active	CD hex		□➔ 📖
ManAct	Manual mode active	T/F	1 D	
AutoAct	Auto mode active	T/F	2	
			4	
			8	
FuelAStt	Current state of fuel A	Enum		➔□➔
FuelA_PV	Fuel A flow process variable	Eng		➔□➔
FuelBStt	Current state of fuel B	Enum		➔□➔
FuelB_PV	Fuel B flow process variable	Eng		➔□➔
AirStt	Current state of the combustion air	Enum		➔□➔ &
Air_PV	Combustion air flow process variable	Eng		➔□➔ &
AutoDmnd	Automatic demand	Eng		➔□➔
Claims	Resource management claims	CD hex		➔□➔
Usr0	Claim #0	T/F	1 D	
Usr1	Claim #1	T/F	2	
Usr2	Claim #2	T/F	4	
Usr3	Claim #3	T/F	8	
Usr4	Claim #4	T/F	1 C	
Usr5	Claim #5	T/F	2	
Usr6	Claim #6	T/F	4	
Usr7	Claim #7	T/F	8	
Acks	Resource management acknowledgements	CD hex		□➔ 📖
Usr0	Acknowledgement #0	T/F	1 D	
Usr1	Acknowledgement #1	T/F	2	
Usr2	Acknowledgement #2	T/F	4	
Usr3	Acknowledgement #3	T/F	8	
Usr4	Acknowledgement #4	T/F	1 C	
Usr5	Acknowledgement #5	T/F	2	
Usr6	Acknowledgement #6	T/F	4	
Usr7	Acknowledgement #7	T/F	8	
FuelDmd	Total fuel heat demand	Eng		□➔ 📖
FuelA_SP	Fuel flow setpoint for fuel A	Eng		□➔ 📖
FuelB_SP	Fuel flow setpoint for fuel B	Eng		□➔ 📖
Air_SP	Combustion air flow setpoint	Eng		□➔ 📖
CurHeat	Current total heat	Eng		□➔ 📖
StchAir	Calculated stoichiometric air flow	Eng		□➔ 📖

Table 4-1: Block parameters

Parameter	Function	Units	Status
Alarms		ABCD hex	□➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	Hardware alarm	T/F	
Afault	Fuel A fault alarm	T/F	
Bfault	Fuel B fault alarm	T/F	
TooRich	Combustion mixture too rich alarm	T/F	
BthModlt	Both fuels modulating alarm	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Heat demand	Eng	□➧
ExcssAir	Excess air requirement	%	➧□➧
Bias	Fuel load, fuel bias	Eng	➧□➧
DblXLval	Double cross-limiting value	%	➧□➧
ModDmnd	Modulating fuel heat demand	Eng	□➧ 📖
CV_A	Calorific value of fuel A	Eng	➧□➧
StchAirA	Stoichiometric air requirement of fuel A	Eng	➧□➧
FxdLoadA	Fixed load heat demand for fuel A	Eng	➧□➧
CV_B	Calorific value of fuel B	Eng	➧□➧
StchAirB	Stoichiometric air requirement of fuel B	Eng	➧□➧
FxdLoadB	Fixed load heat demand for fuel A	Eng	➧□➧
State	Derived state	Enum	
StateAct	Active state	(A)BCD hex	□➧ 📖
Purge	Purge is in progress	T/F	1 D
Bth>Fire	Both fuels are firing	T/F	2 D
A>StrtB	Fuel A is firing and fuel B is starting	T/F	4 D
B>StrtA	Fuel B is firing and fuel A is starting	T/F	8 D
A>Fire	Only fuel A is firing	T/F	1 C
B>Fire	Only fuel B is firing	T/F	2 C
StrtBth	Both fuel A and fuel B are starting	T/F	4 C
StrtA	Fuel A is starting. Fuel B is stopped	T/F	8 C
StrtB	Fuel B is starting. Fuel A is stopped	T/F	1 B
Idle	Neither fuel is starting or firing	T/F	2 B
Fault	Both fuels are in fault	T/F	4 B
			8 B

Table 4-1: Block parameters

Parameter	Function	Units			Status
Status		(A)BCD hex			
BalSP	Force fuel flow controller SP balance	T/F	1	D	
DampenA	Restrict fuel A's ability to increase	T/F	2		
DampenB	Restrict fuel B's ability to increase	T/F	4		
FxLdSel	Fixed load fuel selection	T/F	8		
Astrting	Fuel A is starting	T/F	1	C	
Afiring	Fuel A is firing	T/F	2		
Afault	Fuel A is indicating fault	T/F	4		
Bstrting	Fuel B is starting	T/F	8		
Bfiring	Fuel B is firing	T/F	1	B	
Bfault	Fuel B is indicating fault	T/F	2		
Bit10	User status #10	T/F	4		
Bit11	User status #11	T/F	8		
Options		(C)D hex			
AuDmdTrk	Automatic demand track	T/F	1	D	
Avail1	User device available #1	T/F	2		
Avail2	User device available #2	T/F	4		
NotAvail1	User device not available #1	T/F	8		
NotAvail2	User device not available #1	T/F	1	C	
FpShowB	Show fuel B on faceplate	T/F	2		
FxLdFuel	Fixed load fuel	T/F	4		
DoPurge	Select device state to purge	T/F	8		
DoubleXL	Double cross-limiting combustion	T/F	1	B	
SelAstop	Select fuel A state to stop	T/F	2		
SelAfire	Select fuel A to fire	T/F	4		
SelAflt	Select fuel A state to fault	T/F	8		
SelBstop	Select fuel B state to stop	T/F	1	A	
SelBfire	Select fuel B to fire	T/F	2		
SelBflt	Select fuel B state to fault	T/F	4		
Bit15	User option #15	T/F	8		
Availble	Available for automatic control	T/F			▭➧ 📖
Master	User parameter	Eng			➧▭➧

Table 4-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Mode. (Manual/Auto). Selects current operating mode.

ModeAct. (ManAct/AutoAct). Indicates active operating mode.

FuelAStt. Current state of fuel A. Input used to determine if the fuel is firing. Used in combination with the current state of fuel B to evaluate the combustion state. May be input connected or controlled by option bits.

FuelBStt. Current state of fuel B. Input used to determine if the fuel is firing. Used in combination with the current state of fuel A to evaluate the combustion state. May be input connected or controlled by option bits.

AirStt. Current state of the combustion air. Currently has no effect on the combustion algorithm.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

Claims. Resource Management Claims. Used to request sole control of the control module. The motor resource may be controlled by up to eight sequences or other control module users.

Acks. Resource Management Acknowledgements. Indicates acceptance of a claim.

FuelDmd. Total fuel heat demand. Calculated from cross-limiting algorithm.

FuelA_SP. Fuel flow setpoint for fuel A. Calculated from cross-limiting algorithm.

FuelB_SP. Fuel flow setpoint for fuel B. Calculated from cross-limiting algorithm.

Air_SP. Combustion air flow setpoint. Calculated from cross-limiting algorithm.

CurHeat. Current total heat. The heat calculated as being currently generated from the current fuel flows.

StchAir. Calculated stoichiometric Air Flow. This is based only on the current fuel flows.

Alarms.

- **Software.** Sumcheck error in block’s RAM data.
- **EvalFail.** Evaluation Failure Alarm. Evaluation failure in block’s internal logic.
- **hardware.** Hardware Alarm. Hardware failure flagged by an input or output block associated with the motor.
- **Afault.** Fuel A Fault Alarm. The state of Fuel A indicates fault.
- **Bfault.** Fuel B Fault Alarm. The state of Fuel B indicates fault.
- **TooRich.** Combustion Mixture Too Rich Alarm. The current air to fuel ratio is lower than that required by the cross-limiting algorithm.
- **BthModlt.** Both Fuels Modulating Alarm. The fixed load fuel has had to start modulating because the modulating fuel could not satisfy the demand.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Heat demand. Maintained demand for the required heat.

ExcssAir. Excess air requirement. Percentage air required in excess of stoichiometric combustion for use in the cross-limiting algorithm.

Bias. Fuel load, fuel bias. The bias applied between the two fuels when the fixed load fuel needs to modulate to satisfy the demand.

DblXLval. Double cross-limiting value. The percentage headroom when double cross-limiting is enabled for the combustion air setpoint to increase above that calculated by the cross-limiting algorithm.

ModDmnd. Modulating fuel heat demand. Required for internal use only. The demand less the current fuel fixed load value.

FxdLoadA. Fixed load heat demand for fuel A. The heat demand required from fuel A when fuel B is the modulating fuel and can satisfy its heat demand.

FxdLoadB. Fixed load heat demand for fuel B. The heat demand required from fuel B when fuel A is the modulating fuel and can satisfy its heat demand.

State. Current state derived from the current state of both fuels.

StateAct. Active state derived from the current state of both fuels. Only one state bit is active at any one time.

- **Purge.** Purge is in progress.
- **Bth>Fire.** Both fuels are firing.
- **A>StrtB.** Fuel A is firing and fuel B is starting.
- **B>StrtA.** Fuel B is firing and fuel A is starting.
- **A>Fire.** Only fuel A is firing.
- **B>Fire.** Only fuel B is firing.
- **StrtBth.** Both fuel A and fuel B are starting.
- **StrtA.** Fuel A is starting. Fuel B is stopped.
- **StrtB.** Fuel B is starting. Fuel A is stopped.
- **Idle.** Neither fuel is starting or firing.
- **Fault.** Both fuels are in fault.

Status. This bitfield shows the status of the control module.

- **BalSP.** Force fuel flow controller SP balance. Pulses high on a demand or fixed load fuel change.
- **DampenA.** Restrict fuel A’s ability to increase. High indicates that the fuel A flow controller should be limited such that increases in fuel flow setpoint are rate limited. This gives a modulating fuel B time to respond to load changes before fuel A moves off its fixed load value.
- **DampenB.** Restrict fuel B’s ability to increase. High indicates that the fuel B flow controller should be limited such that increases in fuel flow setpoint are rate limited. This gives a modulating fuel A time to respond to load changes before fuel B moves off its fixed load value.

Options. This bitfield allows inputs to control the operation of the control module.

- **AuDmdTrk.** Automatic Demand Track. TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **Avail1, Avail2.** Available is allowed. FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** Available is not allowed. TRUE prevents the block from signalling that it is available for automatic control.

Availble. Available. TRUE when control module is available for automatic control, i.e. in automatic mode and availability options not prohibitive.

Master. Optional Parameter. Associates additional floating-point data with the control module.

Implementation notes

If CmbnCtrl control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2882 bytes and by 186 bytes for each instance of the control module.

Note	For this application module please refer to factory.

Intentionally left blank

VLV3WAY: SINGLE FUEL COMBUSTION CONTROL BLOCK

Block function

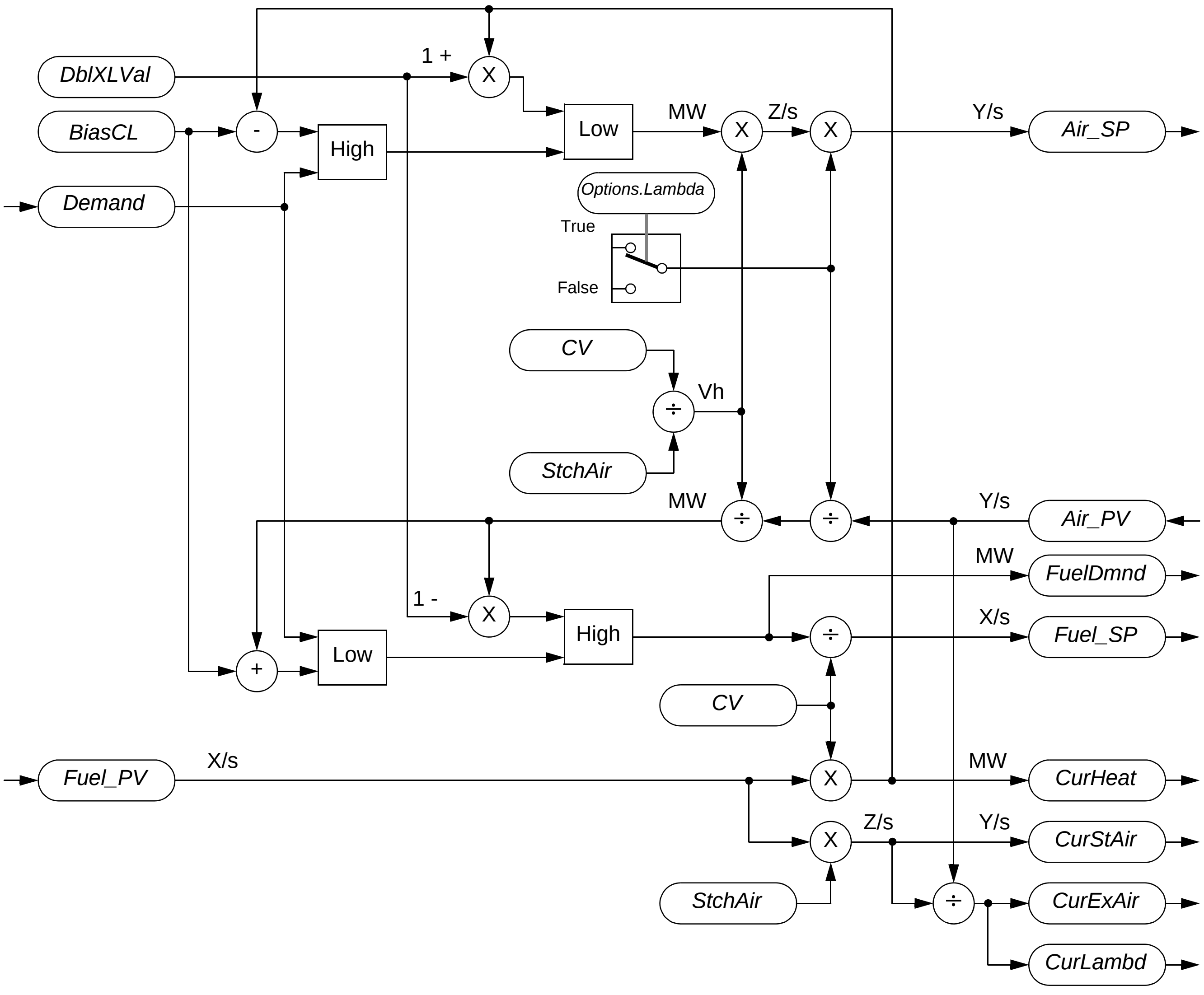


Figure 4-1: Block schematic

This block is an interface to single-fuel burners and is used to generate setpoint values for air flow and fuel flow control modules. Fuel flow and air flow feedback signals are applied to analogue input modules, the outputs of which are wired to the CMBNXLIM block's *Fuel_PV* and *Air_PV* inputs. These inputs are used, together with a Demand input (typically controlled by the LOOP_PID block), to derive fuel flow and air flow setpoint values. These values are wired from the block's *Fuel_SP* and *Air_SP* outputs to the PV inputs of the analogue output modules controlling Fuel flow and Air flow

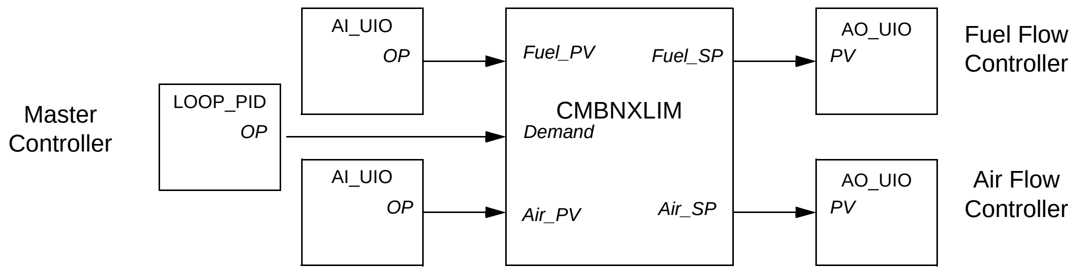


Figure 4-2: Block operation - example

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Demand	Heat demand	MW	➡☐➡
Fuel_PV	Fuel flow process variable	X/s	➡☐➡
Air_PV	Air flow process variable	Y/s	➡☐➡
FuelDmnd	Total fuel heat demand	MW	☐➡ 📖
CurHeat	Current total heat	MW	☐➡ 📖
CurStAir	Current Stoichiometric air flow	Y/s	☐➡ 📖
CurExAir	Current excess air	%	☐➡ 📖
CurLambd	Current Lambda		☐➡ 📖
Alarms		ABCD hex	☐➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Combined	OR-ing of all alarm bits	T/F	
ExcssAir	Excess air requirement	%	➡☐➡
Lambda	Lambda air requirement		➡☐➡
BiasCL	Cross-limiting bias value	Eng	➡☐➡
DblXLval	Double cross-limiting value	%	➡☐➡
CV	Calorific value of fuel	MJ/X	➡☐➡
StchAir	Stoichiometric air requirement for combustion of fuel	%	➡☐➡
Fuel_SP	Fuel flow setpoint for fuel	X/s	☐➡ 📖
Air_SP	Combustion air flow setpoint	Y/s	☐➡ 📖
Options		CD hex	➡☐➡
Lambda	Excess air calculation control	T/F	
DoubleXL	Double cross-limiting control	T/F	
Bit2	User defined bits	T/F	
Bit3	User defined bits	T/F	
Bit4	User defined bits	T/F	
Bit5	User defined bits	T/F	
Bit6	User defined bits	T/F	
Bit7	User defined bits	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 4-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) 375U003, for details of these 'header' fields.

Demand. Heat demand. Maintained demand for the required heat.

Fuel_PV. Shows the amount of fuel to the system as an analogue value. This is the feedback signal from the process in units determined by the I/O Module.

Air_PV. Shows the amount of air to the system as an analogue value. This is the feedback signal from the process in units determined by the I/O Module.

FuelDmd. Shows the total fuel heat demand used to determine *Fuel_SP*. Calculated from cross-limiting algorithm and *BiasCL* (cross-limiting bias). *DblXLval* is included in the algorithm if *Options.DoubleXL* is TRUE.

CurHeat. Shows the current total heat. The heat currently being generated from the current fuel flow.

CurStAir. Shows the current Stoichiometric air flow. This is the air flow currently required for ideal combustion.

CurExAir. Shows the current percentage of excess air in the chamber. This requirement is for use in the cross-limiting algorithm derived from *Fuel_PV* and *Air_PV*. Using more air than the Stoichiometric amount in the combustion chamber increases the chances of complete combustion.

CurLambd. Shows the amount of air flow above or below the Stoichiometric (Lambda) (i.e. the quantity of air in the combustion process relative to the ideal amount). A Lambda value of 1.0 means that exactly the theoretical quantity of air needed for complete combustion is present.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation Failure Alarm. Evaluation failure in block's internal logic.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

ExcssAir. Shows the Excess air requirement. This is the percentage of air required of air in excess of the Stoichiometric combustion for use in the cross-limiting algorithm. This is derived from *Fuel_PV* and *Air_PV*. If *Options.Lambda* is TRUE this value is calculated from *Lambda*.

Lambda. Shows the Lambda air requirement. This is the ratio of air required of air in excess of the Stoichiometric combustion for use in the cross-limiting algorithm. A Lambda value of 1.0 means that exactly the theoretical quantity of air needed for complete combustion is present. If *Options.Lambda* is FALSE this value is calculated from *ExcssAir*. A lambda of >1 means there is more air than required; a lambda of < 1 means there is insufficient air.

BiasCL. Shows the cross-limiting bias. This is the bias applied to prevent small increases in fuel flow from affecting the *Air_SP* value, and small decreases in air flow from affecting the *Fuel_SP* value.

DblXLval. Shows the double cross-limiting value to be applied when *Options.DoublXL* is TRUE. This is the% value to which the <u>combustion air</u> setpoint is held back, above that calculated by the cross-limiting algorithm, whilst demand is increasing, and also the% value to which the <u>fuel</u> setpoint is held back, above that calculated by the cross-limiting algorithm, whilst demand is decreasing.

CV. Shows the calorific value of the fuel. The units applied to this value must correspond to the units used in *Fuel_SP*.

StchAir. Stoichiometric air requirements for combustion of fuel.

Note Air flow and fuel flow calculations must use the same units of measurement.

Fuel_SP. Fuel flow setpoint for fuel. Calculated from cross-limiting algorithm.

Air_SP. Combustion air flow setpoint. Calculated from cross-limiting algorithm.

Options. This bitfield controls the operation of the block.

- **Lambda.** Excess air method control. TRUE enables the value shown in *ExcssAir* to be displayed in *Lambda*. FALSE enables the value shown in *ExcssAir* to be displayed as a% (percentage).
- **DoublXL.** Double cross-limiting control. TRUE enables the use of the value configured in *DblXLval* (double cross-limiting value).

Implementation notes

If CMBNXLIM control modules are included in the configuration, the block is included in the database as a 'foreign' template. The runtime database size is increased by a template overhead of 2882 bytes and by 186 bytes for each instance of the control module.

VLV3WAY: RAISE LOWER CONTROL BLOCK

Block function

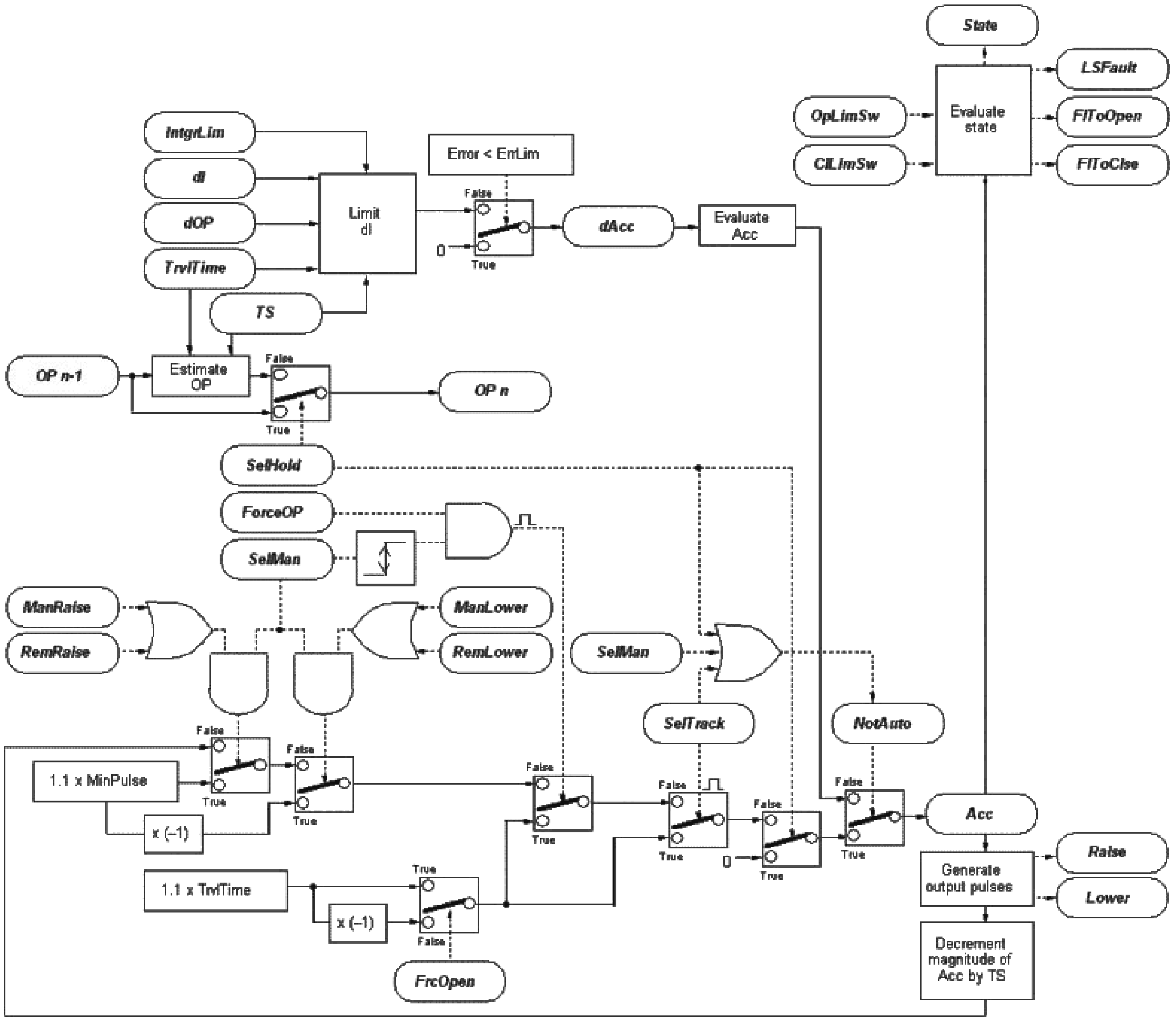


Figure 4-1: Block schematic

This block provides an interface to any actuator with two digital inputs, one for 'Raise' and one for 'Lower'. Both signals are based upon an incremental algorithm from a single set of three term settings. The block state is derived from limit switches and accumulator value.

In Manual, the block's output can be adjusted by manually issuing 'Raise' or 'Lower' signals. If Auto is selected, the block is automatically updated by the corresponding PID algorithm. Hold, Track and Forced Manual are available for interlocking purposes. Track and Forced Manual are used to drive the actuator to its limits. There is no facility to request a Track position other than those.

The RAISELWR block is intended to be used as part of the SETPOINT / 3_TERM / MAN_STAT / MODE combination of control blocks.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status	
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖	
OpLimSw	Open limit switch	T/F	➡□➡	
ClLimSw	Close limit switch	T/F	➡□➡	
dOP	Change in OP this iteration	%	➡□➡	
dI	Change due to I term	%	➡□➡	
TS	Algorithm sampling time	Secs	➡□➡	
Error	PV – SP	Eng	➡□➡	
dAcc	Adjusted change in output	Secs	□➡ 📖	
Acc	Accumulator value	Secs	□➡ 📖	
OP	Estimated value of the output	%	□➡ 📖	
Param1	User parameter # 1	Eng	➡□➡	
Param2	User parameter # 2	Eng	➡□➡	
Alarms		ABCD hex	□➡ 📖 🔈	
Software	Block RAM data sumcheck error/network failure	T/F		
EvalFail	Evaluation failure in block's internal logic	T/F		
Hardware	Hardware alarm	T/F		
LSFault	Limit switch fault	T/F		
FlToOpen	Failed to open	T/F		
FlToClose	Failed to close	T/F		
Combined	OR-ing of all alarm bits	T/F		
State	Derived position state	Enum		
StateAct	Active position state	(A)BCD hex		
Open	Open position	T/F	1	D
Close	Closde position	T/F	2	
Opening	Opening position	T/F	4	
Closing	Closing position	T/F	8	
LSFault	Limit switch fault position	T/F	1	C
StayPut	Stayput position	T/F	2	
			4	
			8	
Raise	Raise output pulse	T/F	□➡ 📖	
Lower	Lower output pulse	T/F	□➡ 📖	
MnRptTmr	Minimum repeat countdown timer	Secs	□➡ 📖	
MnRptTim	Minimum repeat time	Secs	➡□➡	
ManPsTmr	Manual pulse countdown timer	Secs	□➡ 📖	
MinPulse	Minimum pulse length	Secs	➡□➡	
TrvlTime	Travel time	Secs	➡□➡	
ErrLim	Error limit	Eng	➡□➡	
IntgrLim	Integral limit		➡□➡	

Table 4-1: Block parameters

Parameter	Function	Units		Status
Options		(A)BCD hex		
SelTrack	Select track mode	T/F	1	D
SelHold	Select hold mod1	T/F	2	D
SelMan	Select manual mode	T/F	4	D
ForceOP	Enable failsafe action	T/F	8	D
FrcOpen	'Safe' position is open	T/F	1	C
ManRaise	Generate manual raise output pulse	T/F	2	C
ManLower	Generate manual lower output pulse	T/F	4	C
RemRaise	Generate manual raise output pulse	T/F	8	C
RemLower	Generate manual lower output pulse	T/F	1	B
ClearAcc	Clear accumulator	T/F	2	B
Reset	Reset latched alarms	T/F	4	B
			8	B
Status		(A)BCD hex		
Forcing	Output forced	T/F	1	D
LastTrck	Track active during last iteration	T/F	2	D
LastFrc	Output forced during last iteration	T/F	4	D
Pulsing	Output pulses being generated	T/F	8	D
NotAuto	Not in auto mode (i.e. in hold, track or manual)	T/F	1	C
LastNotA	Not in auto during last iteration	T/F	2	C
ManReqst	Manual adjustment of OP requested	T/F	4	C
LastFOpn	Safe position during last iteration	T/F	8	C

Table 4-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document (Control Modules User Guide, HA084102), for details of these 'header' fields.

dOP. Change in the value of *OP* at this block iteration. Should be wired in from the associated 3_TERM block.

dI. Change in the value of the integral term (I) at this block iteration. Should be wired in from the associated 3_TERM block.

TS. PID algorithm sampling time in seconds. Should be wired in from the associated 3_TERM block.

Error. This field shows the difference between PV and SP, i.e. *Error* = PV – SP. Should be wired in from the associated SETPOINT block.

dAcc. Adjusted change in *OP* required by an iteration of the PID algorithm converted to the required length of pulse. This field is further adjusted to limit the integral action.

Acc. Accumulator value calculated as being $Acc_n = Acc_{n-1} + dAcc \pm TS$. The output pulses are switched on as soon as $|Acc| > MinPulse$ and are kept on for as long as $|Acc| > TS$.

If *SelTrack* is TRUE, or *ForceOP* is TRUE and *SelMan* changes from FALSE to TRUE, *Acc* is loaded with the maximum values of 1.1**TrvlTime* if *FrcOpen* is TRUE, and –1.1**TrvlTime* if *FrcOpen* is FALSE.

OP. Estimated value of the output in %.

$OP_n = OP_{n-1} + TS \infty 100 / TrvlTime$ if *Raise* pulse is issued, and

$OP_n = OP_{n-1} - TS \infty 100 / TrvlTime$ if *Lower* pulse is issued.

Param1, Param2. Optional parameters. Associate additional floating-point data with the block.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **LSFault.** Limit switches indicate actuator is both 'Open' and 'Closed'.
- **FlToOpen.** Actuator has not responded to an 'Open' demand. Alarm is latched.
- **FlToClse.** Actuator has not responded to a 'Close' demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

State (Open/Closed/Opening/Closing/LsFault/StayPut). Current position state. Derived from the limit switches and the value of *Acc*.

StateAct. Bitfield indicating the block's active position, i.e. the value of the *State* parameter. Only one *StateAct* bit can be active at one time.

Raise. Real digital output representing the block's 'Raise' demand.

Lower. Real digital output representing the block's 'Lower' demand.

MnRptTmr. Minimum repeat countdown timer in seconds used internally to record a time exceeding the minimum repeat time *MnRptTim*.

MnRptTim. Minimum repeat time in seconds. In Auto, reissuing of consecutive output pulses is disabled until a time greater than *MnRptTim* has elapsed.

ManPsTmr. Manual pulse countdown timer in seconds. Internally set to record if a manually issued pulse has exceeded the minimum pulse length *MinPulse*.

MinPulse. Minimum pulse length in seconds. If |*Acc*| > *MinPulse*, a Raise/Lower pulse is started.

TrvlTime. Travel time from 0% to 100% in seconds.

ErrLim. Error limit. *dAcc* is set to zero if the error magnitude (|*Error*|) is less than *ErrLim*.

IntgrLim. Integral limit used to adjust the ratio of integration rate to travel rate, typically set to 1.2. The value entered in this field should be between 1.0 and 1.4.

Options. Bitfield for selecting the different operational options of the block.

- **SelTrack.** Select track mode. When TRUE, this bitfield forces the output to the 'Closed' position if *FrcOpen* is FALSE or to the 'Open' position if *FrcOpen* is TRUE. Normally wired in from the associated MAN_STAT block.
- **SelHold.** Select hold mode. When TRUE, freezes the output at its current position by switching off the output pulses. Normally wired in from the associated MAN_STAT block.
- **SelMan.** Select manual mode. When TRUE, enables manual adjustment of the output through the option bits *ManRaise*, *ManLower*, *RemRaise* and *RemLower*. Normally wired in from the associated MAN_STAT block.
- **ForceOP.** *ForceOP* TRUE, causes the block to go into a 'safe' position, before manual mode takes over. In this transient mode, OP adopts a value that depends on the state of *FrcOpen*. See *FrcOpen* below. *ForceOP* is normally wired in from the associated MAN_STAT block.
- **FrcOpen.** Used to derive the force or track position. 'Closed' (*OP* = 0%) if TRUE, 'Open' (*OP* = 100%) if FALSE. Normally wired in from the associated MAN_STAT block (*SafeHigh* field).
- **ManRaise, ManLower.** Used to manually generate *Raise*/*Lower* outputs when in manual (e.g. from a T640 front panel). Internally reset to FALSE.
- **RemRaise, RemLower.** Offers similar functionality as *ManRaise* and *ManLower*. Can be used to manually generate *Raise*/*Lower* outputs from a supervisory computer. Internally reset to FALSE.

Status. Bitfield showing status of the calculations.

- **Forcing.** Used internally to indicate that the output is being forced (*Acc* loaded with (±1.1**TrvlTime* or ±1.1**MinPulse*).
- **ManReqst.** Manual adjustment of the output is being requested. In manual, this bit is internally used to allow the loading of *Acc* with a value of magnitude equal to or higher than 1.1**MinPulse*.

Implementation note

If RaiseLwr control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2396 bytes and by 158 bytes for each instance of the control module.

Note	For this application module please refer to factory.

Intentionally left blank

CHAPTER 5 DCM APPLICATION BLOCKS

The Devolved Control Module (DCM) category of Application Function Block Templates provides the control strategy with functions for collecting data from I/O Subsystems. DCM blocks running in the T2900/T800/T940(X) provide communications across a network with remote 2500 (Input/Output System), and other Series 2000 target instruments.

Each DCM block presents a view on particular data values in the target instrument, and also allows configuration of the communications parameters. The remote 2500/Series 2000 data appears in the local DCM block as input and/or output fields that can be 'wired' to the control strategy running in the T2900/T800/T940(X), and so interacted with.

Intentionally left blank

VLV3WAY: 2500E PID LOOP BLOCK

Block function

This block gives a view of a PID control loop running in the 2500E (8-loop) I/O Controller. The block type provides an alternative to the D25eLoop block (described in chapter 8 of the LIN blocks reference manual HA082375U003), giving greater alarm integration and the ability to map user-selected data from the 2500E into this LIN block.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA082375U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units			Status
Port[S]	Specifies comms port and protocol number	Menu			
Profile[S]	Base name of CSV file defining address-map profile	Alphanumeric			
ST_REV	Fieldbus Foundation static revision number	Integer			▶▭▶
Instr_No[S]	Instrument number (default 0=offline)	Integer			
Slot_No[S]	Loop number in 2500	Integer			
Chan_No[S]	Not applicable this software release.	Integer			
Method	Reserved for future use block's ST update routine	Menu			📖
Mode	Operating mode of PID loop	Menu			▶▭▶
RemEnabl	TRUE enables, FALSE disables, remote setpoint value	T/F			▶▭▶
PV	Process variable value	EngA			▭▶ 📖
SP	Setpoint value	EngA			▶▭▶
OP	Output value	EngB			▶▭▶
RemoteSP	Remote Setpoint value	EngA			▶▭▶
A0 to A3	General purpose variables for use in profiles	EngC to EngF			▶▭▶
Word0 to Word1	General purpose variables for use in profiles	ABCD hex			▶▭▶
Bit0	Bit0	T/F	1	D	
Bit1	Bit1	T/F	2	D	
Bit2	Bit2	T/F	4	D	
Bit3	Bit3	T/F	8	D	
Bit4	Bit4	T/F	1	C	
Bit5	Bit5	T/F	2	C	
Bit6	Bit6	T/F	4	C	
Bit7	Bit7	T/F	8	C	
Bit8	Bit8	T/F	1	B	
Bit9	Bit9	T/F	2	B	
Bit10	Bit10	T/F	4	B	
Bit11	Bit11	T/F	8	B	
Bit12	Bit12	T/F	1	A	
Bit13	Bit13	T/F	2	A	
Bit14	Bit14	T/F	4	A	
Bit15	Bit15	T/F	8	A	

Table 5-1: Block parameters

Parameter	Function	Units			Status
Alarms					➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F			
Comms	Any comms error to 2500 (e.g. timeout)	T/F			
Config	Comms config. error, or invalid profile	T/F			
Reset	(*Not implemented*)	T/F			
SnsBrk	Sensor break (LoopSw.Snsr_Brk or Options.SetSnsrB)	T/F			
LoopBrk	Loop break (LoopSw.Loop_Brk or Options.SetLoopB)	T/F			
HiHi	See 'Alarms' in the 'Block Specification Menu', below, for a description	T/F			
Hi	See 'Alarms' in the 'Block Specification Menu', below, for a description	T/F			
HiDev	Trips when PV-SP > HiDev (clears if PV-SP+Hyst < HiDev)	T/F			
LoDev	Trips when SP-PV > LoDev (clears if SP-PV+Hyst < LoDev)	T/F			
Lo	See 'Alarms' in the 'Block Specification Menu', below, for a description	T/F			
LoLo	See 'Alarms' in the 'Block Specification Menu', below, for a description	T/F			
EvalFail	Evaluation failure in block's internal logic	T/F			
Combined	OR-ing of all alarm bits	T/F			
AlarmSW	Process alarm status word	ABCD hex			➡ 📖
Alm1_Act	Alarm 1 active	T/F	1	D	
Alm1_Ack	Alarm 1 acknowledged	T/F	2		
Alm2_Act	Alarm 2 active	T/F	4		
Alm2_Ack	Alarm 2 acknowledged	T/F	8		
Alm3_Act	Alarm 3 active	T/F	1	C	
Alm3_Ack	Alarm 3 acknowledged	T/F	2		
Alm4_Act	Alarm 4 active		4		
Alm4_Ack	Alarm 4 acknowledged		8		
LoopSW	Loop alarm status word	ABCD hex			➡ 📖
CtrlFrz	Control freeze	T/F	1	D	
Snsr_Brk	PV sensor break	T/F	2		
SRL_Act	Setpoint rate limiter active	T/F	4		
RemSpAct	Remote setpoint active	T/F	8		
Servo	PID servo signal	T/F	1	C	
Debump	PID debump signal	T/F	2		
LoopBrk	Loop break	T/F	4		
IntFrz	Integral freeze	T/F	8		
RemFault	Remote fault	T/F	1	B	
DirRev	Direct (TRUE)/Reverse (FALSE) acting	T/F	2		
Unused			4		
OP_Lim	Output limited	T/F	8		
AutoTune	Auto tune active	T/F	1	A	
AdapTune	Adaptive tune enabled	T/F	2		
AutDroop	Automatic droop compensation enabled	T/F	4		
Man_Mode	Mode. TRUE = Manual, FALSE = Auto	T/F	8		
Deviatn	Deviation value, i.e. PV – SP	EngA			➡
Hyst[5]	Hysteresis band	EngA			➡➡
HiHi	High high absolute alarm threshold value	EngA			➡➡
Hi	High absolute alarm threshold value	EngA			➡➡
HiDev	High deviation alarm threshold value	EngA			➡➡

Table 5-1: Block parameters

Parameter	Function	Units	Status
LoDev	Low deviation alarm threshold value	EngA	➔□➔
Lo	Low absolute alarm threshold value	EngA	➔□➔
LoLo	Low low absolute alarm threshold value	EngA	➔□➔
XP[S]	Proportional band		➔□➔
TI[S]	Integral time	hh:mm:ss	➔□➔
TD[S]	Derivative time	hh:mm:ss	➔□➔
Options	Polling options	ABCD hex	➔□➔
AlmDvLcl	TRUE = Abs. alarms derived locally; FALSE = Absolute alarms derived from AlarmSW	T/F — 1	
SetSnsrB	Sensor break (LoopSw.Snsr_Brk or Options.SetSnsrB)	T/F — 2	D
SetLoopB	Loop break (LoopSw.Loop_Brk or Options.SetLoopB)	T/F — 4	
		— 8	
B0, B1	General purpose variables for use in Profiles	T/F	➔□➔

[S] *'Static' data, only read if a 'reset' occurs in either master or slave. Other data are read/written every comms cycle.*

Table 5-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) 375U003, for details of these 'header' fields.

Port. (MODBUS_1/MODBUS_2/MODBUS_3/MODBUS_4/PROFDP_1/PROFDP_2) Definition of comms port and protocol number. This allows for multiple ports on the same protocol. Defaults to 'MODBUS_1'. (*Not all menu items are currently supported.*)

Profile. The base name of a profile-definition file. Any DCM block can have an optional profile that may be used to override the default address mapping of the block. A profile is specified in a CSV format file, and its content depends on which protocol it applies to.

ST_REV. Fieldbus Foundation static revision number. Whenever *ST_REV* changes, or equals 0, all the static data in the devolved control module, marked with [S] in the table, is re-read.

Instr_No. Instrument number, dependent on protocol. For Modbus comms, *Instr_No* is in the range 1-247. For Profibus-DP communications, it is in the range 1-125. The default value is 0, meaning 'offline'. *Instr_No* identifies the slave (e.g. 2500) address. (*Note that for Profibus in the T940, values 1 and 2 are illegal.*)

Slot_No. Slot number of the loop in the instrument, i.e. the instance number of the devolved control module.

Chan_No. Not applicable this software release. Leave as default value.

Method. (Template/Block/Native) Indicates the location of the block's ST update routine. By default this is embedded in the template, but the template update routine may be overridden by a different method located in the block pool. (*Currently this field can be set only to 'Template'.*)

Mode. (AUTO/MANUAL) Current operating mode for the selected loop (as shown in 'Slot No.).

RemEnabl. Enables remote parameter. When TRUE, remote control is permitted. When FALSE, remote control is disabled.

RemoteSP. Remote Setpoint value for use when RemEnabl (above) is True.

Alarms. There are a number of alarms in this block, six of which (HiHi, Hi, HiDev, LoDev, Lo and LoLo) are process alarms.

The deviation alarms (HiDev and LoDev) are always derived locally to the block, and use the calculated 'Deviatn' value (PV-SP) and the 'Hyst' value entered in the relevant configuration field.

If 'Options.AlmDvLcl' = True, the four absolute alarms (HiHi, Hi, Lo and LoLo) are derived locally, using the alarm thresholds entered in the appropriate configuration fields. If 'Options.AlmDvLcl' = False, these configuration values are ignored, and the alarms derived remotely from the 2500 alarm status word.

- **Software.** Sumcheck error in block's RAM data.
- **Comms.** Asserted if any communications error is detected in the 2500, e.g. Timeout value exceeded.
- **Config.** Asserted if any communication configuration error, or invalid profile is detected, e.g. incorrect Port configuration.
- **Reset.** For future use.
- **Snsr_Brk.** Asserted if a Sensor Break has occurred (LoopSw.Snsr_Brk or Options.SetSnsrB).
- **LoopBrk.** Asserted if a Loop Break has occurred (LoopSw.Loop_Brk or Options.SetLoopB).
- **HiHi** If 'Options.AlmDvLcl' = True the alarm is active when the PV value exceeds the configured value 'HiHi'and remains active until the PV value falls below (HiHi-Hyst). If 'Options.AlmDvLcl' = False, the alarm status is derived from the 2500 Alarm status word.
- **LoLo.** If 'Options.AlmDvLcl' = True the alarm is active when the PV value falls below the configured value 'LoLo'and remains active until the PV value rises above (LoLo+Hyst). If 'Options.AlmDvLcl' = False, the alarm status is derived from the 2500 Alarm status word.
- **Hi, Lo.** Similar to HiHi and LoLo respectively, but use the configured values for 'Hi' and 'Lo' instead of 'HiHi' and 'LoLo'.
- **HiDev, LoDev.** Asserted if *Deviatn* > *HiDev*, and *Deviatn* < *LoDev*, respectively, where 'Deviatn' = PV-SP. The alarms do not clear until *Deviatn* < *HiDev-Hyst*, and *Deviatn* > *LoDev+Hyst*, respectively. Deviation alarms are always derived locally and are therefore unaffected by the status of 'AlmDvLcl'.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

AlarmSW. Loop alarm status word.

- **Alm1_Act to Alm4_Act.** Shows the active status of Loop Alarms 1 to 4 (see note below).
- **Alm1_Ack to Alm4_Ack.** Shows the acknowledgement status of Loop Alarms 1 to 4 (see note below).

Note: 'Alm1_Act' and 'Alm_1Ack' refer to the 'HiHi' alarm. Alm_2Act/Ack refer to the 'Hi' alarm. Alm3_Act/Ack refer to the 'Lo' alarm and Alm4_Act/Ack refer to the 'LoLo' alarm.

LoopSW. Alarm generation status word.

- **CtrlFrz.** When set TRUE, control is frozen.
- **Snsr_Brk.** When TRUE, a PV sensor break has occurred. An SnsBrk alarm is set if this, and/or ‘Options. SetSnsrB’ is ‘True’.
- **SRL_Act.** When set TRUE, Setpoint rate limiter is active.
- **RemSpAct.** When set TRUE, Remote setpoint active.
- **Servo.** When set TRUE, a PID servo signal.
- **Debump.** PID debump signal.
- **LoopBrk.** When TRUE, a Loop break has occurred. A LoopBrk alarm is set if this, and/or ‘Options.SetLoopB’ is ‘True’.
- **IntFrz.** When set TRUE, the Integral time value will freeze.
- **RemFault.** When set TRUE, a Remote fault has occurred.
- **DirRev.** Shows current acting direction. TRUE = Direct, FALSE = Reverse.
- **OP_Lim.** When set TRUE, the output limited.
- **AutoTune.** When set TRUE, auto tune active.
- **AdapTune.** When set TRUE, adaptive tune enabled.
- **AutDroop.** When set TRUE, automatic droop compensation is enabled.
- **Man_Mode.** Shows current operating mode. TRUE = Manual, FALSE = Auto.

Deviatn. Calculated deviation (error) between the process variable and the Setpoint (*Deviatn = PV - SP*). This is used to generate the deviation alarms.

Hyst. Hysteresis bandwidth value, applied in engineering units to the absolute and deviation alarms. All alarms are triggered immediately they exceed the limit set, and remain triggered until they are within the limit by more than the hysteresis value.

HiHi, Hi, Lo, LoLo. Absolute alarm thresholds defining when block alarms trip, when AlmDvLcl = True. When False, Absolute alarm status is derived from the 2500 alarm status word.

HiDev, LoDev. Threshold values for deviation alams. HiDev alarms are asserted if *Deviatn > HiDev*; LoDev alarms are asserted if *Deviatn< LoDev*, respectively, where ‘Deviatn’ = PV-SP.

XP. Proportional Band value of the PID control.

TI. Integral time value of the PID control.

TD. Derivative time value of the PID control.

Options. Bitfield selecting alarms derived from option.

- **AlmDvLcL.** When ‘True’, absolute alarms are derived locally (from *PV, HiHi, Hi, Lo and LoLo*). When ‘False’ alarms are derived from the 2500 Alarm status word. AlmDvLcl has no effect on deviation alarms which are always derived locally.
- **SetSnsrB.** The Sensor Break (SnsBrk) alarm becomes active, under sensor break conditions if SetSnsrB and/or LoopSw.Snsr_Brk is True.
- **SetLoopB.** The loop Break (LoopBrk) alarm becomes active, under sensor break conditions if SetSnsrB and/or LoopSw.LoopBrk is True.

UNITS

Some of the block parameters show their default units as EngA, EngB, EngC and so on.

Changing say, ‘EngA’ for one parameter, immediately changes all other parameters’ EngA entries in the block as well. This not only speeds up the configuration process, it also ensures that a coherent set of units is used across the block. For example if a PV’s units were to be entered as ‘mmHg’, then the alarm threhold units (amongst others) would automatically be changed to ‘mmHg’ as well.

CHAPTER 6 DUTY APPLICATION BLOCKS

The DUTY category of Application Function Block Templates provides the control strategy with functions for controlling a pair of on/off motor devices.

Intentionally left blank

VLV3WAY: DUTY STANDBY BLOCK

Block function

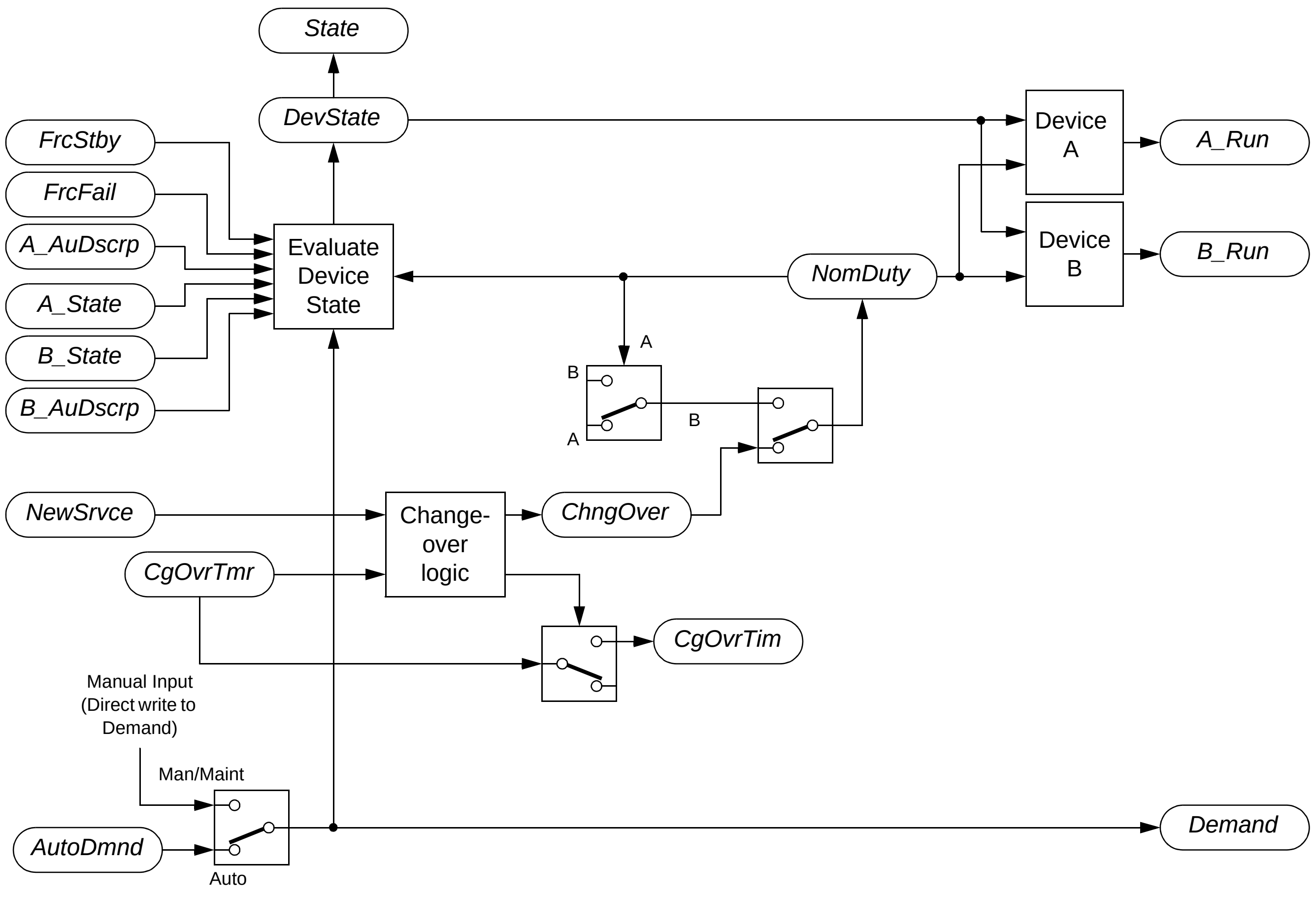


Figure 6-1: Block schematic

Please refer to the schematic. The DutyStby block generates a demand to a pair of on/off motor control modules, by using one device to act as a standby to the device selected as duty.

In manual mode, the motors may be run as a duty/standby pair or stopped by the operator. In automatic mode, the motors may be run as a duty/standby pair or stopped by a controlling sequence. The individual motors cannot be directly started or stopped from this module.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	▭➡ 📖
ManAct	Manual mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
A_Tag	Tag of device A	String	➡▭➡
A_State	State of device A	Enum	➡▭➡
A_AuDscrp	Device A cannot respond to automatic control	T/F	
A_Param1	Device A parameter #1	Eng	
B_Tag	Tag of device B	String	➡▭➡ &
B_State	State of device B	Enum	➡▭➡ &
B_AuDscrp	Device B cannot respond to automatic control	T/F	
B_Param1	Device B parameter #1	Eng	➡▭➡
AutoDmnd	Automatic demand	T/F	➡▭➡
NomDuty	Device nominated as the duty (A/B)	Enum	▭➡
NewSrvce	Operator initiated actions to change the duty service	Enum	▭➡ ?
OP	Output to devices	CD hex	▭➡
A_Run	Run device A	T/F	
B_Run	Run device B	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Claims	Resource management claims	CD hex	➡▭➡
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	▭➡ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 6-1: Block parameters

Parameter	Function	Units	Status
Alarms		ABCD hex	➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	Hardware alarm	T/F	
RunInIdl	Device running in idle state	T/F	
SbyUnavl	Standby device unavailable	T/F	
SbyRunng	Standby device running	T/F	
DevFlSttt	Device fail to start	T/F	
DevFlStp	Device fail to stop	T/F	
SbyFail	Standby device failed	T/F	
BthRun	Both devices running	T/F	
FailFrcd	Devices forced to fail	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	State demand	T/F	➡
CgOvrTmr	Automatic changeover countdown timer	Secs	➡ 📖 🔍
CgOvrTim	Automatic changeover period	Secs	⬅➡
DevState	Device state	Enum	➡ 📖
DevStAct	Active device state	CD hex	➡ 📖
Idle	Idle state	T/F	
DtyStart	Duty starting state	T/F	
DtyRun	Duty running state	T/F	
NewSrvce	New service selection (transient state)	T/F	
SbyStart	Standby starting state	T/F	
SbyRun	Standby running state	T/F	
Stopping	Acknowledgement #6	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
State	Derived state	Enum	➡ 📖
StateAct	Active state	(A)BCD hex	➡ 📖
Stopped	Stopped state	T/F	
Running	Running state	T/F	
Stopping	Stopping state	T/F	
Starting	Starting state	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 6-1: Block parameters

Parameter	**Function**	**Units**			**Status**
Status		ABCD hex			➔☐➔
CgOvrRun*	Changeover timer in operation	T/F	1		
A_Runng*	Device A running	T/F	2	D	
B_Runng*	Device B running	T/F	4		
A_Stoppd*	Device A Stopped	T/F	8		
B_Stoppd*	Device B Stopped	T/F	1		
AisDuty*	Device A is the duty	T/F	2	C	
BisDuty*	Device B is the duty	T/F	4		
ChngOver*	Changeover is being requested	T/F	8		
Bit8	User status #8	T/F	1		
Bit9	User status #9	T/F	2	B	
Bit10	User status #10	T/F	4		
Bit11	User status #11	T/F	8		
Bit12	User status #12	T/F	1		
Bit13	User status #13	T/F	2	A	
FailFrcd	Device forced to fail	T/F	4		
DevFlStp	Device failed to stop	T/F	8		
Hardware	I/O hardware failure input	CD hex			➔☐➔
Bit0	Hardware failure #0	T/F			
Bit1	Hardware failure #1	T/F			
Bit2	Hardware failure #2	T/F			
Bit3	Hardware failure #3	T/F			
Bit4	Hardware failure #4	T/F			
Bit5	Hardware failure #5	T/F			
Bit6	Hardware failure #6	T/F			
Bit7	Hardware failure #7	T/F			
					Ø Ø Ø Ø Ø Ø Ø Ø
Options		(A)BCD hex			➔☐➔
AuDmdTrk	Automatic demand track	T/F	1		
Avail1	User device available #1	T/F	2	D	
Avail2	User device available #2	T/F	4		
NotAvail1	User device not available #1	T/F	8		
NotAvail2	User device not available #1	T/F	1		
OPtkIdle	Show fuel B on faceplate	T/F	2	C	
FrcStby	Fixed load fuel	T/F	4		
FrcFail	Select device state to purge	T/F	8		
ReLdTmr	Double cross-limiting combustion	T/F	1		
Bit9	User option #9	T/F	2	B	
Bit10	User option #10	T/F	4		
Bit11	User option #11	T/F	8		
Reset	Reset latched discrepancy	T/F			?
Availble	Available for automatic control	T/F			☐➔ 📖
FpltType	Supervisory faceplate type	String			
Param1	User parameter # 1	Eng			➔☐➔
Param2	User parameter # 2	Eng			➔☐➔

** Input wiring will disrupt the normal operation of the block*

Table 6-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Mode. (Manual/Auto). Selects current operating mode.

A_Tag, B_Tag. The tag of the specified device.

A_State, B_State. The current state of the specified device.

A_AuDscrp, B_AuDscrp. Automatic Discrepancy. TRUE when the specified control module is not available or its automatic demand is not equal to its demand.

A_Param1, B_Param1. Optional parameter. Associates additional floating-point data with specified control module.

AutoDmnd. Controls the demand in Automatic mode.

NomDuty. (A/B). Nominated duty. Selects either ‘A’ or ‘B’ to be the preferred duty (lead) device.

NewSrvce. (NoAction, ChngOver, MakeDuty). *ChngOver* makes the stopped Standby device the running Duty device. *MakeDuty* makes the running Standby device become the running Duty device. Internally set to *NoAction* after the command action.

OP. Instruction for specified device.

- **A_Run, B_Run.** TRUE runs the specified device.

Claims. Used to request sole control of the control module. The control module resource may be controlled by up to eight sequences or other control module users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Alarms.

- **Software.** Sumcheck error in block’s RAM data.
- **EvalFail.** Evaluation failure in block’s internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the control module.
- **RunInIdl.** Run In Idle. One of the devices is running but the state is idle.
- **SbyUnavl.** Standby Unavailable. The standby device cannot respond if the duty fails.
- **SbyRunng.** Standby Running. The standby device is running i.e. the duty device has failed.
- **DevFlStt.** Device Fail To Start. One of the devices has not responded to a ‘Start’ demand. Alarm is latched.
- **DevFlStp.** Device Fail To Stop. One of the devices has not responded to a ‘Stop’ demand. Alarm is latched.
- **SbyFail.** Standby Fail. The standby device has failed leaving both devices failed.
- **BthRun.** Both devices are Running.
- **FailFrcd.** Failure Forced. A process input has forced both devices to fail to stopped.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand. FALSE = ‘Stop’; TRUE = ‘Run as Duty/Standby pair’.

CgOvrTmr. Changeover Timer. Internally set to the changeover time after each changeover. A device changeover is performed after the timer counts down to zero.

CgOvrTim. Changeover Time. The time, in seconds, between each automatic changeover. Setting the time to zero disables automatic changovers.

DevState. (Idle/DutyStart/DutyRun/NewSrvce/SbyStart/SbyRun/Stopping). Current device state derived from motor ‘A’ and ‘B’ states and the demand.

Status. This bitfield shows the status of the control module.

- **CgOvrRun.** TRUE indicates that an automatic changeover will occur when the changeover timer counts down to zero.
- **A_Runng, B_Runng.** TRUE indicates that the specified device state is ‘Running’.
- **A_Stoppd, B_Stoppd.** TRUE indicates that the specified device state is ‘Stopped’.
- **AisDty, BisDty.** TRUE indicates that the specified device is the nominated duty.
- **ChngOver.** Internal flag indicating command to changeover received.
- **Bit8-Bit13.** Optional status bits. Associates additional Boolean data with the module.
- **FailFrcd.** Latched TRUE when devices have been forced to fail stopped. Set FALSE during the reset action.
- **DevFlStp.** Latched TRUE when one of the devices has not responded to a ‘Stop’ demand. Set FALSE during the reset action.

Hardware. Asserts the hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the control module.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **OPtkIdle.** TRUE makes *OP* track device running signal while in the idle state. FALSE ensures that *OP* is FALSE in the idle state.
- **FrcStby.** Force the stopped standby device to start and the running duty device to stop.
- **FrcFail.** Force the running standby device to stop.
- **ReLdTmr.** Reload changeover timer.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when available for automatic control, i.e. in automatic mode, either device not in Automatic Discrepancy and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the control module. Allows control modules of this type to be represented by different supervisory computer faceplates.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the control module.

Implementation notes

Note	If DutyStby control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2808 bytes and by 164 bytes for each instance of the control module.

VLV3WAY: DEMAND SCHEDULER FOR UP TO SIX ON/OFF DEVICES BLOCK

Block function

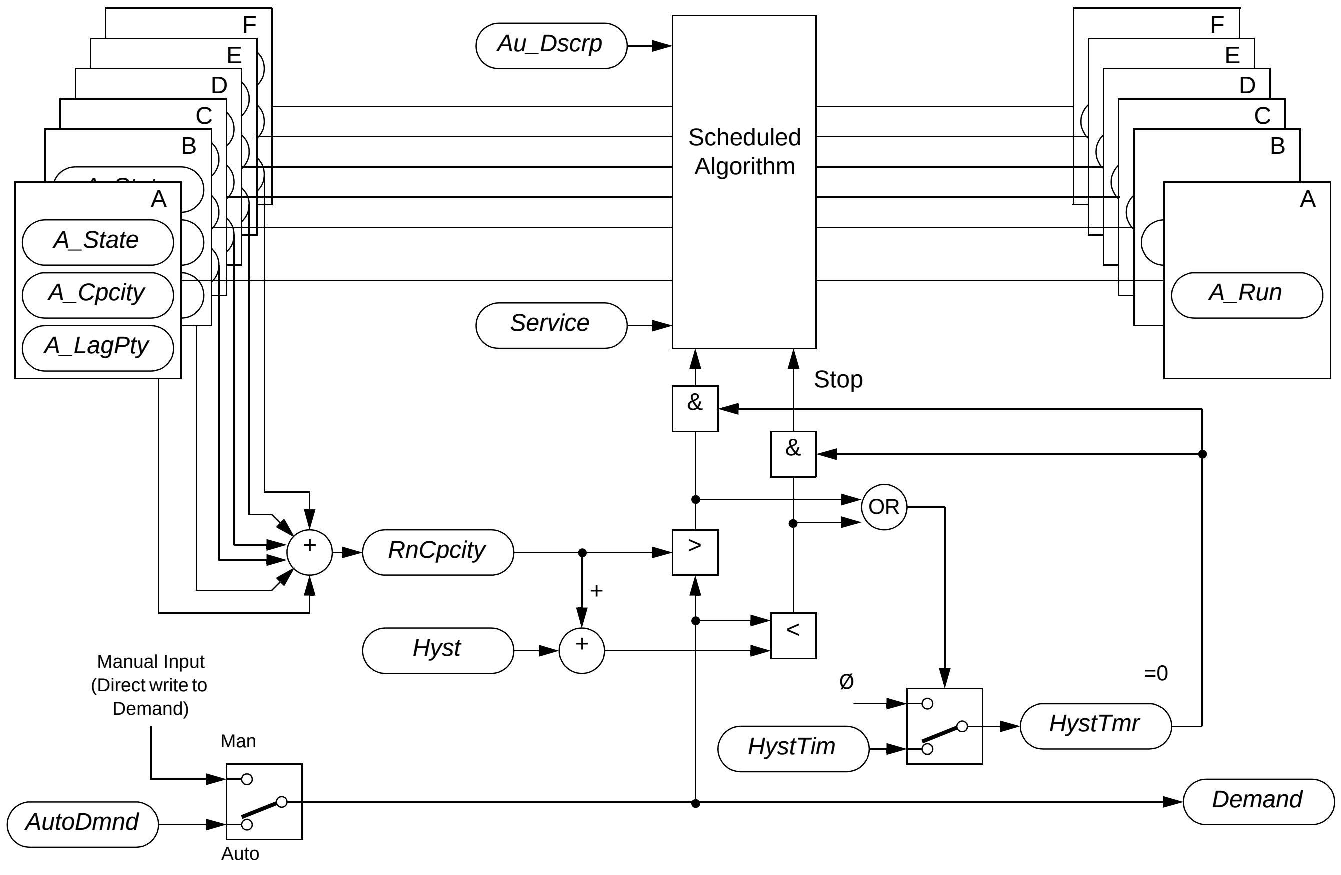


Figure 6-1: Block schematic

Please refer to the schematic. The DmdSchdl block generates a demand to up to six on/off devices by using a lag priority to determine the schedule order. Devices may be declared as lead or lag. All available lead devices are run wherever there is ad demand to be met.

In manual mode, the demand, which ultimately determines the number of running devices, may be set directly by an operator. In automatic mode, the demand may be set by a controlling sequence or master controller.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	▭➔ 📖
ManAct	Manual mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
A_State	State of device A	Enum	➔▭➔
A_Cpcity	Capacity of device A	Eng	➔▭➔
A_LagPty	Priority of device A when a 'Lag'	1-255	➔▭➔
B_State	State of device B	Enum	➔▭➔ &
B_Cpcity	Capacity of device B	Eng	➔▭➔
B_LagPty	Priority of device A when a 'Lag'	1-255	➔▭➔
C_State	State of device C	Enum	➔▭➔
C_Cpcity	Capacity of device C	Eng	➔▭➔
C_LagPty	Priority of device C when a 'Lag'	1-255	➔▭➔
D_State	State of device D	Enum	➔▭➔
D_Cpcity	Capacity of device D	Eng	➔▭➔
D_LagPty	Priority of device D when a 'Lag'	1-255	➔▭➔
E_State	State of device E	Enum	➔▭➔
E_Cpcity	Capacity of device E	Eng	➔▭➔
E_LagPty	Priority of device E when a 'Lag'	1-255	➔▭➔
F_State	State of device F	Enum	➔▭➔
F_Cpcity	Capacity of device F	Eng	➔▭➔
F_LagPty	Priority of device F when a 'Lag'	1-255	➔▭➔
AuDscrp		CD hex	▭➔
A	Device A cannot respond to automatic control	T/F	
B	Device B cannot respond to automatic control	T/F	
C	Device C cannot respond to automatic control	T/F	
D	Device D cannot respond to automatic control	T/F	
E	Device E cannot respond to automatic control	T/F	
F	Device F cannot respond to automatic control	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Service		CD hex	▭➔
A	Service of device A	T/F	
B	Service of device B	T/F	
C	Service of device C	T/F	
D	Service of device D	T/F	
E	Service of device E	T/F	
F	Service of device F	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 6-1: Block parameters

Parameter	Function	Units	Status	
Hyst	Scheduling hysteresis			
AutoDmnd	Automatic demand	T/F	▶□▶	
Alarms		ABCD hex	□▶ 📖 🔊	
Software	Block RAM data sumcheck error/network failure	T/F		
EvalFail	Evaluation failure in block's internal logic	T/F		
Stopped	Hardware alarm	T/F		
LdNotRun	Device running in idle state	T/F		
SttUnavl	Standby device unavailable	T/F		
StpUnavl	Standby device running	T/F		
TooMany	Device fail to start	T/F		
TooFew	Device fail to stop	T/F		
DevFlStp	Standby device failed	T/F		
DevFlStt	Both devices running	T/F		
Combined	OR-ing of all alarm bits	T/F		
Demand	State demand	T/F	□▶	
HystTmr	Hysteresis countdown timer	Enum	□▶ 🔍	
HystTim	Timed hysteresis time	Enum	□▶	
OP	Output to devices	CD hex	□▶	
A_Run	Run device A	T/F		
B_Run	Run device B	T/F		
C_Run	Run device C	T/F		
D_Run	Run device D	T/F		
E_Run	Run device E	T/F		
F_Run	Run device F	T/F		
			Ø Ø Ø Ø Ø Ø Ø Ø	
Status		ABCD hex	▶□▶	
CgOvrRun*	Changeover timer in operation	T/F	1	D
A_Runng*	Device A running	T/F	2	D
B_Runng*	Device B running	T/F	4	D
A_Stoppd*	Device A Stopped	T/F	8	D
B_Stoppd*	Device B Stopped	T/F	1	C
AisDuty*	Device A is the duty	T/F	2	C
BisDuty*	Device B is the duty	T/F	4	C
ChngOver*	Changeover is being requested	T/F	8	C
Bit8	User status #8	T/F	1	B
Bit9	User status #9	T/F	2	B
Bit10	User status #10	T/F	4	B
Bit11	User status #11	T/F	8	B
Bit12	User status #12	T/F	1	A
Bit13	User status #13	T/F	2	A
FailFrcd	Device forced to fail	T/F	4	A
DevFlStp	Device failed to stop	T/F	8	A

Table 6-1: Block parameters

Parameter	Function	Units	Status
Options		(A)BCD hex	⇄
AfrcLead	Force device A to be new lead	T/F	1 D
BfrcLead	Force device B to be new lead	T/F	2 D
CfrcLead	Force device C to be new lead	T/F	4 D
DfrcLead	Force device D to be new lead	T/F	8 D
EfrcLead	Force deviceE to be new lead	T/F	1 C
FfrcLead	Force device F to be new lead	T/F	2 C
AuDmdTrk	Automatic demand track	T/F	4 C
Avail1	User device available #1	T/F	8 C
Avail2	User device available #2	T/F	1 B
NotAvail1	User device not available #1	T/F	2 B
NotAvail2	User device not available #1	T/F	4 B
FrcNewLd	Force a new lead to run	T/F	8 B
CyclLead	Cycle the lead	T/F	1 A
LdMstRun	Lead must always run	T/F	2 A
Bit14	User option #14	T/F	4 A
Bit15	User option #15	T/F	8 A
Reset	Reset latched discrepancy	T/F	
Availble	Available for automatic control	T/F	→ 📖
Claims	Resource management claims	CD hex	⇄
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	→ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
FpltType	Supervisory faceplate type	String	
Param1	User parameter # 1	Eng	⇄
Param2	User parameter # 2	Eng	⇄

Table 6-1: Block parameters

Parameter	Function	Units	Status
OP_check	For implementation only	ABCD hex	
NextStrt	For implementation only	ABCD hex	
LagStrt	For implementation only	ABCD hex	
StrtPty	For implementation only	ABCD hex	
NextStop	For implementation only	ABCD hex	
LagStop	For implementation only	ABCD hex	
StopPty	For implementation only	ABCD hex	
AvlStrt	For implementation only	ABCD hex	
AvlStop	For implementation only	ABCD hex	
RnCpcity	For implementation only - Total capacity of all running device	Eng	

** Input wiring will disrupt the normal operation of the block*

Table 6-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Mode (Manual/Auto). Selects current operating mode.

A_State to F_State. The current state of the specified device.

A_Cpcity to F_Cpcity. The capacity of the specified device in demand units.

A_LagPty to A_LagPty. The priority of the specified device in lag service.

AuDscrp.

- **A to F.** Automatic Discrepancy. TRUE when the specified device is not available or its automatic demand is not equal to its demand.

Service.

- **A to F (Lead/Lag).** Indicates the specified device service.

Hyst. Specifies the scheduler hysteresis demand units.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Stopped.** A device has stopped when required to meet the demand.
- **LdNotRun.** No lead devices are running to meet the demand.
- **SttUnavl.** No devices are available to start.
- **StpUnavl.** No devices are available to stop.
- **TooMany.** Too many devices are running for the current demand.
- **TooFew.** Too few devices are running for the current demand.
- **DevFlStp.** One of the devices has not responded to a 'Start' demand. Alarm is latched.
- **DevFlStt.** One of the devices has not responded to a 'Stop' demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand.

HystTmr. Hysteresis Timer. Internally set to the hysteresis time before a new device is scheduled to start or stop.

HystTim. Hysteresis Time. The tine in seconds from which the demand must remain above the threshold value for a new device to be scheduled to start or stop.

OP.

- **A_Run to F_Run.** TRUE runs the specified device.

Status.

- **A_Runng to F_Runng.** TRUE indicates that the specified device state is 'Running'.
- **A_Stoppd to F_Stoppd.** TRUE indicates that the specified device state is 'Stopped'.

HystTmng. TRUE indicates that the demand is being tested against the threshold value by the hysteresis timer.

- **StartReq.** The demand is such that a start is required.
- **StopReq.** The demand is such that a stop is required.
- **Bit15.** Optional status bit. Associated Boolean data with the control module.

Options. This bitfield allows inputs to control the operation of the motor.

- **AfrcLead to FfrcLead.** TRUE forces the specified device to assume 'Lead' service.
- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **FrcNewLd.** TRUE forces a new lead device to start.
- **CyclLead.** Cycle the current service so that the next device assumes 'lead' service.
- **LdMstRun.** TRUE selects the lead devices to always run i.e. the lead devices are never stopped regardless of the demand.
- **Bit14-Bit15.** Optional option bits. Associates additional Boolean data with the control module.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when control module is available for automatic control, i.e. in automatic mode, at least one device not indicating an automatic discrepancy and availability options not prohibitive.

Claims. Used to request sole control of the control module. The control module resource may be controlled by up to eight sequences or other control module users.

Acks. Acknowledgements. Indicates acceptance of a claim.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the control module. Allows control modules of this type to be represented by different supervisory computer faceplates.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the control module.

Implementation notes

Note	If DmdSchdl control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 3324 bytes and by 174 bytes for each instance of the control module.

CHAPTER 7   LOGIC APPLICATION BLOCKS

The LOGIC category of Application Function Block Templates provides the control strategy with functions for logical operations.

Intentionally left blank

VLV3WAY: 16-BIT AND BLOCK

OR16: 16-BIT OR BLOCK

Block function

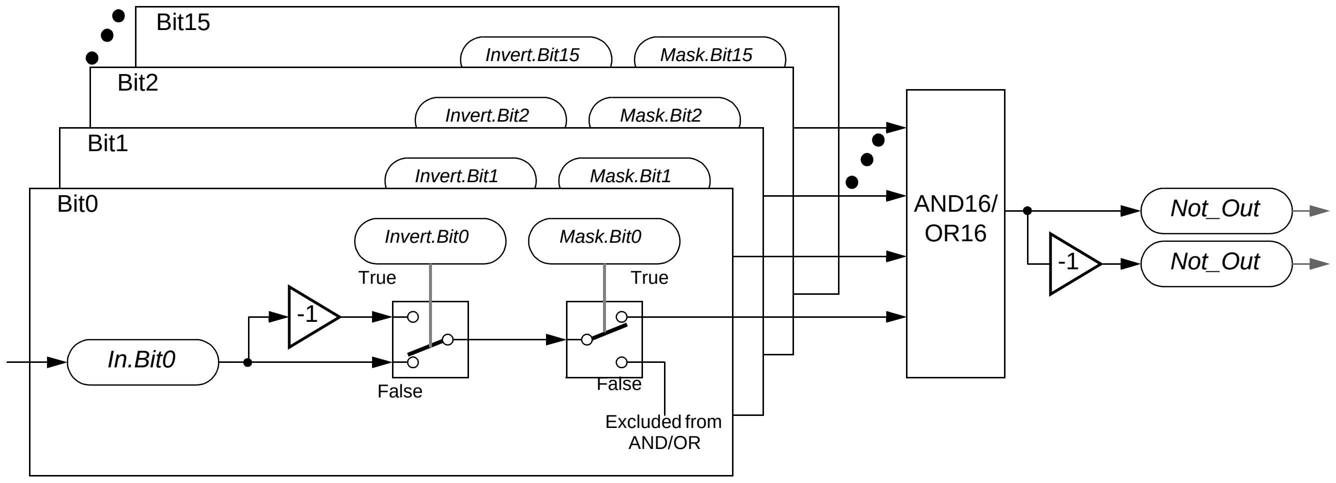


Figure 7-1: Block schematic

Please refer to the schematic in Figure AND16/OR16. This shows one of the 16 block input bits in detail. The diagram shows that each input bit has a dedicated Invert command and an additional Mask command to include, or exclude a channel from the block logic.

The 16 bit subfield input bits permit bussed wiring, e.g. between DG_CONN, Action style block types and the Digital Module (MOD_DI_UIO, and MOD_DO_UIO) block types.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
In	Individual input bits	ABCD hex	▶□▶
Bit0 to Bit15		T/F	1 2 4 8 D to A
Invert	Individual Invert for each input bit	ABCD hex	▶□▶
Bit0 to Bit15		T/F	1 2 4 8 D to A
Mask	Individual inclusion command for each input bit	ABCD hex	▶□▶
Bit0 to Bit15		T/F	1 2 4 8 D to A
Alarms		ABCD hex	□▶ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	.sto file evaluation failure in block's internal logic	T/F	
Out	Follows OUT	T/F	
NOT_OUT	Follows NOT_OUT	T/F	
Combined	OR-ing of all alarm bits	T/F	
Out	Output	T/F	□▶ 📖
Not_Out	Complementary output	Secs	□▶ 📖

Table 7-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Method. (Template/Block/Native). An enumeration indicating the origin of the update routine for the block instance. **Template** shows that the update routine is derived from an internal template, **Block** shows it is derived from the Block itself, whereas **Native** shows that it uses the (native) 'C' update routine.

Note This is derived from the LIN database when the block is activated.

In. 16-bit Block input.

Invert. Invert 16-bit Block input. TRUE, inverts the corresponding *In.Bitn* field.

Mask. Include Mask for each of the 16 input bits. This determines which bits are included in the resulting *Out*, e.g. if *Mask* = >FF00, *Out* includes *In.bit8* to *In.bit15* only, see table below.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data, or Network failure.
- **EvalFail.** Embedded .sto file evaluation failure. It is normally recommended that this alarm be enabled to a alarm priority 6 or more.
- **Out.** This is derived from *Out*. TRUE, if *Out* is set TRUE.
- **Not_Out.** This is derived from *Not_Out*. TRUE, if *Not_Out* is set TRUE.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Out. Output from Boolean operation, see table below.

Not_Out. Equal to logical NOT Out. Complementary form of the block output, see table below.

Note The table below shows an example of the resulting outputs, *Out* and *Not_Out*, derived using the *Invert* and *Mask* subfields.

Block	In	Invert	Mask	Out	Not_Out
AND16	>0004	>FFFF	>0000	FALSE	TRUE
	>0004	>FFFF	>FF00	TRUE	FALSE
	>0004	>0000	>FFFF	FALSE	TRUE
	>0003	>FFFF	>FF00	TRUE	FALSE
	>0003	>0001	>0001	FALSE	TRUE
OR16	>0004	>FFFF	>0000	FALSE	TRUE
	>0004	>FFFF	>FF00	TRUE	FALSE
	>0004	>0000	>FFFF	TRUE	FALSE
	>0003	>FFFF	>FF00	TRUE	FALSE
	>0003	>0001	>0001	FALSE	TRUE

Table 7-2: Block truth table example

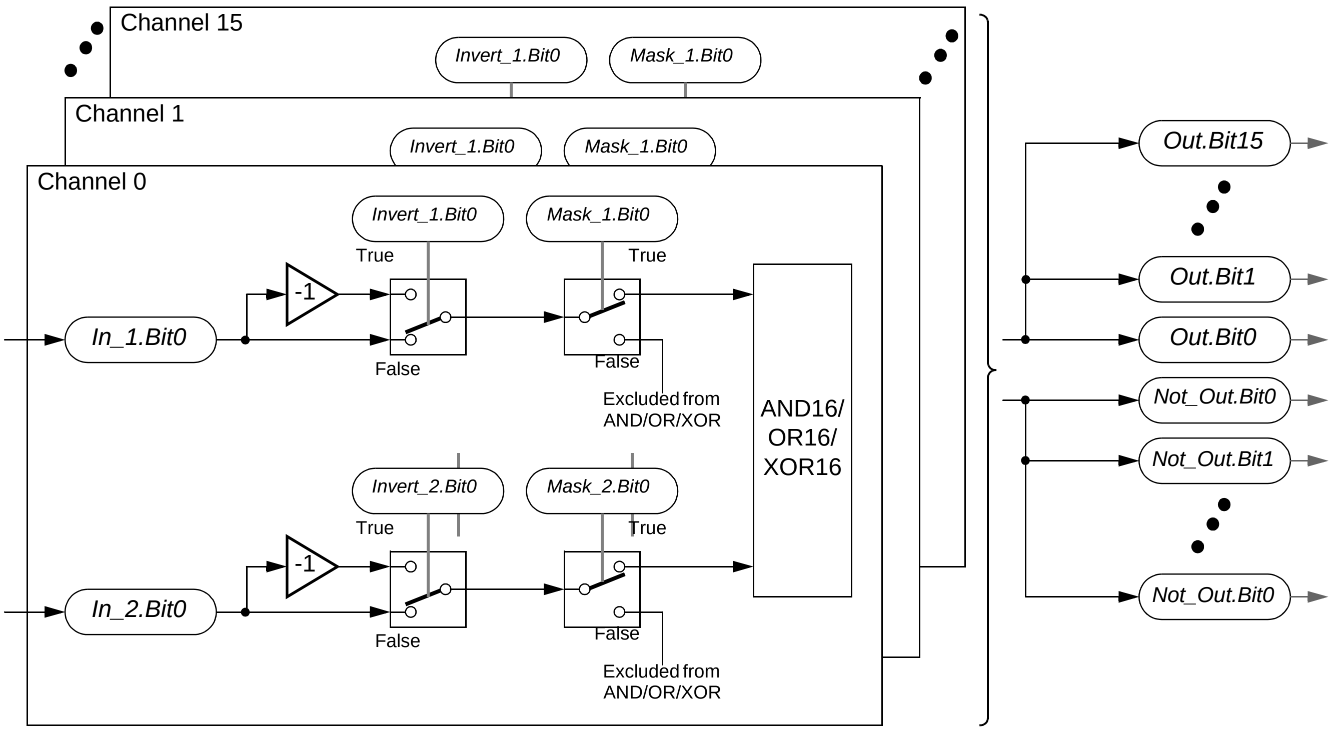
Intentionally left blank

VLV3WAY: 16-BIT AND BLOCK

BITWISE_OR16:16-BIT OR BLOCK

BITWISE_XOR16:16-BIT XOR BLOCK

Block function



Note	The BITWISE_AND16 block supports a single Mask field only.

Figure 7-1: Block schematic

Please refer to the schematic in Figure BITWISE_AND16/BITWISE_OR16/BITWISE_XOR16. This shows one of the block input channels in detail. The diagram shows that a single input channel has two bits, each with a dedicated Invert command and an additional Mask command, if applicable, to include, or exclude a channel from the block logic.

The 16 bit subfield input bits permit bussed wiring, e.g. between DG_CONN, Action style block types and the Digital Module (MOD_DI_UIO, and MOD_DO_UIO) block types.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
In_1	First set of individual input bits	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
Invert_1	Individual Invert for each input bit in the first set	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
Mask_1	Individual inclusion command for each input bit in the first set	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
In_2	Second set of individual input bits	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
Invert_2	Individual Invert for each input bit in the second set	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
Mask_2	Individual inclusion command for each input bit in the second set	ABCD hex	➧☐➧
Bit0 to Bit15		T/F	1 2 4 8 D to A
Alarms		ABCD hex	☐➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	.sto file evaluation failure in block's internal logic	T/F	
Out	Follows OUT	T/F	
NOT_OUT	Follows NOT_OUT	T/F	
Combined	OR-ing of all alarm bits	T/F	
Out	Output	T/F	☐➧ 📖
Not_Out	Complementary output	Secs	☐➧ 📖

Table 7-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Method. (Template/Block/Native). An enumeration indicating the origin of the update routine for the block instance. **Template** shows that the update routine is derived from an internal template, **Block** shows it is derived from the Block itself, whereas **Native** shows that it uses the (native) 'C' update routine.

Note This is derived from the LIN database when the block is activated.

In_1, In_2. Two sets of 16-bit Block inputs.

Invert_1, Invert_2. Invert 16-bit Block inputs. TRUE, inverts the corresponding *In.Bitn* field.

Mask_1, Mask_2. Include Mask for each of the 16 input bit per channel. This determines which bits are included in the resulting *Out*, e.g. if *Mask* = >FF00, *Out* includes *In.bit8* to *In.bit15* only, see table below.

Note The BITWISE_AND16 block has only one Mask field associated with both *In_1* and *In_2* input bits because by the nature of this block two masks would be simply ANDed together.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Embedded .sto file evaluation failure. It is normally recommended that this alarm be enabled to a alarm priority 6 or more.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Out. 16 Output bits from Boolean operation, see table below.

Not_Out. Equal to bitwise NOT Out. Complementary form of the block output, see table below.

Note The table below shows an example of the resulting outputs, *Out* and *Not_Out*, derived using the *Invert_1* and *Invert_2* and *Mask* subfields.

Block	In_1	Invert_1	Mask_1	In_2	Invert_2	Mask_2	Out	Not_Out
BITWISE_AND16	>0004	>FFFF	>0000	>0004	>FFFF	N/A	>0000	>FFFF
	>0004	>FFFF	>FF00	>0004	>FFFF	N/A	>FF00	>00FF
	>0004	>0000	>FFFF	>0004	>0000	N/A	>0004	>FFFB
	>0003	>FFFF	>FF00	>0003	>FFFF	N/A	>FF00	>00FF
	>0003	>0001	>0001	>0003	>0001	N/A	>0000	>FFFF
BITWISE_OR16	>0004	>FFFF	>0000	>0004	>FFFF	>0000	>0000	>FFFF
	>0004	>FFFF	>FF00	>0004	>FFFF	>FF00	>FF00	>00FF
	>0004	>0000	>FFFF	>0004	>0000	>FFFF	>0004	FALSE
	>0003	>FFFF	>FF00	>0003	>FFFF	>FF00	>0000	>FFFF
	>0003	>0001	>0001	>0003	>0001	>0001	>0000	>FFFF
BITWISE_XOR16	>0004	>FFFF	>0000	>0004	>FFFF	>0000	>0000	>FFFF
	>0004	>FFFF	>FF00	>0004	>FFFF	>FF00	>0000	>FFFF
	>0004	>0000	>FFFF	>0004	>0000	>FFFF	TRUE	>FFFB
	>0003	>FFFF	>FF00	>0003	>FFFF	>FF00	>0004	>FFFF
	>0003	>0001	>0001	>0003	>0001	>0001	>0000	>FFFF

Table 7-2: Block truth table example

Intentionally left blank

VLV3WAY: DATE AND TIME COMPARE BLOCK

Block function

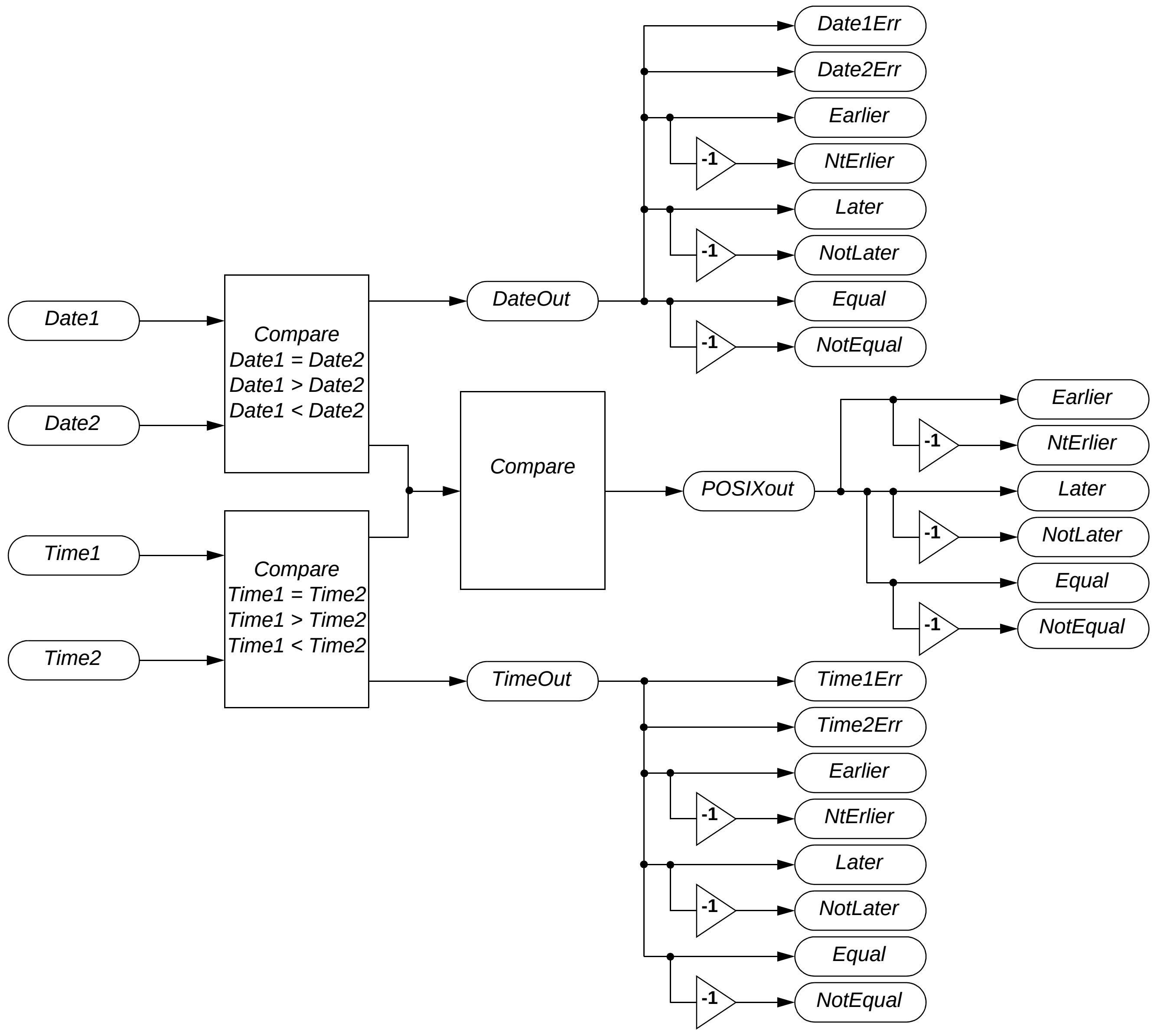


Figure 7-1: Block schematic

Please refer to Figure DT_COMPARE-1. This block is similar to the analogue Compare block as it provides a date and time compare function with outputs for earlier, equal, and later with complementary inverted outputs. It offers two sets of date and time fields, not linked to the system clock, so that arbitrary dates and times can be compared as required.

A comparison is made Time1/Date1 relative to Time2/Date2, e.g. if Time1 is 09:12:00 and Time2 is 14:45:12, the resulting Time comparison would be ‘Earlier’. Each set individually provides the ability to convert between LIN string forms of date/time and integer forms of the date/time and vice-versa.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
SourceT1	Source of Time set 1	Menu	
Time1	LIN time string for Time set 1	Time	
Time1Raw	Internal LIN form of time for Time set 1	ms	📖
Hours1	Quantity of complete hours of Time set 1	Integer	
Mins1	Quantity of complete minutes of Time set 1	Integer	
Secs1	Quantity of complete seconds of Time set 1	Integer	
SourceT2	Source of Time set 2	Menu	
Time2	LIN time string for Time set 2	Time	
Time2Raw	Internal LIN form of time for Time set 2	ms	📖
Hours2	Quantity of complete hours of Time set 2	Integer	
Mins2	Quantity of complete minutes of Time set 2	Integer	
Secs2	Quantity of complete seconds of Time set 2	Integer	
TimeOptn		AB hex	➧☐➧
AplyHMS1	Converts valid Hours1, Mins1 and Secs1 to Time1	T/F	
AplyHMS2	Converts valid Hours2, Mins2 and Secs2 to Time2	T/F	
SetAlmTa	Linked to *Alarms.Time*	T/F	
SetAlmTb	Linked to *Alarms.Time*	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
TimeOut	Time Compare status	(AB)CD hex	☐➧ 📖
Time1Err	Invalid data is detected in Time set 1	T/F	1 D
Time2Err	Invalid data is detected in Time set 2	T/F	2 D
Earlier	Time set 1 is earlier than Time set 2	T/F	4 D
Equal	Time set 1 is equal to Time set 2	T/F	8 D
Later	Time set 1 is later than Time set 2	T/F	1 C
NtErlier	Time set 1 is not earlier than Time set 2	T/F	2 C
NotEqual	Time set 1 is not equal to Time set 2	T/F	4 C
NotLater	Time set 1 is not later than Time set 2	Integer	8 C
POSIX_1	For future use - POSIX Date/Time set 1	Integer	➧☐➧
POSIX_2	For future use - POSIX Date/Time set 2	Integer	➧☐➧
Alarms		ABCD hex	☐➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	.sto file evaluation failure in block's internal logic	T/F	
Date1Err	Invalid data in Date set 1 detected	T/F	
Date2Err	Invalid data in Date set 2 detected	T/F	
Time1Err	Invalid data in Time set 1 detected	T/F	
Time2Err	Invalid data in Time set 2 detected	T/F	
Time	Set by *TimeOptn.SetAlmTa* and/or *TimeOptn.SetAlmTb*	T/F	
Date	Set by *DateOptn.SetAlmDa* and/or *DateOptn.SetAlmDb*	T/F	
DateTime	Set by *POSIXopt.SetAlDTa* and/or *POSIXopt.SetAlDTb*	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 7-1: Block parameters

Parameter	Function	Units	Status
SourceD1	Source of Date set 1	T/F	➧☐
Date1	LIN date string for Date set 1	Time	
Date1Raw	Internal LIN form of date for Date set 1	ms	
Days1	Quantity of complete Days of Date set 1	Integer	
Months1	Quantity of complete minutes of Time set 1	Integer	
Years1	Quantity of complete seconds of Time set 1	Integer	
SourceD2	Source of Date set 2	Menu	➧☐
Date2	LIN date string for Date set 2	Time	
Date2Raw	Internal LIN form of date for Date set 2	ms	
Days2	Quantity of complete hours of Time set 2	Integer	
Months2	Quantity of complete minutes of Time set 2	Integer	
Years2	Quantity of complete seconds of Time set 2	Integer	
DateOptn	Date conversion and alarm options	AB hex	➧☐➧
AplyDMY1	Converts valid Days1, Months1 and Years1 to Date1	T/F	
AplyDMY2	Converts valid Days2, Months2 and Years2 to Time2	T/F	
SetAlmDa	Linked to *Alarms.Date*	T/F	
SetAlmDb	Linked to *Alarms.Date*	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
DateOut	Date Compare status	(A)BCD hex	☐➧ 🕮
Date1Err	Invalid data is detected in Date set 1	T/F	1 D
Date2Err	Invalid data is detected in Date set 2	T/F	2 D
Earlier	Date set 1 is earlier than Date set 2	T/F	4 D
Equal	Date set 1 is equal to Date set 2	T/F	8 D
Later	Date set 1 is later than Date set 2	T/F	1 C
NtErlier	Date set 1 is not earlier than Date set 2	T/F	2 C
NotEqual	Date set 1 is not equal to Date set 2	T/F	4 C
NotLater	Date set 1 is not later than Date set 2	T/F	8 C
LeapYr1	Leap year indication - Internal use only	T/F	1 B
LeapYr2	Leap year indication - Internal use only	T/F	2 B
			4 B
			8 B
POSIXopt	Date/Time alarm options	AB hex	➧☐➧
SetAlDTa	Linked to *Alarms.DateTime*	T/F	
SetAlDTb	Linked to *Alarms.DateTime*	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
POSIXout	Date/Time Compare status	(A)BCD hex	☐➧ 🕮
Earlier	Date set 1 is earlier than Date set 2	T/F	
Equal	Date set 1 is equal to Date set 2	T/F	
Later	Date set 1 is later than Date set 2	T/F	
NtErlier	Date set 1 is not earlier than Date set 2	T/F	
NotEqual	Date set 1 is not equal to Date set 2	T/F	
NotLater	Date set 1 is not later than Date set 2	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 7-1: Block parameters

Block specification menu

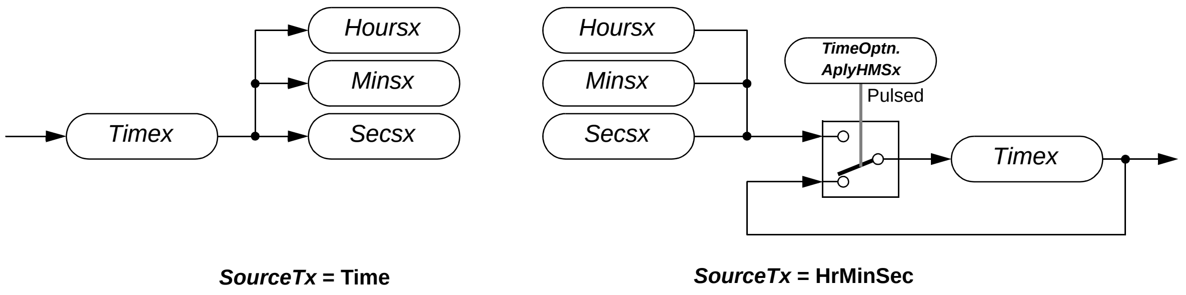
Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Method. (Template/Block/Native). Shows the origin of the Structured Text, ST, update routine for the block instance. The default **Template** shows the update routine is derived from an internal template, **Block** shows it is derived from the Block, and **Native** shows it uses a (native) ‘C’ update routine.

Note	This is derived from the LIN database when the block is activated.

SourceT1, SourceT2. (Time/HrMinSec/POSIX). Defines the update format for each time set.

When *SourceT1* or *SourceT2* show **Time**, the source is continuously derived from *Timex*, where *x* is the set number, and automatically updates *Hoursx*, *Minsx* and *Secsx* on each block execution. If *Timex* shows ??:??:??, *Status.TimeErrx* and *Alarms.TimeErrx* are set TRUE, and the *Hoursx*, *Minsx* and *Secsx* are forced to -1.



When *SourceT1* or *SourceT2* show **HrMinSec**, the source is derived on-demand from *Hoursx*, *Minsx* and *Secsx* where *x* is the set number. *TimeOptn.AplyHMSx* is used to update *Timex*. If invalid source data is detected, e.g. *Secsx* > 59, *Status.TimeErrx* and *Alarms.TimeErrx* are set TRUE, *Timex* shows ??:??:??, and *Hoursx*, *Minsx* and *Secsx* do not change.

Note	When *SourceT1* or *SourceT2* show **POSIX**, *Hoursx*, *Minsx* and *Secsx* are set to -1, and *TimeOut.TimexErr* sets TRUE. POSIX Combined Date/Time sources are not currently supported.

Time1, Time2. (HH:MM:SS/??:??:??). A LIN time string, see *SourceT1, SourceT2.*

Hours1, Hours2, Mins1, Mins2, Secs1, Secs2. Corresponds to Hours, Minutes and Seconds shown in *Timex,*where *x* is the set number, see *SourceT1, SourceT2.*

TimeOptn. Time conversion and alarm options.

- **AplyHMS1, AplyHMS2.** TRUE converts the corresponding valid *Hoursx, Minsx* and *Secsx,* where *x* is the set number, when *SourceTx* shows HrMinSec.
- **SetAlmTa, SetAlmTb.** If either or both show TRUE, *Alarms.Time* is set TRUE.

TimeOut. Time compare status.

- **Time1Err, Time2Err.** TRUE if invalid data is present in related fields when *SourceTx* shows HrMinSec, or *Timex*, when *SourceTx* shows Time. When set TRUE, *TimeOut.Earlier*, *TimeOut.Equal* and *TimeOut.Later* are set FALSE, and all other *TimeOut* subfields are set TRUE.

Note	Remember to consider failure modes, i.e. invalid times, relevant to the application, e.g. including the *TimeOut.TimexErr* in 'event' generation may be useful.

- **Earlier, Later, NtErlier, NtEqual, NotLater.** TRUE identifies how *Time1* and *Time2* differ, see *Time1Err, Time2Err*.
- **Equal.** TRUE when *Time1* shows a time equal to that shown in *Time2*, including secs. This will only set TRUE for one second, but may never set TRUE if the database execution time exceeds 1 second. Using the *TimeOut.Later* allows a maximum of a 1 second delay.

POSIX_1, POSIX_2. Combined Date/Time sets not currently supported.

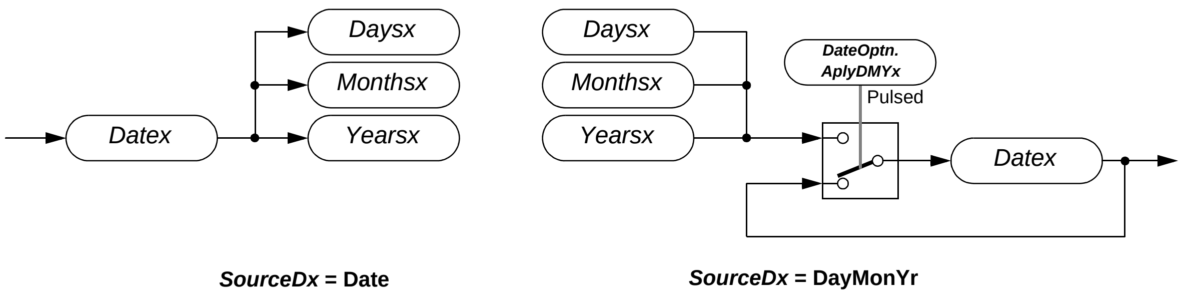
Alarms. See page 11-6 in the LIN Block Reference Manual.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Asserted, if Embedded .sto file evaluation failure, e.g. MOD operator used on non integer arguments. It is normally recommended that this alarm be enabled to a alarm priority 6 or more.
- **Date1Err, Date2Err, Time1Err, Time2Err.** Asserted, if *DateOut.Date1Err*, or *DateOut.Date2Err,* or, *TimeOut.Time1Err,* or *TimeOut.Time2Err* shows TRUE respectively.
- **Time.** Asserted, if *TimeOptn.SetAlmTa* and/or *TimeOptn.SetAlmTb* show TRUE.
- **Date.** Asserted, if *DateOptn.SetAlmDa* and/or *DateOptn.SetAlmDb* show TRUE.
- **DateTime.** Asserted, if *POSIXopt.SetAlDa* and/or *POSIXopt.SetAlDTb* show TRUE.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SourceD1, SourceD2. (Date/DayMonYr/Posix). Defines the update format for each date set.

When *SourceD1* or *SourceD2* show **Date**, the source is continuously derived from *Datex*, where *x* is the set number, and automatically updates *Daysx*, *Monthsx* and *Yearsx* on each block execution. If *Datex* shows ??/??/??, *Status.DateErrx* and *Alarms.DateErrx* are set TRUE, and the *Daysx*, *Monthsx* and *Yearsx* are forced to -1.

When *SourceD1* or *SourceD2* show **DayMonYr**, the source is derived on-demand from *Daysx*, *Monthsx* and *Yearsx*



where *x* is the set number. *DateOptn.AplyDMYx* is used to update *Datex*. If invalid source data is detected, e.g. *Monthsx* > 12, *Status.DateErrx* and *Alarms.DateErrx* are set TRUE, *Datex* shows ??/??/??, and *Daysx*, *Monthsx* and *Yearsx* do not change.

Note	When *SourceD1* or *SourceD2* show **POSIX**, *Daysx*, *Monthsx* and *Yearsx* are set to -1, and *DateOut.DatexErr* sets TRUE. POSIX Combined Date/Time sources are not currently supported.

Date1, Date2. (DD/MM/YY//??/??/??). A LIN date string, see *SourceD1, SourceD2*.

Days1, Days2, Months1, Months2, Years1, Years2. Corresponds to Days, Months and Years shown in *Datex,* where *x* is the set number, see *SourceD1, SourceD2*.

DateOptn. Date conversion and alarm options.

- **AplyDMY1, AplyDMY2.** TRUE converts the corresponding valid *Daysx, Monthsx* and *Yearsx,* where *x* is the set number, when *SourceDx* shows **DayMonYr**.
- **SetAlmDa, SetAlmDb.** If either or both show TRUE, *Alarms.Date* is set TRUE.

DateOut. Date compare status.

- **Date1Err, Date2Err.** TRUE when invalid data is present in corresponding *Daysx*, *Monthsx* and *Yearsx*, when *SourceDx* shows DayMonYr, or *Datex*, when *SourceDx* shows Date. When set TRUE, *DateOut.Earlier*, *DateOut.Equal* and *DateOut.Later* are set FALSE, and the *DateOut.NtErlier*, *DateOut.NotEqual* and *DateOut.NotLater* are set TRUE.
- **Earlier, Equal, Later, NtErlier, NtEqual, NotLater.** TRUE identifies how *Date1* and *Date2* differ, see *Date1Err, date2Err*.

Note Remember to consider failure modes, i.e. invalid dates, relevant to the application, e.g. including the ***DateOut.DatexErr*** in 'event' generation may be useful.

- **LeapYr1, LeapYr2.** Used internally to validate leap year data in *Yearsx* when *SourceDx* shows DayMonYr. Will always show FALSE.

POSIXopt. POSIX Combined Date/Time conversion and alarm options.

- **SetAlDTa, SetAlDTb.** TRUE sets *Alarms.DateTime* TRUE

POSIXout. POSIX combined Date/Time compare status.

Note Remember to consider failure modes, i.e. invalid dates and times, relevant to the application, e.g. including the ***TimeOut.TimexErr*** or ***DateOut.DatexErr*** in 'event' generation may be useful.

- **Earlier, Equal, Later, NtErlier, NtEqual, NotLater.** TRUE identifies how *Date1/Time1* differs from *Date2/Time2*.

CHAPTER 8 MATHS APPLICATION BLOCKS

The MATHS category of Application Function Block Templates provides the control strategy with functions for mathematical calculations.

Intentionally left blank

VLV3WAY: ACTUI8I8 ACTION BLOCK

Block function

The ActUI8I8 block allows a user defined 'Action', separately created in *LINtools Engineering Studio*, to be run in the strategy. Please refer to the *LINtools Engineering Studio help* (Part no. RM 263 001 U055) for detailed information on configuring Structured Text actions.

Note	Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999) for detailed information on configuring structured text actions.

The block provides eight unsigned long integer variables, eight signed long integer variables and 1 (16 bit) 'word' field that can be used by the action as either inputs or outputs, and fields specifying the action's qualifier and associated qualifier time (where appropriate). A digital input provides the means to enable the action, and the elapsed time since the action was enabled is available as an output.

The ActUI8I8 block is very similar to the ACTION block, described in the *LIN Block Reference Manual* and has several identical parameters.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units			Status
ActName	8-character (max.) action name	String			
FileName	8-character (max.) filename	String			
T	Elapsed time since EN parameter TRUE				□➧ 📖
EN	Action enabling input	T/F			➧□
UI0 to UI7	Unsigned long integer variables	Eng			➧□➧
D	Digital variable bits	ABCD hex			➧□➧
Bit0	Digital 0	T/F	1	D	
Bit1	Digital 1	T/F	2		
Bit2	Digital 2	T/F	4		
Bit3	Digital 3	T/F	8		
Bit4	Digital 4	T/F	1	C	
Bit5	Digital 5	T/F	2		
Bit6	Digital 6	T/F	4		
Bit7	Digital 7	T/F	8		
Bit8	Digital 8	T/F	1	B	
Bit9	Digital 9	T/F	2		
Bit10	Digital 10	T/F	4		
Bit11	Digital 11	T/F	8		
Bit12	Digital 12	T/F	1	A	
Bit13	Digital 13	T/F	2		
Bit14	Digital 14	T/F	4		
Bit15	Digital 15	T/F	8		

Table 8-1: Block parameters

Parameter	Function	Units	Status
Alarms		ABCD hex	
Software	Block RAM data sumcheck error/network failure	T/F	
NoAction	Unable to find named action	T/F	
BadActn	Runtime evaluation error	T/F	
Combined	OR-ing of all alarm bits	T/F	
Qual	Action qualifier	Enum	
QualTime	Qualifier time		
TimeBase	Time units for T and QualTime	Enum	
I0 to I7	Signed long integer variables	Eng	

Table 8-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

ActName. Specifies the name of an action contained in a file called *FileName*.STO; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .STO file.

FileName. Specifies the root name (with extension .STO) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename. Note that the .STO file must reside on the E: drive of the instrument.

T. Time elapsed since the action enable parameter (*EN*) became TRUE, in units specified by the *TimeBase* parameter. The timer controlling *T* stops running when *EN* goes FALSE, and restarts from zero when *EN* is set to TRUE again. The block uses *T* to time the starting and stopping of *L*-, *D*-, and *E*-qualified actions by comparing its current value with *QualTime*.

EN. Action enabling input. Setting *EN* TRUE zeroes and starts the *T* timer, and setting it FALSE stops the *T* timer, which holds its value until restarted. The state of *EN* controls the running of the action, as shown in Block parameter table.

Note	A FALSE-to-TRUE transition on *EN* is parallel to the activation of a step in a sequence, and a TRUE-to-FALSE transition is parallel to the step’s deactivation.

UI0 to UI7. Eight unsigned long integer variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the control strategy.

D. Bitfields containing 16 digital variables, *Bit0* to *Bit15*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **NoAction.** Asserted if the block cannot find the action named in *ActName*.
- **BadActn.** Asserted if an evaluation error occurs during the running of the action.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Qual. (P(Initial)/N(Normal)/L(Limited)/D(Delayed)/E(Event)/F(Final)) The action qualifier. *Qual* specifies when the action is to run during the time that the enabling input *EN* is TRUE, as shown in the figure below.

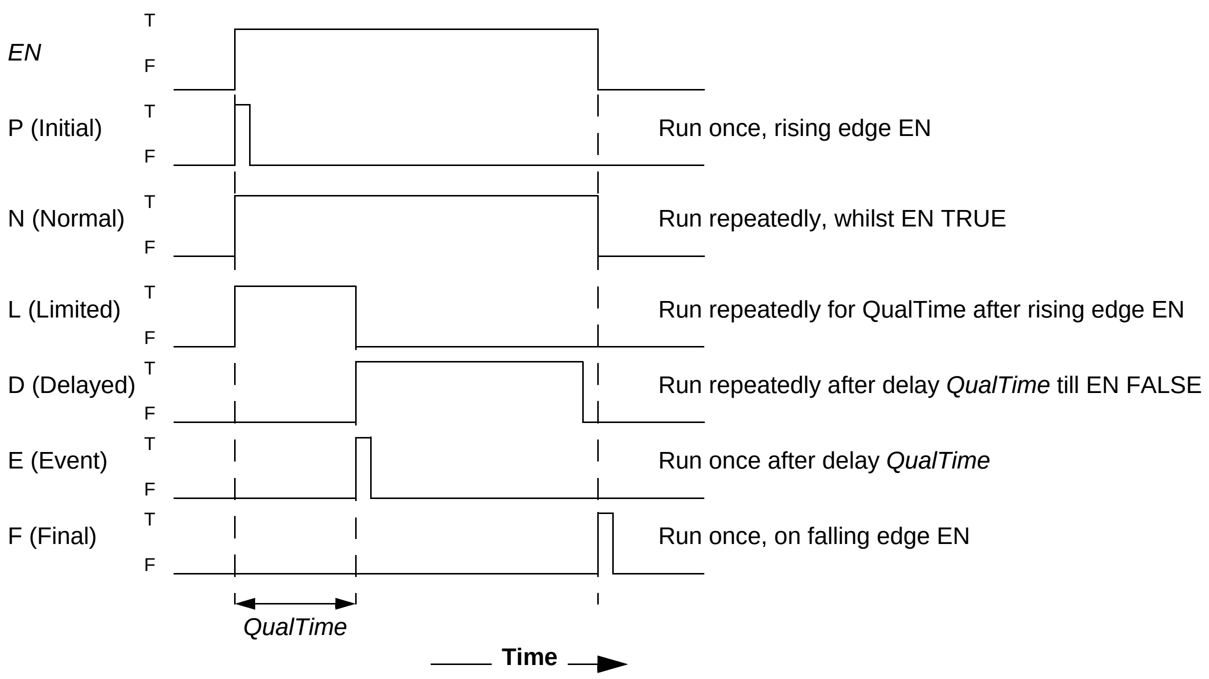

EN
P (Initial)
N (Normal)
L (Limited)
D (Delayed)
E (Event)
F (Final)
T
F
Run once, rising edge EN
Run repeatedly, whilst EN TRUE
Run repeatedly for QualTime after rising edge EN
Run repeatedly after delay QualTime till EN FALSE
Run once after delay QualTime
Run once, on falling edge EN
QualTime
Time

Figure 8-1: Operation of action qualifiers - the Qual parameter

QualTime. Time value associated with the *L*, *D*, and *E* action qualifiers, with units specified by *TimeBase*. *QualTime* specifies a running time or delay for the action, and operates during the period that the *EN* parameter is TRUE, as shown in the figure above.

TimeBase. (Secs/Mins/Hours) Specifies time units for the *T* and *QualTime* parameters.

I0 to I7. Eight signed long integer variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Implementation notes

If ActUI8I8 custom LIN blocks are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 816 bytes and by 170 bytes for each instance of the control module.

When creating action code for the ActUI8I8 block using ST operators on the 32-bit integer variables, accuracy (resolution), but not magnitude can be lost.

The unsigned long integers are 32 bits giving a range of 0 to 4294967295, which is 9½ digits of resolution.

The signed long integers are 32 bits giving a range of –2147483648 to 2147483647, which is 9½ digits of resolution.

The floating point number format used is the LIN database is IEEE single precision floating point format giving a range of approximately ±1E±36 but with only 6½ digits of resolution.

This loss of accuracy when manipulating the signed and unsigned long integers occurs when operators such as * and / are executed. When the data is read from the block to be used in operations that may have a non-integer result, the values are converted internally to single precision floating point format and then back to integers when writing into the block. Therefore, a test on the full numeric range of the used variables should be carried out to check that the action code is behaving as required.

Intentionally left blank

VLV3WAY: ACT15A3W ACTION BLOCK

Block function

The Act15A3W block type allows a user defined ‘Action’, separately created in the *LINtools Engineering Studio* (Part No. RM 263 001 U550), to be run in the strategy. Please refer to the *LINtools Engineering Studio help* (Part no. RM 263 001 U055) for detailed information on configuring Structured Text (ST) actions.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999) for detailed information on configuring structured text actions.

The block provides 15 floating-point variables, 2 (16 bit) ‘word’ fields and 2 (8 bit) ‘byte’ fields that can be either input or output used by the action, and fields specifying the action’s qualifier and associated qualifier time (where appropriate). A digital input provides the means to enable the action, and the elapsed time since the action was enabled is available as an output.

The Act15A3W block is very similar to the ACTION block, described in the *LIN Block Reference Manual* (Part no. HA 082 375 U003) and has several identical parameters.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ActName	8-character (max.) action name	String	
FileName	8-character (max.) filename	String	
T	Elapsed time since EN parameter TRUE		□➔ 📖
EN	Action enabling input	T/F	➔□
A0 to A7	Floating point variables	Eng	➔□➔
Word1 & Word2	Digital variable bits	ABCD hex	➔□➔
Bit0	Digital 0	T/F	1
Bit1	Digital 1	T/F	2 D
Bit2	Digital 2	T/F	4
Bit3	Digital 3	T/F	8
Bit4	Digital 4	T/F	1
Bit5	Digital 5	T/F	2 C
Bit6	Digital 6	T/F	4
Bit7	Digital 7	T/F	8
Bit8	Digital 8	T/F	1
Bit9	Digital 9	T/F	2 B
Bit10	Digital 10	T/F	4
Bit11	Digital 11	T/F	8
Bit12	Digital 12	T/F	1
Bit13	Digital 13	T/F	2 A
Bit14	Digital 14	T/F	4
Bit15	Digital 15	T/F	8

Table 8-1: Block parameters

Parameter	Function	Units			Status
Alarms		ABCD hex			⇨ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F			
NoAction	Unable to find named action	T/F			
BadActn	Runtime evaluation error	T/F			
Combined	OR-ing of all alarm bits	T/F			
Qual	Action qualifier	Enum			
QualTime	Qualifier time				
TimeBase	Time units for T and QualTime	Enum			
A8 to A14	Floating point variables	Eng			⇄
Byte0 & Byte1	Digital variable bits	(AB)CD hex			⇄
Bit0	Digital 0	T/F	1	D	
Bit1	Digital 1	T/F	2	D	
Bit2	Digital 2	T/F	4	D	
Bit3	Digital 3	T/F	8	D	
Bit4	Digital 4	T/F	1	C	
Bit5	Digital 5	T/F	2	C	
Bit6	Digital 6	T/F	4	C	
Bit7	Digital 7	T/F	8	C	

Table 8-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

ActName. Specifies the name of an action contained in a file called *FileName*.STO; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .STO file.

FileName. Specifies the root name (with extension .STO) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename. Note that the .STO file must reside on the E: drive of the instrument.

T. Time elapsed since the action enable parameter (*EN*) became TRUE, in units specified by the *TimeBase* parameter. The timer controlling *T* stops running when *EN* goes FALSE, and restarts from zero when *EN* is set to TRUE again. The block uses *T* to time the starting and stopping of *L*-, *D*-, and *E*-qualified actions by comparing its current value with *QualTime*.

EN. Action enabling input. Setting *EN* TRUE zeroes and starts the *T* timer, and setting it FALSE stops the *T* timer, which holds its value until restarted. The state of *EN* controls the running of the action, as shown in Block parameter table.

Note A FALSE-to-TRUE transition on *EN* is parallel to the activation of a step in a sequence, and a TRUE-to-FALSE transition is parallel to the step’s de-activation

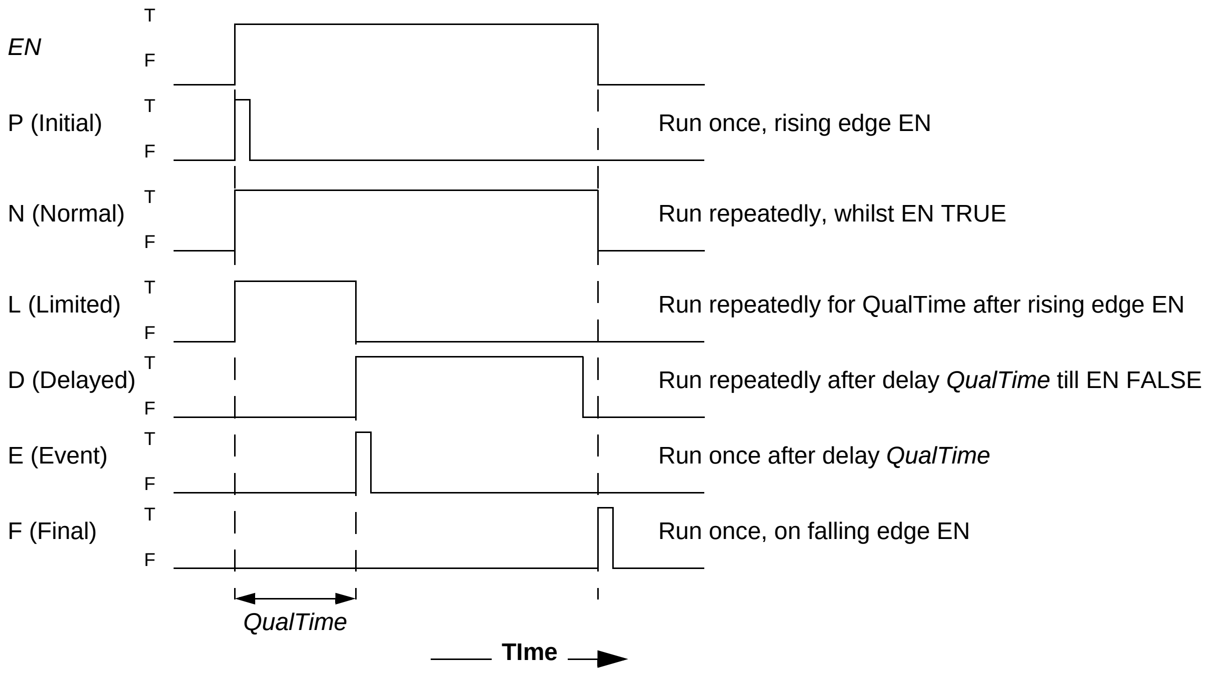
A0 to A7. Eight floating-point variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Word0 to Word1. Bitfields containing 16 digital variables, *Bit0* to *Bit15*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **NoAction.** Asserted if the block cannot find the action named in *ActName*.
- **BadActn.** Asserted if an evaluation error occurs during the running of the action.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Qual. (P(Initial)/N(Normal)/L(Limited)/D(Delayed)/E(Event)/F(Final)) The action qualifier. *Qual* specifies when the action is to run during the time that the enabling input *EN* is TRUE, as shown in the figure below.


EN
T
F
P (Initial)
Run once, rising edge EN
N (Normal)
Run repeatedly, whilst EN TRUE
L (Limited)
Run repeatedly for QualTime after rising edge EN
D (Delayed)
Run repeatedly after delay QualTime till EN FALSE
E (Event)
Run once after delay QualTime
F (Final)
Run once, on falling edge EN
QualTime
TIme

QualTime. Time value associated with the *L*, *D*, and *E* action qualifiers, with units specified by *TimeBase*. *QualTime* specifies a running time or delay for the action, and operates during the period that the *EN* parameter is TRUE, as shown in the figure above.

TimeBase. (Secs/Mins/Hours) Specifies time units for the *T* and *QualTime* parameters.

A8 to A14. Seven floating-point variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Byte0 to Byte1. Two bitfields containing 8 digital variables, Bit0 to Bit7 that can be incoporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Implementation notes

If Act15a3w custom LIN blocks are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 922 bytes and by 170 bytes for each instance of the control module.

Intentionally left blank

VLV3WAY: ACTION BLOCK WITH MULTIPLE WORD FIELDS

Block function

The WORD_ACT block allows a sequence-type ‘Action’, separately created in the *LINtools Engineering Studio*, to be run in the control strategy. Please refer to the *LINtools Engineering Studio help* (Part no. RM 263 001 U055) for detailed information on configuring Structured Text (ST) actions.

The block provides sets of digitals that can be used by the action, and fields specifying the action’s qualifier and associated qualifier time (where appropriate). A digital input provides the means to enable the action, and the elapsed time since the action was enabled is available as an output.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units			Status
ActName	8-character (max.) action name	String			
FileName	8-character (max.) filename	String			
T	Elapsed time since EN parameter TRUE				□➧ 📖
EN	Action enabling input	T/F			➧□
Word0 to Word31	16-bit digital variables	ABCD hex			➧□➧
Bit0	Digital 0	T/F	1	D	
Bit1	Digital 1	T/F	2		
Bit2	Digital 2	T/F	4		
Bit3	Digital 3	T/F	8		
Bit4	Digital 4	T/F	1	C	
Bit5	Digital 5	T/F	2		
Bit6	Digital 6	T/F	4		
Bit7	Digital 7	T/F	8		
Bit8	Digital 8	T/F	1	B	
Bit9	Digital 9	T/F	2		
Bit10	Digital 10	T/F	4		
Bit11	Digital 11	T/F	8		
Bit12	Digital 12	T/F	1	A	
Bit13	Digital 13	T/F	2		
Bit14	Digital 14	T/F	4		
Bit15	Digital 15	T/F	8		
Byte0 & Byte1	Digital variable bits	(AB)CD hex			➧□➧
Bit0	Digital 0	T/F	1	D	
Bit1	Digital 1	T/F	2		
Bit2	Digital 2	T/F	4		
Bit3	Digital 3	T/F	8		
Bit4	Digital 4	T/F	1	C	
Bit5	Digital 5	T/F	2		
Bit6	Digital 6	T/F	4		
Bit7	Digital 7	T/F	8		

Table 8-1: Block parameters

Parameter	Function	Units	Status
Alarms		ABCD hex	➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
NoAction	Unable to find named action	T/F	
BadActn	Runtime evaluation error	T/F	
Combined	OR-ing of all alarm bits	T/F	
Qual	Action qualifier	Enum	
QualTime	Qualifier time		
TimeBase	Time units for T and QualTime	Enum	

Table 8-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

ActName. Specifies the name of an action contained in a file called *FileName*.STO; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .STO file.

FileName. Specifies the root name (with extension .STO) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename. Note that the .STO file must reside on the E: drive of the instrument.

T. Time elapsed since the action enable parameter (*EN*) became TRUE, in units specified by the *TimeBase* parameter. The timer controlling *T* stops running when *EN* goes FALSE, and restarts from zero when *EN* is set to TRUE again. The block uses *T* to time the starting and stopping of *L*-, *D*-, and *E*-qualified actions by comparing its current value with *QualTime*.

EN. Action enabling input. Setting *EN* TRUE zeroes and starts the *T* timer, and setting it FALSE stops the *T* timer, which holds its value until restarted. The state of *EN* controls the running of the action, as shown in *see Figure 8-1:*.

Note A FALSE-to-TRUE transition on EN is parallel to the activation of a step in a sequence, and a TRUE-to-FALSE transition is parallel to the step’s de-activation.

Word0 to Word31. 32 (thirty-two) Word fields containing 16-bit digital variables, *Bit0* to *Bit15*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **NoAction.** Asserted if the block cannot find the action named in *ActName*.
- **BadActn.** Asserted if an evaluation error occurs during the running of the action.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Byte0 and Byte1. Two Byte fields containing 8-bit digital variables, *Bit0* to *Bit7*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the strategy.

Qual. (P(Initial)/N(Normal)/L(Limited)/D(Delayed)/E(Event)/F(Final)). The action qualifier. *Qual* specifies when the action is to run during the time that the enabling input *EN* is TRUE, as shown in *see Figure 8-1:*.

QualTime. Time value associated with the *L*, *D*, and *E* action qualifiers, with units specified by *TimeBase*. *QualTime* specifies a running time or delay for the action, and operates during the period that the *EN* parameter is TRUE, as shown in *see Figure 8-1:*.

TimeBase. (Secs/Mins/Hours) Specifies time units for the *T* and *QualTime* parameters.

APPLICATION NOTES

This block is designed for use if multiple digital bit sets that must be processed using an identical method. It offers 32 Word (16-bit) and 2 Byte (8-bit) fields that provide 528 digital input and/or output bits. Individually configuring these bits would rapidly exceed the maximum Action code space.

The following Structured Text examples illustrate the use of Logic operators for performing bitwise functions.

Example 1

Structured Text providing bitwise invert on 16 input bits.

```
(*
Provide a bitwise invert on 16 input bits
Word0     = 16 INPUT BITS
Word16    = 16 INVERTED INPUT BITS
*)

Word16 := Word0 XOR 16#FFFF;
```

Example 2

Structured Text providing 16 individual start burner commands to 16 burners.

```
(*
Provide a start burner command for each of the 16 burners. Each single burner
is started [Start = True] if the specific burner enabled input is True and the
corresponding burner heat demand is above 10%
Word10    = 16 BURNER ENABLES [TRUE = ENABLED]
Word11    = 16 BURNERS HEAT DEMANDS >10% [TRUE = >10%]
Word16    = 16 BURNER START COMMANDS [TRUE = START]
*)

Word16 := Word10 AND Word11;
```

Function block memory use can be reduced in comparison to the use of multiple discrete Logic blocks, e.g. AND4, OR4, etc., by configuring Actions using the principles illustrated in the examples. When working with bit sets and bitwise processing, subfield bussed wiring can provide a fast and efficient way of making connections in the database.

Function	Word*n*	Word*n*	Word*n* (Result)
INVERT (Xor)	>0000	-	>FFFF
	>0004	-	>FFFB
	>3000	-	>CFFF
	>D3A7	-	>2C58
AND	>0000	>0000	>0000
	>0FFF	>FFF0	>0FF0
	>D3A7	>2C58	>0000
	>3000	>AFFF	>2000
	>FFFF	>03B2	>03B2

Table 8-2: Block truth table example

Implementation notes

If WORD_ACT custom LIN blocks are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 1284 bytes and by 170 bytes for each instance of the Action block.

CHAPTER 9 SIMPLE VARIABLE APPLICATION BLOCKS

The SIMPLE VARIABLE category of Application Function Block Templates provides the control strategy with functions for defining specific type fields for Structured Text (ST), or Ladder Diagram actions in a Sequence.

Intentionally left blank

VLV3WAY: BOOLEANS VARIABLE BLOCK

Block function

The BOOLEANS block supplies twenty Boolean type fields that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➡□➡
Alarms		ABCD hex	□➡ 🕮 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single boolean values for any purpose. Boolean values are single-bit, TRUE/FALSE (1 or 0). Fields can be wired to and from other booleans, bits of subfields and from Alarms.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the number ‘1’, sets field TRUE in LINtools, assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
sfc_con.Run:=0;         (*stop & initialise sequence*)
booleans.Val1:=1;     (*Show sequence is complete*)
```

Tip! In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Intentionally left blank

VLV3WAY: BYTES VARIABLE BLOCK

UBYTES: UNSIGNED BYTES VARIABLE BLOCK

Block function

The BYTES block and UBYTES block supply twenty Byte type fields and Unsigned Byte type fields respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧□➧
Alarms		ABCD hex	□➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single byte values for any purpose. These values are signed 8-bit values from -128 to 127 in the BYTES block and unsigned 8 bit values from 0 to 255 in the UBYTES block. Fields can be wired to and from other byte fields and ubyte fields respectively.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the number ‘100’ assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
bytes.Val1 :=100
```

Note In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

VLV3WAY: DATES VARIABLE BLOCK

Block function

The DATES block supply twenty Date type fields in the DD/MM/YY format, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note	Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	►□►
Alarms		ABCD hex	□► 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) 082 375 U003, for details of these 'header' fields.

Val1 to Val20. Single values for any purpose. These values are date values in the DD/MM/YY format.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the time '01/07/07' assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
dates.Val1 :="01/07/07"
```

Tip!	In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Intentionally left blank

VLV3WAY: INTEGER VARIABLE BLOCK

UINTEGERS: UNSIGNED INTEGER VARIABLE BLOCK

Block function

The INTEGERS block and UINTEGERS block supply twenty Integer type fields and Unsigned Integer type fields respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧□➧
Alarms		ABCD hex	□➧ 🕮 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single values for any purpose. The values range from -32768 to 32767 in the Integers block, and 0 to 65535 in the Unsigned Integers block, see *LINtools Help* for Structured Text details.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the number ‘12’ assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
integers.Val15 := 12;
rcp_line.RecipeNo := integer.Val15;
```

Note In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

VLV3WAY: LONGS VARIABLE BLOCK

ULONGS: UNSIGNED LONGS VARIABLE BLOCK

Block function

The LONG block and ULONG block supply twenty Long Integer type fields and Unsigned Long Integer type fields respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧▭➧
Alarms		ABCD hex	▭➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single values for any purpose, see *LINtools Help* for Structured Text details.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the number ‘1234567890’ assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
longs.Val15:=1234567890;
rcp_line.RecipeNo := longs.Val15;
```

Note In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

VLV3WAY: SINGLES VARIABLE BLOCK

Block function

The SINGLES block supplies twenty single-precision floating-point type fields that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➔▢➔
Alarms		ABCD hex	▢➔ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single values for any purpose. The Single-Precision floating point values are stored as 32 bits. Fields can be wired in and out, to other single-precision floating points, bits of subfields and from Alarms, see *LINtools Help* for Structured Text details.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the number ‘0.00358’ assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
singles.Val1 := 0.00358
```

Tip! In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Intentionally left blank

VLV3WAY: CHARACTER STRING VARIABLE BLOCK

Block function

The STRINGS block supplies twenty 8-character string type fields that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in *LINtools Engineering Studio* (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧☐➧
Alarms		ABCD hex	☐➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Val1 to Val20. Single values for any purpose. These are a maximum 8-character string values, see *LINtools Help* for Structured Text details.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the phrase 'Batch 7' assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
strings.Val17 := "Batch 7";
record1.Caption := strings.Val17;
```

Tip! In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Intentionally left blank

VLV3WAY: 16-BIT SUBFIELD VARIABLE BLOCK

SUBFIELD16S:8-BIT SUBFIELD VARIABLE BLOCK

Block function

The SUBFIELD16S block and SUBFIELD8S block supply twenty 16-bit subfield types and 8-bit subfield types respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧□➧
Alarms		ABCD hex	□➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Val1 to Val20. Single values for any purpose. Each value provides a simple 16-bit and 8-bit TRUE/FALSE reading respectively, see *LINtools Help* for Structured Text details.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows that various bits, or complete bytes or words, have been assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
sub16bit.Val1:=16#F31B;
sub16bit.Val2:=1;
sub16bit.Val3.bit0:=1;
sub8bit.Val1:=16#0F;
sub8bit.Val2:=2#01101011;
sub8bit.Val3.bit0:=1;
```

Note In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

VLV3WAY: TIMES VARIABLE BLOCK

Block function

The TIMES block supplies twenty Time type fields in the HH:MM:SS format, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio (Part no. RM 263 001 U550), to be run in the strategy. Any one of the field values in this block type can be connected to other fields of the same type, linked to Text Variable string configured using the User Screen Editor, and used to store the data that is mapped from a Modbus register configured using the Modbus Tools.

Note Older LIN based Instruments may be configured using the *T500/T550 LINtools software*, refer to the *T500/T550 LINtools Product Manual* (Part no. HA 082 377 U999).

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	T/F	➧☐➧
Alarms		ABCD hex	☐➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Val1 to Val20. Single values for any purpose. These values are time values in the HH:MM:SS format.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows the time ‘14:00:00’ assigned to a defined block and field, see *LINtools Help* for Structured Text details.

```
times.Val1 :="14:00:00"
```

Tip! In ST expressions, use Assign, :=, to configure a defined field with a specified value. The logical comparison functions, e.g. = (Equal), < (Less Than), > (Greater Than), <= (Less Than Or Equal To), >= (Greater Than Or Equal To), <> (Not Equal), is used to compare the result for a particular scenario.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Intentionally left blank

WIDESTR24S: 20 VARIABLES X 24 CHARACTERS BLOCK
WIDESTR128S: 4 VARIABLES X 128 CHARACTERS BLOCK
WIDESTR510: 1 VARIABLE X 510 CHARACTERS BLOCK

Block function

The WIDESTR24S block, WIDESTR128S block and WIDESTR510 block supply twenty, four and one Character type fields respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio. The field values in this block type cannot be connected to other fields.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	Characters	
Alarms		ABCD hex	➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

WIDESTR24S

Val1 to Val20. Twenty x 24 character values for any purpose. These values are sequences of 24 unicode characacters. These fields cannot currently be wired.

WIDESTR128S

Val1 to Val4. Four x 128 character values for any purpose. These values are sequences of 128 unicode characacters. These fields cannot currently be wired.

WIDESTR510

Val. One x 510 character value for any purpose. This value is a sequence of 510 unicode characacters. These fields cannot currently be wired.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows assignment of the text ‘aBcXYz’. See *LINtools Help* for Structured Text details

```
widestr24s.Val1 :=”aBcXYz”;
```

Note In ST expressions, use Assign ‘:=’ to configure a defined field with a specified value.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data or caching error.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

VLV3WAYEQ48S:20 VARIABLES X 48 BYTES BLOCK

BYTESEQ256S: 4 VARIABLES X 256 BYTES BLOCK

BYTESEQ1020: 1 VARIABLE X 1020 BYTES BLOCK

Block function

The BYTESEQ48S block, BYTESEQ256S block and BYTESEQ1020 block supply twenty, four and one Byte-Sequence type fields respectively, that can be used for any purpose, for example as temporary variables for Structured Text (ST) or Ladder Diagram Actions in a Sequence, separately created in LINtools Engineering Studio. The field values in this block type cannot be connected to other fields.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Val1 to Val20	Variable referenced by ST action, etc	Bytes	
Alarms		ABCD hex	⇥ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 9-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

BYTESEQ48S

Val1 to Val20. Twenty x 48 byte-sequence values for any purpose. These values are sequence of 48 bytes, each of which is a value between 0-255. These fields currently cannot be wired.

BYTESEQ256S

Val1 to Val4. Four x 256 byte-sequence values for any purpose. These values are sequence of 256 bytes, each of which is a value between 0-255. These fields currently cannot be wired.

BYTESEQ1020

Val. One x 1020 byte-sequence value for any purpose. This value is a sequence of 1020 bytes, each of which is a value between 0-255. These fields currently cannot be wired.

Note: It is not possible to enter non-printable characters directly from a data entry window, e.g. using LINtools. For example <carriage return><linefeed> is only permitted when encoded as $R$L.

Example

A Structured Text Action, as generated in LINtools, or as part of a User Screen, can be used to set a defined block parameter, shown below. This example shows assignment of the text 'xyz' and includes an 'STX' and 'ETX' character. See *LINtools Help* for Structured Text details.

```
byteseq48s.Val1 :=”$02xyz$03”;
```

Note: In ST expressions, use Assign ':=' to configure a defined field with a specified value.

Alarms. See section 1.4 of this document(Control Modules User Guide, HA084012) for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data or caching error.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

CHAPTER 10 TIMING APPLICATION BLOCKS

The TIMING category of Application Function Block Templates provides the control strategy with functions for timing and simple Programmer operations.

Intentionally left blank

VLV3WAY: DIGITAL DELAY EIGHT FIELDS

Block function

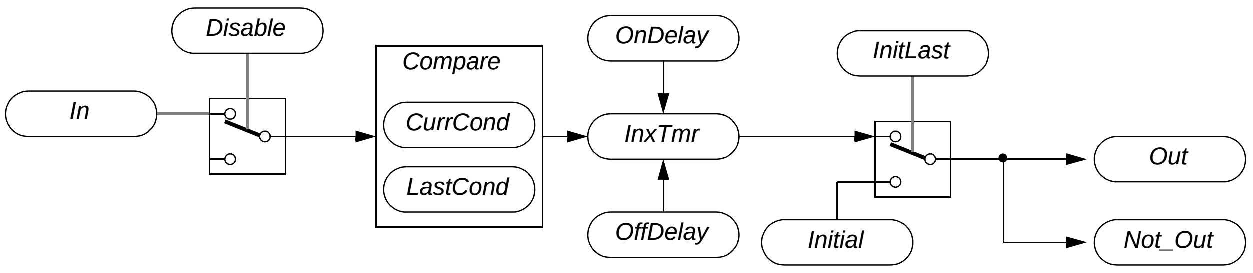


Figure 10-1: Block schematic

The DGDELAY8 block generates delayed outputs. The block allows eight digital inputs to be delayed by use of two - On and Off - separately adjustable delays. Each type of delay and therefore its associated delay value is common to all alarms. Each alarm can be masked (turned off) by a corresponding *Disable* bit. A *Not_Out* field is also available to provide a complementary delayed output.

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units		Status
Method	Reserved for future use block’s ST update routine (default = Template)	Enum		📖
In	Digital inputs from control strategy	CD hex		➧☐➧
Bit0	Digital 0	T/F	Ø	
Bit1	Digital 1	T/F	Ø	
Bit2	Digital 2	T/F	Ø	
Bit3	Digital 3	T/F	Ø	
Bit4	Digital 4	T/F	Ø	
Bit5	Digital 5	T/F	Ø	
Bit6	Digital 6	T/F	Ø	
Bit7	Digital 7	T/F	Ø	
Disable	Output disabling bits	(A)BCD hex		➧☐➧ 📖
Bit0	Digital 0	T/F	1 D	
Bit1	Digital 1	T/F	2 D	
Bit2	Digital 2	T/F	4 D	
Bit3	Digital 3	T/F	8 D	
Bit4	Digital 4	T/F	1 C	
Bit5	Digital 5	T/F	2 C	
Bit6	Digital 6	T/F	4 C	
Bit7	Digital 7	T/F	8 C	
All	Digital 8	T/F	1 B	
			2 B	
			4 B	
			8 B	
OnDelay	On timer delay setting	Secs		➧☐➧
OffDelay	Off timer delay setting	Secs		➧☐➧

Table 10-1: Block parameters

Parameter	Function	Units	Status
Initial	Initialises outputs	CD hex	
Bit0	Digital 0	T/F	
Bit1	Digital 1	T/F	
Bit2	Digital 2	T/F	
Bit3	Digital 3	T/F	
Bit4	Digital 4	T/F	
Bit5	Digital 5	T/F	
Bit6	Digital 6	T/F	
Bit7	Digital 7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Alarms		ABCD hex	
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Block evaluation error	T/F	
Combined	OR-ing of all alarm bits	T/F	
In0Tmr to In7Tmr	Alarm delay timers	Secs	
Current	Current condition flags	CD hex	
Last	Last condition flags	CD hex	
Out	Delayed outputs	CD hex	
Not_Out	Complementary delayed outputs	CD hex	
Status		(C)D hex	
NewCond	New condition flag	T/F	
Combined	OR-ing of all output bits	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Options		(C)D hex	
AckNew	Resets Status.NewCond	T/F	
InitLast	Forces Last to be the inverse of Current	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 10-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

In. Bitfield indicating the actual state of the eight field input connections.

Disable. Disabling (masking) fields. A TRUE on any input turns off the named output. A *Disable.All* bit is provided to mask all outputs collectively.

Initial. Bitfield providing an override for the outputs when *Options.InitLast* is used (e.g. wired in from <HeaderBlock>.*Status.TmpFail*) allowing the block outputs to be initialised to preset values.

Alarms.

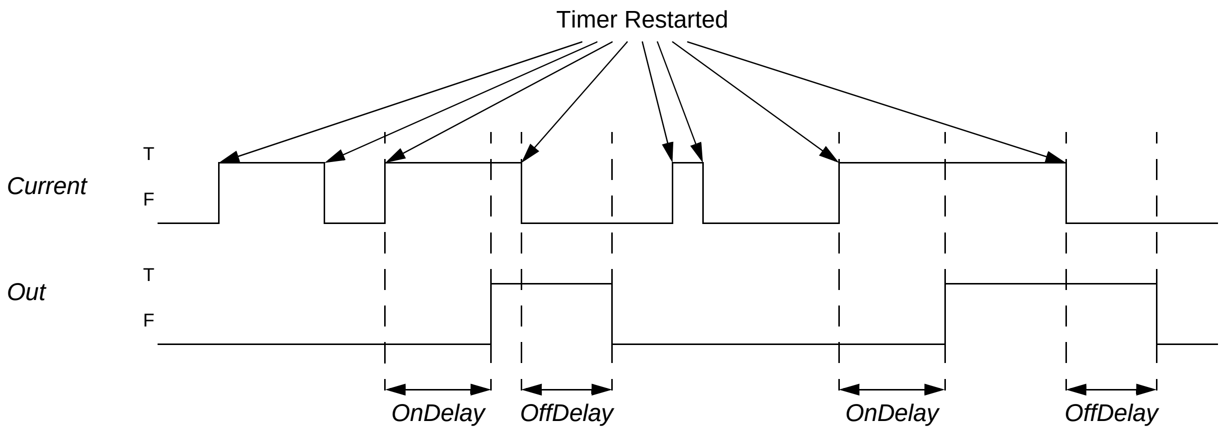
- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

In0Tmr to In7Tmr. Delay timers. Every time the block is processed *Current* is updated with the current condition based upon *In* and disables. If *Current* is different from *Last*, the appropriate *InxTmr* field is loaded with the value of *OnDelay* (if *Current* is TRUE) or *OffDelay* (if *Current* is FALSE). When the timer has reached zero, the output is updated from *Current*.

Current. This subfield shows which input is set after all disables have been taken into consideration.

Last. This subfield shows the status of *Current* at the previous iteration.

Out. This subfield shows the output status after the time delays and the disables have been applied. If the corresponding input has changed state before the associated timer has elapsed, the *InxTmr* field is loaded with the opposite delay value and the timing is restarted.



Status. Subfield reflecting the status of the alarm processing

- **NewCond.** Whenever an output changes from FALSE to TRUE, this bit is set TRUE. *Status.NewCond* can be reset by setting *Options.AckNew* TRUE. To get automatic reset after one database scan, wire *Options.AckNew* to *Status.NewCond.*
- **Combined.** 'OR' of all *Out* bits.

Options. Bitfield for selecting the different operational options of the block.

- **AckNew.** Resets *Status.NewCond.*
- **InitLast.** Setting this bit TRUE forces Last to be the inverse of *Current*. This feature could be useful at start-up.

Implementation notes

If DGDELAY8 control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 1800 bytes and by 154 bytes for each instance of the control module.

Note	For this application module please refer to factory.

Intentionally left blank

CHAPTER 11 CONTROL MODULE BLOCKS

This table shows the supervisory computer graphic objects that map on to the general control module types. Some of the graphic objects can be used with more than one function block type. The ✓ symbols are provided to show the valid permutations.

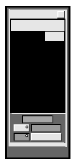
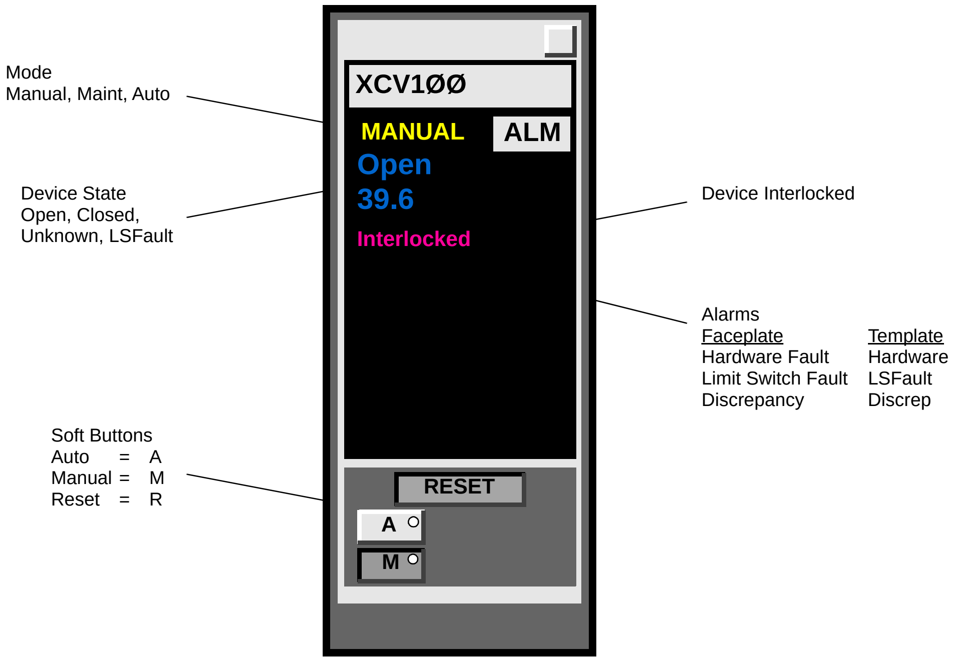
Block Type/ Supervisory computer graphic object	**AnManSt** HA084012U305 Issue 2	**DgManSt** HA084012U306 Issue 2				
CtlFpAA HA084012U301 Issue 2	✓	✓				
CtlFpAB HA084012U302 Issue 2			✓			
CtlIcnAA HA084012U303 Issue 2	✓	✓				
CtlIcnAB HA084012U304 Issue 2			✓			

Table 11-1: Supported block permutations

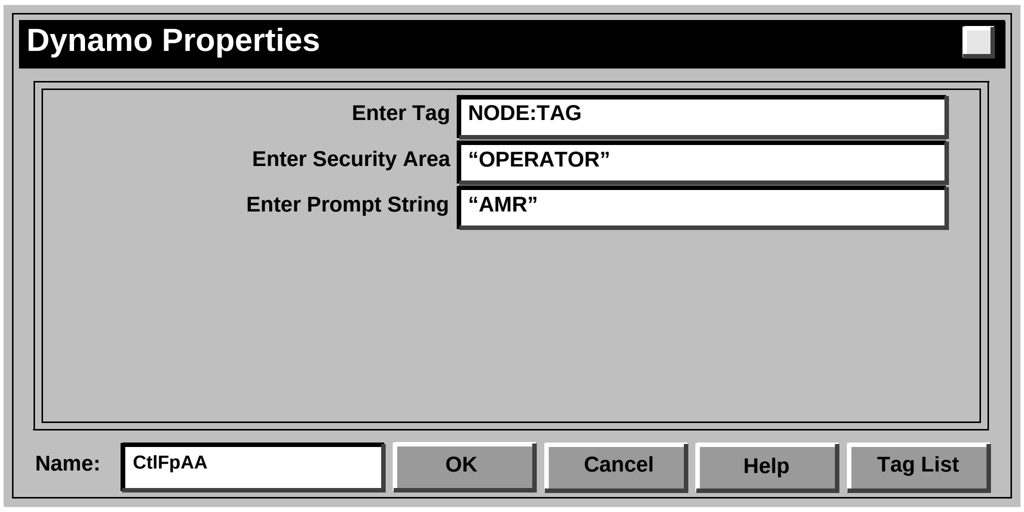
Intentionally left blank

VLV3WAY: CTLFPAA CONTROL GRAPHIC OBJECT

Runtime



Configuration


Dynamo Properties

Enter Tag	NODE:TAG
Enter Security Area	“OPERATOR”
Enter Prompt String	“AMR”

Name: CtlFpAA

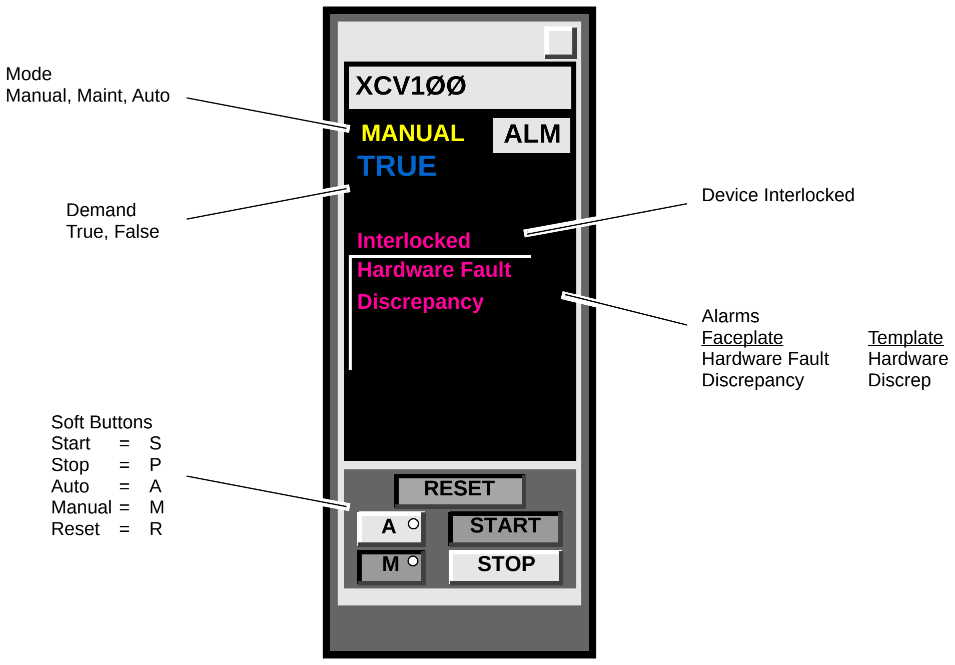
Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Enter Security Area:	Specifies the security area that the operator must have access rights to in order to be able to use the soft buttons. String must be in quotation marks.
Enter Prompt String:	Specifies the soft buttons for which a confirmation prompt is required. Contained within quotation marks are the key letters for each button where confirmation is required.

Table 11-1: Dynamo Properties

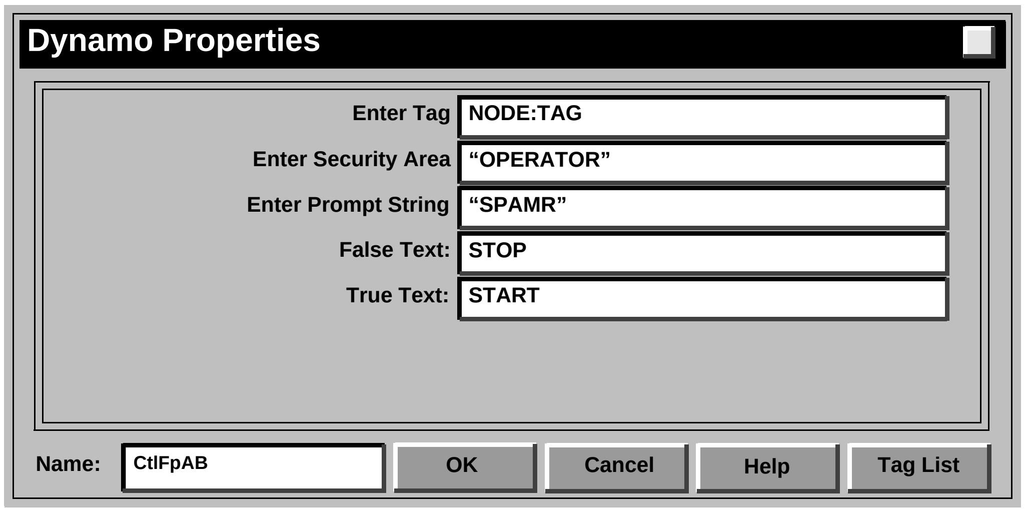
Intentionally left blank

VLV3WAY: CTLFPAB CONTROL GRAPHIC OBJECT

Runtime



Configuration



Enter Tag	NODE:TAG
Enter Security Area	“OPERATOR”
Enter Prompt String	“SPAMR”
False Text:	STOP
True Text:	START
Name:	CtlFpAB

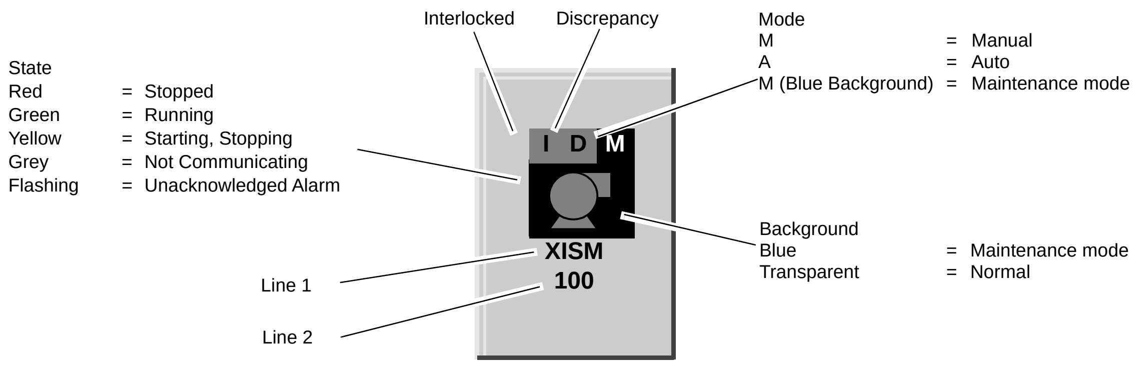
Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Enter Security Area:	Specifies the security area that the operator must have access rights to in order to be able to use the soft buttons. String must be in quotation marks.
Enter Prompt String:	Specifies the soft buttons for which a confirmation prompt is required. Contained within quotation marks are the key letters for each button where confirmation is required.
False Text:	Specifies the text that appears on the stop soft button.
True Text:	Specifies the text that appears on the start soft button.

Table 11-1: Dynamo Properties

Intentionally left blank

VLV3WAY: CTLICNAA CONTROL GRAPHIC OBJECT

Runtime



This icon is available in a number of orientations. These are collected within a common dynamo library, ICN_CT. The dynamos are named as indicated below.

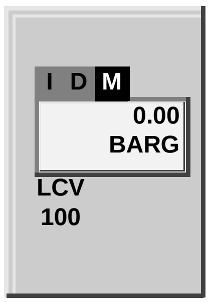
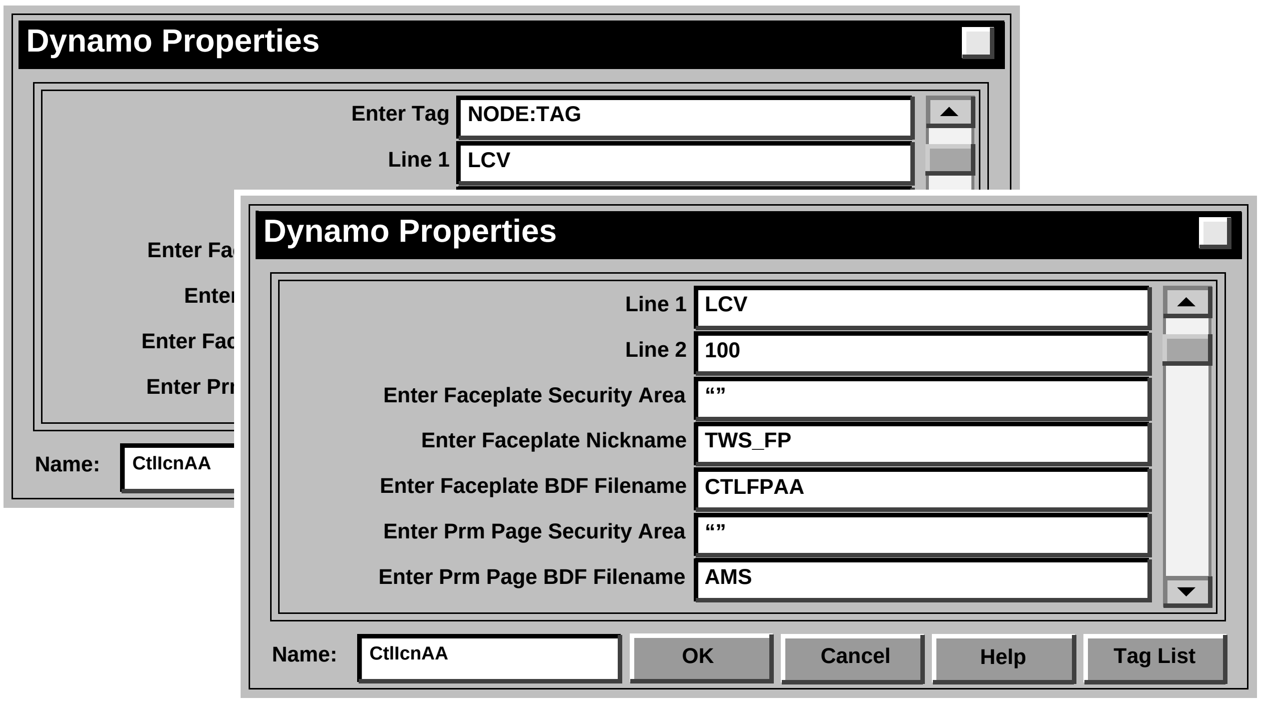
Dynamo name	Function
CTLICNAA	

Table 11-1: Typical graphic object

Configuration



Enter Tag NODE:TAG

Line 1 LCV

Line 2 100

Enter Faceplate Security Area “”

Enter Faceplate Nickname TWS_FP

Enter Faceplate BDF Filename CTLFPAA

Enter Prm Page Security Area “”

Enter Prm Page BDF Filename AMS

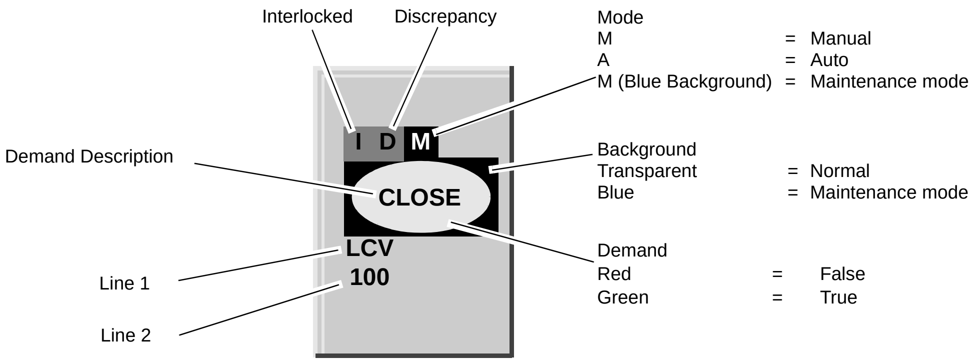
Name: CtllcnAA

Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Line 1/Line 2:	Textural control module tag or description.
Enter Faceplate Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module faceplate. String must be in quotation marks.
Enter Faceplate Nickname:	Specifies an alias name to the faceplate. The supervisory computer ensures that no two faceplates with the same alias may be simultaneously displayed.
Enter Faceplate BDF Filename:	Specifies the file that is activated as the control module faceplate.
Enter Prm Page Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module engineer page. String must be in quotation marks.
Enter Prm Page BDF Filename:	Specifies the file that is activated as the control module engineer page.

Table 11-2: Dynamo Properties

VLV3WAY: CTLICNAB CONTROL GRAPHIC OBJECT

Runtime



This icon is available in a number of orientations. These are collected within a common dynamo library, ICN_CT. The dynamos are named as indicated below.

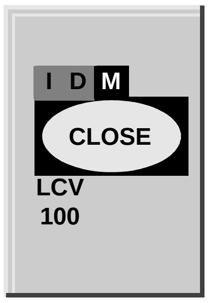
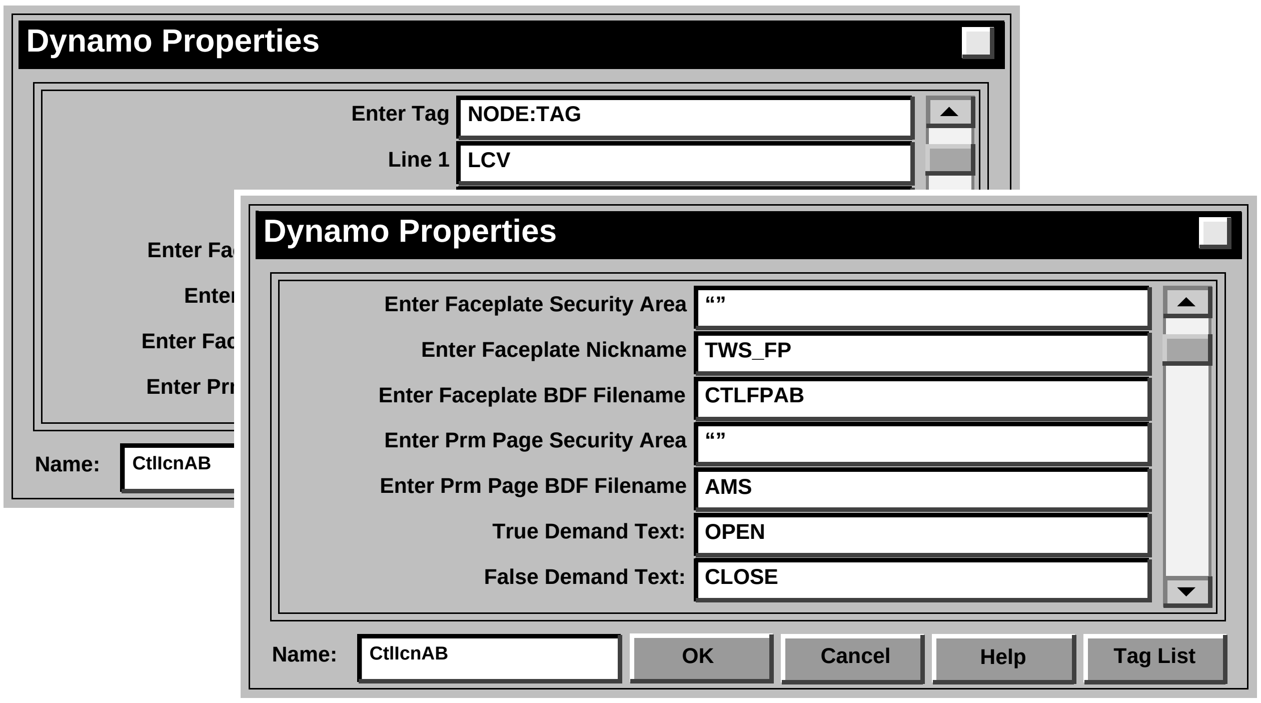
Dynamo name	Function
CTLICNAB	

Table 11-1: Typical graphic object

Configuration



Enter Tag: NODE:TAG
Line 1: LCV
Enter Faceplate Security Area: “”
Enter Faceplate Nickname: TWS_FP
Enter Faceplate BDF Filename: CTLFPAB
Enter Prm Page Security Area: “”
Enter Prm Page BDF Filename: AMS
True Demand Text: OPEN
False Demand Text: CLOSE
Name: CtllcnAB

Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Line 1/Line 2:	Textural control module tag or description.
Enter Faceplate Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module faceplate. String must be in quotation marks.
Enter Faceplate Nickname:	Specifies an alias name to the faceplate. The supervisory computer ensures that no two faceplates with the same alias may be simultaneously displayed.
Enter Faceplate BDF Filename:	Specifies the file that is activated as the control module faceplate.
Enter Prm Page Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module engineer page. String must be in quotation marks.
Enter Prm Page BDF Filename:	Specifies the file that is activated as the control module engineer page.
True Demand Text:	Specifies the text that appears on the open demand.
False Demand Text:	Specifies the text that appears on the close demand.

Table 11-2: Dynamo Properties

VLV3WAY: ANALOGUE MANUAL STATION BLOCK

Block function

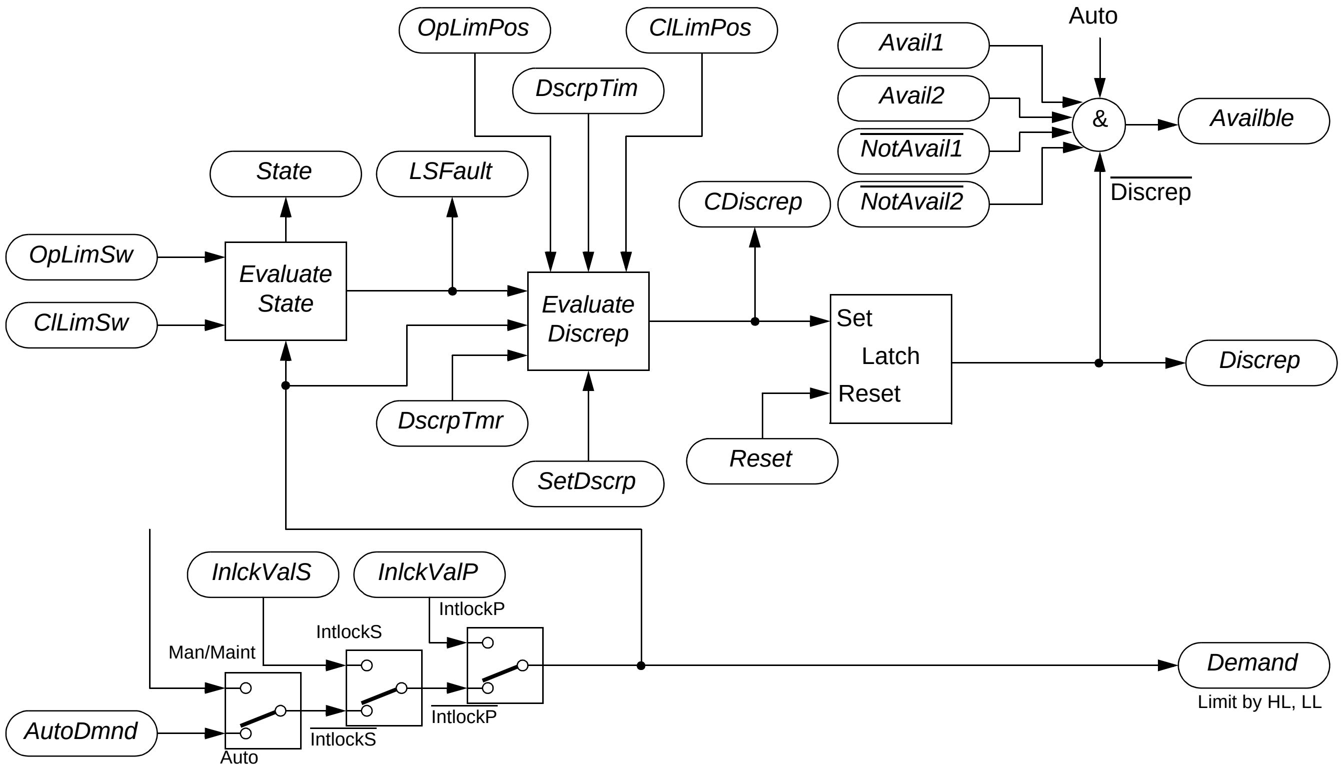
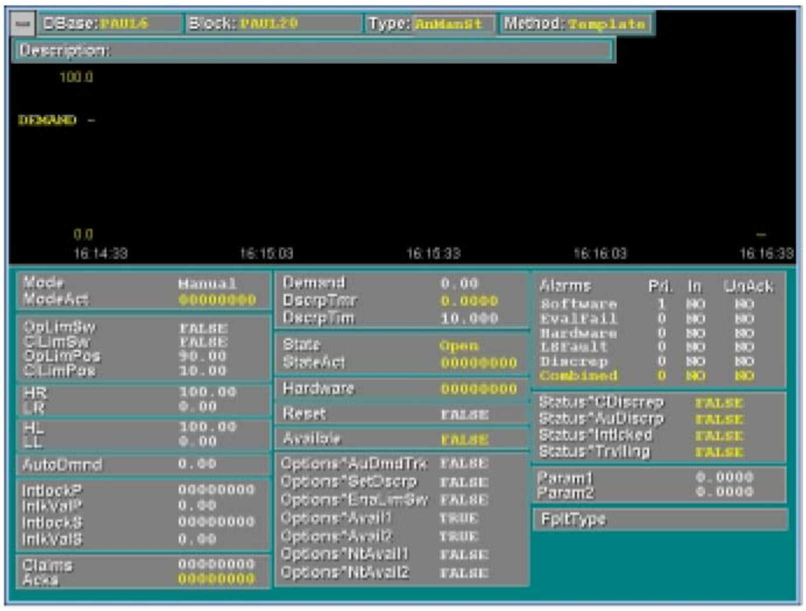


Figure 11-1: Block schematic

This block generates a demand to a single input analogue device. Where the device has dual limit switches, the block derives a device state that is used to evaluate a discrepancy.

In manual mode, the analogue demand may be set within configured limits by an operator. In automatic mode, the analogue demand may be set within configured limits by a controlling sequence or other analogue device user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase: PAUL6   Block: PAUL20   Type: AnManSt   Method: Template
Description:
100.0
DEMAND -
0.0
16:14:33    16:15:03    16:15:33    16:16:03    16:16:33

Mode        Manual      Demand              0.00       Alarms     Pri. In  UnAck
ModeAct     00000000    DscrpTmr            0.0000     Software   1    NO  NO
                        DscrpTim            10.000     EvalFail   0    NO  NO
OpLimSw     FALSE                                      Hardware   0    NO  NO
ClLimSw     FALSE       State               Open       LSFault    0    NO  NO
OpLimPos    90.00       StateAct            00000000   Discrep    0    NO  NO
ClLimPos    10.00                                      Combined   0    NO  NO
                        Hardware            00000000
HR          100.00                                     Status*CDiscrep   FALSE
LR          0.00        Reset               FALSE      Status*AuDiscrp   FALSE
                                                       Status*Intlcked   FALSE
HL          100.00      Availble            FALSE      Status*Trvlling   FALSE
LL          0.00
                        Options*AuDmdTrk    FALSE      Param1     0.0000
AutoDmnd    0.00        Options*SetDscrp    FALSE      Param2     0.0000
                        Options*EnaLimSw    FALSE
IntlockP    00000000    Options*Avail1      TRUE       FpltType
InlkValP    0.00        Options*Avail2      TRUE
IntlockS    00000000    Options*NtAvail1    FALSE
InlkValS    0.00        Options*NtAvail2    FALSE

Claims      00000000
Acks        00000000
```

Figure 11-2: Engineers page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	□➔ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
OpLimSw	Open limit switch	T/F	➔□➔
ClLimSw	Closed limit switch	T/F	➔□➔
OpLimPos	Open limit position	Eng	➔□➔
ClLimPos	Closed limit position	Eng	➔□➔
HR, LR	Demand high and low range	Eng	➔□➔ &
HL, LL	Demand high and low limit	Eng	➔□➔ &
AutoDmnd	Automatic demand	Eng	➔□➔
IntlockP	Primary interlock	CD hex	➔□➔
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValP	Primary interlock value	Eng	➔□➔
IntlockS	Secondary interlock	CD hex	➔□➔
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 11-1: Block parameters

Parameter	Function	Units	Status
InlkValS	Secondary interlock value	Eng	➡▢➡
Claim	Resource management claims	CD hex	➡▢➡
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	▢➡ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Alarms			▢➡ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	I/O hardware fault	T/F	
LSFault	Limit switch fault	T/F	
Discrep	Latched discrepancy	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Position demand	Eng	▢➡
DscrpTmr	Discrepancy countdown timer	Secs	▢➡ 📖 ⏱
DscrpTim	Discrepancy (travel) time	Secs	➡▢➡
PulseTim	Output pulse time	Secs	➡▢➡
StateAct		CD hex	▢➡ 📖
Open	Open position	T/F	
Closed	Closed position	T/F	
Opening	Opening position	T/F	
Closing	Closing position	T/F	
LSFault	Limit switch fault position	T/F	
Unknown	Unknown position	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 11-1: Block parameters

Parameter	Function	Units			Status
Status	Operational status bitfields	ABCD hex			
CDiscrep*	Current discrepancy	T/F	1	D	
AuDiscrp*	Cannot respond to automatic control	T/F	2	D	
Intlcked*	Interlocked	T/F	4	D	
Trvlling*	Travelling	T/F	8	D	
Bit4	Open position	T/F	1	C	
Bit5	Closed position	T/F	2	C	
Bit6	Opening position	T/F	4	C	
Bit7	Closing position	T/F	8	C	
Bit8	Limit switch fault position	T/F	1	B	
Bit9	Unknown position	T/F	2	B	
Bit10		T/F	4	B	
Bit11		T/F	8	B	
Bit12		T/F	1	A	
Bit13		T/F	2	A	
Bit14		T/F	4	A	
Discrep*	Latched discrepancy	T/F	8	A	
Hardware	I/O hardware failure input	CD hex			
Bit0	Hardware failure #0	T/F			Ø
Bit1	Hardware failure #1	T/F			Ø
Bit2	Hardware failure #2	T/F			Ø
Bit3	Hardware failure #3	T/F			Ø
Bit4	Hardware failure #4	T/F			Ø
Bit5	Hardware failure #5	T/F			Ø
Bit6	Hardware failure #6				Ø
Bit7	Hardware failure #7				Ø
Options	Optional configuration bitfields	ABCD hex			
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1	D	
SetDscrp	Set discrepancy alarm	T/F	2	D	
EnaLimSw	Enable limit switch logic	T/F	4	D	
Avail1	User device available #1	T/F	8	D	
Avail2	User device available #2	T/F	1	C	
NotAvail1	User device not available #1	T/F	2	C	
NotAvail2	User device not available #2	T/F	4	C	
Bit7	User option #7	T/F	8	C	
Bit8	User option #8	T/F	1	B	
Bit9	User option #9	T/F	2	B	
Bit10	User option #10	T/F	4	B	
Bit11	User option #11	T/F	8	B	
Bit12	User option #12	T/F	1	A	
Bit13	User option #13	T/F	2	A	
Bit14	User option #14	T/F	4	A	
Bit15	User option #15	T/F	8	A	

Table 11-1: Block parameters

Parameter	Function	Units	Status
Reset	Reset latched discrepancy	T/F	▶□▶
Availble	Available for automatic control	T/F	□▶ 📖
FpltType	Supervisory faceplate type	String	
Param1	User parameter # 1	Eng	▶□▶
Param2	User parameter # 2	Eng	▶□▶

** Input wiring will disrupt the normal operation of the block*

Table 11-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

OpLimPos. Open Limit Position. Sets the demand at which the device should assert its open limit switch.

ClLimPos. Closed Limit Position. Sets the demand at which the device should assert its close limit switch.

HR, LR. Demand High and Low Ranges. Defines the upper and lower bound for the *OpLimPos*, *ClLimPos*, *HR*, *LR*, *HL*, *LL*, *AutoDmnd*, *InlkValP*, *InlkValS* and *Demand* fields. Does not limit the value of the bounded fields.

HL, LL. High and Low Limit for the Demand field. Demand limits are active in all modes of operation.

IntlockP. Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the analogue device. The analogue device resource may be controlled by up to eight sequences or other analogue device users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the analogue device.
- **LSFault.** Limit switches indicate analogue device is both 'Open' and 'Closed'.
- **Discrep.** Analogue device has not responded to the demand or has moved off its limit. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand for the position of the analogue device.

DscrpTmr. Discrepancy Timer. Internally set to the discrepancy time on demand change. The discrepancy alarm is raised if this timer counts down to zero and the analogue device has not reacted to the demand.

DscrpTim. Discrepancy Time. The time given to the analogue device to react before a discrepancy alarm is raised.

State. (Open/Closed/Opening/Closing/LSFault/Unknown). Current analogue device state. Derived from the limit switches. Opening and closing are never displayed.

Status. This bitfield shows the status of the analogue device.

- **CDiscrep.** Current Discrepancy. TRUE when the analogue device is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the analogue device is not available. May be used by a controlling sequence or other analogue device users to determine if the automatic demand is active.
- **Intlcked.** TRUE indicates an asserted interlock.
- **Trvlling.** Internal flag asserted when analogue device travelling.
- **Discrep.** Latched TRUE when the analogue device is in discrepancy. Set FALSE during the reset action.
- **Bit4-Bit14.** Optional status bits. Associates additional Boolean data with the analogue device.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the analogue device.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **SetDscrp.** TRUE asserts the discrepancy alarm regardless of the demand and the limit switches. Allows external logic to extend the discrepancy functionality. Internally set FALSE after the reset action.
- **EnaLimSw.** FALSE disables the limit switch discrepancy logic.
- **Avail1,Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1,NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **Bit7-Bit15.** Optional option bits. Associates additional Boolean data with the analogue device.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when analogue device is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associates a faceplate type or other alphanumeric string with the analogue device. Allows analogue devices of this type to be represented by different supervisory computer faceplates.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the analogue device.

Implementation notes

If AnManSt control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2264 bytes and by 154 bytes for each instance of the control module.

VLV3WAY: DIGITAL MANUAL STATION BLOCK

Block function

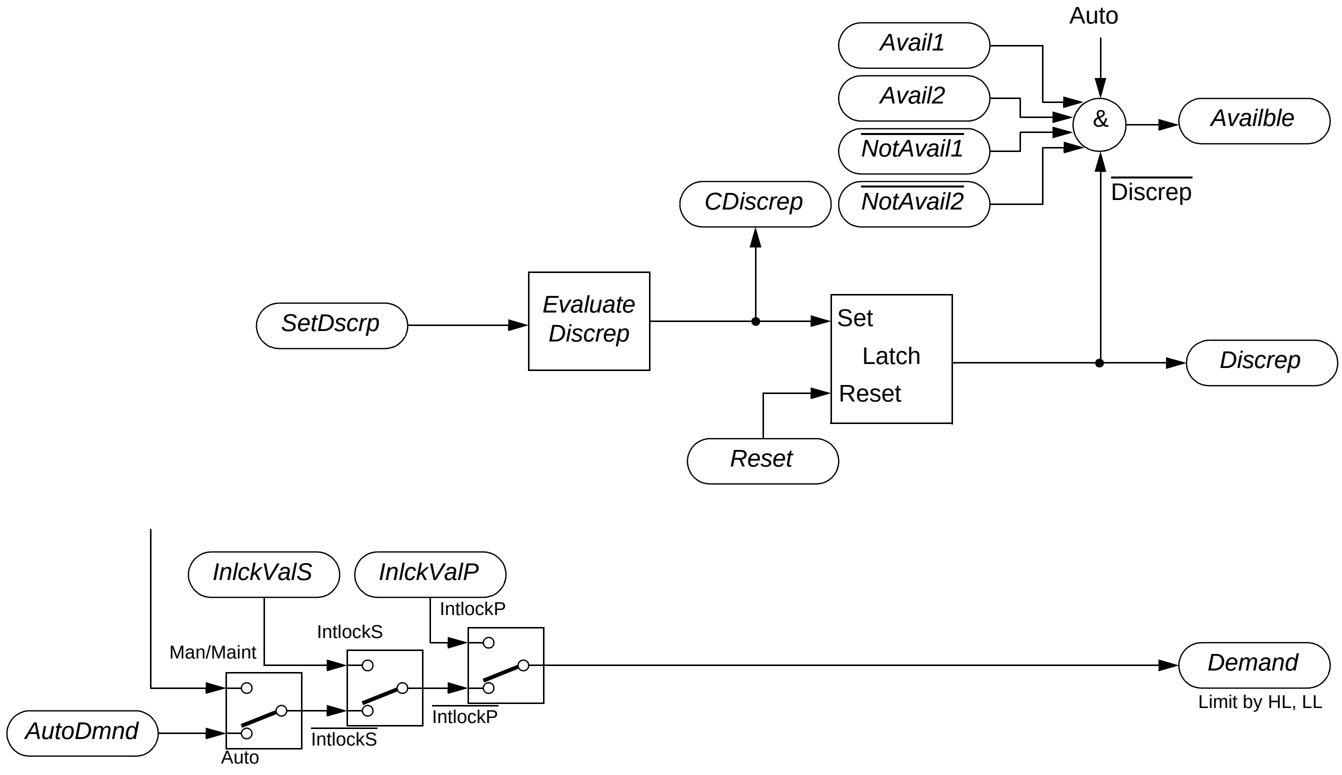
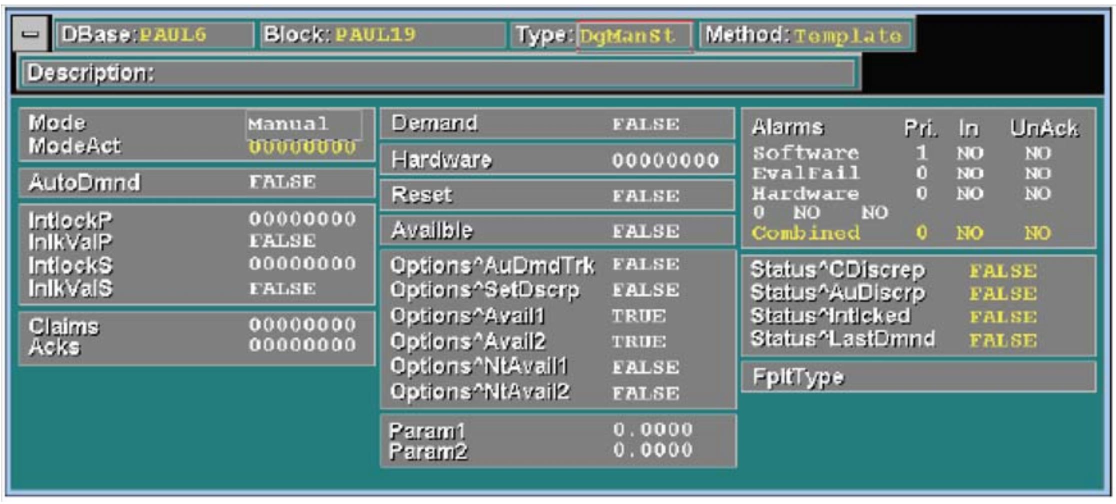


Figure 11-1: Block schematic

Please refer to the schematic. The DgManSt block generates a demand to a single input digital device.

In manual mode, the digital demand may be set by an operator. In automatic mode, the digital demand may be set by a controlling sequence or other digital device user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase: PAUL6    Block: PAUL19    Type: DgManSt    Method: Template
Description:

Mode            Manual       Demand              FALSE      Alarms       Pri.  In   UnAck
ModeAct         00000000     Hardware            00000000   Software     1     NO   NO
AutoDmnd        FALSE        Reset               FALSE      EvalFail     0     NO   NO
IntlockP        00000000     Availble            FALSE      Hardware     0     NO   NO
InlkValP        FALSE        Options^AuDmdTrk    FALSE      0   NO   NO
IntlockS        00000000     Options^SetDscrp    FALSE      Combined     0     NO   NO
InlkValS        FALSE        Options^Avail1      TRUE       Status^CDiscrep    FALSE
Claims          00000000     Options^Avail2      TRUE       Status^AuDiscrp    FALSE
Acks            00000000     Options^NtAvail1    FALSE      Status^Intlcked    FALSE
                             Options^NtAvail2    FALSE      Status^LastDmnd    FALSE
                             Param1              0.0000     FpltType
                             Param2              0.0000
```

Figure 11-2: Engineers page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode actived	CD hex	☐➡ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
AutoDmnd	Automatic demand	Eng	➡☐➡
IntlockP	Primary interlock	CD hex	➡☐➡
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValP	Primary interlock value	Eng	➡☐➡
IntlockS	Secondary interlock	CD hex	➡☐➡
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValS	Secondary interlock value	Eng	➡☐➡
Claim	Resource management claims	CD hex	➡☐➡
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 11-1: Block parameters

Parameter	Function	Units	Status
Acks	Resource management acknowledgements	CD hex	
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	Ø Ø Ø Ø Ø Ø Ø Ø
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	I/O hardware fault	T/F	
Discrep	Latched discrepancy	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Position demand	Eng	
Status	Operational stsus bitfields	ABCD hex	
CDiscrep*	Current discrepancy	T/F	1 D
AuDiscrp*	Cannot respond to automatic control	T/F	2 D
Intlcked*	Interlocked	T/F	4 D
Bit4	User status #4	T/F	8 D
LastDmnd*	Demand at last iteration	T/F	1 C
Bit5	User status #5	T/F	2 C
Bit6	User status #6	T/F	4 C
Bit7	User status #7	T/F	8 C
Bit8	User status #8	T/F	1 B
Bit9	User status #9	T/F	2 B
Bit10	User status #10	T/F	4 B
Bit11	User status #11	T/F	8 B
Bit12	User status #12	T/F	1 A
Bit13	User status #13	T/F	2 A
Bit14	User status #14	T/F	4 A
Discrep*	Latched discrepancy	T/F	8 A
Hardware	I/O hardware failure input	CD hex	
Bit0	Hardware failure #0	T/F	
Bit1	Hardware failure #1	T/F	
Bit2	Hardware failure #2	T/F	
Bit3	Hardware failure #3	T/F	
Bit4	Hardware failure #4	T/F	
Bit5	Hardware failure #5	T/F	
Bit6	Hardware failure #6	T/F	
Bit7	Hardware failure #7	T/F	Ø Ø Ø Ø Ø Ø Ø Ø

Table 11-1: Block parameters

Parameter	Function	Units			Status
Options	Optional configuration bitfields	ABCD hex			▶□▶
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1	D	
SetDscrp	Set discrepancy alarm	T/F	2	D	?
Avail1	User device available #1	T/F	4	D	
Avail2	User device available #2	T/F	8	D	
NotAvail1	User device not available #1	T/F	1	C	
NotAvail2	User device not available #2	T/F	2	C	
Bit6	User option #6	T/F	4	C	
Bit7	User option #7	T/F	8	C	
Bit8	User option #8	T/F	1	B	
Bit9	User option #9	T/F	2	B	
Bit10	User option #10	T/F	4	B	
Bit11	User option #11	T/F	8	B	
Bit12	User option #12	T/F	1	A	
Bit13	User option #13	T/F	2	A	
Bit14	User option #14	T/F	4	A	
Bit15	User option #15	T/F	8	A	
Reset	Reset latched discrepancy	T/F			▶□▶ ?
Availble	Available for automatic control	T/F			□▶ 📖
FpltType	Supervisory faceplate type	String			
Param1	User parameter # 1	Eng			▶□▶
Param2	User parameter # 2	Eng			▶□▶

Table 11-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

IntlockP. Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the digital device. The digital device resource may be controlled by up to eight sequences or other digital device users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the digital device.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.

- **Hardware.** Hardware failure flagged by an input or output block associated with the digital device.
- **Discrep.** Digital device has not responded to the demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand for the position of the digital device.

Status. This bitfield shows the status of the digital device.

- **CDiscrep.** Current Discrepancy. TRUE when the digital device is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the digital device is not available. May be used by a controlling sequence or other digital device users to determine if the automatic demand is active.
- **Intlcked**. TRUE indicates an asserted interlock.
- **Discrep.** Latched TRUE when the digital device is in discrepancy. Set FALSE during the reset action.
- **Bit4, Bit6-Bit14.** Optional status bits. Associates additional Boolean data with the digital device.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the digital device.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **SetDscrp.** TRUE asserts the discrepancy alarm regardless of the demand. Allows external logic to implement the discrepancy functionality. Internally set FALSE after the reset action.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **Bit6-Bit15.** Optional option bits. Associates additional Boolean data with the digital device.

Reset. TRUE resets the discrepancy alarm. Internally set FALSE after the reset action.

Availble. TRUE when digital device is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the digital device. Allows digital devices of this type to be represented by different supervisory computer faceplates.

Implementation notes

If DgManSt control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 1488 bytes and by 100 bytes for each instance of the control module.

Intentionally left blank

CHAPTER 12 MOTOR MODULE BLOCKS

This table shows the supervisory computer graphic objects that map on to the motor control module types. Some of the graphic objects can be used with more than one function block type. The ✓ symbols are provided to show the valid permutations.

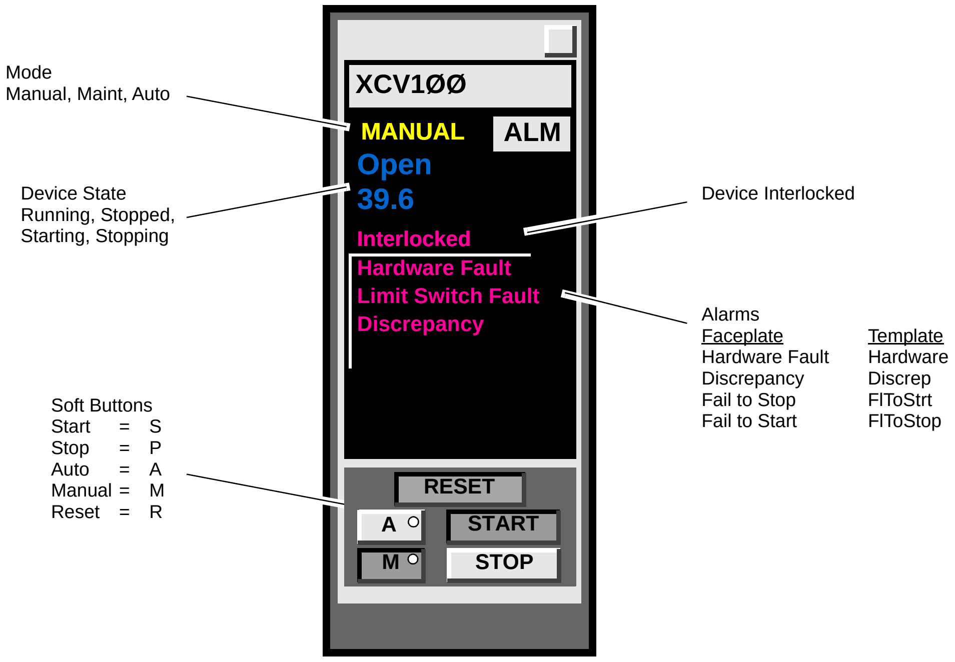
Block Type/ Supervisory computer graphic object	**Mtr3In** HA084012U203 Issue 2					
MtrFPAA HA084012U201 Issue 2	✓					
PMPlcnAA_ha HA084012U202 Issue 2	✓					
PMPlcnAA_hl HA084012U202 Issue 2	✓					

Table 12-1: Supported block permutations

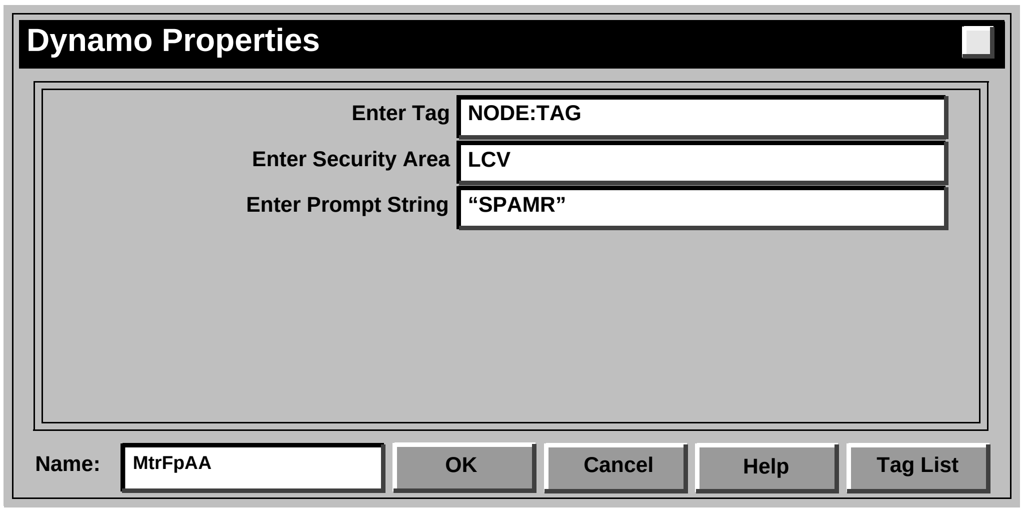
Intentionally left blank

VLV3WAY: MTRFPAA MOTOR GRAPHIC OBJECT

Runtime



Configuration



Enter Tag: NODE:TAG
Enter Security Area: LCV
Enter Prompt String: “SPAMR”
Name: MtrFpAA

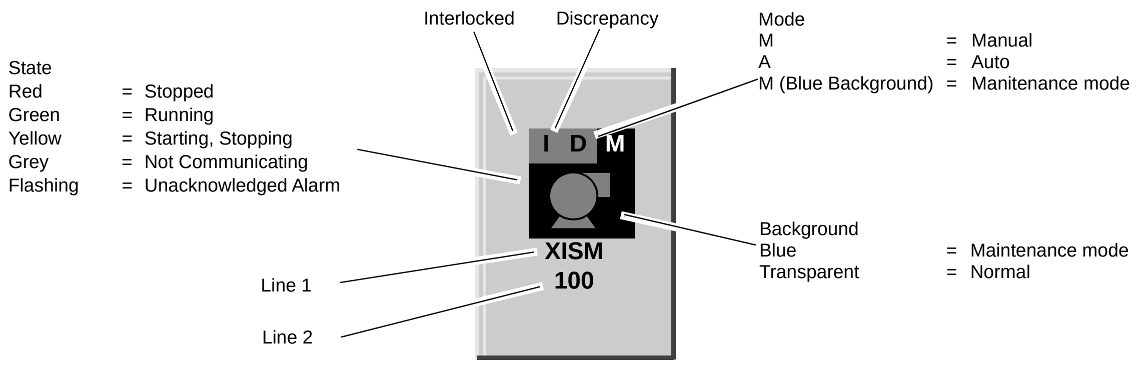
Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Enter Security Area:	Specifies the security area that the operator must have access rights to in order to be able to use the soft buttons. String must be in quotation marks.
Enter Prompt String:	Specifies the soft buttons for which a confirmation prompt is required. Contained within quotation marks are the key letters for each button where confirmation is required.

Table 12-1: Dynamo Properties

Intentionally left blank

VLV3WAY: PMPICNAA MOTOR GRAPHIC OBJECT

Runtime



This icon is available in a number of orientations. These are collected within a common dynamo library, ICN_PMP. The dynamos are named as indicated below.

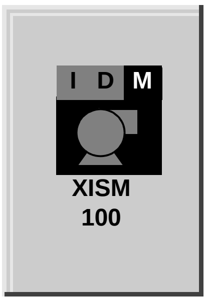
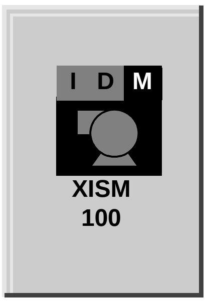
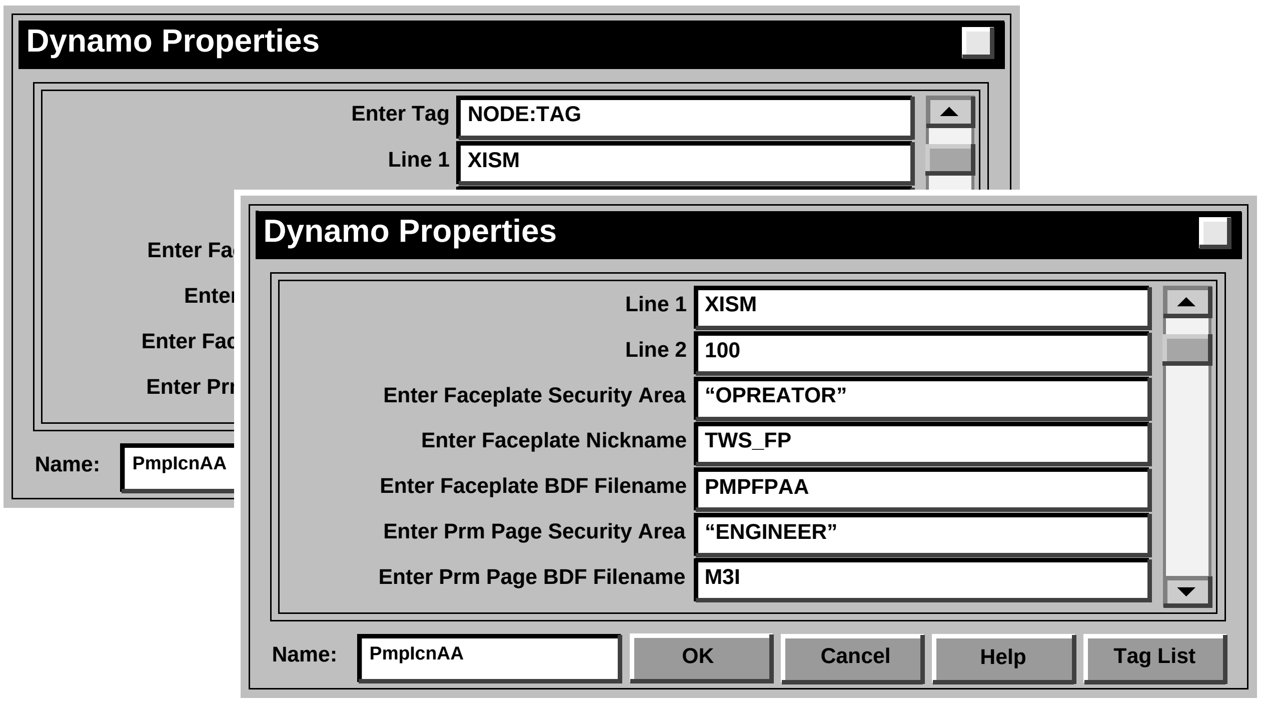
Dynamo name	Function
PmpIcnAA_ha	
PmpIcnAA_vl	

Table 12-1: Typical graphic object

Configuration



Dynamo Properties

Enter Tag	NODE:TAG
Line 1	XISM
Line 2	100
Enter Faceplate Security Area	“OPREATOR”
Enter Faceplate Nickname	TWS_FP
Enter Faceplate BDF Filename	PMPFPAA
Enter Prm Page Security Area	“ENGINEER”
Enter Prm Page BDF Filename	M3I
Name:	PmplcnAA

Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Line 1/Line 2:	Textural control module tag or description.
Enter Faceplate Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module faceplate. String must be in quotation marks.
Enter Faceplate Nickname:	Specifies an alias name to the faceplate. The supervisory computer ensures that no two faceplates with the same alias may be simultaneously displayed.
Enter Faceplate BDF Filename:	Specifies the file that is activated as the control module faceplate.
Enter Prm Page Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module engineer page. String must be in quotation marks.
Enter Prm Page BDF Filename:	Specifies the file that is activated as the control module engineer page.

Table 12-2: Dynamo Properties

VLV3WAY: ON/OFF MOTOR; THREE I/PS, MAINTAINED OR PULSED O/PS BLOCK

Block function

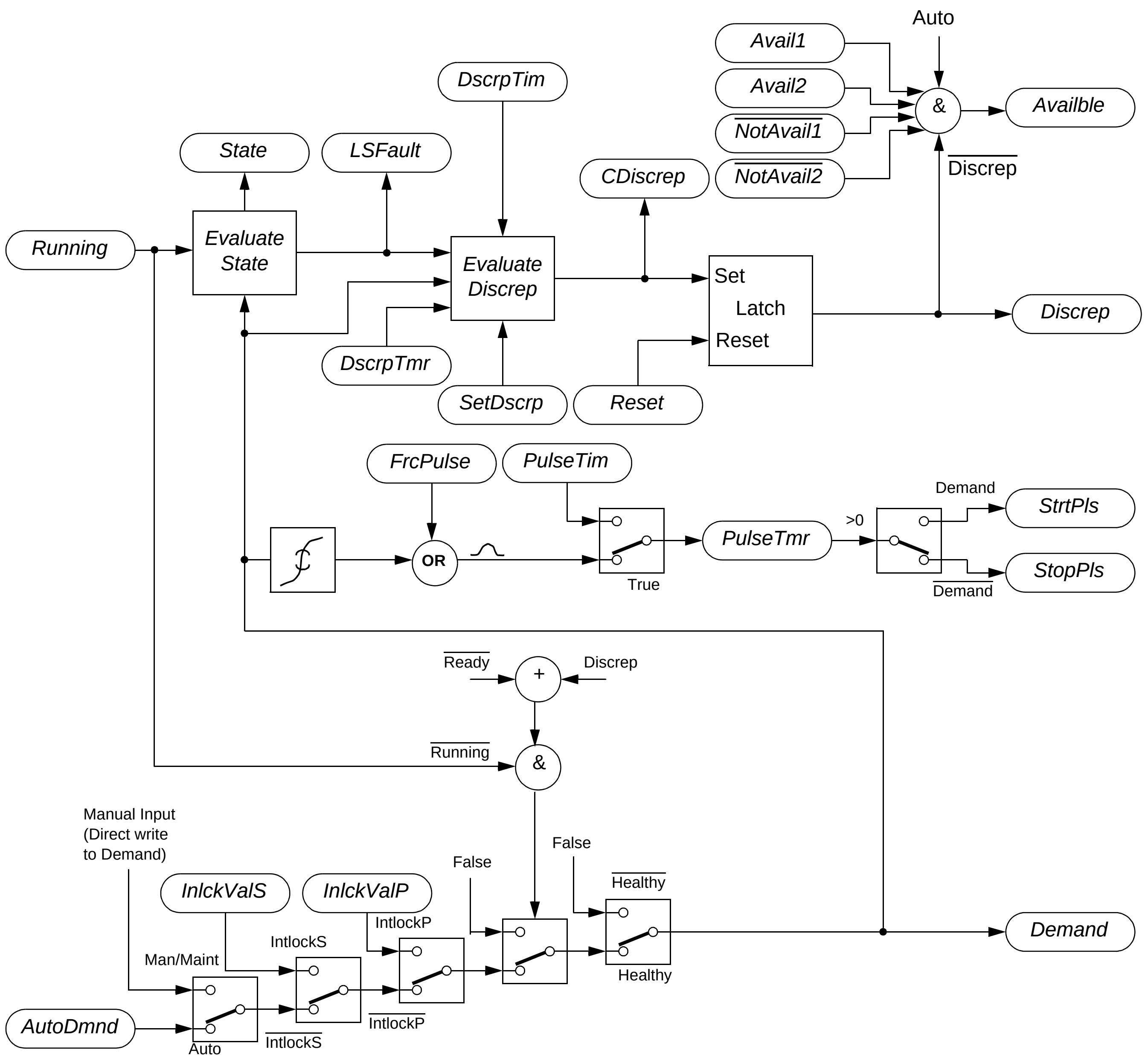
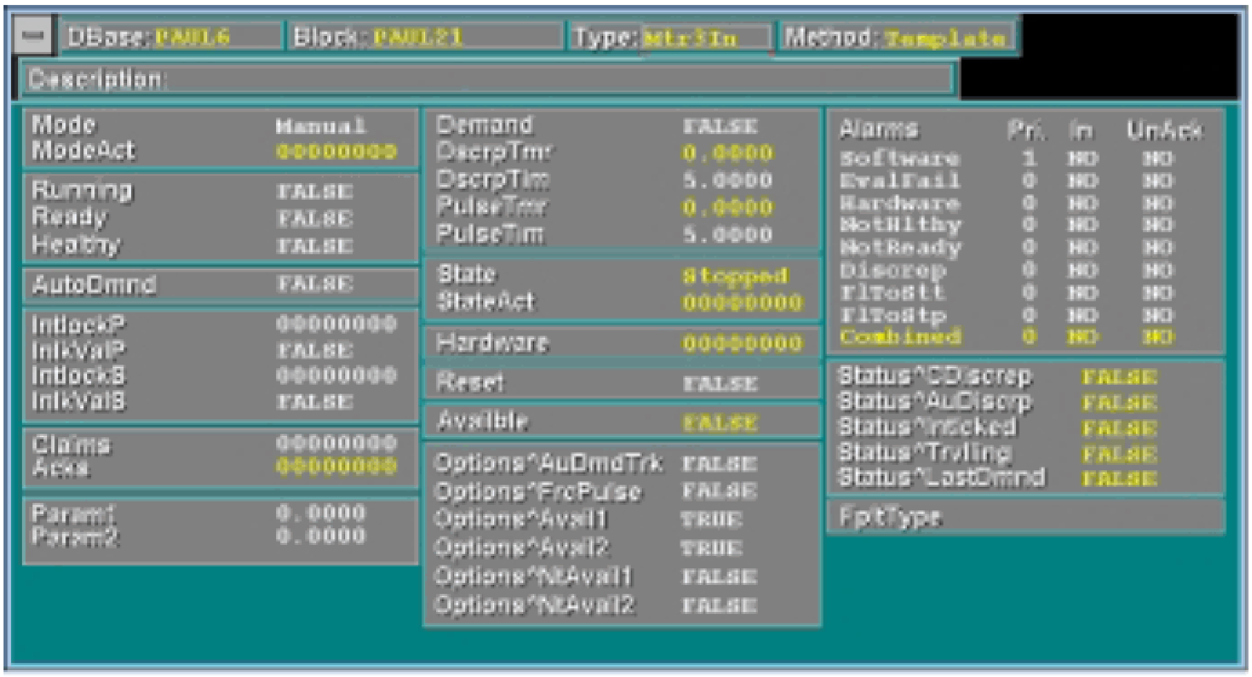


Figure 12-1: Block schematic

Please refer to the schematic. The Mtr3In block generates a demand to an on/off motor. A choice of maintained output (*Demand*) or pulsed outputs (*StateAct.StartPls*, *StateAct.StopPls*) is provided. A single pulse time applies to both start and stop pulses.

In manual mode, the motor may be started and stopped by an operator. In automatic mode, the motor is started and stopped by a controlling sequence or other motor user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase: PAUL6   Block: PAUL21   Type: Mtr3In   Method: Template
Description:

Mode        Manual      Demand                FALSE     Alarms      Pri.  In   UnAck
ModeAct     00000000    DscrpTmr              0.0000    Software    1     NO   NO
                        DscrpTim              5.0000    EvalFail    0     NO   NO
Running     FALSE       PulseTmr              0.0000    Hardware    0     NO   NO
Ready       FALSE       PulseTim              5.0000    NotHlthy    0     NO   NO
Healthy     FALSE                                       NotReady    0     NO   NO
                        State                 Stopped   Discrep     0     NO   NO
AutoDmnd    FALSE       StateAct              00000000  FlToStt     0     NO   NO
                                                        FlToStp     0     NO   NO
IntlockP    00000000    Hardware              00000000  Combined    0     NO   NO
InlkValP    FALSE
IntlockB    00000000    Reset                 FALSE     Status^CDDiscrep   FALSE
InlkValB    FALSE                                       Status^AuDiscrp    FALSE
                        Avalble               FALSE     Status^Intlcked    FALSE
Claims      00000000                                    Status^Trvling     FALSE
Acks        00000000    Options^AuDmdTrk      FALSE     Status^LastDmnd    FALSE
                        Options^FrcPulse      FALSE
Param1      0.0000      Options^Avail1        TRUE      FptType
Param2      0.0000      Options^Avail2        TRUE
                        Options^NtAvail1      FALSE
                        Options^NtAvail2      FALSE
```

Figure 12-2: Engineers page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	➔ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Running	Running signal	T/F	⇄
Ready	Ready signal	T/F	⇄
Healthy	Healthy signal	T/F	⇄
AutoDmnd	Automatic demand	T/F	⇄
IntlockP	Primary interlock	CD hex	⇄
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 12-1: Block parameters

Parameter	Function	Units	Status
InlkValP	Primary interlock value	Eng	➧▭➧
IntlockS	Secondary interlock	CD hex	➧▭➧
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValS	Secondary interlock value	Eng	➧▭➧
Claim	Resource management claims	CD hex	➧▭➧
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	▭➧ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Param1	User parameter # 1	Eng	➧▭➧
Param2	User parameter # 2	Eng	➧▭➧
Alarms			▭➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block’s internal logic	T/F	
Hardware	I/O hardware fault	T/F	
NotHlthy	Not healthy	T/F	
NotReady	Not ready	T/F	
Discrep	Latched discrepancy	T/F	
FlToStt	Failed to start	T/F	
FlToStp	Failed to stop	T/F	
Combined	OR-ing of all alarm bits	T/F	

Table 12-1: Block parameters

Parameter	Function	Units	Status
Demand	Position demand	Eng	☐➡
DscrpTmr	Discrepancy countdown timer	Secs	☐➡ 📖 🕓
DscrpTim	Discrepancy (travel) time	Secs	➡☐➡
PulseTmr	Output pulse timer		☐➡ 📖 🕓
PulseTim	Output pulse time	Secs	➡☐➡
State	Derived state	Enum	☐➡ 📖
StateAct	Active state	CD hex	☐➡ 📖
Stopped	Stopped state	T/F	
Running	Running state	T/F	
Stopping	Stopping state	T/F	
Starting	Starting state	T/F	
Bit4	Reserved for future use	T/F	
Bit5	Reserved for future use	T/F	
StartPls	Start output pulse	T/F	
StopPls	Stop output pulse	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Status	Operational status bitfields	ABCD hex	➡☐➡
CDiscrep*	Current discrepancy	T/F	1 D
AuDiscrp*	Cannot respond to automatic control	T/F	2 D
Intlcked*	Interlocked	T/F	4 D
Trvlling*	Travelling	T/F	8 D
LastDmnd*	Reserved for future use	T/F	1 C
Bit5	User status #5	T/F	2 C
Bit6	User status #6	T/F	4 C
Bit7	User status #7	T/F	8 C
Bit8	User status #8	T/F	1 B
Bit9	User status #9	T/F	2 B
Bit10	User status #10	T/F	4 B
Bit11	User status #11	T/F	8 B
Bit12	User status #12	T/F	1 A
Bit13	User status #13	T/F	2 A
Bit14	User status #14	T/F	4 A
Discrep*	Latched discrepancy	T/F	8 A
Hardware	I/O hardware failure input	CD hex	➡☐➡
Bit0	Hardware failure #0	T/F	
Bit1	Hardware failure #1	T/F	
Bit2	Hardware failure #2	T/F	
Bit3	Hardware failure #3	T/F	
Bit4	Hardware failure #4	T/F	
Bit5	Hardware failure #5	T/F	
Bit6	Hardware failure #6		
Bit7	Hardware failure #7		
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 12-1: Block parameters

Parameter	Function	Units			Status	
Options	Optional configuration bitfields	ABCD hex			➧☐➧	
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1	D		
FrcPulse	Force output pulse	T/F	2	D		
Avail1	User device available #1	T/F	4	D		
Avail2	User device available #2	T/F	8	D		
NotAvail1	User device not available #1	T/F	1	C		
NotAvail2	User device not available #2	T/F	2	C		
Bit6	User option #6	T/F	4	C		
Bit7	User option #7	T/F	8	C		
Bit8	User option #8	T/F	1	B		
Bit9	User option #9	T/F	2	B		
Bit10	User option #10	T/F	4	B		
Bit11	User option #11	T/F	8	B		
Bit12	User option #12	T/F	1	A		
Bit13	User option #13	T/F	2	A		
Bit14	User option #14	T/F	4	A		
Bit15	User option #15	T/F	8	A		
Reset	Reset latched discrepancy	T/F			➧☐➧	?
Availble	Available for automatic control	T/F			☐➧	🕮
FpltType	Supervisory faceplate type	String				

** Input wiring will disrupt the normal operation of the block*

Table 12-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

Ready. Motor is ready to respond to a remote demand, e.g. not local MCC control.

Healthy. Motor is healthy, e.g. not tripped.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

IntlockP. Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the motor. The motor resource may be controlled by up to eight sequences or other motor users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the motor.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the motor.
- **Discrep.** Motor has not responded to the demand or has changed state without a change of demand. Alarm is latched.
- **FlToStrt.** Motor has not responded to a 'Start' demand. Alarm is latched.
- **FlToStop.** Motor has not responded to a 'Stop' demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand for the position of the motor. FALSE = 'Stop'; TRUE = 'Start'.

DscrpTmr. Discrepancy Timer. Internally set to the discrepancy time on demand change. The discrepancy alarm is raised if this timer counts down to zero and the motor has not reacted to the demand.

DscrpTim. Discrepancy Time. The time given to the motor to react before a discrepancy alarm is raised.

PulseTmr. Pulse Timer. Internally set to the pulse time on demand change. Output pulse is asserted while the timer is counting down to zero.

PulseTim. Pulse Time. The minimum time for which a pulsed output is maintained.

State. (Stopped/Running/Starting/Stopping/Unknown). Current motor state. Derived from the running signal.

Status. This bitfield shows the status of the motor.

- **CDiscrep.** Current Discrepancy. TRUE when the motor is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the motor is not available or the automatic demand is not equal to the demand. May be used by a controlling sequence or other motor users to determine if the automatic demand is active.
- **Intlcked.** TRUE indicates an asserted interlock.
- **Trvlling.** Internal flag asserted when motor is starting or stopping.
- **LastDmnd.** Internal flag indicating demand at last iteration.
- **Discrep.** Latched TRUE when the motor is in discrepancy. Set FALSE during the reset action.
- **Bit5-Bit14.** Optional status bits. Associates additional Boolean data with the motor.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the motor.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **FrcPulse.** TRUE forces the control module to re-transmit the last output pulse. Internally set FALSE after the output pulse is started.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **Bit6-Bit15.** Optional option bits. Associates additional Boolean data with the motor.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when motor is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the motor. Allows motors of this type to be represented by different supervisory computer faceplates.

Implementation notes

If Mtr3In control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2198 bytes and by 134 bytes for each instance of the control module.

CHAPTER 13 VALVE MODULE BLOCKS

This table shows the supervisory computer graphic objects that map on to the valve control module types. Some of the graphic objects can be used with more than one function block type. The symbols are provided to show the valid permutations.

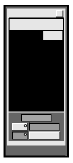
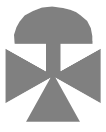
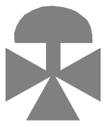
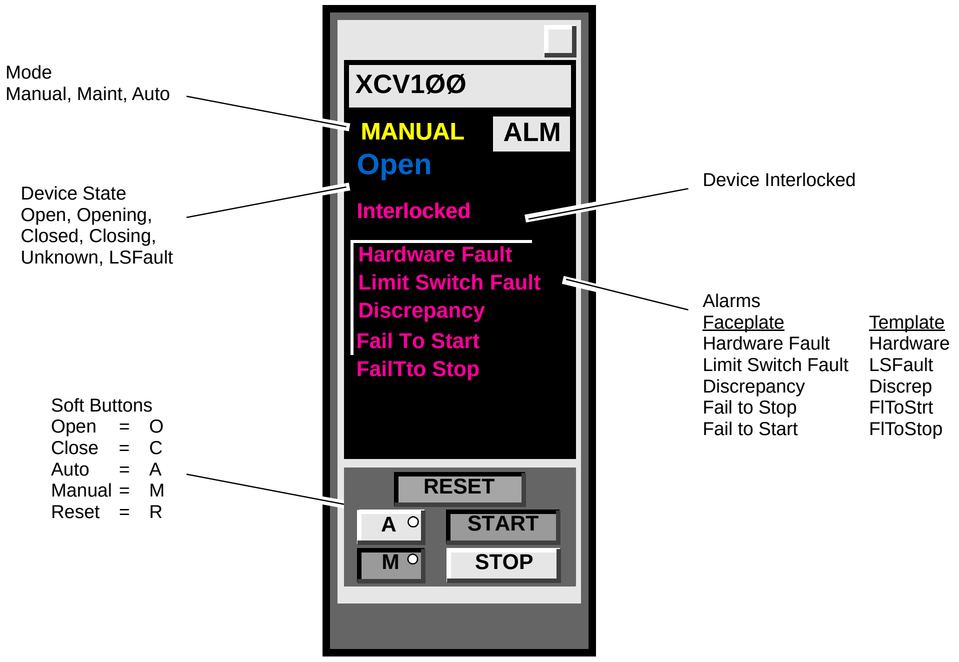
Block Type/ Supervisory computer graphic object	**Vlv1In** HA084012U105 Issue 2	**Vlv2In** HA084012U106 Issue 2	**Vlv3In** HA084012U107 Issue 2			
VlvFPAA HA084012U101 Issue 2	✓	✓				
VlvFPAB HA084012U102 Issue 2			✓			
VlvIcnAA_ha HA084012U103 Issue 2	✓	✓				
VlvIcnAA_vl HA084012U103 Issue 2	✓	✓				
VlvIcnAA_vr HA084012U103 Issue 2	✓	✓				
VlvIcnAB_v1 HA084012U105 Issue 2			✓			
VlvIcnAB_v2 HA084012U104 Issue 2			✓			

Table 13-1: Supported block permutations

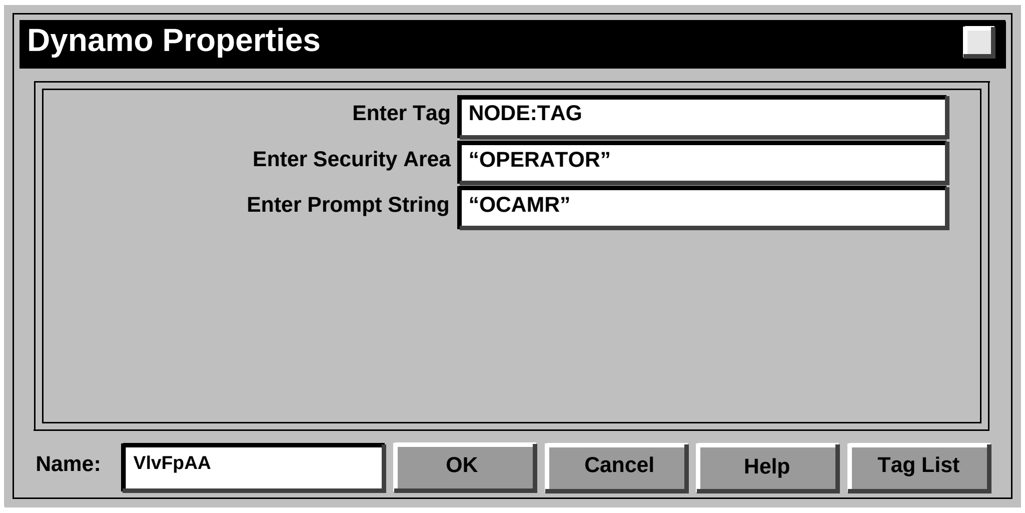
Intentionally left blank

VLV3WAY: VLVFPAA VALVE GRAPHIC OBJECT

Runtime



Configuration



Enter Tag: NODE:TAG

Enter Security Area: “OPERATOR”

Enter Prompt String: “OCAMR”

Name: VlvFpAA

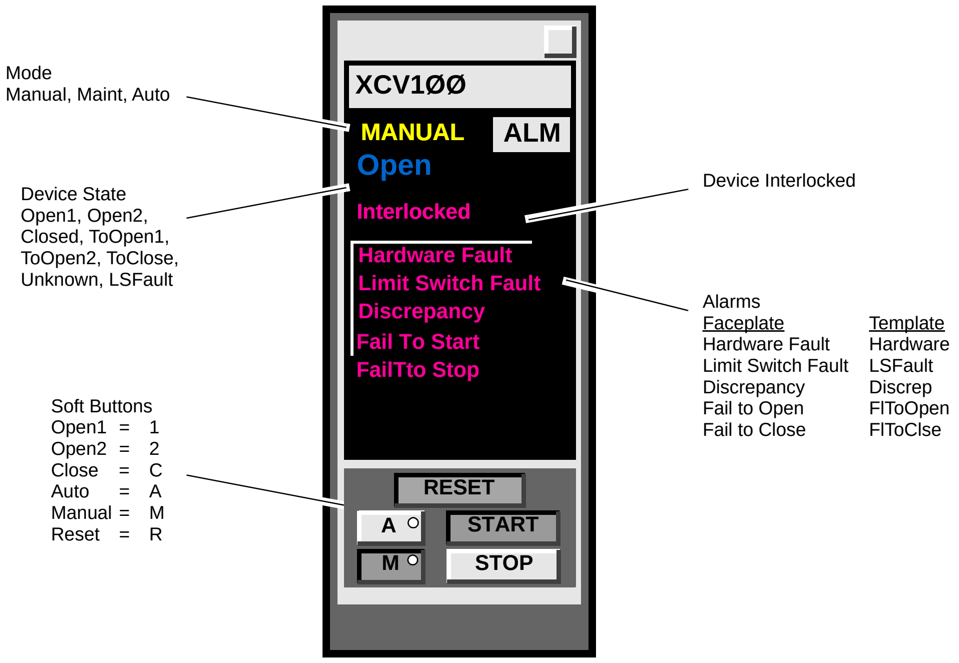
Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Enter Security Area:	Specifies the security area that the operator must have access rights to in order to be able to use the soft buttons. String must be in quotation marks.
Enter Prompt String:	Specifies the soft buttons for which a confirmation prompt is required. Contained within quotation marks are the key letters for each button where confirmation is required.

Table 13-1: Dynamo Properties

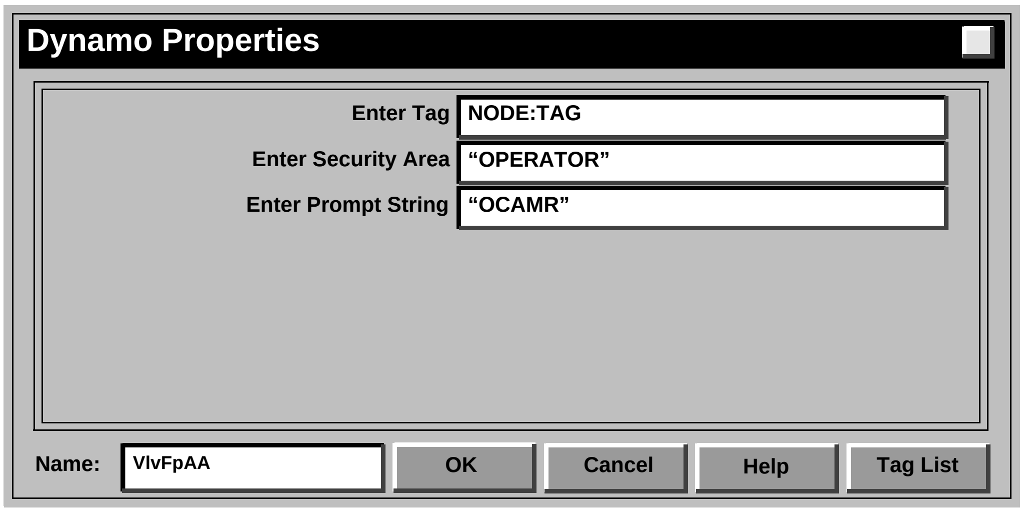
Intentionally left blank

VLV3WAY: VLVFPAB VALVE GRAPHIC OBJECT

Runtime



Configuration



Enter Tag: NODE:TAG
Enter Security Area: "OPERATOR"
Enter Prompt String: "OCAMR"
Name: VlvFpAA

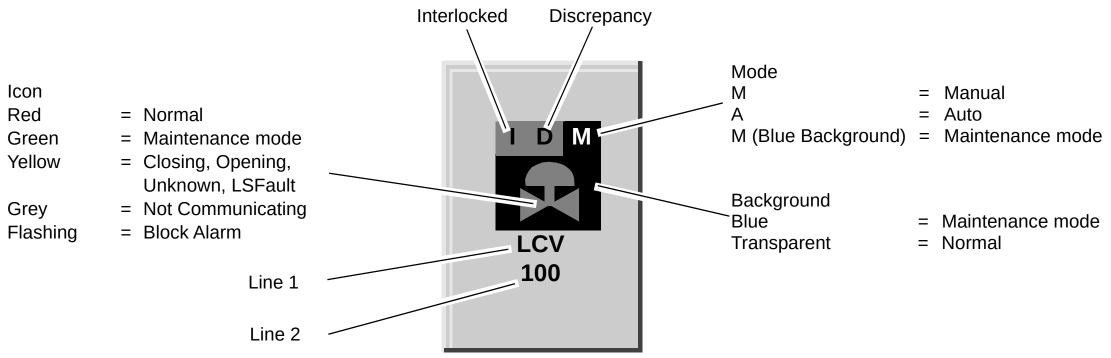
Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Enter Security Area:	Specifies the security area that the operator must have access rights to in order to be able to use the soft buttons. String must be in quotation marks.
Enter Prompt String:	Specifies the soft buttons for which a confirmation prompt is required. Contained within quotation marks are the key letters for each button where confirmation is required.

Table 13-1: Dynamo Properties

Intentionally left blank

VLV3WAY: VLVICNAA VALVE GRAPHIC OBJECT

Runtime



This icon is available in a number of orientations. These are collected within a common dynamo library, ICN_VLV. The dynamos are named as indicated below.

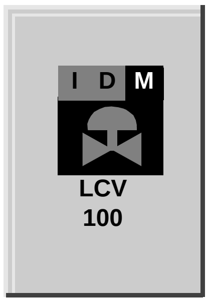
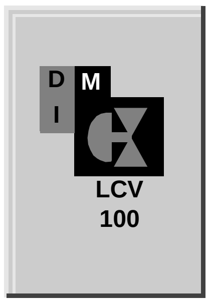
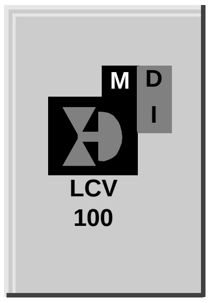
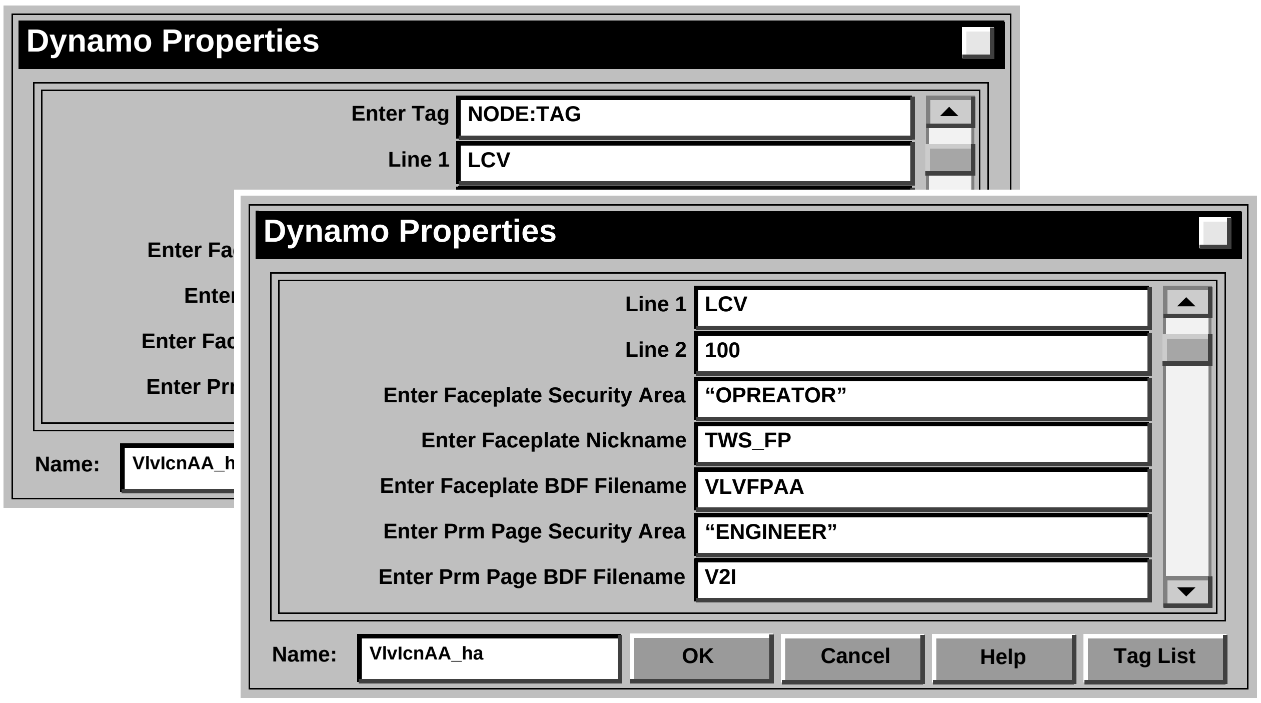
Dynamo name	Function
VlvIcnAA_ha	
VlvIcnAA_vl	
VlvIcnAA_vr	

Table 13-1: Typical graphic object

Configuration


Dynamo Properties

Enter Tag: NODE:TAG

Line 1: LCV

Line 2: 100

Enter Faceplate Security Area: “OPREATOR”

Enter Faceplate Nickname: TWS_FP

Enter Faceplate BDF Filename: VLVFPAA

Enter Prm Page Security Area: “ENGINEER”

Enter Prm Page BDF Filename: V2I

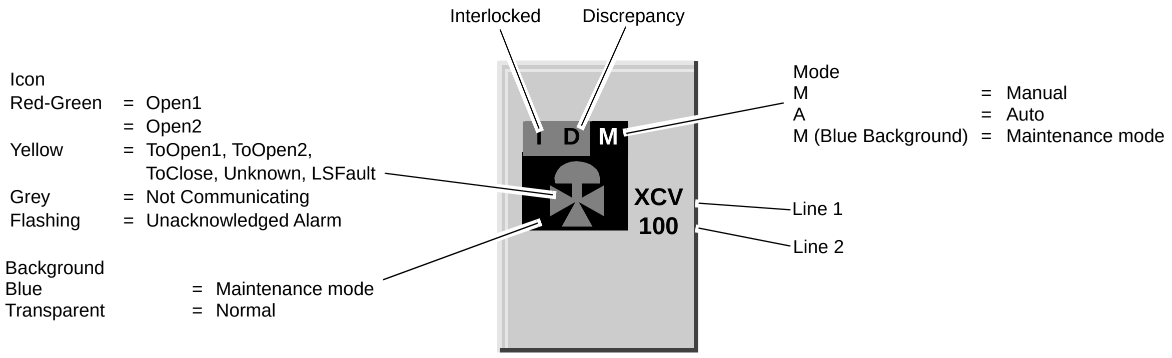
Name: VlvIcnAA_ha

Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Line 1/Line 2:	Textural control module tag or description.
Enter Faceplate Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module faceplate. String must be in quotation marks.
Enter Faceplate Nickname:	Specifies an alias name to the faceplate. The supervisory computer ensures that no two faceplates with the same alias may be simultaneously displayed.
Enter Faceplate BDF Filename:	Specifies the file that is activated as the control module faceplate.
Enter Prm Page Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module engineer page. String must be in quotation marks.
Enter Prm Page BDF Filename:	Specifies the file that is activated as the control module engineer page.

Table 13-2: Dynamo Properties

VLV3WAY: VLVICNAB VALVE GRAPHIC OBJECT

Runtime



This icon is available in a number of orientations. These are collected within a common dynamo library, ICN_VLV. The dynamos are named as indicated below.

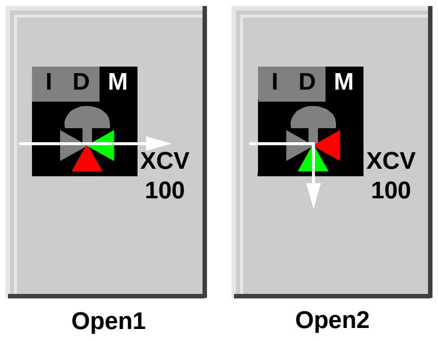
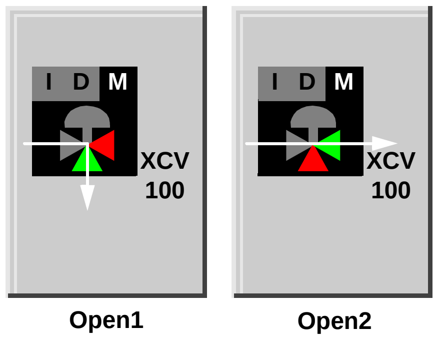
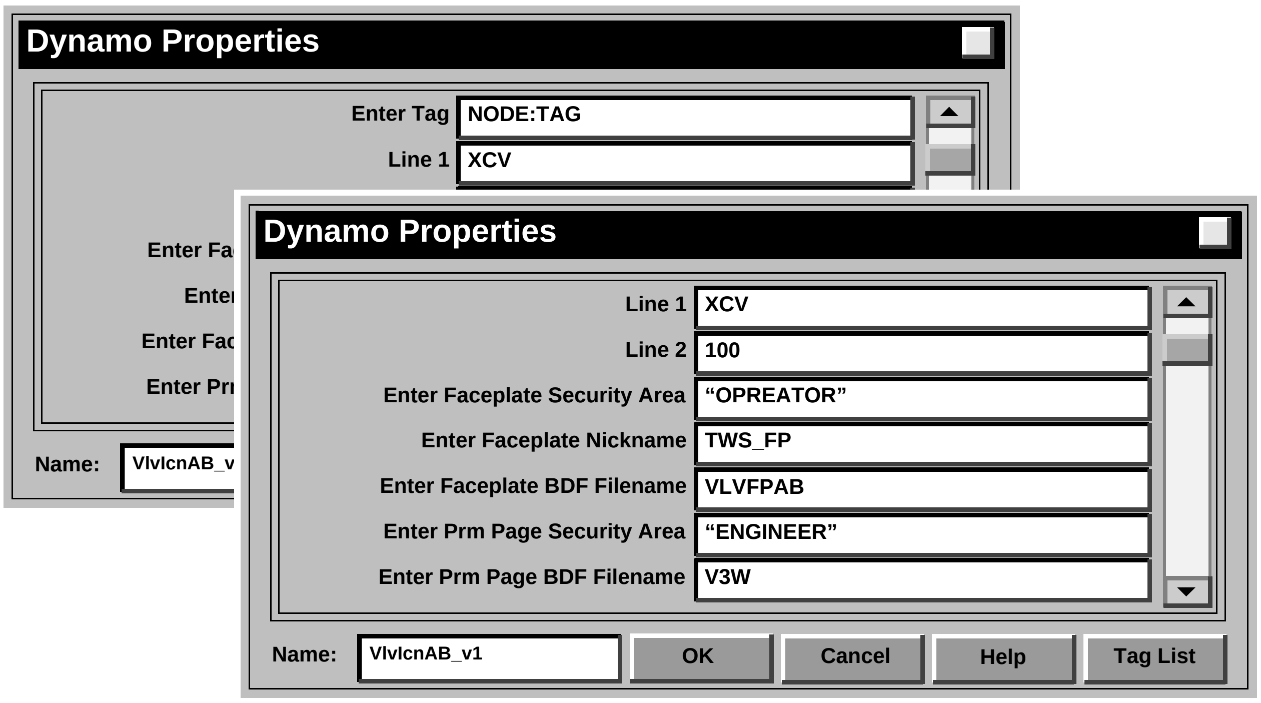
Dynamo name	Function
VlvIcnAB_v1	
VlvIcnAB_v2	

Table 13-1: Typical graphic object

Configuration



Line 1	XCV
Line 2	100
Enter Faceplate Security Area	“OPREATOR”
Enter Faceplate Nickname	TWS_FP
Enter Faceplate BDF Filename	VLVFPAB
Enter Prm Page Security Area	“ENGINEER”
Enter Prm Page BDF Filename	V3W
Name:	VlvlcnAB_v1

Property	Function
Enter Tag:	Specifies the supervisory computer node and control module tag.
Line 1/Line 2:	Textural control module tag or description.
Enter Faceplate Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module faceplate. String must be in quotation marks.
Enter Faceplate Nickname:	Specifies an alias name to the faceplate. The supervisory computer ensures that no two faceplates with the same alias may be simultaneously displayed.
Enter Faceplate BDF Filename:	Specifies the file that is activated as the control module faceplate.
Enter Prm Page Security Area:	Specifies the security area that the operator must have access rights to in order to be able to activate the control module engineer page. String must be in quotation marks.
Enter Prm Page BDF Filename:	Specifies the file that is activated as the control module engineer page.

Table 13-2: Dynamo Properties

VLV3WAY: ON/OFF VALVE; ONE I/P, MAINTAINED OR PULSED O/PS BLOCK

Block function

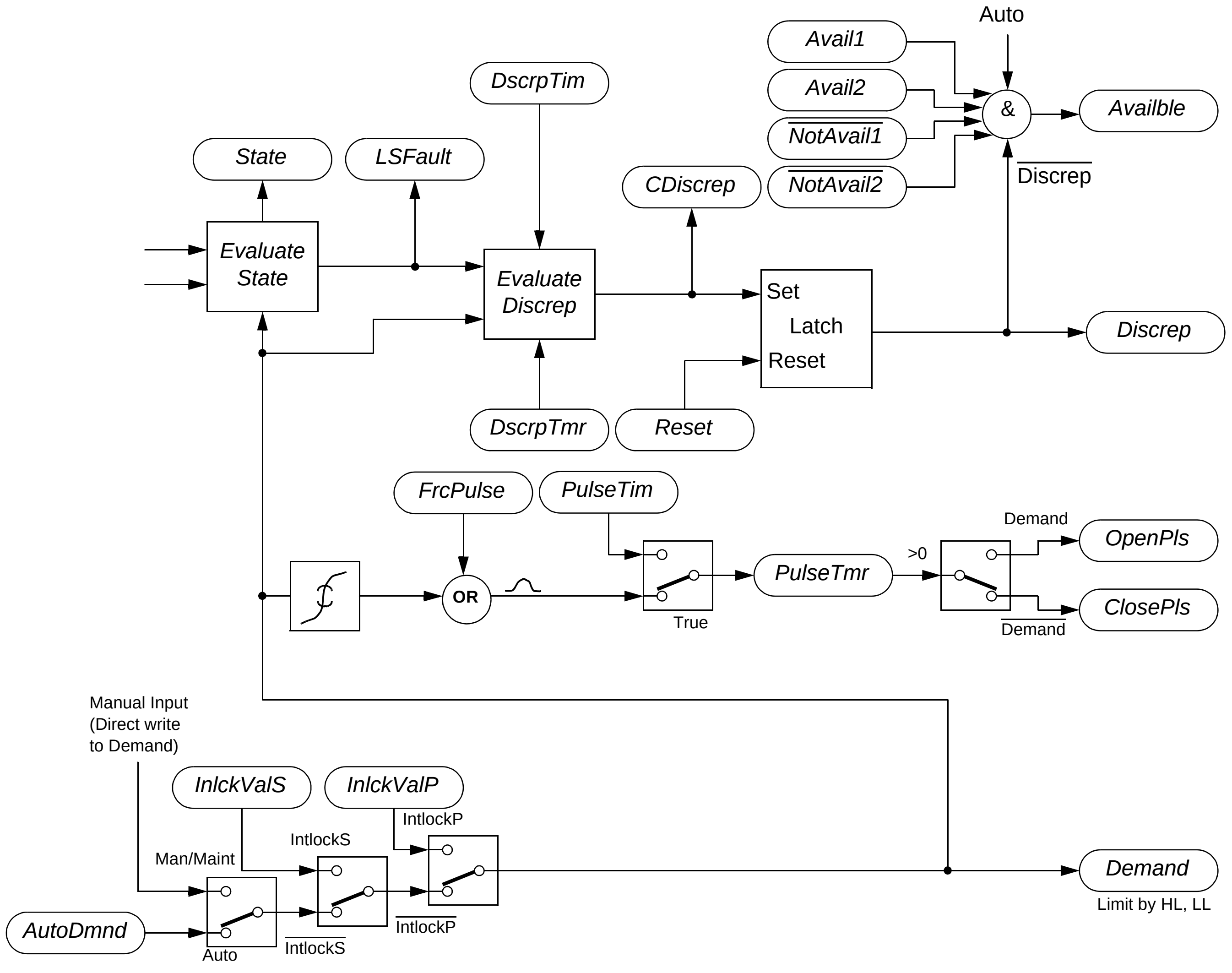
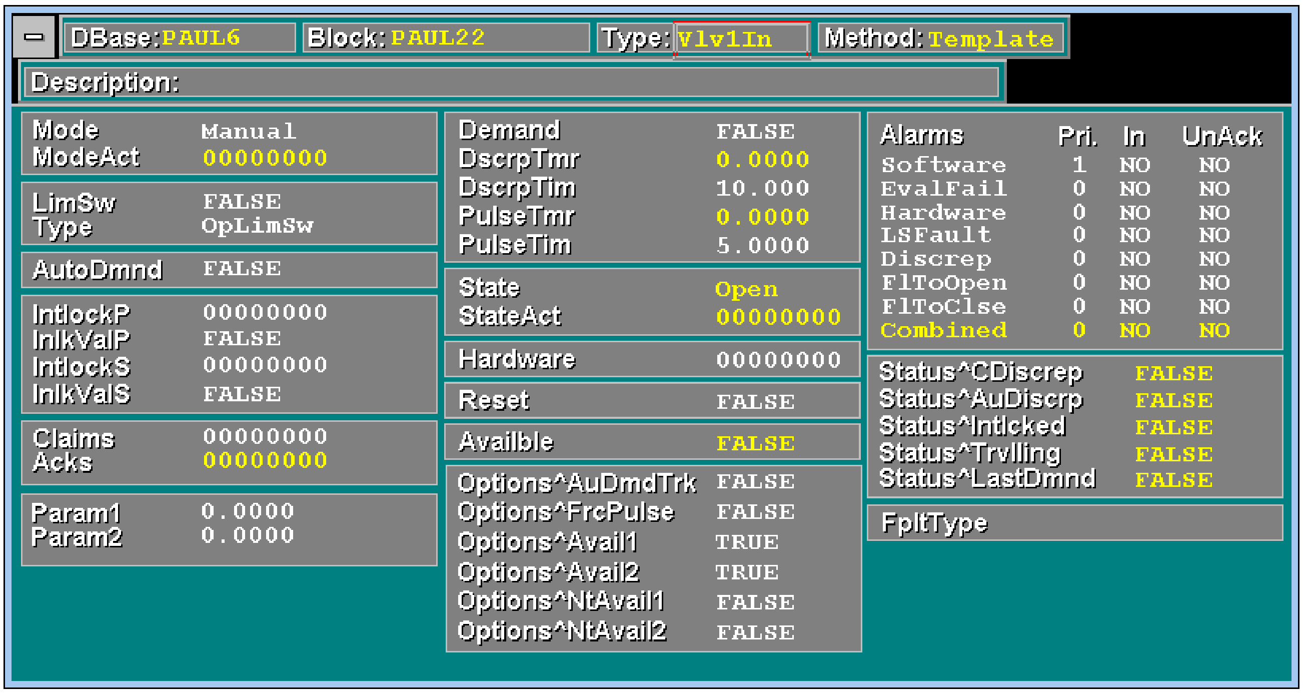


Figure 13-1: Block schematic

The Vlv1In block generates a demand to a single limit switch on/off valve. A choice of maintained output (*Demand*) or pulsed outputs (*StateAct.OpenPls*, *StateAct.ClosePls*) is provided. A single pulse time applies to both open and close pulses.

In manual mode, the valve may be opened and closed by an operator. In automatic mode, the valve is opened and closed by a controlling sequence or other valve user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase:PAUL6   Block:PAUL22   Type:Vlv1In   Method:Template
Description:

Mode        Manual      Demand              FALSE    Alarms      Pri. In  UnAck
ModeAct     00000000    DscrpTmr            0.0000   Software    1    NO  NO
                        DscrpTim            10.000   EvalFail    0    NO  NO
LimSw       FALSE       PulseTmr            0.0000   Hardware    0    NO  NO
Type        OpLimSw     PulseTim            5.0000   LSFault     0    NO  NO
                                                     Discrep     0    NO  NO
AutoDmnd    FALSE       State               Open     FlToOpen    0    NO  NO
                        StateAct            00000000 FlToClse    0    NO  NO
IntlockP    00000000                                 Combined    0    NO  NO
InlkValP    FALSE       Hardware            00000000
IntlockS    00000000                                 Status^CDiscrep   FALSE
InlkValS    FALSE       Reset               FALSE    Status^AuDiscrp   FALSE
                                                     Status^Intlcked   FALSE
Claims      00000000    Availble            FALSE    Status^Trvlling   FALSE
Acks        00000000                                 Status^LastDmnd   FALSE
                        Options^AuDmdTrk    FALSE
Param1      0.0000      Options^FrcPulse    FALSE    FpltType
Param2      0.0000      Options^Avail1      TRUE
                        Options^Avail2      TRUE
                        Options^NtAvail1    FALSE
                        Options^NtAvail2    FALSE
```

Figure 13-2: Engineers page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	▢➧ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
LimSw	Limit switch	T/F	➧▢➧
Type	Limit switch type	Enum	
AutoDmnd	Automatic demand	T/F	➧▢➧
IntlockP	Primary interlock	CD hex	➧▢➧
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 13-1: Block parameters

Parameter	Function	Units	Status
InlkValP	Primary interlock value	Eng	➧▢➧
IntlockS	Secondary interlock	CD hex	➧▢➧
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValS	Secondary interlock value	Eng	➧▢➧
Claim	Resource management claims	CD hex	➧▢➧
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	▢➧ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Param1	User parameter # 1	Eng	➧▢➧
Param2	User parameter # 2	Eng	➧▢➧
Alarms			▢➧ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	I/O hardware fault	T/F	
LSFault	Reserved for future use	T/F	
Discrep	Latched discrepancy	T/F	
FlToOpen	Failed to open	T/F	
FlToClse	Failed to close	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Position demand	T/F	▢➧

Table 13-1: Block parameters

Parameter	Function	Units	Status
DscrpTmr	Discrepancy countdown timer	Secs	▭➧ 📖 ⏱
DscrpTim	Discrepancy (travel) time	Secs	➧▭➧
PulseTmr	Output pulse timer	Secs	▭➧ 📖 ⏱
PulseTim	Output pulse time	Secs	➧▭➧
State	Derived position state	Enum	▭➧ 📖
StateAct	Active position state	CD hex	▭➧ 📖
Open	Open position	T/F	
Closed	Closed position	T/F	
Opening	Opening position	T/F	
Closing	Closing position	T/F	
LSFault	Reserved for future use	T/F	
Unknown	Unknown position	T/F	
OpenPls	Open output pulse	T/F	
ClosePls	Close output pulse	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Status	Operational status bitfields	ABCD hex	➧▭➧
CDiscrep*	Current discrepancy	T/F	1 D
AuDiscrp*	Cannot respond to automatic control	T/F	2 D
Intlcked*	Interlocked	T/F	4 D
Trvlling*	Travelling	T/F	8 D
LastDmnd*	Reserved for future use	T/F	1 C
Bit5	User status #5	T/F	2 C
Bit6	User status #6	T/F	4 C
Bit7	User status #7	T/F	8 C
Bit8	User status #8	T/F	1 B
Bit9	User status #9	T/F	2 B
Bit10	User status #10	T/F	4 B
Bit11	User status #11	T/F	8 B
Bit12	User status #12	T/F	1 A
Bit13	User status #13	T/F	2 A
Bit14	User status #14	T/F	4 A
Discrep*	Latched discrepancy	T/F	8 A
Hardware	I/O hardware failure input	CD hex	➧▭➧
Bit0	Hardware failure #0	T/F	
Bit1	Hardware failure #1	T/F	
Bit2	Hardware failure #2	T/F	
Bit3	Hardware failure #3	T/F	
Bit4	Hardware failure #4	T/F	
Bit5	Hardware failure #5	T/F	
Bit6	Hardware failure #6		
Bit7	Hardware failure #7		
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 13-1: Block parameters

Parameter	Function	Units			Status
Options	Optional configuration bitfields	ABCD hex			
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1	D	
FrcPulse	Force output pulse	T/F	2		
Avail1	User device available #1	T/F	4		
Avail2	User device available #2	T/F	8		
NotAvail1	User device not available #1	T/F	1	C	
NotAvail2	User device not available #2	T/F	2		
Bit6	User option #6	T/F	4		
Bit7	User option #7	T/F	8		
Bit8	User option #8	T/F	1	B	
Bit9	User option #9	T/F	2		
Bit10	User option #10	T/F	4		
Bit11	User option #11	T/F	8		
Bit12	User option #12	T/F	1	A	
Bit13	User option #13	T/F	2		
Bit14	User option #14	T/F	4		
Bit15	User option #15	T/F	8		
Reset	Reset latched discrepancy	T/F			
Availble	Available for automatic control	T/F			
FpltType	Supervisory faceplate type	String			

** Input wiring will disrupt the normal operation of the block*

Table 13-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these 'header' fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

Limit Switch Type. Selects the Limit Switch to either be an Open or a Closed Limit Switch.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

IntlockP. Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the valve. The valve resource may be controlled by up to eight sequences or other valve users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the valve.

Alarms. See page 11-5 in the LIN Block Reference Manual for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the valve.
- **Discrep.** Valve has not responded to the demand or has moved off its limit. Alarm is latched.
- **FlToOpen.** Valve has not responded to an 'Open' demand. Alarm is latched.
- **FlToClse.** Valve has not responded to a 'Close' demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand for the position of the valve. FALSE = 'Close'; TRUE = 'Open'.

DscrpTmr. Discrepancy Timer. Internally set to the discrepancy time on demand change. The discrepancy alarm is raised if this timer counts down to zero and the valve has not reacted to the demand.

DscrpTim. Discrepancy Time. The time given to the valve to react before a discrepancy alarm is raised.

PulseTmr. Pulse Timer. Internally set to the pulse time on demand change. Output pulse is asserted while the timer is counting down to zero.

PulseTim. Pulse Time. The minimum time for which a pulsed output is maintained.

State. (Open/Closed/Opening/Closing/LSFault /Unknown). Current valve state. Derived from the limit switches.

Status. This bitfield shows the status of the valve.

- **CDiscrep.** Current Discrepancy. TRUE when the valve is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the valve is not available or the automatic demand is not equal to the demand. May be used by a controlling sequence or other valve users to determine if the automatic demand is active.
- **Intlcked.** TRUE indicates an asserted interlock.
- **Trvlling.** Internal flag asserted when valve travelling.
- **LastDmnd.** Internal flag indicating demand at last iteration.
- **Discrep.** Latched TRUE when the valve is in discrepancy. Set FALSE during the reset action.
- **Bit5-Bit14.** Optional status bits. Associates additional Boolean data with the valve.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the valve.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **FrcPulse.** TRUE forces the control module to re-transmit the last output pulse. Internally set FALSE after the output pulse is started.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **Bit7-Bit15.** Optional option bits. Associates additional Boolean data with the valve.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when valve is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the valve. Allows valves of this type to be represented by different supervisory computer faceplates.

Implementation notes

If Vlv1In control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2294 bytes and by 132 bytes for each instance of the control module.

Intentionally left blank

VLV3WAY: ON/OFF VALVE; TWO I/PS, MAINTAINED OR PULSED O/PS BLOCK

Block function

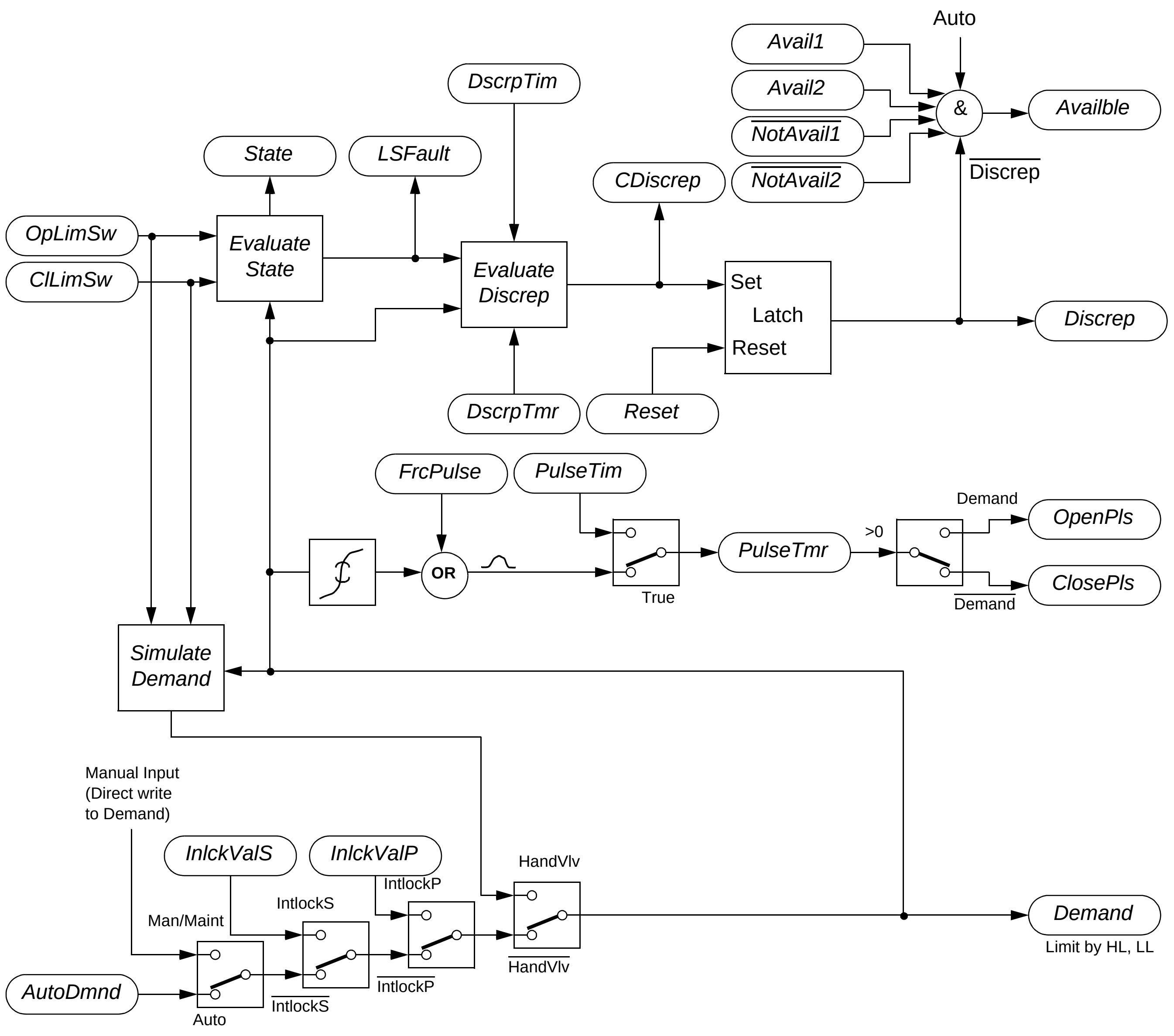
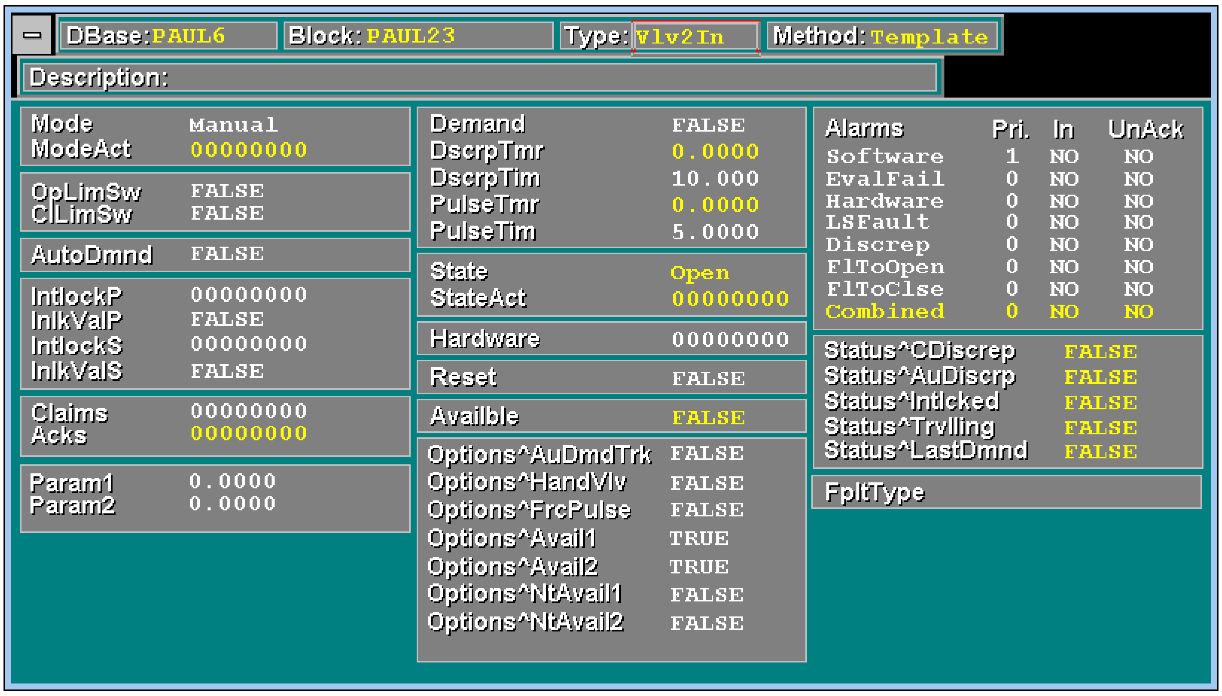


Figure 13-1: Block schematic

Please refer to the schematic. The Vlv2In block generates a demand to a dual limit switch on/off valve. A choice of maintained output (*Demand*) or pulsed outputs (*StateAct.OpenPls*, *StateAct.ClosePls*) is provided. A single pulse time applies to both open and close pulses.

In manual mode, the valve may be opened and closed by an operator. In automatic mode, the valve is opened and closed by a controlling sequence or other valve user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase:PAUL6   Block:PAUL23   Type:Vlv2In   Method:Template
Description:

Mode      Manual      Demand    FALSE     Alarms     Pri. In  UnAck
ModeAct   00000000    DscrpTmr  0.0000    Software   1    NO  NO
                      DscrpTim  10.000    EvalFail   0    NO  NO
OpLimSw   FALSE       PulseTmr  0.0000    Hardware   0    NO  NO
ClLimSw   FALSE       PulseTim  5.0000    LSFault    0    NO  NO
                                          Discrep    0    NO  NO
AutoDmnd  FALSE       State     Open      FlToOpen   0    NO  NO
                      StateAct  00000000  FlToClse   0    NO  NO
IntlockP  00000000                        Combined   0    NO  NO
InlkValP  FALSE       Hardware  00000000
IntlockS  00000000                        Status^CDiscrep  FALSE
InlkValS  FALSE       Reset     FALSE     Status^AuDiscrp  FALSE
                                          Status^Intlcked  FALSE
Claims    00000000    Availble  FALSE     Status^Trvlling  FALSE
Acks      00000000                        Status^LastDmnd  FALSE
                      Options^AuDmdTrk  FALSE
Param1    0.0000      Options^HandVlv   FALSE   FpltType
Param2    0.0000      Options^FrcPulse  FALSE
                      Options^Avail1    TRUE
                      Options^Avail2    TRUE
                      Options^NtAvail1  FALSE
                      Options^NtAvail2  FALSE
```

Figure 13-2: Engineer’s page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block’s ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	➔ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
OpLimSw	Open limit switch	T/F	➔➔
ClLimSw	Close limit switch	Enum	
AutoDmnd	Automatic demand	T/F	➔➔
IntlockP	Primary interlock	CD hex	➔➔
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 13-1: Block parameters

Parameter	Function	Units	Status
InlkValP	Primary interlock value	Eng	
IntlockS	Secondary interlock	CD hex	
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	Ø Ø Ø Ø Ø Ø Ø Ø
InlkValS	Secondary interlock value	Eng	
Claim	Resource management claims	CD hex	
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	Ø Ø Ø Ø Ø Ø Ø Ø
Param1	User parameter # 1	Eng	
Param2	User parameter # 2	Eng	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block's internal logic	T/F	
Hardware	I/O hardware fault	T/F	
LSFault	Reserved for future use	T/F	
Discrep	Latched discrepancy	T/F	
FlToOpen	Failed to open	T/F	
FlToClse	Failed to close	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Position demand	T/F	

Table 13-1: Block parameters

Parameter	Function	Units	Status		
DscrpTmr	Discrepancy countdown timer	Secs	⇨	📖	🕒
DscrpTim	Discrepancy (travel) time	Secs	⇄		
PulseTmr	Output pulse timer	Secs	⇨	📖	🕒
PulseTim	Output pulse time	Secs	⇄		
State	Derived position state	Enum	⇨	📖	
StateAct	Active position state	CD hex	⇨	📖	
Open	Open position	T/F			
Closed	Closed position	T/F			
Opening	Opening position	T/F			
Closing	Closing position	T/F			
LSFault	Reserved for future use	T/F			
Unknown	Unknown position	T/F			
OpenPls	Open output pulse	T/F			
ClosePls	Close output pulse	T/F			
			Ø Ø Ø Ø Ø Ø Ø Ø		
Status	Operational status bitfields	ABCD hex	⇄		
CDiscrep*	Current discrepancy	T/F	1		
AuDiscrp*	Cannot respond to automatic control	T/F	2	D	
Intlcked*	Interlocked	T/F	4		
Trvlling*	Travelling	T/F	8		
LastDmnd*	Reserved for future use	T/F	1		
Bit5	User status #5	T/F	2	C	
Bit6	User status #6	T/F	4		
Bit7	User status #7	T/F	8		
Bit8	User status #8	T/F	1		
Bit9	User status #9	T/F	2	B	
Bit10	User status #10	T/F	4		
Bit11	User status #11	T/F	8		
Bit12	User status #12	T/F	1		
Bit13	User status #13	T/F	2	A	
Bit14	User status #14	T/F	4		
Discrep*	Latched discrepancy	T/F	8		
Hardware	I/O hardware failure input	CD hex	⇄		
Bit0	Hardware failure #0	T/F			
Bit1	Hardware failure #1	T/F			
Bit2	Hardware failure #2	T/F			
Bit3	Hardware failure #3	T/F			
Bit4	Hardware failure #4	T/F			
Bit5	Hardware failure #5	T/F			
Bit6	Hardware failure #6				
Bit7	Hardware failure #7				
			Ø Ø Ø Ø Ø Ø Ø Ø		

Table 13-1: Block parameters

Parameter	Function	Units			Status
Options	Optional configuration bitfields	ABCD hex			➧☐➧
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1	D	
HandVlv	Interface to hand valve (i.e. I/P only)	T/F	2		
FrcPulse	Force output pulse	T/F	4		
Avail1	User device available #1	T/F	8		
Avail2	User device available #2	T/F	1	C	
NotAvail1	User device not available #1	T/F	2		
NotAvail2	User device not available #2	T/F	4		
Bit7	User option #7	T/F	8		
Bit8	User option #8	T/F	1	B	
Bit9	User option #9	T/F	2		
Bit10	User option #10	T/F	4		
Bit11	User option #11	T/F	8		
Bit12	User option #12	T/F	1	A	
Bit13	User option #13	T/F	2		
Bit14	User option #14	T/F	4		
Bit15	User option #15	T/F	8		
Reset	Reset latched discrepancy	T/F			➧☐➧
Availble	Available for automatic control	T/F			☐➧ 🕮
FpltType	Supervisory faceplate type	String			

** Input wiring will disrupt the normal operation of the block*

Table 13-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) 375U003, for details of these 'header' fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

AutoDmnd. Automatic Demand. Controls the demand in Automatic mode.

Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the valve. The valve resource may be controlled by up to eight sequences or other valve users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the valve.

Alarms. See page 11-5 in LIN Block Reference Manual for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the valve.
- **LSFault.** Limit switches indicate valve is both 'Open' and 'Closed'.
- **Discrep.** Valve has not responded to the demand or has moved off its limit. Alarm is latched.
- **FlToOpen.** Valve has not responded to an 'Open' demand. Alarm is latched.
- **FlToClse.** Valve has not responded to a 'Close' demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. Maintained demand for the position of the valve. FALSE = 'Close'; TRUE = 'Open'.

DscrpTmr. Discrepancy Timer. Internally set to the discrepancy time on demand change. The discrepancy alarm is raised if this timer counts down to zero and the valve has not reacted to the demand.

DscrpTim. Discrepancy Time. The time given to the valve to react before a discrepancy alarm is raised.

PulseTmr. Pulse Timer. Internally set to the pulse time on demand change. Output pulse is asserted while the timer is counting down to zero.

PulseTim. Pulse Time. The minimum time for which a pulsed output is maintained.

State. (Open/Closed/Opening/Closing/LSFault /Unknown). Current valve state. Derived from the limit switches.

Status. This bitfield shows the status of the valve.

- **CDiscrep.** Current Discrepancy. TRUE when the valve is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the valve is not available or the automatic demand is not equal to the demand. May be used by a controlling sequence or other valve users to determine if the automatic demand is active.
- **Intlcked.** TRUE indicates an asserted interlock.
- **Trvlling.** Internal flag asserted when valve travelling.
- **LastDmnd.** Internal flag indicating demand at last iteration.
- **Discrep.** Latched TRUE when the valve is in discrepancy. Set FALSE during the reset action.
- **Bit5-Bit14.** Optional status bits. Associates additional Boolean data with the valve.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the valve.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **HandVlv.** TRUE forces the demand to change automatically in response to the valve moving off its limit. Permits the block to interface to a hand valve, where a discrepancy alarm is required when the valve travelling time exceeds the discrepancy time.
- **FrcPulse.** TRUE forces the control module to re-transmit the last output pulse. Internally set FALSE after the output pulse is started.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **Bit7-Bit15.** Optional option bits. Associates additional Boolean data with the valve.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when valve is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the valve. Allows valves of this type to be represented by different supervisory computer faceplates.

Implementation notes

If Vlv2In control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2298 bytes and by 132 bytes for each instance of the control module.

Intentionally left blank

VLV3WAY: THREE WAY VALVE; THREE I/PS, MAINTAINED OR PULSED O/PS BLOCK

Block function

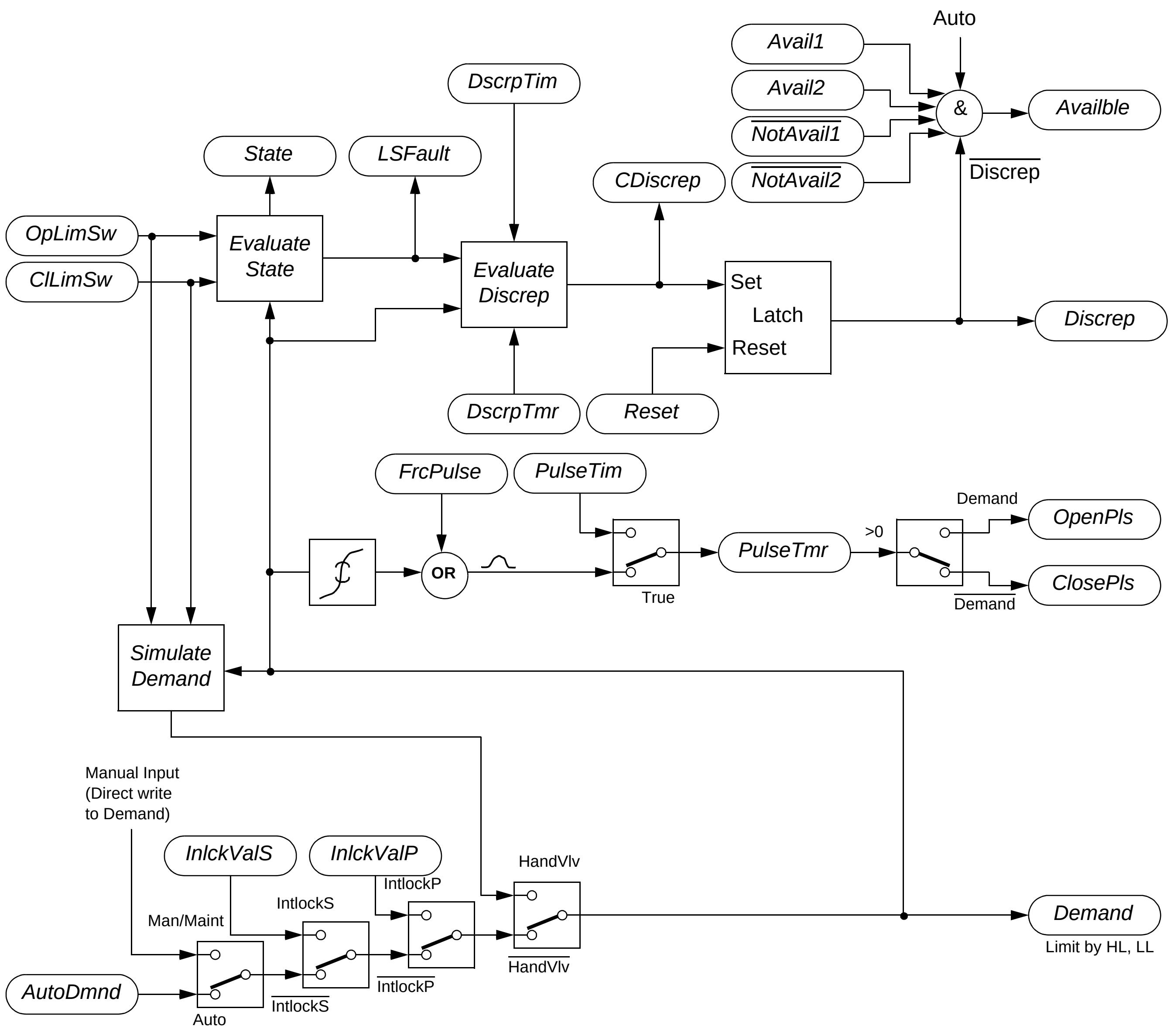
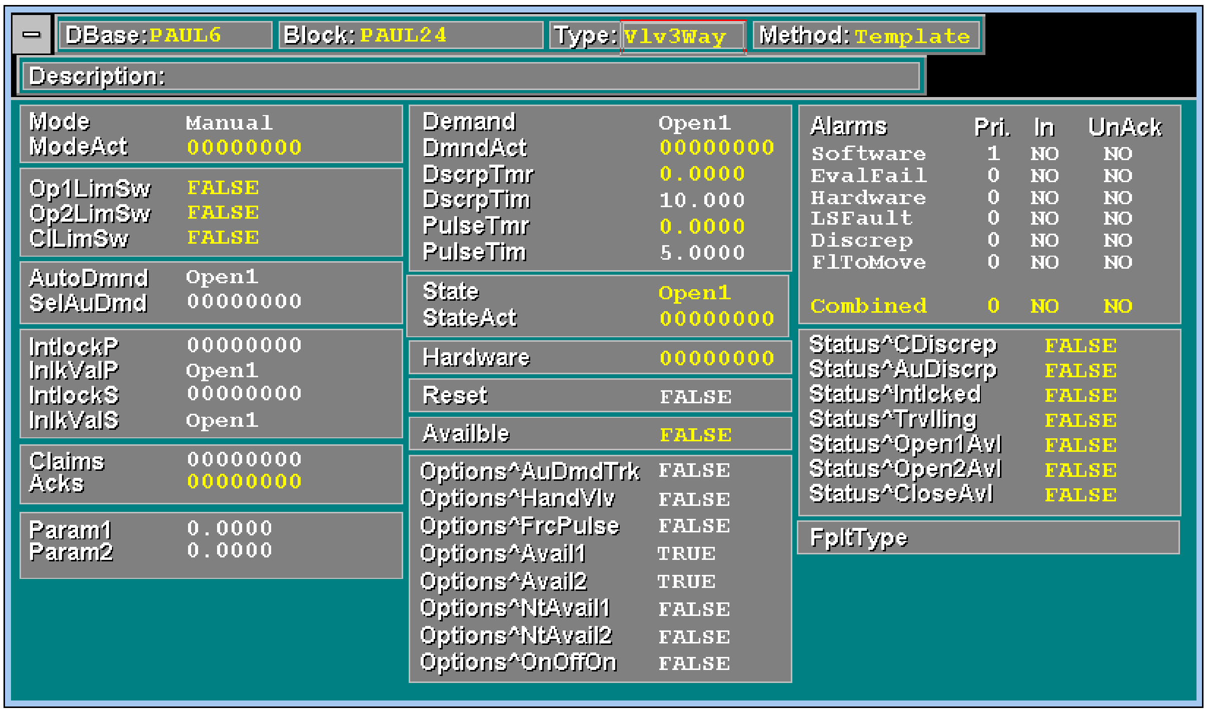


Figure 13-1: Block schematic

Please refer to the schematic. The Vlv3Way block generates a demand to a triple limit switch three-way valve. A choice of maintained output (*Open1Act*, *Open2Act*, *CloseAct*) or pulsed outputs (*Open1Pls*, *Open2Pls*, *ClosePls*) is provided. A single pulse time applies to all pulses.

In manual mode, the valve may be moved by an operator. In automatic mode, the valve is moved by a controlling sequence or other valve user. Maintenance mode is often used to indicate caution in operation but functions identically to manual mode.



```
DBase:PAUL6   Block:PAUL24   Type:Vlv3Way   Method:Template
Description:

Mode       Manual      Demand              Open1      Alarms     Pri.  In  UnAck
ModeAct    00000000    DmndAct             00000000   Software   1     NO  NO
                       DscrpTmr            0.0000     EvalFail   0     NO  NO
Op1LimSw   FALSE       DscrpTim            10.000     Hardware   0     NO  NO
Op2LimSw   FALSE       PulseTmr            0.0000     LSFault    0     NO  NO
ClLimSw    FALSE       PulseTim            5.0000     Discrep    0     NO  NO
                                                      FlToMove   0     NO  NO
AutoDmnd   Open1       State               Open1
SelAuDmd   00000000    StateAct            00000000   Combined   0     NO  NO

IntlockP   00000000    Hardware            00000000   Status^CDiscrep   FALSE
InlkValP   Open1                                      Status^AuDiscrp   FALSE
IntlockS   00000000    Reset               FALSE      Status^Intlcked   FALSE
InlkValS   Open1                                      Status^Trvlling   FALSE
                       Availble            FALSE      Status^Open1Avl   FALSE
Claims     00000000                                   Status^Open2Avl   FALSE
Acks       00000000    Options^AuDmdTrk    FALSE      Status^CloseAvl   FALSE
                       Options^HandVlv     FALSE
Param1     0.0000      Options^FrcPulse    FALSE      FpltType
Param2     0.0000      Options^Avail1      TRUE
                       Options^Avail2      TRUE
                       Options^NtAvail1    FALSE
                       Options^NtAvail2    FALSE
                       Options^OnOffOn     FALSE
```

Figure 13-2: Engineers page

Block parameters

Symbols used in the Block parameter table are explained in Chapter 1 of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Method	Reserved for future use block's ST update routine (default = Template)	Enum	📖
Mode	Current operating mode	Enum	
ModeAct	Mode active	CD hex	➧ 📖
ManAct	Manual mode active	T/F	
MaintAct	Maintenance mode active	T/F	
AutoAct	Auto mode active	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
OpLimSw	Open limit switch	T/F	➧➧
ClLimSw	Close limit switch	Enum	
AutoDmnd	Automatic demand	T/F	➧➧
IntlockP	Primary interlock	CD hex	➧➧
Ilk0	Primary interlock #0	T/F	
Ilk1	Primary interlock #1	T/F	
Ilk2	Primary interlock #2	T/F	
Ilk3	Primary interlock #3	T/F	
Ilk4	Primary interlock #4	T/F	
Ilk5	Primary interlock #5	T/F	
Ilk6	Primary interlock #6	T/F	
Ilk7	Primary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 13-1: Block parameters

Parameter	Function	Units	Status
InlkValP	Primary interlock value	Eng	▶□▶
IntlockS	Secondary interlock	CD hex	▶□▶
Ilk0	Secondary interlock #0	T/F	
Ilk1	Secondary interlock #1	T/F	
Ilk2	Secondary interlock #2	T/F	
Ilk3	Secondary interlock #3	T/F	
Ilk4	Secondary interlock #4	T/F	
Ilk5	Secondary interlock #5	T/F	
Ilk6	Secondary interlock #6	T/F	
Ilk7	Secondary interlock #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
InlkValS	Secondary interlock value	Eng	▶□▶
Claim	Resource management claims	CD hex	▶□▶
Usr0	Claim #0	T/F	
Usr1	Claim #1	T/F	
Usr2	Claim #2	T/F	
Usr3	Claim #3	T/F	
Usr4	Claim #4	T/F	
Usr5	Claim #5	T/F	
Usr6	Claim #6	T/F	
Usr7	Claim #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Acks	Resource management acknowledgements	CD hex	□▶ 📖
Usr0	Acknowledgement #0	T/F	
Usr1	Acknowledgement #1	T/F	
Usr2	Acknowledgement #2	T/F	
Usr3	Acknowledgement #3	T/F	
Usr4	Acknowledgement #4	T/F	
Usr5	Acknowledgement #5	T/F	
Usr6	Acknowledgement #6	T/F	
Usr7	Acknowledgement #7	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Param1	User parameter # 1	Eng	▶□▶
Param2	User parameter # 2	Eng	▶□▶
Alarms			□▶ 📖 🔊
Software	Block RAM data sumcheck error/network failure	T/F	
EvalFail	Evaluation failure in block’s internal logic	T/F	
Hardware	I/O hardware fault	T/F	
LSFault	Reserved for future use	T/F	
Discrep	Latched discrepancy	T/F	
FlToOpen	Failed to open	T/F	
FlToClse	Failed to close	T/F	
Combined	OR-ing of all alarm bits	T/F	
Demand	Position demand	T/F	□▶

Table 13-1: Block parameters

Parameter	Function	Units	Status
DscrpTmr	Discrepancy countdown timer	Secs	□▶ 📖 🕒
DscrpTim	Discrepancy (travel) time	Secs	▶□▶
PulseTmr	Output pulse timer	Secs	□▶ 📖 🕒
PulseTim	Output pulse time	Secs	▶□▶
State	Derived position state	Enum	□▶ 📖
StateAct	Active position state	CD hex	□▶ 📖
Open	Open position	T/F	
Closed	Closed position	T/F	
Opening	Opening position	T/F	
Closing	Closing position	T/F	
LSFault	Reserved for future use	T/F	
Unknown	Unknown position	T/F	
OpenPls	Open output pulse	T/F	
ClosePls	Close output pulse	T/F	
			Ø Ø Ø Ø Ø Ø Ø Ø
Status	Operational status bitfields	ABCD hex	▶□▶
CDiscrep*	Current discrepancy	T/F	1
AuDiscrp*	Cannot respond to automatic control	T/F	2 D
Intlcked*	Interlocked	T/F	4
Trvlling*	Travelling	T/F	8
LastDmnd*	Reserved for future use	T/F	1
Bit5	User status #5	T/F	2 C
Bit6	User status #6	T/F	4
Bit7	User status #7	T/F	8
Bit8	User status #8	T/F	1
Bit9	User status #9	T/F	2 B
Bit10	User status #10	T/F	4
Bit11	User status #11	T/F	8
Bit12	User status #12	T/F	1
Bit13	User status #13	T/F	2 A
Bit14	User status #14	T/F	4
Discrep*	Latched discrepancy	T/F	8
Hardware	I/O hardware failure input	CD hex	▶□▶
Bit0	Hardware failure #0	T/F	
Bit1	Hardware failure #1	T/F	
Bit2	Hardware failure #2	T/F	
Bit3	Hardware failure #3	T/F	
Bit4	Hardware failure #4	T/F	
Bit5	Hardware failure #5	T/F	
Bit6	Hardware failure #6		
Bit7	Hardware failure #7		
			Ø Ø Ø Ø Ø Ø Ø Ø

Table 13-1: Block parameters

Parameter	Function	Units		Status
Options	Optional configuration bitfields	ABCD hex		
AuDmdTrk	AutoDmnd track Demand when not Auto	T/F	1 D	
HandVlv	Interface to hand valve (i.e. I/P only)	T/F	2 D	
FrcPulse	Force output pulse	T/F	4 D	
Avail1	User device available #1	T/F	8 D	
Avail2	User device available #2	T/F	1 C	
NotAvail1	User device not available #1	T/F	2 C	
NotAvail2	User device not available #2	T/F	4 C	
Bit7	User option #7	T/F	8 C	
Bit8	User option #8	T/F	1 B	
Bit9	User option #9	T/F	2 B	
Bit10	User option #10	T/F	4 B	
Bit11	User option #11	T/F	8 B	
Bit12	User option #12	T/F	1 A	
Bit13	User option #13	T/F	2 A	
Bit14	User option #14	T/F	4 A	
Bit15	User option #15	T/F	8 A	
Reset	Reset latched discrepancy	T/F		
Availble	Available for automatic control	T/F		
FpltType	Supervisory faceplate type	String		

** Input wiring will disrupt the normal operation of the block*

Table 13-1: Block parameters

Block specification menu

Dbase, Block, Type. See section 1.3 of this document(Control Modules User Guide, HA084012) for details of these ‘header’ fields.

Mode. (Manual/Maint/Auto). Selects current operating mode.

AutoDmnd. (Open1/Open2/Close). Automatic Demand. Controls the demand in Automatic mode.

SelAuDmd. Selects automatic demand via digital inputs from the strategy.

SelOpen1	SelOpen2	SelClose	AutoDmnd
FALSE	FALSE	TRUE	Close
FALSE	TRUE	FALSE	Open2
FALSE	TRUE	TRUE	Close
TRUE	FALSE	FALSE	Open1
TRUE	FALSE	TRUE	Close
TRUE	TRUE	FALSE	Open2
TRUE	TRUE	TRUE	Close

Table 13-2: Function

IntlockP. Primary Interlock. Asserted if one or more of the bits are set TRUE; usually via digital inputs from the strategy.

InlkValP. (Open1/Open2/Close). Primary Interlock Value. Controls the demand when the primary interlock is asserted, regardless of the secondary interlock.

IntlockS. Secondary Interlock. Asserted if one or more of the bits are set TRUE and the primary interlock is not asserted.

InlkValS. (Open1/Open2/Close). Secondary Interlock Value. Controls the demand when the secondary interlock is asserted.

Claims. Used to request sole control of the valve. The valve resource may be controlled by up to eight sequences or other valve users.

Acks. Acknowledgements. Indicates acceptance of a claim.

Param1, Param2. Optional Parameter. Associates additional floating-point data with the valve.

Alarms.

- **Software.** Sumcheck error in block's RAM data.
- **EvalFail.** Evaluation failure in block's internal logic.
- **Hardware.** Hardware failure flagged by an input or output block associated with the valve.
- **LSFault.** Limit switches indicate valve is both 'Open' and 'Closed'.
- **Discrep.** Valve has not responded to the demand or has moved off its limit. Alarm is latched.
- **FlToMove.** Valve has not responded to a change in demand. Alarm is latched.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the highest priority active alarm in the block.

Demand. (Open1/Open2/Close). Maintained demand for the position of the valve.

DscrpTmr. Discrepancy Timer. Internally set to the discrepancy time on demand change. The discrepancy alarm is raised if this timer counts down to zero and the valve has not reacted to the demand.

DscrpTim. Discrepancy Time. The time given to the valve to react before a discrepancy alarm is raised.

PulseTmr. Pulse Timer. Internally set to the pulse time on demand change. Output pulse is asserted while the timer is counting down to zero.

PulseTim. Pulse Time. The minimum time for which a pulsed output is maintained.

State. (Open1/Open2/Closed/Opening/Closing/LSFault /Unknown). Current valve state. Derived from the limit switches.

Status. This bitfield shows the status of the valve.

- **CDiscrep.** Current Discrepancy. TRUE when the valve is actually in discrepancy.
- **AuDiscrp.** Automatic Discrepancy. TRUE when the valve is not available or the automatic demand is not equal to the demand. May be used by a controlling sequence or other valve users to determine if the automatic demand is active.
- **Intlcked.** TRUE indicates an asserted interlock.
- **Trvlling.** Internal flag asserted when valve travelling.
- **LastOP1,LastOP2,LastCls.** Internal flags indicating demand at last iteration.
- **Open1Avl,Open2Avl,CloseAvl.** Indicates the next realisable demand based upon the 'OnOffOn' option setting.

'OnOffOn' option FALSE		**'OnOffOn' option TRUE**	

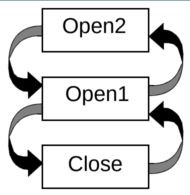
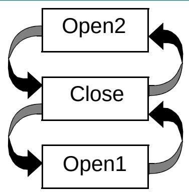
	Open1Avl = TRUE
	
	Open2Avl = TRUE
	CloseAvl = TRUE
	
	Open1Avl = TRUE
			CloseAvl = TRUE
			
			Open1Avl = TRUE
			Open2Avl = TRUE
			
			CloseAvl = TRUE

Table 13-3: Status OnOffOn function

- **Discrep.** Latched TRUE when the valve is in discrepancy. Set FALSE during the reset action.
- **Bit10-Bit14.** Optional status bits. Associates additional Boolean data with the valve.

Hardware. Asserts hardware alarm if one or more of the bits are TRUE.

Options. This bitfield allows inputs to control the operation of the valve.

- **AuDmdTrk.** TRUE forces the automatic demand to track the demand when not operating in auto or when interlocked.
- **HandVlv.** TRUE forces the demand to change automatically in response to the valve moving off its limit. Permits the block to interface to a hand valve, where a discrepancy alarm is required when the valve travelling time exceeds the discrepancy time.
- **FrcPulse.** TRUE forces the control modules to re-transmit the last output pulse. Internally set FALSE after the output pulse is started.
- **Avail1, Avail2.** FALSE prevents the block from signalling that it is available for automatic control.
- **NtAvail1, NtAvail2.** TRUE prevents the block from signalling that it is available for automatic control.
- **OnOffOn.** TRUE specifies that the valve has a centre off position.
- **Bit7-Bit15.** Optional option bits. Associates additional Boolean data with the valve.

Reset. TRUE resets any of the latched alarms. Internally set FALSE after the reset action.

Availble. TRUE when valve is available for automatic control, i.e. in automatic mode, no latched discrepancy alarm and availability options not prohibitive.

FpltType. Faceplate Type. Associate a faceplate type or other alphanumeric string with the valve. Allows valves of this type to be represented by different supervisory computer faceplates.

Implementation notes

If Vlv3Way control modules are included in the configuration, the block is included in the database as a foreign template. The runtime database size is increased by a template overhead of 2668 bytes and by 138 bytes for each instance of the control module.

Intentionally left blank

Index

Scan for local contents

Eurotherm Ltd
Faraday Close
Durrington
Worthing
West Sussex
BN13 3PL
Phone: +44 (0) 1903 268500
www.eurotherm.co.uk

Schneider Electric, Life Is On, Eurotherm, EurothermSuite, Wonderware, InTouch, eCAT, EFit, EPack, EPower,Eycon, Eyris, Chessell, Mini8, nanodac, optivis, piccolo, and versadac are trademarks of Schneider Electric SE, its subsidiaries and affiliated companies. All other trademarks are the property of their respective owners.

HA084012U003 Issue 9 (CN36225)

© 2017 Schneider Electric. All Rights Reserved.