–METASYS®

GPL Programmer's Manual

GPL Programmer's Manual

Introduction

* Indicates those sections where changes have occurred since the last printing.

Welcome to Graphic Programming

Welcome to graphic programming and to the Graphic Programming Language (GPL). It is the standard Metasys® software language for creating control strategies that contain software objects and processes. It is a standalone software package that runs on the Operator Workstation. GPL is designed as an efficient, easy-to-use graphic tool for application engineers, system representatives, and facility operators.

GPL, as its name implies, is graphically oriented. It uses symbols to represent actual control programs. The GPL program, called a control strategy, looks much like an electronic or pneumatic control drawing. The control strategy consists of diagrams, and the diagrams are the programs. If you can read flowcharts and pneumatic control diagrams, you will have little difficulty with reading GPL diagrams.

Graphic programming recognizes that the human mind understands complex relationships more easily if they are presented in symbols. Graphic symbols convey information quickly; for example, a floppy diskette is recognized as meaning Personal Computer (PC) disk operations. The symbols that GPL uses are called icons and function blocks. Icons are images that, when selected, perform a function, such as displaying an option menu. Function blocks are rectangles that represent software objects, processes, and GPL operation and special blocks. You paste down these function blocks on the diagram as you need them.

What GPL Provides

The primary items that you create with the GPL Editor are control strategies, which contain software objects and processes. A software object represents and characterizes a field device, such as an Analog Input object, or a data object, such as a Binary Data object. A process is a set of logical evaluations in graphic form that determines when to perform an action, such as when to turn on a supply fan or enable a chiller. Each process is translated into a process object, which is downloaded to and executed by a Network Control Module (NCM).

In addition to the Editor, GPL provides three other utilities that assist in developing processes and point objects. They are the Expert Checker, Simulator, and Translator/Compiler. The Expert Checker verifies that a control strategy is complete and correct. The Simulator tests the functional operation of control strategies. The Translator/Compiler converts the diagrams of the control strategy into processes.

GPL also features online help screens that give you quick explanations of icons and function blocks. And last, GPL provides a library of HVAC applications that are pre-programmed control strategies you can use.

About This Manual

This manual describes how to create, edit, expert check, simulate, and translate control strategies with GPL. It is split into several major chapters: *Introduction, Graphic Programming, Editor, Expert Checker, Simulator, Translator, Function Blocks, Appendix, Glossary*, and *Index*.

Note: This manual does not support GPL as used on the branch CAE system.

Introduction describes what GPL creates, system requirements, required knowledge, installation, and overview of the steps you perform to write a control strategy.

Graphic Programming describes what you need to know to create a control strategy with GPL.

Editor contains step-by-step instructions of all GPL Editor functions. It includes a tutorial for the beginner that teaches how to draw, modify, and print a diagram.

Expert Checker describes how to use this verification tool to check a control strategy before translation.

Simulator describes how to run the GPL Simulator, which can check the integrity of a control strategy. It contains a tutorial that is helpful to the beginner.

Translator describes how to translate and compile a control strategy into processes for the NCM.

Function Blocks is a detailed reference of all function blocks available in GPL. Many application examples are given. The chapter is split into three sections: object blocks, operation and special blocks, and template field descriptions.

Appendix contains helpful reference information, such as a summary of all GPL file names and extensions.

Glossary defines terms specific to GPL that are used in this manual.

Index is a list of GPL topics, concepts, and terms with a chapter and page number reference.

Another Resource

Also refer to the following training package for an interactive learning experience using GPL:

Metasys Graphic Programming Language: A Computer-Based Training (CBT) program for helping you learn the basics of each GPL utility.

What You Create With GPL

You create four items with the Graphic Programming Language: control strategies, object databases, intermediate source code, and processes (Figure 1). You use three GPL utilities--Editor, Expert Checker, Translator/Compiler--to develop these items.

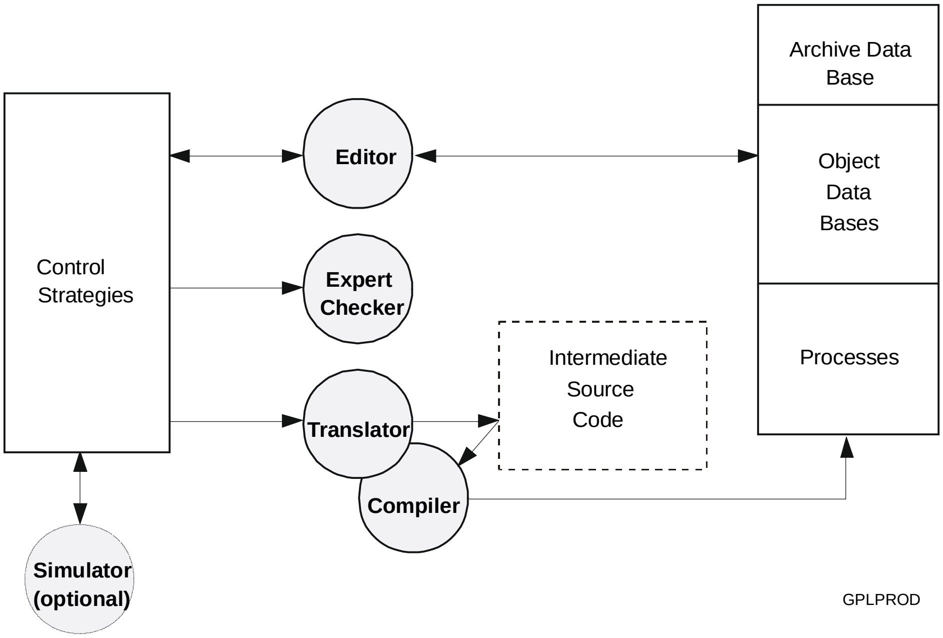


Figure 1: What GPL Produces

The GPL Editor creates and edits control strategies. A control strategy is a set of diagrams that defines the logic that controls the equipment connected to the NCM. For example, a strategy may control a chilled water valve or an entire air handling unit.

The Editor also creates the databases for point objects and function blocks. Each object and function block has an associated database template that defines the characteristics of the object or block. Such characteristics may be the system and object name, high and low alarm limits, and differential.

The GPL Expert Checker verifies the completeness and correctness of a strategy. The Simulator validates the logic of a strategy.

The Translator creates intermediate source code from the process compounds in a control strategy.

The Compiler converts the intermediate source code into downloadable object code. The object code consists of individual process objects that are downloaded to, and executed by, the NCM. The final products of GPL are software objects and downloadable processes that are added to the archive database.

System Requirements

Before you can use GPL, your Operator Workstation or standalone PC requires the following hardware and software.

Notes: GPL supports Windows NT®.

Although certain features may function, starting with Metasys Release 10.0, Windows® 3.1 is no longer supported.

Hardware Requirements

The following hardware is required to use GPL:

- one of the personal computers listed in the *Operator Workstation Configurations Technical Bulletin* in the *Metasys Network Technical Manual (FAN 636).* (If you do not have this manual, contact a local Johnson Controls branch office.)
- one or more floppy diskette drives, 1.44 Mb 3.5 in. only
- math coprocessor (80287 for 286-based computers; 80387 for 386-based computers)
- 3.5 Mb or more available hard disk space
- 640K RAM of conventional memory (525K of which must be available)
- mouse (IBM®, Microsoft® Serial Mouse Systems™ PC Mouse, or equivalent)
- an EGA monochrome monitor or EGA minimum color monitor

Note: Contact the Field Support Center in Milwaukee for additional information.

To print GPL control strategies, you need the following additional hardware (optional):

- IBM Proprinter III™

This hardware is optional but highly recommended:

- 1 Mb of extended memory

Software Requirements

This software is required:

- GPL program diskette (Standard Version)
- IBM DOS 5.0 (IBM machines), or Compaq® MS DOS 5.0 (Compaq machines), or MS DOS 5.0 (CompuAdd® machines)
- mouse driver for IBM, Microsoft, or Mouse Systems mouse

  Note: If you are using an IBM 57SX PC and the IBM mouse driver (PS2MOUSE.COM), you must use Version 1.1 or later of the mouse driver. (Earlier versions will cause system failure.) The mouse driver provided with IBM DOS 5.0 does work properly.

This software is optional but highly recommended:

- Multisoft® Super PC-Kwik® Disk Accelerator 3.55 (for IBM PS/2® and Compaq Deskpro® machines) or Microsoft SMARTDRIVE.SYS 3.03 (for PCs with Person-Machine Interface installed or Compaq SLT/286)
- virtual disk driver (for IBM and Compaq Deskpro models)

Knowledge Up-Front

There are three areas of knowledge you must have to use GPL effectively: HVAC controls, PC operations, and Metasys Network.

Controls Knowledge

To program with GPL, you need to know the basic concepts and methods of HVAC control for commercial buildings. Also, you need to become familiar with the required control sequences before writing control strategies with GPL. The more familiar you are with the control needs, the easier graphic programming will be.

You can create control strategies with GPL using your knowledge, the drawings, job specification, and the sequence of operation (Figure 2).

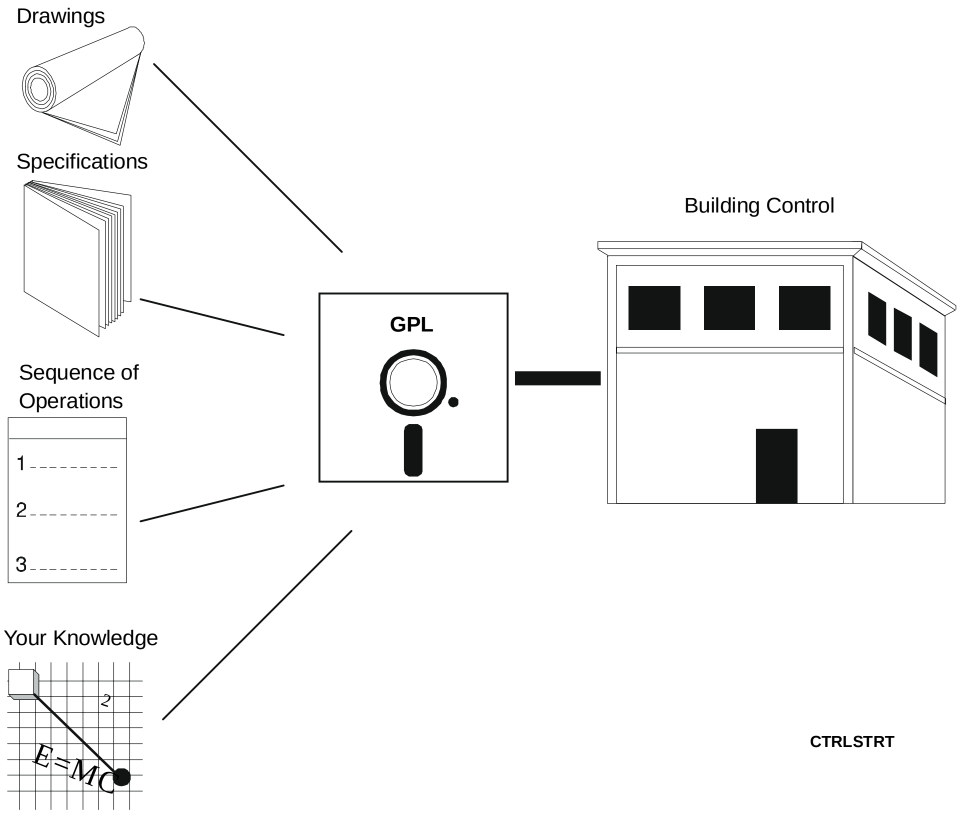


Figure 2: Creating Control Strategies with GPL

PC Knowledge

To use GPL, you need to have basic personal computer skills and know DOS concepts, such as directories and files. You should also know how to use a mouse. A mouse is a small, hand-held pointing device that is used in GPL for moving the cursor, selecting icons, and drawing items on the screen.

Product Knowledge

You also need to be familiar with the functions and features of the Metasys Network, especially software objects and processes. You should have a general knowledge and understanding of Metasys terms and concepts. This knowledge is important to understanding how to program with GPL. For this information, consult the following areas of the *Metasys Network Technical Manual (FAN 636)*:

- *System Architecture*
- *Software Data Sheets*

Getting Started

The first step is to install the GPL software on your PC. The PC should be running properly and displaying the DOS prompt, or it should have Windows 95, Windows 98, or Windows NT®.

IMPORTANT:	Install GPL on the C drive of your PC. GPL will **not** work if installed on the D drive.

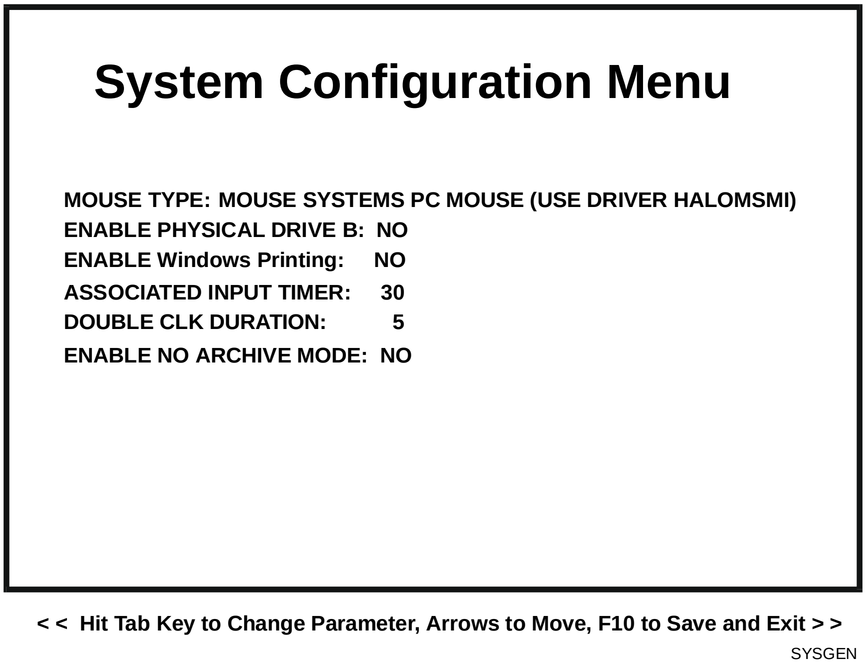
Installing GPL

To install GPL, read the INSTALL file on the GPL program diskette. It contains updated information on how to install GPL on your computer. Also, read the README file, which contains updated information not available in this manual.

Running the System Generation Program

The system generation program (SYSGEN.EXE) configures the GPL software for your particular needs. It specifies six items: what type of mouse you have installed, the timer value for the Simulator, whether your computer has Drive B, the double-click duration for the mouse, and whether No Archive mode is enabled. You must run this program when you first install GPL, and again each time you need to change the configuration.

1. To run the system generation program, go to the directory where the GPL executable files reside. Type SYSGEN at the DOS prompt and press Enter. Six parameters are shown (Figure 3): Mouse Type, Enable Physical Drive B, Enable Windows Printing, Associated Input Timer, Double Clk Duration, and Enable No Archive Mode. The Mouse Type field is back-highlighted in blue to indicate it is selected.
2. Press Tab to change the Mouse Type field until the mouse that your computer uses is shown. Two mouse choices display: Mouse Systems PC Mouse (default) and Microsoft/IBM. The Microsoft/IBM choice applies to either the Microsoft Serial or the IBM mouse (or compatibles). Press Enter to move to the next parameter.



System Configuration Menu

MOUSE TYPE: MOUSE SYSTEMS PC MOUSE (USE DRIVER HALOMSMI)
ENABLE PHYSICAL DRIVE B: NO
ENABLE Windows Printing: NO
ASSOCIATED INPUT TIMER: 30
DOUBLE CLK DURATION: 5
ENABLE NO ARCHIVE MODE: NO

< < Hit Tab Key to Change Parameter, Arrows to Move, F10 to Save and Exit > >

SYSGEN

Figure 3: SYSGEN Screen

3. For the next parameter, Enable Physical Drive B, press Tab to change the entry to Yes if your computer has Drive B. If Drive B is not used, answer No (default). Press Enter to move to the parameter.
4. For the next parameter, Enable Windows Printing, press tab to change the entry to Yes if you wish to print to any Windows compatible printer installed on your PC including non dot matrix printers. If you wish to use the original method of printing, which only supports specific dot matrix printers, select No.
5. For the next parameter, Associated Input Timer, type a timer value that the Simulator will use when simulating associated input applications with the AD and BD objects. The range of the timer is 1 to 255 seconds (default is 30 seconds).

6. For the next parameter, Double Clk Duration, type in the number of clock ticks between the first and second mouse clicks when the mouse is double-clicked. This is a tuning mechanism that is needed to make double-clicking more reliable. The value you select depends on the PC's clock speed. For slower PCs (8 MHz or less), increase duration; for faster PCs (10 MHz or more), decrease duration. The range is 1 to 30 ticks (5 is the default).
7. For the last parameter, Enable No Archive Mode, press Tab to select Yes or No. If Enabled, archive database interactions will not occur, and the Translator will not operate. If Disabled, archive database interactions will occur, and the Translator will operate. Enabling the No Archive mode offers some advantages. For details, refer to the *Graphic Programming* section.
8. Press F10 to save the changes and exit SYSGEN.

Once you have properly installed GPL and configured the SYSGEN program, GPL is ready to run.

Setting Up the Printer

To set up the printer, make sure the printer cable is connected securely to the computer and the printer is online. If a cartridge that provides extended characters is installed, remove it. GPL does not support special cartridges.

To set up the printer to print international language characters, refer to the manufacturer's literature.

Starting GPL with Windows 95, Windows 98, or Windows NT

When running Metasys Operator Workstation (OWS) software with Windows 95/Windows 98, or Windows NT, you can run GPL from the Exit menu on the Network Map as you could in previous releases. However, if you do not have Metasys OWS software loaded on your PC, you can run GPL one of two ways.

- run GPL from the DOS prompt, or
- run GPL from the Windows 95/Windows 98 Start menu.

Running GPL from the Exit Menu

You can run GPL from Metasys OWS software the same way you did in previous releases of Metasys.

To run GPL:

1. Go to the Network Map.
2. Click the Exit menu. The Exit menu appears.
3. Click GPL. The GPL program executes.

Running GPL from the DOS Prompt

IMPORTANT: **Do not** go to DOS by restarting your computer in the MS-DOS mode. GPL does not run properly using this method.

To run GPL from the DOS prompt using Windows 95/98:

1. Click the Start Menu button in the lower, left corner of your monitor. The Start menu appears.
2. Move the mouse over the Programs option. The Programs submenu appears displaying the programs you have loaded on your machine.

Note: In some cases, depending on where you have loaded your DOS prompt icon, you may need to select a submenu.

3. Click the DOS prompt icon. The MS-DOS window appears, allowing you to run a DOS session.
4. Type GPL at the DOS prompt. The GPL program executes.

Some Terms

The following terms are used throughout the manual. You need to understand them before you go on:

Click left: pressing the left mouse button once.

Click right/middle: pressing the right or middle mouse button once. Click right refers to a two button mouse, and click middle refers to a three button mouse.

Double-click left: quickly pressing the left mouse button twice.

Double-click right/middle: quickly pressing the right or middle mouse button twice. Double-click right refers to a two button mouse, and double-click middle refers to a three button mouse.

Select: positioning the cursor on an icon, option, or field and clicking a mouse button.

Drag: holding down a mouse button, moving the mouse, and releasing the button.

Overview of the Steps

To prepare processes for an NCM, you need to follow a series of steps. Figure 5 summarizes these steps in flowchart form.

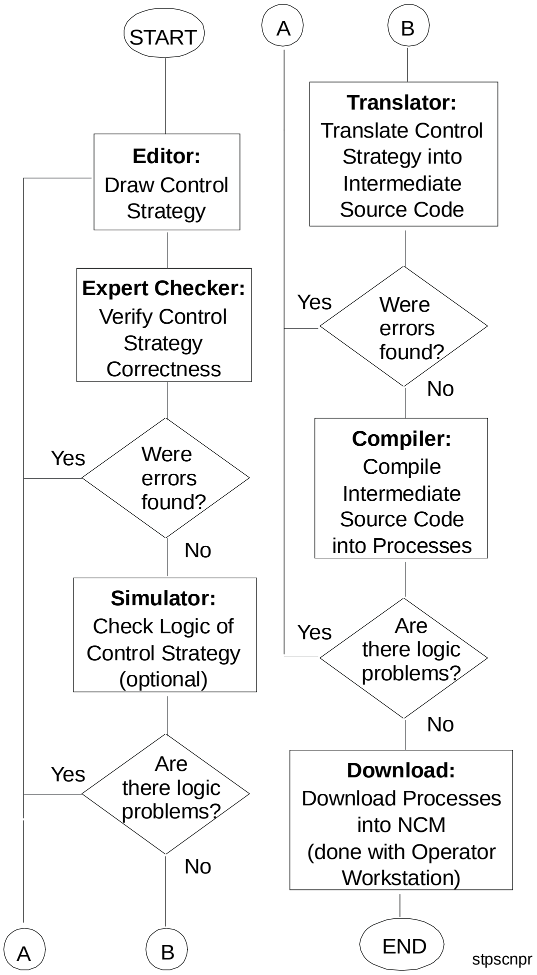

START
Editor:
Draw Control Strategy
Expert Checker:
Verify Control Strategy Correctness
Yes
Were errors found?
No
Simulator:
Check Logic of Control Strategy (optional)
Yes
Are there logic problems?
No
A
B
A
B
Translator:
Translate Control Strategy into Intermediate Source Code
Yes
Were errors found?
No
Compiler:
Compile Intermediate Source Code into Processes
Yes
Are there logic problems?
No
Download:
Download Processes into NCM (done with Operator Workstation)
END
stpscnpr

Figure 5: Steps of Creating a Process

Each of these steps is fully described in various chapters of the manual.

—METASYS®

GPL Programmer’s Manual

Graphic Programming

* Indicates those sections where changes have occurred since the last printing.

Code No. LIT-631020

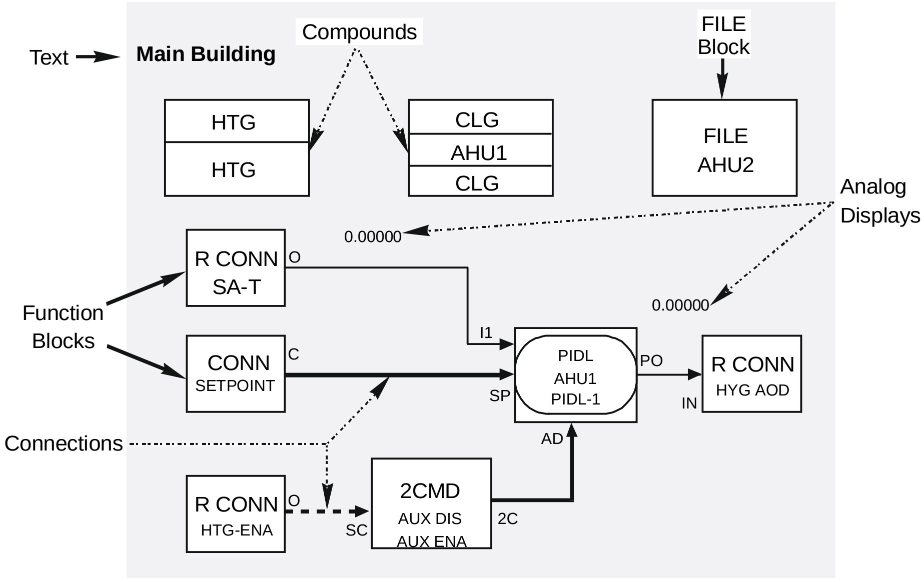
* Indicates those sections where changes have occurred since the last printing.

Overview of the Control Strategy

A control strategy is a series of GPL diagrams that achieves the control objectives for the Metasys Network. Two examples of control strategies are air handling unit and chiller plant control. This section explains the basics of a control strategy.

Components

Figure 1 shows an example of a diagram that is part of a control strategy. A diagram is a drawing in the work area that you create and modify with the GPL Editor.



GPLCOMP

Figure 1: Components of a Diagram

A diagram may contain any mixture of the following components (Figure 1):

- Function blocks--rectangles on the screen that represent an object or GPL operation or special block.
- Compound blocks--rectangles on the screen that look similar to function blocks but have one or two lines through them. They are user-created or pre-programmed blocks that contain one or more diagrams. Some compound blocks represent processes that the NCM can execute.
- Connections--lines that are drawn between blocks to indicate data and control flow.
- Text--comments that label and explain a diagram.
- Analog displays--numeric fields that indicate the outputs of blocks during simulation.
- FILE blocks--special blocks that provide access to other control strategies.

Structure

Figure 2 illustrates the structure of a GPL control strategy. The example features three air handlers. The strategy for each air handling unit is represented by a separate FILE block, all of which are pasted on the highest level diagram. Under each FILE block are compounds and diagrams that contain the logic for the particular air handling unit. The figure shows three compound levels under AHU2. Typically, a control strategy would be composed of many compound levels.

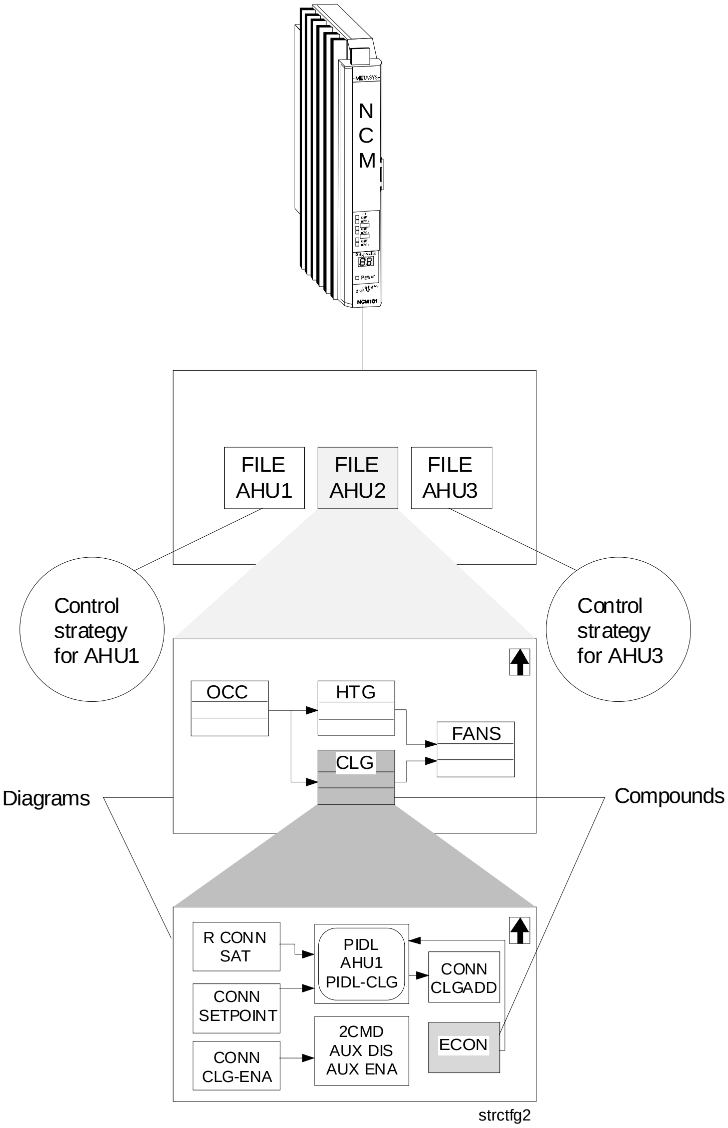


Figure 2: Structure of a Control Strategy

Storage

Each GPL control strategy is stored as a set of three files. Each file holds specific data that contributes to the entire strategy. The files are identified by names in DOS format. The file names consist of eight character maximum names and three character maximum extensions. The descriptions and extensions of control strategy files are:

Description	File Extension	Backup File Extension
Connection Information	.CI	.OCI
Database	.DB	.ODB
Text (includes analog displays)	.TX	.OTX

The .CI file contains data that describes the connections between function blocks. The .DB file contains the information from the database templates. The .TX file contains the text and analog displays that are written in the work area. When you load a control strategy, the GPL Editor merges the three files to create one strategy in memory.

The Editor creates corresponding backup files each time you update an existing control strategy. The backup files are updated each time you save a control strategy. You can use the backup files if you accidentally erase, lose, or revise the originals. The letter “O” for “Old” distinguishes these files.

Just as with any DOS file, strategy files can be copied, deleted, and renamed. Each FILE block represents a control strategy and has a set of files with the same DOS extensions as any other strategy.

When you are saving and loading files, the Editor purposely hides the file name extensions. For example, the files AHU.CI, AHU.DB, and AHU.TX would be collectively identified as AHU in the Editor. The extensions are only important when you are performing file management functions outside of GPL, such as copying or renaming files.

Example

The following is an example of a control strategy that will execute in an NCM (Figure 3). The strategy is for an air handling unit called AHU1 and features three compounds: STATPRS (Static Pressure Control), TEMPCTL (Temperature Control) and FAN-LOGC (Fan Logic). Under each compound are diagrams. For simplicity, only the diagram that is under the STATPRS compound is shown. The sequence of operation in Figure 3 explains the purpose of the strategy.

Sequence of Operation

Static Pressure Control

The DDC controller will control the static pressure upon fan startup.

A static pressure transmitter (located 2/3 the distance of the longest run) will control, through the DDC controller, the supply fan variable frequency drive to maintain 1 in. W.G. duct static pressure. A high limit (hardware) of 4.0 in. W.G. will protect the duct.

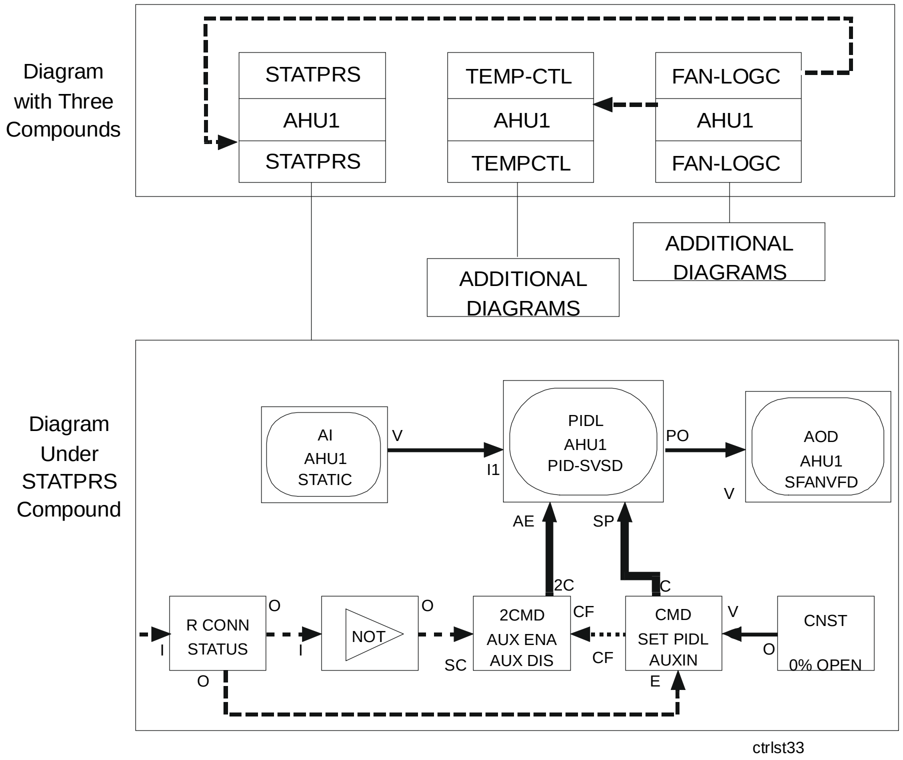


Figure 3: Example of a Control Strategy

Viewing

To view a GPL control strategy (you must have started GPL):

1. Click left on the File icon (disk), which displays a listing of all the defined GPL directories and files.
2. Click left on the name of the strategy you want, then click left on the LOAD option. The control strategy displays in the work area.

When a control strategy is loaded, its highest level is shown. To reach a lower level compound, select the Query icon and double click left inside a compound block. If you happen to single click, the template for the compound displays instead. To clear the template, click left or press the Esc key.

When a lower level compound is displayed, you can easily return to the higher levels. You do so by clicking left on one of the compound names that displays across the bottom of the work area.

Creating

Creating a control strategy involves the following tasks:

- Create diagram.
- Group diagram into compounds.
- Save diagram to a control strategy file.
- Check the control strategy.

The following paragraphs give you an overview of how these operations are performed. For step-by-step instructions, refer to the *Editor* chapter.

Note: The GPL HVAC Library, a library of standard applications, is available for purchase through the branch office. These applications provide a "top-down" approach to GPL programming. The library contains most HVAC control strategies that you will need. You may modify them to meet your needs.

Create Diagram

This task involves pasting down function blocks, defining block databases and connecting the blocks. (The instructions presume that the No Archive mode of the GPL Editor is disabled.)

Pasting Down Function Blocks

The first task in creating a diagram is to paste down the function blocks that the logic requires. Follow these steps:

1. Select the function block from the bottom of the screen by clicking left on it. If the block you want is not currently displayed, you need to change the function block category.
2. To change the function block category, display the list of all categories by clicking left on the LIBRARY field. Position the cursor inside the red check box next to the desired category and click left. The function blocks that line the bottom of the screen change to those in the selected category.
3. Choose a block by clicking left on it.
4. Move the cursor into the work area. A block displays.
5. Position the block in any spot on the work area and click left. The block pastes down.

Defining the Databases

After each function block is pasted down, you need to define its database. The database consists of configurable attributes and parameters that are the unique characteristics of the block. For example, each AI object block has a High Alarm Limit, a configurable attribute you need to define. The values entered affect the way the object is simulated and executed.

To define a database of a function block:

1. Click left on the Query (question mark) icon.
2. Click left inside the function block. Its database template displays.
3. Enter appropriate names and values.
4. Press the F10 key to save the entries. If this is an object block, a message displays on the screen to indicate archive database interaction. Other user messages may then display on the bottom of the work area. See the *Editor* chapter for details.

Connecting Function Blocks

When two function blocks are pasted down in the work area, you can connect them together. Follow these steps:

1. Click left on the Connection (arrow) icon.
2. Click left on the function block at which the connection is to start. An output connection menu displays.
3. Click left on the desired output connection name.
4. Move the connection line into the function block that is to receive the connection. Click left. An input connection menu displays.
5. Click left on the desired input connection name. The blocks are now connected.

Group Diagram into Compounds

The next task is to group the diagram (or part of the diagram) into a compound. You use compounds to logically group blocks, specify process objects, and aid in presenting control strategies. Follow these steps:

1. Click left on the Compound icon.
2. Click left on the MAKE option. Move the mouse and notice a white enclosing box has appeared.
3. Enclose the portion of the diagram that you wish to compound into the white box. If the enclosing box is not the correct size, hold down the right/middle button and drag the mouse to resize it. Click left.
4. Click left on the TEMPLT option to define the name and type of compound in its database template. Press F10 to save the entries. See the *Editor* chapter for details.
5. Enter a name for the compound in the Name field by simply typing from the keyboard. (You do not have to select this field with the mouse.)
6. Click left on SAVE.
7. Click left on DISK to save the new compound to disk.

Save the Control Strategy

The next task is to save the control strategy to a file.

1. Click left on the File icon.
2. Type in a name in the File Name field.
3. Click left on the SAVE option. The file is saved to disk.

Check the Control Strategy

After a control strategy is complete, you need to verify it with the Expert Checker. This is a separate utility from the GPL Editor that verifies the completeness and correctness of the strategy. For example, it checks that all required connections are made and all blocks are correctly nested.

A description of all errors that the Expert Checker finds are placed in a list file. You can view this list file with the Editor by clicking left on the VIEW option under the Query option menu.

To run the Expert Checker, follow these steps:

1. Double-click left on the Tools icon (hammer).
2. Click left on the Expert Checker icon (check mark). The Expert Checker screen then replaces the Editor screen and begins running.
3. After the strategy has been checked, press any key to return to the Editor screen.

At this point, you can check the logic of the control strategy with the Simulator and translate and compile it with the Translator and Compiler.

Function Blocks

This section introduces function blocks. It explains the categories, classes, features, and templates of function blocks. For instructions on how to paste down, edit, and connect function blocks, refer to the *Editor* chapter.

Introduction

A function block is a rectangle on the screen that represents an object or GPL operation or special block. Each function block defines an object or performs a particular action, such as select the highest of two values, or calculate enthalpy.

Each block is pasted down in the work area and defined by filling in its database template. The template lists the attributes and parameters (i.e., characteristics) of the block.

In most cases, each block is connected to other blocks. The connections between blocks establish the data and control flow. Each block has a defined set of allowable inputs and outputs to which lines may be connected. These inputs and outputs are displayed on connection menus. There are nine line types that pass different kinds of data, such as analog, binary, and time. Each line type is described in detail later.

Categories and Classes

All function blocks are organized under a category and a class (Figure 4). A function block category is a grouping of function blocks that have similar uses. The GPL Editor uses block categories to logically organize the 69 function blocks that are available. All categories are listed in the function block library that displays when you click left on the LIBRARY box. There are 14 different categories of blocks.

		CLASS		
		Object	Operation	Special
CATEGORY	**Input/Output**	BI BO AI AOS AOD ACM REF		
	Data	AD BD	SVAR VH	CNST CONN
	Multistate	MSI MSO MSD		
	Controllers	PIDL LCG 210A 260A ZONE		
	Control		PIR BSEQ COMP DFCM DBCM	
	Calculations		RAMP SPAN FILT	
	Psychrometric Equations		ENRH ENDP WBDP WBRH DWPT RH	
	Selectors		HSEL LSEL SWCH SAMP MSEL	
	Logic		AND OR XOR NOT LTCH PULS DLAY	
	Math		ADD SUB MUL DIV AVG EQN	
	Report		PRNT ADV	
	Process Control		PERD WAIT STOP ABRT	
	Object Control		CMD 2CMD READ WRIT	
	Time		TIME RTOT TTOR	
	Reliability		FREL UNRD	
	Miscellaneous		TOT USER	FILE

FCTBL

Figure 4: How Function Blocks are Grouped: by Category and by Class

A class is a grouping of function blocks that have similar definition requirements. For example, some blocks require a system\object name, while others need a block name. There are three classes of blocks: object, operation, and special.

Object Blocks

Object blocks represent software objects in the archive database. Software objects in the Metasys system can be defined in GPL. They are:

ACM	Accumulator	**MSD**	Multistate Data*
AD	Analog Data	**MSI**	Multistate Input*
AI	Analog Input	**MSO**	Multistate Output*
AOD	Analog Output Digital	**PIDL**	PID Loop
AOS	Analog Output Setpoint	**REF**	Generic Object Reference**
BD	Binary Data	**210A**	Control System C210
BI	Binary Input	**260A**	Control System C260
BO	Binary Output	**ZONE**	Fire Zone
LCG	Lighting Control Group		

* Used only in European market.

** Represents C260X, C500X, and CS software objects; also, any hardware object.

Each software object block has at least one input and output. At its input, an object block can accept commands and changes to its writable attributes. For example, a binary output object can be commanded to Start or Stop, or the AUX_IN port of a PIDL object can be commanded to a value. Also, a BO object can accept a change to Minimum On Time, which is a writable attribute.

At the output, the value of an object block is made available for other blocks to read. An example would be a selector block reading the value of an analog input object. The line connection between the selector block and the AI object block is actually a “request” to read an attribute.

Operation Blocks

These blocks instruct the NCM to perform an operation, such as selecting between two values, executing some logic or calculation, or issuing an advisory. They also provide process control functions, such as temporarily suspending the execution of a process. The GPL Translator creates executable code from these blocks. The operation blocks are the largest group of blocks in GPL and include the following:

ABRT	Abort	**PERD**	Period
ADD	Add	**PIR**	Proportional-Integral Reset
AND	And	**PRNT**	Print
AVG	Average	**PULS**	Pulse
BSEQ	Binary Sequencer	**RAMP**	Ramp
CMD	Command	**READ**	Read
COMP	Compare	**RH**	Relative Humidity
DBCM	Deadband Compare	**RTOT**	Real-to-Time Conversion
DFCM	Differential Compare	**SAMP**	Sample and Hold
DIV	Division	**SPAN**	Span
DLAY	Delay	**STOP**	Stop
2CMD	Dual Command	**SUB**	Subtraction
DWPT	Dew Point	**SVAR**	Shared Variable
ENDP	Enthalpy-Dew Point	**SWCH**	Switch
ENRH	Enthalpy-Relative Humidity	**TIME**	Time of Day
EQN	Equation	**TOT**	Totalization
FILT	Filter	**TTOR**	Time-to-Real Conversion
FREL	Force Reliable	**UNRD**	Unreliable Data
HSEL	High Select	**USER**	User
LSEL	Low Select	**VH**	Value Holder
LTCH	Latch	**WAIT**	Wait
MODE	Mode	**WBDP**	Wet Bulb-Dew Point
MUL	Multiply	**WBRH**	Wet Bulb-Relative Humidity
NOT	Not	**WRIT**	Write
OR	Or	**XOR**	Exclusive Or

All operation blocks can provide a control flow input and output. Some blocks have a conditional input called Enable. This input allows for conditional execution (explained in *Order of Process Execution* section*)*. All operation blocks must be placed inside a process compound in order for them to translate into executable code. This is a GPL requirement. This requirement differentiates operation blocks from the other two classes of function blocks--object and special. Object and special blocks do not have to be placed in a process.

Special Blocks

The blocks that belong to this class are the Constant (CNST), Connection (CONN), and File (FILE) blocks. The CNST block specifies a constant value that can be used as an input to any block. The CONN block titles a connection line that comes from a different diagram. The FILE block serves as a link between multiple strategy files for one NCM. With the FILE block, you can paste several files on one screen, then translate all the files together, instead of individually. None of these blocks have to be placed in a process, since they do not translate into executable code.

Features

Some features are similar to all function blocks, while others are unique.

Common Features

The following are the features that all function blocks have in common.

- Each block is rectangle-shaped and displays on the screen at a specific default size. You can adjust this size before or after pasting down the block. As an added convenience, three function keys are available that vary the size of blocks: F2, F3, and F4. These keys provide large, small, and standard default sizes respectively.

- Each block is labeled with an abbreviation of its type name, which appears centered in the upper portion of the block (Figure 5). In addition, you can assign a unique name to each block. For object blocks, this name is a system\object name. For operation and special blocks, it is a block name. For command and dual command blocks, the block name is actually the command name. For all blocks, names are centered in the block. The size of the characters depends on how long the name is; the longer the name, the smaller the characters will be.

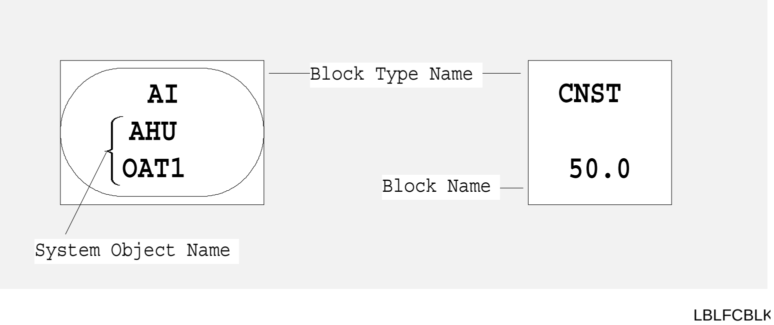


Figure 5: Labeling of Function Blocks

- Each function block has a database template. (See Figure 9 for an example.) The template contains attributes and parameters that describe the characteristics of the block (e.g., number of inputs). For more details, see the *Templates* section.
- A function block can be copied. You may find that copying a defined block is easier than pasting down a new one of the same type, because you do not need to redefine the common parameter values.
- All function blocks have online help screens that describe the block and discuss what makes the output of a block unreliable. Position the cursor on a block in the function block directory and press F1 to get help.

Unique Features

The following are unique block features.

- The most evident differences between blocks are their special markings (Figure 6).

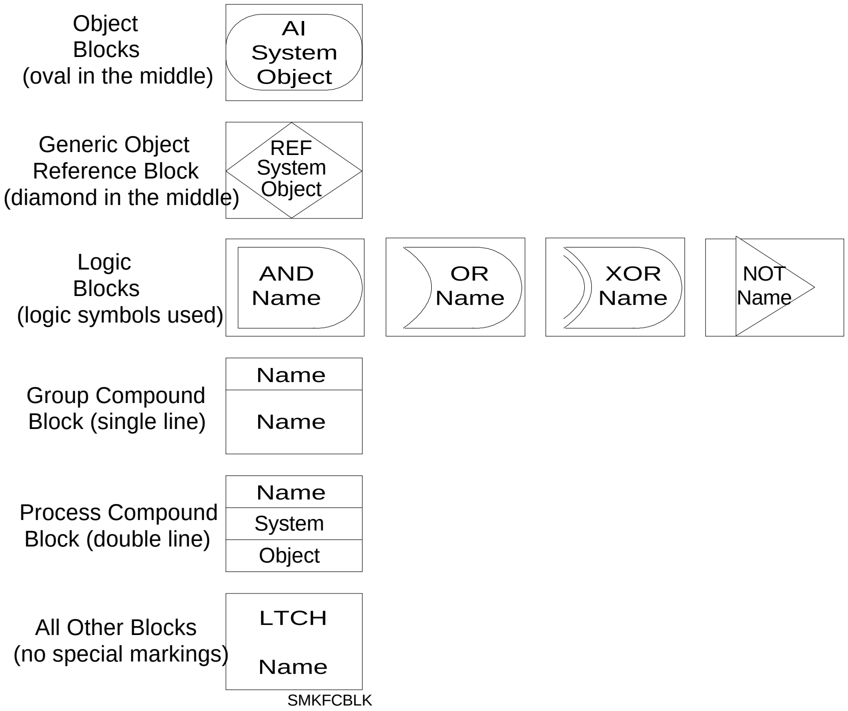


Figure 6: Special Markings on Function Blocks

- An object block pastes down undefined, while an operation or special block pastes down defined. An undefined block is colored magenta and bordered with dots (Figure 7). A defined block is colored brown with solid lines.

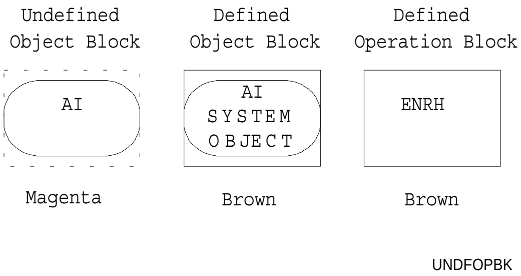
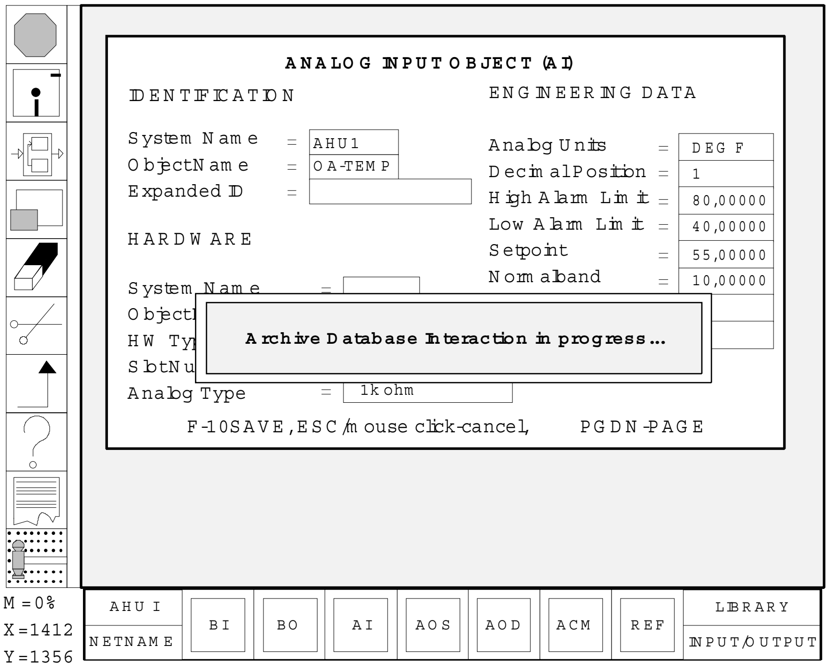


Figure 7: Appearances of an Undefined and Defined Object Block, and Defined Operation Block

- The archive database is invoked each time you add, modify, or delete an object block. It is also invoked when you query an object block for the first time since the file has been loaded. The message `Archive Database Interaction in progress` displays in the middle of the screen (Figure 8). During the interaction, the Editor may display a user message in the bottom of the work area. For example, if you define an invalid hardware object name in the block's template, the message `Archive hardware object record does not exist—{system\object name}` will display. For more details, see the *Archive Database Interface* section at the end of this chapter.



ANALOG INPUT OBJECT (AI)

IDENTIFICATION

System Name = AHU1
Object Name = OA-TEMP
Expanded ID =

HARDWARE

System Name =
Object
HW Typ
SlotNu
Analog Type = 1k ohm

ENGINEERING DATA

Analog Units = DEG F
Decimal Position = 1
High Alarm Limit = 80,00000
Low Alarm Limit = 40,00000
Setpoint = 55,00000
Normalband = 10,00000

Archive Database Interaction in progress...

F-10SAVE, ESC/mouse click-cancel, PGDN-PAGE

ARCHDB

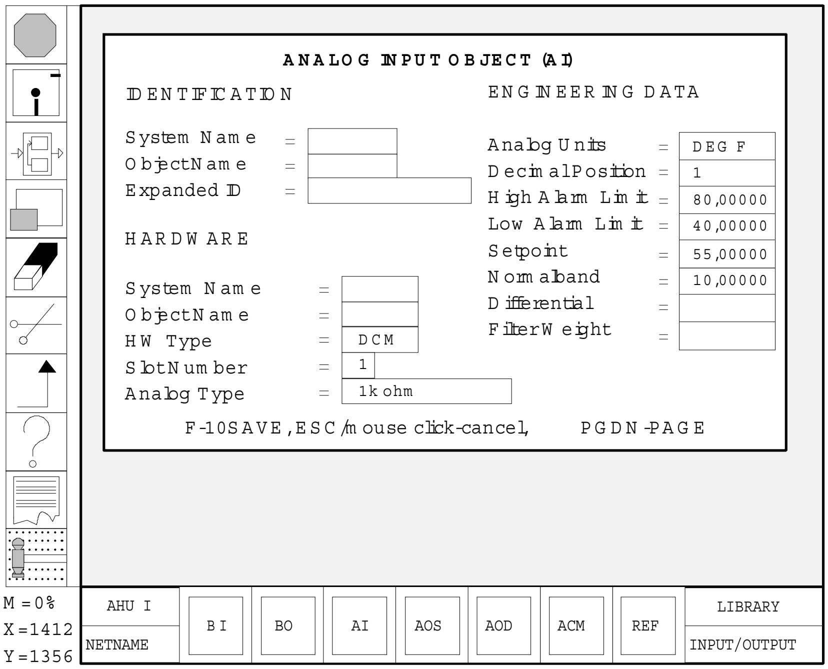
Figure 8: Archive Database Interaction Message

- Copied operation and special blocks are pasted down defined, while copied object blocks are pasted down as undefined. This is because only one instance of an object block can be defined in a strategy file. Therefore, you need to define a unique system\object name for each copied object block.
- When you copy a compound, its object blocks change to undefined. This is a precaution to protect from having multiple objects with the same software system\object name in the same file. You then need to redefine each object block individually. You can do so by simply opening each object block template, assigning a unique system\object name, and pressing F10. When you do this, the Editor reads the archive database, checking if the object is already defined.

Templates

You define the database of each function block by filling in its template. Every block has a database template. Figure 9 shows the first page of the AI object template. A template drops down over the diagram in the work area when you query the block.

The template consists of parameter and attribute fields into which you enter information. These fields describe the details of the function block.



ANALOG INPUT OBJECT (AI)

IDENTIFICATION

System Name =
Object Name =
Expanded ID =

HARDWARE

System Name =
Object Name =
HW Type = DCM
Slot Number = 1
Analog Type = 1k ohm

ENGINEERING DATA

Analog Units = DEG F
Decimal Position = 1
High Alarm Limit = 80,00000
Low Alarm Limit = 40,00000
Setpoint = 55,00000
Normalband = 10,00000
Differential =
Filter Weight =

F-10 SAVE, ESC/mouse click-cancel, PGDN-PAGE

AITEMP

Figure 9: First Page of AI Template

Notice in Figure 9 that the fields on the template are organized under groups. The groups on this template are IDENTIFICATION, HARDWARE, and ENGINEERING DATA. The types and number of groups vary, depending on the function block. But the presentation from block to block is made as consistent as possible. For example, all object blocks have IDENTIFICATION as their first group, located in the upper left corner of the template.

Most parameters have default values. These are acceptable values for the block. In many cases, you will not have to change these values.

The amount of information that the template contains depends on the function block. Some blocks, such as those under the Math category, require very few entries, while others, namely the object blocks, have two or three pages (screens) of entries.

Each time you display a template, the Editor positions the cursor within the upper left parameter field. The field is highlighted in white to indicate it is selected for entry. A red rectangle within a field indicates the cursor's position.

You can use the following keys within the templates:

Key	**Function**
Page Up	Displays the previous page.
Page Down	Displays the next page.
F10	Saves template changes and closes template.
Esc	Ignores template changes and closes template.
Arrows	Moves cursor between fields.
CTRL + Left and Right Arrows	Moves cursor within a field one space at a time.
Backspace	Moves cursor back one space within field, deleting the character.
Insert	Shifts characters one space to the right for you to add a character.
Delete	Deletes a character in a parameter field.

Colors specify the different characteristics of template fields. The colors and the corresponding characteristics are:

Color	Description
Blue	Non-modifiable fields; once you have defined the block, these fields cannot be changed without deleting the object from the archive database, and re-adding it.
Light Blue	Read-only fields.
Magenta	Modifiable fields; use the keyboard to enter appropriate values.
Light Magenta	Non-modifiable fields; can be changed by factory only.
Green	Modifiable Tab fields; press the Tab key to toggle through all available choices.
Light Green	Modifiable Tab fields that become non-modifiable once the object is added to the archive database.
Yellow	Static text in the template (e.g., headers and footers).
White	Currently selected field.

Note: On the portable Operator Workstation, you’ll see variations of gray instead of colors.

Templates have a number of different types of fields. They are: analog, integer, binary, character string, tab, pop-up, blank/delete, and STD range type.

Analog

These fields take analog, real, or floating point numbers. The range for real numbers is 9999999 to 0.00001, 0.0, and -0.000001 to -99999999. Real number fields have eight places including the minus sign and the decimal place.

Note: The Editor converts all decimal numbers into binary format; therefore, only five of the eight possible digits can be displayed reliably. For example, if -49.4567 is entered in the database template, GPL will round it off to -49.4566, not to -49.457. This is because rounding errors are introduced after the fifth digit. Decimal number conversion is unpredictable after the fifth digit.

Also, the Editor forces a fraction to have a 0 before the decimal point. The least significant digit will not be displayed. For example, if -.123456 is entered, -0.12345 will be displayed (the number 6 is dropped).

Integer

Integer fields take integer values that are valid for the type of data requested. An example of an integer field is Number of Inputs for a HSEL block, which can accept the numbers 2, 3, or 4.

Binary

Binary fields take a Yes (Y) or No (N) entry.

Character String

These fields may allow any combination of alphabetic characters (A-Z), international language characters (see *Appendix F: Characters, Symbols, and Reserved Words)*, numbers (0-9), and certain symbols (e.g., “+” sign). For a list of all valid characters and symbols, refer to *Appendix F: Characters, Symbols, and Reserved Words.*

Tab

Tab fields are dynamic fields that have two or more entries; you display these entries by pressing the Tab key. Each time you press Tab, a different selection appears in the field. Once you have scrolled through all the choices, the first choice reappears. To keep a selection, press Enter or an Arrow key.

Pop-up

Pop-up fields are conditional fields. When they display and what they display depends on the entry made on a previous field. For example, on the template of a BI object block, the parameter “Subslot Number” will “pop up” on the screen when you select DCM for the hardware type.

Blank/Delete

These types are real number fields into which you can enter a blank by pressing the space bar. In some instances, these fields can also accept the letter “D” for Delete.

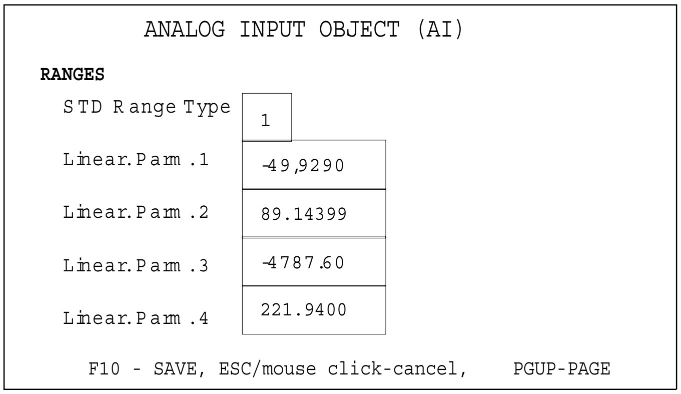
If you do not wish to define a value for an optional attribute, enter a blank. For example, you may not want to specify a high alarm limit for an analog data object. A blank in the high alarm limit template field would leave this attribute unspecified. You may specify a high alarm limit later by editing the template.

If you enter a blank in a Command block template, the command will not send a new value for this parameter. You may also enter the letter “D,” which will delete a parameter when the command is sent.

For example, an Alarms command to an analog data object has three parameters: High Limit, Low Limit, and Differential. If you enter a blank in the Differential field, the existing differential value of the object will remain unchanged. However, if you instead enter the letter “D” in the Differential field, the command would delete the current differential.

STD Range Type

The standard (STD) range type field appears only on the template for the AI point object (Figure 10). It defines the standard range type of the sensor for the AI object (e.g., voltage and temperature range). Below this field are four fields labeled Linearization Parameters. The GPL Editor fills in these fields for you, according to the range type specified above. For example, entering “3” would change the Linearization Parameters to the values associated with range type 3. This saves you the effort of manually entering the values.



ANALOG INPUT OBJECT (AI)

RANGES

STD Range Type	1
Linear.Parm.1	-49,9290
Linear.Parm.2	89.14399
Linear.Parm.3	-4787.60
Linear.Parm.4	221.9400

F10 - SAVE, ESC/mouse click-cancel, PGUP-PAGE

STDTLPI

Figure 10: STD Range Type and Linearization Parameter Fields

The Editor offers 25 standard ranges that provide the linearization parameters for the Metasys family of function modules and sensors. These ranges are listed in a table; refer to the *AI Object Block* in the *Function Blocks* chapter. If you want to specify your own values, enter a range type of “0,” which allows you to edit these fields.

Template Error Checking

The GPL Editor performs syntactic error checking for each template field. It checks for valid data types, proper ranges, field interactions, and duplicate object names.

Valid Data Types and Proper Ranges

The data entered into a field must be of the proper type and within the proper range. For example, if integers are required, the Editor will not allow you to enter a letter. If you try to enter a letter for a field that requires a number, the Editor will ignore your keystroke. If you try to enter a number outside the appropriate range, the Editor will accept the entry; but when you press the Enter key, a warning beep will sound, and the entry will change back to its previous (valid) value. Also, the cursor will remain in the field.

Field Interactions

Some checks involve the interaction between two fields. For example, the high alarm limit you enter for an AI object must be greater than its low alarm limit. The Editor checks for these interactions when you press F10 to save the template. If an interfield error is found, an appropriate error message displays on the bottom of the work area. After you click the mouse to continue, the Editor returns to the first page of the template. Refer to the *User Messages* section of the *Editor* chapter for definitions of all interfield error messages.

Duplicate Object Names

You may use a system\object software name for an object block only once in a single strategy file. You can use the same system\object name in multiple files, but not twice in the same file. The Editor checks for duplicate system\object names when you press F10. If the Editor finds a duplicate, the appropriate error message is displayed. You can either enter a unique system\object name, or click left to exit the template.

Note: If you need to use the value from an object block more than once in the same file, use a fan-out connection. Refer to the section *Fan-In and Fan-Out Connections* for details.

The data in the template is saved when you press the F10 key. For object blocks, the template information is saved in two places: the GPL database file (.DB extension of control strategy) and the appropriate archive database file. For all other function blocks, the template information is saved to the strategy file only. When you translate a control strategy, the GPL Translator uses the template information to create the executable code that will run in the NCM.

Compounds

This section explains compounds. For instructions on how to make and edit compounds, refer to the *Compound Functions* section in the *Editor* chapter.

Introduction

A compound is a grouping of blocks combined to form a high level function, such as optimal start or damper control. Several compounds are combined to form a control strategy. Each compound is represented by a block. Figure 11 shows a compound block and the diagram it represents.

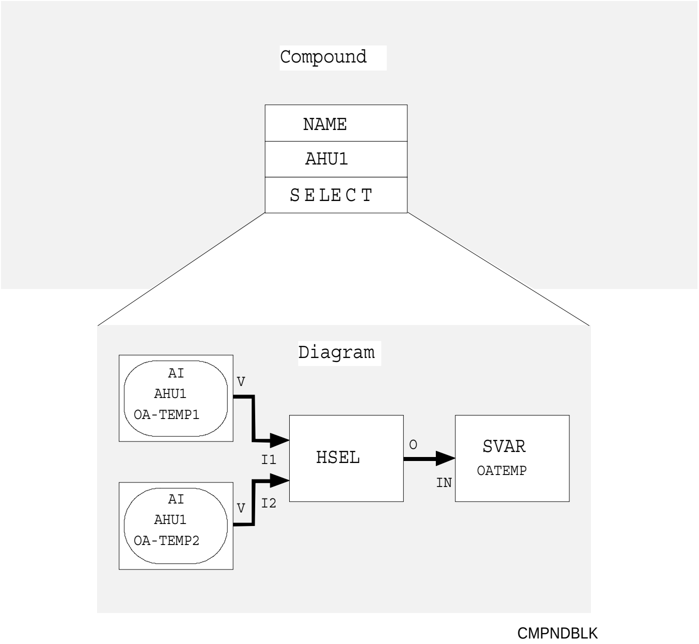
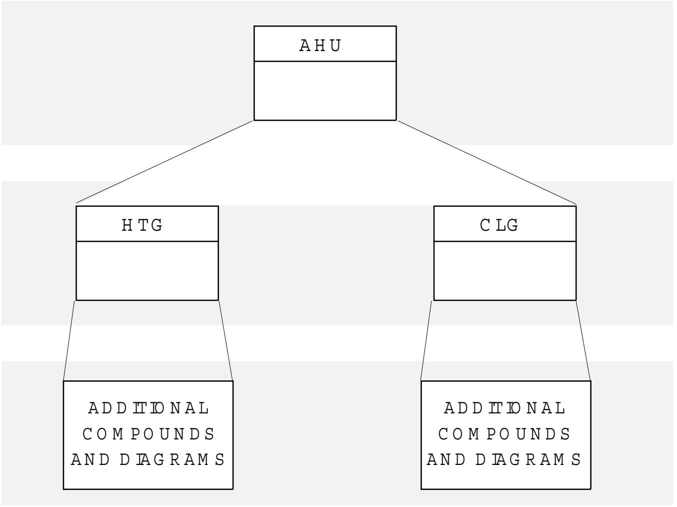


Figure 11: Compound Block and Diagram it Represents

Compounds are used extensively in GPL. A compound has these features:

- Helps you organize many blocks under a diagram by grouping them under functional categories. For example, all heating control logic can be grouped under a heating compound.
- Creates a new function block, such as an 8-input AND, that is made up of various other function blocks.
- Breaks down a complex control strategy into more easily manageable and understandable parts.
- Hides details that would otherwise clutter a diagram.
- Creates multiple levels of diagrams under one strategy file. This is called compound nesting. For example, as in Figure 12, the highest level might be the air handling unit, the next level the heating and cooling control logic, then the different heating and cooling systems, and so forth. A maximum of 30 levels can be nested.
- Creates a standard application that can be stored, reused, and changed as often as needed.
- Provides the mechanism for creating process objects (NCM requirement).


AHU
HTG
CLG
ADDITIONAL COMPOUNDS AND DIAGRAMS
ADDITIONAL COMPOUNDS AND DIAGRAMS

CMPDNST(

Figure 12: Compound Nesting

A compound block shares some of the same features as function blocks: it can be named, defined, edited, and pasted down on a diagram.

When you are displaying one or more nested compound levels, the names of the currently active compound levels are shown on the bottom of the work area, enclosed in rectangular boxes (Figure 13). These boxes are called Compound Name fields.

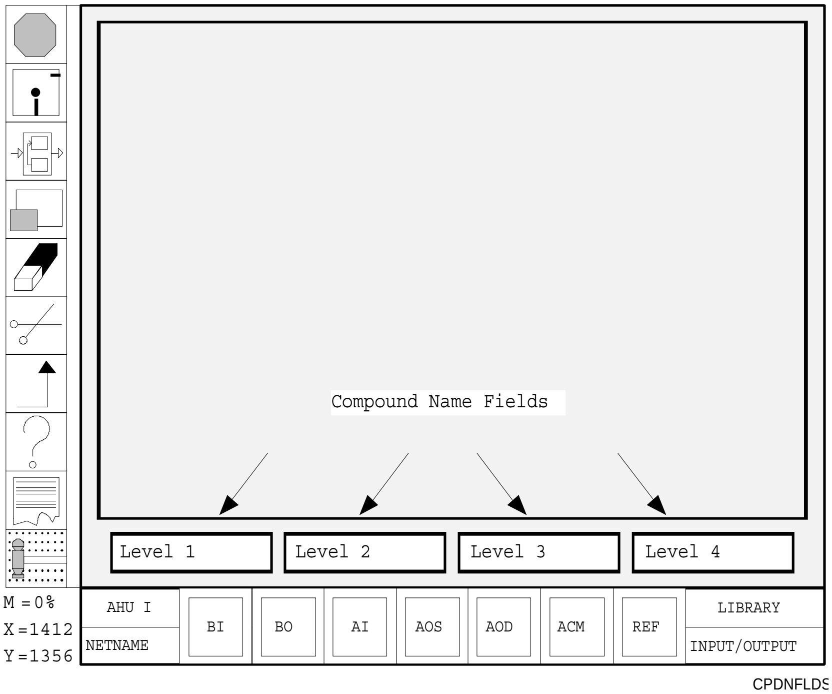


Figure 13: Compound Name Fields

The Compound Name fields have two purposes. The first purpose is to indicate the currently active compound level. The work area can show up to the last four levels. Higher levels are hidden from view, but return as you move toward the highest compound level. The second purpose is to provide for a mechanism of returning to a higher level. Clicking left on a Compound Name field changes the work area to show the diagram that contains the associated compound block. Pressing Esc moves the diagram up one level.

Types

GPL offers three types of compounds: group, process, and Restart process. Figure 14 shows an example of each type. You can distinguish the different types by their appearances. The group compound block has two sections: the first section contains the compound file name; the second section contains the compound block name. The process and Restart compound blocks have three sections: the first section contains the compound file name; the second section contains the system name; the third section contains the object name (always Restart for the Restart compound).

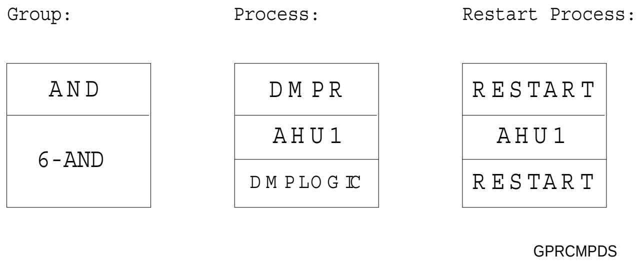


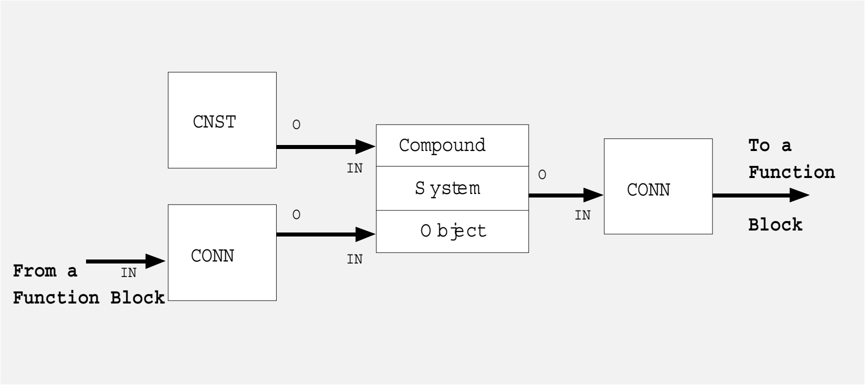
Figure 14: Appearances of Group, Process, and Restart Process Compounds

A group compound is a grouping of blocks, connections, and text. Its purpose is to functionally group a diagram into a single block. For example, a group compound can be used to create a customized block, such as a six input AND block.

The process compound is a special case of the group compound. Besides grouping blocks, connections, and text, it is used to identify part of a diagram as a process object that, when translated, creates downloadable, NCM executable code. Only those diagrams that feature one or more operation blocks must be placed in a process compound, since compounding is the method of creating object code from the operation blocks. In fact, only the operation blocks on a diagram need to be compounded into a process; all others do not have to be compounded. A process compound may contain group compounds and other process compounds. However, the Translator creates an independent process from each process compound.

The Restart compound is a special process compound that is used exclusively for performing required startup processing. It is always the first process to execute when the NCM starts. For example, the Restart compound can be used to initialize shared variables, determine occupancy time, or delay processing until field equipment comes back online. Only one Restart process can exist per NCM.

A compound provided from the factory may be protected. When a compound is protected, you may view and edit its database template, but you cannot view or edit its contents. Also, you cannot connect to or from it, since a protected compound cannot be opened. However, the protected compound may have preconnected blocks, such as the CNST block, which you may connect from, and CONN blocks, which you can connect to or from. These blocks contain preselected input and outputs that you can use. Figure 15 shows two CONN blocks and one CNST block with a protected compound.



CONCNS⁻

Figure 15: CONN and CNST Blocks Used with a Protected Compound

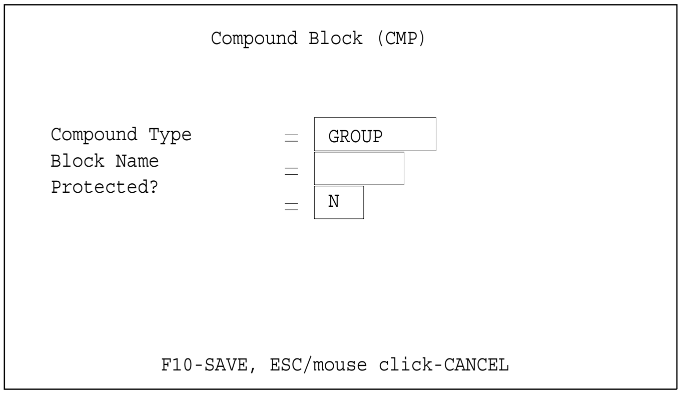
Features

A compound block has the following features:

- The block has a name and a database template. Within this template are parameter values that you enter, such as the type of compound and its name. The next section describes compound templates in detail.
- The compound block or its contents can be pasted down on a diagram. When you paste down a compound block on an existing diagram, all the diagrams that are under that compound are added to the file. You may instead paste down the expanded version of the compound block, which consists of the compound’s first level. You would load in an expanded version if you want to create a new strategy or compound based on an existing one.
- When you make a compound and click on SAVE to save it, you can either save the new compound to the screen or save it to the hard disk. If you elect to save it to the screen, the Editor creates a compound block for the compound and pastes it in the middle of the work area. Then, you can save the new compound to disk. If you elect to save the compound to disk, the Editor will save the compound to a file on the hard disk. Then, you can load the compound block into the work area.
- A connection line can be drawn in a compound block. This has two possible purposes. The first is to command the compound. The three available commands are PRC_ENA (Process Enable), PRC_DIS (Process Disable), and TRIGGER. The second purpose is to connect a line to a block within the compound via the OPEN CMPD connection. You may terminate a connection from one compound level to another, but not up a previous (higher) level. Also, you cannot terminate a connection from a compound level to itself. Lastly, you can connect up to ten levels deep.

Templates

The compound templates contain parameter values that describe the characteristics of the compound, such as its name and its type. The template for the group compound is shown in Figure 16.



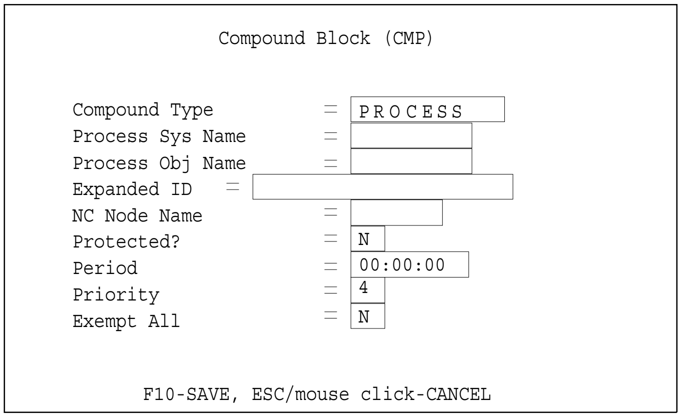
```
                 Compound Block (CMP)


Compound Type          =   GROUP
Block Name             =   
Protected?             =   N



          F10-SAVE, ESC/mouse click-CANCEL
```

Figure 16: Group Compound Template

Template Fields for Group Compound

Name	Type	Default	Range/Choices
Compound Type	TAB	GROUP	GROUP, PROCESS, RESTART PROCESS
Block Name	Character	Blank	8 characters
Protected?	Binary/ Read-only	N	Y (Yes) or N (No)

Figure 17 shows the template for the process compound.



Compound Block (CMP)

Compound Type = PROCESS

Process Sys Name =

Process Obj Name =

Expanded ID =

NC Node Name =

Protected? = N

Period = 00:00:00

Priority = 4

Exempt All = N

F10-SAVE, ESC/mouse click-CANCEL

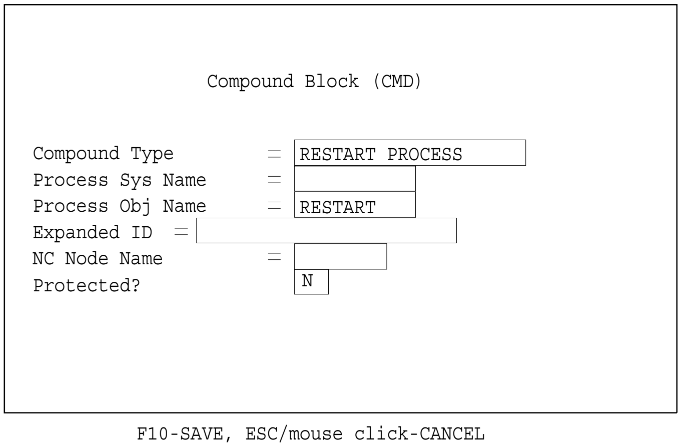
PRCPDTMP

Figure 17: Process Compound Template

Template Fields for Process Compound

Name	Type	Default	Range/Choices
Compound Type	TAB	PROCESS	GROUP, PROCESS, RESTART PROCESS
Process Sys Name	Character	Blank	8 characters
Process Obj Name	Character	Blank	8 characters (See Note.)
Expanded ID	Character	Blank	24 characters
NC Node Name	Character/ Read-only	Blank	8 characters
Protected?	Binary/ Read-only	N	Y (Yes) or N (No)
Period	Time	00:00:00	Time
Priority	Integer	4	1 to 4 (1=highest)
Exempt All?	Binary	N	Y (Yes) or N (No)
Note: For a process compound, entry cannot be RESTART.			

Figure 18 shows the template for the Restart compound.



```
                Compound Block (CMD)

Compound Type          = RESTART PROCESS
Process Sys Name       = 
Process Obj Name       = RESTART
Expanded ID  = 
NC Node Name           = 
Protected?               N

          F10-SAVE, ESC/mouse click-CANCEL
```

RSCPDTMP

Figure 18: Restart Compound Template

Template Fields for Restart Compound

Name	Type	Default	Range/Choices
Compound Type	TAB	RESTART	GROUP, PROCESS, RESTART PROCESS
Process Sys Name	Character	Blank	8 characters
Process Obj Name	Character	RESTART	RESTART
Expanded ID	Character	Blank	24 characters
NC Node Name	Character/ Read-only	Blank	8 characters
Protected?	Binary/ Read-only	N	(Yes) or N (No)

Compounds in the Applications Library

Johnson Controls engineers have developed a software package called the Metasys GPL HVAC Library (WS-SWHLIB-00x) that contains a collection of pre-programmed standard applications. These are tested applications that you can copy and modify to meet your needs. You may load them from the compound option menu. Example files are also available.

Compound Design

Which diagrams you should make into compounds depend on these factors:

- The diagrams that need to be grouped into process compounds. All operation blocks on a diagram must be part of a process compound. For details, refer to the *Processes* section.
- Your programming preferences. For example, you can divide a complex strategy into several compounds, because several small compounds are easier to work with than a few large ones.
- Number of blocks on a diagram. You may want to make it a standard practice to compound any diagram that contains ten or more blocks, to assure the diagram is easy to read.
- The sequence of logic on a diagram. A compound should group a complete sequence of logic. For example, you should compound fan logic separately from economizer logic.

You can create a structure of compounds using either a bottom-up or a top-down design. The bottom-up design involves drawing the lowest level diagram, then the next highest diagram and so forth, making group and process compounds along the way. This method is most useful when you cannot use one of the compounds in the GPL HVAC Library.

The top-down design is the preferred method, since you don’t need to create compounds from scratch. It involves loading in a compound or example file provided in the GPL HVAC Library and editing it to fit your particular application. Another manner of top-down design is to create empty compounds first--the high-level structure--then drawing the diagrams that belong under the compounds. To create an empty compound, simply compound an empty screen.

Processes

This section explains processes. For detailed instructions on how to create and edit a process, see the *Compound Functions* section in the *Editor* chapter.

Introduction

A process is a self-sufficient, modular block of computer instructions for the NCM. Processes are used to implement the logic of control strategies.

To make a process with the GPL Editor, you group the diagram into a process compound. A process compound is a functional grouping of blocks, connections, and text. A process compound can be translated and compiled into a process object. The process object can be downloaded to and executed by the NCM.

A process compound may be one diagram or a series of diagrams. If the compound consists of a series of diagrams, some of these diagrams may also be process compounds. Creating one compound inside of another is called nesting. A nested compound has at least one higher level compound. If multiple process compounds are nested, the Translator will create separate processes from each. Also, even though a process compound may be nested under another process compound, their processes execute independently.

When you designate a compound as a process, you need to specify a number of parameters in the process compound's database template. Since a process compound is an object, you need to assign it a unique software system\object name.

You also may assign a process period to the compound. The period is how often a process is to execute. You can change a period of a process in the template to a value other than 00:00:00, the default period. A 00:00:00 period means the process will not run periodically. That is, the period will run only when triggered or when directed to by a PERD block.

Lastly, in the template, you may select exemption of all triggers for this process.

Deciding on Process Compounds

When deciding which diagrams of a control strategy should be made into processes, follow these recommendations:

- Group diagrams into process compounds based on function and logic.
- Design process compounds so that they execute only as often as necessary.
- Group diagrams into process compounds based on common periods, priorities, triggers, and trigger exemptions.
- Group a diagram into a separate process if it needs to be manually commanded or time programmed.

Note: Pay particular attention to the following blocks that affect process execution: ABRT, BSEQ, CMD, DLAY, PERD, PULS, STOP, 2CMD, WAIT. Refer to *Process Execution* in the *Order for Process Execution* section for details.

- Do not make process compounds too large. Large processes can be more difficult to debug and will take longer to translate, upload, and download. Keep the diagrams to a manageable size. Also, if a process is too large, it increases the chance that it will be time-sliced. Consider the NCM memory requirements, which are as follows:

  Overhead per process: 1 Kb (approximately).
  Maximum memory allowed per process: 32 Kb.
  Maximum number of processes per NCM: 255.

- Make process compounds that contain the 210A or 260A object blocks as small as possible. These processes require that the status of the controller be polled over the L2 Bus, which responds much slower than the N2 Bus. Therefore, keep L2 Bus accesses to a minimum. If the number of L2 devices on a single L2 trunk is greater than 10, assign the process priority to 4 (the lowest).

Important Facts

The following are some important facts regarding processes. (For details about a particular block, refer to the *Function Blocks* chapter.)

- All operation blocks must be in a process. Within a process, operation blocks are governed by the same period, process priority, triggers, and exemptions.
- The period of a process is set by the Period parameter in the compound’s template or by a PERD block in the process. If more than one PERD block is used, the last executed PERD block sets the period. The period timer is started at the end of the process execution.
- When executed, a WAIT block suspends the execution of all blocks that follow it for the wait time specified. The wait timer is canceled if the process is disabled or is triggered by some other means.
- A process may be triggered by the expiration of a timer from a PULS, DLAY, or BSEQ block.
- When a STOP or an ABRT block executes, the remainder of the process will not execute.
- Shared variables and objects are needed to communicate data from one process to another.
- A binary connection line that connects two operation blocks across two different processes will automatically create a binary shared variable that is transparent to the programmer. If the value of the binary shared variable changes, it will cause the second (i.e., destination) process to trigger.
- When an ABRT block executes, you must enable the process before it can be triggered.
- If you are migrating an N1 ARCNET® job to an Ethernet network requiring a change in node address, you must recompile DDL and GPL process databases for the NC.

Process Triggering

A process is run in the NCM only when it is “triggered” to execute. The following events cause a process to execute in the NCM:

- when the process is downloaded as enabled
- when a Process Enable (PRC_ENA) command is sent to the process
- when the process is triggered with a manual or a time-programmed command
- when the process is triggered by a command block that is connected to its process compound block
- when the process is triggered by a command in another process
- when the value of a binary, non-exempted shared variable read in the process changes reliably
- when a triggerable, non-exempted attribute read in the process changes reliably
- when the process period expires
- when the NCM is warm or cold started

Process Priorities

Each process has a priority level associated with it. This priority affects the order in which a process is executed in relation to all others.

The four priority levels are 1 (highest), 2, 3, and 4 (lowest). You define a priority for a process in the database template of the process compound. If you do not enter a priority, Priority 4 will be used as the default.

When a process is triggered, it is entered on a process queue before it actually executes. For example, if the queue contains two processes--Process A at Priority 1 and Process B at Priority 2--Process A will execute before Process B, since it has a higher priority.

Which processes should be assigned higher priorities than others depends on the application. Usually, the processes that contribute to the safety of a building should be assigned higher priorities than all others. Also, if performance of a process is critical, then that process should be assigned a high priority (e.g., 1 or 2).

Restart Process

A Restart process is a special process that is used exclusively for performing required startup processing. For example, the Restart process can be used to change shared variables to values different from their initialization values. Also, the Restart process can determine occupancy time, or delay processing until field equipment comes back online. It is always the first process to execute when the NCM is warm or cold started.

You create a Restart process by making a Restart compound, which you do by selecting RESTART PROCESS in the Compound Type field of its database template. You also need to assign a valid system name. The object name is defined as "Restart."

Since the Restart process is the first process the NCM executes upon warm or cold startup, all other enabled processes will be in a queue until its execution completes without errors. (Processes disabled, in error, or not fully downloaded are not placed in the queue.) During this time, the status of the process is in the Held mode. The Restart process is optional, and it executes only once on a warm or cold startup of the NCM.

Important Facts

Here are some things to keep in mind regarding the Restart process:

- If the status of the Restart process is DISABLED, NOT FULLY DOWNLOADED, or ERROR at warm or cold start, neither the Restart process nor any other process will execute in the NCM.
- If the status of the Restart process is DISABLED during its execution, or it completes in error (executes an ABRT block or detects a fatal error), no other processes in the NCM will execute.

- The following blocks cannot be used in a Restart process: PERD, PULS, DLAY, and BSEQ. The Expert Checker verifies that none of these blocks are used in a Restart Process.

Note: The WAIT block can be used in a Restart process. However, execution of a WAIT block in a Restart process prevents any other process to execute.

- A single NCM can have *only one* Restart process. If two or more Restart processes with the *same* system name are created, then compiled, the last Restart process to compile becomes the actual Restart process the NCM will use. However, if two or more Restart processes with *different* system names are created, compiled, and then downloaded to the *same* NCM, the download will fail and an error message will display indicating the failure.
- The Restart process cannot be enabled, disabled, triggered, or time-programmed.
- The initialization values that the Restart process can change are summarized in the following table:

Type	Initialization Value
Analog (real)	0
Binary (logical)	False
Time	00:00:00
Integer	0
Comparison	False (applied to last known value of comparison)
Period	00:00:00
Attribute	0 (Analog or Integer) False (Binary) or 00:00:00 (Time)
Process Reliability	Reliable

Connecting Function Blocks

This section describes connections between function blocks. It covers the methods of connecting blocks and the types of connections. For detailed instructions on how to add, modify, and delete connections, refer to the *Block Connection and Command Functions* section in the *Editor* chapter.

Data and Control Flow

Drawing a connection line between two function blocks establishes the data flow or control flow on a diagram.

Data Flow

Data flow is the passing of data from one block to another. Three types of data can flow from one block to another: analog, binary, and time. Analog data might be a value of a temperature sensor. Binary data is an On (1) or Off (0) state. Time data could be how long a fan is on or the time of day at which the fan was started. The GPL Editor represents analog and time data by a solid white line; it represents binary data by a dashed white line.

Data flow is read in the direction of the connection arrows. For example, a connection line that has its arrowhead on the right means the data is flowing from left to right. Consider the diagram in Figure 19. In this example, the ENRH block reads the values of two AI sensors that report outdoor air temperature and relative humidity. It then calculates enthalpy, and the SVAR block reads and stores this value.

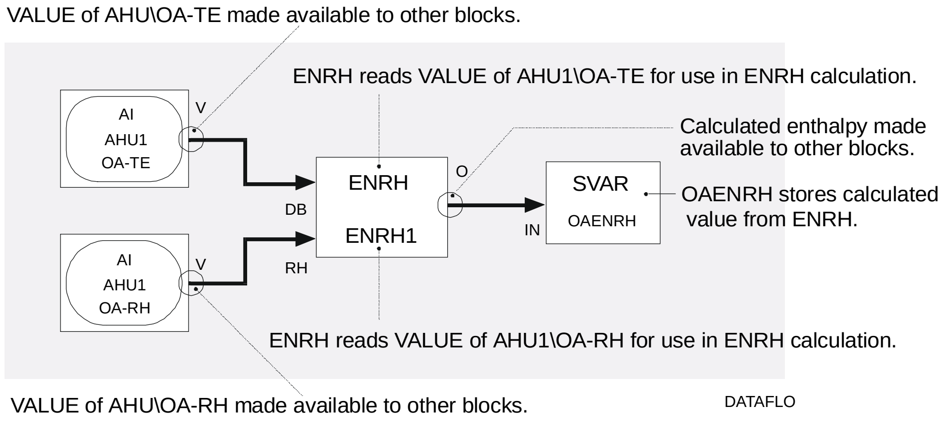

VALUE of AHU\OA-TE made available to other blocks.
ENRH reads VALUE of AHU1\OA-TE for use in ENRH calculation.
AI
AHU1
OA-TE
V
Calculated enthalpy made available to other blocks.
ENRH
ENRH1
O
DB
SVAR
OAENRH
IN
OAENRH stores calculated value from ENRH.
AI
AHU1
OA-RH
V
RH
ENRH reads VALUE of AHU1\OA-RH for use in ENRH calculation.
VALUE of AHU\OA-RH made available to other blocks.
DATAFLO

Figure 19: Example of Data Flow

Line Labels

Notice in Figure 19 that the ends of each connection line are labeled. These are called line labels that abbreviate the names of the connections. For example, the letter “V” at the AHU1\OA-TE block means VALUE, and the letters “DB” into the ENRH1 block mean DRY BULB temperature.

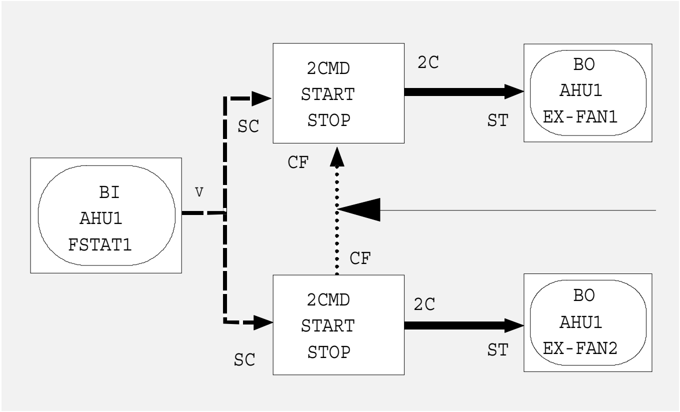
The following are characteristics of line labels:

- The Editor creates each line automatically after you complete a connection.
- The labels are placed at the beginning and ends of the connection line.
- The labels consist of one or two characters that abbreviate the connection name.
- Every function block has a unique set of line labels.
- For command blocks, the line label is based on the name of the command, not the name of the connection.
- Line labels also go into and out of compound blocks.

Control Flow

Control flow dictates the order in which certain blocks execute. Unlike a data flow line, the control flow line does not represent a value. Instead, a control flow line merely indicates the execution order of blocks. It is only needed when data flow lines make the order of execution ambiguous, or when you want to specify a particular order of execution. For example, you can use a control flow line for sequential commands. Control flow is represented by a dotted white line.

Figure 20 shows an example of a control flow. Notice that both ends of the control flow line are labeled “CF” (Control Flow). In this example, a control flow line is drawn from the lower 2CMD block to upper 2CMD block. This allows EX-FAN2 (Exhaust Fan -2) to be commanded first, then EX-FAN1 (Exhaust Fan -1). An important tip to remember for a control flow connection is that the origin block will always be the first block to execute (i.e., the block where the control flow line begins).



Control flow line indicates that lower START\STOP command will execute before upper START\STOP command.

CNTRLFLO

Figure 20: Example of Control Flow

Another example of where a control flow line may be needed is between two diagrams on the same screen. GPL cannot determine which diagram to execute first, so you need to use a control flow line to specify. Figure 21 illustrates. As shown, a control flow line is drawn from the upper CMD block to the lower CMD block. This means PUMP1 will be commanded first (when FSTAT1 is True and Reliable), then PUMP2 will be commanded (when FSTAT2 is True and Reliable). Another control flow line is drawn from the lower CMD block to the ADD block. This means the ADD block will execute after PUMP2 is commanded. In effect, the two control flow lines ensure the NCM will command AHU1\PUMP1 before AHU1\PUMP2.

Note: In the example of Figure 21, the ADD and SVAR blocks will still execute when AHU1\PUMP1 or AHU1\PUMP2 is not commanded.

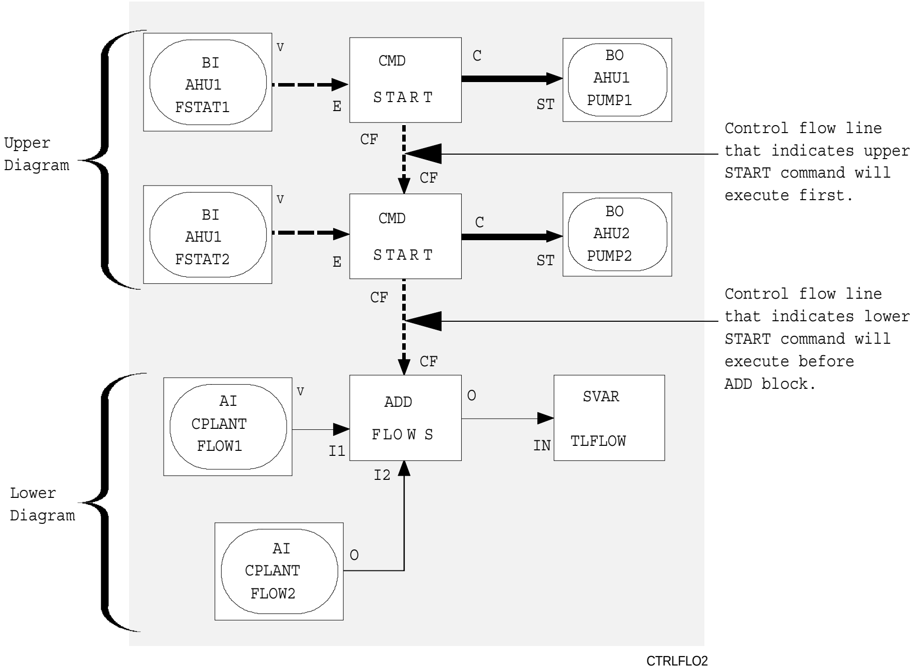


Figure 21: Example of Control Flow Lines Between Two Diagrams

As you may have noticed in the two previous examples, control flow lines are only drawn between operation blocks, not between object blocks. This is because only operation and special blocks have control flow inputs and outputs. Object blocks cannot have control flow lines because GPL does not create executable code from the object blocks as it does from operation and special blocks.

Forming a Loop

While making connections, you may form a loop between two or more blocks. If a loop is formed, the Editor cannot determine which block to execute first. The Editor detects a loop immediately and displays a loop detected error message. It also changes all lines that are involved in the loop to red and asks you to click left on the block that you want to execute first. This block is called the Loop-master block. After you click on the block, the lines return to white, and the line that enters the Loop-master block is encased in a circle. Figure 22 shows an example, in which a process reads the value of an SVAR block and adds 1.0 to it.

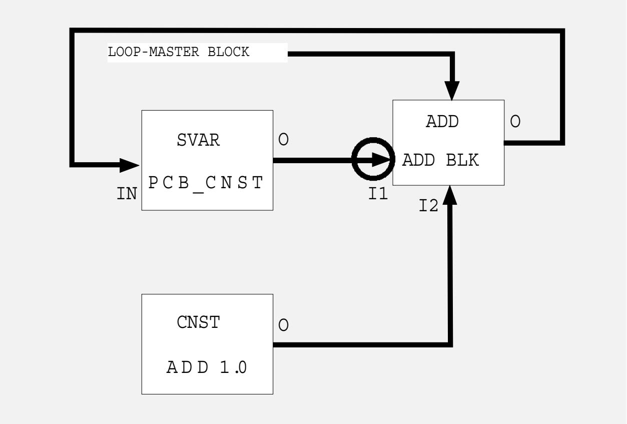


Figure 22: Loop Example Showing Use of Loop-Master Block

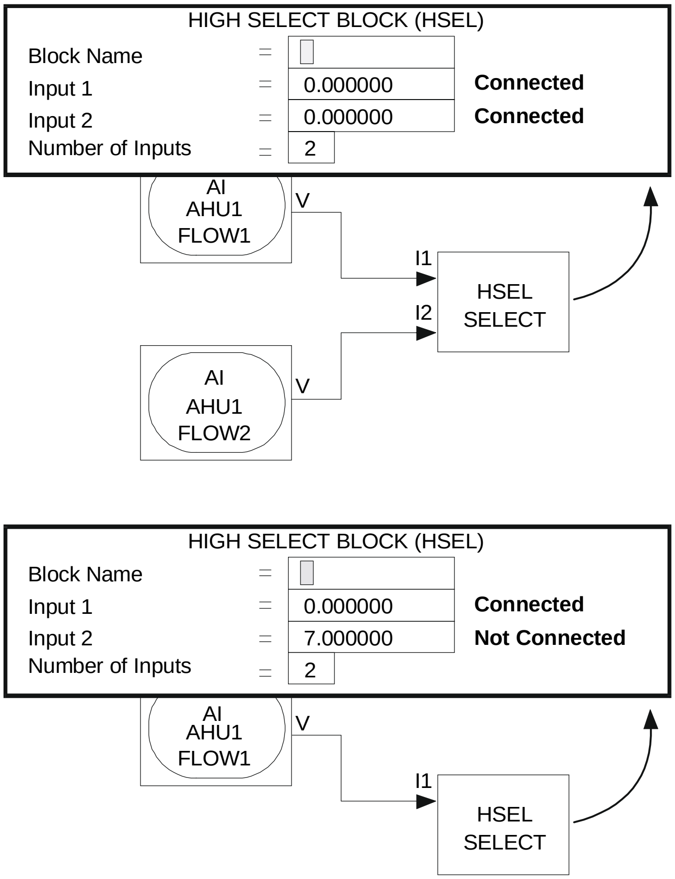
If the block that you want to make the Loop-master block is in a different diagram, you can go to that diagram using the standard method of moving between compounds. To cancel this operation before specifying a Loop-master block, click left anywhere outside the work area. The last connection made erases and the operation aborts.

Required and Optional Lines

Some line connections are required, while others are optional. An input line is required if (1) the block's algorithm needs the value it represents, and (2) the block has no equivalent template field for the value.

An output line is required if it needs to be read by other blocks.

An input line is optional if the block has an equivalent template field for the value. For example, the template for the HSEL block contains Input 1 and Input 2. If you enter a value for Input 2, you do not need to also terminate an Input 2 connection to the HSEL block. Figure 23 illustrates two equivalent methods of specifying Input 2.



HIGH SELECT BLOCK (HSEL)

Block Name =

Input 1 = 0.000000 **Connected**

Input 2 = 0.000000 **Connected**

Number of Inputs = 2

HIGH SELECT BLOCK (HSEL)

Block Name =

Input 1 = 0.000000 **Connected**

Input 2 = 7.000000 **Not Connected**

Number of Inputs = 2

SPCF-INP

Figure 23: Two Options in Specifying Input 2 (I2)

Note: If you connect an input externally, the GPL Editor uses the connection, not its equivalent template value. If you do not connect an input, the Editor uses the template value.

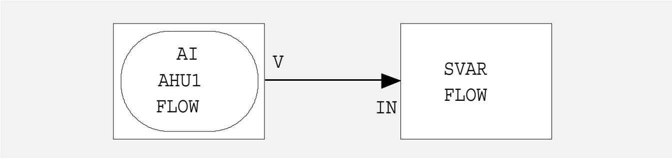
Other optional connections include control flow connections.

Types of Lines

The GPL Editor features nine different types of connection lines that are named after the data that they pass to other blocks. The types are analog, binary, time, command, dual command, read, write, control flow, and configuration. The Editor has different line types so that you can visually distinguish the data type, and so that the Editor can perform error checking while you are connecting (e.g., you could not connect a time line to a command input).

Analog

The analog type of line represents analog data, such as a setpoint or temperature. An analog line is solid white. In the following example (Figure 24), the SVAR block reads the value of an AI object that senses flow.

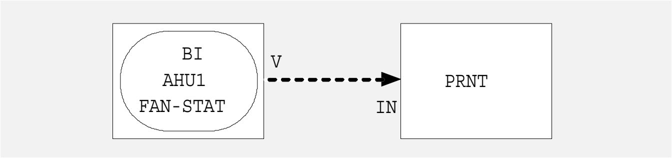


ANL-BIN

Figure 24: Analog Line Example

Binary

The binary type of line represents a True or False state. True is also known as On or 1, and False as Off or 0. A binary line is a dashed white line. In Figure 25, the PRNT block reads the value of a BI object that senses fan status, and if the value of the BI Input is True and Reliable, the PRNT block sends a message reporting its status to the printer.



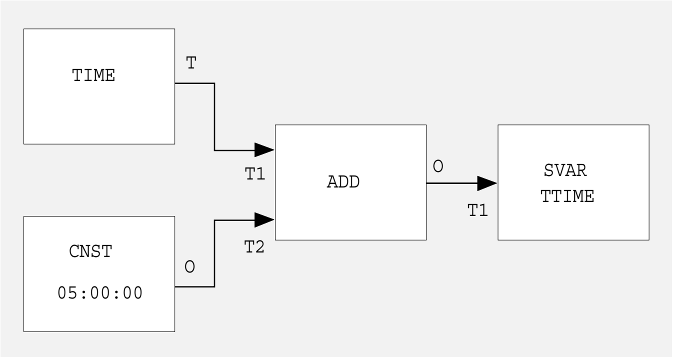
ANL-BIN

Figure 25: Binary Line Example

Time

The time type represents a time value. Time is specified in 24-hour format, in which hours, minutes, and seconds are separated with a colon (:). For example, ten seconds after 2:39 p.m. would be 14:39:10. Time arithmetic also uses 24-hour format (e.g., 3:00:00 - 14:00:00 = 13:00:00). A time line looks the same as an analog line—solid white.

In the following example (Figure 26), the ADD block adds the time values of the TIME and CNST blocks, and the SVAR block reads the value of the ADD block.

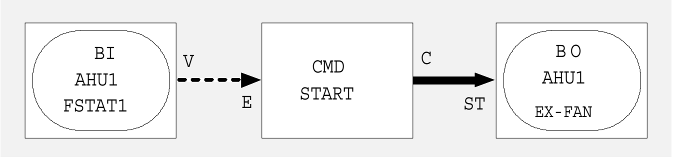


TYM-CMD

Figure 26: Time Line Example

Command

The command type sends a command from a single command block (CMD) to an object block. A command line is a heavy solid white line. In Figure 27, the BO object is commanded to Start when the BI object is True.



TYM-CMI

Figure 27: Command Line Example

Dual Command

The dual command type sends a command from a dual command block (2CMD) to an object block. A dual command line is usually the same as the single command line—a heavy solid white line. In Figure 28, the BO object is commanded to Start or Stop, based on the Select Command input.

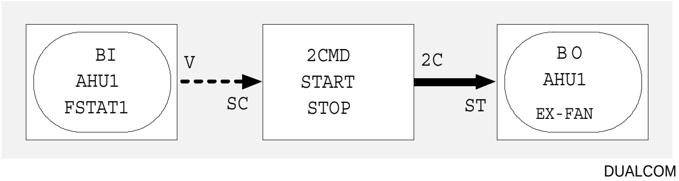


Figure 28: Dual Command Line Example

Read

The read type passes a readable attribute from an object block to the READ block. A read line is only started in an object block. You would use this type of line when the attribute you need to read is not available in the connection menu. The read line can terminate only to a READ block, or to a CONN block that is passing the read line to a READ block. A read line, like the two command lines, is a heavy solid white line.

In the following example (Figure 29), the PRNT block sends a message to the printer when the HI_WARNL (High Warning Limit) attribute of the AHU\FLOW object is greater than the AHU1\HWSETPT object (setpoint). The DFCM block compares HI_WARNL and the setpoint. If HI_WARNL is greater than the setpoint, with an applied differential, the PRNT block sends a message reporting the attribute’s status to the printer.

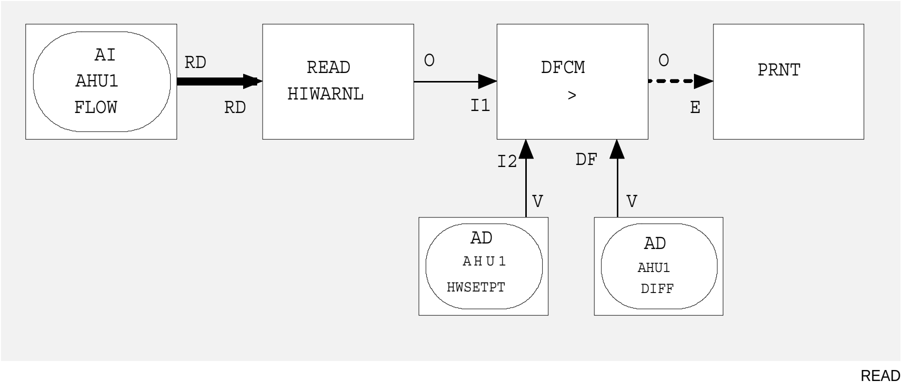


Figure 29: Read Line Example

Write

The write type sends a writable attribute from a write (WRIT) block to an object block. A write line is used only when the attribute that you need to change cannot be changed through a command. The WRIT block modifies an attribute only. The CMD or 2CMD block modifies an attribute and executes an algorithm, such as change-of-state analysis. A write line is a heavy solid white line.

In the example of Figure 30, the WRIT block changes the value of the HI_LIMIT attribute of an AI object, based on the season of the year. When the BD object indicates the summer season, the HI_LIMIT is 85.0°F. When the object indicates the winter season, the HI_LIMIT is 70.0°F.

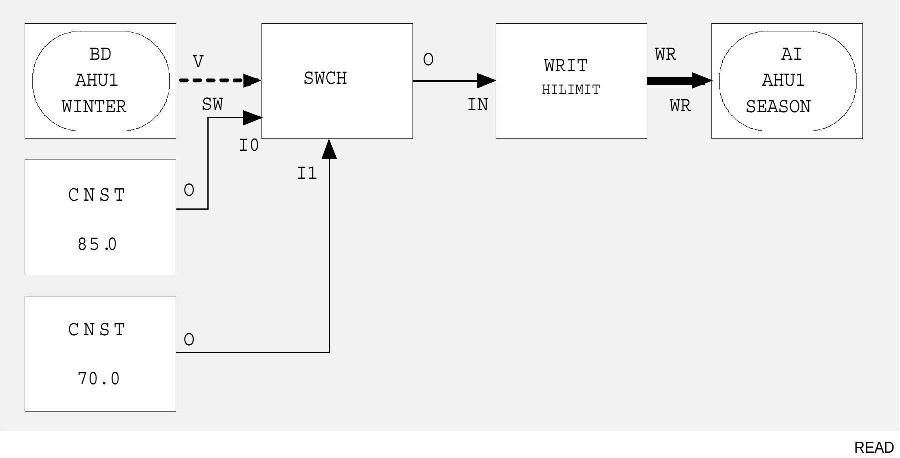


Figure 30: Write Line Example

Control Flow

A control flow line is a continuous line drawn between two function blocks to indicate the order of block execution. A control flow line is a dotted white line. You should use a control flow connection only when block execution order cannot be determined by the data flow lines. For example, in Figure 31, the object called AHU1\EX-FAN2 will be commanded before AHU1\EX-FAN1, as indicated by the control flow line from the lower CMD block to the upper CMD block. Notice that both ends of a control flow line are labeled “CF” (Control Flow).

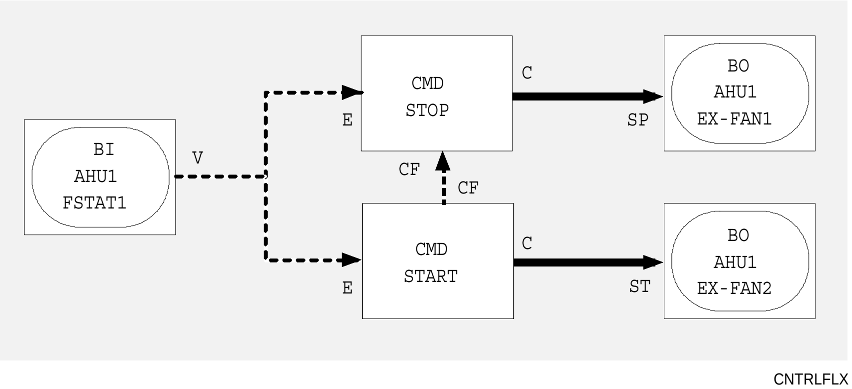


Figure 31: Control Flow Line Example

In a control flow connection, the origin block always executes before the destination block. Control flow lines are seldom used, since data flow lines are usually enough to specify block execution order. Also, control flow connections are allowed only between operation blocks; each operation block has a control input and a control output. Object blocks cannot accept control flow connections.

Configuration

The configuration type of line does not pass data the same way that data lines do. It is used to show a reference that is made within the template of an object block. To understand this concept, consider the example in Figure 32.

This example shows a configuration connection between an AI object and a PIDL object. It also shows the PIDL template, in which the system\object name for the AI object is referenced. The reference made in the template actually “connects” the two blocks. Therefore, the configuration line is not required but is drawn to show the connection graphically.

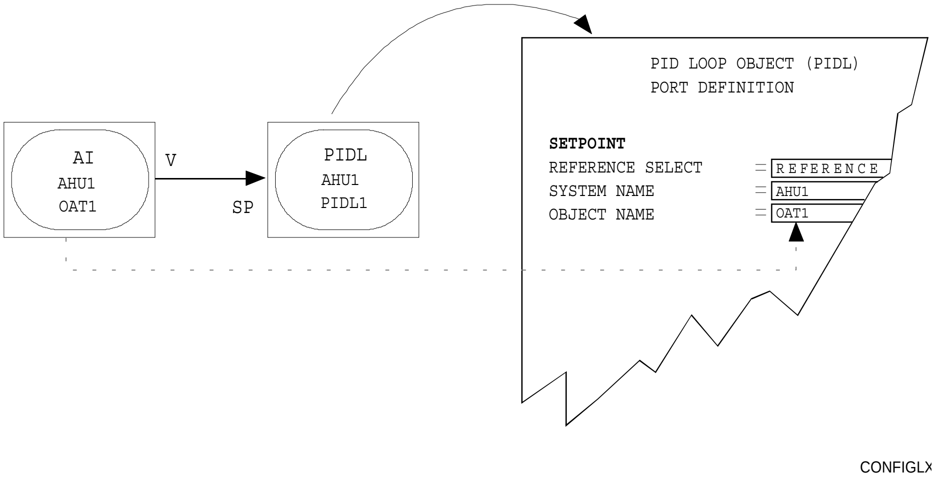


Figure 32: Configuration Line Example

Note: When you draw a configuration connection, the referenced system\object name is not automatically defined within an object’s template. You need to define the referenced system\object name in the template manually.

For clarity, we recommend that you draw all configuration connections between object blocks. In fact, configuration connections are necessary if you want to simulate them.

The way you can tell if a line is a configuration connection is if an object block is connected to another object block, either directly or through one or more CONN blocks. All connections between object blocks are configuration connections. In fact, configuration connections can only exist between object blocks. The Expert Checker verifies that the configuration lines drawn match the reference in the template.

A configuration connection uses the data flow line representation. That is, a configuration connection for analog data is a solid white line, and for binary data, a white dashed line.

Figure 33 shows two additional examples of configuration connections. The top example shows the BI as the feedback object for a BO object. The bottom example shows the AI as the feedback object for an AOS object.

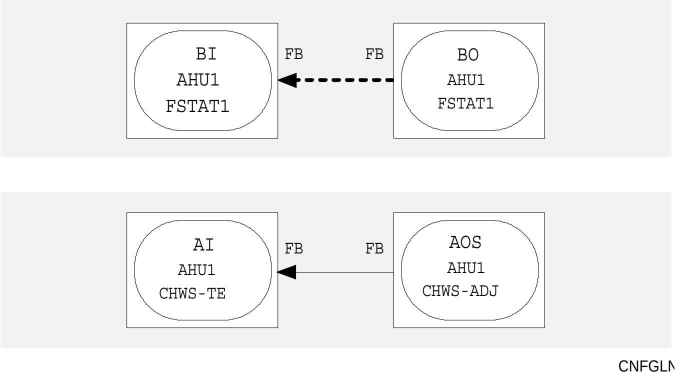


Figure 33: More Configuration Line Examples

In summary, the line types and their visual representations are:

Line Type	Visual Representation
Analog	Solid
Binary	Dashed
Time	Solid
Command	Solid/Heavy
Dual Command	Solid/Heavy
Read	Solid/Heavy
Write	Solid/Heavy
Control Flow	Dotted
Configuration	Solid (analog and time) or Dashed (binary)

Direct Connections

A direct connection is any continuous line between two function blocks that are contained in the same file. The blocks can be either on the same diagram or in different compounds. A direct connection made on the same diagram is the most common and most straightforward type (Figure 34).

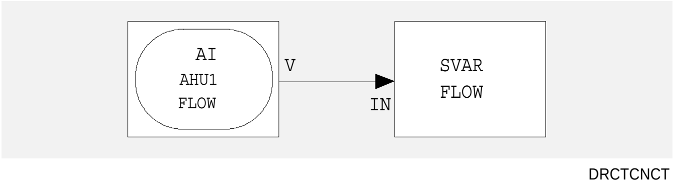


Figure 34: Example of a Direct Connection

A direct connection between two compounds usually requires that the line goes off the edge of the work area. As Figure 35 indicates, the line from the HSEL block runs off the edge of the screen, having no indication of where it is headed.

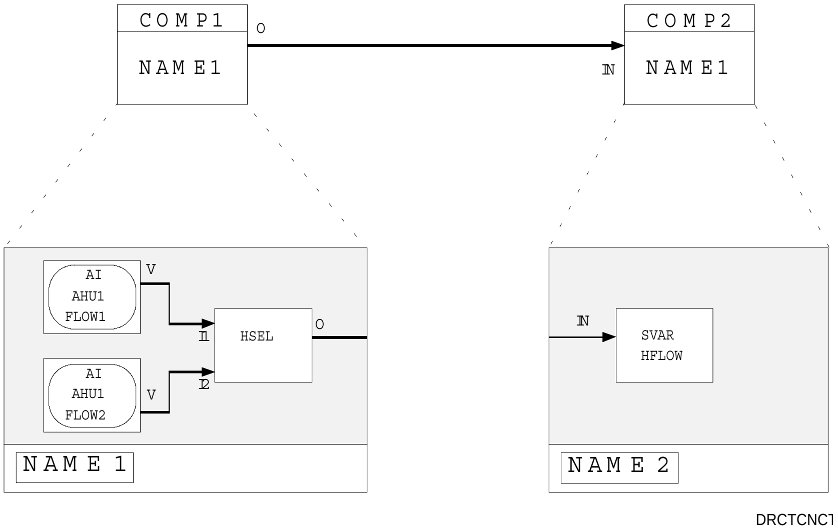


Figure 35: Example of a Direct Connection Between Two Compounds

The line that goes off the work area may use a Connection (CONN) block, which is placed near the edge of the screen before the line exits the screen. Figure 36 repeats the example of Figure 35, but uses a CONN block to identify the line from the HSEL block.

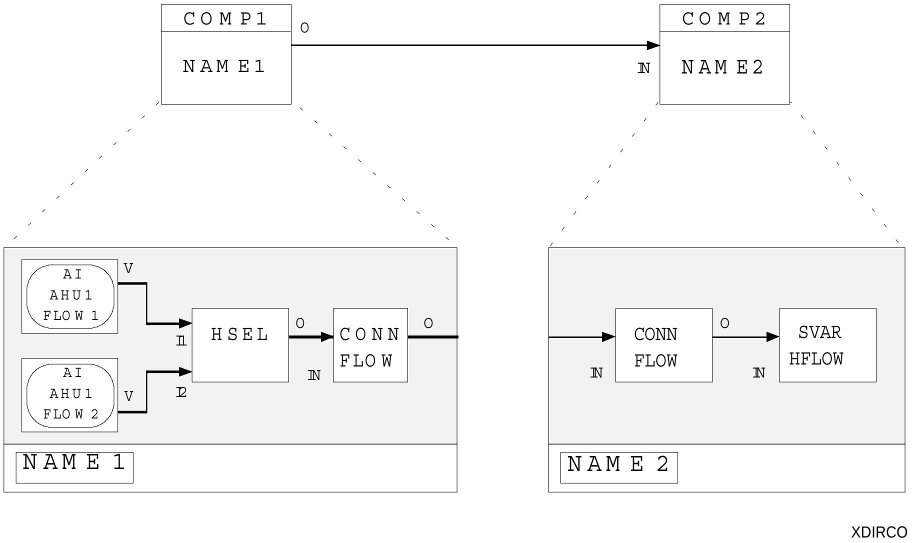


Figure 36: Example of a Direct Connection Using Connection Blocks

The CONN block serves as a line label to identify the destination of the connection; it also serves as a throughway for the data between the origin and destination blocks. The CONN block is explained in more detail in the *Function Blocks* chapter.

Important Facts

Keep in mind these factors that apply to direct connections:

- A direct connection can be any of the nine line types discussed earlier.
- A direct connection can consist of up to 14 line segments. The error message `Too many connection segments` displays if you try to draw more than 14 segments. A direct connection cannot penetrate more than ten compound levels.
- The origin and destination connection points must be of the same data type. For example, you cannot connect a binary output to an analog input. If you try to do so, an appropriate error message will display.
- You may have to change the block's data type in its template before connecting to or from it. For example, if you want to enter time data into an ADD block, you need to configure the ADD block to accept time data before making the connection (analog data is the default). Also, for those blocks that have a variable number of inputs (e.g., AND block), the number of inputs must be specified before the connection menus provide the proper number of inputs.

Remote Connections

A remote connection is a noncontinuous line that connects two blocks. The remote connection is primarily for connecting blocks in different compounds, although you may also remotely connect two blocks on the same compound. A remote connection eliminates the need for drawing long lines that cross compounds. Figure 37 compares the two methods of connecting blocks.

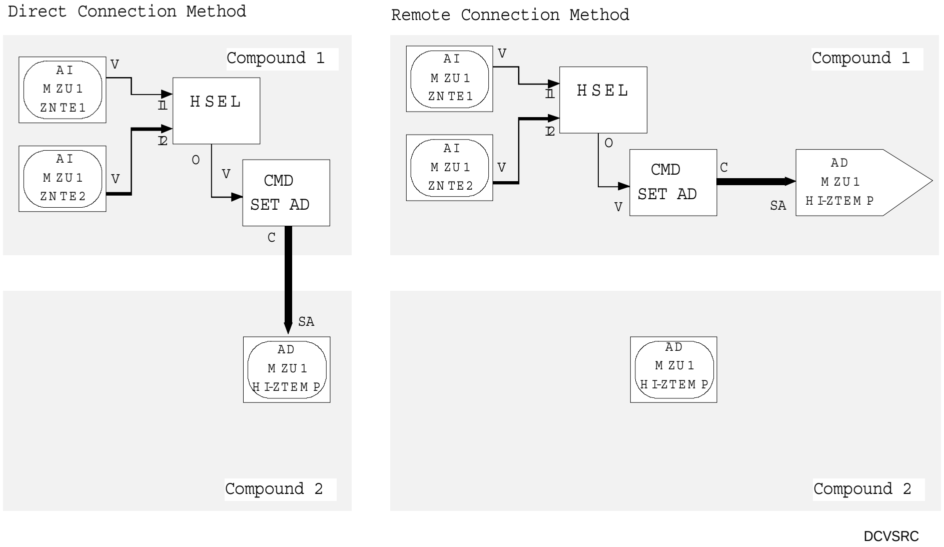


Figure 37: Direct Connection vs. a Remote Connection

A remote connection is equivalent to a direct connection. It differs from a direct connection in that it is represented by an arrowhead called a remote connection symbol (Figure 38). The connection line exits from the “V” side of the symbol only. As Figure 38 shows, the remote connection symbol has four orientations.

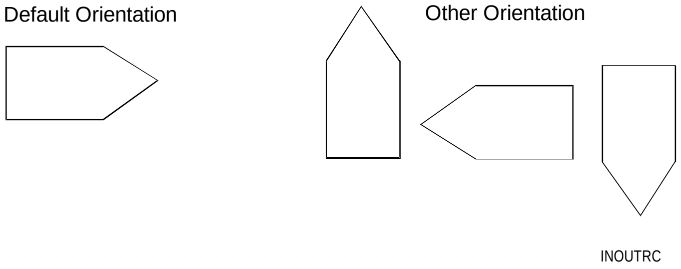


Figure 38: Shape and Orientations of Remote Connection Symbol

The GPL Editor offers two types of remote connections: origin remote connections and destination remote connections. The origin remote connection references the origin block, in which the remote connection enters a destination block (Figure 39). The destination remote connection references the destination block, in which the remote connection accepts an input from an origin block (Figure 39). As shown, each end of the line on a remote connection is labeled just as direct connections are labeled.

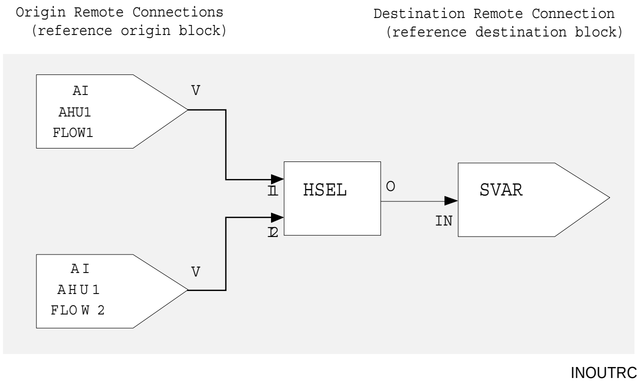

Origin Remote Connections
(reference origin block)
Destination Remote Connection
(reference destination block)
AI
AHU1
FLOW1
V
I1
HSEL
O
SVAR
IN
I2
AI
AHU1
FLOW2
V
INOUTRC

Figure 39: Input and Output Remote Connections

A remote connection symbol points to its reference. The symbol has the same characteristics as its referenced block: It is labeled with the same block name or system\object name, and you can query the template by clicking on the remote connection symbol (with the Query mode selected).

Important Facts

Keep in mind these factors that apply to remote connections:

- Two blocks can be connected remotely *only* if they exist in the same strategy file. A remote connection between two files is not allowed.
- Deleting a remote connection symbol does not also delete the referenced block, since it is a connection, not a block.
- You must name a block for it to be remotely connected.
- When making a remote connection, if there are two blocks with the same name, you cannot distinguish between them on the remote connection menu. Therefore, assign a unique name to each function block.
- When you compound a remote connection symbol without its referenced block, the symbol becomes unlabeled.

Fan-In and Fan-Out Connections

Fan-in and fan-out connections are allowed between function blocks. A fan-in connection is multiple outputs fed into one input. A fan-out connection is the same output going into multiple inputs.

Figure 40 shows an example of a fan-in connection. Even though the line labels into the BO object are different, they are going to the same input connection. Only the following types of lines allow fan-in: command, dual command, write, and control.

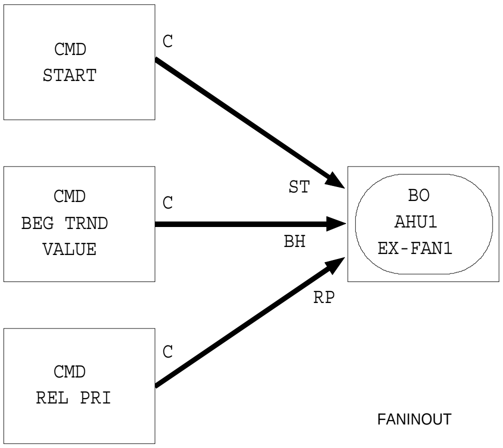


Figure 40: Fan-In of Multiple Commands to an Object

Figure 41 shows an example of a fan-out connection. The three VALUE outputs from the AI object each have the same value. All output connections allow for fan-out, except the command and write outputs of the USER block. A fanned-out connection is especially useful to an object block, since only one instance of an object is allowed per strategy file.

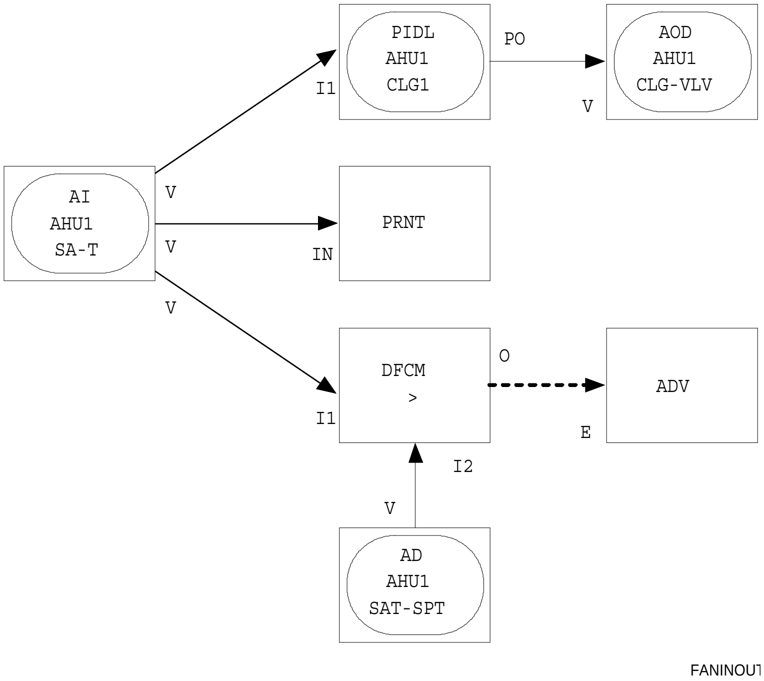


Figure 41: Fan-Out from an Attribute of an Object

Figure 42 shows a second example of a fan-out connection. A single CMD block sends the same Lock Reports command values to all three object blocks. The type of command that you fan out must be valid for each object.

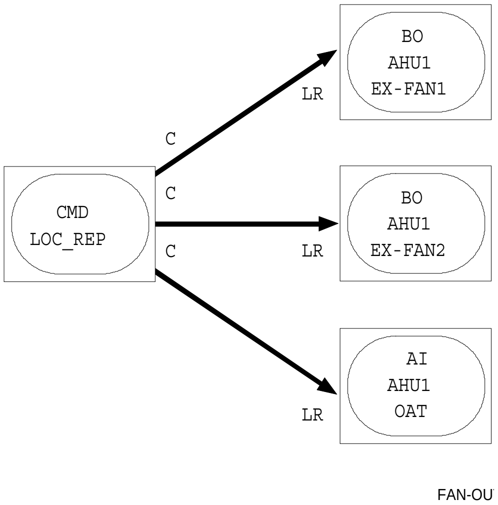


Figure 42: Fan-Out from a Single Command to Multiple Objects

Commanding Objects and Processes

This section discusses commanding objects and process objects. For detailed instructions on how to command objects and processes, see the *Block Connection and Command Functions* section of the *Editor* chapter.

Commanding Objects

You can send a command to an object by connecting a command block to it. Two command blocks are offered: Single Command (CMD) and Dual Command (2CMD). The CMD block is used for commands, for example: SET_PIDL, BEG_TRND, and START. The 2CMD block is for commands that are most often used in pairs, for example: START/STOP, BEG_TOT/END_TOT, and BEG_TOT/STOP. Figure 43 shows two examples of commands to objects.

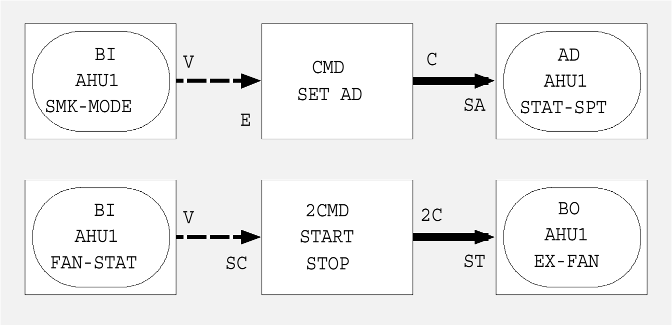


Figure 43: Examples of Commanding an Object

In the top example of Figure 43, a SET_AD command is sent to an AD object when the input of the BI object is True. In the bottom example, a Start command is sent to the BO object when the Select Command input of the BI object is True (1). A Stop command is sent when the Select Command input is False (0).

Note: Notice in Figure 43 that SET AD in the CMD block is separated with spaces, not with an underscore (i.e., SET AD instead of SET_AD). The GPL Translator inserts the underscore during translation. This is true for all commands that require underscores.

The command output connection from a command block (CMD and 2CMD) to an object block must be drawn before the command block can accept a parameter input. This is because the Editor needs to know which command you chose in order to present the correct parameters. (However, if the CMD or 2CMD block is connected to a CONN block, this rule does not apply.) Figure 44 illustrates the steps for drawing the output and input connections for a 2CMD block. (CMD block also applies.)

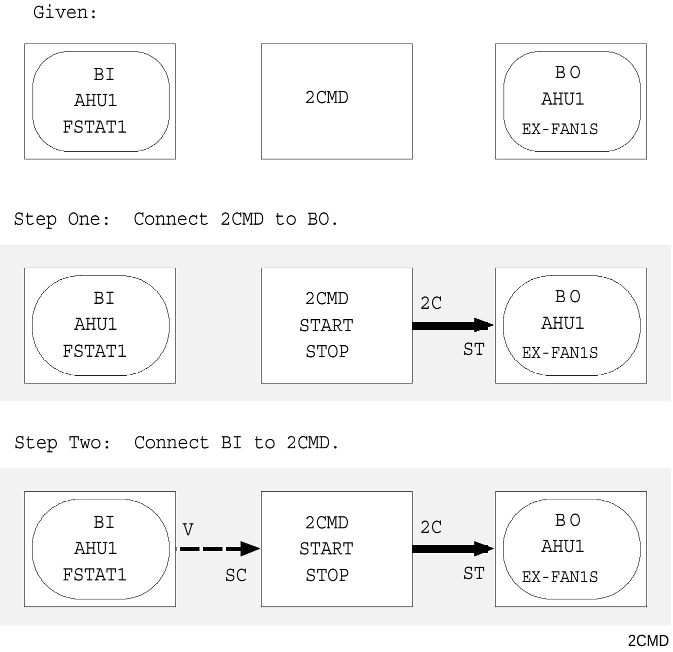


Figure 44: Order of Connecting a 2CMD Block

The command blocks send data to objects. The other operation blocks read their inputs and calculate an output that other blocks can read. Also, the command blocks are different from other operation blocks in that you select the command block name by selecting a command name in the connection menu.

You may send a command to any object at any NCM in the Metasys Network. The object's system name describes to which NCM it belongs.

Commanding Process Objects

A process compound can be commanded by connecting a CMD or 2CMD block to its block. Three possible commands to compounds are available: Enable Process, Disable Process, and Trigger Process. Figure 45 shows an example of each.

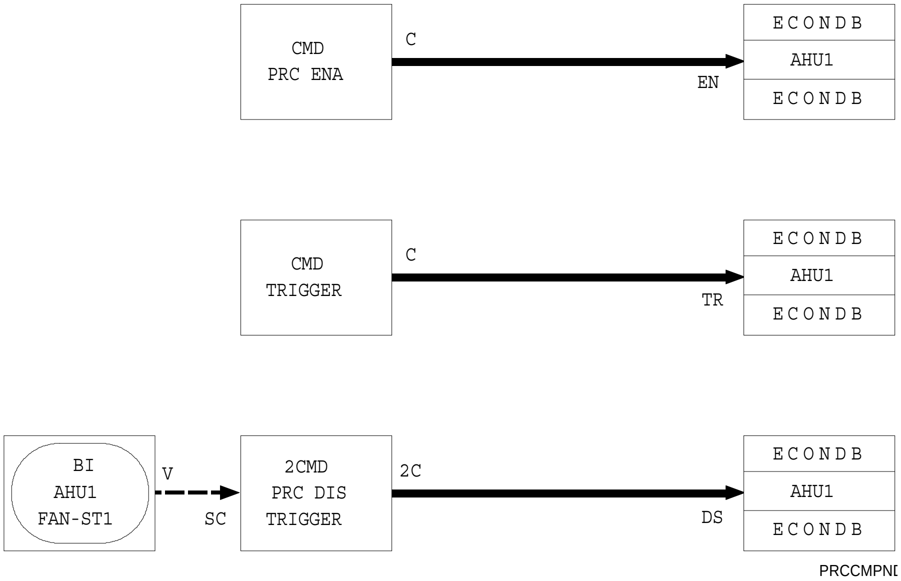


Figure 45: Choices in Commanding a Process Compound

Documenting Control Strategies

This section describes three forms of documentation for control strategies: text, analog displays, and text description file. For detailed instructions on how to add text and analog displays to a diagram, see the *Tools Functions* section of the *Editor* chapter. For instructions on how to prepare a text description file, see *Appendix E: External Functions*.

Text as Comments

The text function allows you to type comments on any space on a diagram (Figure 46). These comments may explain the purpose of the diagram, label the diagram, or contain any other helpful information.

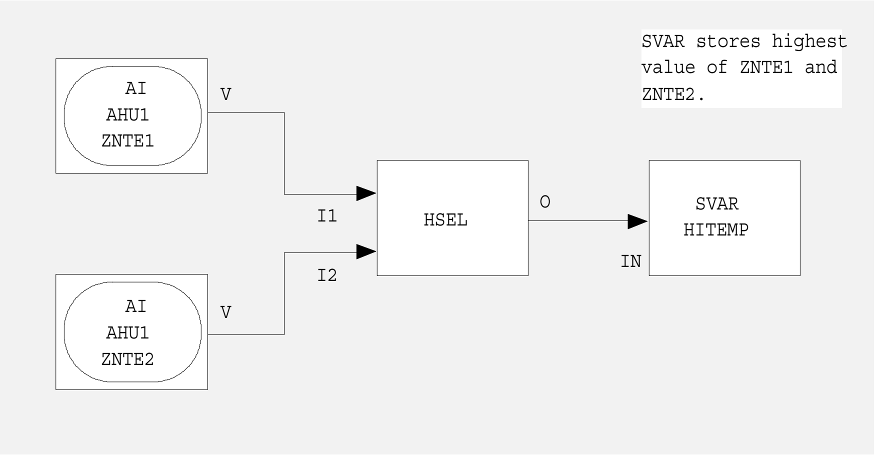


Figure 46: Comments Written on the Screen (Left Justified Text)

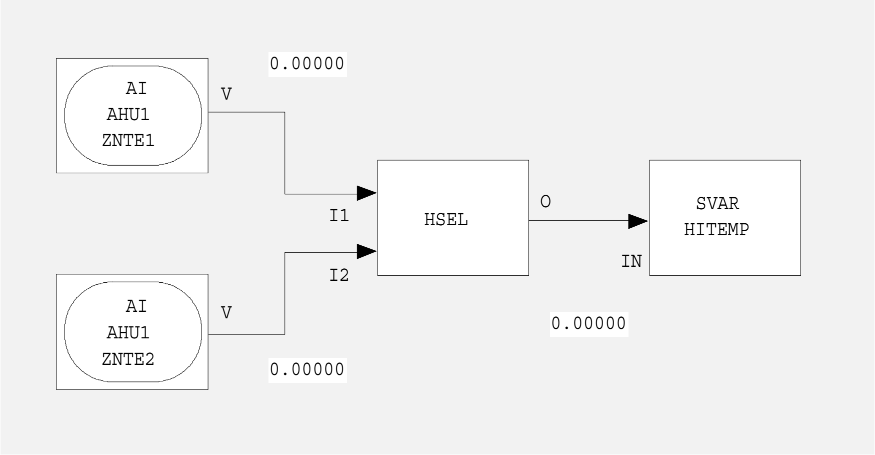
Important Facts

Keep in mind these factors that apply to text:

- Text is written to the screen at a preset size, which on a 12-inch monitor is about 1/4 inch. The Editor does not provide various type sizes, though you can resize text with the Move icon (Scissors).
- Text can include any combination of upper and lower case letters, international language characters, numbers, and most other characters that a standard keyboard offers. The characters not supported are: _ (underscore), | (vertical bar), ` (back apostrophe).
- Text in GPL, just as comments in most programming languages, is not translated into object code. The GPL Translator simply ignores text.

Analog Displays for Simulation

Analog displays allow you to show the analog output value of a block during simulation. Only the GPL Simulator uses analog displays. They provide a convenient readout of the block’s output without having to select the block and display the output value in the Block Window. In the Editor, analog displays are simply zero values that are placed next to an output connection (Figure 47) or elsewhere in the work area. The Expert Checker and the Translator ignore analog displays. Binary and time values cannot be represented by analog displays. See the *Simulator* chapter for details.



XANLDIS

Figure 47: Example of Analog Displays

An analog display is associated with a particular analog connection. The association is made when you select a block's name and its output connection name. You can position the display anywhere on a diagram, but it is best if you paste it close to the block or analog line it represents.

In addition to showing the analog display of a connected line, you may also paste down an analog display for an output that is not connected. Simply select the name of the output from the connection menu and paste down the display in any convenient spot on the diagram.

You can paste down multiple analog displays for the same output. For example, you can have an analog display for an AI object next to its output and duplicate the display at another compound level.

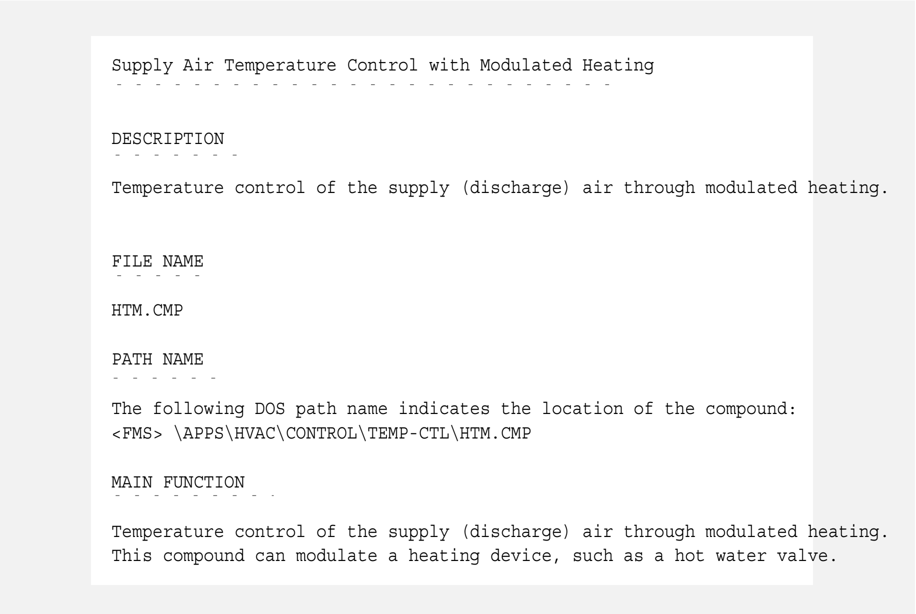
Various decimal resolutions are available, such as 0.00000 and 0.0. You select the resolution before you paste it down. It can also be changed to a different resolution later if you wish.

Text Description File

Each control strategy and compound can have a description file. The purpose of this file is to explain the control logic of the strategy or compound. The file can also be used to:

- Summarize functions.
- Document the sequence of operation.
- Explain the control objectives.
- Contain the author and date of creation.
- List the required input and output connections.
- Contain any other pertinent information.

Figure 48 shows an example of a description file for a supply air temperature control application.



```
Supply Air Temperature Control with Modulated Heating
- - - - - - - - - - - - - - - - - - - - - - - - -


DESCRIPTION
- - - - - - -

Temperature control of the supply (discharge) air through modulated heating.


FILE NAME
- - - - -

HTM.CMP

PATH NAME
- - - - - -

The following DOS path name indicates the location of the compound:
<FMS> \APPS\HVAC\CONTROL\TEMP-CTL\HTM.CMP

MAIN FUNCTION
- - - - - - - - -

Temperature control of the supply (discharge) air through modulated heating.
This compound can modulate a heating device, such as a hot water valve.
```

TXTDSC

Figure 48: Text Description File

To view the description file of a control strategy, select a file and click left on the Description icon under the File option menu. To view the description file of a compound, select a file and click left on the Description icon under the Compound option menu. An advantage of the description file is that you can view it without having to load its strategy or compound.

You prepare the description file outside of GPL with a text editor. For details, refer to *Appendix E: External Functions.*

Order of Process Execution

This section describes the execution order of blocks in a process. The order of execution of a process and between processes is important because the order determines how the process will be translated by GPL and executed by the NCM.

Process Level

Within a process, the order of execution is dictated by the direction of the data and control flow lines. No priority is given to any particular data line. The rules for determining the order of execution are the following.

1. An operation block executes after all operation blocks from which it receives data execute.
2. An operation block executes after all operation blocks from which it receives control flow lines execute.
3. If Rules 1 and 2 cause a loop, the Editor will ask you to select the block that should execute first.
4. An operation block receiving data from an object block executes after all operation blocks commanding that object execute. Note that object blocks execute independently of a process.

To understand how to apply these rules, consider the following examples:

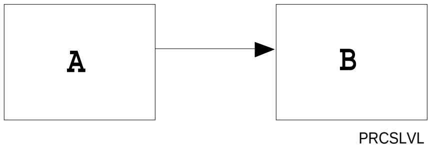


Figure 49: Demonstrating Rule 1

In Figure 49, a data flow line between Blocks A and B determines that Block B will execute after Block A (Rule 1).

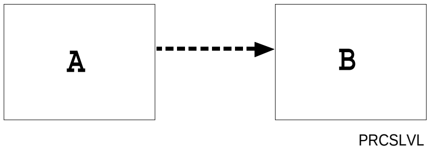


Figure 50: Demonstrating Rule 2

In Figure 50, a control flow line between Blocks A and B determines that Block B will execute after Block A (Rule 2).

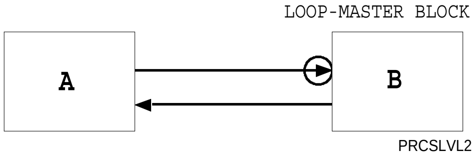


Figure 51: Demonstrating Rule 3

In Figure 51, a loop exists between Blocks A and B, and Block B is the Loop-master block (circled arrow connection). Block B will execute before Block A (Rule 3). The same holds true if Block B were in a different process in the same strategy file.

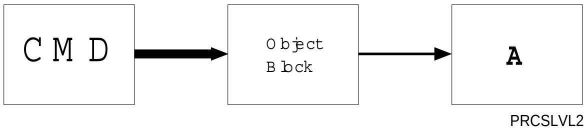


Figure 52: Demonstrating Rule 4

In Figure 52, Block A reads an attribute of an object that is commanded by a CMD block. Block A executes after the CMD block executes (Rule 4).

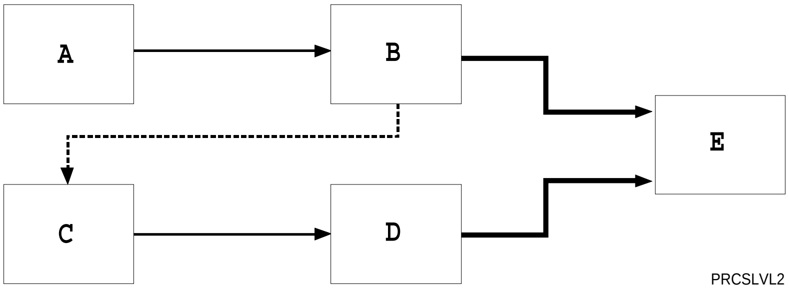


Figure 53: Using a Control Flow Line to Specify Execution Order

In Figure 53, the data flow lines between the five blocks, and the control flow line between Blocks B and C, determine that the order of execution will be: A, B, C, D, and E.

Without the control flow line, the order of execution would be: A before B, C before D, B and D before E. Note that the order of execution cannot be determined between A and C, A and D, B and C, and B and D.

The order of execution cannot be determined except by the four rules stated previously. If the order of execution is important, use control flow lines.

Order of block execution is consistent across group compounds in the same process. For instance, as shown in Figure 54, Block 3 has a connection line drawn to Block 4, but Block 4 is in another compound. In this case, Block 3 executes before Block 4.

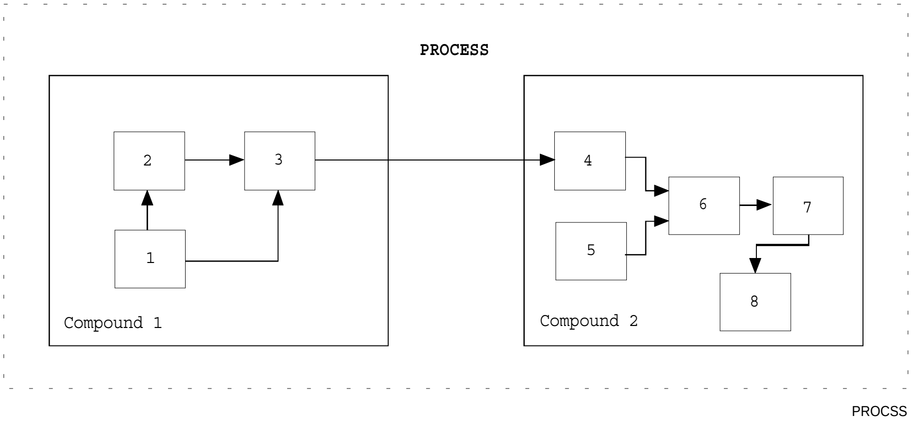


Figure 54: Order of Block Execution Across Two Compounds in Same Process

Process Execution

The Interpreter inside the Network Control Module executes GPL processes. The order in which it executes processes is first by their assigned priorities (1, 2, 3, or 4, where 1 is the highest priority, and 4 is the lowest), then on a first-come, first-served basis. A higher priority process is always executed before a lower priority process. Processes of the same priority are executed on a first-come, first-served basis.

A GPL process must be downloaded for it to execute. The download operation sends process objects to the proper NCM over the N1 LAN. After the process is downloaded, the NCM executes it once.

A GPL process will execute from its beginning in the NCM when one of these events occurs:

- The process is enabled or downloaded as enabled.
- A process Enable (PRC_ENA) command is sent to the process.
- The process is triggered with a manual or a time-programmed command.
- The process is triggered by a command block that is connected to its process compound block.
- The process is triggered by a command in another process.
- The value of a binary, non-exempted shared variable read in a process changes reliably.
- A triggerable, non-exempted attribute read in the process changes reliably.
- The process period expires.
- The NCM is warm or cold started.

A GPL process will execute where it left off when one of these events occurs:

- A wait timer of the process expires.
- A time-sliced process has a chance to continue its execution. A process is time-sliced when it contains so many instructions that it cannot finish its execution.

Blocks That Affect Execution

Several function blocks affect the normal execution of a process. They are Period, Wait, Stop, Abort, Pulse, Delay, and Binary Sequencer. The behaviors of each are summarized below.

Period Block

A Period (PERD) block changes the periodic interval at which the process executes. A trigger that occurs while the period timer is running cancels the timer and runs the process immediately. This block does not guarantee the process will execute once every period, since another PERD block may exist in the same process and may modify the period value. Also, a process with a five second period, for example, will execute approximately every five seconds, since there is some overhead processing that may delay execution by a second or more. The last PERD block in a process to be executed dictates which period is used. The period timer is set at the end of the processes’ execution (i.e., after the last block or immediately after a STOP block executes).

Wait Block

The Wait (WAIT) block suspends execution of an entire process for a specified period of time. The process resumes where it left off when the wait timer expires (i.e., not from its beginning). The timer is canceled if a trigger occurs while the process is in the Waiting state. When the trigger occurs, the process is executed from its beginning.

Stop Block

This block stops the execution of a process immediately and places it in the Ready state. When this block is executed and its enable input is true, the process halts. The blocks that follow the STOP block will not execute. When the process is triggered again, the process executes from its beginning.

Abort Block

The Abort (ABRT) block halts process execution and sets the process to the Error state. The process may not be executed again until a manual command enables the process (manual, process, or time-programming command).

Pulse Block

The Pulse (PULS) block maintains an output of True for a specified time. The block triggers its process after the time period expires. The execution restarts at the beginning of the process.

Note: A PULS block configured as a one-shot does not trigger the process. Also, a PULS block configured as cancelable may cancel the timer under certain conditions. Refer to the *PULS* block in the *Function Blocks* chapter.

Delay Block

The Delay (DLAY) block delays the change of data to True for a specified time. The execution restarts at the beginning of the process after the time period expires.

Note: A DLAY block configured as a one-shot does not retrigger the process. Also, a DLAY block may cancel the timer under certain conditions. Refer to the *DLAY* block in the *Function Blocks* chapter.

Binary Sequencer Block

Internally, the Binary Sequence (BSEQ) block uses a cancelable pulse function and may trigger the process.

For details on any of these blocks, see the *Function Blocks* chapter.

Making Lines Exempt

A triggerable binary attribute or binary shared variable may be suppressed from triggering the process by marking its connection line “exempt.” An exempted line is indicated with an open white square on the input side (Figure 55). You would exempt a line if it represents a binary attribute or binary shared variable that, if changed, will cause unnecessary or unwanted processing.

For example, Figure 55 shows a BO object that represents the lights in a conference room (AHU1\CONF-LTG). Also shown are two BI objects, one representing fan status (AHU1\FSTAT1), and the other a motion detector (AHU1\CONF-OCC). The lights in the conference room are commanded to On when the fan is On, and the motion detector indicates that the room is occupied. However, the connection from the AHU1\FSTAT1 object to the AND block is made exempt, so that this process should not be triggered every time the fan changes state.

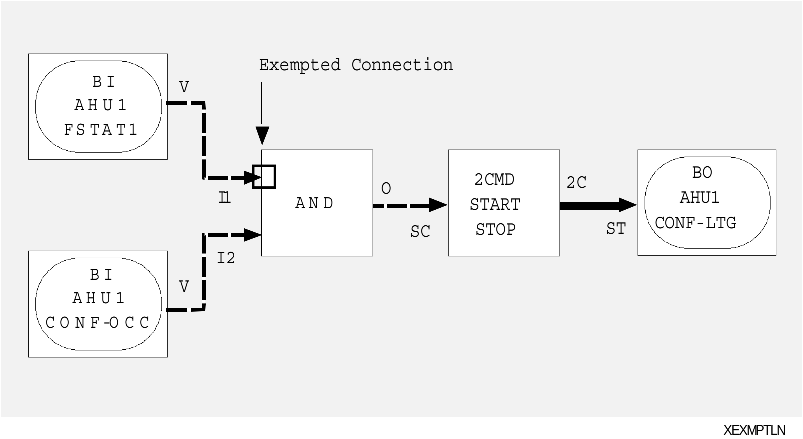


Figure 55: Example of an Exempt Line

Important Facts

Keep in mind these factors that apply to exempt lines:

- You need only exempt a binary triggerable attribute in one diagram in the process to make the binary attribute exempt in all locations. All other uses of this binary attribute are made exempt also, though only one line is actually marked exempt.
- You cannot exempt non-triggerable attributes and analog or time shared variables. The Expert Checker validates all exempt connections.
- You cannot exempt an attribute that is read by a READ block. If you need to exempt such an attribute, use a USER block.

Conditional Execution

Conditional execution is the IF...THEN...ELSE logic that some textual programming languages use. The GPL Editor does not provide blocks named IF, THEN, and ELSE. It instead uses a connection type called Enable to provide for conditional execution. Enable is offered as both an input and output connection.

You can draw conditional execution between two function blocks or between two process compounds. Figure 56 shows conditional execution between function blocks. The example shows two methods of writing conditional logic: a textual representation using JC-BASIC, and a graphical representation using GPL. The representations are equivalent.

TEXTUAL REPRESENTATION

```
IF 'AHU\BI1\VALUE' THEN


TELL 'AHU\BO1' TO "START"



TELL 'AHU\BO2' TO "STOP"



TELL 'AHU\BO3' TO "START"


ELSE
    TELL 'AHU\BO3' TO "STOP"
END IF
```

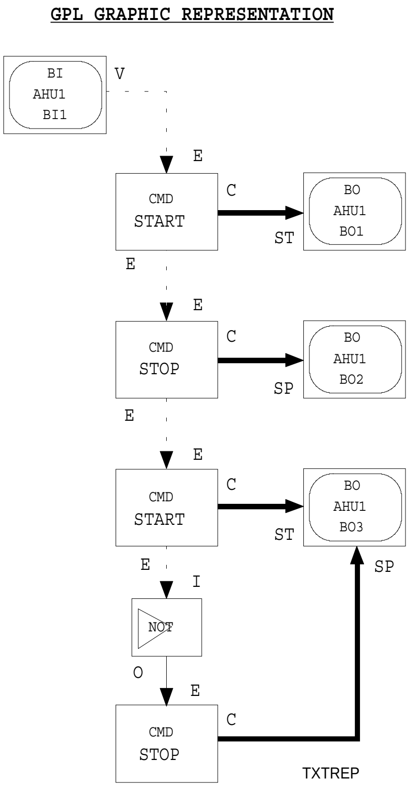


Figure 56: Textual and Graphical Representations of Conditional Logic

Figure 57 shows conditional execution between process compounds, in which the RAMP compound, when it is triggered, triggers the DMP compound. A CMD block for triggering is used.

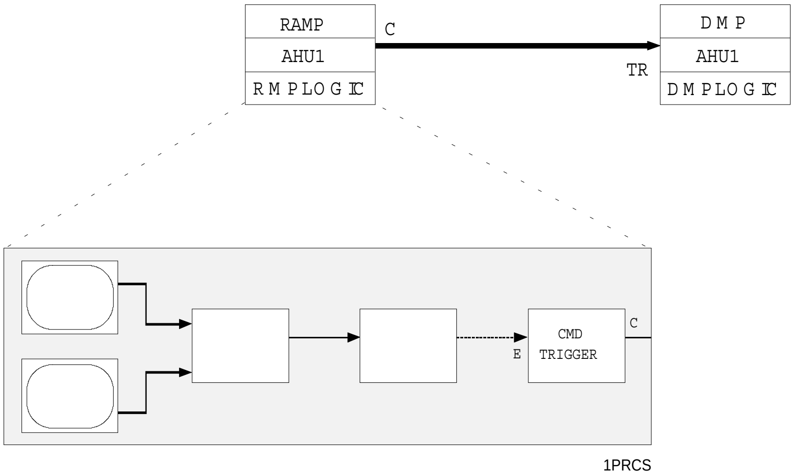


Figure 57: Example of One Process Triggering Another

Conditional execution is available only to the following operation blocks: ABRT, ADV, CMD, PERD, PRNT, READ, STOP, SVAR, 2CMD, WAIT, and WRIT.

One-Shot Execution

The “one-shot” execution is a special case of conditional execution. This type executes only once under certain conditions, then not again until conditions change. The PULS block configured as one-shot provides for this type of conditional logic.

Figure 58 has an example of a PULS block placed between a BI block and a CMD block. When the output of the BI object changes from False to True, the PULS block goes True for one execution, which enables the CMD block to issue a Start command to the BO object. Since a One-shot PULS block is used, the output of the PULS block is True for only one execution, and then it is False. If the BI object stays True, no additional Starts will be sent. Another Start command is sent when the condition of the BI changes from True to False to True again. The PULS block is described in the *Function Blocks* chapter.

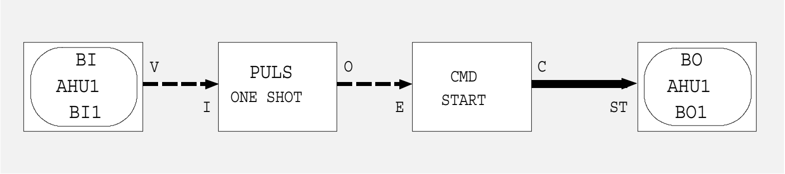


Figure 58: Example of a One-Shot PULS Block Used as an Input to a CMD Block

Sharing Data and Explaining Unreliable Data

This section describes how data is shared between processes and NCMs in a control strategy. It also discusses unreliable data.

Sharing Data Between Processes

Data can be shared between two or more GPL processes on the same NCM by using the shared variable block (SVAR). The SVAR block receives a value from another block, then on a different diagram, another SVAR block of the same name and type receives the same value. The diagram can be in the same file or in a different file. In other words, shared variable blocks can communicate values between files on the same NCM. All files for the same NCM can share data via shared variables of the same name and type.

Note: You cannot use the same SVAR block in separate NCMs in an attempt to share data between NCMs.

When you connect a data flow line between two blocks in different processes, GPL automatically generates a shared variable. The shared variable is exempted in the source process but not in the destination process. This is to prevent unnecessary triggers in the source process. The shared variable does not appear on the diagram as a SVAR block. If the connection is binary, it generates a binary shared variable. As such, the connection can be exempted from triggers at the destination block, or all triggers in that process may be exempted in the template of the destination process compound.

Refer to the *Function Blocks* chapter for more details on the SVAR block.

Sharing Data Between NCMs

Data can be shared between various NCMs by using the object blocks. An object that is defined in a process for one NCM can be referenced in the process for another NCM. For example, let’s say you have two NCMs: NCM1 and NCM2. An analog input object for outside air temperature called AHU1\OAT1 is defined in the process for NCM1. The value of AHU1\OAT1 can be used by an NCM2 process also, by simply pasting down an AI block and calling it AHU1\OAT1. Both instances of the object AHU1\OAT1 will always have the same value.

Notes: You can use a particular object block only once per strategy file. Every line out of an object block is a read of the object. If you need to read an object attribute’s value more than once, assign its value to a value holder or SVAR block. This latter practice is useful when the object’s value is needed in multiple processes in the same NCM.

A fan-out connection from the object block produces multiple reads (messages) of the object. These reads may involve N1 and/or N2 messages.

Using SVAR or Value Holder Block with fan-out instead of a fan-out from an object block **always** produces a faster executing process.

Explaining Unreliable Data

For some control applications, you may need to determine when data values become unreliable. When a value is unreliable, you can program the process to take certain default actions, such as pass a reliable value instead of normal control actions. That is why GPL keeps track of the reliability of all data. The UNRD block provides a way for testing reliability of data.

Unreliable data is entered into a GPL process when:

- The hardware device that the object block represents goes offline.
- An operation block attempts a divide-by-zero calculation.
- An input to an operation block is out of range.
- A shared variable that has an unreliable value is read.

When a process becomes unreliable, it reports as a status alarm. This status alarm is then sent to the associated system’s report group.

Unreliable data from one function block will be passed on to all downstream blocks. In most cases, if any of the inputs are unreliable, the output will also be unreliable. The specifics of how each block handles unreliable data are discussed in the *Function Blocks* chapter.

When an operation block detects an unreliable value from an object attribute, it uses the last known reliable value and sets it unreliable.

Archive Database Interface

This section describes the archive database as it relates to GPL. The archive database stores all data for process objects, hardware objects, software objects, and feature software. Of these three items, GPL creates the software object databases and the process objects. The word “object” in this section applies to object blocks, such as AIs and BOs and process objects.

Note: This section applies when GPL has the No Archive mode disabled. For details on when it is enabled, refer to the next section, *No Archive Mode*.

Interactions

The archive database is accessed when you are adding new objects and querying, deleting, and modifying defined objects with the GPL Editor. To indicate this, the message `Archive Database Interaction in progress` displays in the middle of the work area. If an error occurs during the interaction, an error message displays on the bottom of the work area. For example, if the object you are modifying does not exist in the archive database, a message to that effect would display. The block is then set to undefined. An object is updated as long as no errors are encountered during the archive database interaction.

The Editor requires that the archive database be on the Operator Workstation’s hard disk so it can read from and write to it. The CAE Tool, Data Definition Language (DDL), and Upload feature use the archive database too, since these tools can also change the object block databases and process objects. Each NCM has its own archive database, and each archive database is stored at an Operator Workstation (Figure 59).

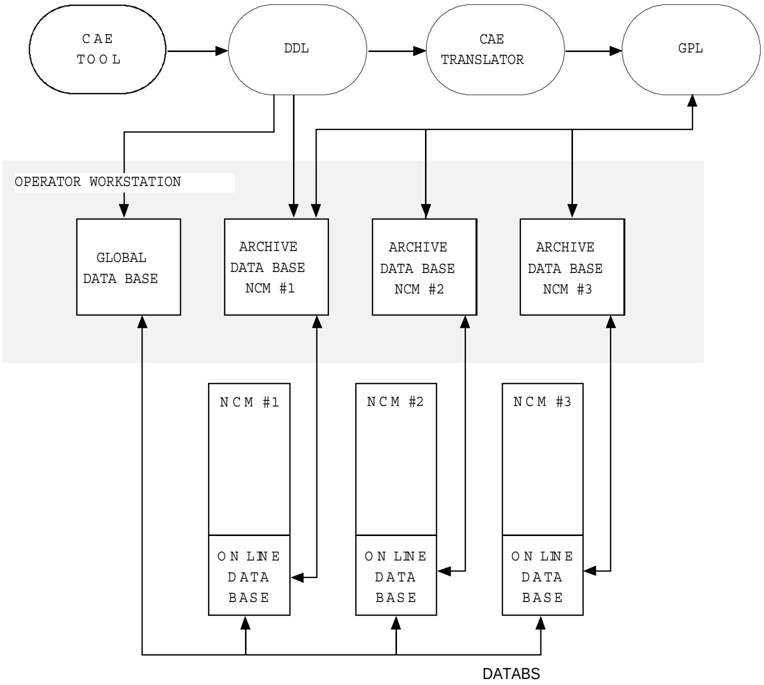


Figure 59: Archive Database Interactions

The strategy file and archive database are updated each time you define, modify, or delete an object. This guarantees that the strategy file and archive database are continually synchronized. Also, both the hardware and the software records of the object stored in the archive database are updated.

Querying a Defined or Undefined Object

When you query a defined object, the Editor interacts with the archive database to read the object's data as stored in the archive. The Editor presents this data in the object's database template. However, when you query an undefined object, the Editor does not interact with the archive database but instead displays the object's data as stored in the strategy file.

Defining a New Object

When you define a new object, the Editor checks the strategy file and the archive database to make sure the object is not already defined. The check occurs when you press F10 to save the object's database template.

If the Editor detects that the object you are trying to define already exists in the strategy file, it does not let you create it and notifies you with an appropriate error message. Only one instance of an object can be used per strategy file.

If the object is not defined in the file but exists in the archive database, the Editor asks whether you want to read the object's current data from the archive database into the strategy file.

If you choose not to do so, you must define a unique name to the object. If you choose to do so, the object template is updated with the data from the archive database.

Then, if this is a process object, the Editor makes two additional checks. First, it checks if object code for the process already exists in the archive database. A user message displays that allows you to specify whether the strategy file version of the process will overwrite the archive database version the next time the process is translated and compiled. Second, the Editor checks if the object code was last built by an application other than strategy (e.g., CAE Translator). Another user message displays that allows you to specify whether the strategy file version of the process will overwrite the archive database version the next time the process is translated and compiled.

After you define an object, some of its template fields become non-modifiable. For example, the software system\object name, the hardware system\object name, and the hardware addresses become unchangeable. If a non-modifiable field must be changed, you need to delete and re-add the object.

Modifying an Object

The GPL Editor also reads the archive database when you modify an existing object. This occurs when you click left on the block with the Query icon highlighted. After you make changes to an object block and press the F10 key, both the strategy file and the archive database are updated (as long as no errors occurred).

However, when you make changes to the template of a defined *process* object, only the strategy file is updated. The archive database is not updated until the process is translated and compiled. If you update the block from the archive database by performing a Session Read or Querying the block, the values in the archive database will overwrite the changes you made to the process template.

Deleting and Erasing an Object

You may delete an object from the archive database and/or erase its block from the strategy file. When you click left on an object block to erase it, a message displays asking you whether you want to delete the object from the archive database. Then, a second message displays, asking you whether you want to erase the block from the strategy file. If you delete an object from the archive, but do not erase the block from the file, the block's border turns to magenta, making it undefined. When you delete an object from the archive database, the available slot numbers for the associated hardware object are updated. If any errors occur during the delete operation, an appropriate message displays.

Optimizing the Archive Database Reads

The archive database is read the first time you query a defined object block to view or modify its template. The template data is then updated in the strategy file. The next time you query the same object, the archive database is not read again, since the data is now the same as in the strategy file. This, in effect, improves the efficiency of the Editor, since it needs to read the archive database only once per object query. This is true until you reload the file, load a different file, or exit GPL.

Using Generic Object Reference Blocks

The archive database is invoked when you attempt to define a generic reference object block (REF) via the F10 key. The Editor lets you define the REF block only if the specified system\object exists in the archive database and is of the correct type. However, by the nature of the REF block, only an association to the actual object in the archive is established; an object named in the REF block is not actually added to or modified in the archive as would occur for an AI block, for example.

Performing a Session Read

The Session Read function updates the data of all object blocks (defined and undefined) in a strategy file with the data for these objects in the current archive database. Session Read accomplishes this in one operation. The alternative would be to query each object block individually.

During a Session Read, when a match is found between a GPL file object block's system\object name and an archive object's system\object name, all data for the object in the archive overwrites data for the object in the GPL file. In other words, the archive is the "master" because its data overwrites the corresponding data in the strategy file.

Perform a Session Read:

- when the archive database has been changed and you want to update the GPL file for simulation
- when you expect to query a number of blocks. If you first perform a Read, the blocks will open immediately (since no archive interaction is required).
- to change the status of undefined object blocks in the GPL file to defined. During a Read, an undefined object block in the GPL file will change to defined if the archive database contains the corresponding object.
- when you load a GPL file (to synchronize the data in the file with the archive)

The Session Read affects object and process object blocks only, not operation or special blocks.

If a block is undefined and the Session Read is successful, the block's status is changed to defined, and the block's template is updated with what is in the archive database.

While it is running, the message `Session Read in progress` displays in the middle of the work area.

When the Session Read is complete, the `Session Read Complete` message tells you:

- how many defined and undefined blocks were read successfully
- how many blocks had errors
- how many defined blocks did not require reading because they had recently been read or queried
- how many undefined blocks could not be read because they lacked a system or object name

All errors and warnings are sent to the list file, which you can display by clicking left on the VIEW option under the Query option menu.

For procedural information on the Session Read function, see *Performing a Session Read,* under *Query Functions,* in the *Editor* section.

Criteria for Editing

When the No Archive mode is disabled, a few prerequisites must be met before you can edit an object in the archive through GPL. They include:

- Operator Workstation archive database must be built and available.
- Global database must have defined all NCMs and all system names, which you plan to use and reference in the strategy file.
- NCM archive database must be available on the Operator Workstation.
- Archive database must have defined all hardware objects that you plan to use and reference in the strategy file. Examples of hardware objects include DCMs and XBNs.
- Strategy file must be loaded under a GPL directory that is under the network directory.

No Archive Mode

The No Archive mode is a limited mode of operation for the GPL Editor. It offers the following advantages:

- The archive database as built by DDL does not have to exist before creating your control strategy. This means that the DDL code and the strategies can be written concurrently, and synchronized later.
- The performance of the GPL Editor increases, since no archive interactions have to take place.
- The GPL Editor can be more easily demonstrated, learned, and used experimentally to try new or modified processes.

The GPL Standard Version is initially configured with No Archive Mode disabled. To enable (or disable) No Archive mode, run the SYSGEN program. (For details, refer to *Introduction*.) The GPL Consultant Version, however, is configured with No Archive Mode permanently enabled. You cannot use the SYSGEN program to disable it.

No Archive mode changes slightly the operation of the Editor, Expert Checker, Translator, and Compiler. (The operation of the Simulator is not affected.) Each change is described in the following paragraphs.

Editor

Network Name Field

The Network Name field, located on the GPL Editor screen in the lower left corner, displays `NO ARCHIVE` instead of the currently selected network.

Querying Objects

When you query a defined object block, the archive database is not accessed to obtain the object’s current definition. Its existing definition, as stored in the GPL control strategy file, is displayed.

Defining and Editing Objects

When you are defining and editing object blocks, the Editor performs no verifies, adds, or deletes to the archive database.

When you press F10 to save changes made to an object block's template, the Editor still checks for required fields, invalid characters and symbols, and duplicate names. However, the Editor does not perform interfield checks (e.g., low limit < high limit), verify that the object exists in the archive, or update the object data in the archive.

When an object block has all required fields entered, it will be shown as defined. Its border turns to brown just as if it were successfully added to the archive database.

Deleting and Erasing Object Blocks

You cannot delete objects from the archive. When you click on a defined object block in the work area (with Eraser icon selected), the block erases immediately from the strategy file. The object delete and block erase verification messages do not display.

Session Read

The Session Read function is not operational. An error message displays if you try to perform a Session Read when the Editor is in No Archive mode.

Expert Checker

The Expert Checker is operational. However, it is not invoked automatically via the Translator, since the Translator is disabled (see next paragraph).

Translator and Compiler

The GPL Translator and Compiler are not operational. If you try to translate and compile, an error message displays to state these utilities are unavailable.

Disabling No Archive Mode

At some point, you will want to disable No Archive mode with SYSGEN in order to add your control strategies to the archive database. After you disable No Archive mode, you must perform a Session Read on each control strategy file that was created or edited when No Archive mode was enabled. This is the most convenient method of synchronizing the strategy file with the archive database. Until you do this, object blocks changed or added while No Archive mode was enabled will appear as defined, even though they are not synchronized with the objects in the archive database. After the Session Read, the blocks for all objects that are defined in the archive remain defined in the strategy file, and their templates are synchronized with the archive database. Those objects that are not defined in the archive are set to undefined (colored magenta) in the strategy file, and an error is written to the list file. For these objects, you will need to open the block templates individually and press F10 to add the objects to the archive database.

Note: If you do not perform a Session Read, the Editor will allow you to expert check, translate, and compile the file. However, the Compiler will detect those objects that are not defined in the archive and write an error to the list file. Note that the Compiler does not change the undefined objects to undefined in the strategy file.

GPL Programmer's Manual

Editor

* Indicates those sections where changes have occurred since the last printing.

* Indicates those sections where changes have occurred since the last printing.

* Indicates those sections where changes have occurred since the last printing.

* Indicates those sections where changes have occurred since the last printing.

Overview

This chapter explains how to use the GPL Editor. It contains step-by-step instructions for all functions the Editor provides. It also has a tutorial that quickly teaches you the basics of GPL editing. The last section of this chapter explains all error messages that the Editor may output.

The GPL Editor:

- Creates and edits control strategies for the NCM. The strategies define the logic that controls the equipment connected to the Metasys Network.
- Creates and edits software objects and compounds.
- Creates and edits process objects.
- Prints control strategy diagrams and database templates.
- Runs the Expert Checker, Simulator, and Translator.

Starting the Editor

To start GPL, type `GPL` at the DOS prompt and press Enter. The GPL program will execute, and the Editor screen will display. To start GPL from the FMS, select GPL from the Exit menu at the Operator Workstation. The computer will exit Windows to DOS, and then execute GPL. The Editor screen will display.

Details

When GPL is started, a blank work area or a control strategy is displayed. A blank area is displayed if the GPLPATH environmental variable in the GPL.BAT file is set to a network directory. A strategy is displayed if the GPLPATH environmental variable is set to file name. If you wish, you may change the GPLPATH statement to a different file or directory. Follow DOS standards for changing a batch file.

Mouse and Keyboard

You use the **mouse** extensively with the GPL Editor for moving the cursor, selecting icons, and drawing items on the screen. Two buttons on the mouse are used; which one you use depends on the function you are performing.

You can use either a two or three button mouse with GPL. With a two button mouse, you use both the left and right buttons. With a three button mouse, you use only the left and middle buttons.

Note: In all GPL operations, pressing the right button on a two button mouse is equivalent to pressing the middle button of a three button mouse.

The instructions in this chapter presume you are familiar with using a mouse. The term “click” means to move the cursor to an icon or option on the screen, and press the mouse button. The following shorthand descriptions for clicking are used:

Description	Meaning
Click left	Press the left mouse button *once*.
Click right/middle	Press the right or middle mouse button *once*.
Double-click left	Quickly press the left mouse button *twice*.
Select	Position the cursor on an icon, option, or field and click a mouse button.
Drag	Hold down a mouse button, move the mouse and release the button.

You use the **keyboard** for entering data, paging screens, displaying help screens, and saving changes made to a template. The GPL Editor has the following special keys:

Key	Function
Alphanumeric keys	Enter information such as names, values, and descriptions.
Arrow keys	Move cursor between parameter fields.
Backspace	Moves cursor back one space within a parameter field.
CTRL/Arrow	Moves cursor within a parameter field.
Esc	Ignores changes made to a template and closes template. Exits a help screen. Clears most items in the work area, such as some option menus and description files.
F1	Displays a help screen.
F2	Changes the function block size to the large default size.
F3	Changes the function block size to the small default size.
F4	Changes the function block size to the standard default size.
F10	Saves changes made to a template and closes template.
Page Up	Displays the next page of a template, help screen, or file.
Page Down	Displays the previous page of a template, help screen, or file.
Insert	Shifts characters one space to the right for you to add a character.
Delete	Deletes a character in a parameter field.

Note: If your PC is configured to run 386MAX.SYS (in CONFIG.SYS file), do not quickly press the mouse button and a keyboard key simultaneously when typing text in the work area; the computer may hang. If this occurs, reboot the computer (CTRL-ALT-DEL) and re-enter all updates made since you last saved the GPL file.

Editor Screen

The **Editor Screen** is displayed each time you start GPL from the FMS or DOS. You create, edit, and delete control strategies with it. In addition, you start the Expert Checker, Simulator, and Translator from the Editor screen.

As Figure 1 shows, the GPL Editor screen is divided into several, functionally different areas.

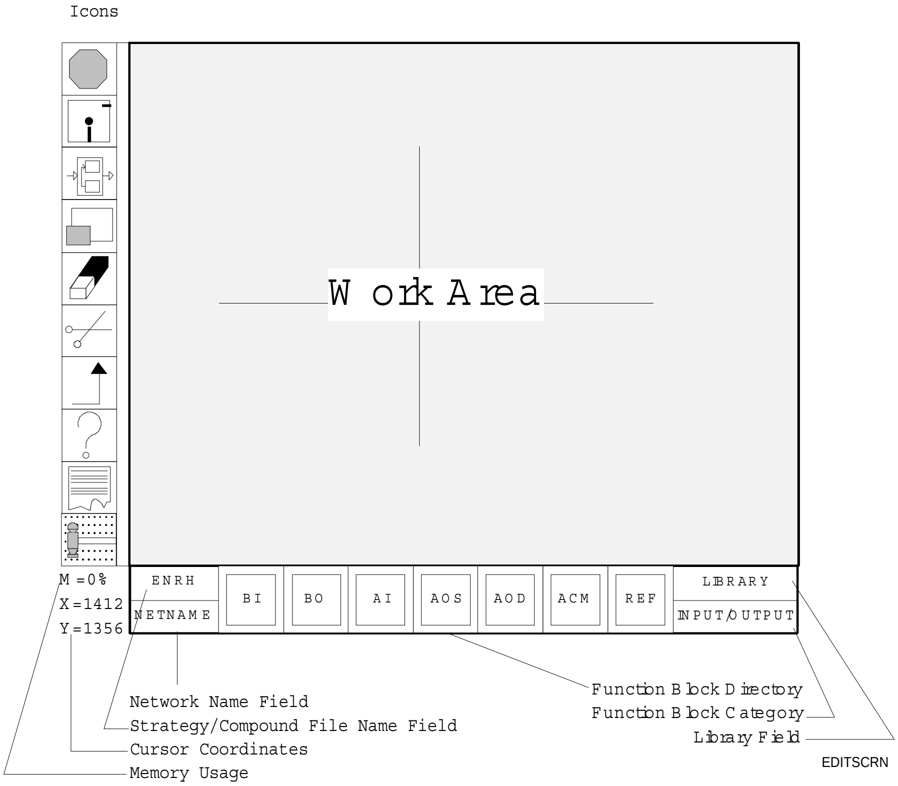


Figure 1: GPL Editor Screen

Icons

Icons are graphic symbols (Figure 1) that represent some function, such as enabling the erase function or displaying an option menu. You enable the function by positioning the cursor on the icon and clicking or double-clicking the left mouse button.

Network Name Field

The **Network Name** field shows the directory name of the network currently selected (Figure 1). This is the network that the Editor uses for all archive database interactions. The field may also display:

`?? ARCH DB`	Network directory is not selected.
`COMPOUND`	Compound make or edit mode is in use.
`NO ARCHIVE`	Archive database interaction is disabled.

Strategy/ Compound File Name Field

The **Strategy/Compound File Name** field indicates the name of the control strategy or compound that is currently loaded (Figure 1).

Cursor Coordinates

The **cursor coordinates** are the x- and y-coordinates of the cursor's position on the screen (Figure 1). They help you to position blocks, lines, and text. The range is 0000 to 1000, with these end values per cursor position:

Cursor Position	End Values
Upper left corner	x=0, y=1000
Upper right corner	x=1000, y=1000
Lower left corner	x=0, y=0
Lower right corner	x=1000, y=0

Memory Usage

The **memory usage** figure (M = Memory) is a percentage of how much computer RAM the current control strategy is consuming (Figure 1). The number increases and decreases as you edit the strategy.

A figure of 100% indicates that one of the three files of the control strategy is full. GPL does not indicate which file is full. When a file is approaching 100%, you might want to add a FILE block to the control strategy to divide the large strategy into one or more smaller strategies.

Function Block Directory

The **Function Block Directory** is the row of function blocks that line the bottom of the screen (Figure 1). Clicking left on one of these blocks selects it to be pasted down in the work area.

Function Block Category

The **Function Block Category** field shows the name of the currently selected function block category (Figure 1). A block category is a grouping of GPL blocks that have similar functions. GPL has over 60 blocks organized under 16 different block categories. The blocks display on the bottom of the screen when you select a different block category.

Library Field

The **Library** field (Figure 1) changes the function blocks that display on the bottom of the screen. Clicking left on this field displays the function block categories down the right side of the work area (Figure 8). Selecting one of these categories changes the function blocks directory across the bottom of the screen.

Work Area

The **Work Area** is the entire middle portion of the Editor screen (Figure 1). In this area, you create and edit control strategies and edit the function block templates. Selection menus, help screens, user messages, the function block library, FILE block icon, the cursors, and the white enclosing box are also displayed here.

Selection Menus

Selection menus offer additional functions that are related to an icon or function. There are four types of selection menus: option menus, submenus, connection menus, and find menus.

An **option menu** appears in the upper portion of the work area, just to the right of the Exit icon (stop sign). Figure 2 shows the option menu for the File icon (disk). Clicking left on an icon (or double-clicking left in some cases) displays the menu. You can clear an option menu from the screen by clicking left in any open spot in the work area.

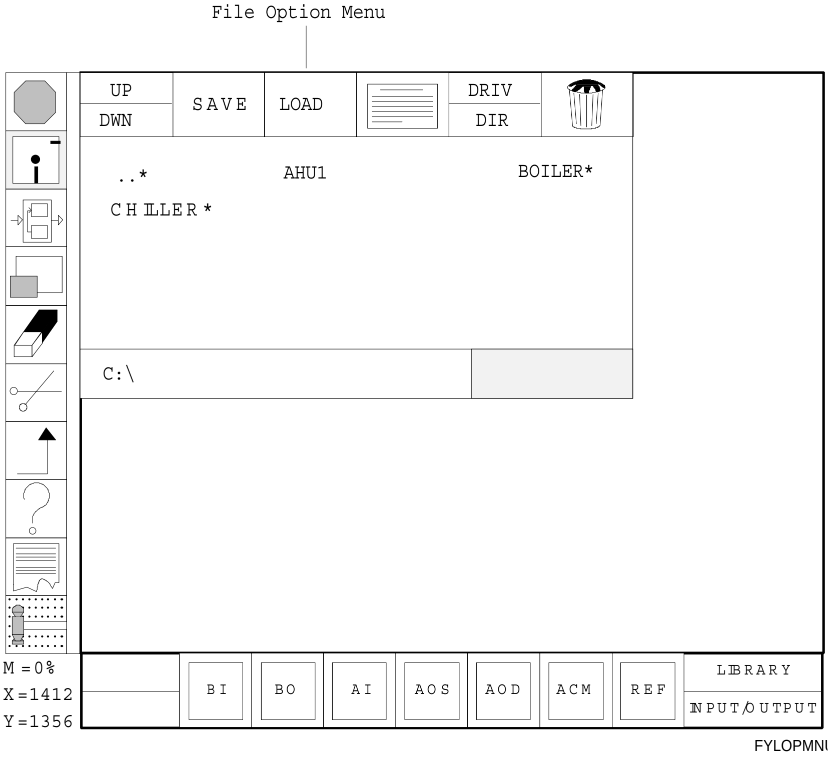


Figure 2: Option Menu for File Icon

A **submenu** appears in the same place as an option menu—in the upper portion of the work area. A submenu is a second menu that appears after you make a selection on an option menu. The GPL Editor has only two submenus: Print submenu and Translator submenu. Both are pictured in Figure 3. Clicking left inside the checkbox selects the function. You clear a submenu from the screen by clicking left in any open spot of the work area. This cancels the operation.

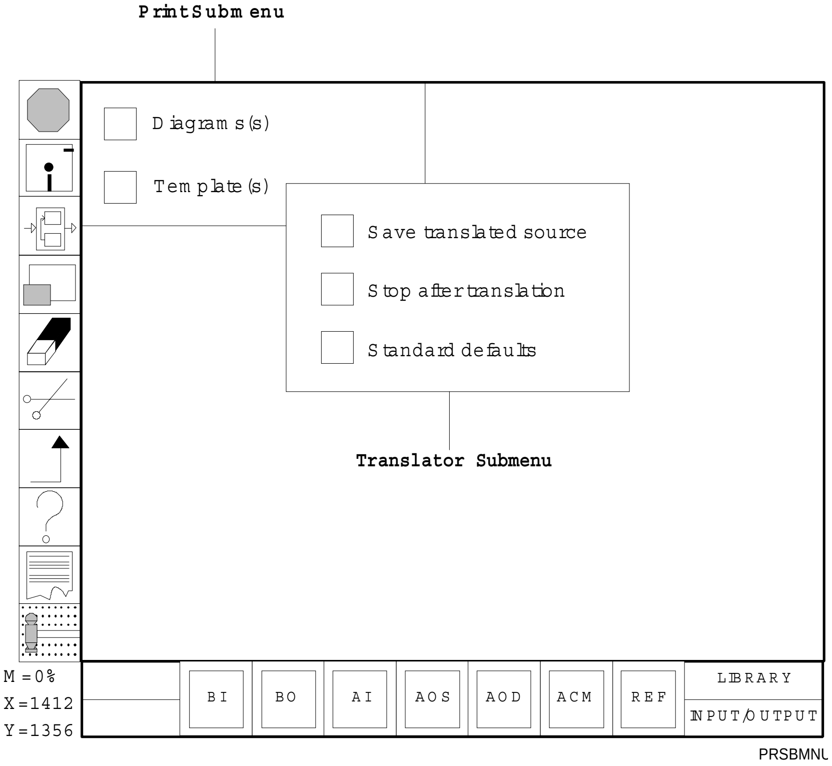
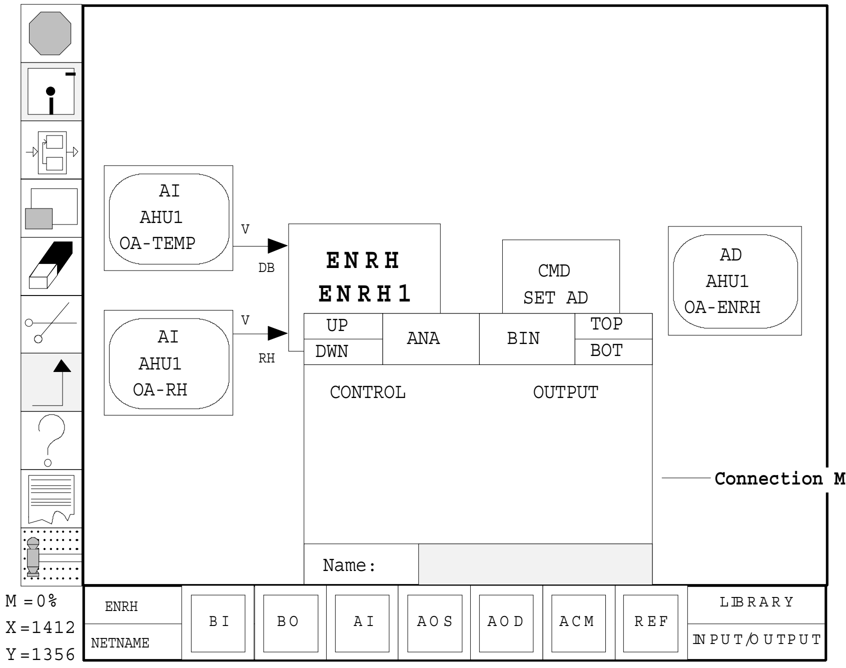


Figure 3: Print and Translator Submenus

A **connection menu** appears while you are connecting two blocks. If you are connecting two blocks remotely, or assigning an analog display to a block's output, a remote or analog display connection menu is used. The connection displays over the diagram as shown in Figure 4. Clicking left on a connection name selects it. You can clear a connection menu from the screen by clicking left in any open spot of the work area.



Connection Menu

CONNMNU

Figure 4: Connection Menu

A **find menu** appears when you click left on the Find option. It displays over the diagram like a connection menu (Figure 5). Clicking left on a system\object name or block name selects it. You can clear a connection menu from the screen by clicking left.

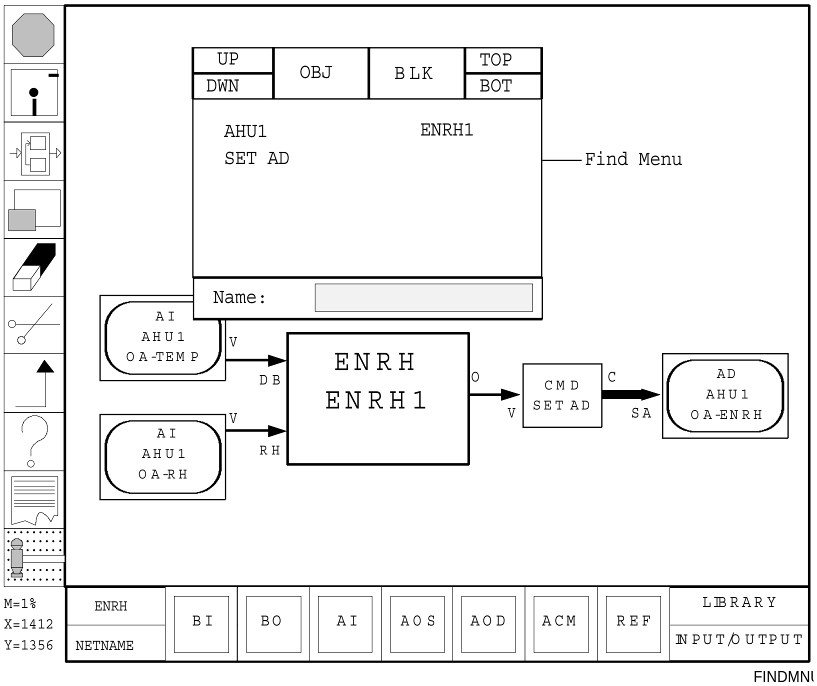
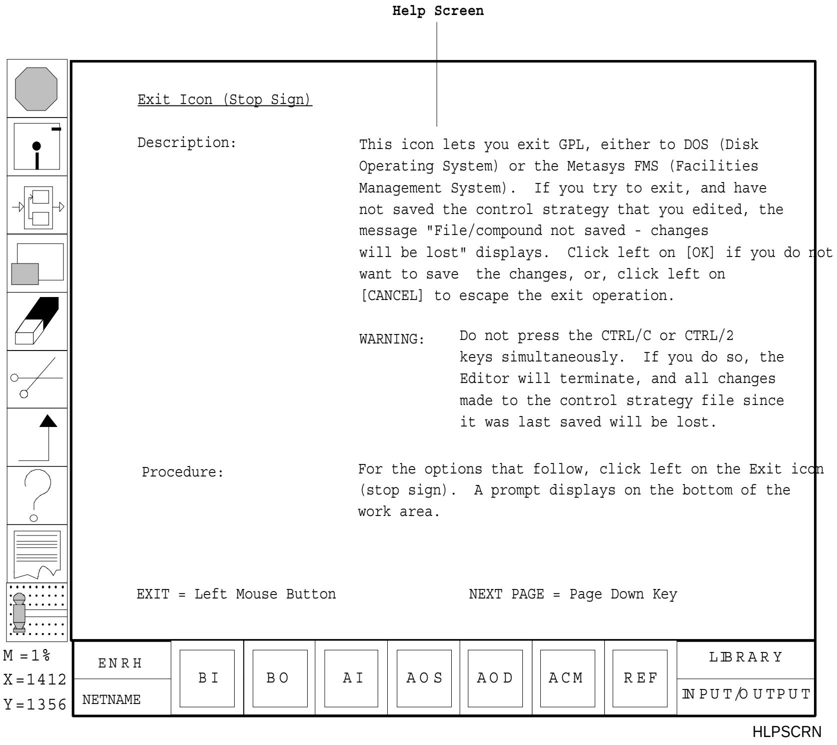


Figure 5: Find Menu

Help Screens

Help screens provide brief instructions and descriptions on any item on the screen, such as an icon or function block. Figure 6 shows the help screen for the Exit icon. To view a help screen, you position the cursor over the area in question and press F1. Help screens are context sensitive, which means the information that appears pertains to the function that is available at the current position of the cursor. Pressing Esc or clicking left returns the screen to its previous condition.



Exit Icon (Stop Sign)

Description:

This icon lets you exit GPL, either to DOS (Disk Operating System) or the Metasys FMS (Facilities Management System). If you try to exit, and have not saved the control strategy that you edited, the message "File/compound not saved - changes will be lost" displays. Click left on [OK] if you do not want to save the changes, or, click left on [CANCEL] to escape the exit operation.

WARNING: Do not press the CTRL/C or CTRL/2 keys simultaneously. If you do so, the Editor will terminate, and all changes made to the control strategy file since it was last saved will be lost.

Procedure:

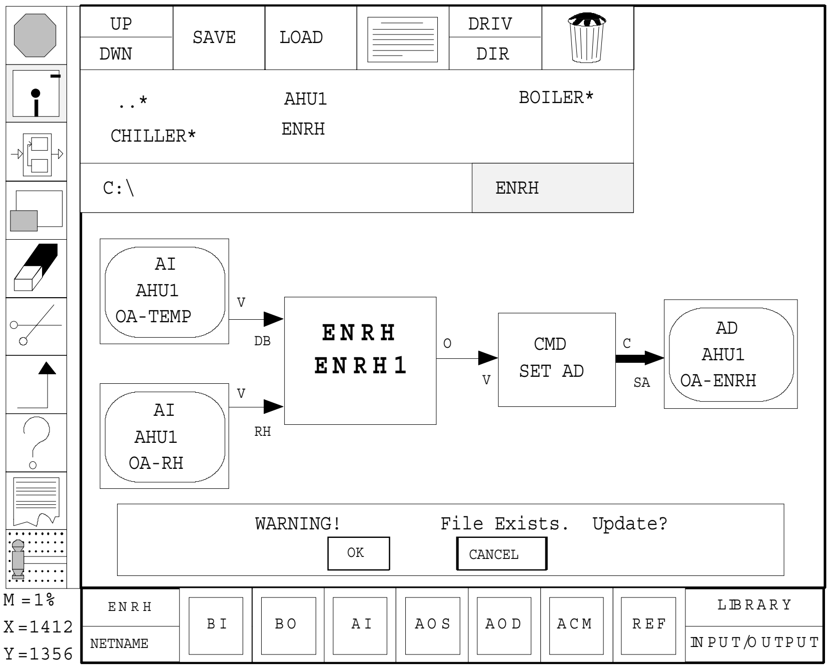
For the options that follow, click left on the Exit icon (stop sign). A prompt displays on the bottom of the work area.

EXIT = Left Mouse Button NEXT PAGE = Page Down Key

Figure 6: Help Screen for Stop Sign Icon

User Messages

User messages are statements that ask you to perform an action, or notify you of an error. Many include [OK] and [CANCEL] fields that you click left on to perform or ignore a function, respectively. User messages display on the lower portion of the work area. Figure 7 shows the user message that displays when you try to update a control strategy. Note that when the Editor displays a user message, it positions the cursor over the most probable choice.



WARNING! File Exists. Update?

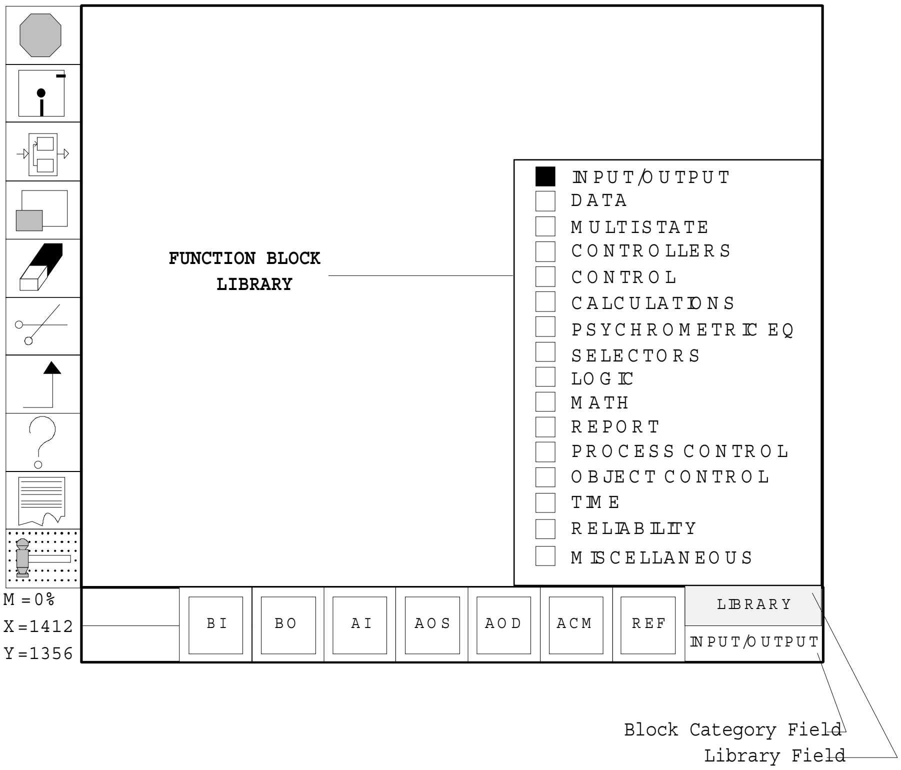
M =1%
X =1412
Y =1356

USRMSSG

Figure 7: User Message for File Update

Function Block Library

The **function block library** shows the 16 function block categories (Figure 8). You need to display the library each time you want to change the function block category. Clicking left in the red checkbox beside a category changes the block directory to make available the blocks under that category.



☒ INPUT/OUTPUT
□ DATA
□ MULTISTATE
□ CONTROLLERS
□ CONTROL
□ CALCULATIONS
□ PSYCHROMETRIC EQ
□ SELECTORS
□ LOGIC
□ MATH
□ REPORT
□ PROCESS CONTROL
□ OBJECT CONTROL
□ TIME
□ RELIABILITY
□ MISCELLANEOUS

FNCBLLIB

Figure 8: Function Block Library

FILE Block Icon

The **FILE block icon** returns you to a previous file when you click left on it (Figure 9). It displays only when the work area is showing the contents of a FILE block.

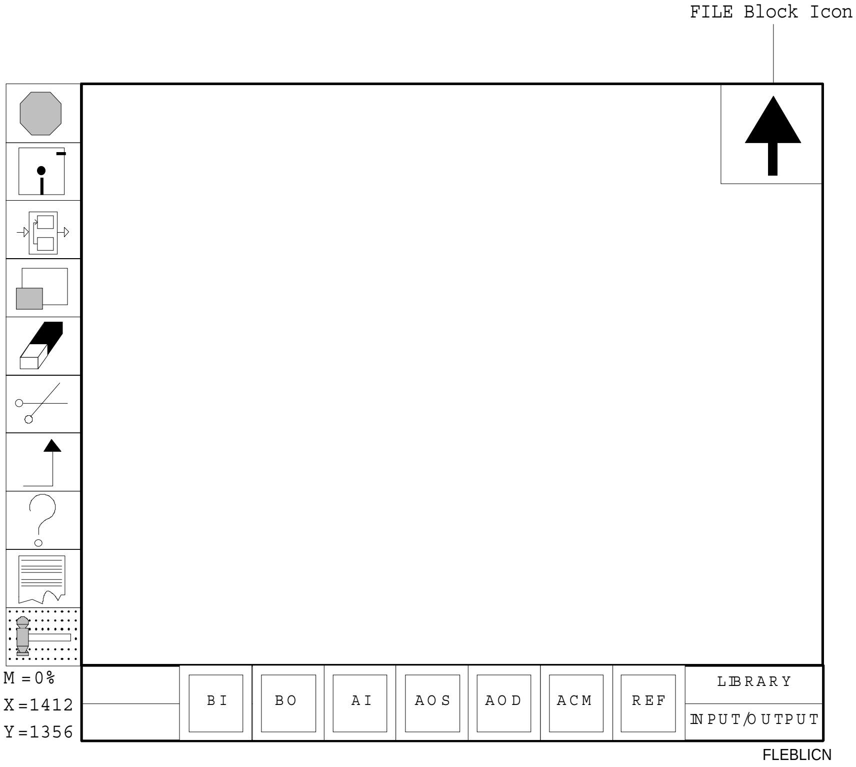


Figure 9: FILE Block Icon

Crossmark and Underline Cursors

The **crossmark cursor** and **underline cursor** are movable markers that indicate a position on the screen (Figure 10). The crossmark cursor is shown when you are selecting blocks and icons. The underline cursor is shown when you are typing text.

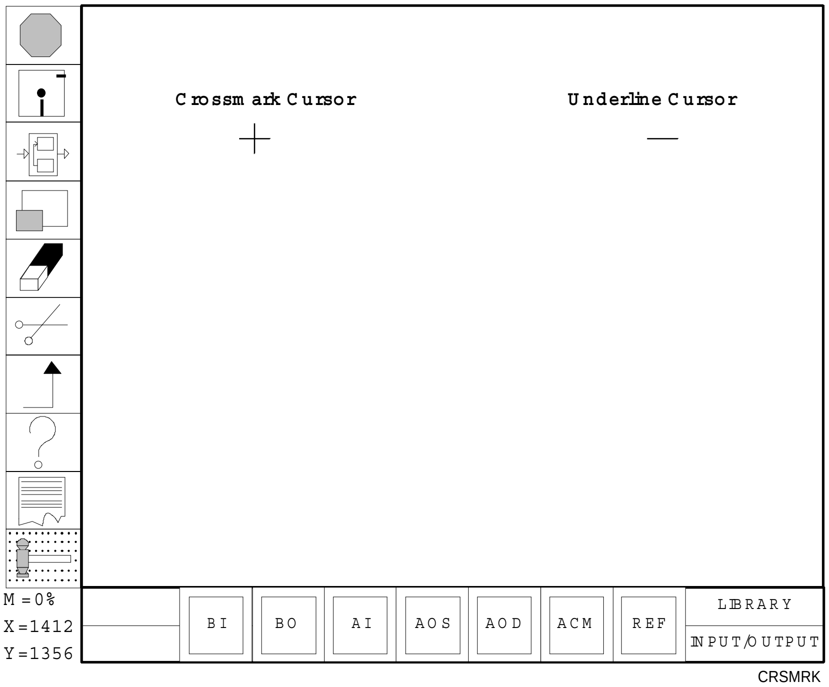


Figure 10: Crossmark and Underline Cursors

Note: You may notice that the crossmark cursor will “jump” to a different area of the screen when you click or double-click on an icon or option. This helps you select the next function more quickly.

White Enclosing Box

You can use the **white enclosing box** when making compounds and selecting a part of the diagram to zoom, move, or copy (Figure 11). The enclosing box comes to the screen automatically when you select the operation. You can resize the enclosing box by dragging the right/middle mouse button.

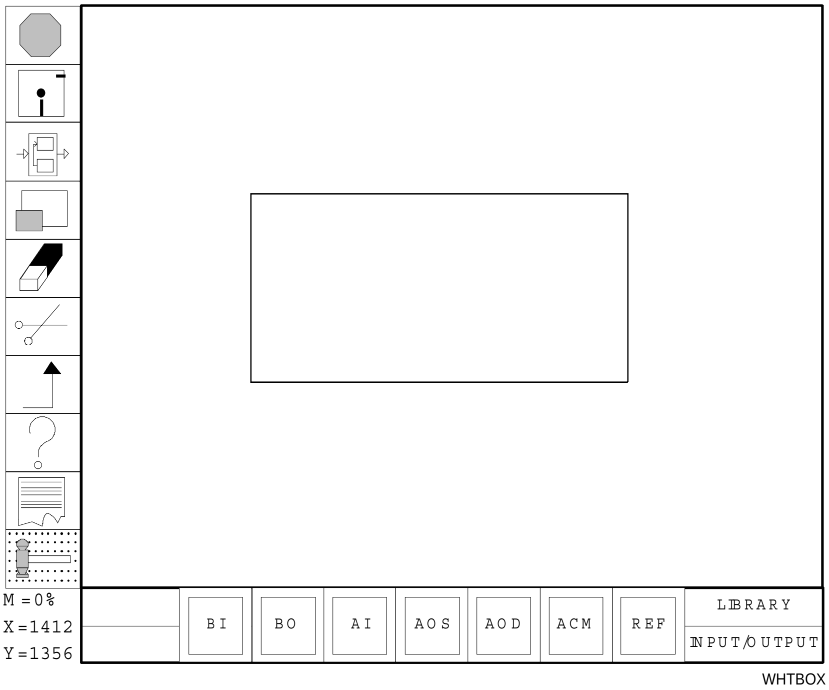


Figure 11: White Enclosing Box

Prerequisites

Before you can create diagrams with the GPL Editor, you must meet the following prerequisites (applies only when archive is enabled):

- The Operator Workstation archive database must be built and available.
- The global database must have defined all NCMs and all system names that you plan to use and reference in the strategy file.
- The NCM archive database must be available on the Operator Workstation.
- The archive database must have defined all hardware objects that you plan to use and reference in the strategy file.
- A strategy file must be loaded under a GPL directory that is under the network directory.

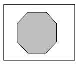
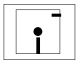
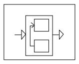
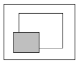
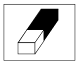
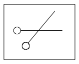
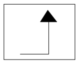
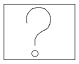
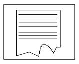
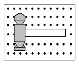
Icons

Icons are graphic symbols that represent some function, such as enabling the erase function or displaying an option menu. You enable or perform the function by positioning the cursor on the icon and clicking or double-clicking the left mouse button. When you select an icon, it back-highlights in red.

Clicking left on an icon either displays an option menu or activates the default option for that icon. The default option is the function that can be performed without choosing it on an option menu. The default option is also included on the option menus, and it is always the left-most choice on the menu. The instructions in this chapter indicate the default functions for each icon.

Some option menus are displayed by double-clicking left on the icon. The instructions in this manual indicate when a single or double-click is needed.

Figure 12 shows a summary of all the main icons.

	Name	**Click Left**	**Double-Click Left**
	Exit (stop sign)	Exits GPL.	Same.
	File (disk)	Displays option menu.	Same.
	Compound	Displays option menu.	Same.
	Zoom & Pan	Enables zoom function.	Displays option menu.
	Erase (eraser)	Enables eraser function.	Displays option menu.
	Move (scissors)	Enables move function	Displays option menu.
	Connection (arrow)	Enables connection function.	Displays option menu.
	Query (question mark)	Enables query function.	Displays option menu.
	Print (page)	Displays option menu.	Same.
	Tools (hammer)	Turns Grid on/off.	Displays option menu.

SUMMI

Figure 12: Summary of Main Icons

Tutorial

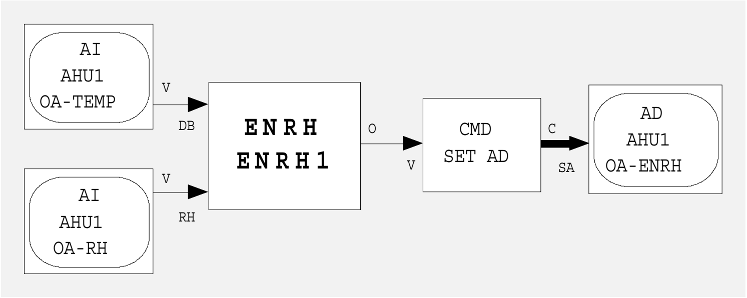
This section is a tutorial of the GPL Editor. It describes the basic steps of drawing, saving, and printing a diagram. The tutorial introduces you to the most-used functions only. You can find the instructions for all Editor functions in later sections of this chapter.

By completing the exercises in the tutorial, you will learn how to:

- Name the diagram.
- Paste down function blocks.
- Define function blocks.
- Connect function blocks.
- Compound a diagram.
- Save the compound into a file.
- Erase a function block and undo the erased block.
- Erase a connection and undo the erased connection.
- Print a diagram.
- Exit GPL.

Note: The exercises presume that the Editor is set to interact with the archive database. If No Archive Mode is enabled, no checks against the archive database will be made, and the object blocks will paste down as defined.

Figure 13 shows the diagram that you will draw.



DGRMEDTF

Figure 13: Diagram You Will Draw

In this GPL diagram, outdoor air enthalpy is calculated using the values of outdoor air dry bulb temperature and outdoor air relative humidity. The result is sent to an analog data object. The analog data object can be viewed on an object summary at the Operator Workstation.

Before you can begin the Tutorial, you need to perform these steps using DDL:

- Compile the Operator Workstation archive database file.
- Define these two items in the global database: NCM name of NCM-FLR1 and a hardware system name of AHU1.
- Define this hardware system\object name in the archive database: AHU1\DCM1 as a DCM hardware type device.

For instructions on how to perform these steps, see the *DDL Programmer's Manual (FAN 630)*.

Now you are ready to start the Tutorial. If you have not already started the GPL Editor, do so now (refer to *Starting the Editor*). When the Editor screen displays, it may show a diagram. To clear it, double-click left on the Erase icon, click left on CLR ALL, and answer OK to the warning message.

Name the Diagram

In this first exercise, you will assign a name to the diagram you will draw. This will also select the network, which is a required step when creating control strategies.

1. Click left on the File icon (disk).
2. Go to the GPL directory under the desired network directory by clicking left on the directory names.
3. Type in `ENRH` in the File Name field.
4. Position the cursor on the SAVE option and click left. The empty ENRH file saves to disk, the Network Name field changes to show the selected network, and the File option menu clears.

   In the next exercise, you will paste down several function blocks.

Paste Down Function Blocks

In the first exercise, you will paste down five function blocks into the work area: two Analog Input (AI) blocks, one Enthalpy-Relative Humidity (ENRH) block, one Command (CMD) block, and one Analog Data (AD) block.

Paste Down AI Blocks

1. Look at the function block category field in the lower right corner of the screen (Figure 1). If INPUT/OUTPUT is displayed, skip to Step 4; otherwise, continue with Step 2.
2. Click left on the LIBRARY field in the lower right corner. A list of all function block categories displays.
3. You will first paste down two analog input blocks, which belong to the INPUT/OUTPUT category. Position the cursor inside the red checkbox next to INPUT/OUTPUT and click left. The function blocks displayed on the bottom change to those that belong to this category.
4. From the function blocks on the bottom of the screen, locate the one labeled AI (analog input). Position the cursor over that block and click left. The Function Block library clears (if it was displayed), and the background of the AI block highlights in red to indicate you have selected it.

5. Move the mouse up. Notice a magenta-colored block appears. This is the AI block.
6. Position the block in the upper left side of the work area. Refer to Figure 13 for proper placement. Click left. The block pastes down and is labeled AI.
7. Move the mouse. Notice a second block forms. This is another AI block ready to be pasted down. Paste it down under the first AI block. Again, refer to Figure 13 for its proper placement.

Paste Down ENRH Block

1. The ENRH block resides under the PSYCHROMETRIC EQ (psychrometric equations) category. Display the function block categories by clicking left on the LIBRARY field. Click left in the checkbox next to the PSYCHROMETRIC EQ category. The blocks on the bottom of the screen change.
2. Locate the ENRH block in the row of function blocks. Click left on it to select it.
3. Paste down one ENRH block about one inch to the right of the AI blocks (see Figure 13).

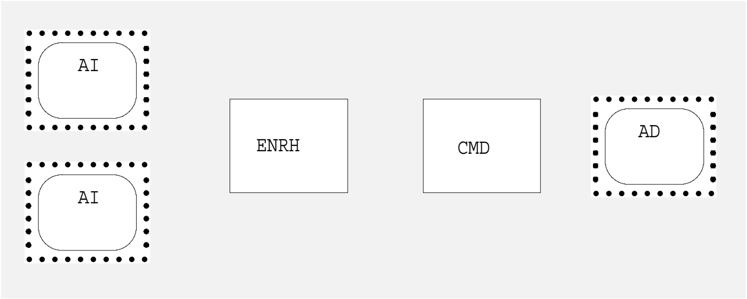
Paste Down CMD Block

1. The CMD block resides under the OBJECT CONTROL category. Change the function block category to OBJECT CONTROL.
2. Locate the CMD block in the row of function blocks. Click left on it to select it.
3. Paste down one CMD block about one-half inch to the right of the ENRH block.

Paste Down AD Block

1. The AD block resides under the DATA category. Select the DATA category to display the AD block on the bottom of the screen.
2. Click left on the AD block and paste it down about one-half inch to the right of the CMD block.

Your diagram should now look something like Figure 14:



PASTEDWN

Figure 14: Five Function Blocks Pasted Down

The next exercise will teach you how to define the function blocks you pasted down.

Define Function Blocks

Each function block has a database template that you fill in that defines its name, parameters, and attributes (i.e., characteristics). This template drops down into the work area. In these steps, you will fill in the templates for four of the blocks that you pasted down in the previous step. You will not define the CMD block, since the Editor defines it for you when you connect it to an object block.

Define Upper AI Block

1. Click left on the Query icon (question mark).
2. Position the cursor inside the upper AI block. Click left.

   The template for the AI block appears in the work area. Notice that two columns of parameters are shown. For this exercise, you will only need to define a few of them, and accept the default values for all others.
3. Notice that the first parameter, System Name, is highlighted in white. When a parameter is highlighted like this, it is selected for entry. Type `AHU1` for the System Name. Use the backspace key if you make a typing mistake. Press Enter.
4. The next field, Object Name, now highlights. Type `OA-TEMP` in the Object Name field. Press Enter.
5. Move to the third parameter, Expanded ID. Type `OUTDOOR AIR TEMPERATURE`. Press Enter.
6. The cursor is now in the first parameter of the HARDWARE category. This data links the hardware device with the software object. Type `AHU1` for System Name and `DCM1` for Object Name.
7. You are now done with this template. Press F10 to save the entries. An archive database interaction message displays, indicating that the Editor is trying to add the new object to the archive.

Note: If an error message displays at this point, refer to the *User Messages* section of this chapter. An error should not occur if the archive database exists and contains the hardware object DCM1.

Notice that the AI block has changed from magenta to brown to indicate it is now defined. Also notice the block is labeled with the System Name and Object Name that you entered.

Define Lower AI Block

1. Define the other AI block in the same manner as you did the first AI block. Refer to the previous steps if you need to. Use the up arrow and down arrow keys to move between the fields as required. Under the IDENTIFICATION column, type these values:

   System Name–`AHU1`
   Object Name–`OA-RH`
   Expanded ID–`OUTDOOR AIR R.H.`

   Under the HARDWARE column, type:

   System Name–`AHU1`
   Object Name–`DCM`
   Slot Number–`2`

   At this point, you would ordinarily change the appropriate attributes in the template to configure the object to represent an outdoor air relative humidity sensor. However, for the purpose of the tutorial, you can skip this step.

2. Press F10 to save. The archive database is again accessed.

Define ENRH Block

1. Define the ENRH block. It is actually defined (block's border is brown), but you should still assign it a unique name. Type a Block Name of `ENRH1`.

2. Press F10 to save. Note that the archive database is not accessed since this is an operation block.

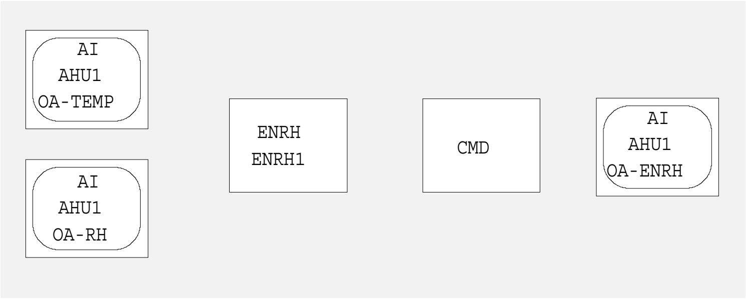
Define AD Block

1. Define the AD block. Under the IDENTIFICATION column, type these values:

   System Name–`AHU1`
   Object Name–`OA-ENRH`
   Expanded ID–`OUTDOOR AIR ENTHALPY`

2. Press F10 to save.

The blocks are now defined and the diagram should look like Figure 15:



FORBLKS

Figure 15: Four Function Blocks Defined

The next exercise will teach you how to connect the function blocks you have defined.

Connect Function Blocks

Connecting blocks establishes the data and control flow between two blocks. The diagram so far has all blocks except the CMD block defined. The connection process will define the CMD block. In these steps, you will connect the two AI blocks to the ENRH block. You will then connect the CMD block to the AD block, and the ENRH block to the CMD block.

Connect AHU1\OA-TEMP Block to ENRH Block

1. Click left on the Connection icon (bent-up arrow).
2. Click left inside the AI block labeled AHU1\OA-TEMP. A connection menu displays. This is called an output connection menu, since it shows all available outputs for this block. The connection names are color-coded to indicate different types.
3. Locate VALUE in this list (colored yellow) and click left on it. The menu clears.
4. Move the mouse. Notice a line has formed.
5. Move the line into the ENRH block and click left. Another connection menu displays, this one is called an input connection menu, which shows all available inputs into the ENRH block.

6. Click left on DRY BULB in the menu. The menu clears and a line connects the two blocks. Notice the line labels at each end of the line: “V” for Value and “DB” for Dry Bulb temperature. This means that the Value attribute of the AI object is used for the Dry Bulb value in the ENRH calculation.

Connect AHU1\OA-RH Block to ENRH Block

1. Click left inside the AI block labeled AHU1\OA-RH. The output connection menu displays.
2. Click left on VALUE in this list.
3. Draw the line into the ENRH block and click left. The input connection menu for the ENRH block displays.
4. Click left on REL HUMID (Relative Humidity) in the menu. The menu erases and a line connects the two blocks. The line labels are “V” for Value at the AI block and “RH” (Relative Humidity) at the ENRH block. The Value attribute of the AI object is used for the Relative Humidity value in the ENRH calculation.

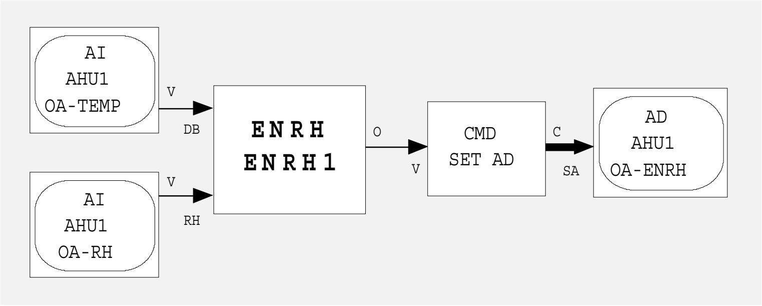
Connect CMD Block to AHU1\OA-ENRH Block

1. Click left on the CMD block. An output connection menu displays.
2. Click left on COMMAND in this list. The menu clears.
3. Move the line into the AD block labeled AHU1\OA-ENRH and click left. The input connection menu for this block displays, which lists all available commands to the AD object.
4. Click left on SET AD in this list. The line connects the two blocks. The line labels are “C” (Command) at the CMD block and “SA” (SET AD) at the AHU1\OA-ENRH block. The CMD block is sending a SET AD command to the AHU1\OA-ENRH object that stores the enthalpy value calculated by the ENRH block.

Connect ENRH Block to CMD Block

1. Click left on the ENRH block. The output connection menu displays.
2. Click left on OUTPUT in this list. The menu clears.
3. Move the line into the CMD block and click left. The input connection menu displays.
4. Click left on VALUE in the menu. The menu erases and a line now connects the two blocks. The line labels are “O” (Output) at the ENRH block and “V” (Value) at the CMD block. The result of the enthalpy calculation, Output, is read by the CMD block, which in turn commands the AHU1\OA-ENRH object to that value.

Your diagram should now resemble Figure 16.



CONDGRN

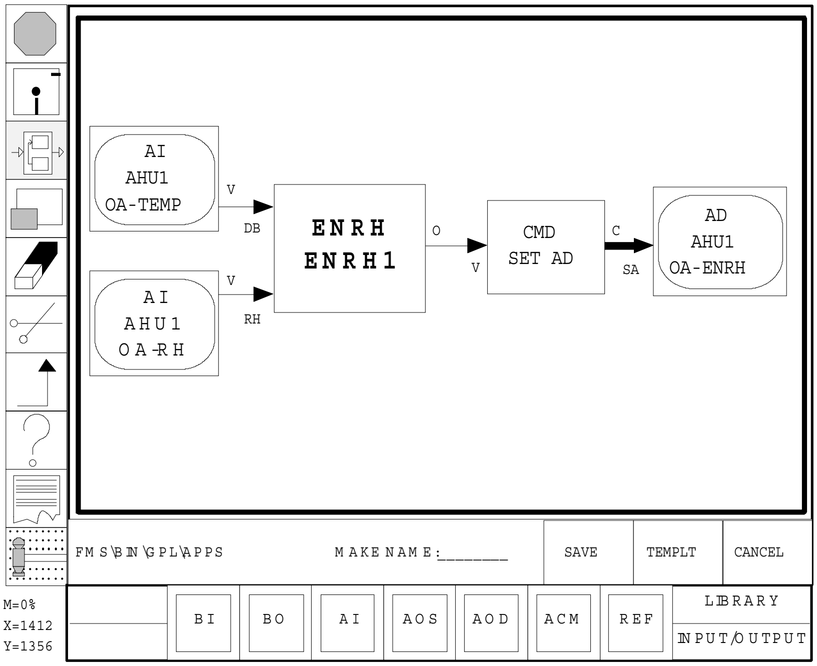
Figure 16: Connections on the Diagram

You have now finished drawing the diagram. The next exercise will teach you how to group the diagram into a process compound.

Compound the Diagram

Compounding is a method of grouping a diagram into a block to provide a high-level function, such as optimal start or damper control. The diagram that you created in the previous exercises is not complete until you group it into a process compound. That is because the diagram contains two operation blocks (ENRH and CMD), and all operation blocks must be placed in a process compound so that they can be translated and compiled into object code. Follow these steps.

1. Click left on the Compound icon. Its option menu displays.
2. Click left on the MAKE option.
3. Move the mouse and notice that a white enclosing box appears. This box is used to enclose the area of the diagram that you want to compound. For this example, you need to compound the entire diagram.
4. Enlarge the enclosing box by holding down the right button and dragging the mouse. Enlarge the box so that it encloses the entire diagram. It is best to size the box so that it is slightly larger than all items to be enclosed.
5. Click left. The Compound Make screen displays (Figure 17).



MAKESCRN

Figure 17: Compound Make Screen

6. Click left on Templt (Template). The database template for the compound appears.
7. The cursor is in the Compound Type field. Press the Tab key once to display `PROCESS` as the compound type. The template changes to show additional fields.
8. Press Enter. Type in `AHU1` for a System Name. Press Enter.
9. Type in `ENTHALPY` for an Object Name. Press Enter.
10. Use the down arrow key to move to the Period field. Type `00:05:00`. Press Enter.
11. Press F10 to save the data you entered. The compound database template clears.
12. Type `OA#ENRH` in the NAME field. This is the name under which the compound will be saved on disk.

13. Click left on the SAVE option. A message displays, giving you the option of saving the compound to the screen or to the disk. Click left on Screen. The blocks on the screen are replaced with a compound block labeled OA-ENRH. The diagram you created is actually “behind” this block. (You can view it by double-clicking left on the block with the Query icon highlighted.)
14. To save this new process compound to the archive database, click left on the compound block with the Query mode selected. Press F10 to save the template information. An archive database interaction message displays. Once the process is added to the archive, the template clears and the block turns brown with solid borders.

The next exercise will teach you how to save the compound into a file.

Save the Compound Into a File

All compounds must be placed into strategy files for them to be used with the other GPL utilities, including the Expert Checker, Simulator, and Translator. In this step, you will save the compound you made in the previous exercise to a file you created earlier called ENRH.

1. Click left on the File icon. An option menu displays. The name ENRH is already in the File Name field, so you do not have to type it in.
2. Position the cursor on the SAVE option and click left. The ENRH file saves to disk and the File option menu clears.

The next exercise will teach you how to erase a function block and undo the erased block.

Erase a Function Block and Undo the Erased Block

Erasing parts of a diagram and undoing (i.e., unerasing) erased items are easy. In this exercise, you will erase the function block labeled ENRH1, then undo the erased block.

1. To redisplay the diagram, double-click left on the compound block with the Query icon selected.
2. Click left on the Erase icon.
3. Move the cursor anywhere inside the block labeled ENRH1 and click left. The block and its connections erase.
4. To undo the erased block, double-click left on the Erase icon. An option menu displays.
5. Click left on the UNDO option. The block and its connections return to the screen. An Undo message displays below that indicates the operation has been performed. Click left to clear the message.

Erase a Connection and Undo the Erased Connection

Erasing a connection is slightly different from erasing a function block. You can place the cursor anywhere inside a function block to erase it. For a line, however, you must place the cursor at either end of the connection, or on a turn in the line. Undoing an erased connection is the same as undoing an erased block.

In these steps, you will erase the connection between the AHU1\OA#TEMP and ENRH1 blocks. The Erase icon should already be selected.

1. Move the cursor on either end of the line between AHU1\OA#TEMP and ENRH1. A yellow box appears. Click left and the connection line erases.
2. To undo the erased connection, double-click left on the Erase icon to display the option menu.
3. Click left on the UNDO option. The connection should return to the screen. Click left to clear the undo message.
4. Save the diagram again by clicking left on SAVE under the File option menu. Click left inside [OK] under the File Exists message.

Print the Diagram

This section of the tutorial explains how to send a diagram to an attached printer. The process requires two steps: creating a print file for the diagram, then sending the print file to the printer.

Note: This procedure requires that the printer is properly configured to accept strategy files. Refer to the *Getting Started* section in the *Introduction* chapter.

1. Click left on the Print icon. Its option menu displays.
2. Click left on the PRT ONE option. The Print submenu displays.
3. Click left inside the checkbox for Diagram(s). The diagram momentarily flashes on the entire screen. A print file of the diagram is created and added to the print queue.
4. Check to make sure the printer is online and ready. Click left on the Start Output option. Printing should begin in a moment. If any other files were listed on the print queue, they will also print.

Exit GPL

You have now completed the tutorial. You can exit GPL and go to the DOS or Metasys FMS environment.

1. Click left on the Exit icon. GPL displays this prompt:

```
                  EXIT TO:
       [ DOS ]  [ FMS ]  [CANCEL]
```

2. Click left on [DOS] to go to the DOS environment. Click left on [FMS] to go to the Metasys Network. Click left on [CANCEL] to escape the Exit operation.

Review

In this Tutorial, you have learned how to draw, change, save, and print a diagram. You also have learned how to compound a diagram. The last step explained how to exit GPL.

At this point, you can go to other parts of the manual to learn more about GPL.

Exiting GPL

This section explains how to exit the GPL Editor and return to DOS or the FMS.

1. Click left on the Exit icon (stop sign). The prompt `EXIT TO: [DOS] [FMS] [CANCEL]` displays.
2. Click left on [DOS] to go to the DOS environment. Click left on [FMS] to go to the Metasys Facilities Management System environment. Click left on [CANCEL] to escape the Exit operation.

Details

You must save a control strategy or compound that you have edited before exiting GPL.

> WARNING: If you exit GPL without saving the control strategy or compound that you edited, all changes made since the last save are lost.
>
> If you press the CTRL/C or CTRL/2 keys simultaneously, the GPL Editor will exit. Since this is a forced exit, all changes made to the strategy or compound file will be lost and unrecoverable.

To return to GPL from DOS, type `GPL` at the DOS prompt and press Enter. The computer will execute GPL, and the Editor screen will display. To return to GPL from the FMS, select GPL from the Exit menu at the Operator Workstation. The computer will exit Windows to DOS, and then execute GPL. The Editor screen will display.

Directory and Control Strategy Functions

This section describes how to perform the following directory and file functions:

- Use the File option menu.
- Page the directory.
- Create, save, load/edit, and delete a strategy.
- Display a description file.
- Change disk drives.
- Create a directory.

The File icon (disk) provides these functions.

Using the File Option Menu

Clicking left on the File icon (disk) displays an option menu (Figure 18).

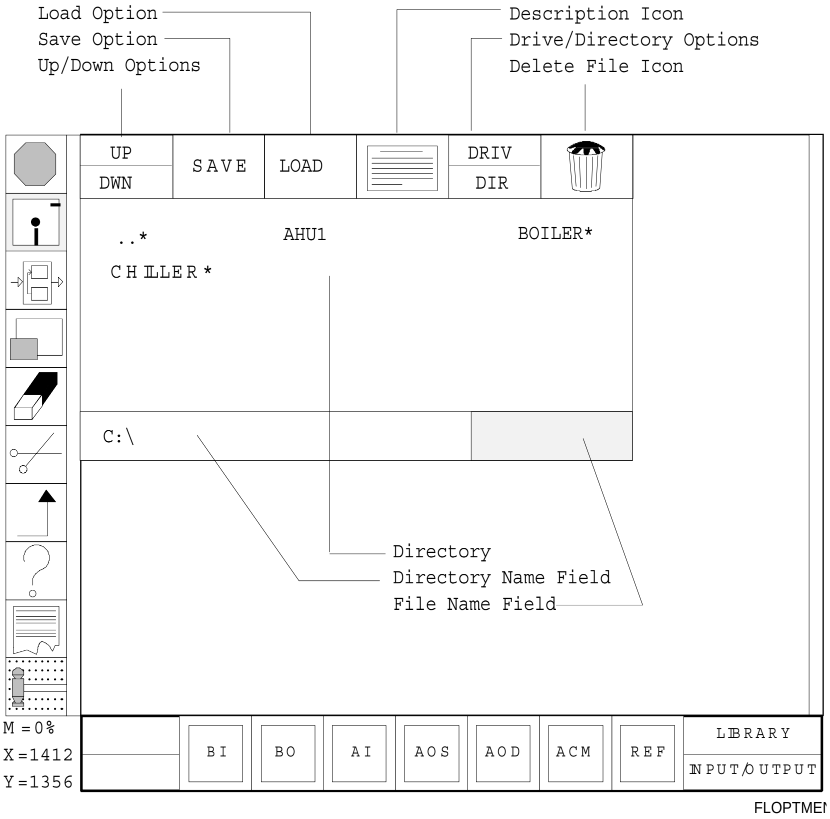
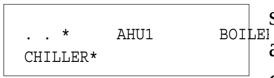


Figure 18: File Option Menu

As Figure 18 indicates, the File option menu features a Directory, Directory Name field, and File Name field. The following paragraphs describe them in detail.

Directory



```
. . *     AHU1       BOILE
CHILLER*
```

The **directory** is a list of all directories and GPL control strategy file names. Only files created with the GPL Editor are shown; all others are filtered out. The items are listed in alphabetical order from left to right. A maximum of 11 items can be listed per page. The listing cannot display directory names that contain extensions (e.g., GPL.DIR).

Items in the listing with asterisks (*) appended to their names are directories. Those without asterisks are files. Clicking on a directory name changes the directory to show all items that are under the directory.

The symbol “. .*” is used to reach higher level directories. Clicking on it changes the directory to display the items under the next-highest directory. The “. .*” symbol usually appears in the upper left corner. Clicking left on a file name selects it for saving, loading, deleting, or viewing its description file.

Directory Name Field

```
C:\
```

The **Directory Name** field shows the current directory. If the full directory name cannot be displayed, the names scroll to the left, dropping off the left-most characters as required to fit the entire lowest-level directory name.

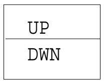
File Name Field

The **File Name** field is a space for typing in a directory or control strategy file name. Up to eight alphanumeric characters are allowed in a directory or file name. As you type in a name, the Editor filters the directory to show only those names beginning with the letters specified. For example, if you type in “S,” the directory changes to show every name that starts with the letter S. Then, if you type in “E” after the S, every name that begins with SE is displayed, and so on. Use this feature to quickly locate an item or to find an entry whose name you cannot fully remember. The DOS wildcard character (*) is not allowed. For other characters and words that are not allowed, refer to *Appendix F: Characters, Symbols, and Reserved Words.*

Note: An alternative to typing in a control strategy file name is to click left on its name in the directory. The Editor then places the selected name in the File Name field.

To type text in the File Name field, you must have the cursor inside the File option menu. The field is not case sensitive.

Paging the Directory



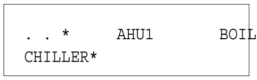
To show other items on the directory in the File option menu, page up and down. The UP option pages the directory back one page; the DWN option pages the directory forward one page.

- To page the directory back one page, click left on the UP option.
- To page the directory forward one page, click left on the DWN option.

Details

If you click left on UP or DWN and the directory does not change, you have reached the first or last page, respectively.

Selecting a Network



```
. . *     AHU1       BOILE
CHILLER*
```

This procedure selects a network. You need to select a network when the Network Name field displays `?? ARCH DB`, and you are creating a new control strategy. The procedure is required so that the Editor knows which archive database to interact with. The Editor interacts with the archive database when you are adding, modifying, and deleting object blocks and processes.

1. Click left on the File icon (disk). The option menu displays.
2. Go to the GPL subdirectory that is under the desired network. If you need to, go to a subdirectory under the GPL subdirectory.
3. Type a name for the strategy in the File Name field.
4. Click left on SAVE. If this strategy does not exist, the Editor creates it, and the Network Name field is updated to display the selected network. If this strategy already exists, a message displays that asks you to write over the file or cancel the save operation.

Saving a Control Strategy

SAVE

This procedure saves a displayed control strategy to any available disk drive after creating or editing. Each control strategy is stored in three files with the following extensions: .DB (database), .CI (connection), and .TX (text). These extensions do not appear in the directory and should not be entered in the File Name field. The following instructions presume the strategy to be saved is currently in the work area.

1. Click left on the File icon (disk). The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives*. If you need to change the directory, refer to *Using the File Option Menu.*
3. Click left on the strategy file name in the directory (if shown) or type a name inside the File Name field.
4. Click left on SAVE. If the file already exists, the message `WARNING! File Exists. UPDATE?` displays. Click left to [OK] to write over the strategy file. You may instead click left on [CANCEL] to escape the save operation.

Details

The file name you specify must follow DOS conventions. For example, the name must consist of alphanumeric characters and be at most eight characters. Also, the file name must not be a reserved word or contain invalid characters. Refer to *Appendix F: Characters, Symbols, and Reserved Words* for details.

When a control strategy is saved, all blocks are saved in their current states—defined or undefined. However, if the control strategy contains a FILE block, the strategy that it represents is not also saved.

To reach a higher-level directory, click left on the symbol “. .*”.

Before GPL saves an existing control strategy, it creates a set of backup files that represent the old version of the strategy. These backup files can be retrieved if the changes you made to the new version of the strategy are no longer wanted. Refer to the section on restoring backup files in *Appendix E: External Functions.*

The Editor will not allow you to exit until you have the chance to save the strategy you edited. If you change a strategy and then try to exit before saving it, the tool will prompt you when you click left on the Exit icon.

Loading a Control Strategy

LOAD

This procedure loads an existing control strategy from any available disk into the work area for viewing and editing. The work area does not have to be clear to load a strategy.

1. Click left on the File icon (disk). The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives.* If you need to change the directory, refer to *Using the File Option Menu.*
3. Click left on the control strategy file name in the directory or type its name in the File Name field.
4. Click left on LOAD. The contents of the strategy will come to the screen.

Details

Click left on the symbol ". .*" to reach a higher-level directory.

Only one control strategy can be loaded at a time.

When the Editor loads the strategy, it updates the strategy/compound File Name field to show the loaded file name.

Displaying a Strategy Description File

Each control strategy may have a strategy description file that you prepare with an ASCII editor outside of GPL. The description file has the same path file name as the control strategy, with a .DDS (diagram description) extension. The purpose of the file is to explain what each diagram in the control strategy does, and to contain any other related information, such as the author of the program and date of creation. The control strategy does not need to be loaded to view its description file. (Refer to *Appendix E: External Functions* for details on how to prepare a description file.)

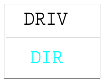
1. Click left on the File icon (disk). Click left on the control strategy file name in the directory or type its name in the file name field.
2. Click left on the Description icon. The description fills the entire work area, replacing a diagram if one is displayed.
3. Use the Page Up and Page Down keys to scroll through the file. Click left or Esc to clear the description and redisplay the diagram (if one was displayed).

Details

The description file can be displayed, but not edited or printed, with the GPL Editor. To print the file, use a DOS utility that can print ASCII files.

The description file has a maximum size of 30 pages. Use the Page Up and Page Down keys to move through the description file. Press Esc or click left to exit the file.

Changing Disk Drives



You can store and retrieve GPL directories and control strategy files from any available DOS disk drive. The GPL Editor recognizes Drive C as the default drive for all file and compound functions in the Editor. The valid disk drives are A through Z.

Note: You cannot use Drive B unless you have specified it in the System Generation (SYSGEN.EXE) file at installation time. See the installation information in the *Introduction* chapter for details.

1. Click left on the File icon (disk). The option menu displays.
2. Type the letter of a disk drive in the File Name field. Disk drives A through K are valid.
3. Click left on the DRIV option. The directory changes to show the contents of the selected disk.

Details

The Editor shows the current directory level that the GPLPATH statement specifies in the GPL.BAT file.

The directory is filtered to show only GPL control strategy files.

Creating a Directory

Use the DIR option to create a DOS directory under which GPL control strategy files can be stored.

1. Click left on the File icon (disk). The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the File Option Menu.*
3. If you want to create a subdirectory, go to the desired directory by clicking left on the directory name.
4. Enter a unique subdirectory name in the File Name field.
5. Click left on DIR. The new directory is created and added to the directory.

Details

Do not type an asterisk (*) after the directory name. For a list of other characters and words that are not allowed, refer to *Appendix F: Characters, Symbols, and Reserved Words.*

A GPL subdirectory may contain up to 100 files. You cannot delete GPL directories within the Editor.

Deleting a Control Strategy

This procedure deletes a control strategy from any available disk.

1. Click left on the File icon (disk). The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the File Option Menu*.
3. Click left on the name of the file in the directory or type its name in the File Name field.
4. Click left on the Delete File icon (trash can). The message `Verify file deletion` displays. To delete the control strategy, click left on [OK]. The control strategy is deleted and the directory refreshes. You may otherwise click left on [CANCEL] to escape this operation.

Details

This procedure deletes a set of files associated with the control strategy; they have the following extensions: .DB, .CI, and .TX. The .DDS (diagram description) file, since it is not created with GPL, is not deleted. It must be deleted outside of GPL. Also, this procedure does not delete the associated backup strategy files.

Once a control strategy is deleted, it cannot be restored within GPL. Use a special restoration program to restore deleted files. Alternatively, you can use the backup files of the control strategy (filename.ODB, .OCI, and .OTX), which should still be intact. Refer to the restoring backup files section in *Appendix E: External Functions*.

You may delete a control strategy that is currently displayed. However, it will not clear from the work area.

Compound Functions

This section describes how to perform the following compound functions:

- Use the Compound option menu.
- Page the directory.
- Make, edit, and delete a compound.
- Load a compound (either its block or contents).
- Display a description file.
- Change disk drives.
- Create a directory.

The Compound icon provides these functions.

Using the Compound Option Menu

Clicking left on the Compound icon displays an option menu (Figure 19).

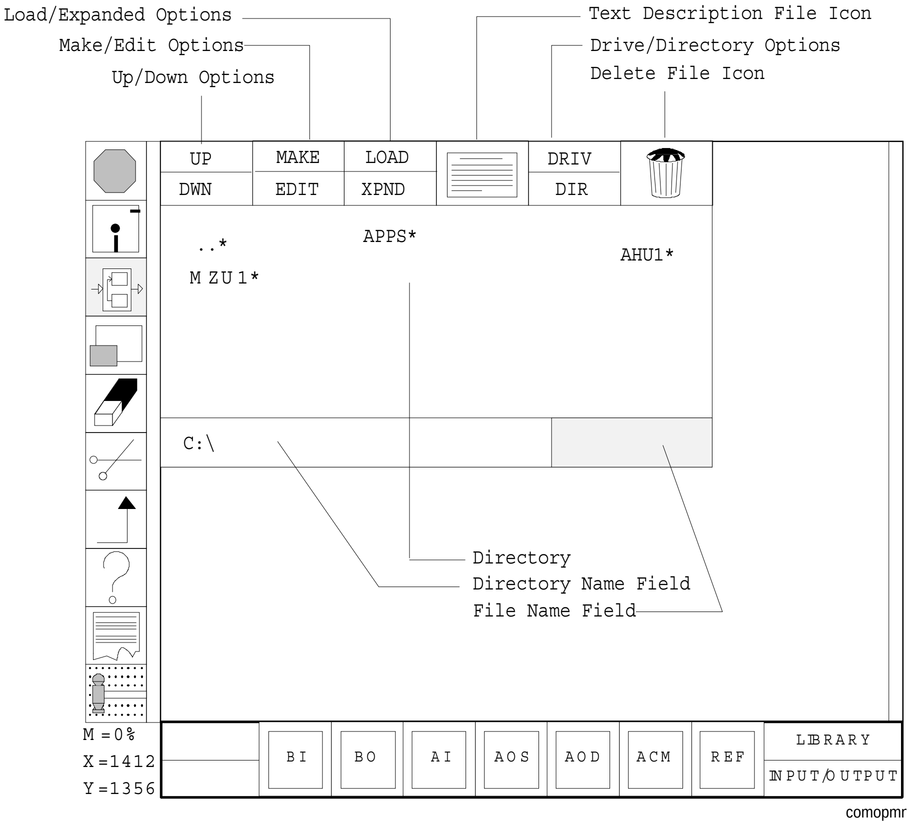
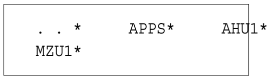


Figure 19: Compound Option Menu

As Figure 19 indicates, the Compound option menu features a Directory, Directory Name field, and File Name field. The following paragraphs describe them in detail.

Directory



```
. . *     APPS*     AHU1*
MZU1*
```

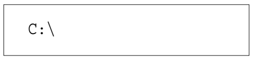
The **directory** is a list that shows all directories and compounds. They are listed in alphabetical order from left to right. A maximum of 11 items can be listed per page. The listing cannot display directory names that have extensions (e.g., GPL.DIR).

Items in the listing with asterisks (*) appended after their names are directories. Those without asterisks are files. Clicking on a directory name changes the list to show all items that are under the directory.

The symbol “. .*” is used to reach higher level directories. Clicking on it changes the list to display the items under the next-highest directory. The “. .*” symbol usually appears in the upper left corner. Clicking left on a compound name selects it for saving, editing, loading, deleting, or viewing its description file.

The SET GPLCMP statement in the GPL.BAT file determines which particular directory is shown when you click left on the Compound icon. If you wish, you may change the directory by editing the statement with a text editor. Follow the DOS conventions for editing a batch file.

Directory Name Field



```
C:\
```

The **Directory Name** field shows the current directory. If the full directory name cannot be displayed, the names scroll to the left, dropping off the left-most characters as required to fit the entire lowest-level directory name.

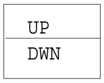
File Name Field

The **File Name** field is a space for typing in a compound, directory, or drive name. Up to eight alphanumeric characters are allowed in the name. As you type in a name, the Editor sorts the directory to show those names beginning with the letters specified. For example, if you type in “S,” the list changes to show every name that starts with the letter S. Then, if you type in “E” after the S, every name that begins with SE is displayed, and so on. Use this feature to quickly locate an item or to find an item whose name you cannot fully remember. The DOS wildcard character (*) is not allowed. For other characters and words that are not allowed, refer to *Appendix F: Characters, Symbols, and Reserved Words*.

Note: An alternative to typing in a compound name is to click left on its name in the directory. The Editor then places the selected name in the File Name field.

To type text in the File Name field, you must have the cursor inside the Compound option menu. The field is not case sensitive.

Paging the Directory



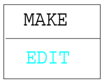
To show other items on the directory in the Compound option menu, page up and down. The UP option pages the directory back one page; the DWN option pages the directory forward one page.

- To page the directory back one page, click left on the UP option.
- To page the directory forward one page, click left on the DWN option.

Details

If you click left on UP or DWN and the directory does not change, you have reached the first or last page, respectively.

Making a Compound



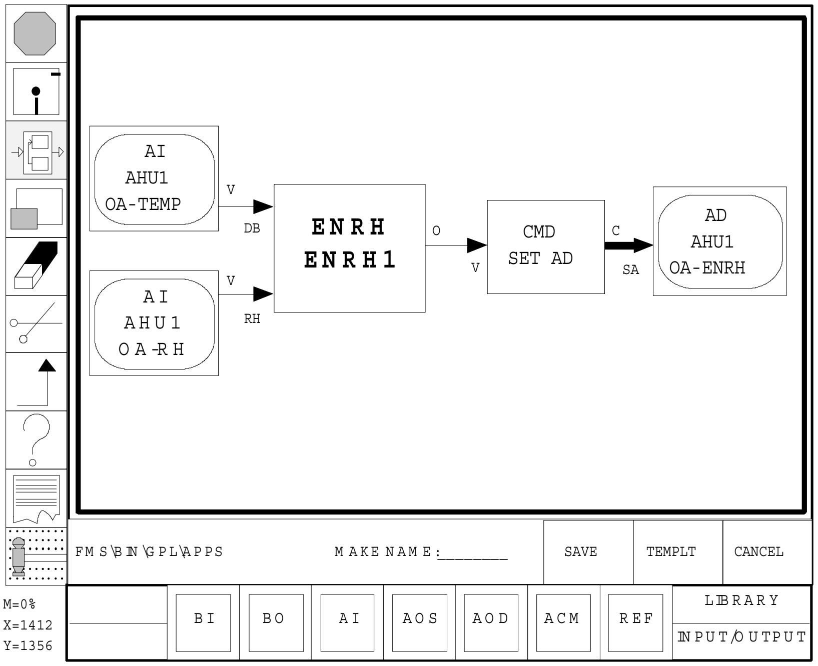
This procedure makes and saves a compound to any available disk drive. The compound file has a .CMP extension.

1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu*.
3. Click left on MAKE.

4. Move the mouse and notice that a white enclosing box appears. Use this box to enclose the area of a diagram that you want to compound.

5. Enclose the area of the diagram that you want to compound within the white box. If the box is not the proper size, hold down the right button and drag the mouse to resize it.

6. Make sure the section of the diagram that you want to compound is totally enclosed inside the white box. Click left. The Compound Make screen appears (Figure 20).



FMS\BIN\GPL\APPS

MAKE NAME:________

M=0%
X=1412
Y=1356

MAKESCRN

Figure 20: Compound Make Screen

7. Enter a name for this compound in the Name field. This is the name that is used to store the compound to disk. The entire pathname is not required, just a compound name. (The pathname is displayed to the left of the Name field.)

8. Click left on TEMPLT. The compound database template appears. Specify the type of compound (GROUP, PROCESS, or RESTART PROCESS) and the parameters that define this compound. Refer to the *Compounds* section in the *Graphic Programming* chapter for more information on the Compound block parameters.
9. Press F10 to save the parameter values. The template clears.
10. Click left on Save. A message displays, giving you the option of saving the compound to the screen or to the disk.

    If you want the Editor to create a compound block for you and paste it on the screen, click left on [SCREEN].

    If you want the Editor to save the compound to the disk, click left on [DISK].
11. The Editor checks if this compound is defined. If the compound already exists, the message `WARNING! File exists. UPDATE?` appears. Click left on [OK] to update the compound or click left on [CANCEL] to return to the Compound Make screen.

Details

The white enclosing box has a default size. When you resize the box, the default size is changed to this new size. Therefore, the next time the white enclosing box is displayed, the box will be this new size. Also, the size of a compound made to the screen matches the size of the enclosing box.

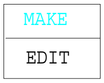
When a diagram is grouped into a compound, all of its function blocks shrink slightly in size. This is because the Compound Make screen is a bit smaller than the standard editing screen.

Before GPL saves a compound, it creates a backup file that represents the old version of the compound. The backup file has an .OCM extension. This backup file can be retrieved if the changes you made to the new version of the compound are no longer wanted. Refer to the section on restoring backup files in *Appendix E: External Functions*.

Click left on CANCEL in the compound editing screen to cancel the Make operation.

The maximum number of nested compound levels is 30.

Editing a Compound



This procedure loads an existing compound from any available disk into the work area for editing.

1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu.*
3. Click left on the name of the compound in the directory, or type in its name in the File Name field.
4. Click left on EDIT. The Compound Edit screen displays in the work area. Make your changes as you normally would in editing a strategy.
5. If you need to edit the compound's database template, click left on TEMPLT, and press F10 to save the changes.
6. After editing, click left on Save. The message `WARNING! File exists. UPDATE?` appears. Click left on [OK] to update the compound or click left on [CANCEL] to cancel the save operation.

Details

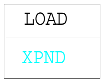
You can bring in a compound for editing while a diagram is in a work area. When you do so, the compound replaces the diagram in the work area. After you are finished with the compound, the diagram returns to the work area.

During editing, you can move up and down compounds that have multiple levels. To reach a lower level, double-click left on the compound block.

Note: To get back to the higher level compound, press Esc. Press Esc once for each level.

While you make changes to the compound or its template during the editing session, the archive database is not accessed. Therefore, object blocks cannot be set to defined.

Loading a Compound Block



This procedure loads a compound block into the work area from any available disk drive.

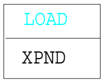
1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu*.
3. Click left on the name of the compound in the directory, or type in its name in the File Name field.
4. Click left on LOAD. Move the mouse and notice the compound block comes to the screen. Before pasting it down, you may resize the compound block by holding down the right/middle button and dragging the mouse.
5. Click left to paste down the compound block in the work area.

Details

When you load a process compound block into the work area, it comes in as undefined, and all object blocks under it also become undefined. This is a precaution that prevents from having duplicate process compounds and objects in the same file.

A compound block that is defined (colored brown) does not mean the blocks at the lower levels are also defined. You need to open the compound to determine if the blocks are defined.

Loading the Contents of a Compound



This procedure loads the contents (i.e., expanded version) of a compound. Use this procedure to create a new strategy or compound based on an existing one. The compound can come from any available disk drive.

1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu.*
3. Click left on the name of the compound in the directory, or type in its name in the File Name field.
4. Click left on XPND. Move the mouse and notice the outline of the compound's contents comes into the work area. Before pasting it down, you may resize the contents by holding down the right/middle button and dragging the mouse.
5. Click left to paste down the compound in the work area.

Details

The size of the compound's contents is the same as it was saved but can be resized before pasting it down. Refer to Step 4 above.

The expanded version of a protected compound cannot be loaded.

The expanded version of the compound does not include the template information of the compound block.

Displaying a Compound Description File

Each compound may have an associated description file that you prepare with an ASCII editor outside of GPL. The description file has the same path file name as the compound, with a .CDS (compound description) extension. The purpose of the file is to explain the logic of the diagrams in the compound, and to contain any other related information, such as the author of the compound and date of creation. The compound does not have to be loaded to view its description file.

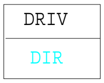
1. Click left on the Compound icon. The option menu displays.
2. Click left on the name of the compound in the directory, or type its name in the File Name field.
3. Click left on the Description icon. The description replaces the diagram in the work area (if displayed).
4. Use the Page Up and Page Down keys to scroll through the file. Click left or Esc to clear the file and redisplay the diagram (if one was displayed).

Details

The description file can only be displayed, not edited or printed, with the GPL Editor. To print the file, use a DOS utility that can print ASCII files.

The description file can be a maximum of 30 pages long.

Changing Disk Drives



You can store and retrieve compounds from any available DOS disk drive. The GPL Editor recognizes Drive C as the default drive for all file and compound management functions in GPL. The valid disk drives are A through Z.

Note: You cannot use Drive B unless you have specified it in the System Generation file (SYSGEN.EXE) at installation time. See the *Getting Started* section in *Introduction*.

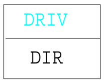
1. Click left on the Compound icon. The option menu displays.
2. Type the letter of a disk drive in the File Name field. The valid disk drives are A through Z.
3. Click left on the DRIV option. The directory changes to show the contents of the selected disk.

Details

The Editor shows the current directory level that the GPLPATH statement specifies in the GPL.BAT file.

The Editor filters the directory to show only GPL compound files.

Creating a Directory



Use the DIR option to create a DOS format directory under which you can store GPL compounds.

1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu*.
3. If this is to be a subdirectory, go to the desired directory by clicking left on the directory name.
4. Enter the new subdirectory name in the File Name field.
5. Click left on DIR. The new directory is created and added to the list.

Details

Do not type an asterisk (*) after the directory name. For a list of other characters and words that are not allowed, refer to *Appendix F: Characters, Symbols, and Reserved Words*.

The Editor supports up to 100 directories. You cannot delete a directory in GPL.

Deleting a Compound

This procedure deletes a compound from any available disk.

1. Click left on the Compound icon. The option menu displays.
2. If you need to change the disk drive, refer to *Changing Disk Drives* for instructions. If you need to change the directory, refer to *Using the Compound Option Menu*.
3. Click left on the name of the compound in the directory, or type in its name in the File Name field.
4. Click left on the Delete File icon (trash can). The message `Verify file deletion` displays. To delete the compound, click left on [OK]. The compound is deleted and the list refreshes. You may otherwise click left on [CANCEL] to escape this operation.

Details

This procedure deletes the file with a .CMP extension. The .CDS (compound description) file is not created with GPL, and therefore is not deleted. It must be deleted outside of GPL. Also, this procedure does not delete the associated backup compound file.

Once a compound is deleted, it cannot be restored within GPL. Use a special restoration program to restore deleted compounds. Alternatively, you can use the backup file of the compound (filename.OCM), which should still be intact. Refer to the restoring backup files section in *Appendix E: External Functions.*

You may delete a compound that has its block currently displayed. However, the block will not clear from the work area.

Zoom and Pan Functions

This section explains how to zoom and pan a diagram. The Zoom and Pan icon provides the zoom and pan functions. Double-clicking left on the Zoom and Pan icon displays an option menu (Figure 21).

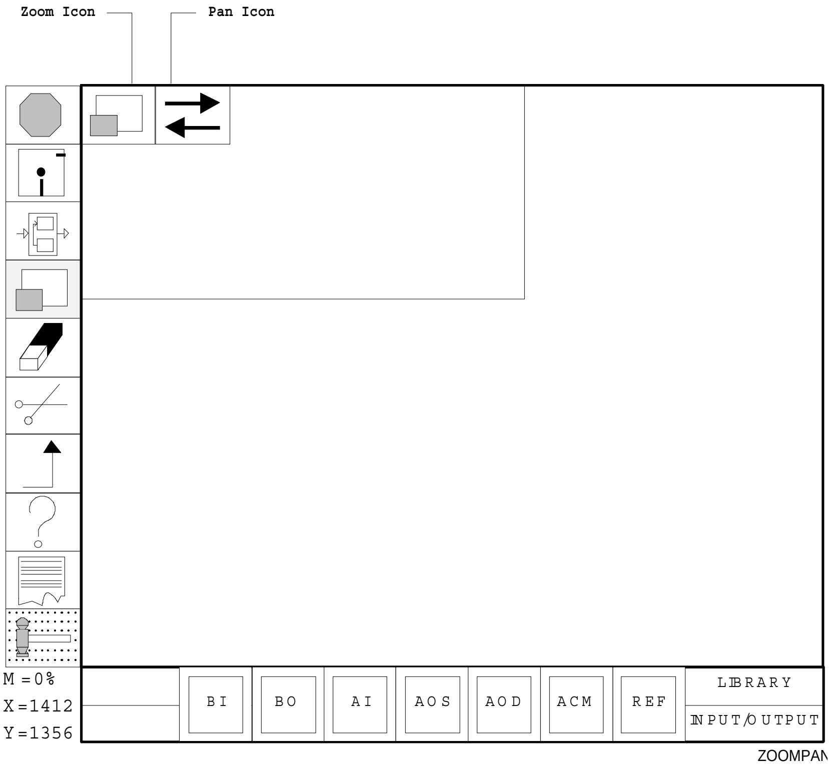


Figure 21: Zoom and Pan Option Menu

Note: The default operation of this icon is zoom in.

The procedures in this section presume a diagram is currently displayed.

Zooming In and Out

The zoom in function enlarges a portion of a diagram for closer viewing, much like a zoom lens on a camera enlarges an image. Zoom out returns it to its previous size. A diagram can be zoomed up to four times by successively clicking left. The GPL Editor does not indicate if a diagram is zoomed.

1. Click left on the Pan and Zoom icon.
2. Move the mouse and notice that a white enclosing box appears. This box selects the portion of the diagram that you want to zoom.
3. While holding down the right/middle button and dragging the mouse, size the white box to a size large enough to enclose the portion of the diagram you want to zoom.
4. Click left. The selected area enlarges.
5. If you want to zoom further, click left again. You can zoom up to four times.
6. To zoom out, double-click left. One double-click will zoom out one level.

Details

The degree of the zoom depends on how large you size the white enclosing box. The smaller the white box, the larger the selected portion is zoomed.

Use the left button to zoom in and out: a single click zooms in; a double-click zooms out. Hold down the right/middle button to resize the white enclosing box.

The white enclosing box has a default size. Even if you resize it, the box returns to its default size after you complete the zoom operation.

You may edit a diagram that is zoomed, but when you return it to regular size, all blocks and text you added will shrink in respect to the size of the zoomed area.

When you zoom in a compound, the compound name fields disappear. They return when you zoom out to the original size.

Panning

⇄

This procedure lets you pan across a zoomed diagram in the work area. Panning “slides” a diagram up, down, and across so you can view particular areas of the diagram. The GPL Editor does not indicate if a diagram is panned.

1. Double-click left on the Pan and Zoom icon. The option menu displays.
2. Click left on the Pan icon.
3. While holding down the left button, drag the mouse across the screen. Release the button. The screen pans in the direction of the mouse.

Details

Panning is only available when the work area is zoomed in. To return the diagram to original size, click left on the Pan and Zoom icon, place the cursor in the work area, and double-click left (double-click for each zoomed level).

Erase and Delete Functions

This section explains how to perform these functions:

- Erase an item or group of items. Items include function and compound blocks, connection lines, text, or analog displays.
- Delete an object from the archive database.
- Clear memory.
- Undo an erased item or group of items.

The Erase icon (eraser) provides these functions. The delete function removes a defined object from the archive database, and the erase function erases an item from the work area.

Double-clicking left on the Erase icon displays an option menu (Figure 22).

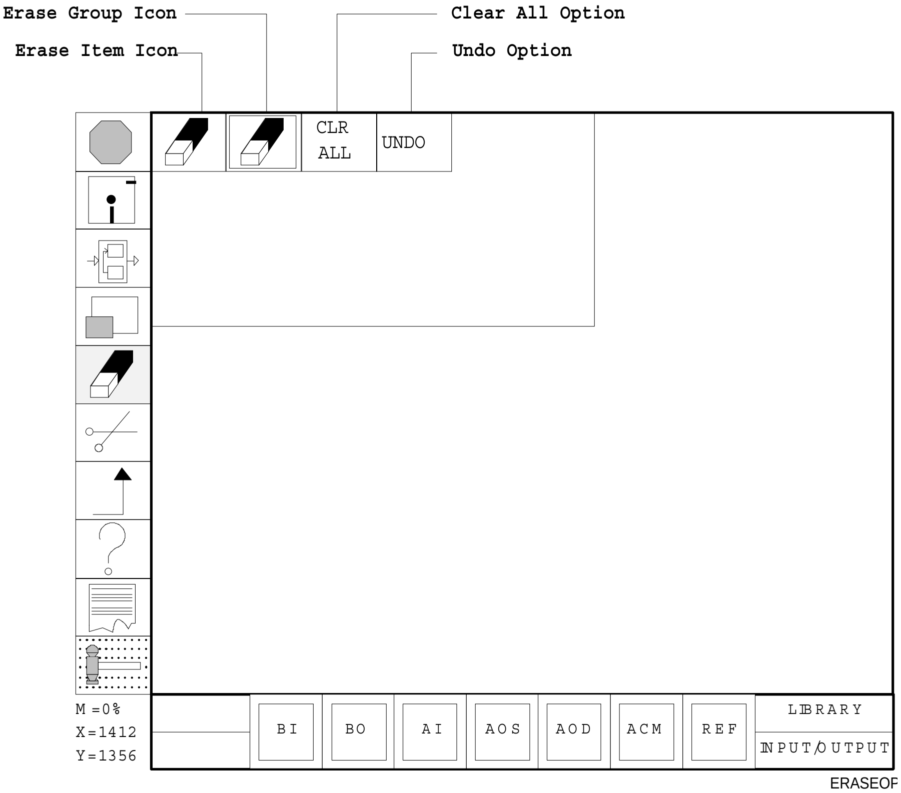


Figure 22: Erase Option Menu

Note: The default operation of this icon is erase item.

Erasing an Item

The items that you can erase from a diagram include function and compound blocks, connection lines, remote connections, text, and analog displays.

1. Click left on the Erase icon (eraser).
2. If you want to erase an undefined object block, or an operation or special block, move the cursor into the block and click left. The block and its connections erase.

If you want to erase a defined object block, move the cursor into the block and click left. The message `Delete this object from the archive database?` displays on the bottom of the work area. To keep the object in the archive database, click left on [NO]. To delete the object from the archive, click left on [YES]. Then, another message displays which asks, `Erase this block from the GPL strategy file?` To erase the block from the file, click left on [YES]; otherwise, to keep the object block, click left on [NO].

If you want to erase a connection line only, move the cursor at either end of the line, or on a turn in a line. A yellow box appears, showing that this connection is selected for deletion. Click left. The connection line erases.

If you want to erase a remote connection, click left on it, or select the end of or a turn in the remote connection line and click left.

If you want to erase text or an analog display, move the cursor on it. A yellow box appears, enclosing the text or analog display to be erased. Click left. The text or analog display erases.

Details

If you accidentally erase an item, use the undo function to get it back. See the section *Undoing an Erased Item or Group of Items* for details.

When you erase an item, the work area does not refresh. Therefore, the blocks, connections, text, and analog displays that are adjacent to the erased block may also erase partially. Simply click left in an open spot on the work area, and these items will return.

If an object block that has a remote connection is erased, the remote connection is also erased.

When selected for erase, a REF object block erases immediately and the Editor does not display the delete object option (just like an operation block). The object for the REF block in the archive database is not affected.

Erasing a Group of Items

This function erases a group of items on a diagram. Items that can be erased include function and compound blocks, connection lines, remote connections, text, and analog displays. However, this function does not delete objects from the archive database.

1. Double-click left on the Erase icon (eraser).
2. Click left on the Erase Group icon (framed eraser).
3. Move the mouse and notice that a white enclosing box appears. This box selects the group of items on the diagram that you want to erase.
4. While holding down the right/middle button and dragging the mouse, size the box large enough to enclose the group that you want to erase. Move the mouse to position the box within the group.
5. Click left. The selected group erases.

Details

Function and compound blocks, remote connections, text, and analog displays must be totally enclosed in the white enclosing box to erase. Connection lines, however, do not. Also, you cannot erase an individual connection line by enclosing it within the white box.

If you inadvertently erase a group of items, use the Undo function to get it back. See the section *Undoing an Erased Item or Group of Items* for details.

When you erase a group, the work area does not refresh. Therefore, the blocks, connections, text, and analog displays that are adjacent to the erased group may also erase partially. Simply click left on any open spot in the work area, and these items will return.

The white enclosing box has a default size. It is changed, however, when you resize it. The new size then becomes the default size—the size of the white enclosing box the next time the eraser function is used.

If an object block that has a remote connection is erased, the remote connection is also erased.

When selected for erase, a REF object block erases immediately and the Editor does not display the delete object option (just like an operation block). The object for the REF block in the archive database is not affected.

Deleting an Object

With this procedure, you can delete a software object or process object from the archive database. Only one object at a time can be deleted from the archive. That means you cannot use the Erase Group icon to delete multiple objects at one time. Also, a REF object block can only be erased from the file, not deleted from the archive, since it is only a reference to the object, not an actual object.

1. Click left on the Erase icon (eraser).
2. Move the cursor into the object block and click left. The message `Delete this object from the archive database?` displays on the bottom of the work area.
3. To delete the object from the archive database, click left on [YES]. To keep the object in the archive database, click left on [NO]. Then, another message displays which asks, `Erase this block from the GPL strategy file?` To erase the block from the file, click left on [YES]; otherwise, to keep the object, click left on [NO].

Details

When you delete an object from the archive database but keep it in the strategy file, its border changes to magenta.

When you delete a process compound from the archive database, only the process object for the compound is deleted, not all object blocks defined in the compound.

Note: After you delete several objects, use ARCHPACK to compress the archive database.

Clearing Memory

CLR
ALL

This function clears the computer's RAM, which effectively clears the entire control strategy from the work area. The function is helpful when you want to "clear the slate" and start over.

> CAUTION: Before this function is performed, the message `WARNING! You are about to clear the entire diagram` displays as a precaution. *If you clear the screen, all changes made since the last save are lost and cannot be recovered with the Undo function*.

1. Double-click left on the Erase icon (eraser). The option menu displays.
2. Click left on CLR ALL. The message `WARNING!  You are about to clear the entire diagram` appears.
3. Click left on [OK] to clear the work area. You may instead click left on [CANCEL] to exit this operation.

Details

After you clear the memory, the network is deselected, the Network Name field displays `?? ARCH DB`, and the memory usage figure changes to 0.

Undoing an Erased Item or Group of Items

The Undo function allows you to retrieve items of a control strategy that you previously erased. The items can be function and compound blocks, connection lines, text, and analog displays. These steps presume that at least one erased action is in memory, ready to be restored.

1. Double-click left on the Erase icon (eraser). The option menu displays.
2. Click left on UNDO. The items that you last erased return to the work area. The message `Last erased item restored. Returning to Erase mode` displays. Click left to clear the message.
3. To retrieve another erased item, repeat the previous two steps.

Details

The Undo function returns the erased items for each individual erase action up to the last time the file was saved. Up to 250 erase actions can be restored.

Since a control strategy may consist of multiple compound levels, it is possible to restore an item that is on a level different from the one currently displayed. In that case, you would not see the item being restored to the work area. Therefore, the message `Last erased item restored. Returning to Erase mode` will always display after each undo action to indicate that the GPL Editor has, in fact, performed the operation.

When an object block is restored, the block is returned as undefined. This eliminates the chance of having two blocks with the same system\object name. However, any changes made in the template remain intact.

An erased object block that has a remote connection can be restored. When that happens, the remote connection symbol returns labeled as before.

Move, Copy, and Resize Functions

This section explains how to perform the following functions:

- Move, copy, and resize an item or group of items. Items include function and compound blocks, connections, text, and analog displays.
- Add a new line segment.

The Move icon (scissors) provides these functions.

Double-clicking left on the Move icon displays an option menu (Figure 23).

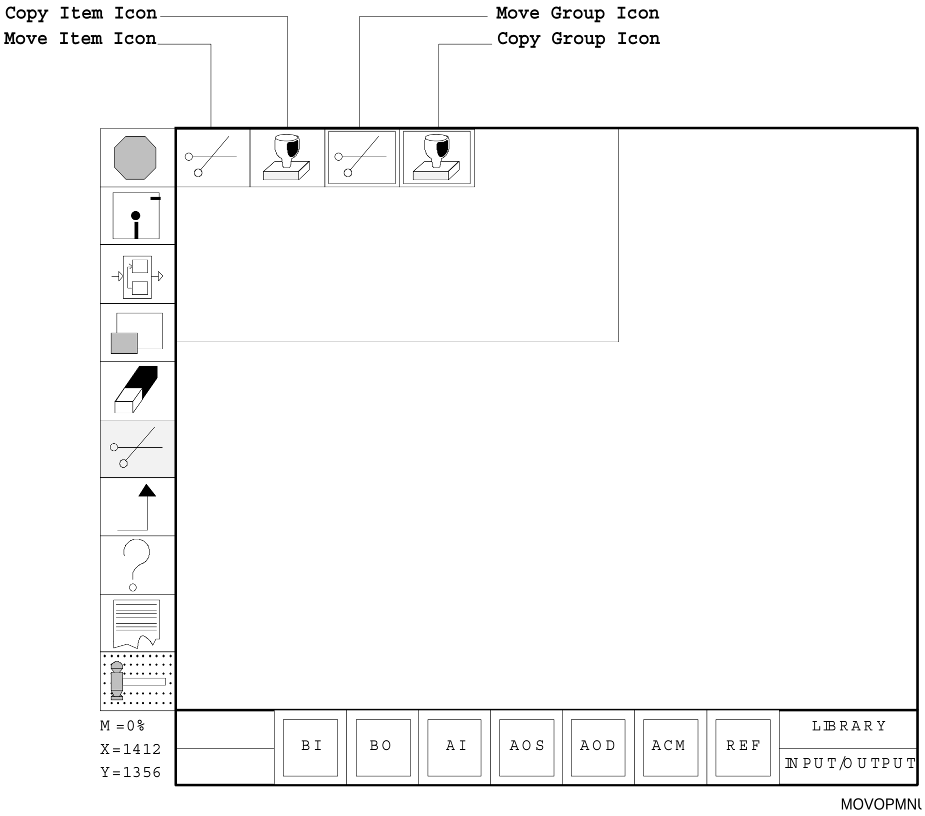


Figure 23: Move Option Menu

Note: The default operation of this icon is move.

The procedures discussed in this section presume a diagram is currently displayed.

Moving an Item

You can move any item pasted down on a diagram, including blocks, line segments, text, and analog displays. The item can be moved to any spot on the work area.

1. Click left on the Move icon (scissors).
2. If you want to move a block (and its connection lines), click left on it. The title of the block disappears. Move the block to a new position and click again to paste it down.

   If you want to move a segment of a connection line, move the cursor to the beginning, end, or turn in the line.
   A yellow box appears that indicates the line is selected for move. Click left. Move the line to a new position and click left again to paste down.

   If you want to move text, position the cursor on the text and a yellow box appears. Click left. The text disappears and a white box appears. Move the box to where the text should go and click left to paste it down.

   If you want to move an analog display, position the cursor on the display and a yellow box appears. Click left. The display becomes free to move. Move it to a new position and click left to paste down. The analog display is still associated with its connection.

Details

When you select a block for move, its label disappears. That is normal. When you select text, it disappears and a white box appears. That, too, is normal.

All connection lines leaving or entering a block move with the block.

You may notice that the GPL Editor limits the area in which the beginning or end of a line can be moved. This is necessary so that the line remains connected to the block.

You can straighten a connection line that has two right angle turns as follows (Figure 24):

1. Position the cursor at the upper turn until the yellow box appears. Click left and move the mouse until the upper turn is in line with the bottom line. Click left again to paste it down.
2. Position the cursor at the end of the line (i.e., on the arrowhead). With the yellow box highlighted, click left and move it to straighten the line; click again to paste it down.

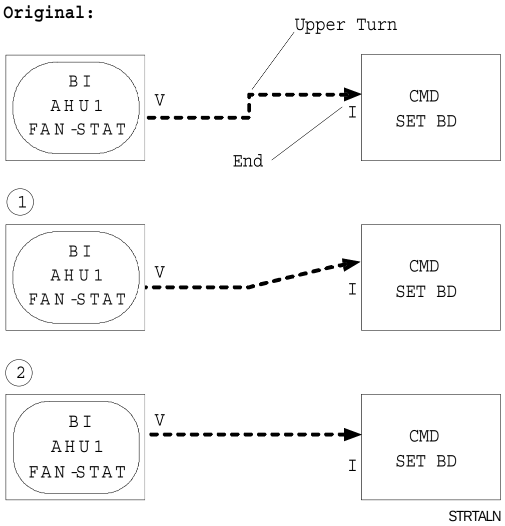


Figure 24: Straightening a Line

You may resize an item that you are moving before pasting it down. Hold down the right/middle button and drag the mouse. An individual analog display should not be resized, since it will not display as the new size in Simulator.

You can move items from one area of the screen to another. The GPL Editor does not allow you to move an item from one diagram to another. You may, however, group the area to be moved into a group compound, and load the expanded version of the compound into another diagram.

If you undo an erased connection line, and you moved the function block after the erased action, the line will be restored at its original position, not at the block's new position.

Copying an Item

This function makes a copy of a block, text string, or analog display. You may find it easier to copy an existing item than to paste down a new one, because copying a block also copies its template information.

1. Double-click left on the Move icon (scissors). The option menu displays.
2. Click on the Copy icon (stamp).
3. If you want to copy a function or compound block, click left anywhere inside the block. A copy displays on top of the original. Move the duplicate to a desired position, and click left to paste down.

   If you want to copy a text string, position the cursor on the text and a yellow box appears. Click left. The text disappears and a white box appears. Move the box to where the duplicate should go and click left to paste it down.

   If you want to copy an analog display, position the cursor on the display and a yellow box appears. Click left. A duplicate appears. Move it to a position and click left to paste it down.

Details

When you select a block for duplication, its name disappears. That is normal. When you select text, it disappears and a white box appears. That, too, is normal.

You can copy items from one area of the screen to another. The GPL Editor does not allow you to copy an item from one diagram to another. You may, however, group the area to be copied into a group compound, and load the expanded version of the compound into another diagram.

You may resize an item you are copying before pasting it down. Hold down the right/middle button and drag the mouse. You should not resize an individual analog display, since it will not display as the new size in the Simulator.

When you copy a function block, you also copy its database. A copied operation or special block is defined, but a copied object or process block is undefined. You need to define all copied object and process blocks separately, since only one object of the same name can exist in the same strategy file. Also, all object blocks under a copied process compound block are changed to undefined.

When a compound block is copied, all compounds and diagrams under it are also copied. In this way, you can use the copy function to duplicate a complete system control strategy in one step. For example, you may want to create a new air handling strategy by copying an existing strategy and merely editing the system\object names in the new strategy.

If the error message `Too many items for move/copy operation` displays when you try to copy a compound, the temporary buffer used in this operation is full. If the compound exists on disk, try instead to paste down the compound using the LOAD option. Or, create a new compound to disk, selecting the parts of this compound you wish to copy; then, paste down that compound.

Moving a Group of Items

A group of blocks, connections, text, and analog displays can be moved together at one time.

1. Double-click left on the Move icon (scissors). The option menu displays.
2. Click on the Move Group icon (framed scissors).
3. Move the mouse and notice a white enclosing box appears. This box selects the group that you want to move.
4. While holding down the right/middle button and dragging the mouse, size the white enclosing box large enough to enclose the group. Position the box within the group and click left. The block labels disappear.
5. Move the group to its new position and click left to paste it down.

Details

When you select a group of items for move, their labels disappear. That is normal.

Each item in the group must be completely enclosed in the white enclosing box for it to be selected for move. The connection line is an exception. If the block it is connected to is completely enclosed, the line does not have to be enclosed.

You may resize a group of items you are moving before pasting it down. Hold down the right/middle button and drag the mouse. Resizing is performed proportionally.

The white enclosing box has a default size. It is changed, however, when you resize it. The new size then becomes the default size—the size of the white enclosing box the next time the Move Group function is used (or any other function that uses the enclosing box).

You can move a group of items from one area of the screen to another. The GPL Editor does not allow you to move a group from one diagram to another. You may, however, group the items to be moved into a group compound, and load the expanded version of the compound into another diagram.

If the error message `Too many items for move/copy operation` displays when you try a group move, the temporary buffer used in this operation is full. Try instead to move less items or move each item individually.

Copying a Group of Items

You can copy a group of blocks, connections, text, and analog displays together at one time.

1. Double-click left on the Move icon (scissors). The option menu displays.
2. Click on the Copy Group icon (framed stamp).
3. Move the mouse and notice a white enclosing box appears. This box selects the group that you want to copy.
4. While holding down the right/middle button and dragging the mouse, size the white enclosing box large enough to enclose the group. Position the box within the group and click left. The block labels disappear.
5. Move the copy to a spot in the work area and click left to paste down.

Details

When you select a group of items for copy, their labels disappear. That is normal.

For a group to be selected, make sure each item in the group is completely enclosed in the white enclosing box. The connection line is an exception. If the block it is connected to is completely enclosed, the line does not have to be enclosed.

You may resize a group of items you are copying before pasting it down. Hold down the right/middle button and drag the mouse.

The white enclosing box has a default size. It is changed, however, when you resize it. The new size then becomes the default size—the size of the white enclosing box the next time the Copy Group function is used (or any other function that uses the enclosing box).

You can copy a group of items from one area of the screen to another. The GPL Editor does not allow you to copy a group from one diagram to another. You may, however, group the items to be copied into a group compound, and load the compound into another diagram.

If the error message `Too many items for move/copy operation` displays when you try a group copy, the temporary buffer used in this operation is full. Try instead to copy less items or copy each item individually. If this is an individual compound and it exists on disk, paste down the compound using the LOAD option. Or, create a new compound to disk, selecting the parts of this compound you wish to copy; then, paste down that compound.

Resizing an Item

You can resize a block or text. Resizing is performed proportionally.

Note: Connections and analog displays can be resized only with the Resize Group function.

1. Click left on the Move icon (scissors).
2. If you want to resize a block, click left on the block. The block becomes free to move and resize. Hold down the right/middle button and drag the mouse to resize. Position the block and click left to paste down.

   If you want to resize a text string, position the cursor on the text and a yellow box appears. Click left. The text disappears and a white box appears. Hold down the right/middle button and drag the mouse to size the white box. Position the text and click left to paste it down.

Resizing a Group of Items

You can resize a group of blocks, connections, text, and analog displays together at one time. Resizing is performed proportionally.

1. Double-click left on the Move icon (scissors). The option menu displays.
2. Click on the Move Group icon (framed scissors).
3. Move the mouse and notice a white enclosing box appears. This box selects the group that you want to resize.

4. While holding down the right/middle button and dragging the mouse, size the white enclosing box large enough to enclose the group. Position the box within the group and click left. The block labels disappear.
5. Hold down the right/middle button and drag the mouse. Resize the group. Click left to paste down.

Details

When you select a group of items for resizing, their labels disappear. That is normal.

For a group to be selected, make sure each item in the group is completely enclosed in the white enclosing box. The connection line is an exception. If the block it is connected to is completely enclosed, the line does not have to be enclosed.

The white enclosing box has a default size. It is changed, however, when you resize it. The new size then becomes the default size--the size of the white enclosing box the next time the Resize Group function is used (or any other function that uses the enclosing box).

Adding a Line Segment

You can add a line segment to an existing line by following this procedure.

1. Click left on the Move icon (scissors).
2. Position the cursor on the turn in the line where the new segment is to be formed. A yellow highlighting box appears. Click right and drag the mouse. A new segment appears. Click left to paste it down.

Details

You cannot add a line segment to the end of a connection (i.e., at its arrowhead).

Block Connection and Command Functions

This section discusses the basics of drawing connections and how to perform these functions:

- Connect two function or compound blocks, either within the same diagram or different diagrams.
- Connect two function or compound blocks remotely.
- Command objects and processes.
- Exempting a connection from triggering a process.

The Connection icon (bent-up arrow) provides these functions. Double-clicking left on the Connection icon displays an option menu (Figure 25).

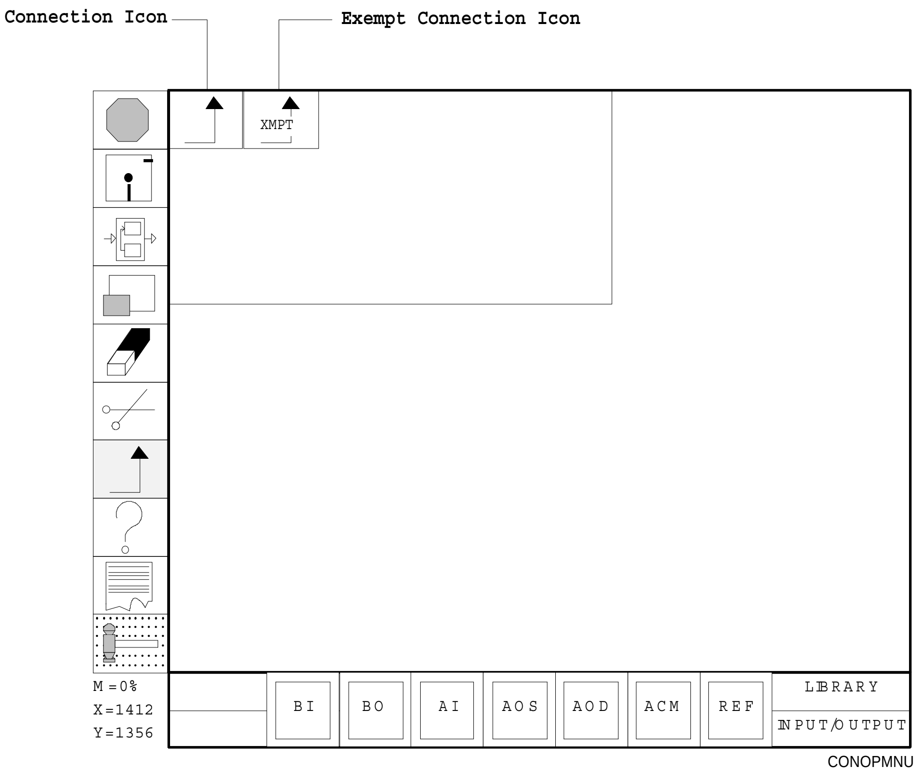


Figure 25: Connection Option Menu

Note: The default operation of this icon is connect block.

Learning the Basics of Connecting Blocks

By drawing lines between function or compound blocks, you accomplish GPL data and control flow. You draw lines by selecting from a list of inputs and outputs for each block. These inputs and outputs are listed on connection menus that display in the work area while you are making a connection (Figure 26).

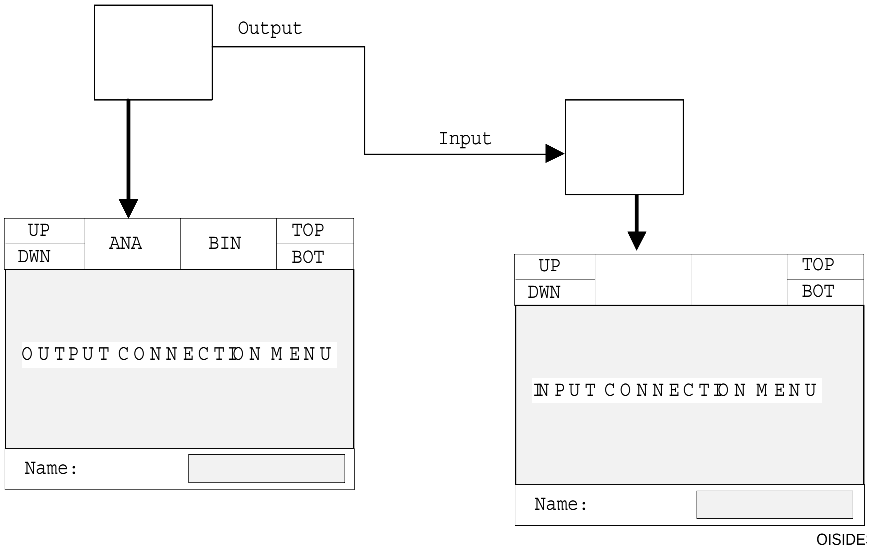


Figure 26: Output and Input Sides of a Connection

Clicking left on a block displays its output connection menu. Selecting an output from the menu starts the line from the edge of the block. Then, moving the line into another block and clicking left again displays the input connection menu. After you select an input, the Editor draws the line between the blocks.

All connection lines are white, but the appearance of the line varies according to the type of data being exchanged. The following table outlines the data types:

Type of Data	Line Shape
Analog or Time	Solid
Binary	Dashed
Control	Dotted
Command	Heavy Solid
Dual Command	Heavy Solid
Read	Heavy Solid
Write	Heavy Solid

Details

The only prerequisite to connecting two blocks is that both blocks are pasted down in the work area. In almost all cases, blocks do not have to be defined to be connected. However, you may have to specify in the database template the type of data that can be connected to a particular block. For example, before you connect a time data line to an ADD block, you have to first configure the ADD block to accept time data. You do this in the block's database template.

Almost all function blocks require either an input or output connection. The Editor presents all available choices in the connection menus. Object blocks, however, can stand alone on a diagram without connections. Also, two other function blocks may not require connections: the FILE block (never connected) and the USER block (depends on application).

You may cancel the drawing of a line any time during the connection procedure. To cancel before you selected an output connection, click anywhere outside the menu. To cancel after you selected an output connection, click the right/middle button once to erase each formed line segment. To cancel while the input menu is displayed, click anywhere outside the menu.

The instructions in this section refer to an origin block and a destination block (Figure 27). The **origin block** is where a connection *starts*; the **destination block** is where a connection *ends*.

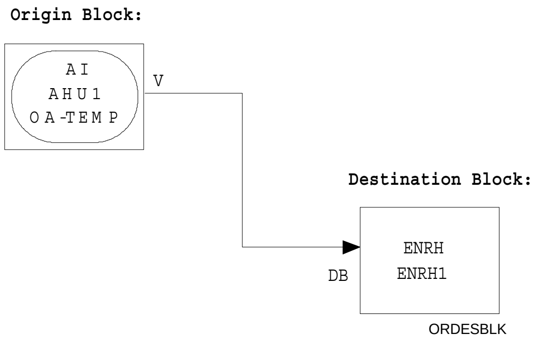


Figure 27: Origin and Destination Blocks

When you make connections with the grid on, the lines “snap” at 90 degree angles. That is, the lines you draw diagonally paste down as straight lines. This speeds up the process of drawing neat and consistent lines. If the grid is off, the lines are pasted down exactly as drawn. See the *Tools Functions* section for details.

Once you connect two blocks, each end of the line has a 1- or 2-character label that abbreviates the selected connection type (Figure 28). The *Function Blocks* chapter lists the meanings of all line labels per block.

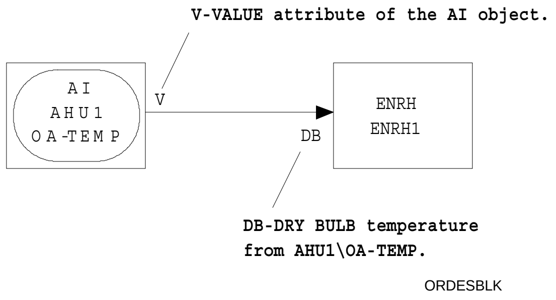


Figure 28: Line Labels for Two Connected Blocks

You may draw connection lines across or on top of each other. GPL is able to distinguish between the two different lines.

A direct connection can have up to 14 line segments, and a remote connection, seven line segments. Also, a direct connection can penetrate up to ten compound levels.

Connection lines to show data or control flow can be drawn in any direction. Figure 29 shows various directions. However, for easy readability, it is best to show data flow from left to right and from top to bottom.

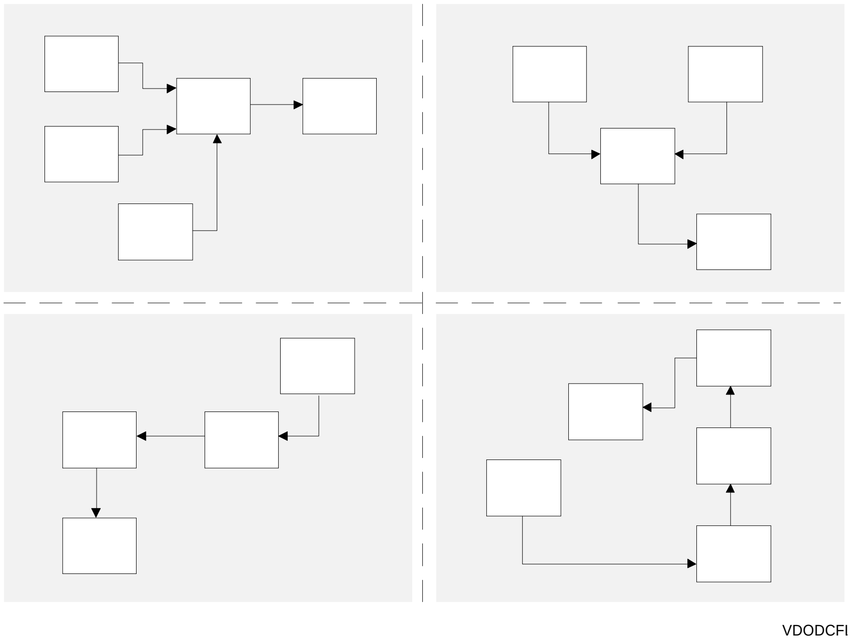
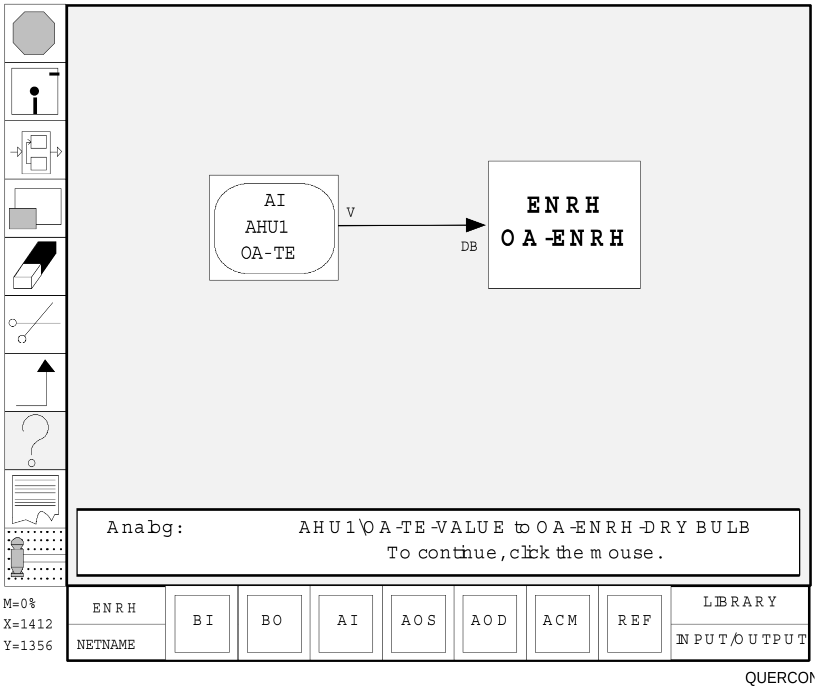


Figure 29: Various Directions of Data or Control Flow

The number of lines that can enter or exit a function block is preset. If you try to connect more inputs or outputs than is allowed, the GPL Editor will notify you with an appropriate error message. The *User Messages* section at the end of this chapter explains all error messages.

For some blocks, you may configure (i.e., specify) the number of connection lines in the block’s database template. The AND block, for example, may have two, three, or four input connections. The Number of Inputs field in the AND block template determines how many inputs the Editor makes available in the input connection menu. You can change the number of configured inputs at any time.

Whenever you want information about an existing connection, a brief description of it is available. Select the Query icon and position the cursor on the beginning, end, or corner of the line. A yellow box displays to indicate the line is selected. Click left, and a brief description about the origin and destination blocks appears in the lower portion of the work area (Figure 30).



Analog: AHU1\OA-TE-VALUE to OA-ENRH-DRY BULB
To continue, click the mouse.

Figure 30: Querying a Connection

The Expert Checker verifies that you have made every required connection. It does not, however, check if the connections you made are sensible. The *Function Blocks* chapter indicates all required and optional connections for each block.

Connecting Two Blocks

Connecting blocks establishes the data or control flow from one block to another. When you make a connection, the Editor checks to see if this is a valid connection; if not, a message displays on the bottom of the work area. For example, the Editor would not allow an analog type line to be an input to a binary connection.

Introduction

The process of drawing a line between blocks involves selecting the correct input and output connection name from the connection menus that display. Figure 31 is a sample of a direct connection menu.

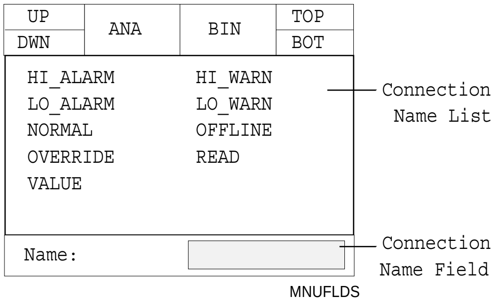


Figure 31: Direct Connection Menu

Menu Fields

As shown, the menu is composed of the following fields, which perform these operations:

UP

Pages the Connection Name list back one page. If you click left on UP and the list does not change, you have reached the first page.

DWN

Pages the Connection Name list forward one page. If you click left on DWN and the list does not change, you have reached the last page.

ANA

Filters the Connection Name list to show only analog connections. Click left on ANA to filter the list to show all analog connections (ANA back-highlights in red). Click left on ANA again to show all connections (ANA dehighlights).

BIN

Filters the Connection Name list to show all binary connections. Click left on BIN to filter the list to show all binary connections (BIN back-highlights in red). Click left on BIN again to show all connections (BIN dehighlights).

TOP

Displays the first page (top) of the Connection Name list. If you click left on TOP and the list does not change, you have reached the first page.

BOT

Displays the last page (bottom) of the Connection Name list. If you click left on BOT and the list does not change, you have reached the last page.

Connection Name List

Lists all possible connections for the block, arranged alphabetically from left to right. Click left on the connection name to select it. The connection names are color coded in the menu to distinguish their types.

Connection Type	Color
Analog	Yellow
Binary	Blue
Time	Brown
Control	Green
Command	White
Dual Command	White
Read	White
Write	White

Connection Name Field

Field for typing in a valid connection name. As you type the name, the Editor filters the list to show only those entries beginning with the letters specified. Use the backspace key to erase a typed letter and refilter the list. The field is not case sensitive.

Procedure

To make a direct connection on the same diagram, follow these steps. For making a connection between compounds (i.e., different diagrams), or making a remote or command connection, see the later sections.

1. Click left on the Connection icon (arrow).
2. Click left anywhere inside the origin block. An output connection menu displays, such as the example in Figure 32.

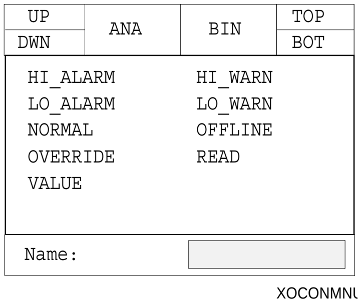


Figure 32: Example of an Output Connection Menu

3. Select the appropriate output connection name by clicking left on it. (Optionally, you may click left on ANA or BIN to filter by connection type.) The menu disappears.
4. Move the mouse and notice a line has formed.
5. Move the line out of the origin block. To add a line segment, click left. To erase a line segment, click right/middle.
6. Move the line anywhere inside the destination block and click left. The input connection menu displays.
7. From the menu, click left on the appropriate input connection name. (Optionally, you may click left on ANA or BIN to filter by connection type.) The two blocks connect.

Details

You may need to turn a line for it to reach the destination block, or to make a tidy connection. Figure 33 shows an example of two turns in the connection between the AHU1\OA-TEMP and ENRH1 blocks. To turn the line, simply click left anywhere in the work area while drawing the line. A new line segment forms. Make as many turns as needed to reach the destination block (14 segments maximum).

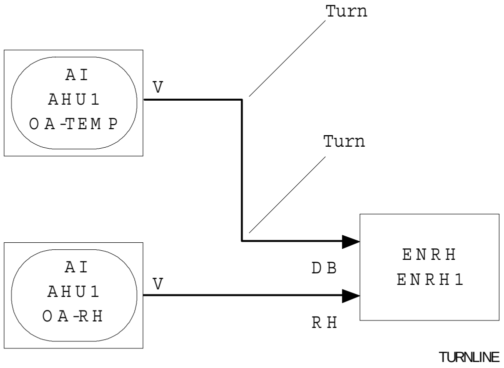


Figure 33: Turns in a Line to Reach Destination Block

Figure 34 shows additional examples of various methods for connecting blocks. Fan in is multiple outputs fed into one input; fan out is the same output going into multiple inputs.

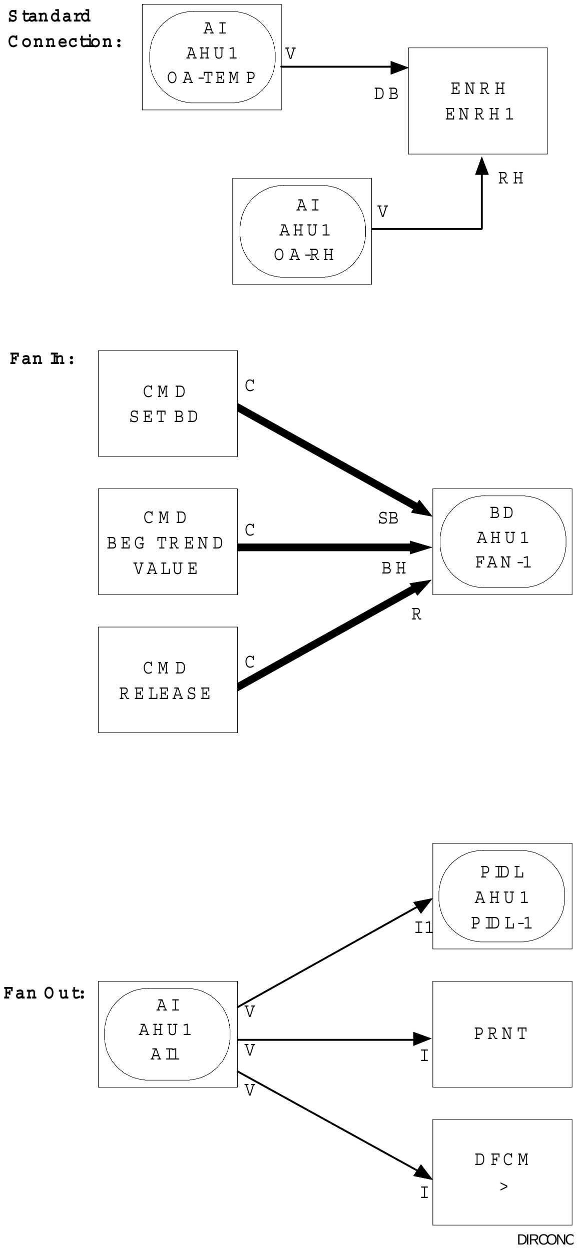
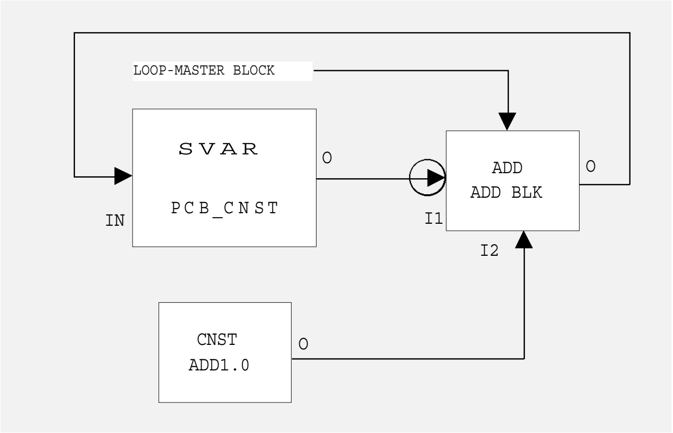


Figure 34: Direct Connection, Fan In Connection, and Fan Out Connection

When making connections, you may form a loop between two or more blocks. A loop is formed when two blocks are fed the inputs and outputs of each other (Figure 35). In this case, the Editor does not know which block to execute first. When the Editor detects a loop, the warning message `Loop detected! Select Loop-Master block` is displayed and the loop turns red. To complete a loop, you need to choose which function block GPL will execute first by clicking left on that block. This block is called the loop-master block. The end of the loop-master block connection will then change to an open circle.



FORMLOOP

Figure 35: Forming a Loop

Note: When the message `Loop detected! Select Loop-Master block` is displayed, the only items in the work area that are active include the function blocks that are in the loop, all compound blocks, and the Compound Name fields. If you click on an inactive item in the work area, nothing happens. If you click anywhere outside the work area, the last line drawn that caused the loop erases, canceling the loop.

If the loop-master block is on another diagram, you can go to that diagram in the normal manner and click left on the block. If the loop-master block has a remote connection as an input, the remote connection symbol is marked as the loop-master block.

Connecting Two Blocks in Separate Compounds

You can connect blocks that exist in different compounds but in the same file. The procedure is similar to connecting two blocks that are on the same screen. These steps presume that the origin block is in the current compound and the destination block is in another compound.

1. Click left on the Connection icon (arrow).
2. Double-click left on the compound that contains the origin block.
3. Click left anywhere inside the origin block. An output connection menu displays.
4. Select the appropriate output connection name by clicking left on it. (Optionally, you may click left on ANA or BIN to filter by connection type.) The menu disappears.
5. Move the mouse and notice a line forms. Move the line past the green edge of the work area and click left. This displays the diagram at the next-highest level.
6. Draw the line into the compound block that contains the destination block. Click left. The contents of that compound display.
7. Locate the destination block. Move the line anywhere inside the destination block and click left. The input connection menu displays.
8. From the menu, select the appropriate input connection name. (Optionally, you may click left on ANA or BIN to filter by connection type.) The two blocks connect.

Details

While drawing the line from the origin block, continue to click into lower level compounds until you reach the desired destination block. You can go down as many as ten compound levels.

The relative position that a line leaves the origin compound is where it is drawn coming out of the compound or compound block at the higher level. For example, if a line leaves the origin compound at the right side of the work area, it will appear on the left side of the destination compound at relatively the same vertical position (Figure 36).

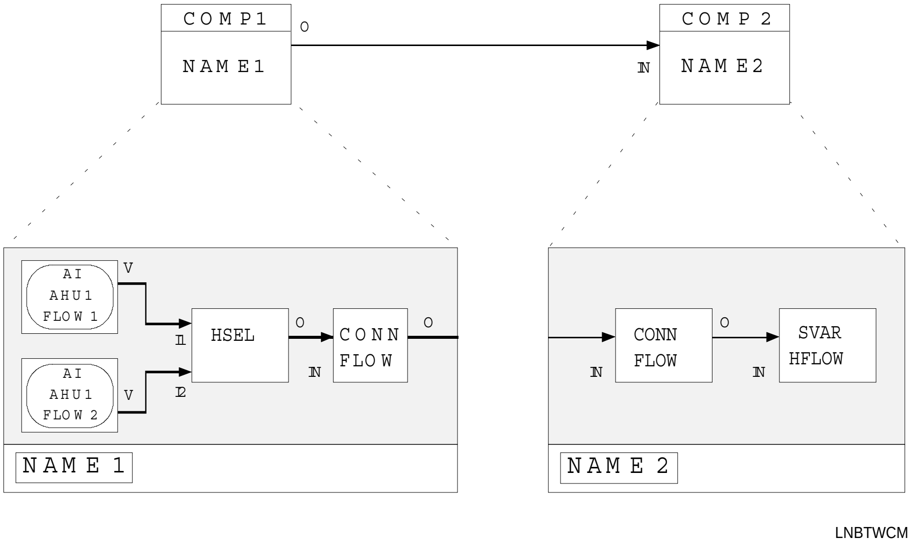


Figure 36: Relative Position of a Line Between Compounds

If you move a connection line that exits a compound headed for another compound, the other segment of the line inside the compound remains in its original position. It will not move when the other segment is moved.

Connecting Two Blocks Remotely

With this procedure, you can draw a remote connection between two blocks. The blocks you want to connect may be in the same diagram or in different diagrams, but they must be in the same file. Remote connections between separate files is not allowed.

Note: A block must be named for it to be connected remotely. That is because the name is required to identify the block in the remote connection menu. Object blocks can be either defined or undefined.

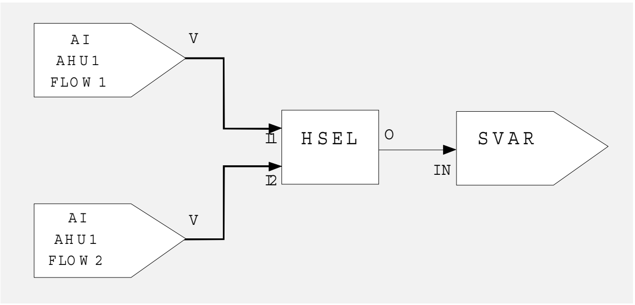
Introduction

The procedure for drawing a remote connection is different from drawing a direct connection. The primary difference is that you double-click anywhere on the screen to start the remote connection.

GPL has two types of remote connections: origin remote connection and destination remote connection. The origin connection has the origin block as its reference, in which the remote connection is an input to a destination block (Figure 37). The destination remote connection has the destination block as its reference, in which the remote connection is an output from an origin block (Figure 37).

Origin Remote Connections
(references origin block)

Destination Remote Connection
(references destination block)



INOUTRC

Figure 37: Origin and Destination Remote Connections

The steps in making an origin remote connection or a destination remote connection differ slightly. To make an origin remote connection, you double-click left in the work area, which displays the remote connection symbol. To make a destination remote connection, you first need to start the connection line from the origin block, then double-click left in the work area. The complete instructions for both types of remote connections are explained separately below.

The remote connection menu (Figure 38) displays while you are making the remote connection. It differs slightly from the direct connection menu.

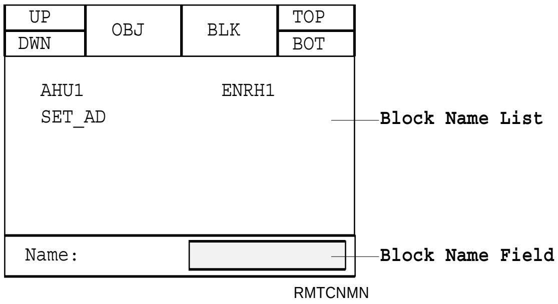


Figure 38: Remote Connection Menu

Menu Fields

As shown, the menu is composed of the following fields, which perform these operations:

UP

Pages the Block Name list back one page. If you click left on UP and the list does not change, you have reached the first page.

DWN

Pages the Block Name list forward one page. If you click left on DWN and the list does not change, you have reached the last page.

OBJ

Filters the Block Name list to show only system names that are defined in the file. Click left on OBJ to filter the list to show all system names. (OBJ back-highlights in red). Click left on OBJ again to list all system names, and operation and special block names (OBJ dehighlights).

BLK

Filters the Block Name list to show only operation and special block names. Click left on BLK to filter the list to show all operation and special block names (BLK back-highlights in red). Click left on it again to list all system names, and operation and special block names (BLK dehighlights).

TOP

Displays the first page (top) of the Block Name list. If you click left on TOP and the list does not change, you have reached the first page.

BOT

Displays the last page (bottom) of the Block Name list. If you click left on BOT and the list does not change, you have reached the last page.

Block Name List

Lists all system names, and the names of object, operation, and special blocks in the control strategy. Object names and blocks within protected compounds are not listed. A REF object block name may have appended a block label in blue to differentiate it from other REF blocks. Click left on a system name to change the list to show all object blocks under that system. Click left on an object name to change the list to show all connections under that object. Similarly, click left on an operation or special block to change the list to show all connections under that block.

The list is arranged alphabetically from left to right. The names in the list are color coded to distinguish their types.

Name	Color
System Name	Red
Object Block Name	Red
Operation Block Name	White
Special Block Name	White

Block Name Field

Field for typing in a valid system name or block name. As you type the name, the Editor filters the list to show only those entries beginning with the letters specified. Use the backspace key to erase a typed letter and refilter the list.

Procedure: Origin Remote Connection

Follow these steps to make an origin remote connection (steps for making a destination remote connection follow):

1. Click left on the Connection icon (arrow).
2. Double-click left in any empty space in the work area.
   A remote connection symbol appears.
3. If you need to resize the remote connection symbol, hold down the right/middle mouse button and drag the mouse.
4. Position the remote connection symbol and select the proper orientation, clicking the right/middle button to change orientation (Figure 39). If you click and the orientation does not change, move the symbol closer to the middle of the work area.

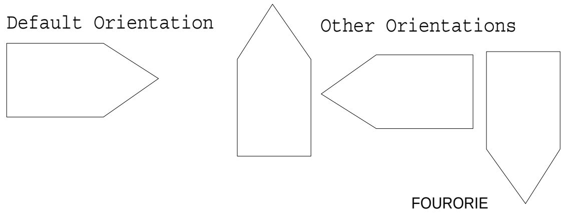


Figure 39: Four Orientations for
Remote Connection Symbol

5. With the remote connection symbol in the desired position, click left. The remote connection menu displays.
6. From the connection menu, locate the origin block.

   If the origin block is an object, click left on the desired system name in red. A list of all objects under that system displays. Select an object name and a list of all output connections displays.

   If the origin block is an operation or special block, click left on the block name in white. A list of all output connections displays.
7. Click left on an output connection name. Notice the remote connection is labeled.
8. Move the mouse and a line forms from the “V” side.
9. Route the line to anywhere inside the destination block and click left. The input connection menu displays.
10. Click left on a valid input connection name. The origin block connects to the destination block, via the remote connection.

Procedure: Destination Remote Connection

Follow these steps to make a destination remote connection:

1. Click left on the Connection icon (arrow).
2. Click left on the origin block. Select an output connection name. Slowly move the mouse outside the block and notice a line forms.
3. Move the line to a desired position from the block and double-click left. A remote connection symbol appears.
4. If you need to resize the remote connection symbol, hold down the right mouse button and drag the mouse.
5. If you need to change the orientation of the remote connection symbol, click the right/middle button. Figure 38 shows the different orientations. If you click and the orientation does not change, move the symbol closer to the middle of the work area.
6. With the remote connection symbol in the desired position, click left. The remote connection menu displays.
7. From the connection menu, locate the destination block.

   If the destination block is an object, click left on the desired system name in red. A list of all objects under that system displays. Select an object name and a list of all input connections displays.

   If the destination block is an operation or special block, click left on the block name in white. A list of all input connections displays.
8. Click left on an input connection name. The origin block connects to the destination block, via the remote connection.

Details

With the Move icon selected, you can move or resize a remote connection symbol just as you would any function block.

A remote connection can consist of up to seven line segments.

To reassign a remote connection, select the Connection icon, double-click left on the remote connection symbol, and select a different block. If you are reassigning an origin remote connection from a CMD, 2CMD, or WRIT block, it is best to erase the remote connection, then re-add it. Even though you may select a different origin block, the Editor does not update the line labels or verify that you made a valid selection for that object. Invalid commands and attributes are detected when the file is translated and compiled.

You can view and edit a template for the remote connection, which is actually the template of the referenced (i.e., origin or destination) block. Changing the template from the remote connection is equivalent to changing it from the referenced block.

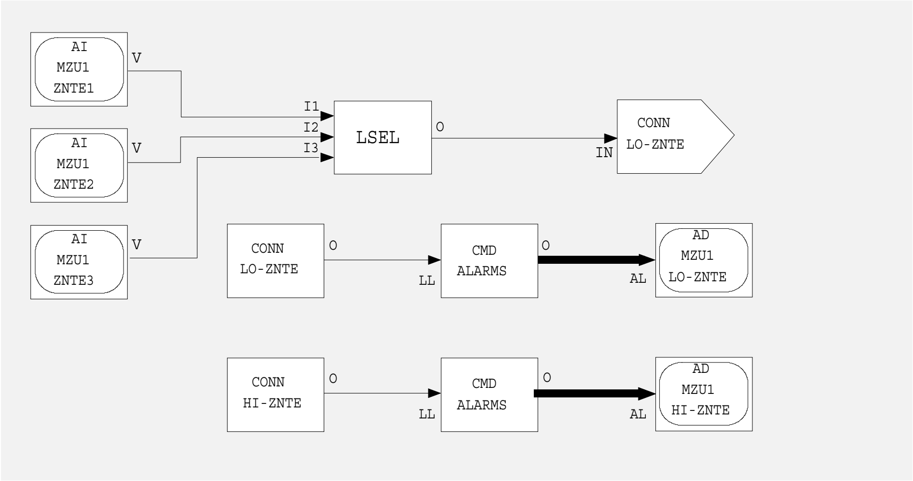
The size of the line label for the remote connection is based on the size of the referenced block, not on the size of the remote connection symbol. For example, if the referenced block has been enlarged, the line label for the remote connection will appear enlarged, even though the remote connection symbol is default size.

When a compound is created that contains a remote connection symbol, the remote connection symbol may become unassigned. This will happen only when the referenced block is not saved as part of the same compound. The remote connection does not lose its assignment if the referenced block is saved in the same compound. To reassign a remote connection, double-click left on its symbol.

If you erase a block that has a remote connection, the remote connection symbol is also erased. If you restore the block with the Undo function, the block and the remote connection symbol return just as before.

All blocks can accept a remote connection, provided there are connections available and the block is assigned a name. The same holds true for operation and special blocks, even though naming these blocks is optional. You must name an operation block for it to show up on the remote connection menu. Therefore, if you are planning to connect an operation block remotely, you must name it. In the remote connection menu, the 2CMD block shows the command name that represents CMD0.

If there are two blocks with the same name, both block names will display in the remote connection menu. Since you cannot determine which name goes with which block, we suggest that you assign unique names to all blocks. However, if there are two commands of the same type in the same file, you cannot do this. Instead, you need to place a Connection (CONN) block between the reference CMD block and the object block. Otherwise, there would be two command blocks with identical names on the same diagram. Figure 40 illustrates:



CONNCMD

Figure 40: CONN Block Used to Label Second CMD Block

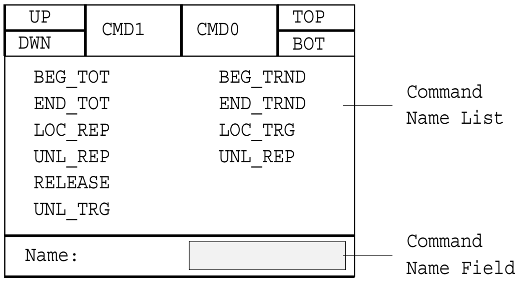
Commanding an Object

Objects are commanded in GPL by connecting a command block to an object block. Commands such as Start/Stop and Begin/End Totalization are examples. Objects, including process compounds, can accept a command line as an input. Refer to the next section, *Commanding a Process*, for instructions on how to connect to process compound blocks.

Introduction

Two command blocks are offered: CMD (single command) or 2CMD (dual command). The procedure for making a command connection involves selecting COMMAND or DUAL CMD on the output connection menu, and selecting the proper command or commands on the input connection menu.

The output connection menu for the CMD block is similar to a direct connection menu, but it displays command names, not connect names, and does not have the ANA and BIN options. However, the output connection menu for the 2CMD block is different in several ways. Figure 41 shows an example.



2CMDBLK

Figure 41: Output Connection Menu for the 2CMD Block

Menu Fields

The menu is composed of the following fields, which perform these operations:

UP

Pages the Command Name list back one page. If you click left on UP and the list does not change, you have reached the first page.

DWN

Pages the Connection Name list forward one page. If you click left on DWN and the list does not change, you have reached the last page.

CMD1

When this field is highlighted, the command you select will be the one sent when the Select Command input from the origin block is True (1).

CMD0

When this field is highlighted, the command you select will be the one sent when the Select Command input from the origin block is False (0).

TOP

Displays the first page (top) of the Command Name list. If you click left on TOP and the list does not change, you have reached the first page.

BOT

Displays the last page (bottom) of the Command Name list. If you click left on BOT and the list does not change, you have reached the last page.

Command Name List

Lists all possible command names for the block, arranged alphabetically from left to right. Click left on the command name to select it.

Command Name Field

Field for typing in a valid command name. As you type the name, the Editor filters the list to show only those entries beginning with the letters specified. Use the backspace key to erase a typed letter and refilter the list.

The command blocks are labeled according to the command name. Figure 42 illustrates:

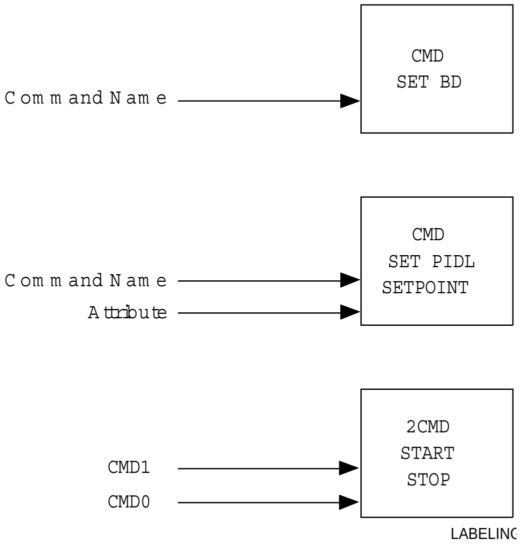


Figure 42: Various Labeling Techniques for Command Blocks

Procedure

Follow these steps to connect a command block to an object block:

1. Click left on the Connection icon (arrow).
2. Click left on the CMD or 2CMD block. The output connection menu displays.
3. If this is a CMD block, click left on COMMAND in the connection menu. If this a 2CMD block, click left on DUAL CMD. The menu disappears.

4. Move the mouse and notice a line has formed. Move the line out of the origin block. To add a line segment, click left. To erase a line segment, click right/middle.
5. Draw the line anywhere inside the object block (or a CONN block configured to accept a command), and click left. The input connection menu displays.
6. If the connection is from a CMD block, click left on the name of the command. The CMD block then connects to the object block or CONN block.

   If the connection is from a 2CMD block, two selections in the connection menu display: CMD1 and CMD0. You need to choose one command for each selection. On the CMD1 level (CMD1 is highlighted), click left on the command that will be sent when the Select Command input is True (1). On the CMD0 level (CMD0 is highlighted), click left on the command that will be sent when the Select Command input is False (0). The two blocks connect.

Details

GPL labels the command block for you, based on the command you selected. Some of the command names have underscores (e.g., SET_AD). The GPL Editor cannot display underscores on the face of the block, so a space is used in the name instead (e.g., SET AD). The GPL Translator will add underscores in the commands when the file is translated.

You cannot edit a CMD block to change the type of command (e.g., SET_AD to BEG_TOT). To change a command, you need to delete the command block, paste down a new one, and reconnect it.

Only those commands valid for the type of object selected are indicated in the input connection menu.

If you query a command block that is not selected, the template contains the statement `Command unselected—Connect block to object for command selection.`

If you bring a command line into a group compound, it will open immediately, so that you can terminate the line to an object block or to a properly configured CONN block.

Commanding a Process

You can command process objects (i.e., process compounds) with this procedure. The three possible commands are Enable Process, Disable Process, and Trigger Process. You can use either the CMD or 2CMD block.

1. Click left on the Connection icon (arrow).
2. Click left on the CMD or 2CMD block. The connection menu displays.
3. If this is a CMD block, click left on COMMAND in the connection menu. If this a 2CMD block, click left on DUAL CMD. The menu disappears.
4. Move the mouse and notice a line has formed. Move the line out of the origin block. To add a line segment, click left. To delete a line segment, click right/middle.
5. Draw the line anywhere inside the process compound block and click left. The input connection menu displays.
6. If the connection is from a CMD block, click left on a command, either PRC_ENA (enables process), PRC_DIS (disables process), or TRIGGER (triggers process). The CMD block then connects to the compound block.

   If the connection is from a 2CMD block, two levels in the connection menu display: CMD1 and CMD0. On the CMD1 level (CMD1 is highlighted), click left on the command that will be executed when the Select Command input is True (1). On the CMD0 level (CMD0 is highlighted), click left on the command that will be executed when the Select Command input is False (0). The two blocks connect.

Details

You must define the compound as a process for it to accept a command (i.e., you must configure the Compound Type field to PROCESS or RESTART PROCESS). Also, if you want to command an object inside a process, select OPEN CMPD (open compound). The compound will open so that you can terminate the connection to an object.

Exempting a Connection from Triggers

XMPT

With this procedure, you can make a binary triggerable connection line exempt from triggering a process. In some applications, you may need to exempt a binary line from triggering a process, since it may otherwise cause unnecessary or unwanted processing. For details, refer to the *Order of Process Execution* section in the *Graphic Programming* chapter.

1. Double-click left on the Connection icon (arrow).
2. Click left on the Exempt Connection icon (XMPT arrow).
3. Position the cursor over either end of or turn in a binary line. A yellow highlight box appears. Click left. An open square overlays the arrowhead. This indicates that the line is exempt.

Details

The exempt connection acts as a toggle switch, which means clicking left on either end or corner in the line alternates between exempt and non-exempt.

You may select for exemption all the binary triggerable attributes and binary shared variables for a process by answering Yes to `Exempt All?` in the compound template.

You can exempt from triggering a process only the triggerable attributes of objects and binary shared variables. Also, only those binary triggerable attributes that are listed on the standard connection menus are available for exemption. This means that you cannot exempt a triggerable attribute that is used with a READ block. In this case, you need a USER block to read and exempt the attribute.

If the exempted connection is part of a fanned-out connection, the other connections in the fan-out are also exempted, even though they are not marked as such. This is true only for fanned-out connections that go to the same process. For fanned-out connections between processes, the connection that is in a different process from the exempted connection will not be exempted.

Do not exempt a line entering a CONN (Connect) block if the CONN block has fanned out connections. Otherwise, the GPL Editor will not be able to determine which of the fanned out blocks is exempt.

Query Functions

This section explains how to perform the following query functions:

- View and edit a database template for an operation or special block.
- Read and modify an object in the archive database.
- Query a compound.
- Access a FILE block.
- View the macro file of a USER block.
- Query a connection.
- Find a function block.
- Replace system names in a strategy file.
- View the list file.
- Perform a Session Read.

The Query icon (question mark) provides these functions. Double-clicking on the Query icon displays an option menu (Figure 43).

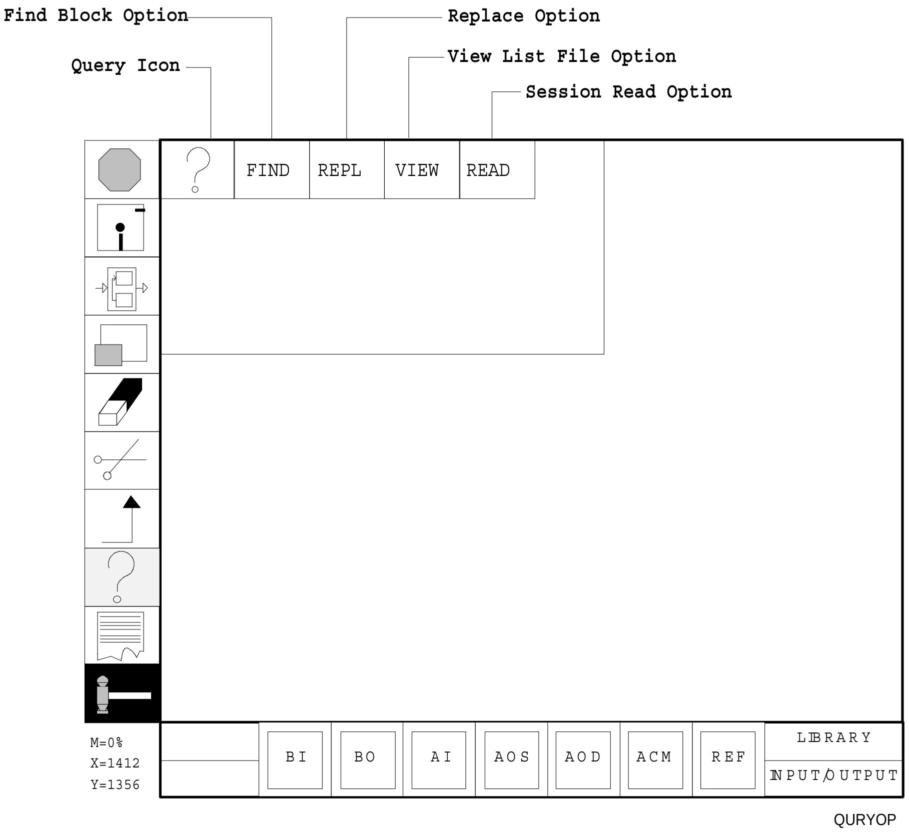


Figure 43: Query Option Menu

Note: The default operation of this icon is query.

All instructions in this section presume a diagram is currently displayed.

The following keys have specific functions when any template is displayed:

Key	Function
Arrow keys	Moves cursor between fields.
Backspace	Moves cursor back one space within a field, erasing entries.
CTRL + Arrow Keys	Moves cursor within a field one space at a time.
F10	Saves changes, accesses archive database, and closes template.
Left mouse button/Esc	Ignores changes and closes template.
Page Up	Displays previous page of template.
Page Down	Displays next page of template.

Viewing and Editing a Block Template

?

Each operation, special, and group compound block has a database that defines its name, parameter values, and other characteristics. The Editor represents these databases as templates that display in the work area. Use this procedure to view and edit database templates.

1. Click left on the Query icon (question mark).
2. Click left anywhere inside a function block. A template displays in the work area. Use the Page Up and Page Down keys to scroll through templates with multiple pages.
3. After you are done viewing the template, press the Esc key or click left to exit. If you want to make changes, go to the next step.
4. Make the changes to the parameters as needed. Each is defined in the *Template Field Descriptions* section of the *Function Blocks* chapter.
5. To save the changes and exit the template, press F10. Or, to ignore the changes you made, press Esc or click left.

Details

You may assign the same name to two operation or two special blocks in the same file, or assign them no name (blank). The operation blocks without an assigned name cannot be connected remotely, and will not show on the Find menu.

The Editor checks to make sure you enter proper values into the parameter fields. It checks for valid data types, proper ranges, field interactions, and duplicate object names. Refer to the *Function Blocks* section in the *Graphic Programming* chapter for details on template error checking. For an explanation of a user message, refer to the *User Messages* section at the end of this chapter.

Some parameters in a database template may also be available through input connections. If the parameter is specified in the template and is connected as an input, the value of the input, not of the parameter field, will be used.

Refer to the previous section, *Query Functions*, for a table listing the keys that have specific template functions.

Reading and Modifying an Object

Each object and process compound block has a database that defines its name, attribute values, and other characteristics. This data is stored in the strategy file and the archive database. The Editor represents these databases as templates that display in the work area. Use this procedure to read and modify the templates (i.e., objects).

1. Click left on the Query icon (question mark).
2. Click left anywhere inside the object or process block. An archive database interaction message may display. This indicates that the Editor is reading the template information from the archive database into the strategy file database. (If you get a message at this point, refer to the *User Messages* section in this chapter.)
3. Use the Page Up and Page Down keys to scroll through templates with multiple pages.
4. After you are done reading the template, press the Esc key or click left to exit. If you want to make changes, go to the next step.

5. Make the changes to the attributes and parameters as needed. Each is defined in the *Template Field Descriptions* section of the *Function Blocks* chapter.

6. To save the changes and exit the template, press F10. An archive database interaction message may display. (Then, if an error message displays, refer to the *User Messages* section.) Or, to ignore the changes you made, press Esc or click left. The archive will not be accessed.

Details

You cannot assign the same software system\object name to two object blocks that are in the same file. When an object or process compound block is defined, its color changes from magenta to brown.

The Editor checks to make sure you enter proper values into the parameter fields. It checks for valid data types, proper ranges, field interactions, and duplicate object names. Refer to the *Function Blocks* section in the *Graphic Programming* chapter for details on template error checking. For an explanation of a user message, refer to the *User Messages* section at the end of this chapter.

Some attributes in a database template may also be available through input connections. If the attribute is specified in the template and is connected as an input, the value of the input, not of the parameter field, will be used.

Querying a Compound?

?

This function displays the contents of a compound block.

1. Click left on the Query icon (question mark).
2. Position the cursor anywhere inside the compound block.
3. Double-click left. The contents of the compound display.

Details

The names of the currently active compound levels are shown on the bottom of the work area, enclosed in rectangular boxes (Figure 44). These boxes are called Compound Name fields, which contain the system\object name for a process or Restart process compound, or the compound name for a group compound. The work area can only show up to four levels in the compound name fields. Higher levels are hidden from view, but return as you move toward the highest compound level.

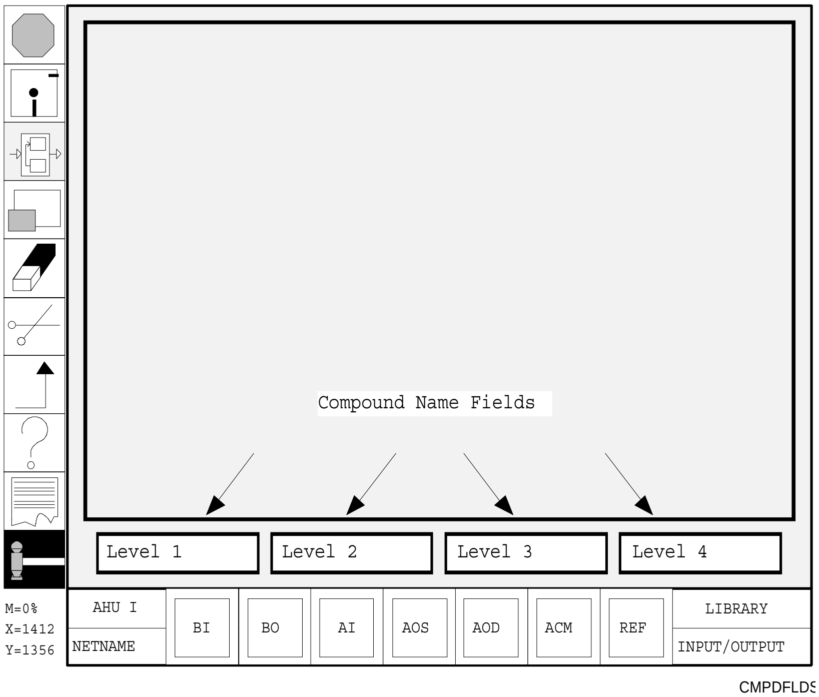


Figure 44: Compound Name Fields

The Compound Name fields have two purposes: to indicate the level of the currently displayed compound, and to provide for a mechanism of returning to the higher levels. To return to a higher level, click left on its name in the field.

To show the data template for a compound, click left on its block.

Accessing a FILE Block

?

This function accesses the contents of a FILE block. For details on the FILE block, refer to the *Function Blocks* chapter.

1. Click left on the Query icon (question mark).
2. Double-click left on the FILE block that you wish to open. The contents of the external file are loaded and the FILE Block icon (arrow) displays in the upper right corner of the work area.
3. After you are finished with viewing and editing the external file, return to the parent file by clicking left on the File Block icon.

Details

The file that you want to access through the FILE block must be in the current directory. Also, to return to the parent file, the parent file must be in the same directory.

You may penetrate up to 30 levels of FILE blocks.

Viewing a USER Block Macro File

?

This function displays the macro file for a USER block. For details on how to write a USER block macro file, refer to the *Function Blocks* chapter.

1. Click left on the Query icon (question mark).
2. Double-click left on the USER block. The macro file for the block displays.
3. Use the Page Up and Page Down keys to scroll through the file. After you are finished viewing, click left or Esc to display the previous contents of the work area.

Details

The SET GPLUMAC statement in the GPL.BAT file determines under which particular directory all USER block macro files should reside. The Editor searches for the file in this directory when you double-click left on the USER block. If you wish, you may change the directory by editing the GPL.BAT file with a text editor. Use the DOS conventions for editing a batch file.

You cannot edit or print the macro file with the Editor.

Querying a Connection

This function displays a brief description of a connection. It is especially helpful for a connection that spreads across compounds.

1. Click left on the Query icon (question mark).
2. Position the cursor on either end of or turn in the connection you want to query. Notice a yellow highlighting box appears.
3. Click left. A description displays on the bottom of the work area. Click left again to clear the description.

Details

The query connection message describes the connection types between two function blocks. It follows this format:

[Connection] Type: [Origin Block]—[Connection Name] to [Destination Block]—[Connection Name]

where:

[Connection]: Analog, Binary, Time, Cmd (Command or Dual Command), Read, Write, or Control.

[Origin Block] or [Destination Block]: For object and process compound blocks, the system\object name; for operation and special blocks, the block name.

[Connection Name]: name of the connection as displayed in the input or output connection menu.

Here are two examples:

```
Analog:     AHU1\OA-TEMP-VALUE to
            ENRH1-DRY BULB

Cmd Type:   SET AD-COMMAND to
            AHU1\OA-ENRH-SET AD
```

Finding a Function Block

This feature is similar in concept to a Find feature of a word processor. It searches the file to locate each instance of a block name. When an instance is found, it displays the diagram that contains the block, and highlights the block in yellow.

FIND

Introduction

The Find menu (Figure 45) displays on the work area when you click left on the Find icon. It contains all system\object names and block names that are used in the currently loaded control strategy file.

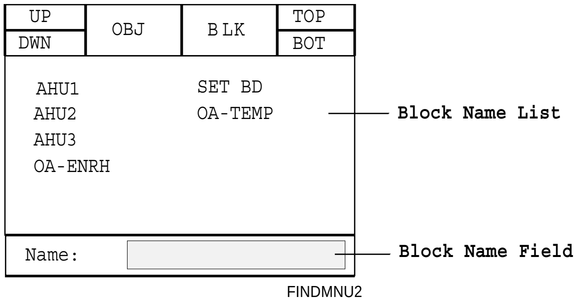


Figure 45: Find Menu

Menu Fields

As shown, the menu is composed of the following fields, which perform these operations:

UP

Pages the Block Name list back one page. If you click left on UP and the list does not change, you have reached the first page.

DWN

Pages the Block Name list forward one page. If you click left on DWN and the list does not change, you have reached the last page.

OBJ

Filters the Block Name list to show only system names that are defined in the file. Click left on OBJ to sort the list to show all system names. (OBJ back-highlights in red). Click left on OBJ again to list all system names, and operation and special block names (OBJ dehighlights).

BLK

Filters the Block Name list to show only operation and special block names that are defined in the file. Click left on BLK to sort the list to show all operation and special block names (BLK back-highlights in red). Click left on BLK again to list all system names, and operation and special block names (BLK dehighlights).

TOP

Displays the first page (top) of the Block Name list. If you click left on TOP and the list does not change, you have reached the first page.

BOT

Displays the last page (bottom) of the Block Name list. If you click left on BOT and the list does not change, you have reached the last page.

Block Name List

Lists all defined and undefined object and process compound blocks, and all named operation, special, and group compound blocks. However, the menu does not show block names that are inside protected compounds. Also, if the Find menu references a 2CMD block, the command associated with CMD0 is shown only, not CMD1. A REF object block name, if a duplicate of another REF block, is appended with a unique block label in blue for identification purposes. All block names are arranged alphabetically from left to right.

To find an object or process compound block, click left on its system name, then its object name. To find an operation, special, or group compound block, click left on its block name. To distinguish between the block classes, these colors are used in the Find menu:

Name	Color
System Name	Red
Object Name	Red
Operation Block Name	White
Special Block Name	White
Group Compound Block Name	White

Block Name Field

Field for typing in a valid block name. As you type the name, the Editor filters the list to show only those entries beginning with the letters specified. Use the backspace key to erase a typed letter and resort the list. The field is not case sensitive.

Procedure

Follow these steps to use the Find feature:

1. Double-click left on the Query icon (question mark).
2. Click left on the FIND option. The Find menu displays like the one in Figure 46.

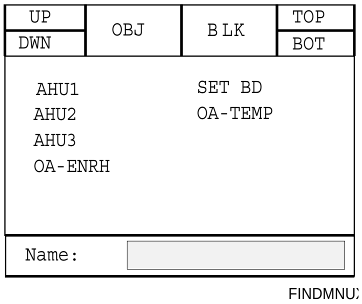


Figure 46: Example of a Find Menu

At this point, the instructions vary depending on if you want to locate an object, process compound, operation, special, or group compound block.

3. To locate an object or process compound block, click left on the system name under which this object resides. Then the objects under that system display. Locate the object name in the list and click left on it. The Editor will move to the first diagram that contains this object block, and highlight the block's border in yellow. The message `Find Next Occurrence?` displays. Go to Step 4.

   To locate an operation, special, or group compound block, locate the block name in the list and click left on its name. The Editor will move to the first diagram that contains this block, and highlight the block's border in yellow. The message `Find Next Occurrence?` displays.

4. Click left on [OK] to find the next occurrence or click left on [CANCEL] to exit the find operation.

Details

A block must be named in the database template for it to show on the list. In the Find menu, the 2CMD block shows the command that represents CMD0.

No special message is given when no occurrences are found.

If a strategy has two blocks with the same block name, the first one the Editor will find will be the block that was pasted down first.

To refresh the work area from showing the highlighted block, click left in an open area of the screen. If you delete a highlighted block, the yellow outline remains on the screen. To clear it, simply refresh the work area.

When you are editing a compound, only those blocks in the immediate compound level are represented in the Find menu.

If a diagram is zoomed when you start the find operation, zoom cancels when the reference is found.

Replacing System Names

REPL

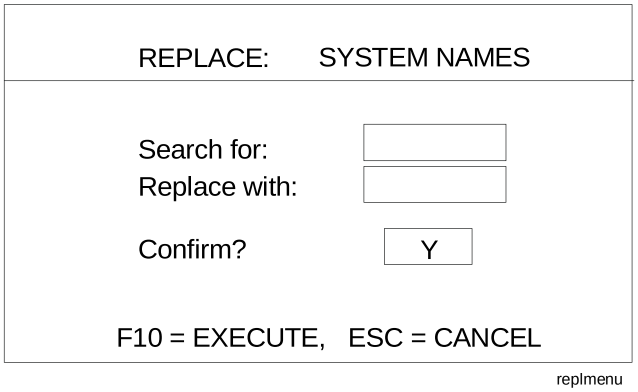
This function replaces system names of the selected blocks in a strategy file. The purpose of this function is to allow you to replace multiple system names in a strategy file in a single operation.

To use the Replace:

1. Double-click left on the Query icon (question mark).
2. Click left on the REPL option. A message appears telling you to use the white enclosing box to select blocks to include in the Replace.
3. Click left to continue. The white enclosing box appears.
4. Use the white enclosing box to select the blocks you want to include in the Replace. The Replace will be performed on all object blocks within the enclosing box, and all object blocks in compounds within the enclosing box.

   To shrink and enlarge the enclosing box, hold down the right/middle button and drag the mouse. Click left when the blocks are selected.

   Once the blocks are selected, the Replace menu shown in Figure 47 displays.



REPLACE: SYSTEM NAMES

Search for: □

Replace with: □

Confirm? Y

F10 = EXECUTE, ESC = CANCEL

replmenu

Figure 47: Replace Menu

To move between the fields in the Replace menu, use the up and down arrow or Enter keys.

5. Type the system name you want to change in the Search for field. This name must match the system name exactly (no wildcard characters). However, the search is case-insensitive.

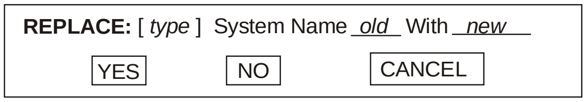
6. Type the new system name in the Replace with field. The name will be replaced exactly as you type it (including whether the letters are lower or upper case).

7. Select whether you want to confirm each Replace by typing y (yes) or n (no) in the Confirm ? field. Yes is the default.

8. To execute the Replace, press F10. Or, to Cancel, press Esc.

   If you selected to not confirm each Replace, all the system names in the selected blocks that match the system name being searched for will be automatically changed.

   If you selected to confirm each Replace, the Confirm message box shown in Figure 48 appears as each system name to be replaced is found. The Confirm box includes the type of system name to be replaced (e.g., software, hardware, associated input, feedback, reference), and both the old and new system names.



REPLACE: [*type*] System Name _*old*_ With _*new*_

Replconf

Figure 48: Confirm Message Box

To replace the system name, click YES.

Or, to skip the replace for this field, click NO. Or, to cancel the Replace process, click CANCEL.

When the Replace is complete, a message appears telling you how many system names were replaced. Click left to clear the message from the work area.

Details

The Replace will be performed on all object blocks within an area you select, and on all object blocks within compounds in the selected area. The Replace will not be performed on subfiles (when using FILE blocks).

The Replace function searches for all system names in object templates, including Process object system names, feedback, associated input, PIDL inputs, etc.

In Compound Edit mode, the Replace function replaces system names for only those objects that are in the compound. It does not change the compound system name itself.

Once a system name has been replaced in an object template of a defined object, the block's status changes to undefined. To define objects with replaced system names, and to synchronize the strategy file with the archive database, you have two options:

- either open each undefined object block and press F10 to save the block to the archive database
- or use the Session Read function, which reads all object blocks in the file at one time (the more efficient option)

Note: Remember that in a Session Read, the Read looks only for a match between the system\object name in the strategy file and the system\object name in the archive database. When it finds a match, data in the archive database for the object overwrites all data for the object in the strategy file.

The Replace function is useful in these situations:

- Use Replace to simply change system names in a file or a portion of a file in case of a mistake or reconsideration of what the names should be.
- Use Replace for a facility that has multiple applications that operate identically (e.g., multiple air handling units or chillers). Create a GPL program for one of these applications, and copy the program for each additional application. Then use the Replace function to provide unique system names for the objects in the duplicated files.

These two procedures are described in the following section.

Changing System Names

Use this procedure to change system names in the archive database and in the corresponding GPL strategy file, and to synchronize the file with the archive database.

1. Change the appropriate system names in the DDL source file using a text editor.

   If the system name is new, add it to the GLOBAL.DDL file.

2. Recompile the DDL files.
3. Change the system names in the GPL strategy file using the GPL Replace function.
4. Perform a Session Read on the GPL strategy file to synchronize the file with the archive database.

Copying a GPL Strategy

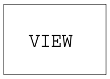
Use this procedure to copy a portion of the archive database and the corresponding GPL strategy file, change system names in the copied file, and synchronize the GPL file with the archive database.

1. Make a copy of the DDL source code that corresponds to the GPL strategy file you intend to copy.
2. Change the appropriate system names in the DDL source code using a text editor.

   If the system name is new, add it to the GLOBAL.DDL file.

3. Recompile the DDL.
4. Make a copy of the GPL strategy file that corresponds to the copied DDL source code file.
5. Change the system names in the copied GPL file using the Replace function.
6. Perform a Session Read on the GPL strategy file to synchronize the file and the archive database.

Viewing the List File


VIEW

The list file is a text file that the Session Read, Expert Checker, Translator, and Compiler use to record errors. This procedure allows you to view that file.

1. Double-click left on the Query icon (question mark).
2. Click left on the VIEW option. The list file displays. Use the Page Up and Page Down keys to move through the listing.
3. Click left or press Esc to clear the file from the work area.

Details

With the GPL Editor, you can view the first 30 pages of the list file. If the file is more than 30 pages, you need to use a text editor outside of GPL to view the subsequent pages.

The list file has the same file name and is placed in the same directory as the control strategy that was updated with Session Read, expert checked, or translated. It has a .LST extension. Also, the list file is not created until one of the GPL utilities runs.

The list file cannot be edited, deleted, or printed with the GPL Editor. Use a text editor outside of GPL to do these functions.

Performing a Session Read

This function performs a Session Read, which updates the data of all defined and undefined objects in a strategy file with the data for these objects in the current archive database. The entire strategy file is updated in one step, which is much more efficient and convenient than updating each block individually.

1. Double-click left on the Query icon (question mark).
2. Click left on the READ option. The message `Session Read In Progress` displays in the middle of the work area. When done, a `Session Read Complete` message appears. (This message is described under *Details*.)
3. Press any key to return to the Editor screen.

Details

The Read function synchronizes objects (defined and undefined) in the control strategy with the objects in the archive database. The archive database is the “master” because its data overwrites the corresponding data in the strategy file.

All warnings and errors that are found during processing go to the list file. You can view this list file by clicking left on the VIEW option in the Query option menu. (See *Viewing the List File*, earlier in this section, for more information.)

Each object block that caused an error is set to undefined, and the template information for the block is not updated with the archive data. Refer to the *User Messages* section for descriptions of all Session Read error messages.

If no errors are found during processing, the object block databases in the strategy file are updated to reflect those in the archive database.

All data that is updated during processing is not saved to the file until you save the strategy to disk.

The Session Read Complete message box displays the following information:

- how many defined blocks were successfully read
- how many defined blocks had errors or warnings, and therefore couldn’t be read
- how many defined blocks didn’t need to be read because they had been recently read or queried
- how many undefined blocks were successfully read
- how many undefined blocks had errors or warnings, and therefore couldn’t be read
- how many undefined blocks could not be read because they lacked a system or object name

Note: Session Read does not add blocks to the GPL strategy; if an object is in the archive database but is not in the GPL file, the Read will not add the block to the GPL file.

Print Functions

This section explains the two Print menus, the basics of printing, and how to perform these print functions:

- Page the print queue.
- Create print files.
- Send print files to the printer.
- Delete print files from the queue.

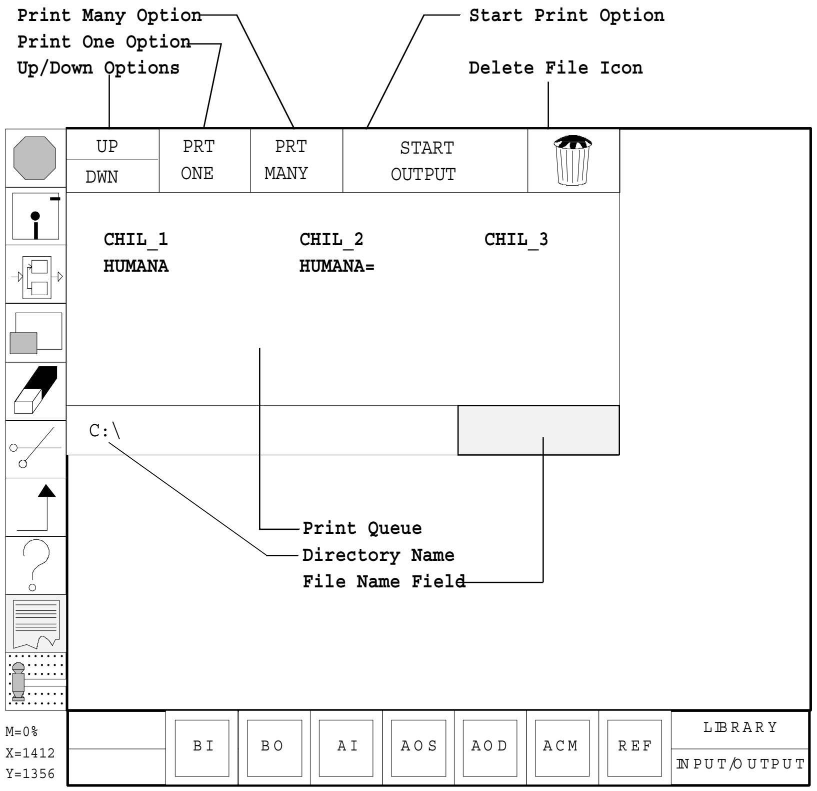
The Print icon provides these functions.

Using the Print Option Menu and Submenu

The Print icon uses two menus: an option menu and a submenu.

Print Option Menu

Clicking left on the Print icon displays an option menu (Figure 49).

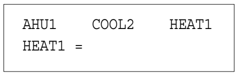


RDVIEW

Figure 49: Print Option Menu

As Figure 49 indicates, the Print option menu features a print queue, Directory Name field, and File Name field. The following paragraphs describe them in detail.

Print Queue



```
AHU1    COOL2    HEAT1
HEAT1 =
```

This is a list of all files ready for printing. The print files are listed in alphabetical order from left to right. A maximum of 12 files can be listed per page. Templates have an = character after their names to distinguish template print files from diagram print files. The name of a print file is the same as the name of the strategy file or compound block it represents.

Note: The Editor erases the temporary print file after it sends it to the printer. Also, you can manually erase a print file with the Delete File icon. The print file remains in the queue even if you exit GPL. This means that you can queue up several print files, exit GPL, then sometime later, re-enter GPL to print the files.

Directory Name Field

```
C:\
```

This field shows the printing directory, which is preset at C:\{FMSPATH}\BIN\GPL. All temporary print files will go to this directory. You cannot change it to send print files to a different directory.

File Name Field

This space is for typing in a print file name; up to eight alphanumeric characters are allowed. While you type in a name, the Editor filters the queue to show only those files beginning with the letters specified. For example, if you type in “S,” the list changes to show every file that starts with the letter S. Then, if you type in “E” after the S, every file that begins with SE is displayed, and so on.

Note: An alternative to typing in a print file name is to click left on its name in the queue. The Editor then places the selected name in the File Name field.

To type text in the File Name field, you must have the cursor inside the Print option menu. The field is not case sensitive.

Print Submenu

The Print function has a submenu that displays after you click left on PRT ONE or PRT MANY (Figure 50).

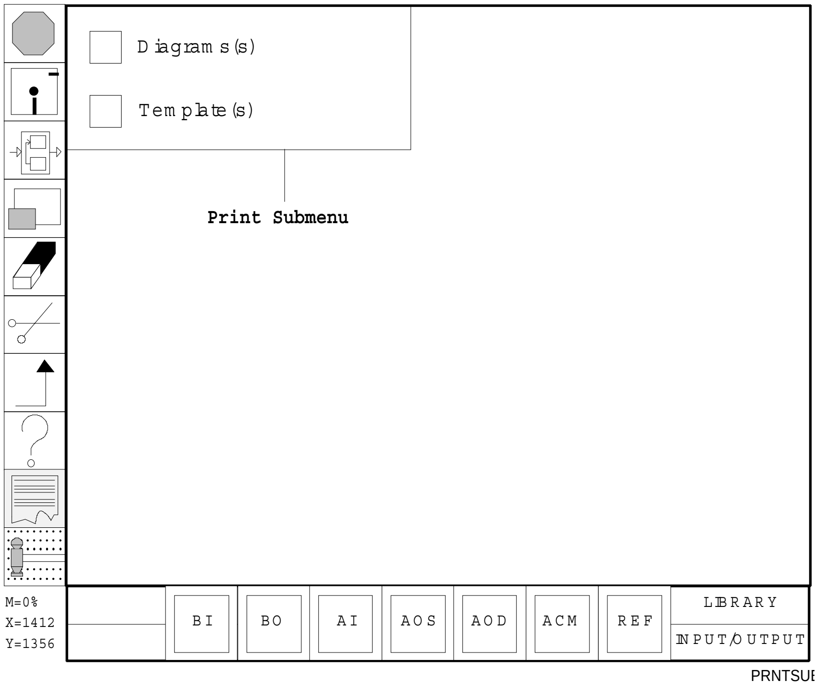


Figure 50: Print Submenu

The submenu offers two options:

Diagram(s)

Select if you want to send one or more diagrams to the print queue.

Template(s)

Select if you want to send one or more block templates to the print queue.

To cancel the Print operation with the Print submenu displayed, click left outside the submenu.

Learning the Basics of Printing

The GPL printing process involves the following steps:

- Display the diagram that you want to print.
- Display the Print option menu.
- Select PRT ONE or PRT MANY.
- Select Diagram(s) or Template(s).
- Click left on Start Output.

All printouts are labeled with the print file name. They are also labeled with the date on which they were sent to the print queue, not when they were actually printed. For diagrams, the file name and date appear in the lower left corner (Figure 52). For templates, this information is in the upper left corner (Figure 53).

Foreign language characters as part of text in a control strategy will print only if your printer has the capability to do so.

Depending on the characters used, printed text may be slightly shorter in length than on the Editor screen.

The GPL Editor prints diagrams and templates in a specific order. The order is as follows:

1. The highest level of a control strategy.
2. All compounds nested below the compound on the highest level that was pasted down last.
3. All compounds nested below the compound on the next lowest level that was pasted down, and so forth.

Figure 51 is a sample showing the order in which each compound of an entire file might be printed. Compounds that are labeled with the same number are on the same screen. The letters after the numbers indicate the order of printing within the compound.

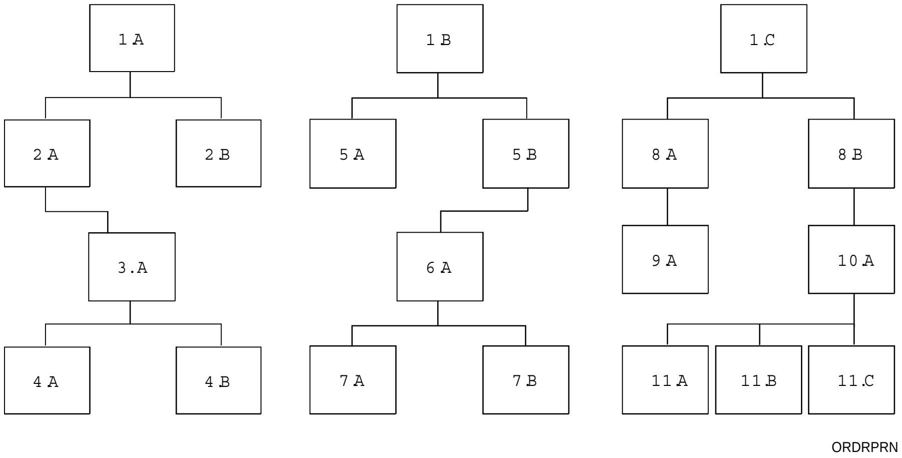
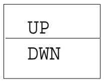


Figure 51: Order of Printing

Paging the Print Queue



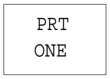
You can page up and down the print queue in the Print option menu to show other entries. The UP option pages the queue back one page; the DWN option pages the queue forward one page.

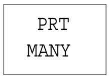
- To page the queue back one page, click left on the UP option.
- To page the queue forward one page, click left on the DWN option.

Details

If you click left on UP or DWN and the list does not change, you have reached the first or last page, respectively.

Creating a Diagram Print File





This function creates a print file for one or more diagrams in a control strategy. Select PRT ONE to create a print file for the currently displayed diagram. Select PRT MANY to create a print file for the currently displayed diagram and all nested diagrams.

The print file is placed on the print queue, which can be sent to a printer. Each diagram print file has the same name as the strategy or compound file name, with a .PRD (print diagram) extension. Even though a strategy may contain many diagrams, only one print file containing all diagrams is created.

1. Load the strategy containing the diagrams that you want to print. If you want to print multiple diagrams under one strategy, display the diagram at the level from which you want to start the printing.
2. Click left on the Print icon (page). The option menu displays.
3. If you are printing a single diagram, click left on the PRT ONE option. If you are printing multiple diagrams, click left on PRT MANY. A submenu appears.
4. Click left inside the checkbox next to Diagram(s). Each diagram will now flash on the entire screen momentarily as it is sent to the print queue. When all diagrams are sent to the queue, the print option menu returns.
5. You are now ready to send the diagrams to the printer. Refer to the section called *Printing Diagrams and Templates*.

Details

The print file name used is the same as the control strategy or compound file name. If the strategy has no name defined, the print file name will be “DEFAULT.”

By clicking left on PRT MANY, you can create a print file for all the diagrams that are under a particular control strategy. However, if the strategy contains a FILE block, a print file is not created for the diagrams that are under the FILE block. To create a print file for these, you must first access the external control strategy file by double-clicking left on the FILE block.

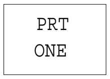
A diagram that is zoomed will print zoomed. If you are printing multiple levels, the level that is zoomed will be printed as zoomed, but all other levels will print regular size.

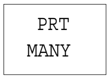
All the lower levels of the compounds that are on the screen when zoomed will be printed.

Two print files cannot share the same name. If you attempt to print files to the queue that have the same names, a file exists error message will display.

The create print file operation is continuous. You cannot pause it. The only way to halt or cancel the operation once it begins is to exit GPL and answer Y (yes) to the printing in progress warning message.

Creating a Template Print File





This function creates a print file for the database templates in a control strategy. Select PRT ONE to create a print file for the currently displayed diagram. Select PRT MANY to create a print file for the currently displayed diagram and all nested diagrams.

The print file is placed on the print queue, which can be sent to a printer. Each template print file has the same name as the strategy or compound file name, with a .PRT (print template) extension. Even though a strategy may contain many diagrams, only one print file containing all templates is created.

1. Load the strategy containing the templates that you want to print. If you want to print multiple templates under one strategy, display the diagram at the level from which you want to start the printing.
2. Click left on the Print icon (page). The option menu displays.
3. If you are printing the templates for a single diagram, click left on the PRT ONE option. If you are printing templates for multiple diagrams, click left on PRT MANY. A submenu appears.
4. Click left inside the checkbox next to Template(s). When all templates are sent to the queue, the print option menu returns.
5. You are now ready to send the templates to the printer. Refer to the section called *Printing Diagrams and Templates*.

Details

In the print queue, template print files have an = character at the end of them to distinguish them from diagram print files.

By clicking left on PRT ONE, you create a print file that contains templates of all blocks on the displayed diagram. The GPL Editor cannot selectively print the template of an individual block.

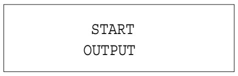
The print file name used is the same as the control strategy or compound name. If the strategy has no name defined, the print file name will be “DEFAULT=.”

By clicking left on PRT MANY, you can create a print file for all the templates that are under a particular control strategy. However, if the strategy contains a FILE block, a print file is not created for the templates that are part of the external file. To create a print file for these, you must first access the external control strategy file by double-clicking left on the FILE block.

Two print files cannot share the same name. If you attempt to print files to the queue that have the same names, a file exists error message will display.

The create print file operation is continuous. You cannot pause it. The only way to halt or cancel the operation once it begins is to exit GPL and answer Y (yes) to the printing in progress warning message.

Printing Diagrams and Templates



Use this procedure to print all the diagrams and templates that are in the print queue. You cannot select certain items in the queue that you want to print. Figures 52 and 53 are examples.

1. Double-click left on the Print icon (page).
2. Click left on Start Output. The option menu closes.

Details

If Enable Windows Printing was set to Yes when GPL was configured with SYSGEN, a printer dialog box will display. To begin printing, select any printer installed on your PC and click OK. After printing, reactivate GPL from the taskbar.

If Enable Windows Printing was set to No when the GPL was configured, printing will begin immediately after selecting Start Output.

Note: GPL automatically prints diagrams with landscape orientation. Selecting landscape again under printer properties will result in incorrect printing.

As diagrams and templates are being printed, you are free to perform other GPL functions. However, when the Print function accesses the disk drive, the cursor may freeze temporarily. Also, some functions may temporarily halt the printing. This is normal, and printing will resume when that action is completed.

After a diagram or template is printed, the GPL Editor deletes its print file.

If you exit the GPL Editor while a diagram or template is printing, the printout will stop immediately. The diagram and template that were not completely printed, and all others that did not start printing, remain in the print queue.

As diagrams and templates are being printed, you can create additional print files, either from the same strategy or from a different one. Note that a new print file added to the queue will start printing even if you don’t manually start it by clicking left on Start Output. That is because the Editor continually scans the print queue, and will start the new entries before you have a chance to start them manually.

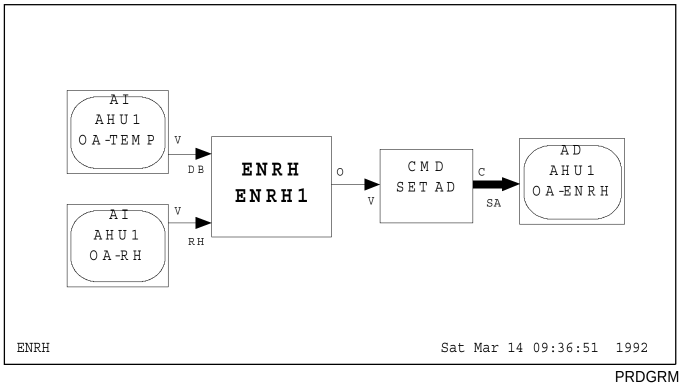
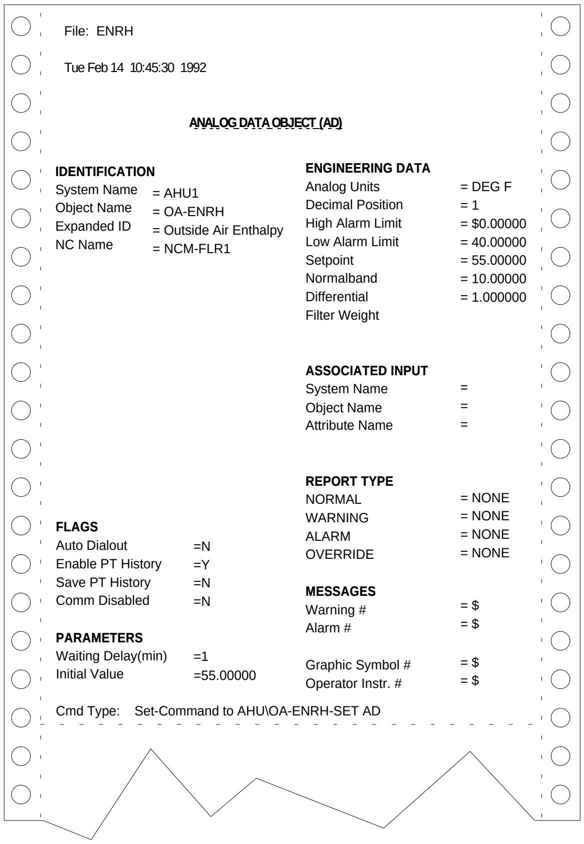


Figure 52: Example of a Printed Diagram



File: ENRH

Tue Feb 14 10:45:30 1992

ANALOG DATA OBJECT (AD)

IDENTIFICATION

System Name	= AHU1
Object Name	= OA-ENRH
Expanded ID	= Outside Air Enthalpy
NC Name	= NCM-FLR1

ENGINEERING DATA

Analog Units	= DEG F
Decimal Position	= 1
High Alarm Limit	= $0.00000
Low Alarm Limit	= 40.00000
Setpoint	= 55.00000
Normalband	= 10.00000
Differential	= 1.000000
Filter Weight	

ASSOCIATED INPUT

System Name	=
Object Name	=
Attribute Name	=

REPORT TYPE

NORMAL	= NONE
WARNING	= NONE
ALARM	= NONE
OVERRIDE	= NONE

FLAGS

Auto Dialout	=N
Enable PT History	=Y
Save PT History	=N
Comm Disabled	=N

MESSAGES

Warning #	= $
Alarm #	= $
Graphic Symbol #	= $
Operator Instr. #	= $

PARAMETERS

Waiting Delay(min)	=1
Initial Value	=55.00000

Cmd Type: Set-Command to AHU\OA-ENRH-SET AD

XPRTTM

Figure 53: Example of a Printed Template

Deleting a Print File

Use this procedure to delete a diagram or template print file from the print queue.

1. Click left on the Print icon (page).
2. Select the print file that you want to delete by clicking left on its name in the queue. (Or enter its name in the File Name field.) Template files are identified with the = symbol.
3. Click left on the Delete File icon (trash can). The item is removed from the queue and from disk.

Details

If the print file is erased while it is currently printing, the printer stops and moves to the top of the form. You will then have to begin the printing operation again by clicking left on Start Output.

Tools Functions

This section explains how to perform the following functions:

- Turn the grid on and off.
- Type text in the work area.
- Paste down an analog display.
- Run the Expert Checker, Simulator, and Translator.

The Tools icon (hammer) provides these functions. Double-clicking on the Tools icon displays an option menu (Figure 54).

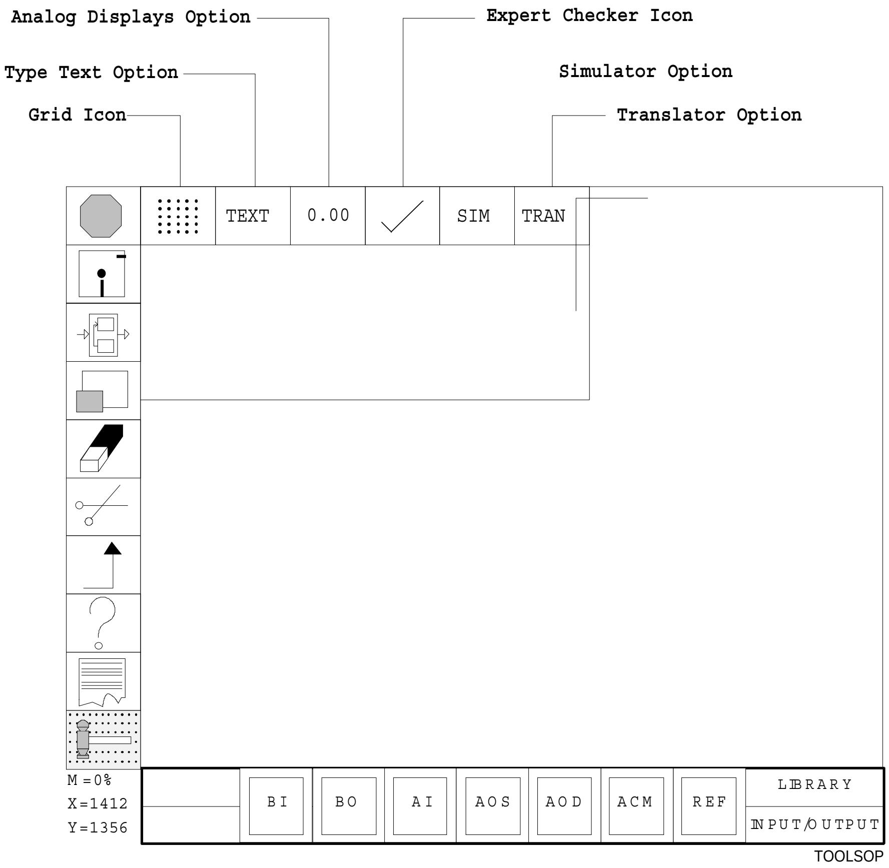


Figure 54: Tools Option Menu

Note: The default option of this icon is Grid On/Off.

Turning the Grid On and Off

The Grid function aids you in drawing neat and tidy diagrams. When it is on, blocks, connections, text, and analog displays “snap” into place, which helps you line up and orient these items on a diagram. The grid is defaulted to on and, therefore, is always on unless you turn it off. Use this procedure to turn the grid on or off.

- To turn the grid on, click left on the Tools icon (hammer). Blue dots appear in the background of the Tools icon.
- To turn the grid off, click left on the Tools icon (hammer). The blue dots in the background of the Tools icon disappear.

Details

You can change the grid in two ways: by clicking left on the Tools icon, or double-clicking left on the Tools icon and clicking left on the Grid icon.

Typing Text

TEXT

You can add comments and statements to the diagram to explain its logic or its function blocks.

1. Double-click left on the Tools icon (hammer). The option menu appears.
2. Click left on the TEXT option. The cursor changes to an underline.
3. Position the underline cursor where you wish to begin typing and click the left button. Type away. Use the Enter key to start a new line directly under the first. When complete, press Esc.
4. You can now move to another spot in the work area to continue, or you may stop this operation by clicking left on any icon.

Details

While you are tying text, use the Backspace key to erase previously typed characters. When you are finished typing a line, press Enter. The cursor moves to the next line directly under the first character of the previous line. This allows you to type text that is left justified.

You can move, size, delete, and duplicate text just as function blocks. Refer to the *Move, Copy, and Resize Functions* and *Erase and Delete Functions* sections for details. However, you cannot change text that is pasted down; you need to erase and retype it.

Depending on the characters used, text may be slightly shorter in length on the Simulator screen and on printouts than on the Editor screen.

The following foreign language characters can be typed as text in a control strategy:

Ä,Å,à,á,â,ä,å	Ñ,ñ
Çç	Ö,ó,ô,ö
É,è,é,ê,ë	Ü,ù,ú,û,ü
í,î	ß

Pasting Down Analog Displays

0.00

Analog displays are numeric fields that are placed at the outputs of function blocks to indicate their values during simulation.

Note: You can use analog displays only for function blocks that output analog data. The outputs of binary blocks are indicated by colored lines in the Simulator. Blocks that output time data cannot be represented. Refer to the *Simulator* chapter for details.

1. Double-click left on the Tools icon (hammer).
2. Click left on the Analog Displays (0.00) option. A string of zeros comes to the screen.
3. Click right/middle to select the decimal position and resolution you want.
4. Position the field as desired and click left. The analog display pastes down and a menu displays.
5. From this menu, select the system\object name or block name that you want to associate with the analog display. Then, select the connection name. The analog display is now associated with that block’s output connection. Another analog display appears, which you can associate with a different block.

Details

You can move, copy, delete, and resize an analog display just as a function block. If resized, the Simulator will not show the size change. Refer to the *Move, Copy, and Resize Functions* and *Erase and Delete Functions* sections for details.

You can associate an analog display with the same connect name multiple times in the same file.

The decimal positions and resolutions that are available range from 0.0 to 000000. See the *Simulator* chapter for details.

You may change the decimal position for an analog display that is pasted down by selecting the Move icon, clicking left on the display, and clicking the right\middle button until the position you want is shown.

If a REF block has a block label specified, the label is shown in the analog displays menu in blue, next to its object name.

Running the Expert Checker

The GPL Expert Checker checks the correctness and completeness of GPL diagrams and compounds in a control strategy. Below are the instructions for starting the Expert Checker. Refer to the *Expert Checker* chapter for more details.

Note: We recommend that you use the Expert Checker on a control strategy file before simulating or translating it.

1. Double-click left on the Tools icon (hammer).
2. Click left on the Expert Checker icon (checkmark).

   Note: If the file has been changed and not yet saved, the message `WARNING! File/Compound not saved——changes will be lost` displays. To bring the Expert Checker to the screen and ignore the changes, click left on [OK]. The checker will use the last saved version of the file. To cancel the Expert Checker, click left on [CANCEL].
3. The Editor clears and the Expert Checker comes to the screen. When checking is complete, press any key to return to the Editor.

Details

While the Expert Checker is running, the screen displays these messages:

```
Expert Checking in progress for file:

{FILE PATHNAME}

Blocks Checked [XXXX]

Expert Check Complete—Number of errors
found: {XX}

Return to the Editor

Strike a key when ready. . .
```

The errors that the Expert Checker finds are sent to the list file. View the list file by clicking left on the VIEW option under the Query option menu.

If a control strategy uses a FILE block, the diagrams under that file are also expert checked.

Running the Simulator

SIM

The GPL Simulator emulates the operation of processes in a control strategy file. Below are the instructions for starting the Simulator. Refer to the *Simulator* chapter for details on Simulator operation.

1. Double-click left on the Tools icon (hammer).
2. Click left on the SIM option (Simulator).

   If the file has been changed and not yet saved, the message `WARNING! File/Compound not saved——changes will be lost` displays. Click left on [OK] to execute the Simulator and ignore the changes. The Simulator will use the last saved version of the file. Click left on [CANCEL] to cancel the Simulator.
3. The Editor clears and the Simulator comes to the screen. When you are done simulating, select QUIT from the command bar menu to return to the Editor.

Running the Translator and Compiler

TRAN

The GPL Translator creates intermediate source code from the process compounds in a control strategy file. The GPL Compiler creates process objects from the intermediate source code. Below are the instructions for running these utilities. Refer to the *Translator* chapter for details.

Note: The Translator will invoke the Expert Checker if the file has not yet been expert checked.

1. Double-click left on the Tools icon (hammer).
2. Click left on the TRAN option (Translator).

   If the file has been changed and not yet saved, the message `WARNING! File/Compound not saved—changes will be lost` displays. Click left on [OK] to bring the Translator to the screen and ignore the changes. The Translator will use the last saved version of the file. Click left on [CANCEL] to cancel the Translator.
3. A submenu displays that provides these options (Figure 55):

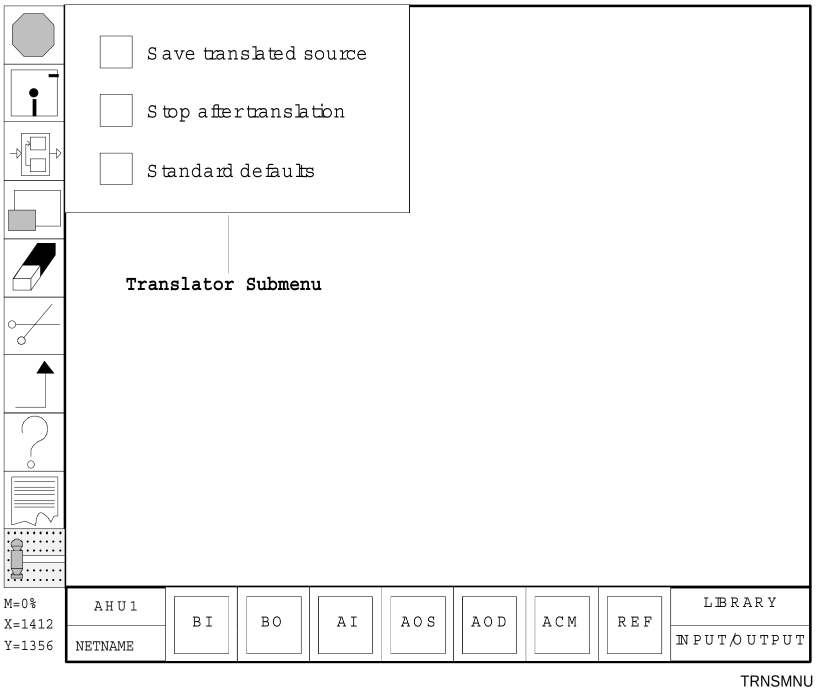


Figure 55: Translator Submenu

Save translated source

Click left inside this checkbox if you want to save the intermediate source file after the executable process objects are created.

Stop after translation

Click left inside this checkbox if you want only the intermediate source file saved. The Compiler will not run, and process objects will not be created.

Standard defaults

Click left inside this checkbox if you want to create the executable process objects and then delete the intermediate source file.

4. After you select one of the above choices, the Editor screen clears and the Translator begins. (The Expert Checker runs first if not yet run on this file.)
5. When the Translator and Compiler are done, press any key to return to the Editor.

Details

While the Translator is running, the screen displays these messages:

```
Translation in progress for {FILENAME}

Blocks Translated {XX}

Translation Complete: {XX} errors

Return to the Editor

Strike a key when ready. . .
```

When the Compiler is running, the screen displays these messages:

```
Compile in progress for {FILENAME}

Compiling Process [XX]-{SYSTEM\OBJECT NAME}

Compile Complete: {XX} Errors {YY} Warnings

Return to the Editor

Strike a key when ready. . .
```

The errors that the Translator and Compiler find are sent to the list file. View the list file by clicking left on VIEW under the Query option menu. If no errors are found, and you elected to save the source file, the list file will contain the source file and some reference tables.

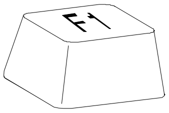
If a control strategy uses a FILE block, the diagrams under that file are also translated and compiled.

Miscellaneous Functions

This section explains how to perform these functions:

- Display a help screen.
- Select a library category.
- Size and paste down a function block in the work area.

Displaying a Help Screen



FIPIC

Use this procedure to display a help screen about the size of the Editor screen. Online help is available for all areas of the screen, including the icons, function blocks, and Editor fields. However, online help is not available with the Expert Checker, Simulator, or Translator utilities.

Note: Online help is not available if an option menu or the function block library is displayed. Also, help is not available when the underline cursor for writing text is displayed.

1. Position the cursor in the area on which you want help.
2. Press F1. The work area clears and the message `WORKING` displays temporarily. The help screen displays in the work area.

Details

Use the Page Down key to display the next page, and the Page Up key to display the previous page. The page number is displayed in the upper right corner of the screen. Click left or press Esc to exit a help screen.

The maximum length of help screens for the icons is three pages, and for the block help screens, five pages.

Help is not available for pasted down function blocks. To display a help screen for a function block, change the function block category to display the block in the block directory (if required), position the cursor on the block, and press F1.

Selecting a Function Block Category

LIBRARY

This procedure changes the function block category. The blocks that belong to the category display in the Function Block Directory (on bottom of screen). You need to change the category if the block you want to paste down is under a category other than the one currently active. GPL offers over 60 blocks that are grouped in 16 categories.

1. Click left on the LIBRARY field (Figure 56). The function block library appears on the right side of the work area.

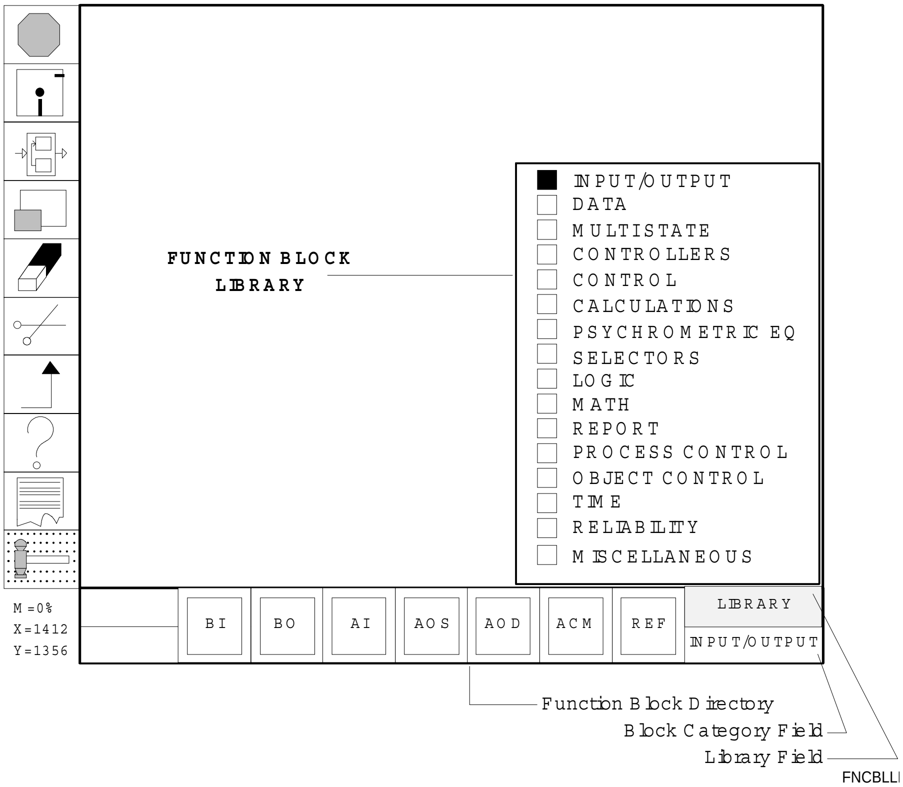


Figure 56: Function Block Library

2. A small checkbox appears to the left of each block category. Click left inside the checkbox for the category you want. The block directory changes to show those that belong to the selected category. Also, the Block Category field changes to the one you selected.

Details

Each function block category has a set number of blocks. A category can have at most seven blocks, all of which display on the block directory.

The Multi-state category contains MSI and MSO object blocks, and the Data category contains the MSD object block. These blocks require special NCM software that is currently available in the European market only.

To clear the function block library, click left in an empty area of the screen.

Sizing and Pasting Down a Function Block

You can size and paste down a function block in the work area. Use the following procedure.

1. Locate the function block that you want to paste down in the Function Block Directory (on bottom of screen). If it is not shown, select the category under which it is assigned. (Refer to the previous section, *Selecting a Function Block Category*, for the steps.)
2. Position the cursor on a function block and click left. The block’s background will back-highlight in red to indicate you have selected it.
3. Move the mouse into the work area and notice a block comes to the screen.

   At this point, you can size the block. You can either size the block manually, or use a function key to change the block to one of three available default sizes. The default size is the size at which a block comes to the screen when you select it.
4. To size the block manually, hold down the right/middle mouse button and drag the mouse.

   To size the block to one of its three default sizes, press F2, F3, or F4. F2 provides a large default size, which is about four times the size of the standard default size. F3 provides a small default size; and F4, the standard default size.
5. Once the block is at the desired size, position it and click left to paste it down in the work area.

Details

After you size a function block, either manually or by using F2, F3, or F4, the size of subsequent blocks that you display for pasting down will be the new size. The function keys are especially important in this case, since you can enlarge the size of one function block, then easily return to the default size for the next block by pressing F4.

After you paste down a function block, another appears on top of it that is a duplicate of the first. This lets you quickly paste down several blocks of the same type. To clear the block, either select another block, or click left on the Query icon.

Object blocks paste down undefined; operation and special blocks paste down defined. Undefined object blocks are colored magenta with dotted borders. They turn brown with solid borders once defined. Defined operation and special blocks are also colored brown with solid borders.

User Messages

This chapter explains all user messages that the GPL Editor may output. The messages are organized alphabetically. Most user messages display on the bottom of the work area, just above the row of function blocks (Figure 57). A few messages appear in the middle of the work area.

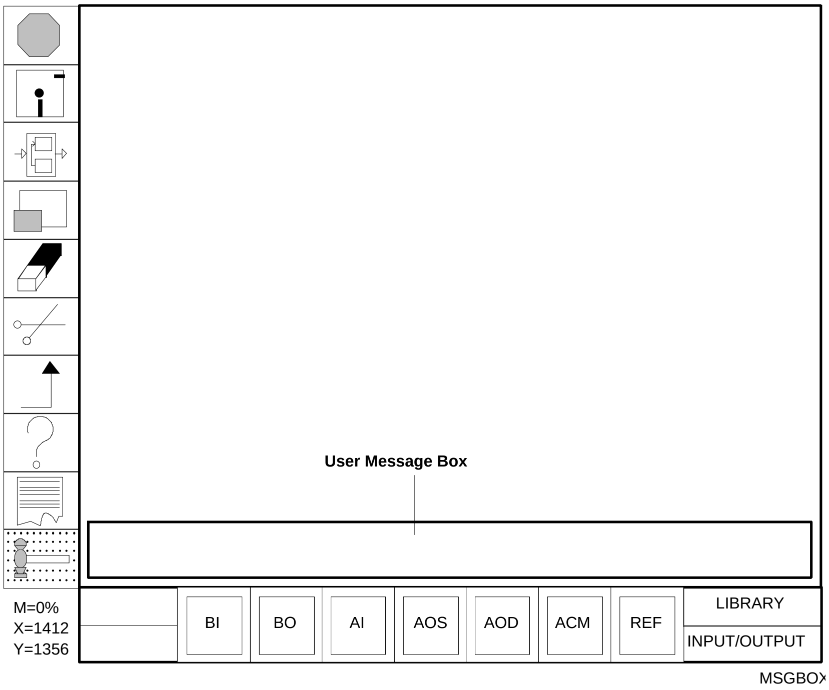


Figure 57: User Messages Box in Work Area

The action message `To continue, click the mouse` accompanies most user messages. If a particular message has a different action message, it is described here.

The items in brackets { } will be replaced by actual system\object names and values on the screen. The following are the user messages:

```
Address, time, or date value is out of
range.
```

The value of the address, time, or date parameter is not within the valid range.

```
All erased items have been restored.
```

There are no more erased items to recover.

```
All files are output--file selection not
allowed.
```

You are trying to print a single file by clicking left on its name in the print queue. This is not the proper method for sending a file to the printer. Instead, click left on the Start Output option.

```
Already at highest compound level.
```

By pressing Esc, you are trying to reach a higher compound level, but the compound displayed is the highest.

```
Analog Type for MAI pt type cannot be
100 ohm--{system\object name}.
```

You have attempted to specify a 100 ohm analog type when the point type is MAI. The 100 ohm analog type is available only if the point type is AI.

```
Archive Database Interaction in progress ...
```

Data in the archive database is being read from or written to the strategy file.

Note: This message appears boxed in the middle of the work area, overlaying the displayed database template.

```
Archive Database not found in the current
directory.
```

The Editor could not find the archive database. The control strategy file must reside under the same directory or subdirectory as the archive database.

```
Archive hardware object record does not
exist--{system\object name}.
```

The object record for the system\object name you specified under the Hardware category does not exist in the archive database, and therefore you cannot use it in a control strategy.

```
Archive object type mismatch--Delete
aborted.
```

You are trying to delete an existing object from the archive database that is of a different type than in the strategy file database.

```
Archive object type mismatch--
{system\object name}.
```

The object in the archive is of a different type than in the strategy file database.

```
Assoc Off Input cannot equal Dbl
Mmnt/Assoc Inp--{system\object name}.
```

The number you entered for Switch Input # cannot be the same as that entered for Off Switch Input #.

```
Assoc Sys\Obj Name not in archive DB--
{system\object name}.
```

The system\object name you defined under the Associated Input category is not defined in the archive database; therefore, you cannot use it in a GPL control strategy.

```
Associated Sys\Obj\Attr, all defined or
all blank--{system\object name}.
```

You must define or leave blank the system\object and attribute fields under the Associated Input category.

```
At least one PIDL output is required.
```

The PIDL object definition is not complete until you define at least one output.

`Attribute name of Assoc. Object is the wrong type--{system\object name}.`

The attribute you specified is not the correct type for the object (e.g., you entered a binary attribute for an AD object).

`Attribute name of Assoc. Object is not valid--{system\object name}.`

The attribute you specified is not valid for the Associated Object.

`Aux Switch must be an AI type--{system\object name}.`

The object that you defined as a reference under the Aux Switch Input category must be an AI object.

`Aux Switch must be on same DCM as PIDL--{system\object name}.`

The object that you defined as a reference under the Aux Switch Input category must be on the same DCM as the PIDL object.

`Aux Switch Sys\Obj Name not in archive DB--{system\object name}.`

The system\object name that you defined as a reference under the Aux Switch Input category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

`Bad drive specification.`

The drive letter you entered is not valid. The valid disk drives are A through Z.

`Cannot add or modify software object mapped to LON hardware.`

You are trying to define or modify a LON object in GPL. These objects must be defined or modified using DDL.

`Cannot make directory.`

You are trying to make a directory that already exists.

`Cannot paste FILE block--at maximum penetration level.`

You are trying to add a FILE block to a file that has reached its maximum number of FILE block levels.

`Cannot paste CMP block--at maximum penetration level.`

You are trying to add a compound level to a file that has reached its maximum number of compound levels.

`Cannot pick last vertex for insertion.`

You are trying to add a line segment at the end of a connection. This is not allowed.

`Cannot put detector on forward activated list.`

A forward activated list cannot contain fire points that are detectors.

`Cannot save the strategy file while in Compound Edit mode.`

You are trying to save the control strategy while editing a compound. This is not allowed.

`Cannot stamp a single connection.`

You are trying to copy a single connection. That is not a valid operation. You must copy the connection with the block to which it is connected.

`Cannot terminate connection within selected compound.`

You have looped back to the originating compound in trying to connect two blocks in different compounds. A connection from one compound must terminate in another compound, not the same compound.

`Cleaning Crew must be NO for Assoc Inp/Assoc Off--{system\object name}.`

The switch numbers you designated for Switch Input # and Off Switch Input # cannot also be defined as “Y” in the Cleaning Crew Switches table.

```
Compound does not contain any blocks for
this loop.
```

You tried to open a compound that does not have any function blocks that are part of the loop.

```
Connection attempted to invalid command.
```

You are trying to connect the wrong type of command to an object block; or, you are trying to connect two CONN blocks that are configured for the same type, but are passing different command types (e.g., Warning command into a CONN block that passes an Alarm command).

```
Connection menu is empty.
```

You are trying to make an output connection from a FILE block. This block has no output connections.

```
Connection not made--Sequential connection
limit exceeded.
```

You have reached the maximum number of sequential connections (120) allowed.

```
Could not initialize the DBM--cannot open
win.ini file.
```

The win.ini file could not be located. Either it is in the wrong directory or it does not exist. The file must be under the same directory as the FMSDOS environmental variable specifies.

```
Could not initialize the DBM--keyword
undefined in win.ini file.
```

The win.ini file does not contain one or more of the following keywords: DATA, FMSDATA, or MODELS. All three keywords are required.

```
Could not initialize the DBM--keyword
undefined in win.ini file.
```

The win.ini file does not contain the keyword value for one or more of the following keywords: DATA, FMSDATA, or MODELS. All three keywords require values.

```
Could not initialize the DBM--out of
environmental space.
```

Your computer does not have enough memory to run GPL. Make sure the SHELL command in the CONFIG.SYS file is set to a large enough number.

```
Could not initialize the DBM--status : xx
```

There is a problem with the database manager. Call Technical Support in Milwaukee, noting the status number that displays.

```
Could not open .LST error file.
```

The Editor cannot open the list file for writing messages that are created with Session Read. Click left on [OK] to continue.

```
Could not set FMSDOS environmental
variable.
```

Your computer does not have enough memory to run GPL. Increase the environmental size as specified by the SHELL command in the CONFIG.SYS file.

```
CS object already assigned to DSC--
{system\object name}.
```

The hardware object you specified is assigned to a C210A or C260A controller.

```
Cursor too close to workspace boundary.
```

You are trying to display a remote connection symbol with the cursor too close to the edge of the work area.

```
Day of week prefix incorrect or repeated
in TIM function.
```

The day-of-week abbreviation entered is incorrect or repeated within the interlock statement. Valid abbreviations are SU, MO, TU, WE, TR, FR, and SA for Sunday through Saturday, respectively.

```
DBM add error.
```

The database manager of the GPL Editor cannot add the object block as you have requested. The database may be faulty. Click left on [OK] to continue.

```
DBM delete error.
```

The database manager of the GPL Editor cannot delete the object block as you have requested. The database may be faulty. Click left on [OK] to continue.

```
DBM error. --{status #}
```

The archive database interface is not working properly. Call the Technical Support Group (TSG) in Milwaukee for assistance, taking note of the status number that displayed.

```
DBM failure. (Spawn--{status #})
```

The Editor cannot access the archive database. This message usually indicates your computer does not have enough memory available.

```
DBM write error.
```

The database manager of the GPL Editor cannot save the changes you made to the object block's template. The database may be faulty. Click left on [OK] to continue.

```
Delete this object from the archive
database?
```

You have selected to delete this object block from the archive database. Perform the deletion by clicking left on [YES]. Or, you may keep the object in the archive by clicking left on [NO].

```
Differential must be undefined (blank)--
{system\object name}.
```

You must leave the Differential blank if the High Limit, Low Limit, Setpoint, and Normalband are all blank (undefined). The Differential is optional.

```
Differential value too high--
{system\object name}.
```

The Differential value you have entered is too high. The following must be true:

(Setpoint - Normalband/2 + Differential) <

(Setpoint + Normalband/2 - Differential)

```
Disk not inserted or not ready.
```

The drive letter you typed is not of a valid drive, or the floppy diskette is missing from the drive.

```
DSC hardware object does not exist in
archive.
```

The object record for the system\object name you specified under the Hardware category does not exist in the archive database; therefore, you cannot use it in a control strategy.

```
DSC logical point type invalid.
```

The logical point type you specified is not valid for this object.

```
DSC type is wrong for this CS object--
{system\object name}.
```

The Hardware object you specified must be of a C210A or C260A type, depending on which CS object template you are editing.

```
Enter or select a name.
```

You selected a function under the File or Compound option menu without clicking left on, or typing in, the name of the strategy or compound.

```
Environment variable GPLCMP not defined--
using default of c:\.
```

The GPLCMP environmental variable is not defined, or not defined properly, in the GPL.BAT file. As a default, the GPL Editor will read the root directory (c:\) when displaying the compound directory.

```
Environment variable GPLPATH not defined--
using default of c:\.
```

The GPLPATH environmental variable is not defined, or not defined properly, in the GPL.BAT file. As a default, the GPL Editor will read the root directory (c:\) when displaying the strategy file directory.

```
Environment variable GPLUMAC not defined--
using default directory.
```

The GPLUMAC environmental variable is not defined, or not defined properly, in the GPL.BAT file. As a default, the GPL Editor will use the C:\FMS\DATA\GPL\JCIUMACS directory for the USER block functions.

```
Erase this block from the GPL Strategy
file?
```

You have selected to erase this object block from the work area and the strategy file. Perform the erase action by clicking left on [YES]. Or, you may keep the object on the work area and in the strategy file by clicking left on [NO].

```
ERROR!  Connection memory exceeds limit.
```

You have reached the maximum size of the connection subfile (.CI) for the control strategy.

```
ERROR!  Database memory exceeds limit.
```

There is not enough PC memory available to load or edit the selected compound. Or, you have reached the maximum allowable mixture of function blocks that the database file (.DB) of the control strategy can contain.

```
ERROR!  Graphic symbol memory exceeds
limit.
```

You have reached the maximum number of text characters that the database file (.DB) of the control strategy can contain.

```
ERROR!  File version number does not match.
```

The control strategy or compound you are trying to load was created with a different version of GPL.

```
Error!  Illegal Operator:
```

The operator you entered is not one of the following: +, -, *, /, ^.

```
Error!  Invalid identifier:
```

You entered an invalid identifier. Only use the following characters and symbols in the equation: I1-I4, C1-C4, +, -, *, /, (,), SQR, ^, LOG, SIN, COS, TAN, ABS, MIN, MAX, AVG, and PI.

```
ERROR!  Missing compound name.
```

You tried to save a compound without specifying a name in the Name field. All compounds must be named.

```
Error!  Missing expression/operand.
```

The function requires an expression or operand.

```
Error!  Missing left parenthesis.
```

A required left parenthesis is missing.

```
Error!  Missing operator.
```

A required operator is missing.

```
Error!  Missing right parenthesis.
```

A required right parenthesis is missing.

```
Error!  No expression.
```

The right parenthesis has no matching left parenthesis.

```
Error!  Too many parameters.
```

You have defined too many parameters for the function.

`ERROR!  Total blocks exceeds limit.`

You have reached the maximum number of blocks in a strategy file (999).

`ERROR!  Total connections exceeds limit.`

You have reached the maximum number of connections in a strategy file (2200).

`ERROR!  Total graphic symbols exceeds limit.`

You have reached the maximum number of text lines in a strategy file (800).

`ERROR!  Write to disk failed—disk full, write protect, etc.`

The write procedure to the disk has failed. The possible causes include: disk is full, disk is write protected, or some other cause. Note that the strategy or compound file name is created, but its contents is blank.

`EXIT TO:`

You have selected the Exit icon to exit the GPL Editor. Click left on [DOS] to go to the DOS environment. Click left on [FMS] to go to the Metasys Network. Or, click left on [CANCEL] to escape the exit operation.

`Feedback Object type must be AI/AD/ACM--{system\object name}.`

The object you defined in the Feedback category must be an AI, AD, or ACM object.

`Feedback Object type must be BI/BD--{system\object name}.`

The object you defined in the Feedback category must be a BI or BD object.

```
Feedback Sys\Obj, all defined or all
blank--{system\object name}.
```

You must define or leave blank the system name and object name fields under the Feedback category.

```
Feedback Sys\Obj names are required--
{system\object name}.
```

You must define the system name and object name fields under the Feedback category.

```
Feedback Sys\Obj names must be blank--
{system\object name}.
```

You must leave blank the system name and object name fields under the Feedback category for an AOS object that is defined for local control.

```
Feedback Sys\Obj Name not in archive DB--
{system\object name}.
```

The system\object name you specified under the Feedback category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
File not found or not in current
directory.
```

The file you selected or typed in does not exist in the currently selected directory.

```
File being printed--cannot update.
```

You tried to add a print file to the queue that is currently printing.

```
Find next occurrence?
```

Whether you wish to find the next place this block is used in the control strategy. Click left on [OK] to find the next reference, or click left on [CANCEL] to exit the Find operation.

`Group Number already used--{system\object name}.`

The group number you specified for the LCG object is already being used by another LCG object.

`Hardware object record does not exist.`

The object record for the system\object name you specified under the Hardware category does not exist in the archive database; therefore, you cannot use it in a control strategy.

`Hardware slot already used--{system\object name}.`

The hardware slot you specified is taken by another object.

`Hardware slot must be an odd number--{system\object name}.`

The hardware slot for a Form C Point Type of an ACM or BI object must be an odd number.

`Hardware slot not available--{system\object name}.`

The hardware slot you specified is already taken by a different object.

`Hardware subslot already used--{system\object name}.`

The hardware subslot you specified is already taken by a different object.

`Hardware type mismatch in the archive DB--{system\object name}.`

The HW (hardware) type you selected does not match the Hardware System\Object name specified.

`High Input Span must be > Low Input Span--{system\object name}.`

You must define a Span High Input that is greater than the Span Low Input (if defined).

```
High Limit must be > Low Limit--
{system\object name}.
```

You must define a High Alarm Limit that is greater than Low Alarm Limit (if defined).

```
High Limit value too low--{system\object
name}.
```

The High Limit value you have entered is too low. The following must be true:

(High Limit - Differential) > (Setpoint + Normalband/2)

```
High Output Span cannot equal Low Output
Span--{system\object name}.
```

You must define a Span High Output that is not equal to the Span Low Output (if defined).

```
High Sat must be an AI type--
{system\object name}.
```

The object that you defined as a reference under the High Saturation category must be an AI object.

```
High Sat must be on same DCM as PIDL--
{system\object name}.
```

The object that you defined as a reference under the High Saturation category must be on the same DCM as the PIDL object.

```
High Sat Sys\Obj Name not in archive DB--
{system\object name}.
```

The system\object name that you defined as a reference under the High Saturation category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
HR:MIN:SEC requires colon (:).
```

A colon (:) is required to separate the hour, minute, and second values.

```
Input {1-6} must be an AI type--
{system\object name}.
```

The object that you defined as a reference under the Input n (1-6) category must be an AI object.

```
Input {1-6} must be on same DCM as PIDL--
{system\object name}.
```

The object that you defined as a reference under the Input n (1-6) category must be on the same DCM as the PIDL object.

```
Input {1-6} Sys\Obj Name not in archive
DB--{system\object name}.
```

The system\object name that you defined as a reference under the Input n (1-6) category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
Insufficient memory to run Editor.
```

There is not enough memory in RAM to run the GPL Editor.

```
Invalid block name.
```

You have entered a reserved word that cannot be used for the Block Name of a SVAR block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Invalid character in file name.
```

You have entered an invalid character in the File Name field of the FILE block template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid character in the process obj name.
```

You have entered an invalid character in the Object Name field of the process compound template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid character in the software object
name.
```

You have entered an invalid character in the Object Name field of an object block template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid character in block name.
```

You have entered an invalid character in the Block Name field of the SVAR block template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid character in the software system
name.
```

You have entered an invalid character in the software System Name field of an object block template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid character in type/file name.
```

You have entered an invalid character in the Type/File Name field of the USER block template. *Appendix F: Characters, Symbols, and Reserved Words* lists all invalid characters.

```
Invalid command type.
```

You are trying to connect a command line from a command block that is already configured to an object that cannot accept this type of command.

```
Invalid directory in GPLCMP--using default
of c:\.
```

The directory specified for the GPLCMP environmental variable in the AUTOEXEC.BAT file is invalid. As a default, the GPL Editor will read the root directory (c:\) when displaying the compound directory. Refer to the *Introduction* chapter for details.

`Invalid directory in GPLUMAC--using default directory.`

The directory specified for the GPLUMAC environmental variable in the AUTOEXEC.BAT file is invalid. As a default, the GPL Editor will use the C:\FMS\DATA\GPL\JCIUMACS directory for the USER block functions. Refer to the *Introduction* chapter for details.

`Invalid file name.`

You have entered a reserved word that cannot be used for the Block Name of a FILE block. *Appendix F*: *Characters, Symbols, and Reserved Words* lists all reserved words.

`Invalid logical point number for ASP type.`

The logical point number you entered is outside the range for an AOS object that has ASP as its logical point type. The valid range is 4 to 67.

`Invalid logical point number for BSP type.`

The logical point number you entered is outside the range for an BO object that has BSP as its logical point type. The valid range is 5 to 7.

`Invalid name.`

You have entered a reserved word that cannot be used for the name of a directory, file, or compound block. *Appendix F* lists all reserved words.

`Invalid Number of States and Wired 0 for H/W Ref--{system\object name}.`

You have entered a number of states and wired 0 that is invalid for the specified hardware reference.

`Invalid Number of States for H/W Ref--{system\object name}.`

You have entered a number of states that is invalid for the specified hardware reference.

`Invalid operand.`

A legal parameter such as Z1 or L2D5 was expected but not found at the point of the error.

```
Invalid path in GPLPATH--using default of
c:\.
```

The directory pathname specified for the GPLPATH environmental variable in the AUTOEXEC.BAT file is invalid. As a default, the GPL Editor will read the root directory (c:\) when displaying the strategy file directory. Refer to the *Introduction* chapter for details.

```
Invalid process obj name.
```

You have entered a reserved word that cannot be used for the Process Object Name of a compound block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Invalid process sys name.
```

You have entered a reserved word that cannot be used for the Process System Name of a compound block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Invalid software object name.
```

You have entered a reserved word that cannot be used for the software Object Name of an object block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Invalid software system name.
```

You have entered a reserved word that cannot be used for the software System Name of an object block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Invalid type/file name.
```

You have entered a reserved word that cannot be used for the name of a USER block. *Appendix F: Characters, Symbols, and Reserved Words* lists all reserved words.

```
Last erased item restored. Returning to
Erase mode.
```

The item that you erased last has returned to the diagram. The Erase function is now enabled.

```
Logical point number already used--
{system\object name}.
```

The logical point number you specified is taken by another object.

```
Loop Detected!  Select the Loop-master
block.
```

You have formed a loop while making connections on the diagram. To correct the loop, choose which function block you want GPL to execute first by clicking left on it.

```
Loop Number already used - {system\object
name}.
```

The loop number you specified for the PIDL object is already being used by another PIDL object.

```
Low Limit value too high--{system\object
name}.
```

The Low Limit value you have entered is too high. The following must be true:

(Low Limit + Differential) < (Setpoint - Normalband/2)

```
Low Sat must be an AI type--{system\object
name}.
```

The object that you defined as a reference under the Low Saturation category must be an AI object.

```
Low Sat must be on same DCM as PIDL--
{system\object name}.
```

The object that you defined as a reference under the Low Saturation category must be on the same DCM as the PIDL object.

```
Low Sat Sys\Obj Name not in archive DB--
{system\object name}.
```

The system\object name that you defined as a reference under the Low Saturation category is not defined in the archive database; therefore, you cannot use it in a GPL control strategy.

```
Maximum compound penetration level has
been reached.
```

You are trying to query a compound in a file that has reached its maximum number of compound levels.

```
Maximum file penetration level has been
reached.
```

You are trying to access a FILE block in a file that has reached its maximum number of FILE block levels.

```
Minimum On Time too high for this Load--
{system\object name}
```

You have attempted to modify the Minimum On Time for this object to a value that is greater than the load's Minimum Release Time. (This condition can occur if the object has been defined as a DLLR load when previously added to the archive database via DDL or online generation.)

```
Minimum Off Time too low for this Load--
{system\object name}
```

You have attempted to modify the Minimum Off Time for this object to a value that is greater than the Comfort Override's Minimum Shed Time. (This condition can occur if the object has been defined as a DLLR load with a Comfort Override attribute when previously added to the archive database via DDL or online generation.)

```
Missing Hardware System\Object Name--a
required field.
```

You must define a hardware system\object name for the object.

```
Missing operator inside DEL (Delay)
function.
```

The Delay function requires an operator.

```
Missing System\Object Name--a required
field.
```

You must define a system\object name for the object.

```
Missing System\Object Name for PIDL--
Aux Switch. Inp.
```

You must define a system\object name of the object that the PIDL object will use for its Auxiliary Switch Input.

```
Missing System\Object Name for PIDL--
Hi Sat Limit.
```

You must define a system\object name of the object that the PIDL object will use for its High Saturation Limit input.

```
Missing System\Object Name for PIDL--
Input {1-6}.
```

You must define a system\object name of the object that the PIDL object will use for its Input {1-6}.

```
Missing System\Object Name for PIDL--
Low Sat Limit.
```

You must define a system\object name of the object that the PIDL object will use for its Low Saturation Limit input.

```
Missing System\Object Name for PIDL--
Offset.
```

You must define a system\object name of the object that the PIDL object will use for its Offset input.

```
Missing System\Object Name for PIDL--
Sel Input Signal.
```

You must define a system\object name of the object that the PIDL object will use for its Selector Input.

```
Missing System\Object Name for PIDL--
Setpoint.
```

You must define a system\object name of the object that the PIDL object will use for its Setpoint input.

```
Missing {XXXXX}--a required field.
```

You must define a value for {XXXXX}, a required field. The {XXXXX} can be any valid template field.

`MM-DD-YY requires hyphen (-).`

A hyphen (-) is required to separate the month, day, and year values.

`Must terminate connection within current compound.`

You are trying to enter a higher compound level to terminate a connection to a block. You must end the connection within the displayed compound.

`No Analog connections are available.`

This block has no analog connections available.

`No Binary connections are available.`

This block has no binary connections available.

`No blocks available for selection.`

The control strategy has no blocks that can accept a remote connection or analog display, or be located with the Find function.

`No COMMAND connections are available.`

This block has no command connections available.

`No COMMAND/WRITE connections are available.`

This block has no command or write connections available.

`No description available.`

This control strategy or compound has no text description file. The file must have the same name as the control strategy or compound with a .DDS (strategy) or .CDS (compound) extension.

Note: This message displays in the middle of the work area. To clear it, click left.

`No DUAL COMMAND connections are available.`

This block has no dual command connections available.

`No help available.`

You tried to display a help screen, but the Editor could not locate the help file. The two help files, ICON.HLP and BLOCK.HLP, must be in the following directory: C:\FMS\BIN\GPL.

`No loop devices allowed as parameters for the XZONE function.`

The XZONE operator only allows fire zones as parameters.

`No Loop-master selected. Last connection will be deleted.`

You clicked outside the work area instead of clicking on the Loop-master block. When you click the mouse again, the connection will erase and you'll need to draw it again.

`No .LST file available.`

You tried to display the list file, but the Editor could not locate the file. Either the file is in the wrong directory, or it has not yet been created with Session Read, the Expert Checker, or the Translator/Compiler. The .LST file should be under the same directory as the control strategy file.

`No READ connections are available.`

This block has no read connections available.

`No Session Read required.`

You tried to perform a Session Read on a strategy file whose object blocks are already matched with the data in the archive database.

`No source code available.`

This USER block has no source file. The file you specify in the database template must be the same name as the ASCII source file. Also, the source file must be in the appropriate directory with a .MAC extension.

Note: This message displays in the middle of the work area. To clear it, click left.

`No TIME connections are available.`

This block has no time connections available.

`No WRITE connections are available.`

This block has no write connections available.

`Non-existent directory.`

The currently displayed directory is showing the root directory.

`Normal State cannot be NONE when latching is Yes--{system\object name}.`

If you answered Y (Yes) for the Latching Point field, you cannot specify NONE in the Normal State field.

`Not a binary connection.`

You are trying to exempt a non-binary connection. Only binary connections can be exempt from triggering a process.

`Not a connection.`

You are trying to exempt something other than a connection (e.g., a block).

`Not enough memory available for DBM operation.`

The GPL Editor could not perform a Database Manager (DBM) function because the computer's RAM is full. Check to make sure no unnecessary TSR programs are in memory.

`NOT function operates on only one value.`

Only one value can be specified with the NOT function. The value can come from a single parameter (e.g., Z5 or L1D3) or an equation that results in a single value (e.g., (OR(Z28 Z29) or OR(Z1 AND (Z3 Z4)). An example of an invalid function is: NOT(Z1 Z2), which has two parameters.

`Object does not exist, Delete aborted.`

The object block you are trying to delete from the archive database does not exist.

```
Object record does not exist in archive
DB--{system\object name}.
```

The object you are trying to read from the archive database is missing.

```
Off Switch In—already used on LC Device--
{system\object name}.
```

The Off Switch Input Number you specified is already used by the Intelligent Lighting Controller.

```
Offset must be an AI type--{system\object
name}.
```

The object that you defined as a reference under the Offset category must be an AI object.

```
Offset must be on same DCM as PIDL--
{system\object name}.
```

The object that you defined as a reference under the Offset category must be on the same DCM as the PIDL object.

```
Offset Sys\Obj Name not in archive DB--
{system\object name}.
```

The system\object name that you defined as a reference under the Offset category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
Only 128 Hardware Refs allowed for this
device--{system\object name}.
```

While attempting to add a software object to a System 9100 device, you have used the 129th hardware reference. Each System 9100 controller can have a maximum of 128 of its hardware references mapped to software objects.

```
Output {1-8} attr, VALUE if AOD / not
VALUE if PIDL--{system\object name}.
```

The attribute that you specified under the Output *n* {1-8} category must be Value if the system\object name is of an AOD object; or, you must specify some other valid attribute other than Value if the system\object name is of a PIDL object.

```
Output {1-8} must be an AOD/PIDL type--
{system\object name}.
```

The object that you defined under the Output *n* {1-8} category must be an AOD or PIDL object.

```
Output {1-8} must be on same DCM as PIDL--
{system\object name}.
```

The object that you defined under the Output *n* {1-8} category must be on the same DCM as the PIDL object.

```
Output {1-8} Sys\Obj\Attr, all defined or
all blank--{system\object name}.
```

You must define or leave blank the System Name, Object Name, and Attribute fields under the Output *n* {1-8} category.

```
Output {1-8} Sys\Obj Name not in archive
DB--{system\object name}.
```

The system\object name that you defined under the Output *n* {1-8} category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
Panel/Pt. addresses must both be zero or
non-zero--{system\object name}.
```

You must enter either a zero or non-zero number for both the Annunciator Panel--and Point--address parameters.

```
Parentheses mismatch. Left and right
parentheses do not match.
```

A required left or right parenthesis is missing.

```
PID Loop number for DCM must be 1-16--
{system\object name}.
```

You attempted to enter a PID Loop number greater than 16 for a PID Loop object on a DCM. For a DCM, the PID Loop number range is 1-16. For a DCM140, the range is 1-20.

```
Point address already used.
```

The point address you specified is taken by another object.

`Point address out of range.`

The point address you specified is not within the valid range for the OTHER controller type. Each OTHER controller has its own range. Refer to the engineering document for your particular controller.

`Printer is already active.`

You clicked left on Start Output while the print operation is already in progress. The print operation will continue until all files are sent to the printer.

`Printer not ready.`

The printer to which you are trying to send a print file is offline or disconnected.

`Process has been recompiled without using GPL.`

The process compound that you have queried has been compiled with some other utility than the GPL Compiler. If you want the strategy file version of the process to overwrite the archive database version the next time it is translated with GPL, click left on [OVRWRITE]. The process compound block will stay defined. If you do not want the strategy file version of the process to overwrite the archive database version, click left on [IGNORE]. The block will change to undefined.

`Process object code already exists in the archive database.`

You have pasted down a process compound that has object code in the archive database. If you want the strategy file version of the process to overwrite the archive database version the next time it is translated with GPL, click left on [OVRWRITE]. The process compound block will stay defined. If you do not want the strategy file version of the process to overwrite the archive database version, click left on [IGNORE]. The block will change to undefined.

```
Process object code does not exist--
{system\object name}.
```

The process compound that you have queried has no associated object code. This is merely a status message, not an error message.

```
Protected process/compound--expansion/edit
not allowed.
```

You tried to edit or load the contents of a protected compound. You cannot perform this operation on protected compounds.

```
Reader Number specified does not exist--
(system\object name).
```

You specified an undefined Reader Number for a Binary Input object. The Reader Number you specify must reference a defined Card Reader object on a defined D600 Access Controller.

```
Required DBM files not found in correct
path.
```

The GPL Editor could not perform a Database Manager (DBM) function. Either the DBM executable files are missing from the C:\FMS\BIN directory, or the C:\FMS\BIN directory is not specified in the AUTOEXEC.BAT PATH statement.

```
Remote destination not allowed if crossed
into compound.
```

You are trying to create a remote connection that is at a different compound level from which it started. You must start and end the remote connection at the same compound level.

```
Same file has been penetrated--Check file
name.
```

You are trying to access a file that has the same name as its parent file. Open the FILE block template and assign it a unique name.

```
Sel Inp Signal must be an AI type--
{system\object name}.
```

The object that you defined as a reference under the Select Input category must be an AI object.

```
Sel Inp Signal must be on same DCM as
PIDL--{system\object name}.
```

The object that you defined as a reference under the Select Input Signal category must be on the same DCM as the PIDL object.

```
Sel Inp Sig Sys\Obj Name not in archive
DB--{system\object name}.
```

The system\object name that you defined as a reference under the Select Input category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

```
Session Read Complete--{n} Error(s) Found.
```

The Session Read operation for this strategy file is complete. If any errors were found, examine the list file by clicking left on VIEW under the Query option menu.

```
Session Read in Progress ...
```

The Session Read function, which matches the strategy file database with the archive database, is in progress.

Note: This message appears boxed in the middle of the work area, overlaying whatever may be displayed.

```
Session Read not available when in No
Archive Mode.
```

The Session Read operation is available only when the Editor is interacting with the archive database.

```
Setpoint & Normalband must both be
defined/unde --{system\object name}.
```

You must define or leave blank the Setpoint and Normalband values.

```
Setpoint must be an AI type—{system\object
name}.
```

The object that you defined as a reference under the Setpoint category must be an AI object.

`Setpoint must be on same DCM as PIDL--{system\object name}.`

The object that you defined as a reference under the Setpoint category must be on the same DCM as the PIDL object.

`Setpoint Sys\Obj Name not in archive DB--{system\object name}.`

The system\object name that you defined as a reference under the Setpoint category is not defined in the archive database, and therefore you cannot use it in a GPL control strategy.

`Span Values must all be defined or undefined—{system\object name}.`

You must define or leave blank all four span values (Span High Input, Span Low Input, Span High Output, Span Low Output).

`Software and hardware obj must be on same NCM--{system\object name}.`

The objects you defined under the Identification and Hardware categories must be on the same NCM.

`STD Range value for FPU must be 0, 26-33--{system\object name}.`

The standard range value you specified is outside the accepted range for an FPU hardware type. The range is 0 and any number between 26 and 33.

`Stop time must be greater than start time within TIM function.`

The stop time must be a later time than the start time. Crossing over midnight is not allowed.

`Switch In--already used on LC Devic --{system\object name}.`

The Switch Input Number you specified is already used by the Intelligent Lighting Controller.

```
System name does not exist in the archive
DB--{system\object name}.
```

The system name you specified is not defined in the archive database; therefore, you cannot use it in a GPL control strategy.

```
System\Object name already exists in
archive DB--{system\object name}.
```

The system\object name you specified is already defined in the archive database, and therefore cannot be added to the archive.

```
System\Object name and block label are not
unique.
```

Two REF blocks in the same file have the same system\object name and block label. The block label for each REF block that has the same system\object name must be unique.

```
Text found past closing parenthesis.
```

A parameter or operator is found past the point where the forward activated list or reverse activated equation should end.

```
The L/W System\Object name is not unique
in this GPL strategy file.
```

You are trying to duplicate an object block that already exists in the strategy file. A strategy file can have at most one instance of a particular object block (i.e., the same system\object name).

```
This function is not allowed while in the
Compound Edit mode.
```

You are trying to access the contents of a FILE block while editing a compound. This is not allowed.

```
To continue, click the mouse.
```

To clear the work area of the user message, click the left mouse button.

```
Time zone specified is not defined--
{system\object name}.
```

You have entered a value in the Suppress TZ field that doesn’t match an existing time zone.

```
Too many connection levels.
```

You are trying to draw a connection through more than ten compound levels (the maximum).

```
Too many connection segments.
```

You have reached the maximum allowed connection segments. A direct connection can have up to 14 segments, and a remote connection, seven segments. Click right/middle to erase the last connection segment.

```
Too many items for move/copy operation.
```

There is not enough memory in RAM to perform the move or copy operation, or to load the expanded compound.

```
Too many operands and operators.
```

The interlock statement is too large to be stored properly.

```
Translator not available when in No Archive
Mode.
```

The Translator utility is available only when the Editor is interacting with the archive database.

```
Tune Noise Band must be greater than
Deadband--{system\object name}.
```

You must enter a Tune Noise Band that is greater than the Deadband.

```
Type designation incorrect for module or
detector loop device.
```

The type designation for this parameter is incorrect. A parameter identified as a loop device must be either a module or a detector.

```
Unknown operator detected in the reverse
equation.
```

The reverse equation contains an unknown operator.

```
Update GPL block with archive database
data?
```

This object or process block already exists in the archive database. Click left on [UPDATE] to copy in block’s template information from the archive database. Or, click left on [IGNORE] to keep the block’s template information as stored in the strategy file.

```
Verify file deletion.
```

Indicate that you wish to delete the file or compound by clicking left on [OK]. Or, click left on [CANCEL] to exit this operation.

```
WARNING!!!  Disk space is low--loss of
data possible.
```

There is not enough computer memory available for the GPL Editor. Take out any unnecessary TSRs and set up a RAMDISK. Refer to the README file that came with the GPL program diskettes.

```
WARNING!  File/compound not saved--changes
will be lost.
```

You edited the file or compound and have not yet saved the changes. Click left on [OK] to ignore the changes and execute the selected operation. Or, click left on [CANCEL] to clear this message so you can save the file or compound.

```
WARNING!  File exists. UPDATE?
```

The control strategy file, compound file, or print file already exists. Click left on [OK] to write over the file and update it; or, click left on [CANCEL] to escape the operation. Note that the Editor does not allow you to create two print files with the same name. For example, if the file name of a control strategy is AHU1\COOLING and the file name of a compound is JOB123\AHU1\COOLING, the Editor will not allow you to create two print files called COOLING.PRD.

```
WARNING!  Printing in progress--exiting
now may lose output!
```

You are trying to exit the Editor or go to one of the GPL utilities while GPL printing is in progress. Click left on [OK] to exit GPL or go to the utility; or, click left on [CANCEL] to remain in the Editor and continue printing.

```
WARNING!  Process object code does not
exist.
```

You queried a defined process compound that has never been translated and compiled, and therefore, its object code does not exist in the archive database. This message points out that the process is not complete until it is translated and compiled.

```
WARNING!  Write to disk failed. Out of
disk space.
```

You tried to save a compound to a disk that is full.

```
WARNING!  You are about to clear the
entire diagram.
```

The GPL Editor is about to clear the entire control strategy file from the computer's RAM. Click left on [OK] to clear memory; or, click left on [CANCEL] to escape the CLR ALL operation.

```
Wired 0 invalid for Local Contact H/W
Reference--{system\object name}.
```

You have attempted to define wired 0 for this local contact when the specified hardware reference is not valid for wired 0.

```
Working ...
```

The GPL Editor is loading information from the help file.

```
Zone on wrong side of Last Forward Zone—
{system\object name}.
```

For a forward activated zone, the number you enter must be less than or equal to the zone boundary. For a reverse equation, the number must be greater than the value of the zone boundary.

-METASYS®

GPL Programmer's Manual

Expert Checker

* Indicates those sections where changes have occurred since the last printing.

Overview

This chapter contains instructions for using the Expert Checker.

The Expert Checker is a software program that verifies the completeness and correctness of GPL control strategies. Use this debugging utility to check a strategy file for errors before simulating or translating the file.

Any time you create or edit a strategy file, you must use the Expert Checker to check the file for completeness and correctness.

The Expert Checker:

- Performs error checks on control diagrams generated with the GPL Editor or CAE Translator.
- Saves all errors in a list file, which you can view in the Editor.
- Returns you to the GPL Editor so you can correct the errors before simulating or translating the file.

The Expert Checker checks a strategy file for missing connections, improper configurations, missing template information, and undefined blocks.

The Expert Checker saves messages about all errors (except fatal errors) in a list file. This list file has the same file name and is placed in the same directory as the strategy file being checked, with a .LST extension.

The Expert Checker examines all subfiles that are referenced by FILE blocks in the main file. This allows you to check all subfiles at one time. Errors from subfiles are saved in the same list file as errors from the main file.

Using the Expert Checker

This section tells you how to use the Expert Checker and describes the list file. The section includes:

- Running the Expert Checker
- During the Expert Check
- Viewing Errors in the List File

Running the Expert Checker

Starting the Expert Checker from the Translator

Before the Translator will translate a strategy file into downloadable process objects, the file must be run through the Expert Checker and found to be free of errors. If you attempt to translate a file that has not been checked, GPL automatically starts the Expert Checker to check the file. You'll find complete information on the Translator in the Translator chapter.

Starting the Expert Checker from the Editor

To run the Expert Checker from the Editor, use the following steps:

1. Load the control strategy file that you want to check into the work area of the Editor.
2. Double-click left on the Tools icon to display the Tools option menu.

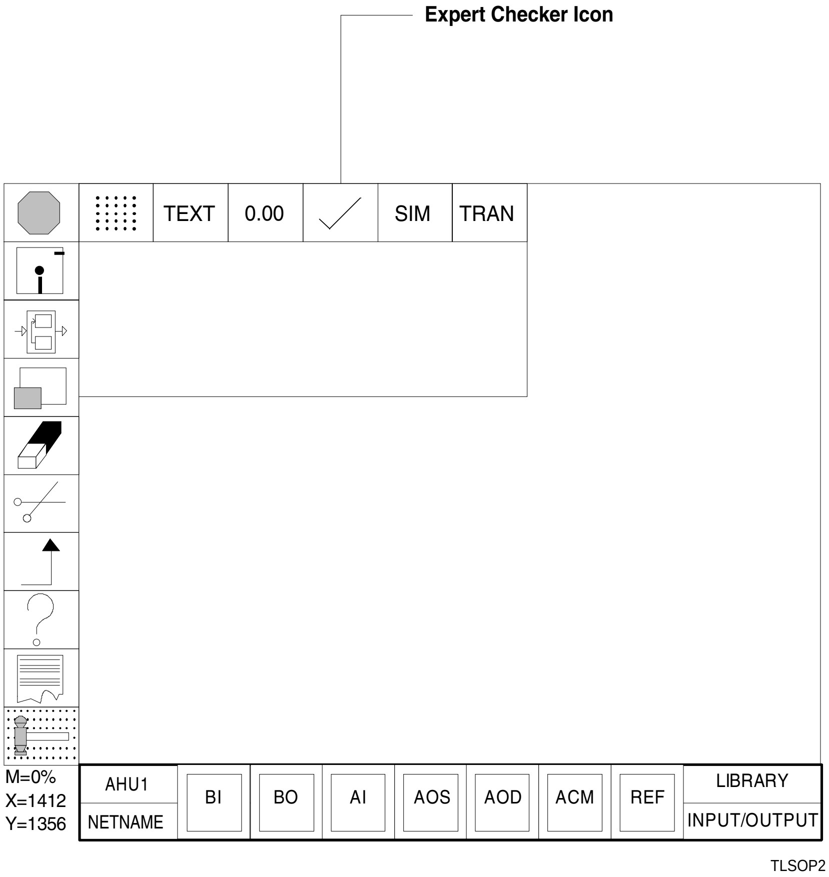


Figure 1: Tools Option Menu

3. Click left on the Expert Checker icon (check mark) in the Tools option menu.

As soon as you select the Expert Checker, the Editor screen clears and the Expert Checker begins checking the file. Messages about the progress of the check appear on the screen. These messages are described below. After the check, press any key to return to the Editor.

During the Expert Check

Expert Checked Flag

The first thing the Expert Checker does is check the Expert Checked flag (which every strategy file has). This flag indicates whether checking is required. Whenever you save a new or edited strategy file, the Expert Checked flag is set to No, indicating that checking is required. When a file has been checked and found to be free of errors, the Expert Checked flag is set to Yes, indicating that checking is not required.

If the Expert Checker finds the flag set to Yes, it does not check the file, and the following information appears on the screen.

```
Expert Checking in progress for file:

{FILENAME}

Blocks Checked [   0]

Expert Check complete - Number of errors found: 0

Return to the Editor

Strike a key when ready...
```

The zeros in the Blocks Checked and Number of Errors Found fields indicate that checking did not occur because the file has already been checked and found to be free of errors.

In this case, the Expert Checker sends a message to the list file stating that the file has been previously checked and found to be free of errors.

Expert Checker In Progress

If the Expert Checked flag is set to No, checking takes place, and the following message appears on the screen.

```
Expert Checking in progress for file:

{FILENAME}

Blocks Checked [  XX]
```

This message displays the name of the file being checked and the number of blocks checked so far. The Blocks Checked field increments by one as each block is checked.

All subfiles referenced by FILE blocks are also checked.

Fatal Errors

If a fatal error occurs, the message associated with the error is shown on the screen, and the Expert Checker is terminated immediately. Fatal errors are listed in the Error Messages section later in this chapter.

Here is an example of a message that appears when a fatal error occurs.

```
Expert Checking in progress for file:

{FILENAME}

Cannot create list file, {FILENAME.LST}

Expert Check complete - Fatal error

Return to the Editor

Strike a key when Ready...
```

No Errors Found

When the checking procedure is complete and no errors are found, the following message appears.

```
Expert Checking in progress for file:

{FILENAME}

Blocks Checked [  XX]

Expert Check complete - No errors

Return to the Editor

Strike a key when Ready...
```

Errors Found

If non-fatal errors are found, the following message appears.

```
Expert Checking in progress for file:

{FILENAME}

Blocks Checked [  XX]

Expert Check complete - Number of errors found: {XX}

Return to the Editor

Strike a key when Ready...
```

The Expert Checker saves all errors (except fatal errors) in the list file, which can be viewed in the Editor.

Viewing Errors in the List File

The list file displays informative error messages for all non-fatal errors found by the Expert Checker.

If the list file does not exist, the Expert Checker creates it. If the list file does exist, the Expert Checker clears its contents before saving any new error messages in it.

The list file has the same name as the strategy file you checked, with a .LST extension (it is also placed in the same directory). For example, the list file for a file called AHU1 is AHU1.LST.

When a subfile is checked, the current content of its list file is cleared and replaced with a message stating the file has been checked as a subfile (if the subfile doesn't have a list file, one is created). The message directs you to the file that contains the subfile errors. For example, let's say you check a file called AHU1 that has a subfile called CLG. All errors are saved in a list file called AHU1.LST, including the errors in the CLG subfile. If you view the list file for CLG, you will find this message.

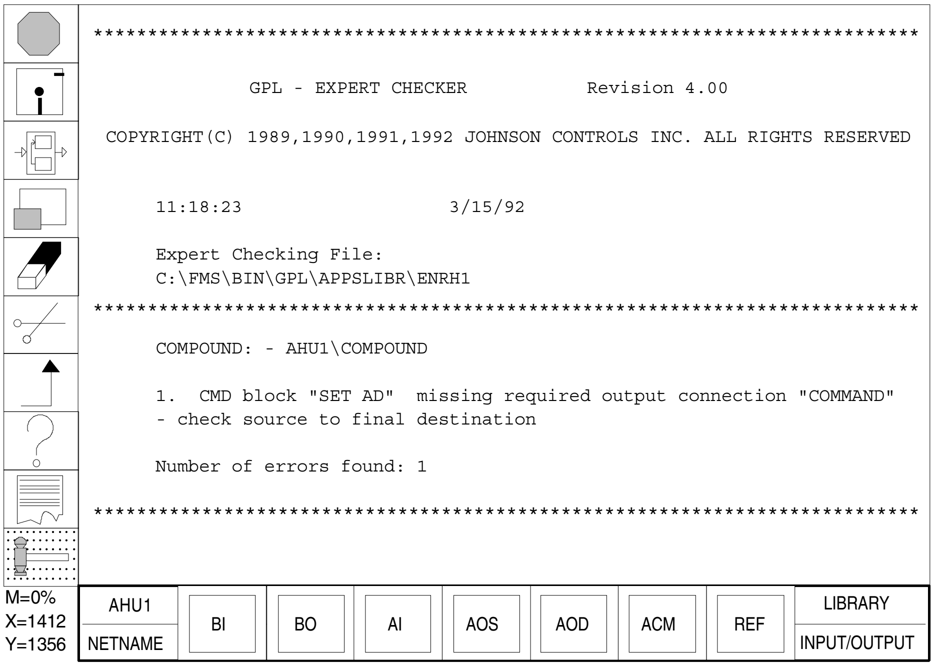
```
CLG last used as a subfile

12:15:02    2/16/90

See AHU1.LST for messages
```

The list file can also be written to by the Translator and Compiler. When the Expert Checker writes to the list file, it first clears the file. The list file header displays the time and date, strategy file name, and name of the program (e.g., Expert Checker, Translator, Compiler) that wrote the current information to the list file.

The list file is an ASCII text file. You can view the file with the GPL Editor, or print and view it with any text editor.

To view the list file in the GPL Editor, follow these steps:

1. Load the file you checked into the Editor. If the file is already displayed in the Editor, this step is not necessary.
2. Double-click left on the Query icon (question mark) to display the Query option menu.
3. Click left on the VIEW option in the Query option menu. The screen displays up to 30 pages of the list file for the currently loaded file. Here is an example of a list file (Figure 2).



```
****************************************************************************

              GPL - EXPERT CHECKER           Revision 4.00

 COPYRIGHT(C) 1989,1990,1991,1992 JOHNSON CONTROLS INC. ALL RIGHTS RESERVED


     11:18:23                  3/15/92

     Expert Checking File:
     C:\FMS\BIN\GPL\APPSLIBR\ENRH1
****************************************************************************
     COMPOUND: - AHU1\COMPOUND

     1.  CMD block "SET AD"  missing required output connection "COMMAND"
     - check source to final destination

     Number of errors found: 1

****************************************************************************
```

Figure 2: Example of List File

The errors found in the main file are listed first, followed by the errors found in subfiles. Within each file, the errors are organized by compound. The compound pathname appears as a heading, followed by the errors in the compound. The compound pathname is represented like a DOS pathname with the highest level compound first.

The total number of errors found is displayed at the end of the file.

If necessary, use the PageDn and PageUp keys to scroll to additional errors. The GPL Editor can display up to 30 pages (screens) of the list file. (If the list file is more than 30 pages, exit the GPL Editor and view additional pages in DOS.)

Error Messages

This section explains all error messages. There are eight types of errors found by the Expert Checker:

Fatal	USER Block
Initialization	Shared Variable
Operation Block	Diagrammatic
Object Block	File Nesting

All error messages except fatal error messages are saved in the list file.

In the list file and on your screen, the bracketed items will be replaced by real values from the actual file. For example, where the message in this section reads, `Cannot read list file, {FILENAME}.LST`, the bracketed portion will be replaced by the actual file name of the file you are attempting to check.

Fatal Errors

This type of error involves the inability to create or write to the disk, or the inability to allocate memory. A fatal error causes the Expert Checker program to immediately abort. The error message is displayed on the Expert Checker screen as soon as the error occurs. (This is the only type of error not saved in the list file.)

```
Cannot create .LST file:

{FILENAME}.LST
```

The Expert Checker cannot open a list file due to the disk being full, corrupt, or because of write protect constrictions.

```
Cannot create sub LIST file:

{FILENAME}.LST
```

The Expert Checker cannot open the subfile's .LST file because the disk is full, or because too many files are open.

```
Error Opening configuration file, GPL.CFG
```

The Expert Checker cannot open the configuration file (GPL.CFG) to get the name of the file to be checked. Disk may be corrupt.

```
Error while writing data base file
```

The Expert Checker cannot write to the data base file (after setting the Expert Checked flag).

```
Error while writing to .LST file:
{FILENAME}.LST
```

The Expert Checker cannot write to the list file because the disk is full or corrupt.

```
Error while writing to sub LIST file:
{FILENAME}.LST
```

The Expert Checker cannot write to the subfile’s .LST file because the disk is full.

```
No GPL File to Expert Check
```

The Expert Checker cannot find the name of the file to be checked in the GPL configuration file. The Expert Checker searches the GPL configuration file for the name of the file to be checked.

```
Out of memory for data base and connection RAM
```

The Expert Checker cannot allocate enough memory to load the GPL block data base.

```
Unable to allocate memory
```

The Expert Checker cannot allocate enough memory to perform operations necessary to continue.

Initialization Errors

This type of error may occur when the data base and connection files are loaded from disk. An Initialization error will stop the error checking process for the file being checked. The Expert Checker will continue checking nested subfiles.

```
Connection File not found, {FILENAME}.CI
```

```
Data base File not found, {FILENAME}.DB
```

The named file is missing.

```
Connection File not readable, {FILENAME}.CI
```

```
Data base File not readable, {FILENAME}.DB
```

The named file is corrupt.

Operation Block Errors

This type of error indicates a required connection may not have been made, a block may not have been grouped into a control process, or an illegal block is contained in a Restart process. In addition, certain types of blocks have special requirements such as interfield dependencies (for example, a PIR block's high range must be greater than low range) and types of blocks to which they can be legally connected. If these requirements are not met, the following Operation block errors will occur.

```
Hi input must be greater than lo input in SPAN block
{NAME}
```

```
Hi range must be greater than lo range in PIR block
{NAME}
```

There are conflicting interfield dependencies in the named operation block. These checks are made for default values entered in template fields only. The Expert Checker cannot check for conflicting interfield dependencies that result from runtime values.

```
Input connection {CONN} of CONN block {NAME} cannot
be marked exempt——mark exemption on connection to
final destination
```

An input connection to a CONN block has been marked exempt, creating an ambiguous condition. It is not possible to determine the destination for this exempt line. To avoid ambiguity, mark the final destination block as exempt.

```
Input Connection {CONN} of {OPER BLOCK TYPE} block
{NAME} must come directly from an object block
```

The input connection to a block is from an invalid block type. This message is generated if a TOT block has an input connection that is not directly from an object block.

```
{OPER BLOCK TYPE} block {NAME} cannot be used in a
Restart process.
```

An illegal block type has been found in a Restart process.

```
{OPER BLOCK TYPE} block {NAME} cannot have an ENA
OUT connection without an ENA IN connection.
```

A block has been found that has an ENA OUT connection without an ENA IN connection.

```
{OPER BLOCK TYPE} block {NAME} missing required
{INPUT/OUTPUT} connection {CONN}
```

The required connection is missing. {CONN} indicates the name of the missing connection.

```
{OPER BLOCK TYPE} block {NAME} must be in process
compound
```

A block has been found that is not in a process compound.

```
Output connection of {OPER BLOCK TYPE} block {NAME}
not triggerable, exemption undefined
```

A binary output that is not triggerable has been marked as exempt. Only triggerable binary variables can be made exempt.

```
Remote (input/output) connection unspecified for
{CONN} of {OPER BLOCK TYPE} block {NAME}
```

The named block has an unspecified remote input or output connection.

Object Block Errors

This type of error indicates one of three things for an object block: the block is not defined, a connection is illegal, or the connection creates conflicts.

```
Configuration lines from REF block {SYS\OBJ NAME——
BLOCK LABEL} to {SYS\OBJ NAME} are not allowed
```

A REF block may not be used as the source of a configuration connection, unless the block is connected to an AD or BD object block.

```
Conflicting configuration references for {REF NAME}
in {SYS\OBJ NAME} and (input/output) connection
{CONN} of {BLK TYPE} block {NAME or SYS\OBJ NAME}
{MSG}
```

A reference mismatch. The origin of a line connected as a configuration reference to a block conflicts with the reference system\object name in the block's data base template.

If a READ block is connected to the ASSOC IN connection of an AD or BD block, {MSG} will be replaced by the phrase "connection must come directly from object, not from READ attribute block." In all other cases, {MSG} will not appear.

```
Input connection {CONN1} of {SYS\OBJ NAME} must come
from output connection {CONN2} of block type {OBJ
BLOCK TYPE}
```

The input connection to a block is from an invalid source. The input connection must be a valid connection between the following block types: BO/BI, BO/BD, AOS/AI, AOS/AD, AOS/ACM, PIDL/AOD.

```
Port {CONN} of PIDL {NAME} configured NORMAL——output
connection can only come from PIDL OUT of PIDL block
```

A block other than a PIDL is connected to a PIDL port defined as Normal, or an output connection from a PIDL other than a PIDL OUT is connected to a PIDL port defined as Normal. When a PIDL port is defined as Normal, it can only be connected to another PIDL.

```
Remote (input/output) connection unspecified for
{CONN} of {SYS\OBJ NAME}
```

A remote connection to an input or output of an object block exists that is not referenced to an actual object block.

```
String not triggerable, exemption undefined for
{SYS\OBJ}
```

An attribute that has been marked as exempt is not triggerable.

```
Undefined {OBJ BLOCK TYPE} block found
```

An object is not defined, meaning the block templates are incomplete or have not been successfully saved to the archive data base.

USER Block Errors

This type of error indicates a USER block exists with the same TYPE/FILE name as another USER block but with different input/output configurations.

```
Configurations for USER Block {TYPE/FILE NAME} do
not match for {BLK NAME1} and {BLK NAME2}
```

There is a duplicate USER block in the file with the same TYPE/FILE name but a different number of inputs or outputs.

```
Output connection {CONN} for USER block {NAME}
cannot be fanned out
```

The output Write or Command connections of a USER block are fanned out.

Shared Variable Block Errors

This type of error indicates a Shared Variable (SVAR) block does not have a name associated with it, or that there is more than one SVAR block with the same name in the process.

```
SVAR block must have a block name
```

A SVAR block has been found without a block name.

```
SVAR block {NAME} may only be used once in process
{PRC CMP NAME}
```

A SVAR block occurs more than once in a process.

Diagrammatic Errors

This type of error indicates ambiguities in the control flow.

```
No loop defined between {BLK TYPE} block {NAME or
SYS\OBJ NAME} and {BLK TYPE} block {NAME or SYS\OBJ
NAME}
```

There are loop errors for the named operation blocks. The order of execution cannot be determined. There is either a master block without a loop or there are multiple loops with more than one master block.

File Nesting Errors

This type of error indicates that file nesting is not hierarchical.

```
Illegal file block nesting for file:
```

```
{FILENAME}
```

FILE blocks have been illegally nested. A subfile cannot reference a file that is previous to itself in the hierarchy of files. For example, if block A references block B, and block B references block C, block C cannot reference block A.

-METASYS-™

GPL Programmer's Manual

Simulator

* Indicates those sections where changes have occurred since the last printing.

Code No. LIT-631050

* Indicates those sections where changes have occurred since the last printing.

Overview

This chapter describes Simulator functions and contains instructions and a hands-on tutorial that quickly teach you how to use the Simulator.

You should be familiar with using the Editor before using the Simulator. You'll find more information in the *Editor* chapter.

Purpose of Simulator

The Simulator is a software utility that allows you to test the functional operation of GPL control strategies, which include process objects and objects. Use the Simulator to test control strategies and the behavior of objects before translating the strategies into processes and downloading the processes into an NCM.

The Simulator allows you to test control strategies in a safe environment. In addition, the Simulator gives you extensive control over the execution of processes. This means that you can simulate various conditions (e.g., offline and unreliable states) to test how the logic will respond.

The Simulator allows you to:

- simulate execution of the processes within the file
- examine the data throughout in the process data flow
- test a section of a process under all conditions
- simulate the interaction of objects with processes and other objects
- temporarily change object attributes to simulate different runtime states
- view how your changes affect subsequent function blocks
- save any changes made for future simulation sessions

Simulator and NCM Execution

The Simulator attempts to match as closely as possible the behavior of blocks and processes in the NCM. Like the NCM, the Simulator executes function blocks from “upstream to downstream” in the process data flow. When executing a function block, the Simulator evaluates the block’s current inputs, applies the block’s algorithm to the input values, and generates output values. These output values then become the input values for the downstream function blocks.

Automatic and Manual Modes

In the Simulator, blocks have two modes: Automatic mode (default) and Manual mode. When a block is in Automatic mode, it executes normally. When a block is in Manual mode, you can manipulate the data fields and outputs of the block to simulate different conditions. Then, when the block is executed, you can view the impact of your changes on downstream blocks. You can repeat this cycle of switching between Automatic and Manual modes, modifying function block outputs, and observing the results until you are satisfied with the outcome.

Getting Started

This section introduces you to the procedures you'll need to know to use the Simulator effectively. This section includes:

- Preparing a Strategy File for Simulation
- Starting the Simulator
- The Simulator Screen
- Moving Around the Simulator Screen
- Changing a Block's Data or Outputs
- Exiting the Simulator

The tutorial following this section guides you through the process of simulating a control strategy.

Following the tutorial are sections explaining all Simulator functions.

Preparing a Strategy File for Simulation

Preparatory steps for simulating a file include pasting analog displays, typing text, and expert checking the file. All three steps are optional, but recommended. Paste down analog displays to help you easily view the changing analog values that occur during simulation. Type text as needed to clearly identify the outputs and analog displays in the diagram. Expert check a file to assure that the file to be simulated is without errors. Perform the following steps:

1. Load the control strategy file into the work area of the Editor.
2. Paste down analog displays using the 0.00 option in the Tools option menu. Paste down explanatory text using the TEXT option in the Tools menu. Both of these functions are described in the *Tools Functions* section of the *Editor* chapter.
3. Save the file with the pasted analog displays and explanatory text.

4. Run the file through the Expert Checker. A file should only be simulated after it has been run through the Expert Checker and found to be free of errors.

Starting the Simulator

To start the Simulator:

1. If you haven't loaded the control strategy file into the work area yet, do so now.

2. Double-click left on the Tools icon (hammer) to display the Tools option menu (Figure 1).

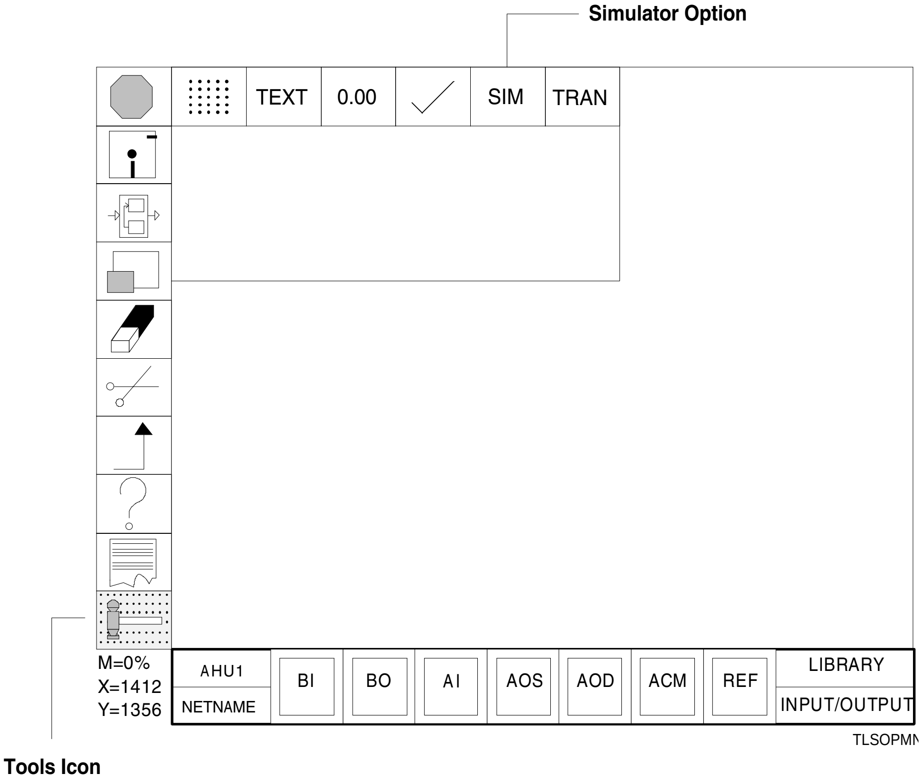


Figure 1: Tools Option Menu

3. Click left on the SIM option in the Tools menu. The screen clears and the message, `Initializing Simulator...please wait,` appears briefly. Then the Simulator screen appears (Figure 2).

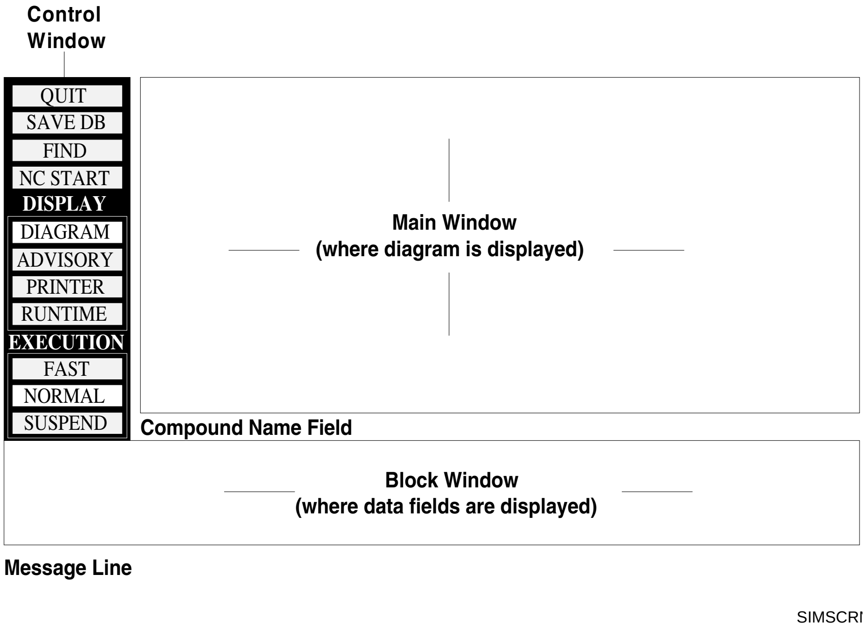


Figure 2: Simulator Screen

The Simulator Screen

As shown in Figure 2, the Simulator screen is divided into four major areas: the Main window, the Control window, the Block window, and the Message line.

Main Window

The Main window displays the control strategy diagram you selected. The diagram appears as it does in the Editor, with the exception of text, which may be slightly shortened in length.

If the function blocks displayed in the diagram are the contents of a compound, a green box appears around the diagram. As in the Editor, you can move into and back out of compound levels. To move into a compound, double-click the compound block in the diagram. To back out of a compound, double-click the compound name field in the lower left corner of the Main window.

In the Main window, the currently selected block is highlighted. Use the mouse to select a block.

The Main window displays some of the results of simulation. For example, where analog displays have been pasted down, you can quickly view changing analog values.

In the Main window:

- Blocks in Manual mode are drawn in yellow.
- True (closed/1/yes) binary connection lines are green.
- False (open/0/no) binary connection lines are red.
- Unreliable connection lines flash blue every half second.
- If a connection line is a possible trigger, its connecting arrow is outlined in light cyan.
- If a connection line is a possible trigger, but originates from an object with locked triggers, its connecting arrow is outlined in yellow.
- The order of execution for each operation block in a process appears in the upper left corner of the operation block. For example, the number 3 appears in the upper left corner of the operation block that is third in the execution order.

Control Window

The Control window (Figure 2) displays the following Simulator options.

MENU OPTION	FUNCTION
QUIT	Exits Simulator and returns to Editor
SAVE DB	Saves changes made to data fields for future Simulator sessions
FIND	Displays a list of all function blocks in the file and allows selection
NC START	Simulates either a warm or cold NC start
DIAGRAM	Redisplays diagram after displaying messages
ADVISORY	Displays Advisory messages
PRINTER	Displays Print messages
RUNTIME	Displays Runtime messages
FAST	Executes timing functions at ten times the normal speed
NORMAL	Executes timing functions at normal speed
SUSPEND	Stops executing timing functions

On the Simulator screen, the DIAGRAM, ADVISORY, PRINTER, and RUNTIME options are grouped under a DISPLAY heading, and the FAST, NORMAL, and SUSPEND options are grouped under an EXECUTION heading. These and other options are described in detail in *Using the Control Window.*

Block Window

The Block window (Figure 2) displays data specific to one selected block. In the Block window, you can view the current block values, put a block in Manual mode, and modify some of the block's outputs and data fields. By modifying block outputs and data fields, you can simulate various conditions and see the results on downstream logic.

The colors of the fields in the Block window indicate the type of data to be viewed or entered in the fields.

COLORS	TYPE OF DATA
Brown	Time value
Green	Tab fields
Light Cyan	Binary value
Magenta	String value
Yellow	Analog/Integer value

Message Line

The bottom line of the screen displays the name of the file currently being simulated, the function keys currently available, and user messages.

The file name appears to the far left. (The file name is the strategy file name.)

Following the file name are the names of currently available function keys. You can select the named functions either by pressing the appropriate function key, or by using the mouse. To use the mouse, move the mouse cursor over the name of the function you want to perform. Click left to perform the function.

User messages overwrite the file name and function key display when they occur. The user messages appear when you perform certain actions during simulation. For example, if you attempt to put an object block into Automatic mode and the object has interfield configuration errors, a message appears in this line and explains the error. (See *User Messages*.)

Gray Scale Setup for Portable Workstation

To insure that the Portable Workstation clearly displays the simulation information, use the following gray scale setup. You'll find information on setting up the gray scales in your Portable Workstation manual.

When setting up the gray scales for the Portable Workstation, use the following gray scale continuum:

1----2----3----4----5----6----7----8

White Black

COLORS	PORTABLE SHADESETTING
Green, Yellow, Bright White	1 (White)
Black, Blue	4 (Light Gray)
Cyan, Magenta, Brown, White, Dark Gray, Light Blue, Light Green, Light Red, Light Magenta	7 (Dark Gray)
Red, Light Cyan	8 (Black)

Portable Operator Workstation Simulator Screen

Using the suggested gray scale settings, the Portable Operator Workstation display has the following differences:

- Blocks in Manual mode are displayed in high intensity.
- Blocks in Automatic mode are displayed in low intensity.
- True binary connection lines are white.
- False binary connection lines are black.
- Unreliable connection lines flash on and off every half second.

Analog Displays in the Simulator

The Simulator uses analog displays to represent analog output values of the blocks. (Analog displays, however, cannot represent time values.) Using the Editor, you can paste down these displays on the diagram. However, the analog display in the Main window may show something other than expected. This would occur if you did not specify the precision and width of the analog display correctly.

Precision: Number of digits to the right of the decimal point (e.g., 0.00 has a precision of two).

Width: Total number of digits including the decimal point (e.g., 0.00 has a width of four).

With these definitions in mind, note the following points to consider when values are displayed in the Main window:

Precision and Width: If the value to display is large and requires more digits to the left of the decimal point than specified in the analog display, precision is sacrificed, so that width is not exceeded. Example:

SPECIFIED DISPLAY	VALUE TO DISPLAY	ACTUAL VALUE DISPLAYED
0.00	17.48	17.5

Overflow: If the value is so large that it requires more digits to the left of the decimal point than the total width of the specified analog display, the value is shown as asterisks. Also, the sign of the number (+ or -) appears in front of the asterisks. Example:

SPECIFIED DISPLAY	VALUE TO DISPLAY	ACTUAL VALUE DISPLAYED
0.00	10000	+****

Underflow: If the value is so small that all digits shown are zero, the value displayed is zero, preceded by the sign of the value. Example:

SPECIFIED DISPLAY	VALUE TO DISPLAY	ACTUAL VALUE DISPLAYED
0.00	0.001	+0.00

If an analog display is not specified correctly, use the Editor to change the display. Select the Move icon (Scissors), click left on the display, and click the right/middle button until an appropriate display is shown.

Analog values displayed in the Block Window use the 0.00000 format. Note that the Simulator does not use the Decimal Position field, which is a template parameter for the object blocks.

Moving Around the Simulator Screen

The mouse is active on the entire Simulator screen. For example, use the mouse to select a block, select Control Window options, and toggle between Automatic and Manual modes. Many of the mouse functions have equivalent keyboard functions, such as moving to a data field, which can be done with the mouse or the arrow keys. Some of the equivalent functions require you to press ALT and a letter simultaneously.

Both the left and right buttons of the mouse are used. For example, the left button is used to select a data field, and the right button is used to toggle between the various states of this field.

The following tables explain the keystrokes and mouse actions used in the Main, Control, and Block windows, and in the Message line.

KEYSTROKES	
	FUNCTION
Alt Key and Highlighted Letter of Option	Performs option (e.g., press ALT/Y for Yes and ALT/T for CLR TOP)
Alphanumeric Character Set	Specifies values in fields
Arrow Keys	Move between data fields to select a field
PageUp/PageDn	Displays multiple pages of data fields
Tab	Scrolls through possible entries in some data fields
Enter	Enters new data in currently selected field
Esc Key	Cancels the selected control option while the verify or selection box is displayed
F4	Toggles between Automatic and Manual modes
F5	Triggers execution of the process displayed in Block window
F7	Sets all blocks in the file into Manual mode
F8	Sets all blocks in the file into Automatic mode

IN THE MAIN WINDOW	
MOUSE ACTION	**FUNCTION**
Click Left on Block	Displays selected block's data in Block window
Click Left on Verify or Selection Box	Performs the selected option
Click Right on Block	Toggles the manual and automatic state of the selected block
Double-Click Left on Compound Block or Compound Name Field	Moves into or backs out of compound levels
Mouse Movement	Moves highlight around diagram to select block

IN THE CONTROL WINDOW	
MOUSE ACTION	**FUNCTION**
Click Left	Performs option
Mouse Movement	Moves between Simulator options

IN THE BLOCK WINDOW	
MOUSE ACTION	**FUNCTION**
Click Down Arrow	Displays next page
Click Left	Selects a data field for user entry
Click Right	Toggles between two binary entries or TAB entries
Click Up Arrow	Displays previous page
Mouse Movement	Highlights a data field for selection

IN THE MESSAGE LINE	
MOUSE ACTION	**FUNCTION**
Click Left	Performs function under mouse cursor
Mouse Movement	Moves mouse cursor between function key options

Changing Block's Data or Outputs

The following steps tell you how to perform one of the most basic Simulator functions: changing a block's data fields in the Block window. For a technical explanation of what the Simulator does under various conditions, see *Simulator Functions*.

To change a function block's data or outputs, you must perform two tasks: (1) put the block in Manual mode, and (2) change the output or data fields in the Block window. These two tasks are described below.

Putting a Block in Manual Mode

1. In the Main window, use the mouse to select the block you want to put in Manual mode. The selected block is highlighted.
2. Click left on the block. The data fields for the selected block are displayed in the Block window.
3. Click right on the MANUAL ? field (or press F4) to put the block in Manual mode.

   Use either the right mouse button or F4 key to toggle the MANUAL ? field between N (no) and Y (yes). If the block is in Automatic mode, the field displays N. If the block is in Manual mode, the field displays Y. Automatic mode (N) is the default.

   When the block is in Manual mode, you can change the outputs and modifiable data fields. In the Block window, the modifiable fields are shown in high intensity.

   In the Main window, the block that you put in Manual mode is drawn in yellow.

   You can leave a block in Manual mode or put it back in Automatic mode before executing the process containing the block. Processes can be executed with blocks in both Manual and Automatic modes.

Changing Data in the Block Window

One data field in the Block window is always highlighted. This indicates that you can use the keyboard to modify the data in that field.

1. While the block is in Manual mode, use the mouse to highlight the field you want to change. A green highlight box appears around the field to show it is selected.
2. Click left. The data in the field is back-highlighted.
3. Either type the new value over the old value and press Enter, or click right (or press Tab) to scroll through the possible values. When you type characters, they appear red until you press Enter, when they change to blue.

   The scroll function is not available with all fields. With scroll fields, you do not need to press Enter.

   If the value you enter is out of range, a beep will sound, and the field will change back to the original value.

Returning a Block to Automatic Mode

You can execute a process that contains blocks in both Manual and Automatic modes, as long as the process block itself is in Automatic mode. Therefore, returning a block to Automatic mode is optional.

To return a block to Automatic mode:

1. Display the block's data fields in the Block window.
2. Click right on the MANUAL ? field (or press F4). This action toggles the block between Automatic and Manual modes. When the block is in Automatic mode, N appears in the MANUAL ? field.

Exiting the Simulator

To exit the Simulator:

1. Click left on the Quit option on the Control window (or press ALT/Q). The QUIT ? verify box displays.
2. Click left on YES (or press ALT/Y). This exits the Simulator and displays the Editor screen.

Tutorial

This hands-on tutorial guides you through the basic steps of simulating a GPL control strategy to test its logic. This tutorial introduces the Simulator functions you are most likely to use. You'll find information on all Simulator functions in the sections following this tutorial.

By completing this tutorial, you will learn how to:

- Display a block's data fields in the Block window.
- Put a block in Manual mode.
- Trigger execution of a process.
- Modify a block's outputs.
- View results of the modification.
- Save changes made to outputs and object data fields in the Simulator.
- Exit the Simulator.

Starting the Tutorial

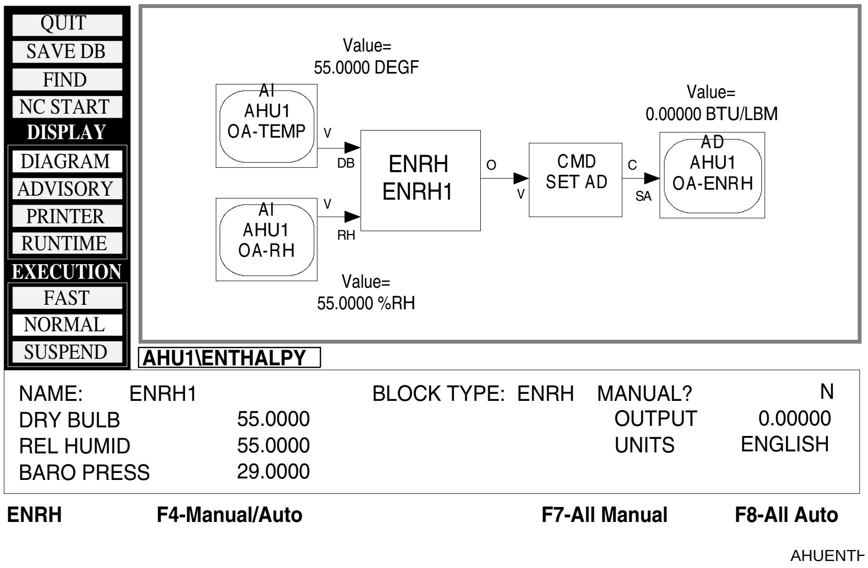
As a prerequisite for using this tutorial, you should have completed the tutorial in the Editor chapter. The Simulator tutorial uses the same process compound you created in the Editor tutorial. You saved this process compound as the ENRH file.

To start the tutorial:

1. Load the ENRH file into the work area of the Editor.
2. Double-click left on the AHU1\ENTHALPY process compound to display its contents.
3. Prepare the ENRH file for simulation as described in *Getting Started.*

   While preparing the file, paste down analog displays next to the AHU1\OA-TEMP, AHU1\OA-RH, and AHU1\OA-ENRH blocks (Figure 3). Associate the analog displays with the VALUE attributes of the objects.

Next to the AHU1\OA-TEMP analog display, type text strings stating **Value =** and **Deg F**. Next to the AHU1\OA-RH analog display, type text strings stating **Value =** and **% RH**. Next to the AHU1\OA-ENRH analog display, type text strings stating **Value =** and **BTU/LBM** (as shown in Figure 3).



NAME: ENRH1 BLOCK TYPE: ENRH MANUAL? N
DRY BULB 55.0000 OUTPUT 0.00000
REL HUMID 55.0000 UNITS ENGLISH
BARO PRESS 29.0000

Figure 3: AHU1\ENTHALPY Process Compound Displayed in Simulator Before First Execution of Process

4. Change the initial value of the AHU1\OA-ENRH block to 0.00000.

   To change the initial value, display the block's template and go to the INITIAL VALUE field (second page of the template). Type `0.0` and press Enter. Then save the block by pressing F10.

5. Save the revised file by clicking left on the File icon (Disk) and then clicking left on the SAVE option.

6. Run the file through the Expert Checker to make sure it is free of errors.

7. Start the Simulator by double-clicking on the Tools icon and clicking left on the SIM option. The process compound should appear on the Simulator screen as shown in Figure 3. If the ENRH1 fields are not displayed in the Block window, use the mouse to select the ENRH1 block and click left.

In this tutorial, you'll simulate the process with initial block values. Then you'll modify the AHU1\OA-TEMP object block's value and simulate the process again to see the effect on the AHU1\OA-ENRH object's value.

For a complete explanation of the operation of the AHU1\ENTHALPY process, including a description of the blocks, review the tutorial in the *Editor* chapter.

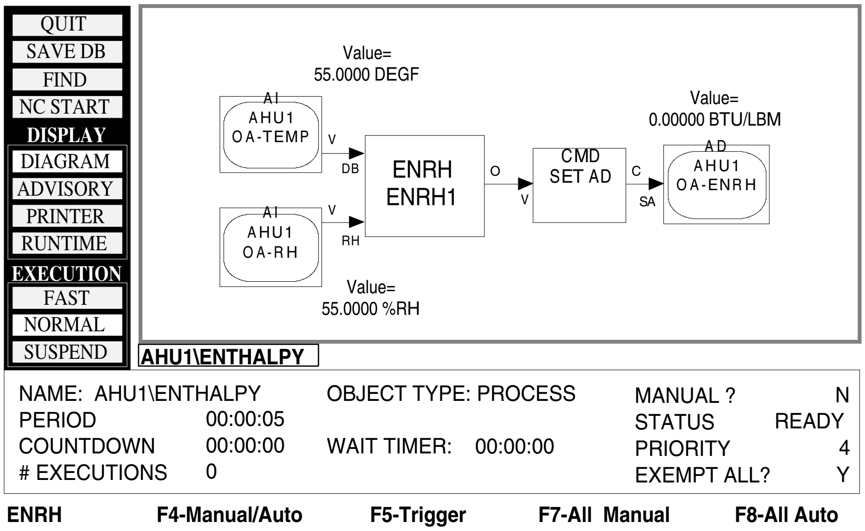
Executing a Process

In these steps, you'll execute the process to see the value that results in the AHU1\OA-ENRH block when the process executes with initial values. Whenever an object is simulated for the first time, certain data fields are automatically initialized. In this case, the value attributes of the AI objects will be initialized to their setpoints, and the value of the AD object will be initialized to its initial value.

To execute the process:

1. Display the AHU1\ENTHALPY process compound data fields in the Block window. To do this, click left on the Compound Name field in the lower left corner of the Main window (the Compound Name field displays the system\object name of the process).

   The screen appears as shown in Figure 4. The cursor appears on the MANUAL ? field in the upper right corner of the Block window.



NAME: AHU1\ENTHALPY
OBJECT TYPE: PROCESS
PERIOD 00:00:05
COUNTDOWN 00:00:00
WAIT TIMER: 00:00:00
# EXECUTIONS 0
MANUAL ? N
STATUS READY
PRIORITY 4
EXEMPT ALL? Y

prcmpd

Figure 4: Process Compound Data Displayed in Block Window

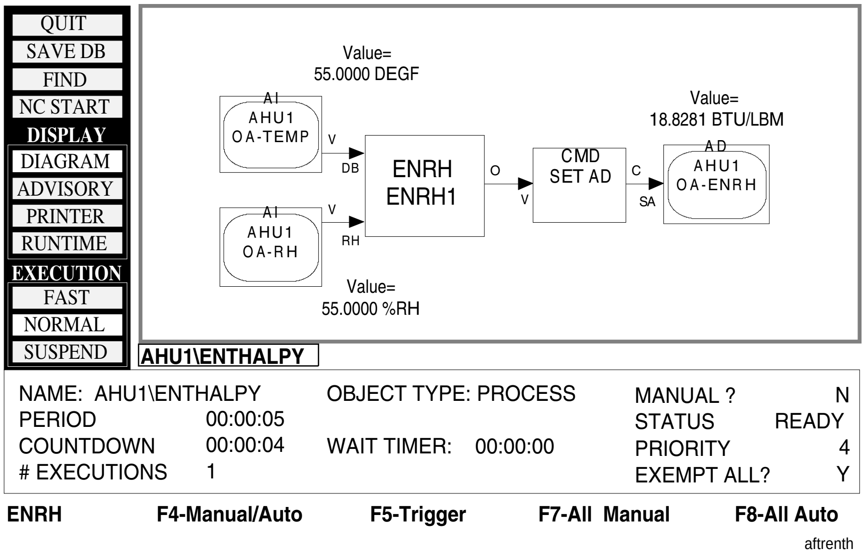
2. With the mouse, highlight the MANUAL ? field and click right (or press F4) to put the process block called AHU1\ENTHALPY in Manual mode. When a block is in Manual mode, the MANUAL ? field displays Y and modifiable data fields are shown in high intensity.

3. Click left on the STATUS field. Repeatedly click right (or press Tab) to scroll through the possible states (e.g., READY, WAIT, ERROR, DISABLED) until TRIGGER is displayed.

4. When TRIGGER is displayed in the STATUS field, put the block back in Automatic mode.

The process immediately executes. Note that the COUNTDOWN field in the lower left corner of the Block window counts down from the specified period length of 00:00:05 (5 seconds). The process executes whenever the countdown timer reaches 00:00:00.

Here is how the process diagram looks after execution (Figure 5).



NAME: AHU1\ENTHALPY
PERIOD 00:00:05
COUNTDOWN 00:00:04
EXECUTIONS 1
OBJECT TYPE: PROCESS
WAIT TIMER: 00:00:00
MANUAL ? N
STATUS READY
PRIORITY 4
EXEMPT ALL? Y

Figure 5: AHU1\ENTHALPY Process Compound After Execution of Process

After executing the process with its initial values, the AHU1\OA-ENRH block displays a value of 18.8281. This is the value of the outdoor air enthalpy.

The enthalpy values arrived at in this tutorial assume that the ENRH1 block is using the default value of barometric pressure (29.0000 in. Hg).

Process Execution

The following occurs during execution of the process:

1. The ENRH1 block reads the values of dry bulb temperature and relative humidity from the AI blocks, performs its algorithm, and generates an output value, which represents outdoor air enthalpy.

2. The CMD block reads the output value from the ENRH1 block and sends a SET AD command to the AHU1\OA-ENRH object block.
3. The AHU1\OA-ENRH object block responds to the SET AD command and changes its value to the value sent from the CMD block.

Modifying a Block Output

In this exercise, you'll modify the value in the AHU1\OA-TEMP block (dry bulb temperature) and see the results in the AHU1\OA-ENRH block.

To modify the value in the AHU1\OA-TEMP block:

1. In the Main window, click left on the AHU1\OA-TEMP block to display its data fields in the Block window.

   The cursor appears on the MANUAL ? field in the upper right corner of the Block window.

   The default N (for no) appears in the MANUAL ? field, indicating the block is in Automatic mode.
2. Highlight the MANUAL ? field and click right (or press F4) to put the block in Manual mode. A Y (yes) appears in the field. In the Main window, the AHU1\OA-TEMP block is drawn in yellow, indicating it is in Manual mode.
3. Click left on VALUE field. The value 55.0000 appears highlighted.
4. Type the value 100.00 over the old value. Press Enter. You just changed the value of the outdoor air temperature (AHU1\OA-TEMP) to 100.0°F.

When the process executes again (within five seconds), the value of the AHU1\OA-ENRH block changes from 18.8281 to 50.1914 BTU/lb as a result of changing the AHU1\OA-TEMP value.

Saving Changes Made in the Simulator

After making changes to object data fields and operation block outputs, you might want to save the changes for later simulation. Use the SAVE DB option in the Control window to save the changes.

If you save changes to object blocks in the Simulator, the changes will appear in the block fields the next time you load the file into the Simulator. However, if you load the file into the Editor and perform any function that causes the Editor to read the archive data base, the changes will be overwritten with the values from the archive data base. For example, if you perform a Query on a block, any changes to the block that you made in the Simulator will be overwritten by the archive values. If the strategy file is then saved in the Editor, all Simulator changes for that object block will be lost.

You can transfer the object block changes you make in the Simulator to the archive data base. This is described in the *Using the Control Window* section of this chapter.

Changes to operation block outputs cannot be transferred to the archive data base and downloaded into the NCM. These values are used by the Simulator only.

When blocks are copied in the Editor, any saved simulation changes to the original blocks will appear in the copies as well.

To save the changes you made in the Simulator:

1. Click left on SAVE DB on the Control window. The SAVE DB ? verify box displays.
2. Click left on YES (or press ALT/Y). The changes you made to the outputs and object block data fields are saved.

Exiting the Simulator

To exit the Simulator and return to the Editor:

1. Click left on the Quit option on the Control window (or press ALT/Q). The QUIT ? verify box displays.
2. Click left on YES (or press ALT/Y). This exits the Simulator and returns you to the Editor screen. The diagram appears as it did before you entered the Simulator.

You are now finished with the Simulator tutorial. The next section explains all the options in the Control window.

Using the Control Window

This section explains the following options in the Simulator Control window:

- QUIT
- SAVE DB
- FIND
- NC START
- DIAGRAM
- ADVISORY
- PRINTER
- RUNTIME
- FAST
- NORMAL
- SUSPEND

To select an option from the Control window, move the mouse to position the cursor over the option's box and click left. Or, simultaneously press the ALT key and type the highlighted letter of the option; for example, "ALT/F" for FIND or "ALT/Q" for QUIT.

QUIT

Use the QUIT option to exit the Simulator and return to the Editor. When you select QUIT, all simulation is halted.

To exit the Simulator:

1. Click left on the Quit option on the Control window (or press ALT/Q). The QUIT ? verify box displays.
2. Click left on YES (or press ALT/Y). This exits the Simulator and returns you to the Editor screen. The diagram appears as it did before you entered the Simulator.

SAVE DB

Use the SAVE DB option to save the changes you made during simulation to object data fields, operation block outputs, and special block outputs. When you select SAVE DB, all simulation is halted.

To save the changes, position the cursor over the SAVE DB option and click left (or press ALT/S). The SAVE DB ? verify box displays. Click left on the YES option (or press ALT/Y).

In the Editor, if you perform any function on an object block with saved Simulator changes that reads the archive data base for that object, the saved Simulator changes are overwritten by the values from the archive data base. For example, if you query a defined object block with changes, the changes are overwritten because the Query function reads the archive data base.

If you do not perform a function on an object block (in the Editor) that reads the archive data base, the saved Simulator changes for that object block appear the next time you load the file into the Simulator.

If you load undefined object blocks into the Simulator, change their data fields, and then save the changes, the changes will appear as the block defaults in the Editor.

The Manual mode Yes/No flags, the connection reliability flags, and changes to operation and special block outputs are not overwritten in the Editor. These values are used by the Simulator only and cannot be saved to the archive data base.

Advisory, printer, or runtime messages generated during simulation are not saved, even when you use the SAVE DB option.

Transferring Simulator Changes to Archive Data Base

In the Simulator, you might make changes to object block attributes (such as setpoint) that you want to transfer to the archive data base because they improve the functioning of the process.

The set of transferrable configuration attributes are those that appear in both the object block's template (in the Editor) and in the object's Block window data fields (in the Simulator).

Note: The PERIOD field in the Simulator represents the current period of a process object. The initial period, specified in the Editor template, cannot be changed from the Simulator.

To transfer changes to the archive data base:

1. Save the changes with the SAVE DB option in the Simulator.
2. Go to the Editor.
3. Delete the object from the archive data base (but not from the diagram).
4. Query the object with the changes. The changes you made in the Simulator appear in the object's template.
5. Save the object by pressing F10. The object is added again to the archive data base with the changes you made in the Simulator.

Here is another way to transfer changes to the archive data base:

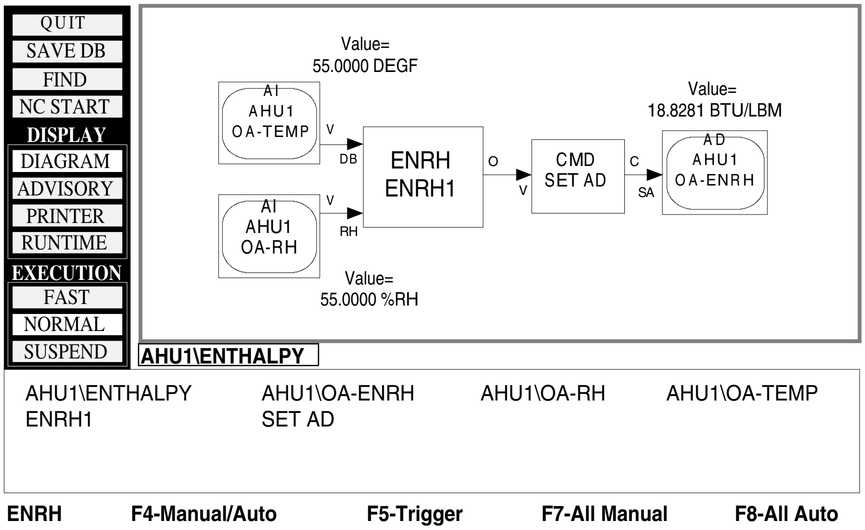
1. Load a file with undefined object blocks into the Simulator.
2. Make the desired changes to the undefined object block's configuration attributes.

3. Simulate the file until you are satisfied with the outcome.
4. Save the changes with the SAVE DB option.
5. Go to the Editor.
6. Query the undefined object with the changes. The changes you made in the Simulator appear in the object's template.
7. Save the object by pressing F10. The object is saved to the archive data base with the changes you made in the Simulator.

FIND

The FIND option displays a list of all named blocks in the file. You can select a block from this list to display in the Block window.

For example, if you select FIND when the ENRH file (used in the tutorial) is loaded in the Simulator, the following list of blocks appears in the Block window (Figure 6).



AHU1\ENTHALPY AHU1\OA-ENRH AHU1\OA-RH AHU1\OA-TEMP
ENRH1 SET AD

FNDFNC

Figure 6: Find Function Displaying All Blocks in ENRH File

If there are more than 16 blocks in the file, click left on the up and down arrows in the right corner (or use the PageDn and PageUp keys) to display additional pages of blocks.

If more than one block has the same name, the name appears for each occurrence. For example, if two blocks have the same name, the name appears twice.

To select a block and display its data in the Block window, position the mouse over the block name. A green highlight box appears, indicating it is selected. Then click left. The selected block's data fields display in the Block window.

The names of group compounds and process objects appear yellow. To list the blocks in one compound, position the cursor over the compound name and click right. Only those blocks in the selected compound are listed. To display block data for the compound in the Block window, position the cursor over the compound name and click left.

You can use the FIND option to trigger processes other than the one displayed in the Main window. To do so, click left on the process system\object name in the list of blocks. The process's data fields display in the Block window. Then, while the process is in Automatic mode, press F5 to trigger the process. (Or, trigger the process by putting the process in Manual mode, changing the status to TRIGGER, and returning the process to Automatic mode.)

NC START

The NC START function simulates a cold or warm start of the NCM following a download, reboot, or power failure.

Note: The NC START command is a Simulator command only; it is not a command you can issue to an actual NCM.

When you select NC START from the Control window, a box appears prompting you to select either a cold or warm restart. Click the button that corresponds to the type of restart you want to simulate. To cancel the NC START function, click outside the box (or press ESC).

The Simulator assumes that all processes and objects reside on the same NCM, meaning that all objects will be affected by the NC START. To exclude a process or object from the NC START operation, put it in Manual mode before selecting NC START.

The only way to trigger a Restart process is with the NC START function. (The Restart process cannot be triggered with the F5 function key.)

Cold and Warm Start

As in the NC, the following actions take place for both a cold and warm NC START:

- All local variables contained within processes (e.g., operation block outputs, and temporary values such as first pass flags and DIFF comparisons) are set to 0/0.0/FALSE/00:00:00 and reliable.
- The Restart process, if it exists and is in Automatic mode, begins execution while other enabled processes are put into the Held state.

  When the Restart process successfully completes, all processes that were in the Held state are put into the Trigger state and triggered in priority order. After triggering once, periodic processes will execute according to their defined periods.

  If the Restart process error locks, other processes remain in the Held state.
- If no Restart process exists, or it is in Manual mode, all enabled processes are immediately put into the Trigger state and executed in priority order.

Cold Start Only

The following actions take place for a cold NC START only:

- All shared variables (SVAR blocks) in the file are initialized to 0/0.0/FALSE/00:00:00 and reliable.
- All object runtime attributes that are not directly obtained from the hardware or features are set to their default values.

  The attributes Command Priority, Reports Locked, and Trigger Locked are defaulted.

  Value attributes of objects are set to their Initial Values, if the Initial Value attribute applies. Otherwise, Value attributes remain unchanged.

  Change of state analysis is performed so that other runtime attributes (e.g., S/W Override, alarm) are synchronized with the Value and Command Priority attributes.
- The priority of all 12 PIDL ports is defaulted to 3, and the port values are set to their priority 3 values. The PIDL algorithms run once, to synchronize all other attribute values.

What the NC START Does Not Change

During simulation of an NC START, the following values are left unchanged:

- shared variables (SVAR blocks) that are in Manual mode
- all shared and local variables originating from operation blocks in Manual mode
- process compound blocks in Manual mode, and all operation blocks within them
- attributes of objects in Manual mode
- attributes of objects that are directly obtained from hardware. These include Offline, HOA, and LC HOA flags, and the Value attribute and Reliable flags of the input objects (ACM, AI, BI, LCG, MSI, ZONE). These attributes maintain their values following an NC START.
- the Early Time and Late Time attributes of the BD, BO, MSD, and MSO objects. These objects maintain their values following an NC START because features are not simulated.

DIAGRAM

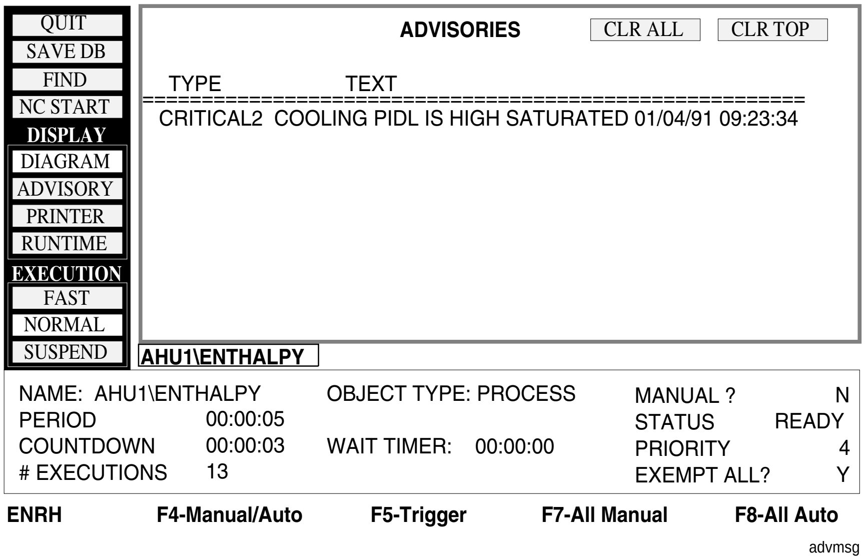
Use the DIAGRAM option to redisplay the GPL diagram after you have used the ADVISORY, PRINTER, or RUNTIME option to display messages. When you display messages, the diagram in the Main window is cleared and replaced with the message list.

ADVISORY

Use the ADVISORY option to display messages generated by Advisory (ADV) blocks. These messages are generated only when ADV blocks are executed.

To display advisory messages, position the cursor over the ADVISORY option on the Control window and click left. (Or, press ALT/V.)

The diagram in the Main window clears and generated messages are displayed on the screen. The following figure is an example of an advisory messages screen.



```
  TYPE              TEXT
========================================================
 CRITICAL2  COOLING PIDL IS HIGH SATURATED 01/04/91 09:23:34
```

```
NAME:  AHU1\ENTHALPY         OBJECT TYPE: PROCESS        MANUAL ?            N
PERIOD           00:00:05                                STATUS          READY
COUNTDOWN        00:00:03    WAIT TIMER:    00:00:00     PRIORITY            4
# EXECUTIONS     13                                      EXEMPT ALL?         Y
```

Figure 7: Screen Displaying Advisory Messages

The advisories screen displays the type of message (e.g., CRITICAL1, CRITICAL2), the message text (as much as can be shown), and the time and date the message was generated. If an input is connected to the ADV block, its value is printed at the end of the message text.

PRINTER

Use the PRINTER option to display printer messages generated by Print (PRNT) blocks. These messages are generated only when PRNT blocks are executed.

To display printer messages, position the cursor over the PRINTER option on the Control window and click left. (Or, press ALT/P.)

The printer screen displays the object name of the printer receiving the message, the message text (as much as can be shown), and the date and time the message is generated (optional). The information to be displayed with the message is specified in the PRNT block's template. If an input is connected to the PRNT block, its value is printed at the end of the message text.

RUNTIME

Use the RUNTIME option to display runtime and fatal errors generated by the execution of operation blocks.

To display runtime error messages, position the cursor over the RUNTIME option on the Control window and click left. (Or, press ALT/R.)

The runtime messages screen displays the process system\object name, block type, the message text, and the time the runtime or fatal error message was generated.

Message Similarities

Advisory, printer, and runtime messages generated during simulation cannot be saved, even when you use the SAVE DB option. In addition, messages cannot be sent to a printer.

Each message is limited to one line on the screen. The message text may be truncated so that other information can be fully displayed. If the message text is truncated, ellipsis dots (...) will follow the text.

Each message screen can display up to 21 messages. The newest messages are displayed at the bottom. If the 20-second message is generated, the old message at the top of the screen is cleared to make room for the new message.

The messages are displayed in different colors to distinguish between the different data types being displayed. The following table lists the different colors by data type.

COLORS	TYPE OF DATA
Brown	Time (ADVISORY, PRINTER, RUNTIME)
Cyan	Date (ADVISORY, PRINTER) Type (RUNTIME non-fatal)
Gray	Value (ADVISORY, PRINTER)
Green	Text (ADVISORY, PRINTER, RUNTIME)
Red	Type (RUNTIME fatal)
Yellow	Type (ADVISORY) Printer (PRINTER) Process (RUNTIME)

Clearing Messages

To clear all messages from the screen, click left on the CLR ALL option in the Main window (or press ALT/C). To clear only the top message, click left on CLR TOP (or press ALT/T).

Returning to the GPL Diagram

To redisplay the GPL diagram, click left on the DIAGRAM option in the Control window. The diagram is displayed at the same compound level that was showing before you selected one of the message's options.

FAST Execution Mode

Use the FAST option to accelerate the execution of time functions. When FAST is selected, the Simulator runs at ten times the normal speed, meaning one second of Simulator time is equal to ten seconds of real time. Using the FAST option allows you to quickly see how processes will run over an extended period.

When FAST mode is selected, the same screen information is displayed as in NORMAL mode. However, the values update faster.

To leave FAST mode, select either the NORMAL or SUSPEND execution option.

NORMAL Execution Mode

Use the NORMAL option to return to the normal execution of time functions. When NORMAL is selected, one second of Simulator time equals one second of real time, and timing functions operate normally.

Of the three Execution options (FAST, NORMAL, and SUSPEND), NORMAL is the default.

SUSPEND Execution Mode

The SUSPEND option stops all timing functions. This allows you to view multiple compounds without the values constantly changing. SUSPEND mode also allows you to completely set up a scenario before testing it.

While SUSPEND mode is selected, if you put an object in Manual mode, change its values, and put it back in Automatic mode, all change-of-state and trigger analysis is performed. However, triggered processes will not execute until either the NORMAL or FAST execution mode is selected.

In SUSPEND mode, all connection (CONN) blocks in Automatic mode will continue to be updated once per second. In addition, unreliable connections will continue to flash once every half second.

To leave SUSPEND mode, select either the NORMAL or FAST execution option.

Execution Mode Details

TIME blocks, when executed, will read an internal Simulator clock that updates at ten times the normal rate, updates in real time, or stops updating according to which Execution mode is selected (FAST, NORMAL, SUSPEND).

The following timing functions are also affected by the Execution mode:

- process COUNTDOWN timers
- PULS, DLAY, and BSEQ countdown timers
- PIDL countdown timers
- AD and BD associated input mapping

Simulator Functions

This section describes Simulator functions. This section includes:

- Automatic Mode
- Manual Mode
- Simulating Processes
- Order of Execution of Function Blocks
- Modifying Object Block Configuration
- Simulating Configuration Connections
- Simulating Unreliability
- Simulating Manual Commands
- Simulating a PID Loop Object
- Simulating a TIME Block
- Simulating CONN, CNST, and SVAR Blocks
- Simulating PULS, DLAY, and BSEQ Timers

This section describes what the Simulator does under various conditions. You’ll find information on how to use the Simulator in the previous sections.

Automatic Mode

When in Automatic mode (the default mode), an object block is run continuously, independent of the execution of any process. Even when the process containing the object is not being executed, the object within the process performs its algorithm. This differs from an operation block, which performs its algorithm only when the process it is contained in is executed.

To put a block in Automatic mode, click left on the block to display the block's data fields in the Block window. Click right on the MANUAL ? field (or press F4) to change it to N.

You can put *all* blocks in the file in Automatic mode by pressing the F8 function key (All Automatic).

You will not be allowed to put a block back in Automatic mode if there are any interfield errors in the Block window data fields. In this case, an error message appears in the Message line, informing you of the error. Correct the error and attempt to put the block in Automatic mode again.

Manual Mode

When in Manual mode, a block is not executed. The inputs and algorithm of the block are ignored, but its outputs are still passed to downstream blocks.

When an object block is in Manual mode, it does not accept any commands from CMD or 2CMD blocks.

A block in Manual mode interrupts the data flow and passes its outputs to downstream blocks.

When an object block is in Manual mode, you can change the block's configuration data and outputs, set reliable and offline flags, and manipulate command fields (e.g., S/W OVERRIDE, REPORTS LOCKED). In the Block window, modifiable fields are shown in high intensity.

Only the outputs of operation blocks and special blocks are modifiable in Manual mode.

In the Main window, blocks in Manual mode are drawn in yellow.

To put a block in Manual mode, click left on the block to display the block's data fields in the Block window. Click right on the MANUAL ? field (or press F4) to change it to Y.

You can put *all* blocks in the file in Manual mode by pressing the F7 function key (All Manual).

You can put a block in Manual mode and modify its values as many times as you find necessary for simulation.

Simulating Processes

Processes (and the operation blocks contained within them) are executed only when the process is in Automatic mode and it is either triggered or its countdown timer expires. Object blocks within processes perform their algorithms continuously, even when the process is not executing.

Process Triggering

The Simulator triggers processes on conditions very similar to normal NCM execution.

A process is triggered when:

- its countdown timer expires
- it receives a trigger command
- a trigger connection changes state
- an NC START is executed

A process is not triggered when:

- it is in Manual mode
- it is in the ERROR or DISABLED state

Trigger Connections

A possible trigger is identified as any binary connection whose destination is an input to an operation block and whose source is either a triggerable attribute or a binary shared variable (either from a binary SVAR block or from a binary operation block output that crosses processes).

The following are not possible triggers:

- a connection marked EXEMPT
- a connection in a process that has its EXEMPT ALL flag set to Yes
- a connection from USER, REF, and READ blocks

If a series of CONN blocks are used to pass the value of a triggerable attribute or binary SVAR, only the output of the last CONN block will be identified as a possible trigger.

So you can immediately see which connections are possible triggers in the Main window, the connection arrow for the possible trigger connection is outlined in light cyan. The remainder of the line appears as usual. If the trigger connection is locked, the connection arrow is outlined in yellow instead of light cyan.

A process is triggered by a trigger connection when the:

- value of the trigger connection changes and the connection is reliable and not locked
- trigger connection changes to a reliable state and is not locked
- trigger connection changes to an unlocked state and is reliable

Triggers can occur any time the value of the trigger connection changes, even if the origin block of the connection is in Manual mode.

Manual modification of the output of binary SVAR blocks can cause triggers in all processes that read the value of that SVAR.

How to Manually Trigger a Process

To trigger a process:

1. Put the process block in Manual mode.
2. Move the cursor to the STATUS field and click right (or press Tab) to scroll through the possible process states (READY, WAIT, ERROR, DISABLED) until TRIGGER displays.
3. Put the process block back in Automatic mode. The process immediately executes. If a period is specified, the COUNTDOWN field counts down the length of the period, and the process is executed again whenever the countdown expires.

Another way to trigger the process displayed in the Block window is to leave the process in Automatic mode and press F5.

Process States

In the Simulator, a process compound block can be in one of the following states:

READY: The process is ready for execution. When the process is triggered or whenever the countdown expires, the process will execute.

TRIGGER: The process will execute immediately when in Automatic mode.

HELD: An NC START has been simulated and all processes are put on hold while the Restart process executes. The process will be put into the TRIGGER state as soon as the Restart process successfully completes execution.

Note: If the Restart process error locks, all other processes will remain in the Held state.

WAIT: The process has a WAIT block that has just been executed. The process will resume execution when the WAIT timer expires. You can accelerate the expiration of the WAIT timer by manually changing the WAIT timer from the process's Block window to a small value (e.g., 00:00:01).

DISABLED: The process will not be executed. When the status is Disabled, the only way to execute the process is to change the status to TRIGGER or READY, or have another process send a PRC_ENA (process enable) command to the process. When a process is disabled, all timers associated with the PULS, DLAY, and BSEQ blocks are canceled (i.e., set to 00:00:00).

ERROR: The process will not be executed. This state is caused by a fatal error occurring during execution of an operation block within the process. All fatal errors generate Runtime error messages, which can be viewed with the RUNTIME option. When the status is ERROR, the only way to execute the process is to change the status manually to TRIGGER or READY, or have another process send a PRC_ENA (process enable) command to the process. When a process is in the ERROR state, all timers associated with the PULS, DLAY, and BSEQ blocks are canceled (i.e., set to 00:00:00).

Process PERIOD and COUNTDOWN Fields

The PERIOD field initially displays the period defined in the Editor process compound template. This value determines how often the process will execute. For example, a period of 00:00:05 means the process will execute every five seconds, following the first execution.

The PERIOD field is modifiable in Manual mode.

To change the period, put the block in Manual mode, click left on the PERIOD field to select it, type the new time, and press Enter. Then put the block back in Automatic mode. The process will use the new period after its next execution.

To execute the process only when triggered (rather than periodically), use a period of 00:00:00.

The period may be changed by the execution of a PERD block within the process. A process can contain multiple PERD blocks, which are conditionally or unconditionally executed.

When a process object is initialized, the PERIOD field is set back to the initial period defined in the Editor template.

The COUNTDOWN field counts from the period length to 00:00:00 to show you the amount of time that remains until the process will be executed again. For example, if the period is 00:00:05 seconds, the COUNTDOWN field counts down from 00:00:05 seconds to 00:00:00. When the countdown reaches 00:00:00, the process executes, and the COUNTDOWN field resets to the period and begins counting down again. This cycle repeats for as long as the process remains in Automatic mode, is not disabled, and does not encounter a fatal error.

Order of Execution of Function Blocks

In the Simulator, function blocks are executed in the same order as they will be in the NCM. For NCM execution, this order is established when the file is translated by the Translator. You'll find more information on the order of execution of blocks in the *Graphic Programming* chapter.

The order of execution of operation blocks in a process is shown in the Main window. In the upper left corner of each operation block, the number of the block's position in the execution order is displayed. For example, a number 3 will appear in the upper left corner of an operation block that is the third block to execute in the process.

Modifying Object Block Configuration

To modify an object's configuration means to modify its attribute data fields. For example, you can modify an object's setpoint or high alarm limit. You can then see the effect these changes have on the behavior of the object.

To modify an object block's configuration, first display the block's data fields in the Block window. Then put the block in Manual mode, make the desired changes, and put the block back in Automatic mode.

Simulating Configuration Connections

To simulate configuration connections (Associated Input, Feedback, PIDL Ports, PIDL Output), you must make a connection line between the two blocks in the Editor. This is a requirement specific to the Simulator. The Editor and NCM require only that the Associated Input, Feedback, or Reference be defined in the object's template. The Simulator requires both the definition in the Editor template and the connection line.

Associated Input

When defining an AD or BD block in the Editor, you may define an associated input. To do this, you specify the system\object name of the associated input object and the name of its attribute in the object's template. To simulate this configuration connection, you must also make a connection line between the two objects in the Editor. If you do not make the connection line, the configuration connection will not be simulated.

In the Simulator, the AD or BD will read the associated input attribute at 30-second intervals. You can change this interval in the Associated Input Timer field using the SYSGEN program. In the NCM, the associated input attribute is read once every four seconds for BDs and every 30 seconds for ADs. These intervals cannot be changed. You'll find more information on SYSGEN in the Introduction.

Feedback

When defining an AOS or BO block in the Editor, you may define another object as feedback. To do this, you specify the system\object name of the feedback object in the AOS or BO object's template. To simulate this configuration connection, you must also make a connection line between the AOS or BO block and the feedback object in the Editor. If you do not make the connection line, the configuration connection will not be simulated.

Specifically, you must make a connection line between the following blocks for feedback to be simulated: BO/BI, BO/BD, AOS/AI, AOS/AD, AOS/ACM.

The value of the configuration line will reflect the commanded value of the AOS or BO object.

PIDL Port

When defining a PIDL object block in the Editor, you may define an AI object's VALUE attribute as the value for one of the 12 ports of the PIDL block. To do this, you specify the system\object name of the AI object in the Reference field for that port in the PIDL block's template. To simulate this configuration connection, you must also make a connection line between the AI value output and the PIDL port input. If you do not make the connection line, the configuration connection will not be simulated.

PIDL Output

When defining a PIDL block in the Editor, you may define a port of another PIDL or an AOD block as an output of the PIDL block. To do this, you specify the system\object\attribute name of the PIDL or AOD object in the Output Reference field in the PIDL block's template. To simulate this configuration connection, you must also make a connection line between the PIDL OUT connection and the PIDL port or the AOD block's value. If you do not make the connection line, the configuration connection will not be simulated.

Simulating Unreliability

In the Simulator (as in the NCM), when an object becomes unreliable, the values of some of its attributes will be returned along with the unreliable indicator.

In general, an object becomes unreliable when the object's Offline flag is set to Yes. For other conditions that may make an object unreliable, see the object block descriptions in the *Function Blocks* chapter.

You can simulate an unreliable or offline state and see the effects on downstream function blocks. To simulate an unreliable state, click right on the object block's RELIABLE field to change it to N. To simulate an offline state, click right to display Y in the object block's OFFLINE field.

You can also simulate unreliability in a CONN or SVAR block by putting the block in Manual mode and displaying N in the block's Reliable field. This value will not be overwritten as long as the block is in Manual mode.

In the Main window, unreliable data connection lines (binary, analog, and time data lines) flash blue every one-half second. This allows you to see how unreliability is passed to downstream blocks. (In the Portable Operator Workstation, unreliable data connection lines flash on and off every one-half second.)

For a list of the attributes that become unreliable when an object's Offline flag is set True or the Reliable flag is set False, see the object block descriptions in the *Function Blocks* chapter.

The outputs of operation blocks generally are unreliable when any of their inputs are unreliable. Some operation blocks can generate unreliable data when their inputs are all reliable but out of range. Operation blocks that generate unreliable data will log an error message in the Runtime message list.

Simulating Manual Commands

You can simulate commands from the Operator Workstation or NT by modifying the values of the object attributes that are affected by the commands. For example, you can simulate an Override command or an Auto command. You can then view how the command affects the behavior of the object.

To simulate an Override command for an AOS:

1. Suspend execution by selecting the SUSPEND option in the Control window.
2. Put the block in Manual mode.
3. Change the value to the desired override value.
4. Change the Command priority to 1.

Note: When you override a BO object, you must set the Command priority to 3.

5. Return the block to Automatic mode. The Override flag is automatically set, and the AOS will not change its value when a SET AOS command is sent from a process.

Note: Do not confuse Automatic mode in the Simulator with the Auto command performed at the Operator Workstation. The Auto command at the workstation releases an Override command. See the description that follows.

6. Change execution to Normal by selecting the NORMAL option in the Control window.

Releasing an Override Command

At the Operator Workstation, you release an Override command with the Auto command. In the Simulator, release the Override command by changing the priority to a level lower than 1 (in the case of a BO, a level lower than 3). This mimics the Auto command from the workstation.

Simulating a PID Loop Object

When simulated, the PIDL object responds to commands and periodically performs its algorithms (Input Conditioning, PID, Auxiliary Signal Switch, Output Filter, Selector, and Reliability Switch) according to the sample period specified in the PID Loop template.

The algorithms are similar to the PIDL algorithms run in an NCU except that only the proportional term of the PID algorithm is calculated. The integral and derivative terms are not calculated. In addition, the Simulator does not simulate Self-tuning Algorithm functions, nor simulate suspension of the PID algorithm when no AOD objects are connected as outputs or all of the objects serving as PIDL outputs are in override.

In the Block window, the SAMPLE PERIOD field displays the sample period specified when the PIDL is defined. This value determines how often the PIDL algorithms will execute. For example, a sample period of 5 means the PIDL algorithms will execute every five seconds. This field is modifiable in Manual mode.

The Simulator runs all of the PID Loop algorithms according to the sample period. This differs from the DCM, which runs only the Input Conditioning and PID algorithms according to the sample period. The DCM runs the remainder of the algorithms whenever there is a change in an attribute used by the algorithms.

In the Simulator, if the PID algorithm is disabled (all input scalars set to 0.0) but the other algorithms are used, we suggest that you set the sample period of the disabled PIDL equal to the sample period of the PIDL generating the PID control signal.

The COUNTDOWN field in the Block window counts down the sample period, indicating when the PIDL algorithms will execute next. At the end of the countdown, the PIDL algorithms execute, and the COUNTDOWN field resets to the sample period and begins counting down again.

You'll find more information on the PIDL block in the *Function Blocks* chapter.

Simulating TIME Blocks

When a TIME block executes in Automatic mode, the Simulator reads the current time and date information from the Simulator clock.

The operation of the Simulator clock depends on the selected execution mode (FAST, NORMAL, or SUSPEND). If FAST is selected, the Simulator clock runs ten times faster than the system clock on the computer you are using. In other words, one second of simulator time equals ten seconds of real time. If NORMAL (the default) is selected, the Simulator clock runs the same as the system clock. If SUSPEND is selected, the Simulator clock is stopped.

Whenever the Simulator is started, the Simulator clock is reset to the system clock time.

The values read from the Simulator clock are used as the outputs of the TIME block when it executes. However, by putting a block in Manual mode, you can manually set the outputs of the TIME block to observe how a process will perform at a particular instant.

To manually change the outputs in the TIME block and have the Simulator use these settings:

1. Click left on the TIME block in the Main window to display its data fields in the Block window.
2. Put the TIME block in Manual mode.
3. Change the output fields in the TIME block as desired. You can change the time, day of week, day of month, month, and year. As long as the TIME block is in Manual mode, the Simulator will use the time and date information specified in the TIME block when the process executes.

Simulating CONN, CNST, and SVAR Blocks

In the Simulator, the Connection (CONN), Constant (CNST), and Shared Variable (SVAR) blocks have the following functions in Automatic and Manual modes.

CONN

The Simulator Block window displays both the CONN block's INPUT and OUTPUT fields. If the input is unreliable, the input line is flashing.

Automatic Mode: The input value and reliability are passed through to the output. While in automatic, the pass through operation is performed continuously, independent of the execution of any process.

Manual Mode: Both the output value and output reliability are modifiable. The input is not passed through to the output.

Only CONN blocks that connect data flow (analog, binary, and time types) will show and allow modification of data. If the CONN block does not connect data flow, the following message appears in the Block window when the CONN block is selected:

```
Configuration Connection - No Data Flow.
```

If a CONN block does not have an input, it is ignored by the Simulator, and the following message appears in the Block window when the CONN block is selected:

```
No Input Connection - CONN Block Ignored.
```

CNST

The Simulator Block window displays both the CNST block's VALUE and OUTPUT fields.

Automatic Mode: The output is set equal to the value configured in the Editor. If the output is modified while in Manual mode, it is set equal to the Editor value as soon as it is put into Automatic mode, independent of the execution of any process.

Manual Mode: The output is modifiable. The value of the constant (which will ultimately be used by the process in the NCM) is not modifiable in the Simulator.

SVAR

All SVAR blocks that have the same name and type share the same value. These blocks are put into Automatic and Manual modes as a group; that is, putting one into Manual mode causes all others with the same name and type to be put into Manual mode.

Automatic Mode: If an input to the block exists and the block is enabled, the SVAR value and value reliability are set equal to the value and reliability of the input (an assignment to the SVAR is made). If there are any other SVAR blocks in the file with the same name and type, their values will also be updated.

Manual Mode: The value and value reliability are modifiable. Any changes made to this block will also be made to the other SVAR blocks in the file with the same name and type.

Simulating PULS, DLAY, and BSEQ Timers

The timers used in the PULS (Pulse), DLAY (Delay) and BSEQ (Binary Sequencer) blocks are simulated in the Automatic and Manual Modes as described below.

Note: Execution of timers depends on which execution mode is selected in the Control window (FAST, NORMAL, or SUSPEND). In FAST mode, timers are simulated ten times as fast as normal, meaning one second of Simulator time equals ten seconds of real time. In SUSPEND mode, all timing functions are stopped. In NORMAL mode (the default), timing functions execute normally.

PULS (Cancelable and Non-Cancelable)

The Simulator Block window displays a TIMER field, which represents the dynamic value of the timer that is started when the input changes from False to True.

Automatic Mode: If the process is neither disabled nor in the ERROR state, the timer counts down. When the timer reaches 00:00:00, the process that contains the PULS block is executed and the PULS output goes False. If the status of the process becomes either DISABLED or ERROR, the timer is canceled (i.e., set to 00:00:00). The PULS timer stops counting down when the process is in Manual mode, even if the PULS block is in Automatic mode.

Manual Mode: Timer stops counting down and its value becomes modifiable in the Block window. By changing the value of the timer, you can accelerate the expiration of the timer. If you enter 00:00:00, you cancel the timer and the process will not be triggered automatically. When you take the block out of manual mode, the TIMER field resumes counting down. Refer to the PULS block in the *Function Blocks* chapter for details on its operation, including the one-shot type.

DLAY (Cancelable)

The Simulator Block window displays a TIMER field, which represents the dynamic value of the timer that is started when the input changes from False to True.

Automatic Mode: If the process is neither disabled nor in the ERROR state, the timer counts down. When the timer reaches 00:00:00, the process that contains the DLAY block is executed and the DLAY output goes True, assuming that the input is still True. If the status of the process becomes either DISABLED or ERROR, the timer is canceled (i.e., set to 00:00:00). The DLAY timer stops counting down when the process is in Manual mode, even if the DLAY block is in Automatic mode.

Manual Mode: Timer stops counting down and its value becomes modifiable in the Block window. By changing the value of the timer, you can accelerate the expiration of the timer. If you enter 00:00:00, you cancel the timer and the process will not be triggered automatically. When you take the block out of manual mode, the TIMER field resumes counting down. Refer to the DLAY block in the *Function Blocks* chapter for details on its operation, including the one-shot type.

BSEQ

The Simulator Block window displays ON TIMER and OFF TIMER. These are time fields representing the dynamic value of the timers that are started when a stage change occurs in the BSEQ block. When a stage change takes place, the ON TIMER field is set equal to the configured DELAY ON TIME for the next highest stage (or 00:00:00 if already at highest stage), and the OFF TIMER field is set to the configured DELAY OFF TIME for the current stage (or 00:00:00 if at stage 0).

Automatic Mode: If the process is neither disabled nor in the ERROR state, the timers count down. When either timer reaches 00:00:00, the process that contains the BSEQ block is executed and the BSEQ algorithm is run again to determine if a stage change should occur. Whenever a stage change occurs, the ON TIMER and OFF TIMER are reset to the configured values for DELAY ON TIME and DELAY OFF TIME. If the status of the process becomes either DISABLED or ERROR, the timer is canceled (i.e., set to 00:00:00). The BSEQ timers stop counting down when the process is in Manual mode, even if the BSEQ block is in Automatic mode.

Manual Mode: The ON TIMER and OFF TIMER fields stop counting down and their values become modifiable in the Block window. By changing the values of the timers, you can accelerate the expiration of the timers. If you enter 00:00:00 in both fields, you cancel the timers and the process will not be triggered automatically. When you take the block out of manual mode, the timer fields resume counting down. Refer to the BSEQ block in the *Function Blocks* chapter for details on its operation.

Simulating Local Control

In the Block window, the AOS and BO objects may contain the data field Local Control. It represents the LOC_CNTL attribute of the object, which indicates the current status of the local control, either Yes or No. If local control is Yes, the ASC is controlling the hardware. If local control is No, the NCM is controlling.

The local control field is a read-only status field, and will show in the Block window only if you configured local control to Yes when defining the block with the Editor. If you answered No in the block's template, or the object is mapping to a hardware type that does not allow local control, the field will not show in the Simulator. To simulate a BO object under local control of the hardware, enter a 9 in the CMD PRIORITY field for the BO. To simulate an AOS object under local control of the hardware, enter a 4 in the CMD PRIORITY field for the AOS. For more details on how local control works, refer to the *AOS Object* and *BO Object Technical Bulletins* in the *Metasys Network Technical Manual.*

Differences Between Simulator and NCM Execution

Though the Simulator attempts to match as closely as possible the execution of blocks and processes at the NCM, there are some differences. This section explains the differences between NCM execution and Simulator execution.

Manual Mode

The most obvious difference between Simulator and NCM execution is the Manual mode capability of the Simulator. The Simulator allows you to put a function block in Manual mode and change its outputs or object data fields. The NCM does not allow this.

Note: Do not confuse Manual mode in the Simulator with the manual override switches found on the output Function Modules and the XRL and XRM modules.

PID Loop

The Simulator calculates only the proportional term of the PID algorithm. The derivative and integral terms are not simulated. In addition, the Simulator does not simulate the Self-tuning Algorithm functions, nor simulate suspension of the PID algorithm when no AOD objects are connected as outputs or all objects serving as PIDL outputs are in override. In these cases, the Simulator will continue to calculate the proportional term of the PID algorithm.

The Simulator runs all of the PID Loop algorithms (Input Conditioning, PID, Auxiliary Signal Switch, Output Filter, Selector, and Reliability Switch) according to the sample period. This differs from the DCM, which runs only the Input Conditioning and PID algorithms according to the sample period. The DCM runs the remainder of the algorithms whenever there is a change in an attribute used by the algorithms.

In the Simulator, if the PID algorithm is disabled, we suggest that you set the sample period of the disabled PIDL equal to the sample period of the PIDL generating the PID control signal.

Lighting Control Group

In the Simulator, the LCG prioritization of commands are represented differently than they are in the LCG Focus window at the Operator Workstation. Priority 1 represents an Operator Workstation or NT override condition. Priority 2 represents an On or Off command from a process. Priority 3 represents control at the hardware device, either from a TIMED ON command from a process, ILC schedule, or local switch.

Configuration Connections

For a configuration connection to be valid in the NCM, it is enough that the referenced object is configured in the main object's data base template. For example, feedback will be provided as long as the AI object is referenced in the AOS object's template. However, for simulation, you must also make a connection line between the main object and the referenced object in the Editor. For example, you must make a connection line between the AOS and AI objects for the feedback to be simulated.

Specifically, you must make a connection line between the following blocks for the configuration connection to be simulated: BO/BI, BO/BD, AOS/AI, AOS/AD, AOS/ACM, AI/PIDL, PIDL/AOD, PIDL/PIDL.

Restart Processes

In the Simulator, the only way to simulate a RESTART process is to use the NC START command. An NC START command simulates either a cold or warm start of the NC. The Restart process, if it exists and is enabled, is the first process to execute, while the other processes are placed in a HELD state. You'll find information on how the Restart process operates in the NCM in the *Graphic Programming* section of this manual.

Note: The NC START command is a Simulator command only; it is not a command you can issue to an actual NCM.

Process Priority

The Simulator does not support time-slicing or process interruption. Higher priority processes will not execute more often than lower priority processes. If two processes are to begin simulation in the same one second interval, the higher priority process will be simulated before the lower priority process.

The Simulator supports process priority 1, 2, 3, and 4 (lowest).

Change-of-State Analysis

In the Simulator, change-of-state analysis is triggered for the same reasons that it is in the NCM (e.g., an object receives a command, a PIDL completes execution). In the Simulator, change-of-state analysis is also triggered when a block is put back into Automatic mode.

Change-of-State Feedback Delay Timers

The Simulator does not implement change-of-state feedback delay timers. The value of the object used for feedback will automatically be changed to the value of the BO or AOS it is providing feedback for. Correct and immediate field gear response is assumed.

Blocks and Commands Not Fully Simulated

In the Simulator, the following blocks and commands are not executed as they are in the NCM. An explanation of how the Simulator handles each block or command follows its name.

REF: Always in Manual mode. Outputs (attributes) are always modifiable. All commands to the REF block are ignored. No connections from REF block binary attributes will be considered as possible triggers.

TOT: Always in Manual mode. Outputs are always modifiable.

210A: Always in Manual mode. Outputs (attributes) are always modifiable. All commands to the 210A are ignored.

260A: Always in Manual mode. Outputs (attributes) are always modifiable. All commands to the 260A are ignored.

USER: Always in Manual mode. Binary, analog, and time outputs are always modifiable. The USER block code is not simulated. No connections from USER blocks will be considered as possible triggers.

READ: In Automatic mode, the attribute value is not read; however, the value of the enable input is passed to the enable output. You can simulate a read by putting the READ block in Manual mode and changing the output to reflect a change in the current attribute value. During simulation, the READ output is not overwritten. No connections from READ block binary attributes will be considered as possible triggers.

WRIT: In Automatic mode, the attribute value in the connected object is not changed; however, the value of the enable input is passed to the enable output. Change the writable attribute's value within the object block's data fields to simulate a write.

BEG TRND/END TRND commands: Object ignores the command; however, the value of the enable input is passed to the enable output.

BEG TOT/END TOT commands: Object ignores the command; however, the value of the enable input is passed to the enable output.

RES TOT command: Object ignores the command; however, the value of the enable input is passed to the enable output.

FILE: File blocks are not simulated. The Simulator simulates only one file at a time. To simulate a subfile, go to the Editor and load the subfile, and then access the Simulator.

Object Attributes Not Simulated

For objects, if an attribute is not included in the Simulator Block Window for the object, its function will **not** be simulated. Included in the list of attributes that are not simulated are: Heavy Equipment Delay, Minimum On and Off Times, Maximum Starts per Hour, Alarm Delay, Warning Delay, Override Default Delay, and Auto Restore.

User Messages

There are four types of user messages that can occur when you use the Simulator.

- Fatal Initialization Error Messages
- User Status Messages
- User Error Messages
- Runtime Operation Block Error Messages (fatal and non-fatal)

Fatal Initialization Error Messages

Fatal initialization error messages appear on the screen when you attempt to access the Simulator and there is an error that prevents access.

The following fatal initialization errors may occur.

```
Checksum error for configuration file, GPL.CFG
```

A checksum error is encountered while reading the configuration file.

```
Configuration file not found, GPL.CFG
```

The configuration file is not found in the GPL directory.

```
Connection file not found, {FILENAME}.CI
```

The connection file is not found in the working directory.

```
Data base file not found, {FILENAME}.DB
```

The data base file is not found in the working directory.

```
Error while reading configuration file, GPL.CFG
```

A read error is encountered while reading the configuration file.

```
Error while reading connection file, {FILENAME}.CI
```

A read error is encountered while reading the connection file.

```
Error while reading data base file, {FILENAME}.DB
```

A read error is encountered while reading the data base file.

```
Error while reading text file, {FILENAME}.TXT
```

A read error is encountered while reading the text file.

```
Not enough memory available to run Simulator
```

Not enough memory can be allocated to run the Simulator.

```
Text file not found {FILENAME}.TX
```

The text file is not found in the working directory.

Recovering from a Fatal Initialization Error

To recover from a fatal initialization error, press Enter to return to the Editor. Check the integrity of the GPL configuration. If necessary, restore the backup versions of the file set before attempting to simulate again.

User Status Messages

Status messages appear in the Message line while the Simulator is performing a function. These messages tell you what the Simulator is doing. When you press any key or click the mouse button, the status message disappears.

The following status messages may appear.

```
Initialization of all objects complete
```

This message appears after an OBJ INIT for all objects has completed.

```
{system\object name} :  Initialization complete
```

This message appears after a successful OBJ INIT for a single object has completed.

```
SAVE in PROGRESS...
```

This message appears while a SAVE DB operation is being performed.

```
SAVE COMPLETE
```

This message appears after a SAVE DB operation has completed.

User Error Messages

Error messages appear in the Message line in two cases. They appear when you attempt to:

- put a block into Automatic mode and there are interfield configuration errors in the Block window data fields
- perform an invalid action

When you enter information into data fields in the Block window, you must use the same valid ranges and correct formats you use when entering information into Editor templates. If you enter incorrect information and attempt to put the block into Automatic mode, a message appears and explains the error.

To recover from an object interfield configuration error, simply correct the invalid information and attempt to put the block into Automatic mode again.

If you attempt to perform an invalid action (for example, if you double-click a protected compound), a message appears and explains the error.

The following list includes all messages that result from interfield configuration errors and invalid actions. These messages are arranged in alphabetical order for easy reference. An explanation of the error follows each message.

```
(X) block(s) found with interfield errors, these
remain in Manual
```

You are attempting to place all blocks in Automatic mode and one or more blocks (actual number indicated by X) have errors. These blocks cannot be placed in Automatic mode and remain in Manual.

```
DIFFERENTIAL must be undefined (blank)
```

You must leave the Differential blank if the High Limit, Low Limit, Setpoint, and Normalband are all blank (undefined). The Differential is optional.

```
DIFFERENTIAL value too high
```

The differential value you have entered is too high. The following must be true:

(Setpoint - Normalband/2 + Differential) < (Setpoint + Normalband/2 - Differential)

```
Error while writing connection file, {FILENAME}.CI
```

A disk error occurred during a SAVE DB operation.

```
Error while writing database file, {FILENAME}.DB
```

A disk error occurred during a SAVE DB operation.

```
HIGH ALARM LIM must be > LOW ALARM LIM
```

You must define a High Alarm Limit that is greater than the Low Alarm Limit (if defined).

```
HIGH ALARM LIM value too low
```

The High Limit value you entered is too low. The following must be true:

(High Limit-Differential) > (Setpoint + Normalband/2)

```
Invalid selection: Cannot view PROTECTED compound
```

You are attempting to access a protected compound.

```
LOW ALARM LIM value too high
```

The Low Limit value you entered is too high. The following must be true:

(Low Limit+Differential) < (Setpoint - Normalband/2)

```
NORMAL STATE cannot be NONE when LATCHING POINT is Y
```

If you entered Y (Yes) for the Latching Point field, you cannot specify 0 “None” in the Normal State field.

```
Open failed on connection file, {FILENAME}.CI
```

Connection file does not exist on the disk when you attempt to perform a SAVE DB operation.

```
Open failed on database file, {FILENAME}.DB
```

The data base file does not already exist on the disk when you attempt to perform a SAVE DB operation.

```
SETPOINT & NORMALBAND must both be defined/-
undefined
```

The Setpoint and Normalband values must both be defined or both be blank.

```
SPAN HI Input must be > SPAN LO Input
```

You must define a Span High Input that is greater than the Span Low Input (if defined).

```
SPAN HI OUTPUT cannot equal SPAN LO OUTPUT
```

You must define a Span High Output that is not equal to the Span Low Output (if defined).

```
SPAN values must all be defined or undefined (blank)
```

The four span values must all be defined, or all be blank. (Span High Input, Span Low Input, Span High Output, Span Low Output).

```
WAIT TIMER can’t = 00:00:00 in WAIT state
```

The process status is WAIT and the Wait Timer is 00:00:00; the Simulator will only take a process out of the WAIT state when the Wait Timer changes from 00:00:01 to 00:00:00.

Runtime Operation Block Error Messages

During execution of operation blocks, runtime errors may occur because of invalid data, which the operation blocks use in performing their algorithms. The error messages are similar to the runtime messages generated by a process in the NCM.

There are two types of runtime operation block errors: fatal and non-fatal. Fatal runtime errors stop execution of the process and cause the process to be put into an ERROR state. Non-fatal runtime errors cause the block outputs to become unreliable. The process will continue execution using the unreliable data.

To display runtime error messages, click left on the RUNTIME command in the Control window.

The format of runtime messages is as follows:

```
ERROR <{process sys\obj name}> {BLOCK TYPE} {ERROR
MESSAGE}
```

The following fatal and non-fatal runtime error messages may occur. All errors are non-fatal unless stated otherwise.

```
baro press < 0.5, using 0.5
```

The Barometric Pressure value is out of range in metric units for the ENDP, ENRH, WBDP, WBRH blocks.

```
baro press > 1.08, using 1.08
```

The Barometric Pressure value is out of range in metric units for the ENDP, ENRH, WBDP, WBRH blocks.

```
baro press < 15.0, using 15.0
```

The Barometric Pressure value is out of range in English units for the ENDP, ENRH, WBDP, WBRH blocks.

```
baro press > 32.0, using 32.0
```

The Barometric Pressure value is out of range in English units for the ENDP, ENRH, WBDP, WBRH blocks.

`cos expr too large, partial loss`

The input to a cosine function (in an EQN block) is large, which causes a partial loss of significance in the result.

`cos expr too large, total loss`

The input to a cosine function (in an EQN block) is very large, which causes a complete loss of significance in the result.

`deadband < 0.0, using 0.0`

The Deadband value of a PIR block is less than zero.

`dew pt < -99.9, using -99.9`

The Dew Point value is out of range in metric units for the ENDP, RH, WBDP blocks.

`dew pt > 199.9, using 199.9`

The Dew Point value is out of range in metric units for the ENDP, RH, WBDP blocks.

`dew pt < -147.9, using -147.9`

The Dew Point value is out of range in English units for the ENDP, RH, WBDP blocks.

`dew pt > 391.9, using 391.9`

The Dew Point value is out of range in English units for the ENDP, RH, WBDP blocks.

`dew pt > dry bulb,using dry bulb`

The Dew Point value is greater than the Dry Bulb value for the ENDP, RH, WBDP blocks; the Dry Bulb value is used for the Dew Point in the psychometric calculations.

`differential < 0.0, using 0.0`

The Differential value of a DFCM block is less than zero.

```
divide by zero
```

Attempting to divide by zero in the DIV or EQN block.

```
dry bulb < -99.9, using -99.9
```

The Dry Bulb value is out of range in metric units for the DWPT, ENDP, ENRH, RH, WBDP, WBRH blocks.

```
dry bulb > 199.9, using 199.9
```

The Dry Bulb value is out of range in metric units for the DWPT, ENDP, ENRH, RH, WBDP, WBRH blocks.

```
dry bulb < -147.9, using -147.9
```

The Dry Bulb value is out of range in English units for the DWPT, ENDP, ENRH, RH, WBDP, WBRH blocks.

```
dry bulb > 391.9, using 391.9
```

The Dry Bulb value is out of range in English units for the DWPT, ENDP, ENRH, RH, WBDP, WBRH blocks

```
floating point error
```

An overflow occurred during a math operation (fatal error).

```
input >= 1440.0 minutes
```

The Input value of a RTOT block is greater than or equal to 1440.0 minutes; the calculated result will be 23:59:59.

```
input < 0.0 minutes
```

The Input value of a RTOT block is less than 0.0 minutes; the calculated result will be 00:00:00.

```
integral time < 0.0, using 5.0
```

The Integral Time value of a PIR block is less than zero.

```
log expression <= 0.0
```

The input to a log function (in an EQN block) is less than zero; the calculated result will be a very large negative value.

```
low input >= high input
```

The Low Input value of a SPAN block is greater than or equal to the High Input value (fatal error).

```
low output >= high output
```

The Low Output value of a PIR block is greater than or equal to the High Output value (fatal error).

```
rel humid < 0.0, using 0.0
```

The Relative Humidity value is out of range for the DWPT, ENRH, WBRH blocks.

```
rel humid > 100.0, using 100.0
```

The Relative Humidity value is out of range for the DWPT, ENRH, WBRH blocks.

```
sin expr too large, partial loss
```

The input to a sine function (in an EQN block) is large, which causes a partial loss of significance in the result.

```
sin expr too large, total loss
```

The input to a sine function (in an EQN block) is very large, which causes a complete loss of significance in the result.

```
square root expression < 0.0
```

The input to a square root function (in an EQN block) is less than zero; the calculated result will be zero.

```
stepsize < 0.0, using 0.0
```

The Stepsize value in a RAMP block is less than zero.

```
tan expr too large, partial loss
```

The input to a tangent function (in an EQN block) is large, which causes a partial loss of significance in the result.

```
tan expr too large, total loss
```

The input to a tangent function (in an EQN block) is very large, which causes a complete loss of significance in the result.

```
time = 00:00:00, ignored
```

The Time In value of a WAIT block is zero; the WAIT block will be ignored (no WAIT will occur).

```
x^y, x = 0.0, y = 0.0
```

When computing x raised to the yth power (in an EQN block) and both x and y are zero; the calculated result is zero.

```
x^y, x = 0.0, y < 0.0
```

When computing x raised to the yth power (in an EQN block) and x is zero, and y is negative; the calculated result is a very large positive value.

```
x^y, x < 0.0, y not integer
```

When computing x raised to the yth power (in an EQN block) and x is negative, and y is not an integer; the calculated result is zero.

```
weight < 1.0, using 1.0
```

The Filter Weight value of a FILT block is less than one.

GPL Programmer's Manual

Translator

* Indicates those sections where changes have occurred since the last printing.

Overview

This chapter contains instructions for using the Translator.

The GPL Translator is a utility that translates control strategies in a GPL strategy file into processes that can be downloaded into an NCM.

The Translator:

- Translates a GPL strategy file into downloadable process objects.
- Saves all errors in a list file, which you can view in the Editor.

In addition to translating the currently loaded file, the Translator translates all subfiles referenced by FILE blocks.

Expert Checker

Before the Translator translates a file, the file must be run through the Expert Checker and found to be free of errors. If the last saved version of the file has not been checked, the Translator automatically invokes the Expert Checker to check the file. If the file is free of errors, translation takes place. If errors are found, the file is not translated. In this case, you must return to the Editor and correct the errors before the file can be translated.

Using the Translator

This section explains how to use the Translator. This section includes:

- Running the Translator
- During Translation
- Viewing Errors in the List File

Running the Translator

The Translator requires the .DB and .CI extension files from the control strategy file set.

To run the Translator:

1. Load the file you want to check into the work area of the Editor.
2. Double-click left on the Tools icon (hammer). The Tools option menu appears.

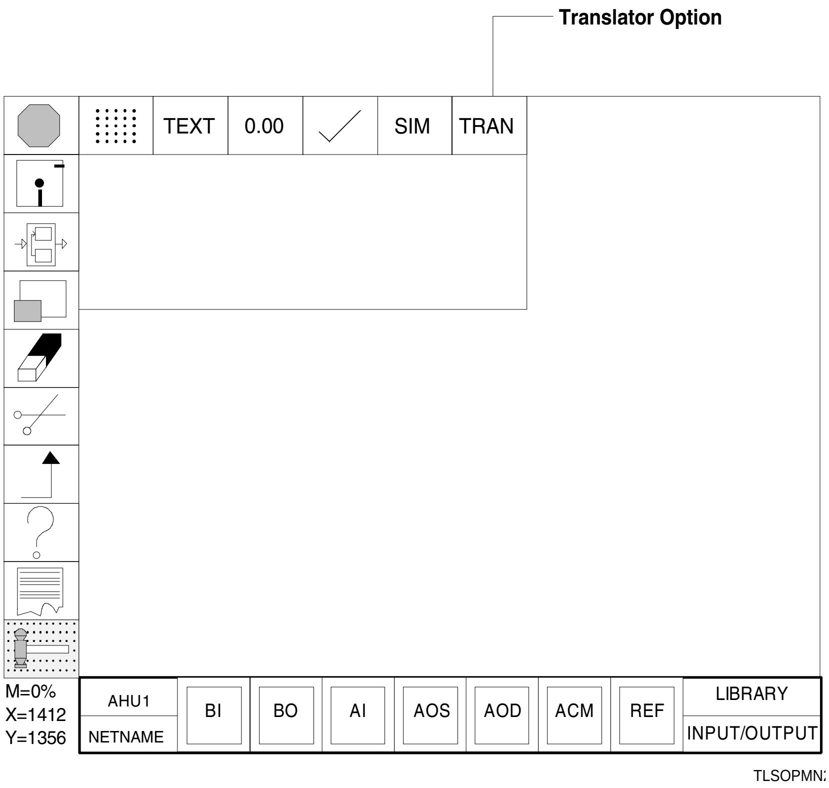


Figure 1: Tools Option Menu

3. Click left on the TRAN option in the Tools option menu. The following Translator submenu appears.

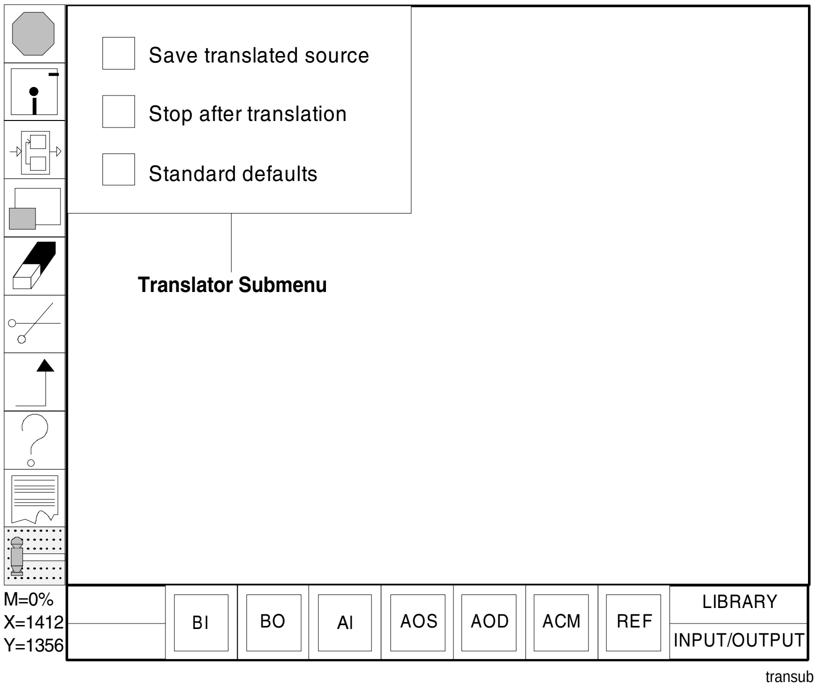


Figure 2: Translator Submenu

4. Click left on the option that corresponds to your choice. The options are as follows:

Save translated source: Click left inside this checkbox if you want to save the intermediate source file (.BAS) after the executable process objects (.OBJ) are created. A list file (.LST) detailing the compilation is generated.

Stop after translation: Click left inside this checkbox if you want only the intermediate source file saved. Process objects will not be created.

Standard defaults: Click left inside this checkbox if you want to create the executable process objects and then delete the intermediate source file. A list file detailing the compilation is generated.

As soon as you select one of these options, the Editor screen clears, and the Expert Checker is invoked to make sure the file has been checked and found to be free of errors. If the file is found to be free of errors, translation begins.

If the file is not free of errors, translation does not take place. Strike any key to return to the Editor and correct the errors.

During Translation

If the file is found to be free of errors, translation occurs according to the selections you made in the Translator submenu. The GPL strategy file (including all nested subfiles referenced by FILE blocks) is translated into a source file and the source file is compiled into downloadable processes.

Each compiled process has the same object name as the process object created in GPL, with an .OBJ extension. The process is also placed in the same directory. For example, if you created a process object with the system\object name AHU1\ECON, this process object is called ECON.OBJ when compiled, and it is placed in the AHU1 system directory.

After compiling, the source file (.BAS) is deleted or not deleted according to your selection. The default is to delete the source file.

If errors are found, the Translator saves the errors in the list file. If translation and compilation are successful, the list file contains detailed information about the compiled process objects.

Translation Messages

During translation, the screen displays the following messages about the progress of the Translation:

```
Translation in progress for {FILENAME}

Blocks Translated: {XX}
```

This message displays the name of the file being translated and the number of blocks that have been translated so far. The Blocks Translated field increments by one as each block is translated.

```
Translation Complete: {XX} errors

Return to Editor

Strike a key when ready...
```

This message tells you that translation is complete and displays the number of errors found. If errors are found, you can view them in the list file (described later in this section).

Compilation Messages

The following messages appear during compilation (if you selected to compile the source file into process objects).

```
Compile in progress for file - {FILENAME}

Compiling process [XX] - {OBJECT NAME}
```

This message displays the name of the file being compiled, the number of processes compiled so far, and the system\object name of the process currently being compiled.

```
Compile Complete: {XX} Errors  {YY}
Warnings

Return to the Editor

Strike a key when ready...
```

This message tells you that compilation is complete and displays the number of errors and warnings found. You can view the errors and warnings in the list file (described later in this section).

Viewing Errors in the List File

If errors are found during translation or compilation, the list file displays the errors.

If the translation and compilation process is successful, the list file displays the following information about the compiled processes:

- source code
- local variable, shared variable, object reference, and trigger/exempt tables
- size of the process object (in bytes) in the PC data base
- number of errors and warnings (0 if none)

The list file has the same name as the file you are translating, with a .LST extension (it is also placed in the same directory). For example, the list file for a strategy file called AHU1 is AHU1.LST.

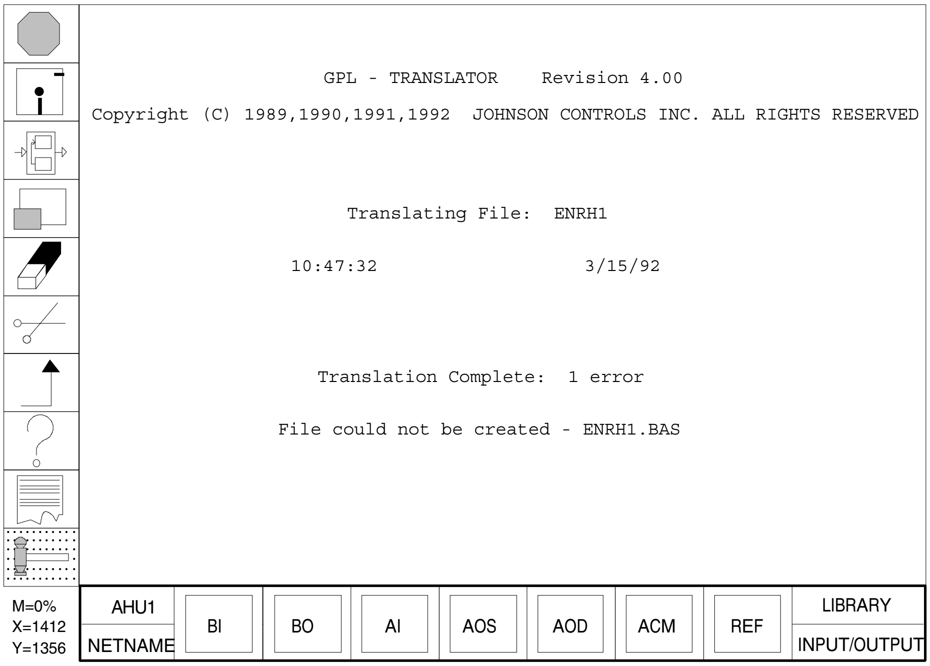
The list file is created the first time the file is run through the Expert Checker. Whenever the file is Expert Checked again or run through the Translator, the list file is cleared and any new error messages are written to it. The list file header displays the time and date, file name, and name of the program (e.g., Expert Checker, Translator, Compiler) that wrote the current information to the list file.

The list file is an ASCII text file. You can view the list with the GPL Editor, or view and print it with any text editor.

To display the list file from the Editor:

1. Load the file you translated into the Editor. If the file is already displayed on the Editor screen, this step is not necessary.
2. Double-click left on the Query icon (question mark) to display the Query option menu.

3. Click left on the VIEW option in the Query option menu. The errors found during translation or compilation are displayed in the list file. Here is an example of a list file.



```
                 GPL - TRANSLATOR    Revision 4.00
Copyright (C) 1989,1990,1991,1992  JOHNSON CONTROLS INC. ALL RIGHTS RESERVED


                   Translating File:  ENRH1

              10:47:32                    3/15/92


                 Translation Complete:  1 error

             File could not be created - ENRH1.BAS
```

Figure 3: Example of a List File

If necessary, use the PageDn and PageUp keys to scroll to additional pages. The GPL Editor can display up to 30 pages (screens) of the list file. (If the list file is longer than 30 pages, exit the GPL Editor and display the additional pages in DOS.)

Error Messages

This section explains the following types of Translator errors:

- Fatal
- Non-fatal
- Compiler

On your screen, the bracketed items will be replaced by actual names from the GPL control strategy file. For example, where this section displays Missing Macro—{BLKTYPE:BLOCKNAME}, your screen will display the actual type and name of the block that is missing the macro.

Fatal Errors

Fatal errors cause translation to stop immediately. Fatal error messages are displayed on the screen; they are not saved in the list file.

If one of these messages appears on the screen, press any key to return to the Editor.

```
Disk failure—read/write—{NAME OF XL
ROUTINE}
```

The translator encountered either a read- or write-to-disk failure. The name of the affected routine is displayed.

```
Disk Full
```

There is not enough room on disk to write the data to disk.

```
File could not be opened—{FILENAME.LST}
```

The Translator cannot find the list file for the strategy file you selected to translate. The file either is not in the current directory or doesn’t exist, there are too many files opened, or the .LST file is read only.

Non-Fatal Errors

Non-fatal errors are saved in the list file. These errors do not cause translation to stop. However, even though translation may continue, the translation is considered unsuccessful if any error occurs. When the translation is unsuccessful, the source file is automatically deleted.

If you selected to compile the source file and the translation is unsuccessful, compilation does not occur. If you selected to compile the source file and the compilation is unsuccessful, no process objects are generated.

`File could not be created—{FILENAME.BAS}`

The Translator cannot create the output file.

`File could not be opened—{FILENAME}`

The Translator cannot find the list file for the strategy file you selected to translate. The file either is not in the current directory or doesn't exist, there are too many files opened, or the .LST file is read only.

`Macro File does not exist—{FILENAME.MAC}`

The macro definition file does not exist within the defined directory.

```
Missing Connect Name—
{BLKTYPE:BLOCKNAME:CONNECT NAME}
```

The specified connect name does not exist in the macro definition file for the block.

`Missing Macro—{BLKTYPE:BLOCKNAME}`

The macro associated with the block type does not exist in the macro definition file.

Compiler Errors

All errors found during compilation are displayed in the list file. If errors are found in a process, the process is not compiled into a process object (however, other error-free processes within the file are compiled). The list file displays each error message below the source code line that contains the error. The error message tells you the number of the line with the error and explains the error.

The following is a list of the error messages that the compiler could write to the list file.

ERROR CODE	MESSAGE
100	Function parameter cannot be an array
101	Attribute is not a triggerable attribute
102	Attribute is not a writable attribute
103	Too many characters in attribute name
104	Too few characters in attribute name
105	Characters after line continuation have been ignored
106	Left and/or right side of + operator is logical
107	Advisory type incorrect
108	Left and/or right side of AND operator is not logical
109	Array subscript too large
110	Illegal character found in attribute name
111	Illegal character found in object name
112	Illegal character found in string
113	Illegal character found in system name
114	Control variable not a local or shared variable
115	Control variable not an integer or real variable
116	Decimal point found without any accompanying digits
117	Non-numeric expression found to left of operator
118	Non-numeric expression found to right of operator
119	Non-numeric expression found to the right of DIFF
120	<> relational operator not allowed in DIFF expression
121	Incorrect DIM variable
122	Unknown compiler directive
123	Left and/or right side of / operator is logical or time
124	Unknown END keyword
125	EXEMPT variable must be attribute or shared variable
126	Unknown item found in expression

Continued on next page . . .

ERROR CODE	MESSAGE
127	Illegal character found in format string
128	Can’t assign value to function outside its DEF
129	Unknown GO keyword
130	Missing/incorrect INCLUDE filename
131	Cannot assign a logical value to an integer variable
132	Cannot assign an integer value to a logical variable
133	Cannot assign a real value to a logical variable
134	Cannot assign a time value to a logical variable
135	Bad variable name
136	Left and/or right side of * operator is logical or time
137	Missing/incorrect network address
138	Incorrect FOR variable given
139	Right side of NOT operator is not logical
140	Bad opcode found while resolving a forward label reference
141	Left and/or right side of OR operator is not logical
142	Incorrect parameter type
143	Left and/or right side of ^ operator is logical or time
144	Missing/incorrect printer name
145	Priority value must be in the range 1 - 4
146	Missing/incorrect process name
147	Cannot assign a logical value to a real variable
148	Cannot assign a time value to a real variable
149	Incorrect relative operator for logical expression
150	Missing/incorrect #REPLACE id
151	Instruction not allowed in a RESTART process
152	Shared variable must be of type logical
153	Incorrect SHARED variable
154	Statement not allowed in a simple IF statement
155	Subscript type is not integer or real
156	Left and/or right side of - operator is logical
157	Missing/incorrect TELL command
158	Cannot assign a logical value to a time variable
159	Cannot assign a real value to a time variable
160	Right side of unary - is logical or time
161	Right side of unary + is logical or time

Continued on next page . . .

ERROR CODE	MESSAGE
162	Instruction not allowed in a user defined function
163	Need at least 1 character after FN in function name
164	Left and/or right side of XOR operator is not logical
165	Can’t create timer
166	Can’t get source date and time
167	Can’t kill timer
168	Control variable in FOR statement cannot be an array
169	Compiler directive must be 1st item on physical line
170	Label already defined
171	Parameter already defined
172	ELSE statement found inside a FIRST PASS statement
173	END IF statement found inside a FIRST PASS statement
174	Expected a real or integer expression
175	Expected a real, integer, time, or logical parameter
176	Expected an attribute parameter
177	Expected user defined function name
178	Expected an integer constant
179	Expected a logical expression
180	Expected a logical type parameter
181	Expected a real or integer type parameter
182	Expected an object reference
183	Expected a string
184	Expected a time expression
185	Expected a time type parameter
186	Expected the keyword ENGLISH or METRIC
187	Exponent too large
188	Extra characters found at end of statement
189	Unable to open file
190	Keyword FIRST PASS not allowed here
191	Function can’t be part of an expression inside its own definition
192	Hours value is too large
193	Variable doesn’t match the nearest FOR loop
194	Incorrect number of parameters - expecting
195	Integer constant is too large
196	Label defined inside of a user defined function

Continued on next page . . .

ERROR CODE	MESSAGE
197	Label defined outside of a user defined function
198	Label defined inside of another user defined function
199	Keyword not allowed in expression
200	Label too large
201	Line is too large to decompile
202	Logical line exceeds 1024 characters
203	Logical & non-logical data used with operator
204	Loop crosses user defined function definition
205	Minutes value is too large
206	Negative sign must be at the beginning or end of the format string
207	Missing left parenthesis on array variable
208	Missing right parenthesis on array variable
209	Missing back slash in system object name
210	Missing comma
211	Missing double quote at end of string
212	Missing END PROCESS keyword
213	Missing single quote at end of system object name
214	Missing equal sign
215	Missing exempt item
216	Missing exponent
217	Missing closing right parenthesis in expression
218	Missing format string
219	Missing left parenthesis on function parameter list
220	Missing right parenthesis on function parameter list
221	Missing GOTO/GOSUB keyword
222	Missing/incorrect label reference
223	Missing minutes field in time constant
224	Missing parameter after comma
225	Format string must have at least 1 pound sign
226	Missing PROCESS keyword
227	Missing/incorrect process descriptor string
228	Missing required parameter
229	Missing seconds field in time constant
230	Missing THEN keyword
231	Missing TO keyword

Continued on next page . . .

ERROR CODE	MESSAGE
232	Missing variable reference after comma
233	Only 1 decimal point allowed in format string
234	Only 1 negative sign allowed in format string
235	Too many characters in name token
236	Multiple ELSE statements found
237	Nested replacements not allowed
238	Nested user defined functions not allowed
239	Not part of a WHILE statement
240	Not part of a FIRST PASS statement
241	Not part of a user defined function definition
242	Not part of an IF statement
243	Missing keyword DEF
244	Missing $ in keyword DATE$
245	Not part of a FOR statement
246	Nested INCLUDE files not allowed
247	Missing PASS keyword
248	Numeric & non-numeric data used with operator
249	Process object exceeds 32K
250	Too many characters in object name
251	Too few characters in object name
252	Can’t open include file
253	More that 1 PRIORITY statement found
254	Real constant is too large
255	Real constant is too small
256	This TELL command requires an attribute destination
257	This TELL command requires an object destination
258	Seconds value is too large
259	Source line too large
260	String attributes not allowed in TELL commands
261	Too many characters in string
262	Too many characters in system name
263	Time and non-time data used with operator
264	Too many characters in time constant
265	Too many errors found - compilation aborted
266	Too many internal labels
267	Too many internal variables

Continued on next page . . .

ERROR CODE	MESSAGE
268	Too many levels of parentheses in expression
269	Two label definitions on same logical line
270	Unable to allocate memory block—system error
271	Unable to add process object to database
272	Unable to close network. Status =
273	Unable to create JC-BASIC compiler status dialog box
274	Unable to create JC-BASIC compiler status dialog box template
275	Unable to delete process object
276	Unable to get user initials
277	Unable to lock memory block—system error
278	Unable to open network. Status =
279	Unable to start compile
280	User defined function has not been defined
281	Unfinished FOR loop found
282	Unfinished IF loop found
283	Unfinished user function definition found
284	Unfinished WHILE loop found
285	Unknown/incorrect attribute found
286	Unknown/incorrect object found
287	Undefined REPLACE identifier
288	Unknown statement type found
289	Name does not begin with a letter
290	Label reference never defined
291	User defined function has already been defined
292	Variable has already been defined
293	Manufacturing file not found
294	Incorrect NCM for system

GPL Programmer’s Manual

Function Blocks

* Indicates those sections where changes have occurred since the last printing.

Code No. LIT-631070

Introduction

This chapter contains detailed descriptions of all the function blocks available with the Graphic Programming Language. The block descriptions are split into two sections: *Object Blocks* and *Operation and Special Blocks*. The last section of this chapter, *Template Field Descriptions*, is a glossary of all attributes and parameters that can be defined in the data base templates.

A function block is a rectangle on the screen that represents a Metasys software object, process, or GPL function. Each function block performs a particular action, such as selecting the highest of two values, or calculating enthalpy. GPL features three different types of function blocks: object, operation, and special (Figure 1).

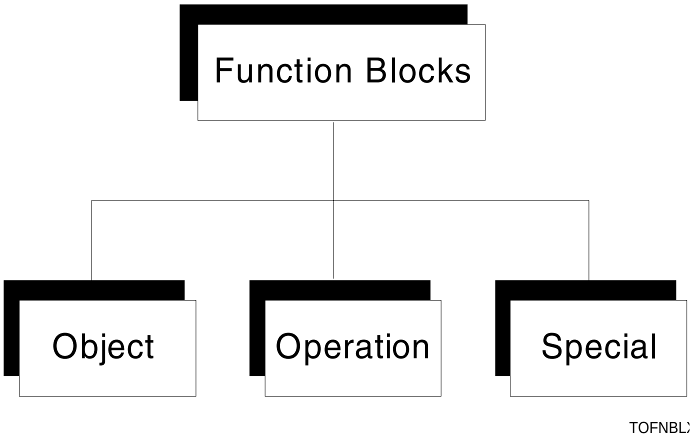


Figure 1: Types of Function Blocks

You paste down each block in the work area, and then define its data base by filling in a template that drops down into the work area. The values in the template define its characteristics.

In most cases, each block is connected to other blocks. The connections between the blocks establish the data and control flow on a diagram. Each block has a defined set of allowable inputs and outputs to which lines may be connected. These inputs and outputs display on connection menus, which come to the screen when connecting the blocks.

For more information on Function Blocks, refer to the *Graphic Programming* chapter.

Object Blocks

The section titled *Object Blocks* describes all the object blocks that are available in GPL. The object blocks include:

ACM	Accumulator	**MSD**	Multistate Data*
AD	Analog Data	**MSI**	Multistate Input*
AI	Analog Input	**MSO**	Multistate Output*
AOD	Analog Output Digital	**PIDL**	PID Loop
AOS	Analog Output Setpoint	**REF**	Generic Object Reference**
BD	Binary Data	**210A**	Control System C210
BI	Binary Input	**260A**	Control System C260
BO	Binary Output	**ZONE**	Fire Zone
LCG	Lighting Control Group		

* Used only in European market.

** Represents C260X, C500X, and CS software objects; also, any hardware objects.

Operation and Special Blocks

The section titled *Operation and Special Blocks* describes all the operation and special blocks that are available in GPL. The operation blocks include:

ABRT	Abort	**PIR**	PI Reset
ADD	Addition	**PRNT**	Print
ADV	Advisory	**PULS**	Pulse
AND	And	**RAMP**	Ramp
AVG	Average	**READ**	Read Attribute
BSEQ	Binary Sequencer	**RH**	Relative Humidity
CMD	Command	**RTOT**	Real-to-Time
COMP	Compare	**SAMP**	Sample and Hold
DBCM	Deadband Compare	**SPAN**	Span
DFCM	Differential Compare	**STOP**	Stop
DIV	Divide	**SUB**	Subtraction
DLAY	Delay	**SVAR**	Shared Variable
DWPT	Dew Point	**SWCH**	Switch
ENDP	Enthalpy Dew Point	**TIME**	Time
ENRH	Enthalpy Relative Humidity	**TOT**	Totalization
EQN	Equation	**TTOR**	Time-to-Real
FILT	Filter	**2CMD**	Dual Command
FREL	Forced Reliable	**UNRD**	Unreliable Data
HSEL	High Select	**USER**	User
LSEL	Low Select	**VH**	Value Holder
LTCH	Latch	**WAIT**	Wait
MSEL	Mode Selector	**WBDP**	Wetbulb Dew Point
MUL	Multiply	**WBRH**	Wetbulb Relative Humidity
NOT	Not	**WRIT**	Write Attribute
OR	Or	**XOR**	Exclusive Or
PERD	Period		

The special blocks include:

CNST	Constant	**FILE**	File
CONN	Connection		

-METASYS®

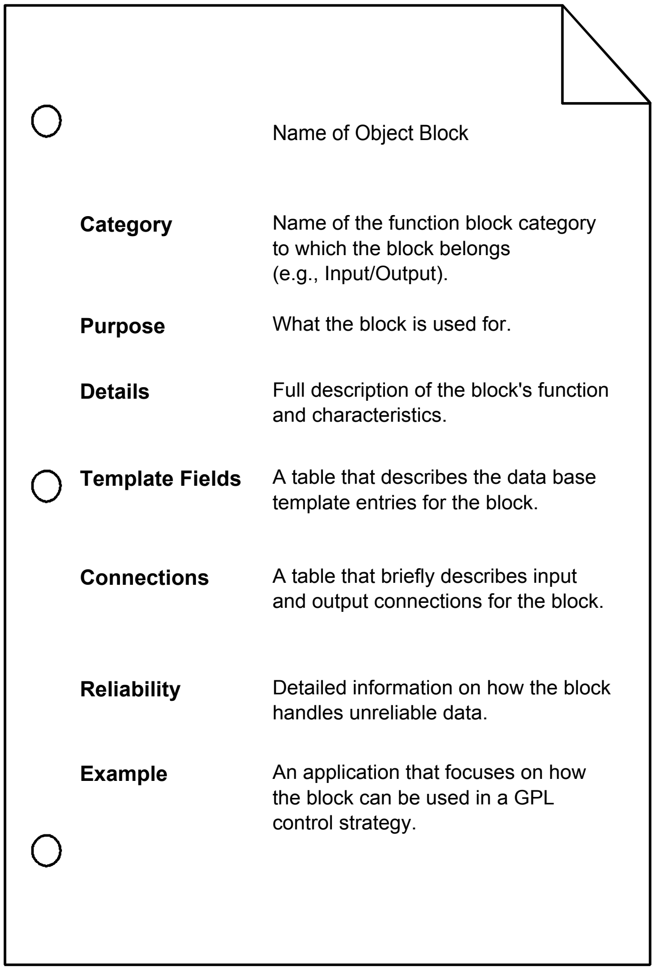
GPL Programmer's Manual

Object Blocks

Introduction

This section presents a detailed description of each object block available with the Graphic Programming Language. The blocks are organized alphabetically.

The description of each block in this section follows a standard format. Figure 2 shows the format with the different sections.



Name of Object Block

Category Name of the function block category to which the block belongs (e.g., Input/Output).

Purpose What the block is used for.

Details Full description of the block's function and characteristics.

Template Fields A table that describes the data base template entries for the block.

Connections A table that briefly describes input and output connections for the block.

Reliability Detailed information on how the block handles unreliable data.

Example An application that focuses on how the block can be used in a GPL control strategy.

STDRDFMT

Figure 2: Standard Format for Block Descriptions

Four of these headings require further explanation: category, template fields, connections, and example.

Category

Object blocks are organized in the GPL Editor by category. Four different categories group the object blocks: Input/Output, Data, Multistate, and Controllers.

Input/Output Blocks

BI	BO	AI	AOS	AOD	ACM	REF

These blocks represent Metasys® software point objects that are mapped to actual hardware. The input blocks convert signals provided by the input hardware to meaningful values that the GPL algorithms can use. The input and output blocks accept commands from GPL command blocks and perform the control function on the output hardware.

Data Blocks

SVAR	CNST	CONN	VH	AD	BD	

These blocks represent Metasys software point objects and are holders of data that can be used either in the same process or other processes.

Multistate Blocks

MSI	MSO	MSD				

These blocks represent Metasys multistate software point objects.

Controller Blocks

PIDL	LCG	210A	260A	ZONE		

These blocks represent Metasys software objects mapped to controller hardware. They are used for HVAC control applications.

Template Fields

The Template Fields section has a table that describes the details about database template entries. The following is the header for the table and descriptions of its columns:

Category	Field Name	Type	Default	Range/Choices

Category

A major grouping of related fields. An example is Engineering Data in the AI block template.

Field Name

The name of the attribute, value, or parameter as it appears in the database template. An example is Expanded ID.

Note: Some template entries may also be connection menu selections. For example, the Auxiliary Switch Input parameter of the PIDL block is in the block’s template and in the block’s input connection menu. You may enter a value in the template and connect the value to the block, but the connected value will always take precedence over template value. Those entries that are both in the template and in connection menus are indicated by an asterisk (*) in the chart.

For definitions of all attributes, values, and parameters, refer to the section called *Template Field Descriptions.*

Type

The type of data that describes the template entry. Included are:

ANA: Analog, real or floating point fields. An example is the Hi Alarm field for an AI block.

BIN: Binary (Boolean) fields that take a Yes (Y) or No (N) entry. An example is the Save PT History? field for a BD object.

INT: Integer fields that take any integer value. An example is the Decimal Position field for an AOD object.

N: Fields that cannot be modified once you add the object to the archive database. An example is the Slot Number field for an ACM block.

READ-ONLY: Read-only fields that cannot be edited; the Editor enters these values for you. An example is the NC Name field for an AD block.

POP-UP: Fields that “pop up” on the screen when you make a selection in a previous field. Which field displays depends on the selection of the previous field. For example, on the AOS object template, the two parameters Step Ratio and Saturation Size pop up when you select INCR for the Point Type. Some pop-up fields may also be tab fields.

STR: Character string fields that allow any combination of alphabetic characters (A-Z, a-z), international language characters, numbers (0-9), and the underscore (_). An example is the System Name field (most blocks). Refer to *Appendix F: Characters, Symbols, and Reserved Words* for a list of international characters, reserved names, and invalid symbols that cannot be used in STR fields.

TAB: Fields that toggle between different entries. A different entry displays each time you press the Tab key. Once you have tabbed through all the choices, the first choice reappears. To keep a selection, press Enter or an arrow key. An example is the Report Type field for the AOS object.

Default

The value that the Editor automatically puts in the field. The default value is based on common applications; therefore, you do not have to change it in most cases.

Range/Choices

The range of acceptable values; or, the acceptable or available choices. The “Real” and “Time” entries mean any number in these ranges:

Real: 99999999 to 0.000001, 0.0, -0.00001 to -9999999.

Time: 00:00:00 to 23:59:59

Notes: GPL converts all decimal numbers into binary format; therefore, only five of the eight possible digits can be displayed reliably. For example, if -49.4567 is entered, GPL will display -49.4566, not -49.457. This is because rounding errors are introduced after the fifth digit. Decimal number conversion is unpredictable after the fifth digit.

Also, GPL forces a fraction to have a 0 before the decimal point. The least significant digit will not be displayed. For example, if you enter -.123456, GPL displays -0.12345 (the number 6 is dropped).

Although they are included as choices, not all LONWORKS® compatible devices are available at the publication date of this document.

Connections

This heading has a table that briefly describes all available input and output connections for the block. The input connections are organized under the Input (GPL) and Input (Commands) rows. The output connections are organized under the Output (GPL) and Output (Attributes) rows. The table has four columns: Connection, Name, Type, and Label.

Connection

The type of connection (e.g., Input [GPL]).

Name

The full name of connection type (e.g., READ and VALUE).

Type

The type of data, including:

Data Type	Description
ANA	Analog connections
BIN	Binary (Boolean) connections
CMD	Single command connections
CTL	Control flow connections
READ	Read attribute connections
TIM	Time connections
2CMD	Dual command connections
WRIT	Write attribute connections

Label

A one- or two-character abbreviation of the connection name (e.g., I1 for Input 1). The label, which identifies a connection, appears at the end of a line.

Note: An asterisk (*) next to an entry indicates that it is in the template and in the connection menu.

All input and output connections for object blocks are optional.

Example

Each example indicates a process period, which is how often the process should execute. These period values are guidelines only. The actual process periods you choose should be based on the application and the type of controlled mechanical equipment.

ACM (Accumulator) Object

Category

Input/Output

Purpose

Creates a software representation of a binary input device that is measuring flow or consumption by monitoring a rate of contact change.

Details

Figure 3 shows the general model of how the ACM object operates. For more information, see the *Metasys Network Technical Manual (FAN 636), Software Data Sheets, Accumulator (ACM) Technical Bulletin.*

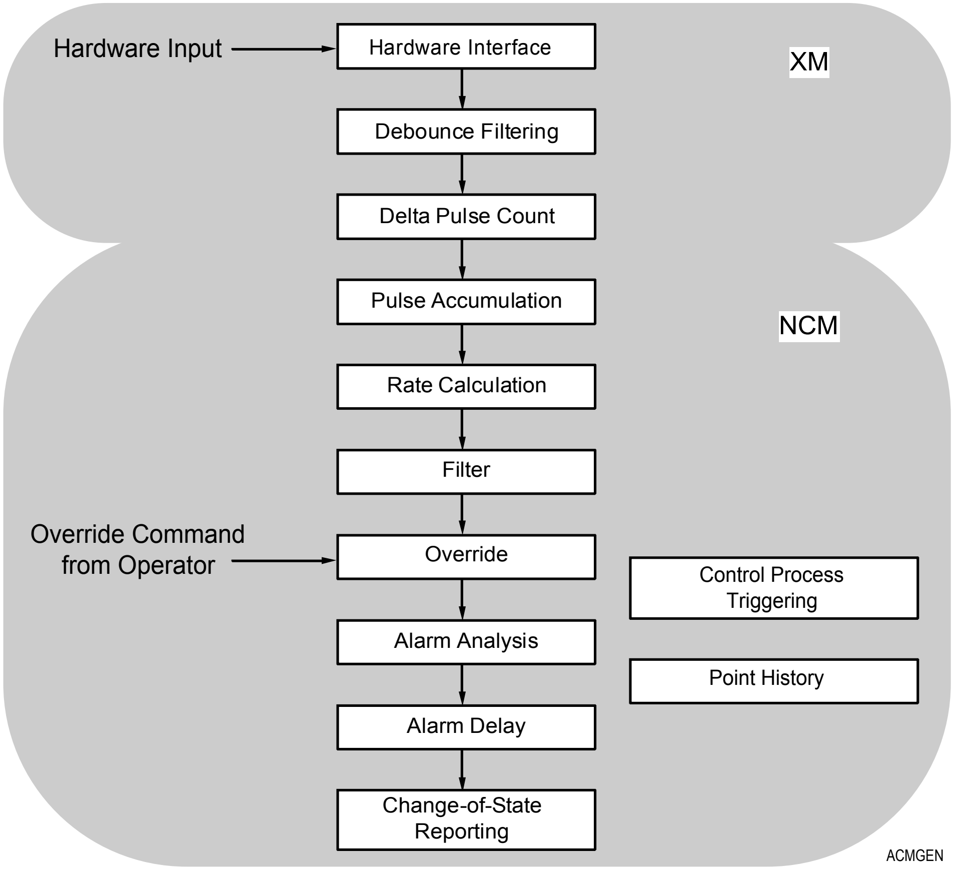


Figure 3: ACM General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type*	TAB/N	XBN	XBN, XRM/XRL/2X, AHU, VAV, UNT, OTHER, DC9100, DX9100, XT9100, XTM, DX91ECH, FPU, DSC8500
	Slot Number			
	(XBN)	INT/N	1	1 to 32
	(XRM,XRL,2X)	INT/N	1	1 to 8
	(FPU)	INT/N	1	1 to 16
	Point Type			
	(For XMs)	TAB/N	SINGLE	SINGLE or FORM C
	(For AHU, VAV, UNT, OTHER)	READ ONLY/N/POP-UP	BI	BI
	Debounce Filter	ANA/ POP-UP	24	12 to 3060 msec (multiples of 12 only)
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)
	Logical Pnt Nmbr	INT/N	1	1 to 255
	Logical Pnt Type	READ ONLY/N	TOT	TOT
	Point Address			
	(For AHU)	INT/N	7	7 or 8
	(For VAV, UNT)	INT/N	4	4
	(For OTHER)	INT/N	1	1 to 256
	Hardware Reference			
	(For DC9100)	TAB/N	Total1	Total1-2
	(For DX9100, DX91ECH)	TAB/N	CNT1	CNT1-8 PM1-12AC1-8 XT1-8CNT1-8
	(For XT9100, XTM)	TAB/N	CNT1	CNT1-8

* To define an XRE for HW Type, select XRM/XRL/2X.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
Engineering Data	Analog Units	STR	KW	6 characters
	Analog Con Units	STR	KWH	6 characters
	Decimal Position	INT	1	0 to 3
	High Alarm Limit	ANA	Blank	Real or Blank (not defined)
	Low Alarm Limit	ANA	Blank	Real or Blank (not defined)
	Setpoint	ANA	Blank	Real or Blank (not defined)*
	Normalband	ANA	Blank	Real > 0.0 or Blank (not defined)*
	Differential	ANA	Blank	Real > 0.0 or Blank (not defined)
	Filter Weight	ANA	Blank	Real > 1.0 or Blank (not defined)
* Either both Setpoint and Normalband must be defined, or both must be undefined (blank).				

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Parameters	Pulse Constant	ANA	0.000000	Real or Blank (not defined)
	Rate Constant	TAB	HOUR	HOUR,SEC,MIN
	Warning Delay	INT	1	0 to 255 minutes
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Warning	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS

Continued on next page . . .

Category (Cont.)	**Field Name**	**Type**	**Default**	**Range/Choices**
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Warning No.	INT	0	0 to 255
	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr. No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	ANA	FB
	WRITE	WRIT	WR
Input (Commands)	ALARMS	CMD	AC
	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RES_TOT	CMD	RT
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	WARNINGS	CMD	WC
Output (GPL)	READ	READ	RD
Output (Attributes)	HI_ALARM	BIN	HA
	HI_WARN	BIN	HW
	LO_ALARM	BIN	LA
	LO_WARN	BIN	LW
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	ANA	V

Reliability

The ACM object becomes unreliable when the hardware it represents goes offline or reports an unreliable count. The following ACM attributes also become unreliable: VALUE, DISPLAY, LO_ALARM, LO_WARN, HI_ALARM, HI_WARN, NORMAL, and STATUS.

Note: The TOTAL attribute is always reliable.

The ACM object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an ACM object block sends unreliable data to the object, the WRIT block will execute, but the ACM object will ignore the write and remain in its current state.

Example

The ACM object block in this example (Figure 4) is an input to the DFCM block. The DFCM block compares the current value of an accumulator object (AHU1\ACM1) to a setpoint (SETPOINT). If the ACM value is greater than the setpoint, given an applied differential (DIFF), this process will print a message to an output device. The message is:

```
THE 2ND FLOOR METER THAT MEASURES KW HAS
EXCEEDED <value>
```

Notice that this message includes a description of the engineering units (kilowatts). That is because units cannot be appended to the end of the message.

A period of 00:02:00 minutes is defined for this process, which means it will run once every two minutes.

Note: The DFCM and PRNT blocks, both operation blocks, must be placed in a process.

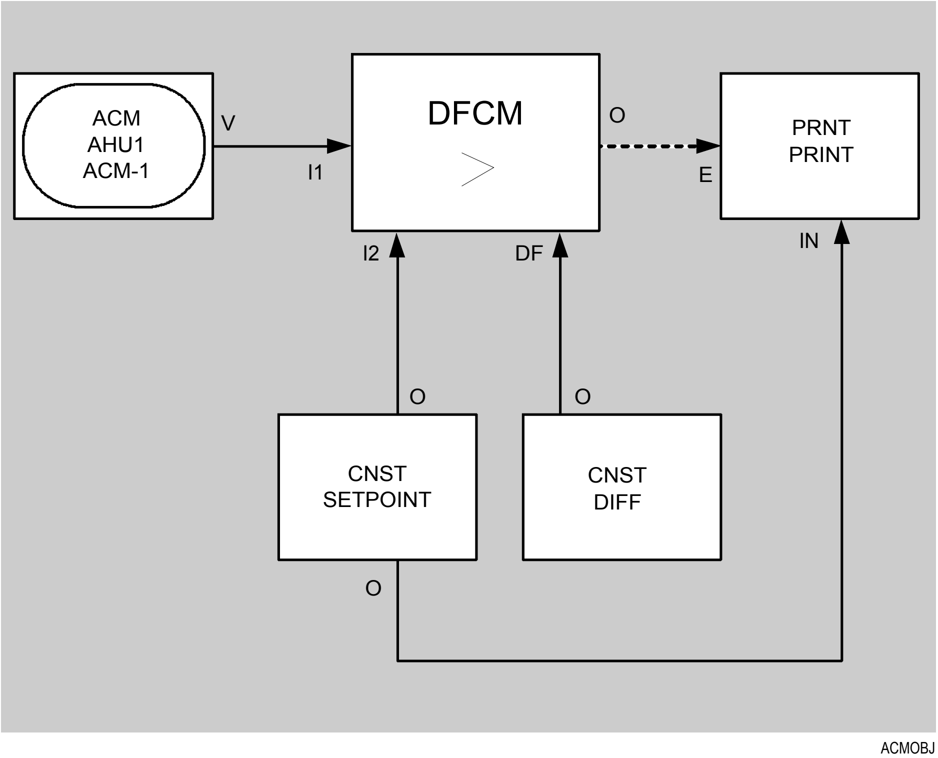


Figure 4: ACM Object Example

AD (Analog Data) Object

Category

Data

Purpose

Creates a storage location for an analog value. The AD object has no associated hardware. It can receive a command or an analog value from another object, or store the result of some calculation.

Details

Figure 5 shows the general model of how the AD object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Analog Data (AD) Technical Bulletin*.

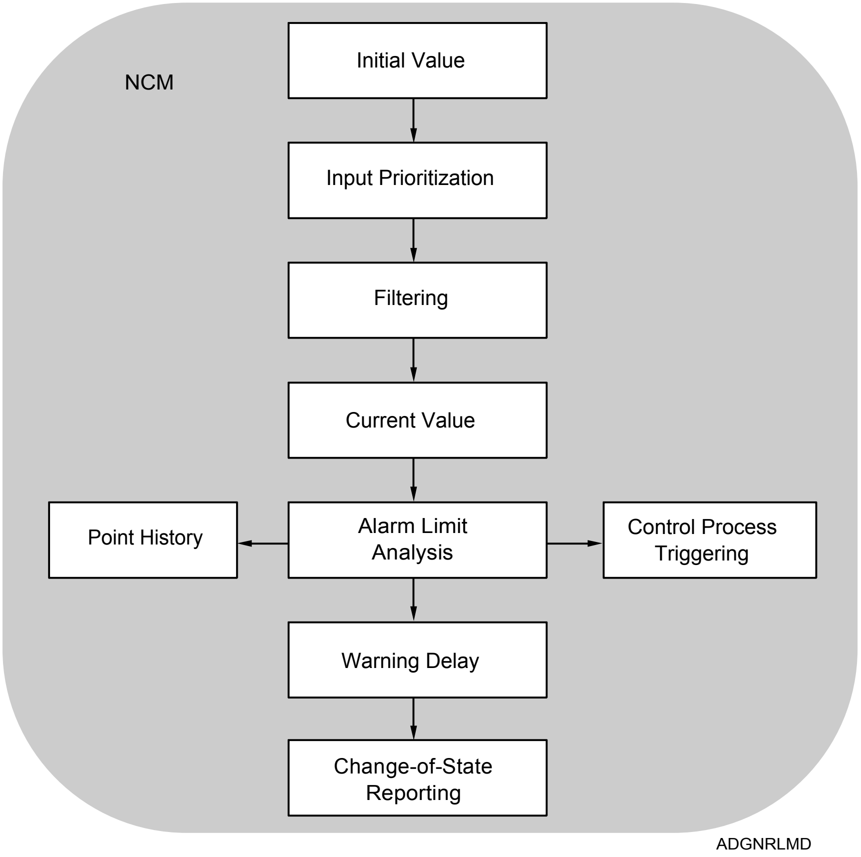


Figure 5: AD General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
	NC Name	READ ONLY/N	Blank	8 characters
Engineering Data	Analog Units	STR	DEG F	6 characters
	Decimal Position	INT	1	0 to 3
	High Alarm Limit	ANA	80.00000	Real or Blank (not defined)
	Low Alarm Limit	ANA	40.00000	Real or Blank (not defined)
	Setpoint	ANA	55.00000	Real or Blank (not defined)*
	Normalband	ANA	10.00000	Real > 0.0 or Blank (not defined)*
	Differential	ANA	1.000000	Real > 0.0 or Blank (not defined)
	Filter Weight	ANA	Blank	Real > 1.0 or Blank (not defined)
Associated Input	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Attribute Name	STR/N	Blank	8 characters

* Either both Setpoint and Normalband must be defined, or both must be undefined (blank).

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable Pt Hist.	BIN	N	Y (Yes) or N (No)
	Save Pt History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Parameters	Warning Delay	INT	1	0 to 255 Minutes
	Initial Value	ANA	55.00000	Real
	Adjust Disabled	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Warning	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
Messages	Warning No.	INT	0	0 to 255
	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	ASSOC IN	ANA	AS
	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	ANA	FB
	WRITE	WRIT	WR
Input (Commands)	ALARMS	CMD	AC
	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RELEASE	CMD	R
	RES_TOT	CMD	RT
	SET_AD	CMD	SA
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	WARNINGS	CMD	WC
Output (GPL)	READ	READ	RD
Output (Attributes)	HI_ALARM	BIN	HA
	HI_WARN	BIN	HW
	LO_ALARM	BIN	LA
	LO_WARN	BIN	LW
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	ANA	V

Reliability

The AD object becomes unreliable if the assigned attribute it is sampling for its value becomes unreliable. It also becomes unreliable if GPL commands an unreliable value to the object. The following AD attributes also become unreliable: DISPLAY, LO_ALARM, LO_WARN, HI_ALARM, HI_WARN, NORMAL, STATUS, and VALUE.

The AD object ignores all commands that contain an unreliable parameter. However, the AD object does accept a SET_AD command whose Value parameter is unreliable. In this case, the command forces the AD object to unreliable.

If a WRIT block that is connected to an AD object block sends unreliable data to the object, the WRIT block will execute, but the AD object will ignore the write and remain in its current state.

Example

This example has an AD object block as an adjustable setpoint for economizer control (Figure 6). If the outside air temperature (AHU1\OA-TEMP) is less than the AD setpoint (AHU1\EC-SETPT), given an applied differential of 2.0°F, economizer (ECON) is set.

A period of 00:02:00 minutes is defined for this process, which means it will run once every two minutes.

Note: The DFCM and SVAR blocks, both operation blocks, must be placed in a process.

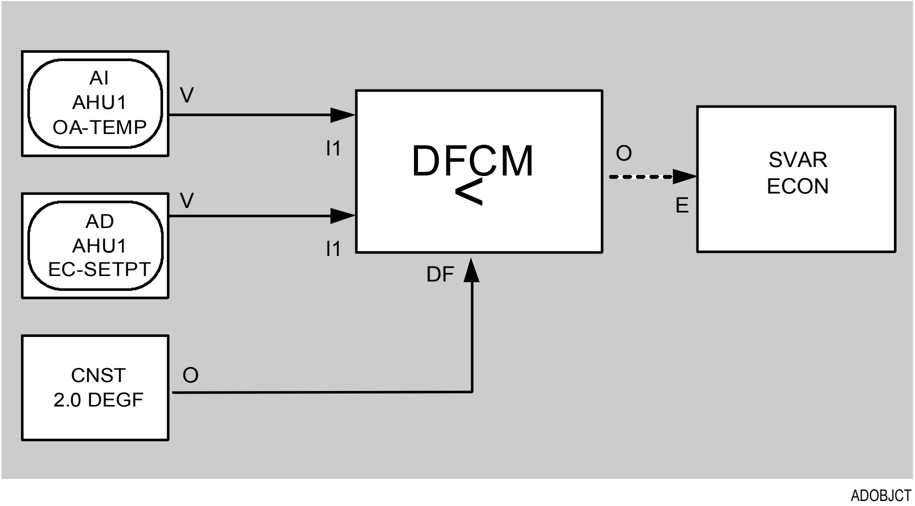


Figure 6: AD Object Example

AI (Analog Input) Object

Category

Input/Output

Purpose

Creates a software representation of a hardware device that is monitoring an analog value. The primary function of an AI object is to convert the raw hardware signal (analog-to-digital counts) from an analog input device to data that can be used in operator displays, alarm limit analysis, and control processes.

Details

Figure 7 shows a general model of how the AI object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Analog Input (AI) Technical Bulletin.*

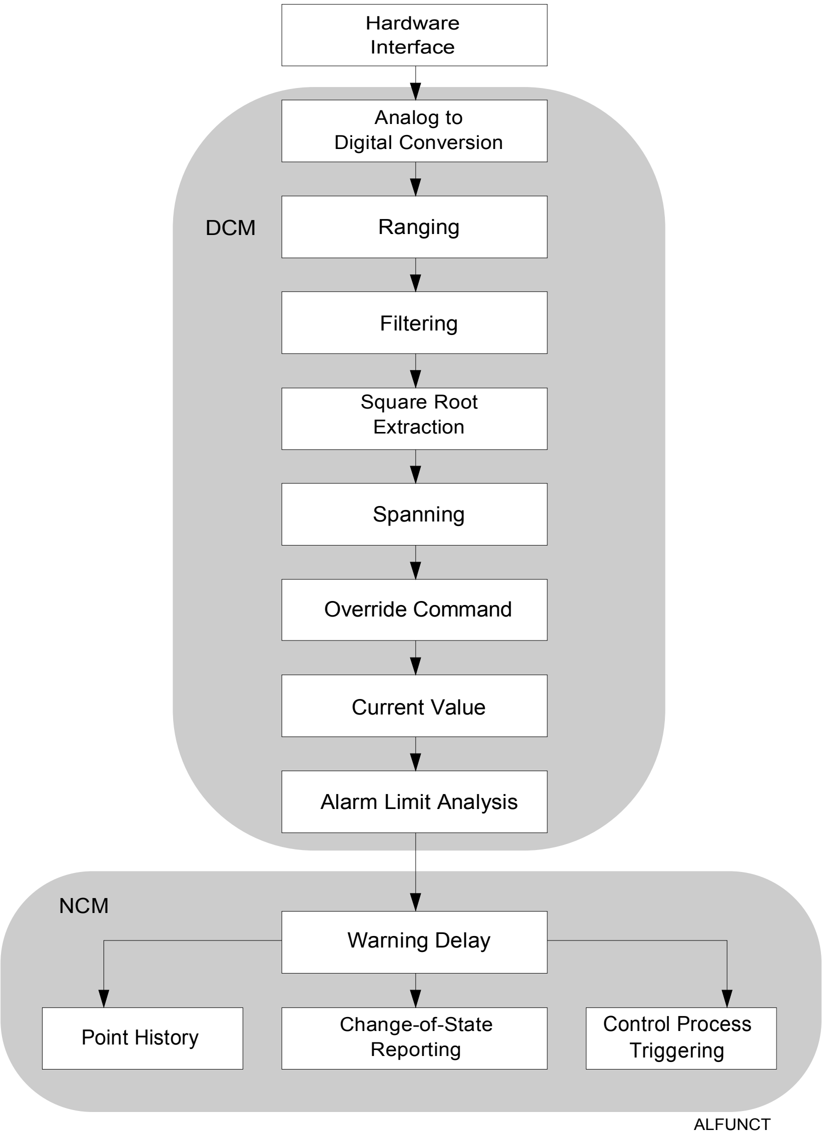


Figure 7: AI Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140, FPU, DSC8500, AHU, VAV, VMA, UNT, OTHER, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[3]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For FPU)	INT/N/ POP-UP	1	1 to 16
	Analog Type			
	(For DCM)	INT/N/ POP-UP	1K ohm	1K ohm, 100 ohm, or VOLT/AMP
	(For DCM140)	INT/N/ POP-UP	1K ohm	1K ohm, 100 ohm, VOLT/AMP or V/A LOW END REL*
	Point Type	TAB/N/ POP-UP	AI	0 = AI 1 = MAI[1]
	Subslot Number	INT/N/ POP-UP	1	1 to 2
	Filter Weight	ANA/ POP-UP	Blank	Real >1.0 or Blank (not defined)
	Flow Coefficient	ANA/ POP-UP	Blank	Real > 0.0 or blank (not defined)
	Span Low Input[2]	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span High Input[2]	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span Low Output[2]	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span High Output[2]	ANA/ POP-UP	Blank	Real or blank (not defined)

[1] If Point Type = MAI, 100 ohm option is invalid for analog type.

[2] Spanning is not supported for an AI object on a DSC8500.

[3] LON Hardware type is not definable from GPL. See *Example* section for details.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	STD Range Type			
	(For DCM and DCM140)	INT/ POP-UP	1	0 to 25, 34 to 111
	(For FPU)	INT/ POP-UP	26	0, 26 to 33
	Linear Parm. 1	ANA/ POP-UP	See Default AI Ranges Table	Real
	Linear Parm. 2	ANA/ POP-UP	See Default AI Ranges Table	Real
	Linear Parm. 3	ANA/ POP-UP	See Default AI Ranges Table	Real
	Linear Parm. 4	ANA/ POP-UP	See Default AI Ranges Table	Real
	Filter Tolerance	ANA/ POP-UP	0.200000	0.200000 to 100.0000%
	Logical Pnt Nmbr	INT/N/ POP-UP	1	1 to 255
	Logical Pnt Type	TAB/N/ POP-UP	FUL	LTD, FUL, RAT, TOT, INC, ASP, or ADP
	Point Address			
	(For AHU)	INT/N/ POP-UP	1	1 to 8
	(For VAV, UNT)	INT/N/ POP-UP	1	1 to 12
	(For OTHER)	INT/N/ POP-UP	1	1 to 256
	(For VMA)	INT/N/ POP-UP	1	1 to 5
	Point Type	READ ONLY/N/ POP-UP	AI	AI
	Hardware Reference			
	(For DR9100)	TAB/N	AI1	AI1-4
	(For DC9100)	TAB/N	AI1	AI1-8, Total1-2[1] (for DO9100 only)
	(For LCP)	TAB/N	AI1	AI1-8
	(For DX9100)	TAB/N	AI1	AI1-8, XT1-8AI1-8, CNT1-8[1], PM1-12AC1-8[1], XT1-8CNT1-8[1]
	(For DX91ECH)	TAB/N	AI1	AI1-8, XT1-8AI1-8, CNT1-8[1], PM1-12AC1-8[1], XT1-8CNT1-8[1]

[1] Restrictions apply to all AI objects mapped to these H/W references. Please refer to AI documentation.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	(For XT9100, XTM)	TAB/N	AI1	AI1-8, CNT1-8[1]
	(For TC9100)	TAB/N	AI1	AI1-4
	(For LON)	TAB/N	01AI01	Definition not available
Engineering Data	Analog Units	STR	DEG F	6 characters
	Decimal Position	INT	1	0 to 3
	High Alarm Limit	ANA	80.00000	Real or Blank (not defined)
	Low Alarm Limit	ANA	40.00000	Real or Blank (not defined)
	Setpoint	ANA	55.00000	Real or Blank (not defined) [2]
	Normalband	ANA	10.00000	Real > 0.0 or Blank (not defined) [2]
	Differential	ANA	Blank	Real > 0.0 or Blank (not defined)
2 Either both Setpoint and Normalband must be defined, or both must be undefined (blank).				

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Parameters	Warning Delay	ANA	1	0 to 255 minutes
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Warning	TAB	NON	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Warning No.	INT	0	0 to 255
	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Linear Parameter For Standard Ranges

For DCM and DCM140, use standard ranges 1 to 25, and 34 to 111.
For FPU, use standard ranges 26 to 33.

Range	Type	Parm 1	Parm 2	Parm 3	Parm 4
1	DEGF 1000 ohm Ni	- 49.929	89.144	-4787.6	221.94
2	DEGF 1000 ohm Pl	-116.32	100.75	979.87	10.941
3	DEGF CPD silicon	- 16.936	66.482	-5624.1	414.23
4	DEGF 100 ohm Pl	- 49.972	71.417	437.8	3.0646
5	DEGC 1000 ohm Ni	- 45.516	49.525	-2659.7	123.3
6	DEGC 1000 ohm Pl	- 82.397	55.987	6.2715	543.93
7	DEGC CPD silicon	- 27.197	36.979	-3153.7	235.82
8	DEGC 100 ohm Pl	- 45.54	39.676	243.26	1.6975
9	WC IDP001 0/.1	- 0.025	0.03656	0.0	0.0
10	WC IDP002 0/.25	- 0.0625	0.09139	0.0	0.0
11	WC IDP005 0/.5	- 0.125	0.18278	0.0	0.0
12	WC IDP010 0/1	- 0.25	0.36557	0.0	0.0
13	WC IDP030 0/3	- 0.75	1.09670	0.0	0.0
14	WC IDP050 0/5	- 1.25	1.82784	0.0	0.0
15	WC IDP100 0/10	- 2.5	3.65568	0.0	0.0
16	mBAR IDP001 0/0.249	-0.06228	0.09106	0.0	0.0
17	mBAR IDP002 0/0.623	- 0.15576	0.22775	0.0	0.0
18	mBAR IDP005 0/1.245	- 0.31125	0.45513	0.0	0.0
19	mBAR IDP010 0/2.491	- 0.62275	0.91063	0.0	0.0
20	mBAR IDP030 0/7.472	- 1.8680	2.73152	0.0	0.0
21	mBAR IDP050 0/12.45	- 3.1125	4.55132	0.0	0.0
22	mBAR IDP100 0/24.91	- 6.2275	9.10629	0.0	0.0
The following ranges display mA or VDC. Use span function to show appropriate engineering units.					
23	4/20 mA input	0.0	5.86081	0.0	0.0
24	0/10 VDC input	0.0	2.92454	0.0	0.0
Note: If you use the AI with a Delta-P sensor to calculate flow, you cannot use ranges 23 or 24; you must range the AI to match the sensor.					
25	W560/HE-6110 0/5 VDC	0.0	58.4908	0.0	0.0

Continued on next page . . .

Linear Parameter For Standard Ranges (Cont.)

For FPU, use standard ranges 26 to 33.

For DCM and DCM140, use standard ranges 1 to 25, and 34 to 111.

Range	Type	Parm 1	Parm 2	Parm 3	Parm 4
26	ANT101 0 to 100F	-0.03364	52.1918	-1741.36	57.6882
27	ANT102 40 to 140F	39.94782	52.2460	-1783.97	67.5819
28	ANT103 -50 to 150F	- 49.98669	111.394	-7642.14	464.659
29	ANT104 -50 to 250F	- 50.05439	179.105	-19755.2	1930.92
30	ANT101 -18 to 38C	- 17.7962	28.9954	-967.423	32.0491
31	ANT102 4 to 60C	4.41553	29.0255	-991.095	37.5455
32	ANT103 -46 to 66C	- 45.5480	61.8858	-4245.63	258.144
33	ANT104 -46 to 121C	- 45.5857	99.5000	-10975.1	1072.73
If you are adding an ANV, ANC, or ANP analog card to an AI software object, you must calculate the linearization parameters. You'll find the equations for calculating linearization parameters later in this document.					
34	WC IDP250 0/25	-6.250000	9.139063	0.0	0.0
35	WC IDPB001 -0.1/0.1	-0.150000	0.073113	0.0	0.0
36	WC IDPB002 -0.25/0.25	-0.375000	0.182781	0.0	0.0
37	WC IDPB005 -0.5/0.5	-0.750000	0.365563	0.0	0.0
38	WC IDPB010 -1.0/1.0	-1.500000	0.731125	0.0	0.0
39	WC IDPBO25 -2.5/2.5	-3.75000	1.827813	0.0	0.0
40	WC IDPB050 -5/5	-7.500000	3.655625	0.0	0.0
41	WC DPT-2640-1 0/0.1	0.0	0.058490	0.0	0.0
42	WC DPT-2640-2 0/0.25	0.0	0.146225	0.0	0.0
43	WC DPT-2640-3 0/0.5	0.0	0.292450	0.0	0.0
44	WC DPT-2640-4 0/1	0.0	0.584900	0.0	0.0
45	WC DPT-2640-5 0/2.5	0.0	1.462250	0.0	0.0
46	WC DPT-2640-6 0/5	0.0	2.924500	0.0	0.0
47	WC DPT-2640-7 0/10	0.0	5.849000	0.0	0.0
48	WC DPT-2640-8 0/25	0.0	14.622500	0.0	0.0
49	WC DPT-2640-21 -0.1/0.1	-0.100000	0.116980	0.0	0.0
50	WC DPT-2640-22 -0.25/0.25	-0.250000	0.292450	0.0	0.0
51	WC DPT-2640-23 -0.5/0.5	-0.500000	0.584900	0.0	0.0
52	WC DPT-2640-24 -1/1	-1.000000	1.169800	0.0	0.0
53	WC DPT-2640-25 -2.5/2.5	-2.500000	2.924500	0.0	0.0
54	WC DPT-2640-26 -5/5	-5.000000	5.849000	0.0	0.0
55	WC DPT-2640-27 -10/10	-10.000000	11.698000	0.0	0.0
56	WC DPT-2640-28 -25/25	-25.000000	29.245000	0.0	0.0

Continued on next page . . .

Linear Parameter For Standard Ranges (Cont.)

For FPU, use standard ranges 26 to 33.

For DCM and DCM140, use standard ranges 1 to 25, and 34 to 111.

Range	Type	Parm 1	Parm 2	Parm 3	Parm 4
57	WC DPT-2641-1 0/0.1	-0.025000	0.03663	0.0	0.0
58	WC DPT-2641-2 0/0.25	-0.062500	0.091575	0.0	0.0
59	WC DPT-2641-3 0/0.5	-0.125000	0.18315	0.0	0.0
60	WC DPT-2641-4 0/1	-0.250000	0.3663	0.0	0.0
61	WC DPT-2641-5 0/2.5	-0.625000	0.915750	0.0	0.0
62	WC DPT-2641-6 0/5	-1.250000	1.831500	0.0	0.0
63	WC DPT-2641-7 0/10	-2.500000	3.663000	0.0	0.0
64	WC DPT-2641-8 0/25	-6.250000	9.157500	0.0	0.0
65	WC DPT-2641-21 -0.1/0.1	-0.150000	0.073260	0.0	0.0
66	WC DPT-2641-22 -0.25/0.25	-0.375000	0.183150	0.0	0.0
68	WC DPT-2641-23 -0.50/0.50	-0.750000	0.366300	0.0	0.0
67	WC DPT-2641-24 -1.0/1.0	-1.500000	0.732600	0.0	0.0
69	WC DPT-2641-25 -2.5/2.5	-3.750000	1.831500	0.0	0.0
70	WC DPT-2641-26 -5.0/5.0	-7.500000	3.663000	0.0	0.0
71	WC DPT-2641-27 -10.0/10.0	-15.000000	7.326000	0.0	0.0
72	WC DPT-2641-28 -25.0/25.0	-37.500000	18.315000	0.0	0.0
73	mBAR IDP250 0/62.28	-15.56875	22.76540	0.0	0.0
74	mBAR IDPB001 -0.249/0.249	-0.37365	0.18212	0.0	0.0
75	mBAR IDPB002 -0.623/0.623	-0.93450	0.45549	0.0	0.0
76	mBAR IDPB005 -1.245/1.245	-1.86750	0.91025	0.0	0.0
77	mBAR IDPB010 -2.491/2.491	-3.73650	1.82123	0.0	0.0
78	mBAR IDPB025 -6.228/6.228	-9.34125	4.55308	0.0	0.0
79	mBAR IDPB050 -12.45/12.45	-18.67500	9.10251	0.0	0.0
80	mBAR DPT-2640-1 0/0.249	0.0	0.14570	0.0	0.0
81	mBAR DPT-2640-2 0/0.623	0.0	0.36439	0.0	0.0
82	mBAR DPT-2640-3 0/1.245	0.0	0.72820	0.0	0.0
83	mBAR DPT-2640-4 0/2.491	0.0	1.45699	0.0	0.0
84	mBAR DPT-2640-5 0/6.228	0.0	3.64246	0.0	0.0
85	mBAR DPT-2640-6 0/12.450	0.0	7.28201	0.0	0.0
86	mBAR DPT-2640-7 0/24.910	0.0	14.56986	0.0	0.0
87	mBAR DPT-2640-8 0/62.275	0.0	36.42465	0.0	0.0

Continued on next page . . .

Linear Parameter For Standard Ranges (Cont.)

For FPU, use standard ranges 26 to 33.

For DCM and DCM140, use standard ranges 1 to 25, and 34 to 111.

Range	Type	Parm 1	Parm 2	Parm 3	Parm 4
88	mBAR DPT-2640-21 -0.249/0.249	-0.24910	0.29140	0.0	0.0
89	mBAR DPT-2640-22 -0.623/0.623	-0.62300	0.72879	0.0	0.0
90	mBAR DPT-2640-23 -1.245/1.245	-1.24500	1.45640	0.0	0.0
91	mBAR DPT-2640-24 -2.491/2.491	-2.49100	2.91397	0.0	0.0
92	mBAR DPT-2640-25 -6.228/6.228	-6.22750	7.28493	0.0	0.0
93	mBAR DPT-2640-26 -12.45/12.45	-12.45000	14.56401	0.0	0.0
94	mBAR DPT-2640-27 -24.91/24.91	-24.91000	29.13972	0.0	0.0
95	mBAR DPT-2640-28 -62.275/62.275	-62.27500	72.84930	0.0	0.0
96	mBAR DPT-2641-1 0/0.249	-0.06228	0.09125	0.0	0.0
97	mBAR DPT-2641-2 0/0.623	-0.15575	0.22820	0.0	0.0
98	mBAR DPT-2641-3 0/1.245	-0.31125	0.45604	0.0	0.0
99	mBAR DPT-2641-4 0/2.491	-0.62275	0.91245	0.0	0.0
100	mBAR DPT-2641-5 0/6.228	-1.55688	2.28113	0.0	0.0
101	mBAR DPT-2641-6 0/12.450	-3.11250	4.56044	0.0	0.0
102	mBAR DPT-2641-7 0/24.910	-6.22750	9.12453	0.0	0.0
103	mBAR DPT-2641-8 0/62.275	-15.56875	22.81133	0.0	0.0
104	mBAR DPT-2641-21 -0.249/0.249	-0.37365	0.18249	0.0	0.0
105	mBAR DPT-2641-22 -0.623/0.623	-0.93450	0.45641	0.0	0.0
106	mBAR DPT-2641-23 -1.245/1.245	-1.86750	0.91209	0.0	0.0
107	mBAR DPT-2641-24 -2.491/2.491	-3.73650	1.82491	0.0	0.0
108	mBAR DPT-2641-25 -6.228/6.228	-9.34125	4.56227	0.0	0.0
109	mBAR DPT-2641-26 -12.45/12.45	-18.67500	9.12087	0.0	0.0
110	mBAR DPT-2641-27 -24.91/24.91	-37.36500	18.24907	0.0	0.0
111	mBAR DPT-2641-28 -62.275/62.275	-93.41250	45.62267	0.0	0.0

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	ANA	FB
	WRITE	WRIT	WR
Input (Commands)	ALARMS	CMD	AC
	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RES_TOT	CMD	RT
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	WARNINGS	CMD	WC
Output (GPL)	READ	READ	RD
Output (Attributes)	HI_ALARM	BIN	HA
	HI_WARN	BIN	HW
	LO_ALARM	BIN	LA
	LO_WARN	BIN	LW
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	ANA	V

Reliability

The AI object becomes unreliable when the hardware it represents goes offline or reports an unreliable status. The following AI attributes also become unreliable: AD_COUNT, DISPLAY, FLTR_VAL, HI_ALARM, HI_WARN, LO_ALARM, LO_WARN, NORMAL, STATUS, PRE_VAL, and VALUE.

The AI object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an AI object block sends unreliable data to the object, the WRIT block will execute, but the AI object will ignore the write and remain in its current state.

Example

The AI object block in this example (Figure 8) represents outside air temperature (AHU1\OA-TEMP). It is commanded to new alarm limits and differential based on the state of the BD object (AHU1\SUM-WINT), which determines summer and winter mode. When the mode changes from summer to winter, the CMD block sends to the AI object a high limit of 70.0°F, low limit of 38.0°F, and a differential of 2.0°F. When the mode changes from winter to summer, the AI object is updated with a high limit of 95.0°F, low limit of 50.0°F, and a differential of 2.0°F.

A period of 00:00:00 is defined for this process since it should run only when triggered by the BD object.

Note: The CMD and NOT blocks, operation blocks, must be placed in a process.

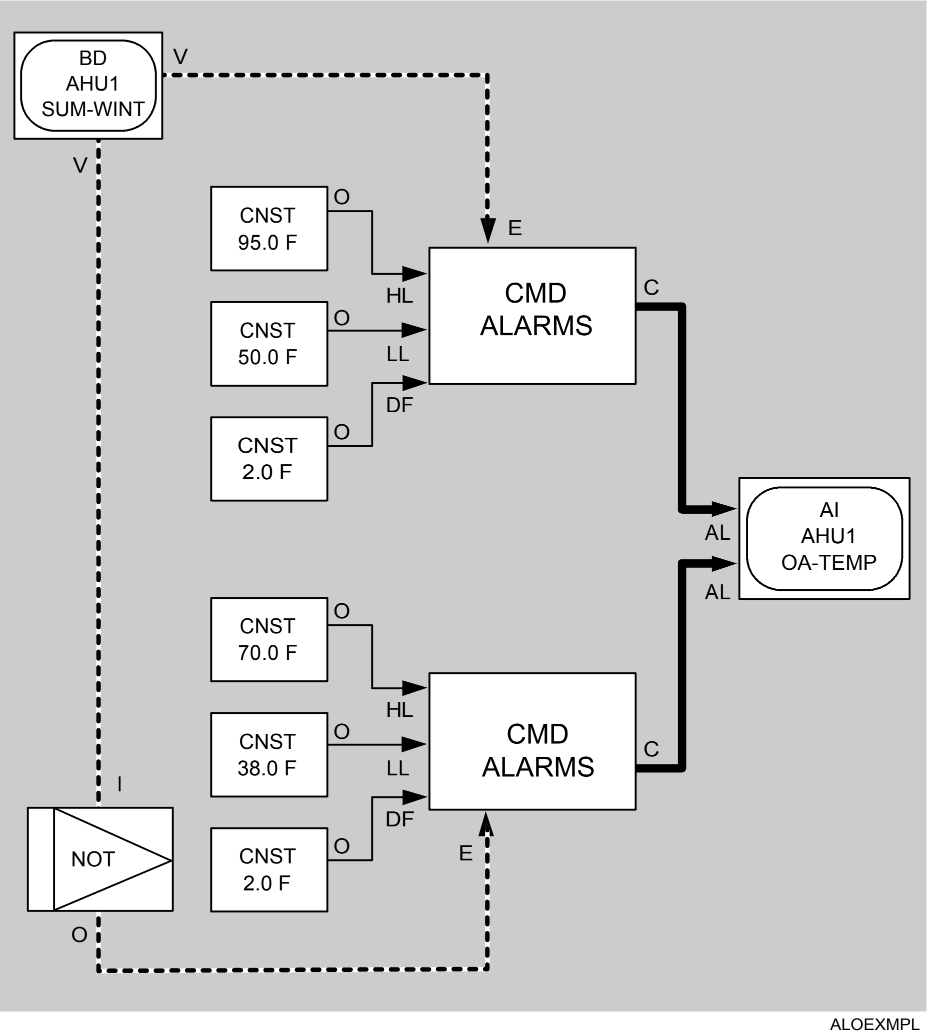


Figure 8: AI Object Example

Notes: The AI block can reference an AI software object mapped to a LON device. However, the AI software object must already be present in the archive database. You cannot create it by setting an AI block in the GPL editor.

The AI block will not show what hardware reference the AI actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

AOD (Analog Output Digital) Object

Category

Input/Output

Purpose

Creates a software representation of an analog output used for position control, such as opening a damper 25%.

Details

Figure 9 shows a general model of how the AOD object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Analog Output Digital (AOD) Technical Bulletin*.

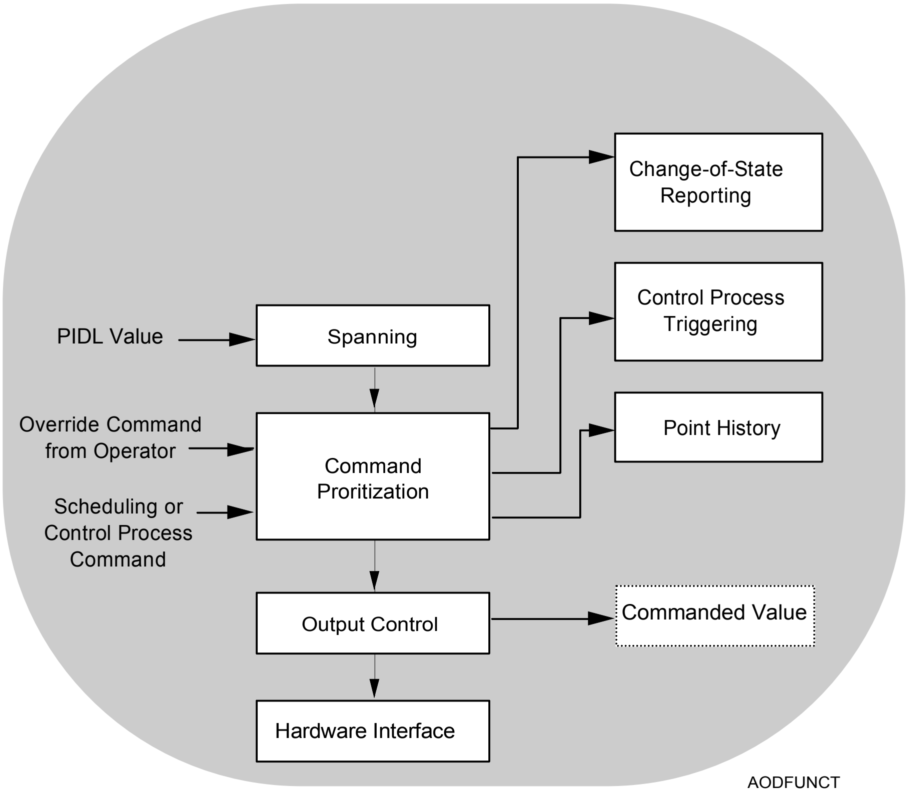

Change-of-State Reporting
Control Process Triggering
PIDL Value
Spanning
Point History
Override Command from Operator
Command Proritization
Scheduling or Control Process Command
Output Control
Commanded Value
Hardware Interface
AODFUNCT

Figure 9: AOD Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140
	Slot Number	INT/N	1	1 to 10
	Point Type			
	(For DCM)	TAB/N	PROP	PROP or INCR
	(For DCM140)	TAB/N	PROP	PROP, INCR, or MAO
	Subslot Number	INT/N POP-UP	1	1 to 2
	Step Ratio	ANA/ POP-UP	0.900000	Real
	Saturation Size	INT/ POP-UP	90	0 to 255 seconds
Engineering Data	Analog Units	STR	PCT	6 characters
	Decimal Position	INT	1	0 to 3

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Auto Restore	BIN	Y	Y (Yes) or N (No)
Parameters	Span Low Input	ANA	Blank	Real or blank (not defined)
	Span High Input	ANA	Blank	Real or blank (not defined)
	Span Low Output	ANA	Blank	Real or blank (not defined)
	Span High Output	ANA	Blank	Real or blank (not defined)
Report Type	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr. No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	VALUE	ANA	V
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RELEASE	CMD/2CMD	R
	RES_TOT	CMD	RT
	SET_AOD	CMD	SA
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD
Output (Attributes)	HOA	BIN	HO
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	ANA	V

Reliability

The AOD object becomes unreliable when it goes offline and the object is being controlled by the DCM. However, the object can be offline and reliable if it currently has a command pending at Priority 1 or 2. When the object is unreliable, the following attributes are also unreliable: DISPLAY, HOA, and VALUE.

The AOD object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an AOD object block sends unreliable data to the object, the WRIT block will execute, but the AOD object will ignore the write and remain in its current state.

Example

This example uses an AOD object block in a static pressure loop (Figure 10). If smoke is detected by the BI object (AHU1\SMK-1), the CMD block sends a command of 100% Open to the AOD object (AHU1\SF-INL), which tells the hardware to open the vanes. The AI input for the PIDL loop (AHU1\STAT-PRS) is from AHU1\STATIC.

This example assumes that the SET_AOD command to AHU1\SF-INL is released by some other process when the building returns to normal condition.

A period of 00:00:00 is specified because this process is triggered when the value of the BI object changes from Off to On or On to Off.

Note: The CMD block, an operation block, must be placed in a process.

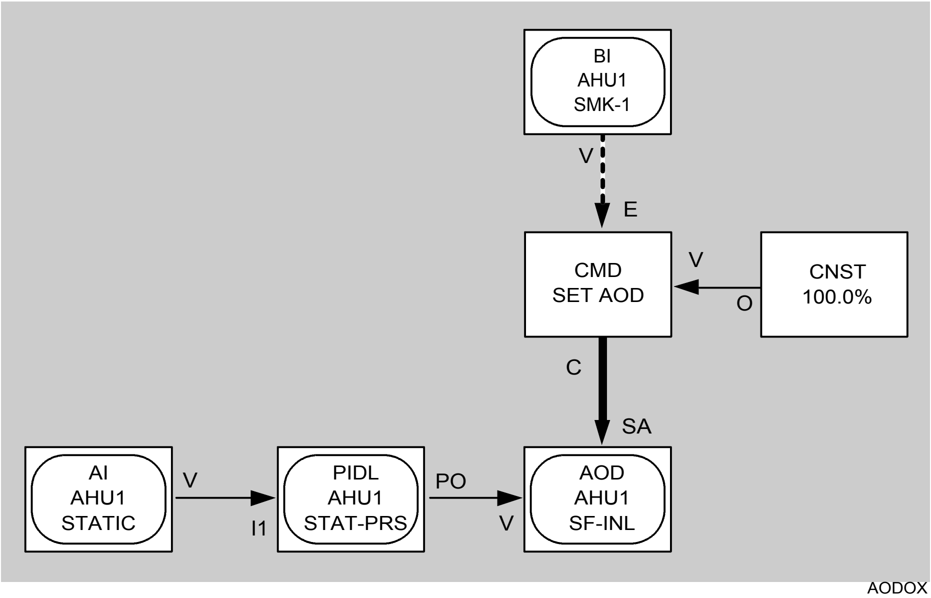


Figure 10: AOD Object Example

AOS (Analog Output Setpoint) Object

Category

Input/Output

Purpose

Creates a software representation of an analog output device used for position control or remote setpoint adjustment.

Details

Figure 11 shows a general model of how the AOS object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Analog Output Setpoint (AOS) Object Technical Bulletin*.

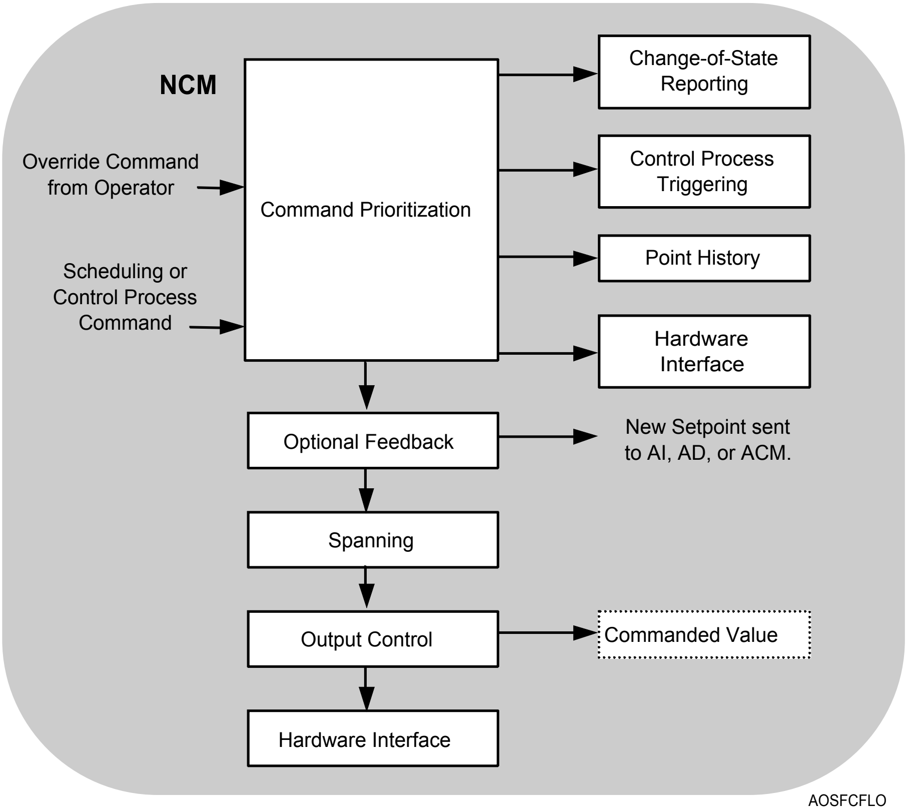

NCM
Override Command from Operator
Scheduling or Control Process Command
Command Prioritization
Change-of-State Reporting
Control Process Triggering
Point History
Hardware Interface
Optional Feedback
New Setpoint sent to AI, AD, or ACM.
Spanning
Output Control
Commanded Value
Hardware Interface
AOSFCFLO

Figure 11: AOS Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140, FPU, DSC8500, AHU, VAV, UNT, OTHER, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, VMA, LON[1]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For FPU)	INT/N/ POP-UP	1	1 to 16
	Span Low Input	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span High Input	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span Low Output	ANA/ POP-UP	Blank	Real or blank (not defined)
	Span High Output	ANA/ POP-UP	Blank	Real or blank (not defined)
	Point Type			
	(For DCM)	TAB/N/ POP-UP	PROP	PROP or INCR
	(For DCM140)	TAB/N/ POP-UP	PROP	PROP, INCR, or MAO
	(For AHU, VAV, VMA, UNT, OTHER)	READ ONLY/N/ POP-UP	AO	AO, ADF, ADI
	Logical Pnt Nmbr	INT/N/ POP-UP	1	1 to 255
	Logical Pnt Type	TAB/N/ POP-UP	INC	INC, ASP, or ADP

[1] LON Hardware type not definable from GPL. See *Example* section for details.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Point Address			
	(For AHU and AO)	INT/N/ POP-UP	1	1 to 8
	(For AHU and ADF, ADI)	INT/N/ POP-UP	129	129 to 256
	(For VAV, UNT and AO)	INT/N/ POP-UP	1	1 to 8
	(For VAV, UNT and ADF, ADI)	INT/N/ POP-UP	65	65 to 256
	(For OTHER)	INT/N/ POP-UP	1	1 to 256
	(For VMA and AO)	INT/N/ POP-UP	1	1 to 2
	(For VMA, VAV, UNT and ADF, ADI)	INT/N/ POP-UP	65	65 to 256
	Step Ratio	ANA/ POP-UP	0.900000	Real
	Saturation Size	INT/ POP-UP	90	0 to 255 seconds
	Local Control[2]	BIN/N/ POP-UP	N	Y (Yes) or N (No)
	Hardware Reference			
	(For DC9100)	TAB/N	OUT1	OUT1-8, ACO1-4
	(For LCP)	TAB/N	OUT1	OUT1-8, ACO1-4
	(For DX9100)	TAB/N	OUT1	OUT1-14, ACO1-8, XT1-8AO1-8
	(For DX91ECH)	TAB/N	OUT1	OUT1-14, ACO1-8 XT1-8AO1-8
	(For XT9100, XTM)	TAB/N	AO8	AO1-8
	(For LON)[1]	TAB/N	01AO02	Definition not available
Engineering Data	Analog Units	STR	DEG F	6 characters
	Decimal Position	INT	1	0 to 3
Feedback	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters

[1] LON Hardware type not definable from GPL. See *Example* section for details.

[2] Local Control is not available on XT9100 and XTM.

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Auto Restore	BIN	Y	Y (Yes) or N (No)
Parameters	Initial Value	ANA	Blank	Real or blank (not defined)
Report Type	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr. No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RELEASE	CMD	R
	RELEASE3	CMD/2CMD	R3
	RES_TOT	CMD	RT
	SET_AOS	CMD	SA
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	FEEDBACK	ANA	FB
	READ	READ	RD
Output (Attributes)	HOA	BIN	HO
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	ANA	V

Reliability

The AOS object is unreliable when its HOA attribute becomes unreliable. If the AOS object goes offline, its VALUE attribute remains reliable.

The AOS object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an AOS object block sends unreliable data to the object, the WRIT block will execute, but the AOS object will ignore the write and remain in its current state.

Example

The AOS object block in the following example (Figure 12) is a setpoint for a pneumatic controller (AHU1\SETPOINT). The PIR block calculates a new setpoint for the AOS based on outside air temperature (AHU1\OA-TEMP). The CMD block sends the new setpoint to the AOS object.

A period of 00:03:00 minutes is defined for this process, which means the process will execute once every three minutes.

Note: The PIR and CMD blocks, both operation blocks, must be placed in a process.

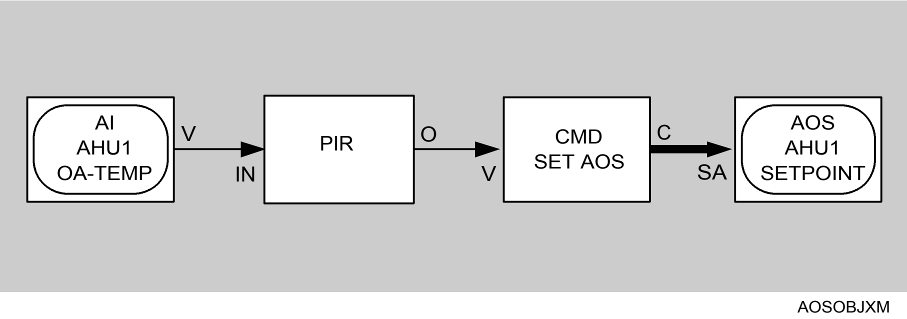


Figure 12: AOS Object Example

Notes: The AOS block can reference an AOS software object mapped to a LON device. However, the AOS software object must already be present in the archive database. You cannot create it by setting an AOS block in the GPL editor.

The AOS block will not show what hardware reference the AOS actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

BD (Binary Data) Object

Category

Data

Purpose

Creates a storage location for binary data. The BD object has no associated hardware. It can receive a command or a binary value from a binary attribute of an associated object.

Details

Figure 13 shows a general model of how the BD object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Binary Data (BD) Object Technical Bulletin.*

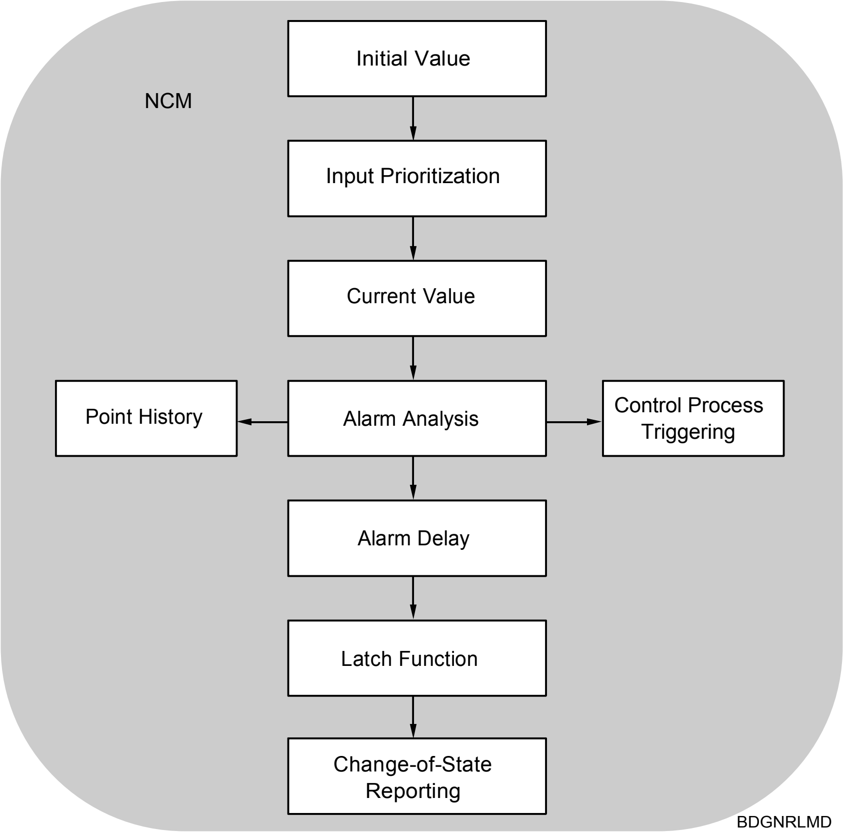


Figure 13: BD General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
	NC Name	READ ONLY/N	Blank	8 characters
Engineering Data	State 0 Units	STR	OFF	6 characters
	State 1 Units	STR	ON	6 characters
Associated Input	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Attribute Name	STR/N	Blank	8 characters

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	N	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Latching Point	BIN	N	Y (Yes) or N (No)
Parameters	Normal State	TAB	NONE	STATE 0, STATE 1, or NONE
				STATE 1 or NONE
	Initial Value	TAB	STATE 0	STATE 0 or STATE 1
	Alarm Delay	INT	30	0 to 255 seconds
	Delay All Alarms	BIN	N	Y (Yes) or N (No)
	Adjust Disabled	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Connections

Connections	Name	Type	Label
Input (GPL)	ASSOC IN	BIN	AS
	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	BIN	FB
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RELEASE	CMD	R
	RES_TOT	CMD	RT
	SET_BD	CMD	SB
	UNL_REP	CMD/2CMD	UR
Continued on next page . . .			

Connections (Cont.)	Name	Type	Label
	UNL_TRG	CMD/2CMD	UT
	UNLATCH	CMD	UL
Output (GPL)	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	EARLY_TM	TIM	ES
	LATE_TM	TIM	LS
	LATCH	BIN	L
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	BIN	V

Reliability

The BD object becomes unreliable if the assigned binary attribute it is sampling for its value becomes unreliable. It also becomes unreliable if GPL commands an unreliable value to the object. The following BD attributes become unreliable if the object is unreliable: ALARM, DISPLAY, NORMAL, STATUS, and VALUE.

The BD object ignores all commands that contain an unreliable parameter. However, the BD object does accept a SET_BD command whose Value parameter is unreliable. In this case, the command forces the BD object to unreliable.

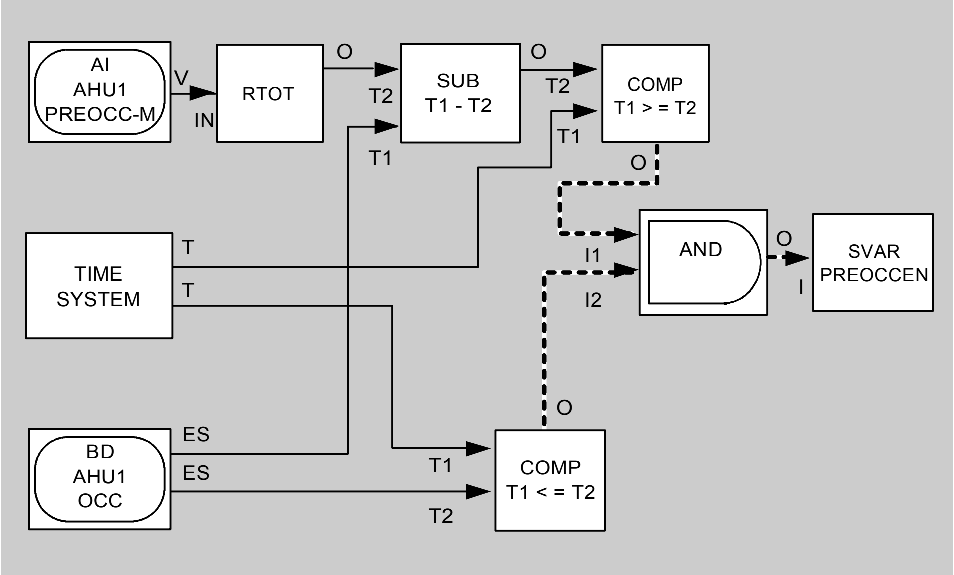
If a WRIT block that is connected to a BD object block sends unreliable data to the object, the WRIT block will execute, but the BD object will ignore the write and remain in its current state.

Example

In this example, (Figure 14), the Early Start attribute of a BD object is used in a preoccupancy mode strategy. When the process executes, the preoccupancy time in minutes (AHU1\PREOCC-M) is read and converted to time format (RTOT). The early start time (AHU1\OCC\EARLY_TM) is then subtracted from the amount of preoccupancy time in time format. If the amount of preoccupancy time is greater than or equal to system time (SYSTEM), and the system time is less than or equal to occupancy time, then preoccupancy mode is enabled (PREOCCEN).

A period of 00:02:00 is defined for this process, which means it will run once every 2 minutes.

Note: All operation blocks in this example must be placed in a process. These include: RTOT, SUB, AND, CNST, SVAR, TIME, and both COMP blocks.



BDOBEX

Figure 14: BD Object Example

BI (Binary Input) Object

Category

Input/Output

Purpose

Creates a software representation of a hardware sensor that is monitoring a two position field condition.

Details

Figure 15 shows a general model of how the BI object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Binary Input (BI) Object Technical Bulletin*.

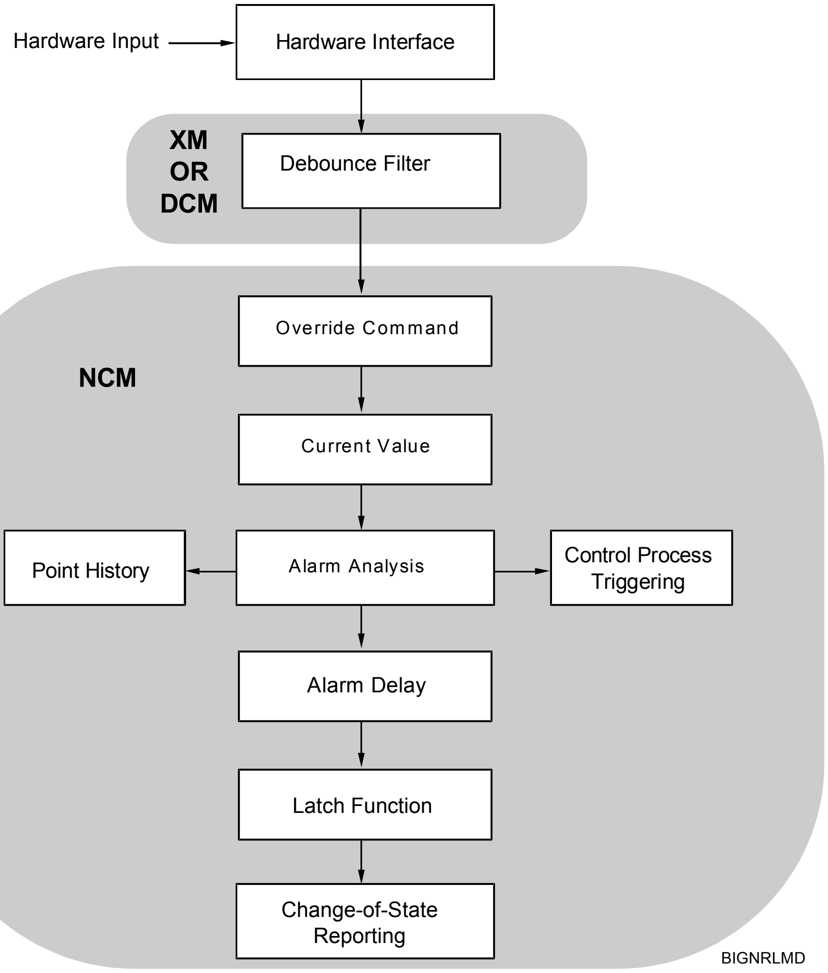


Figure 15: BI General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type[2]	TAB/N	DCM	DCM, DCM140, FPU, DSC8500, XBN, XRM/XRL/2X, AHU, VAV, VMA, UNT, OTHER, D600, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For XBN)	INT/N/ POP-UP	1	1 to 32
	(For XRM/ XRL/2X)	INT/N/ POP-UP	1	1 to 8
	(For FPU)	INT/N/ POP-UP	1	1 to 16
	Point Type			
	(For DCM and DCM140)	TAB/N/ POP-UP	BI	BI or MBI
	(For XBN, XRM/XRL/2X)	TAB/N/ POP-UP	SINGLE	SINGLE or FORM C
	(For AHU, VAV, VMA, UNT, OTHER)	READ ONLY/N/ POP-UP	BI	BI
	Debounce Filter			
	(For DCM and DCM140)	INT/ POP-UP	2	1 to 255 seconds
	(For XBN, XRM/XRL/2X)	INT/ POP-UP	24	12 to 3060 msec (multiples of 12 only)
	Subslot Number	INT/N/ POP-UP	1	1 or 2
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)

[1] LON Hardware type not definable from GPL. See *Example* section for details.

[2] To define an XRE for HW Type, select XRM/XRL/2X.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Binary Type	TAB/N/ POP-UP	BIN101	BIN101, BIF101, BIS101, SST101, or SST102
	Logical Pnt Addr	INT/N/ POP-UP	1	1 to 255
	Logical Pnt Type	TAB/N/ POP-UP	CON	CON, MAN, BSP, or BDP
	Point Address			
	(For AHU)	INT/N/ POP-UP	1	1 to 8
	(For VAV, UNT)	INT/N/ POP-UP	1	1 to 12
	(For OTHER)	INT/N/ POP-UP	1	1 to 256
	(For VMA and BI)	INT/N/ POP-UP	1	1 to 3
	Reader Number	INT/N/ POP-UP	1	1 to 16
	BI Point Number	INT/N/ POP-UP	1	1 to 8
	Input Type	READ ONLY/ POP-UP	2-STATE	2-STATE
	PT Enabled	BIN/ POP-UP	Y	Y (Yes) or N (No)
	Suppress TZ	INT/ POP-UP	0	0 to 8
	Alarm if Set	BIN/ POP-UP	N	Y (Yes) or N (No)
	Quiet if Reset	BIN/ POP-UP	N	Y (Yes) or N (No)
	Hardware Reference			
	(For DR9100)	TAB/N	WIN	WIN, OCC, AIRQ
	(For DC9100)	TAB/N	DI1	DI1-8, LCM1-4
	(For LCP)	TAB/N	DI1	DI1-8, LCM1-4
	(For DX9100)	TAB/N	DI1	DI1-8, XT1-8DI1-8, LRS1-32
	(For DX91ECH)	TAB/N	DI1	DI1-8, XT1-8DI1-8, LRS1-64
	(For XT9100, XTM)	TAB/N	1DI1	1DI1-8 2DI1-8
	(For TC9100)	TAB/N	WIN	WIN, OCC, AIRQ. ALM, AFM
	(For LON)[1]	TAB/N	01BI07	definition not available
Engineering Data	State 0 Units	STR	OFF	6 characters
	State 1 Units	STR	ON	6 characters

[1] LON Hardware type not definable from GPL. See *Example* section for details.

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Latching Point	BIN	N	Y (Yes) or N (No)
Parameters	Normal State	TAB	NONE	STATE 0, STATE 1 or NONE
	Alarm Delay	INT	30	0 to 255 seconds
	Delay All Alarms	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	BIN	FB
	WRITE	WRIT	WR
Input (Commands)	ALARMS	CMD	AC
	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RES_TOT	CMD	RT
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	UNLATCH	CMD	UL
Output (GPL)	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	LATCH	BIN	L
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	BIN	V

Reliability

The BI object becomes unreliable when the hardware it represents goes offline or reports an unreliable status. The following BI attributes also become unreliable: ALARM, DISPLAY, NORMAL, STATUS, and VALUE.

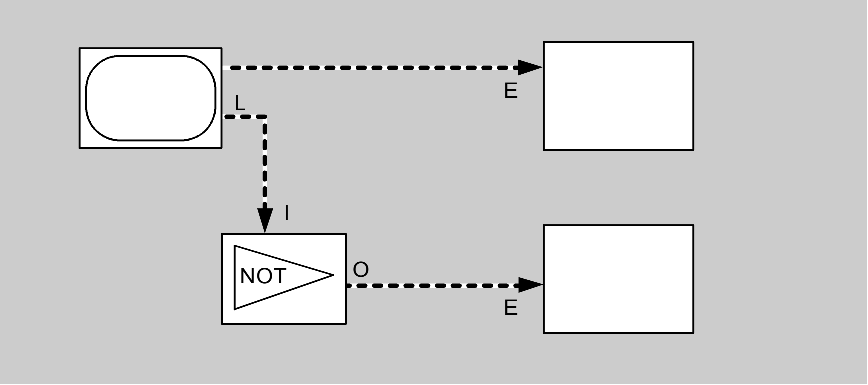
The BI object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to a BI object block sends unreliable data to the object, the WRIT block will execute, but the BI object will ignore the write and remain in its current state.

Example

The BI object block in the following example (Figure 16) is a latching point (AHU1\SWITCH). When it latches, the statement AHU1\SWITCH IS LATCHED is printed; when it unlatches, the statement AHU1\SWITCH IS UNLATCHED is printed.

A period of 00:00:00 is defined for this process since it needs to run only when triggered by the BI object.

Note: The NOT and PRNT blocks, operation blocks, must be placed in a process.



BOBJEXPL

Figure 16: BI Object Example

Notes: The BI block can reference a BI software object mapped to a LON device. However, the BI software object must already be present in the archive database. You cannot create it by setting a BI block in the GPL editor.

The BI block will not show what hardware reference the BI actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

BO (Binary Output) Object

Category

Input/Output

Purpose

Creates a software representation of a two position controlled device.

Details

Figure 17 shows the functional flow of how the BO object operates. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, Binary Output (BO) Technical Bulletin.*

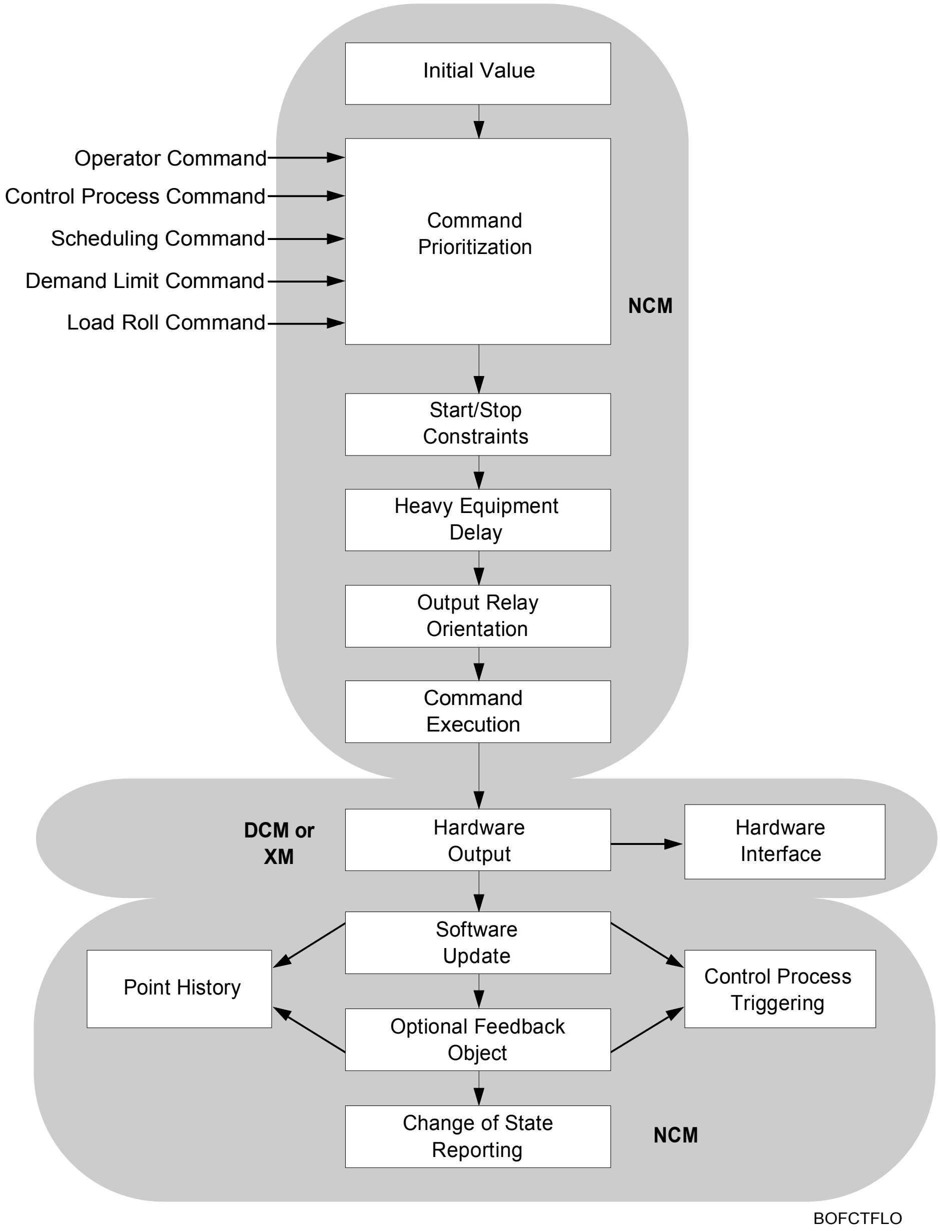


Figure 17: BO Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type[2]	TAB/N	DCM	DCM, DCM140, FPU, DSC8500, XRM/XRL/2X, AHU, VAV, VMA, UNT, OTHER, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For XRM/ XRL/2X)	INT/N/ POP-UP	1	1 to 8
	(For FPU)	INT/N/ POP-UP	1	1 to 16
	Point Type			
	(For DCM and DCM140)	TAB/N/ POP-UP	MAINT.	MAINTAINED or LATCHED
	(For AHU, VAV, VMA, UNT, OTHER)	TAB/N/ POP-UP	BO	BO or BD
	Pulse Duration			
	(For DCM and DCM140)	INT/ POP-UP	200	20 to 5100 msec (multiples of 20 only)
	(For XRM/XRL/2X)*	INT/ POP-UP	252	12 to 3060 msec (multiples of 12 only)
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)
	Binary Type	TAB/N/ POP-UP	SST101	SST101 or SST102
	Logical Pnt Nmbr	INT/N/ POP-UP	1	1 to 255
	Logical Pnt Type	TAB/N/ POP-UP	MOM	MOM, MAN, BOF, BSP, or BDP

[1] LON hardware type not definable from GPL. See *Example* section for details.

[2] To define an XRE for HW Type, select XRM/XRL/2X. The Pulse Duration field automatically pops up if the HW Type is XRM/XRL/2X. However, if you are using the XRM/XRL/2X option to define an XRE, ignore the Pulse Duration field.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Point Address			
	(For AHU)	INT/N/ POP-UP	1	1 to 10 (for BO) 193 to 256 (for BD)
	(For VAV, UNT)	INT/N/ POP-UP	1	1 to 14 (for BO) 225 to 256 (for BD)
	(For OTHER)	INT/N/ POP-UP	1	1 to 256 (for BO and BD)
	(For VMA and BO type)	INT/N/ POP-UP	1	1 to 5
	(For VMA and BD type)	INT/N/ POP-UP	65	65 to 256
	Hardware Reference			
	(For DR9100)	TAB/N	DO3	DO3-7, STUP, SOFF
	(For DC9100)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-4
	(For LCP)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-4
	(For DX9100)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	(For DX91ECH)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	(For XT9100, XTM)	TAB/N	1DO1	1DO1-8 2DO1-8
	(For TC9100)	TAB/N	DO3	DO1-7 STUP SOFF
	(For LON)[1]	TAB/N	01BO13	Definition not available
	Local Control[2]	BIN/N/ POP-UP	N	Y (Yes) or N (No)
Engineering Data	State 0 Units	STR	OFF	6 characters
	State 1 Units	STR	ON	6 characters
Feedback	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters

[1] LON hardware type not definable from GPL. See *Example* section for details.

[2] Local Control is not available on XT9100 and XTM.

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Auto Restore	BIN	Y	Y (Yes) or N (No)
Parameters	Output Relay (CLOSED for START)	BIN	Y	Y (Yes) or N (No)
	Feedback (CLOSED for START)	BIN	Y	Y (Yes) or N (No)
	Initial Value	TAB	NONE	STATE 0, STATE 1 or NONE
	Heavy Equip Dlay	INT	5	0 to 255 seconds
	Min ON Time	INT	1	0 to 255 seconds
	Min OFF Time	INT	0	0 to 255 seconds
	Max Starts/Hour	INT	255	1 to 255
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr. No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_PRI	CMD	RP
	RES_TOT	CMD	RT
	START	CMD/2CMD	ST
	STOP	CMD/2CMD	SP
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	FEEDBACK	BIN	FB
	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	EARLY_ TM	TIM	ES
	HOA	BIN	HO
	LATE_TM	TIM	LS
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	BIN	V

Reliability

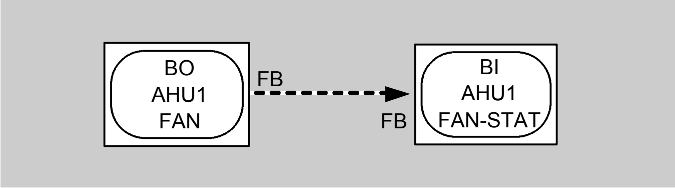
The BO object is always reliable, unless it has an associated feedback object. If it has a feedback object assigned, the BO object will become unreliable when the associated feedback object becomes unreliable. The following BO attributes also become unreliable: ALARM, DISPLAY, NORMAL, and STATUS.

The BO object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to a BO object block sends unreliable data to the object, the WRIT block will execute, but the BO object will ignore the write and remain in its current state.

Example

In this example (Figure 18), a BO object that represents fan start/stop (AHU1\FAN) provides feedback to the BI object (AHU1\FAN-STAT) when it Starts or Stops.

A period of 00:00:00 is defined for this process since it needs to run only when triggered by the BO object.



BOBJEX

Figure 18: BO Object Example

Notes: The BO block can reference a BO software object mapped to a LON device. However, the BO software object must already be present in the archive database. You cannot create it by setting a BO block in the GPL editor.

The BO block will not show what hardware reference the BO actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

LCG (Lighting Control Group) Object

Category

Controllers

Purpose

Creates a software representation of a group of lighting circuits that are controlled by the Intelligent Lighting Controller (ILC).

Details

The purpose of an LCG is to control lights (circuits) in areas that perform the same functions, such as an office area or a parking lot. Up to 32 individual LCG objects can be defined per ILC.

In the LCG database template, you define the software definition of the LCG and the ILC to which this group belongs. The entries include selecting the Blink feature, Cleaning Crew feature, and Relay Outputs.

Note: DDL and online generation allow definition of Event Scheduling for the LCG object. GPL does not allow this. If you use GPL to modify an existing LCG object (that has events defined through DDL or online generation), the Event Scheduling for the modified object will remain unchanged.

For more information, see the *Metasys Intelligent Lighting Controller Technical Manual (FAN 638.5)*.

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 character
	Object Name	STR/N	Blank	8 character
	Expanded ID	STR	Blank	24 character
Hardware	System Name	STR/N	Blank	8 character
	Object Name	STR/N	Blank	8 character
	Group Number	INT/N	1	1 to 32
Associated Switch	Switch Input No.	INT/N	0	0 to 32, 0=None
	Switch Type	TAB/N	MAINT.	MAINTAINED, MOMENTARY or DBL MOMENT
	Off Switch Inp No.	INT/N/ POP-UP	0	0 to 32, 0=None

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Override	Ovrd Def Delay	ANA	1.000000	0.0 to 99.9 hours
Parameters	Blnk at Turn OFF	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Template Fields (Third Screen)

Category	Field Name	Type	Default	Range/Choices
Relay Outputs	1-40	BIN	N	Y (Assign relay/No blink), N (No assignment), or B (Assign relay/Yes blink)
Cleaning Crew Switches	1-32	BIN	N	Y (Assign inputs) or N (No assignment)

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	OFF	CMD/2CMD	OF
	ON	CMD/2CMD	ON
	RELEASE	CMD	R
	RES_TOT	CMD	RT
	TIMED_ON	CMD	T
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD
Output (Attributes)	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	VALUE	BIN	V

Reliability

The LCG object becomes unreliable when it goes offline. The following attributes also become unreliable: DISPLAY and VALUE.

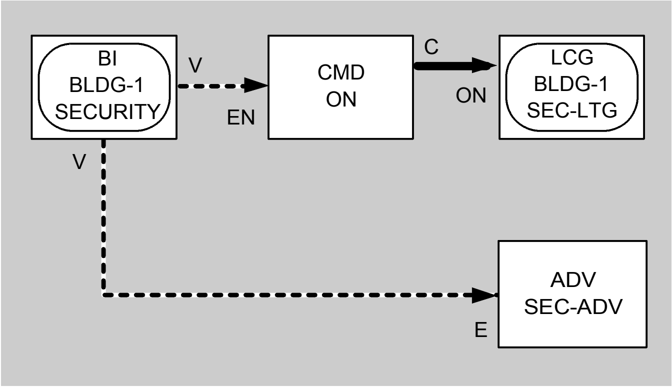
The LCG object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an LCG object block sends unreliable data to the object, the WRIT block will execute, but the LCG object will ignore the write and remain in its current state.

Example

In the following example, (Figure 19), a lighting control group is used in a security application. When the building security point is breached (BLDG-1\SECURITY), a command is issued to turn on the security lighting group (BLDG-1\SEC-LTG) and an advisory (SEC-ADV) is printed to notify the operator.

A period of 00:00:00 is defined for this process since it needs to run only when triggered by the BI object.

Note: The ADV and CMD blocks, both operation blocks, must be placed in a process.



LCGOBX

Figure 19: LCG Object Example

MSD Object

Category

Multistate

Purpose

Creates a storage location for multistate data. The MSD object has no associated hardware. It can receive a command or an analog READ value from an analog attribute of an associated object.

Details

Figure 20 shows a general model of how the MSD object operates. For more information, see the *Metasys European Technical Notes, Software Data Sheets, MSD Engineering*.

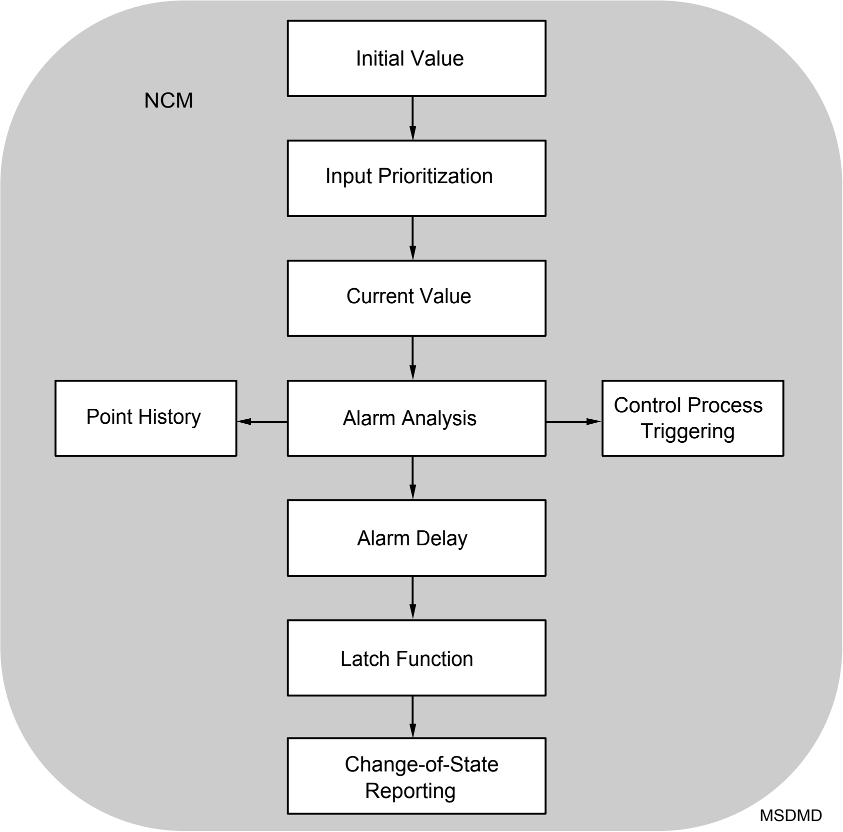


Figure 20: MSD General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
	NC Name	READ ONLY/N	Blank	8 characters
Engineering Data	Number of States	INT/N	2	2 to 4
	State 0 Units	STR	S0	6 characters
	State 1 Units	STR	S1	6 characters
	State 2 Units	STR	S2	6 characters
	State 3 Units	STR	S3	6 characters
Associated Input	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Attribute Name	STR/N	Blank	8 characters

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Latching Point	BIN	N	Y (Yes) or N (No)
Parameters	Normal State	TAB	NONE	STATE 0, STATE 1, STATE 2, STATE 3, or NONE
	Initial Value	TAB	STATE 0	STATE 0, STATE 1, STATE 2, or STATE 3
	Alarm Delay	INT	30	0 to 255 seconds
	Delay All Alarms	BIN	N	Y (Yes) or N (No)
	Adjust Disabled	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Connections

Connections	Name	Type	Label
Input (GPL)	ASSOC IN	BIN	AS
	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	ANA	FB
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RELEASE	CMD/2CMD	R
	RES_TOT	CMD	RT
	SET_MSD	CMD	SM
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	UNLATCH	CMD/2CMD	UL
Output (GPL)	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	EARLY_TM	TIM	ES
	LATE_TM	TIM	LS
	LATCH	BIN	L
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	STATE_0	BIN	S0
	STATE_1	BIN	S1
	STATE_2 *a	BIN	S2
	STATE_3 *a	BIN	S3
Note: *a Displayed only as appropriate for object's Number of States.			

Reliability

The MSD object becomes unreliable if the assigned analog attribute it is sampling for its value becomes unreliable. It also becomes unreliable if GPL commands an unreliable value to the object. The following MSD attributes become unreliable if the object is unreliable: ALARM, DISPLAY, NORMAL, STATE_n values, and STATUS.

The MSD object ignores all commands that contain an unreliable parameter. However, the MSD object does accept a SET_MSD command whose Value parameter is unreliable. In this case, the command forces the MSD object to unreliable.

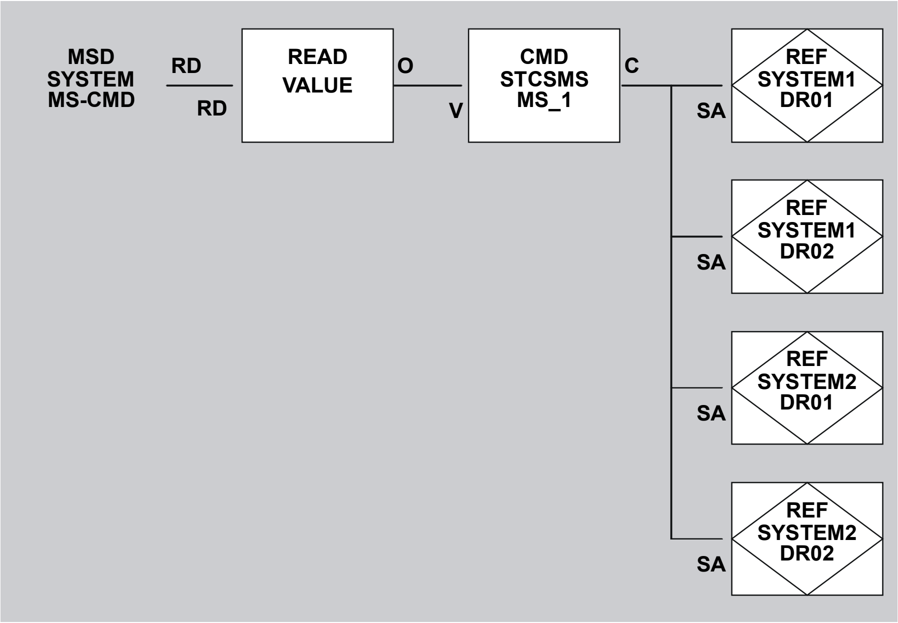
If a WRIT block that is connected to an MSD object block sends unreliable data to the object, the WRIT block will execute, but the MSD object will ignore the write and remain in its current state.

Example

In this example, (Figure 21), an MSD object is used to dispatch an operator command to a group of System 91 Controllers (DR-9100). The command values (units) are defined in the MSD object as OFF, STNDBY, and COMFRT to correspond to the three operating modes of the DR-9100 controllers. This enables the operator to set all controllers of the group to the desired mode with a single command.

The VALUE attribute of the MSD object is accessible through an analog READ block (note that the VALUE attribute is not directly accessible for GPL connections). The CMD block, in turn, commands the MS_1 attribute of each of the CS objects, which are each mapped to the MODC item of the corresponding DR-9100.

No period (i.e., 00:00:00) is defined for this process, since it is meant to trigger only when the operator issues a command.



msd-exmp

Figure 21: MSD Object Example

MSI Object

Category

Multistate

Purpose

Creates a software representation of a hardware sensor that is monitoring a multi-position field condition.

Details

Figure 22 shows a general model of how the MSI object operates. For more information, see the *Metasys European Technical Notes, Software Data Sheets, MSI Engineering*.

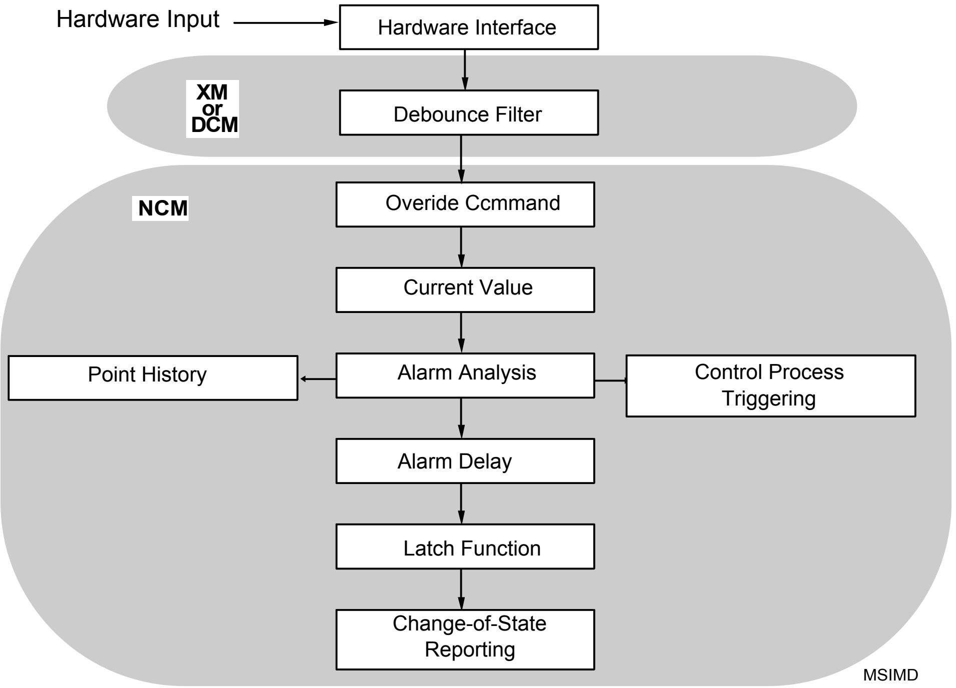


Figure 22: MSI General Model

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140, XBN, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For XBN)	INT/N/ POP-UP	1	1 to 32
	(For XRM/ XRL/2X)	INT/N/ POP-UP	1	1 to 8
	Hardware Reference			
	(For DR9100)	TAB/N	WIN	WIN, OCC, AIRQ
	(For DC9100)	TAB/N	DI1	DI1-8, LCM1-4
	(For LCP)	TAB/N	DI1	DI1-8, LCM1-4
	(For DX9100)	TAB/N	DI1	DI1-8, LRS1-32, XT1-8DI1-8
	(For DX91ECH)	TAB/N	DI1	DI1-8, LRS1-64 XT1-8DI1-8
	(For XT9100 and XTM with 2 states)	TAB/N	1DI1	1DI1-8 2DI1-8
	(For XTM with 2 states and wired 0)	TAB/N	1DI1	1DI1, 1DI3, 1DI5, 1DI7 2DI1, 2DI3, 2DI5, 2DI7
	(For XTM with 3 or 4 states with or without wired 0)	TAB/N	1DI1	1DI1, 1DI5 2DI1, 2DI5
	(For TC9100)	TAB/N	WIN	WIN, OCC, AIRQ, ALM, AFM
	(For LON)[1]	TAB/N	01MI06	Definition not available

[1] LON hardware type not definable from GPL. See *Example* section for details.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Point Type (For DCM, DCM140)	TAB/N/ POP-UP	BI	BI or MBI
	Debounce Filter			
	(For DCM and DCM140)	INT/ POP-UP	2	1 to 255 seconds
	(For XBN, XRM/XRL/2X)	INT/ POP-UP	24	12 to 3060 msec (multiples of 12 only)
	Subslot Number	INT/N/ POP-UP	1	1 or 2
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)
	Wired 0	BIN/N/ POP-UP	N	Y (Yes) or N (No)
Engineering Data	Number of States	INT/N	2	2 to 4 [2]
	State 0 Units	STR	S0	6 characters
	State 1 Units	STR	S1	6 characters
	State 2 Units	STR	S2	6 characters
	State 3 Units	STR	S3	6 characters

[2] Number of States can be > 2 for XRM2X, XRL2X or XTM. For all other controllers, must be 2.

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Latching Point	BIN	N	Y (Yes) or N (No)
Parameters	Normal State	TAB	NONE	STATE 0, STATE 1, STATE 2, STATE 3 or NONE
	Alarm Delay	INT	30	0 to 255 seconds
	Delay All Alarms	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operator Instr. No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	FEEDBACK	ANA	FB
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	RES_TOT	CMD	RT
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
	UNLATCH	CMD/2CMD	UL
Output (GPL)	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	LATCH	BIN	L
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	STATE_0	BIN	S0
	STATE_1	BIN	S1
	STATE_2 *a	BIN	S2
	STATE_3 *a	BIN	S3
Note: *a Displayed only as appropriate for object's Number of States.			

Reliability

The MSI object becomes unreliable when the hardware it represents goes offline or reports an unreliable status. The following MSI attributes also become unreliable: ALARM, DISPLAY, NORMAL, STATE_n values, and STATUS.

The MSI object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an MSI object block sends unreliable data to the object, the WRIT block will execute, but the MSI object will ignore the write and remain in its current state.

Example

The MSI object block in the following example (Figure 23) is a fan with three states, where STATE_0 is off. When the fan is switched to STATE_1, the statement AHU1\R_FAN IS IN STATE_1 is printed; when it is switched to STATE_2, the statement AHU1\R_FAN IS IN STATE_2 is printed.

A period of 00:00:00 is defined for this process since it needs to run only when triggered by the MSI object.

Note: The two PRNT operation blocks in this example must be placed in a process.

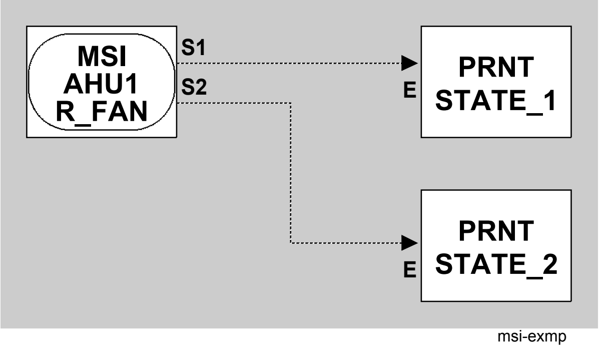


Figure 23: MSI Object Example

Notes: The MSI block can reference an MSI software object mapped to a LON device. However, the MSI software object must already be present in the archive database. You cannot create it by setting an MSI block in the GPL editor.

The MSI block will not show what hardware reference the MSI actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

MSO Object

Category

Multistate

Purpose

Creates a software representation of a multi-position controlled device.

Details

Figure 24 shows the functional flow of how the MSO object operates. For more information, see the *Metasys European Technical Notes, Software Data Sheets, MSO Engineering*.

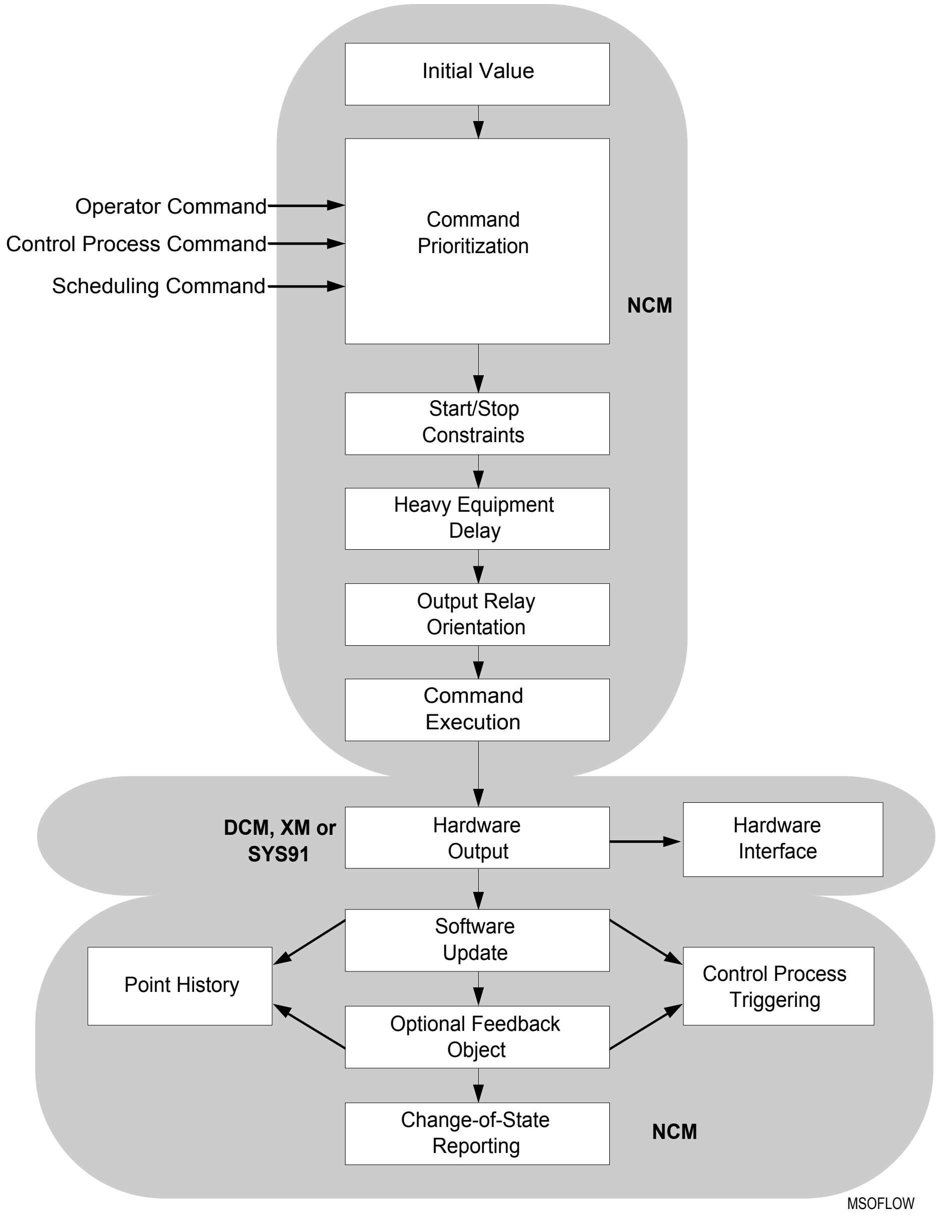


Figure 24: MSO Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For XRM/XRL/2X)	INT/N/ POP-UP	1	1 to 8
	Hardware Reference			
	(For DR9100)	TAB/N	DO3	DO3-7, STUP, SOFF
	(For DC9100)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-4
	(For LCP)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-4
	(For DX9100)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	(For DX91ECH)	TAB/N	DO3	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	(For XT9100 and XTM with 2 states)	TAB/N	1DO1	1DO1-8 2DO1-8
	(For XTM with 3 or 4 states)	TAB/N	1DO1	1DO1, 1DO5 2DO1, 2DO5
	(For TC9100)	TAB/N	DO3	DO1-7 STUP SOFF
	(For LON)[1]	TAB/N	01MO051	Definition not available
	Point Type	TAB/N/ POP-UP	MAINT.	MAINTAINED or LATCHED

[1] LON hardware type not available from GPL. See *Example* section for details.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	Pulse Duration			
	(For DCM and DCM140)	INT/ POP-UP	200	20 to 5100 msec (multiples of 20 only)
	(For XRM/XRL/2X)	INT/ POP-UP	252	12 to 3060 msec (multiples of 12 only)
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)
Feedback	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
Engineering Data	Number of States	INT/N	2	2 to 4[2]
	State 0 Units	STR	S0	6 characters
	State 1 Units	STR	S1	6 characters
	State 2 Units	STR	S2	6 characters
	State 3 Units	STR	S3	6 characters
Local Contact	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	HW Type	TAB/N	DCM	DCM, DCM140, XBN, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100
	Slot Number			
	(For DCM and DCM140)	INT/N/ POP-UP	1	1 to 10
	(For XBN)	INT/N/ POP-UP	1	1 to 32
	(For XRM/XRL/2X)	INT/N/ POP-UP	1	1 to 8
	Hardware Reference			
	(For DR9100)	TAB/N	WIN	WIN, OCC, AIRQ
	(For DC9100)	TAB/N	DI1	DI1-8, LCM1-4
	(For LCP)	TAB/N	DI1	DI1-8, LCM1-4
	(For DX9100)	TAB/N	DI1	DI1-8, LRS1-32
	(For DX91ECH)	TAB/N	DI1	DI1-8, LRS1-64 XT1-8DI1-8
	(For XT9100 and XTM without wired 0)	TAB/N	1DI1	1DI1-8 2DI1-8
	(For XTM with wired 0)	TAB/N	1DI1	1DI1, 1DI3, 1DI5, 1DI7 2DI1, 2DI3, 2DI5, 2DI7

[2] Number of States can be > 2 for XRM2X, XRL2X, or XTM. For all other controllers, must be 2.

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
	(For TC9100)	TAB/N	WIN	WIN, OCC, AIRQ, ALM, AFM
	Local Control*	BIN/POP-UP	n	Y (Yes) or N (No)
	* Local Control is not available on XT9100 and XTM.			
	Point Type	TAB/N/ POP-UP	BI	BI or MBI
	Subslot Number	INT/N/ POP-UP	1	1 or 2
	Debounce Filter			
	(For DCM and DCM140)	INT/ POP-UP	2	1 to 255 seconds
	(For XBN and XRM/XRL/2X)	INT/ POP-UP	24	12 to 3060 msec (multiples of 12 only)
	LED ON when CLO	BIN/ POP-UP	Y	Y (Yes) or N (No)
	Wired 0	BIN/N/ POP-UP	N	Y (Yes) or N (No)

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Enable PT Hist.	BIN	Y	Y (Yes) or N (No)
	Save PT History	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
	Auto Restore	BIN	Y	Y (Yes) or N (No)
Parameters	Output Relay (CLOSED for START)	BIN	Y	Y (Yes) or N (No)
	Feedback (CLOSED for START)	BIN	Y	Y (Yes) or N (No)
	Initial Value	TAB	NONE	STATE 0, STATE 1, STATE 2, STATE 3 or NONE
	Heavy Equip Dlay	INT	5	0 to 255 seconds
	Min ON Time	INT	1	0 to 255 seconds
	Min OFF Time	INT	0	0 to 255 seconds
	Max Starts/Hour	INT	255	1 to 255

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices

Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr. No.	INT	0	0 to 32767

Corrections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_PRI	CMD/2CMD	RP
	RES_TOT	CMD	RT
	SET_MSO	CMD	SM
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	FEEDBACK	ANA	FB
	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	EARLY_ TM	TIM	ES
	HOA	BIN	HO
	LATE_TM	TIM	LS
	LC_HOA	BIN	LO
	NORMAL	BIN	N
	OFFLINE	BIN	OL
	OVERRIDE	BIN	SO
	STATE_0	BIN	S0
	STATE_1	BIN	S1
	STATE_2 *a	BIN	S2
	STATE_3 *a	BIN	S3
Note: *a Displayed only as appropriate for object’s Number of States.			

Reliability

The MSO object is always reliable, unless it has an associated feedback object. If it has a feedback object assigned, the MSO object will become unreliable when the associated feedback object becomes unreliable. The following MSO attributes also become unreliable: ALARM, DISPLAY, NORMAL, and STATUS.

The MSO object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to an MSO object block sends unreliable data to the object, the WRIT block will execute, but the MSO object will ignore the write and remain in its current state.

Example

In this example (Figure 25), a 4-state MSO object that represents a multistate fan control (AHU1\S_FAN) receives commands according to damper position. The damper value is spanned in the SPAN block to a range, which can be processed by the MODE input of the MSEL block. Inputs 1 through 4 of the MSEL block select appropriate states, which are then sent as commands via the CMD block to the MSO object (AHU1\S_FAN).

A period of 00:02:00 is defined for this process, which means that it will run once every 2 minutes.

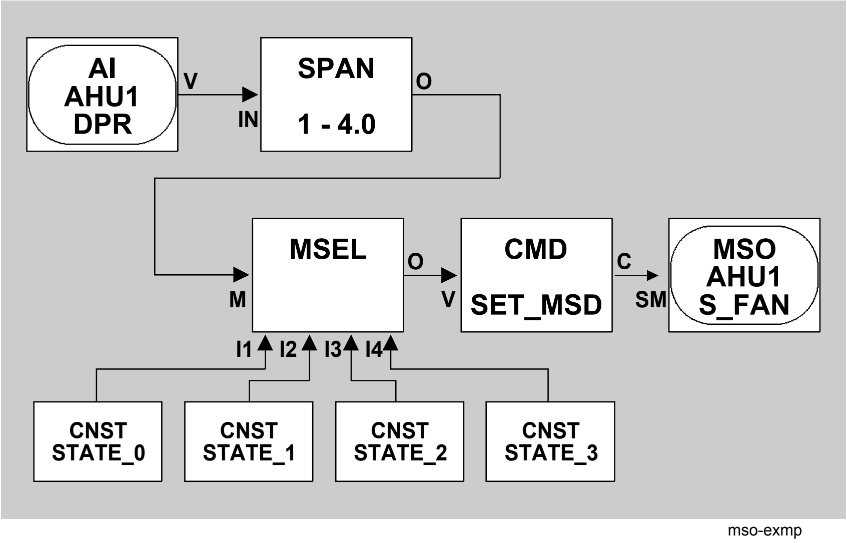


Figure 25: MSO Object Example

Notes: The MSO block can reference an MSO software object mapped to a LON device. However, the MSO software object must already be present in the archive database. You cannot create it by setting an MSO block in the GPL editor.

The MSO block will not show what hardware reference the MSO actually uses in the device. When working with software objects on LON devices, create your archive database with DDL first, then design your control strategy within the GPL editor.

PIDL (PID Loop) Object

Category

Controllers

Purpose

Creates a software representation of a PID Loop running in the DCM. Its main function is to provide a proportional plus integral plus derivative control algorithm based on sampling a feedback value at consistent time intervals.

Details

The PID algorithm incorporates the proportional, integral, and derivative terms independently. This means that the PID Loop can be configured as a proportional only, a proportional plus integral, or a proportional plus integral plus derivative controller.

Figure 26 shows a general model of how the PIDL object works. For more information, see the *Metasys Network Technical Manual, Software Data Sheets, PID Loop (PIDL) Technical Bulletin*.

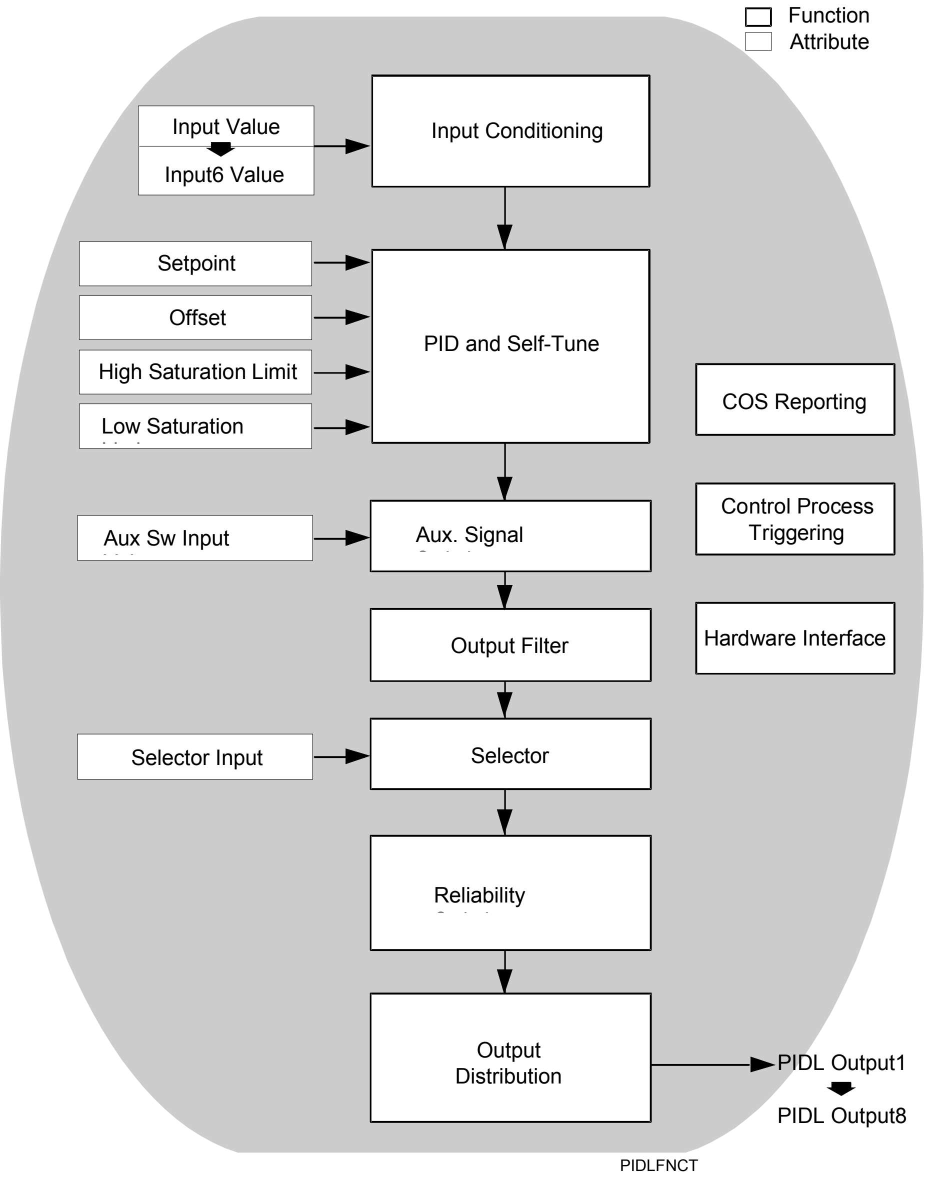


Figure 26: PIDL Functional Flow

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	PID Loop Number	INT/N	1	DCM: 1 through 16 DCM140: 1 through 20
Engineering Data	Analog Units	STR	DEG F	6 characters
	Decimal Position	INT	1	0 to 3

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Initial Control Parameters	Input Function	TAB	SUM	SUM, MIN or MAX
	Sample Period	INT	5	1 to 32, 767 seconds
	Proportional Bnd	ANA	20.00000	Real
	Integral Time*	ANA	60.00000	Real ≥ 0.0 seconds
	Derivative Wgt	ANA	0	Real ≥ 0.0
	Deadband*	ANA	1.000000	Real ≥ 0.0
	Hysteresis Comp	ANA	0.000000	0.0 to 100.0%
	Tune Noise Band	ANA	4.000000	Real ≥ 0.0
	Tune Chng Factor	ANA	0.000000	Real ≥ 0.0
	Aux Switch Ena	BIN	N	Y (Yes) or N (No)
	Filter Weight	ANA	1.000000	Real ≥ 1.0
	Selector Type=HI	BIN	Y	Y (Yes) or N (No)
	Select Unrl Dflt	BIN	N	Y (Yes) or N (No)

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
Initial Control Parameters	Unrel Dflt Resp	ANA	0.000000	Real
Report Type	Normal	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Template Fields (Third Screen: Port Definition)

Category	Field Name	Type	Default	Range/Choices
Setpoint	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	55.00000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters
Offset	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters
Aux Switch Input	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters
High Saturation	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	100.0000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters

Continued on next page . . .

Category (Cont.)	**Field Name**	**Type**	**Default**	**Range/Choices**
Low Saturation	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters
Selector Input	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters

Template Fields (Fourth Screen: Port Definition)

Category	Field Name	Type	Default	Range/Choices
Input 1	Scalar	ANA	1.000000	Real
	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters
Input 2 to Input 6	Scalar	ANA	0.000000	Real
	Reference Select	TAB/N	NORMAL	NORMAL or REFERENCE
	Value*	ANA/ POP-UP	0.000000	Real
	System Name	STR/N/ POP-UP	Blank	8 characters
	Object Name	STR/N/ POP-UP	Blank	8 characters

Template Fields (Fifth Screen: Output Definition)

Category	Field Name	Type	Default	Range/Choices
Output 1 to Output 8	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Attribute Name	TAB/N	Blank	Blank,VALUE,INP1VAL,
				INP2VAL,INP3VAL,INP4VAL,
				INP5VAL,INP6VAL,
				SETPOINT,OFFSET,
				HI_SAT_V,LO_SAT_V,
				AUX_IN,SEL_INP

Connections

Connection	Name	Type	Label
Input (GPL)	AUX_IN	ANA	AI
	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	HI_SAT_V	ANA	HS
	INPUT 1*	ANA	I1
	INPUT 2*	ANA	I2
	INPUT 3*	ANA	I3
	INPUT 4*	ANA	I4
	INPUT 5*	ANA	I5
	INPUT 6*	ANA	I6
	LO_SAT_V	ANA	LS
	OFFSET	ANA	OS
	SEL_INP	ANA	SL
	SETPOINT	ANA	SP
	WRITE	WRIT	WR
Input (Commands)	AUX_DIS	CMD/2CMD	AD
	AUX_ENA	CMD/2CMD	AE
	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_PIDL	CMD	RP
	RES_TOT	CMD	RT
	SET_PIDL	CMD	SP
	STARTUP	CMD	SU
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	FEEDBACK	ANA	FB
	READ	READ	RD

Continued on next page . . .

Connection (Cont.)	Name	Type	Label
Output (Attributes)	AUX_IN*	ANA	AI
	DEADBAND*	ANA	DB
	HI_SAT_V*	ANA	HS
	HI_SAT_F	BIN	HF
	INT_TIME*	ANA	IT
	LO_SAT_V*	ANA	LS
	LO_SAT_F	BIN	LF
	OFFLINE	BIN	OL
	OFFSET*	ANA	OS
	OVERRIDE	BIN	SO
	PIDA_REL	BIN	PR
	PIDL_OUT	ANA	PO
	PROPBAND	ANA	PB
	RELIABLE	BIN	R
	SEL_FLAG	BIN	SF
	SETPOINT*	ANA	SP
	SEL_INP*	ANA	SL
	VALUE	ANA	V

Note: To send the result of the PID Loop calculation to an AOD or another PIDL block, use the PIDL OUT output connection, not VALUE.

Reliability

The PIDL object becomes unreliable if it goes offline. The following attributes also become unreliable if the PIDL object goes offline or if an internal PIDL attribute becomes unreliable:

> AUX_IN, DISPLAY, FEEDBACK, FLTR_VAL, HI_SAT_V, INP1VAL, INP2VAL, INP3VAL, INP4VAL, INP5VAL, INP6VAL, LO_SAT_V, OFFSET, PID_CALC, SEL_INP, SEL_OUT, SETPOINT, SWCH_OUT, and VALUE.

In addition, these attributes become unreliable if just the PIDL object goes offline:

> HI_SAT_F, INT_TIME, LO_SAT_F, PIDA_REL, PROPBAND, and SEL_FLAG, and UNREL.

The PIDL object ignores all commands that contain an unreliable parameter. However, the PIDL object accepts a SET_PIDL command whose value parameter is unreliable. In this case, the command may cause the PIDL object to be unreliable.

If a WRIT block that is connected to a PIDL object block sends unreliable data to the object, the WRIT block will execute, but the PIDL object will ignore the write and remain in its current state.

REF (Generic Object Reference) Block

Category

Input/Output

Purpose

Allows GPL processes to reference a software object/model or hardware object that exists in the archive database, but is of a type not supported by a dedicated GPL block. An example is a Control System (CS) object.

Details

The Generic Object Reference (REF) block provides a GPL reference to an object that exists in the NCM archive database. By this reference, you may issue commands to the object from a process or read attributes of the object for use in a process. The REF block is generic in that it can reference a multitude of object types, including those listed in the following table. Use the Ref Object Type field in the template to select the type of object.

Note: The REF block allows you to read OFFLINE attributes, but it does not trigger the process.

Object (Ref. Object Type)	Type of Object
CS (Control System)	Software Model
C260X	Software
C500X	Software
DSC-1000	Hardware
XM (Includes XBN, XRL, XRE, XRM)	Hardware
DCM	Hardware
LCD	Hardware
AHU	Hardware
VAV	Hardware
UNT	Hardware
LCP	Hardware
DC9100	Hardware
DR9100	Hardware
FIRE	Hardware
FPU	Hardware
DSC8500	Hardware
D600	Hardware
READER	Software
MIG	Hardware
DX9100	Hardware
XT9100	Hardware
DCM140	Hardware
XTM	Hardware
VND	Hardware
PHX	Hardware
DX91ECH	Hardware
TC9100	Hardware
MC	Software
NDM	Hardware
VMA	Hardware
LONTCU, LONTCUA	Hardware
LONVMA, LONVMAA	
Other (Includes any object not covered by given categories.)	Software or Hardware

The REF block has the following characteristics:

- The object that the REF block references must already exist in the archive database. Define the object in the DDL file or define it online with the Operator Workstation and upload the file. Unlike other dedicated object block types (e.g., AI), the object that the REF block references is not added to, modified, or deleted from the archive database when updated with the Editor.

- The Type Name field in the template labels the REF block. This name will appear on the outside of the block. You may use the default name, REF, or any eight character maximum name. We recommend that you match this name with the entry in the Ref Object Type field. However, if the type is CS, match the Type Name with the model name; for example, AHU12.

- The second page of the REF block template lists the attributes that will be direct output connections. You do not need to specify any output connections in the template if you intend to draw only input connections to the REF block (e.g., a CMD block into a REF block). Also, when commanding an attribute of a REF object, do not enter the attribute name in the REF block template, just enter it in the CMD or 2CMD block template. The REF template shows eight binary, eight analog, and two time output connections. Each connection has two fields: Attribute and Description.

  Attribute: Enter the exact name of the attribute for the software object/model or hardware object (e.g., AI_2). This name must match the corresponding attribute as defined in the DDL file or at the Operator Workstation. The attribute name becomes a connection name on the output connection menu. Enter each attribute name carefully; they are not verified until the process is compiled. *Appendix G: Attributes* lists the attributes available for each object type.

  Note: To determine which attribute name corresponds to which symbol name, refer to the symbol table from HVAC PRO or the CS Model summary at the Operator Workstation.

Description: Enter a description of the attribute (e.g., Dis_Temp). If the attribute is configured in a software object/model or hardware object, we recommend that you match the attribute's description in the REF block with its description in the model. In that way, you will be able to easily identify the attribute in GPL and at the Operator Workstation. The Editor does not check for matching attribute descriptions.

- The REF object can access all the readable attributes of a referenced object/model. Use direct connections to access those attributes that are configured in the template. Use the READ block to access integer attributes.
- You can use the WRIT block to change the value of an attribute configured in the REF block. Refer to the WRIT block for details.
- If more direct connections are needed in the process than are supplied from one REF block, you can access the other attributes by connecting a READ block to the REF block. Or, you may define multiple REF blocks that reference the same system\object in the same strategy file. In this case, the file would have two or more REF blocks with identical system\object names. To differentiate between the blocks, use a block label. The Block Label is a one character identifier that is part of the block name. The label can be any valid alphanumeric character or symbol. It is always in the upper right corner of the block, colored blue. If it is blank, no label will appear on the outside of the block.
- An attribute of the REF block may be used as an associated input to an AD or BD object. If you use this application, make sure the attribute name you enter in the REF block template matches exactly the attribute name in the AD or BD template (Associated Input column). The Expert Checker verifies valid attribute connections between REF and AD/BD blocks.
- The REF block does not allow direct connections for multistate integer attributes of objects. To read a multistate integer attribute, connect the REF block to a READ block, and enter the multistate attribute name in the READ block template. The READ block will convert the integer to analog.

IMPORTANT:	The GPL Editor does not check the use of multistate attributes. Therefore, if you read a multistate attribute without going through a READ block, calculations may provide unexpected results, especially in the case of division.

- If you specify Other in the Ref Object Type field, no type check is made. The Editor assumes you have defined everything properly in DDL and at the OWS.
- If you exempt a binary attribute connection, the Editor and Expert Checker assume it is a triggerable attribute. The Compiler will detect incorrect exemptions.

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Description	STR	Blank	24 characters
	Block Label	STR/N	Blank	1 character
	Type Name	STR	REF	8 characters
	Ref. Object Type	TAB/N	CS	CS
				C260X
				C500X
				DSC-1000
				XM
				DCM
				LCD
				AHU
				VAV
				UNT
				LCP
				DC9100
				DR9100
				FIRE
				FPU
				DSC8500
				D600
				READER
				MIG
				DX9100
				XT9100
				DCM140
				XTM
				VND
				PHX

Continued on next page . . .

Category (Cont.)	Field Name	Type	Default	Range/Choices
				DX91ECH
				TC9100
				MC
				NDM
				VMA
				LONTCU, LONTCUA
				LONVMA, LONVMAA
				OTHER

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Binary Connections	Attribute (B1-B8)	STR	Blank	8 characters
	Description (B1-B8)	STR	Blank	8 characters
Analog Connections	Attribute (A1-A8)	STR	Blank	8 characters
	Description (A1-A8)	STR	Blank	8 characters
Time Connections	Attribute (T1-T2)	STR	Blank	8 characters
	Description (T1-T2)	STR	Blank	8 characters

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	BEG_TRND	CMD/2CMD	BH
	END_TOT	CMD/2CMD	ET
	END_TRND	CMD/2CMD	EH
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_260X	CMD	RL
	REL_500X	CMD	RL
	REL_CS	CMD	RL
	RES_TOT	CMD	RT
	SET_MC	CMD	SM
	ST260XAN	CMD	SA
	ST260XBN	CMD	SB
	ST260XSP	CMD	SA
	ST500XAN	CMD	SA
	ST500XBN	CMD	SB
	ST500XSP	CMD	SA
	STCSAN	CMD	SA
	STCSBN	CMD	SB
	STCSMS	CMD	SA
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD
Output (Attributes)	{A1 to A8 attribute}	ANA	A1 to A8
	{B1 to B8 attribute}	BIN	B1 to B8
	{T1 or T2 attribute}	TIM	T1 or T2

Reliability

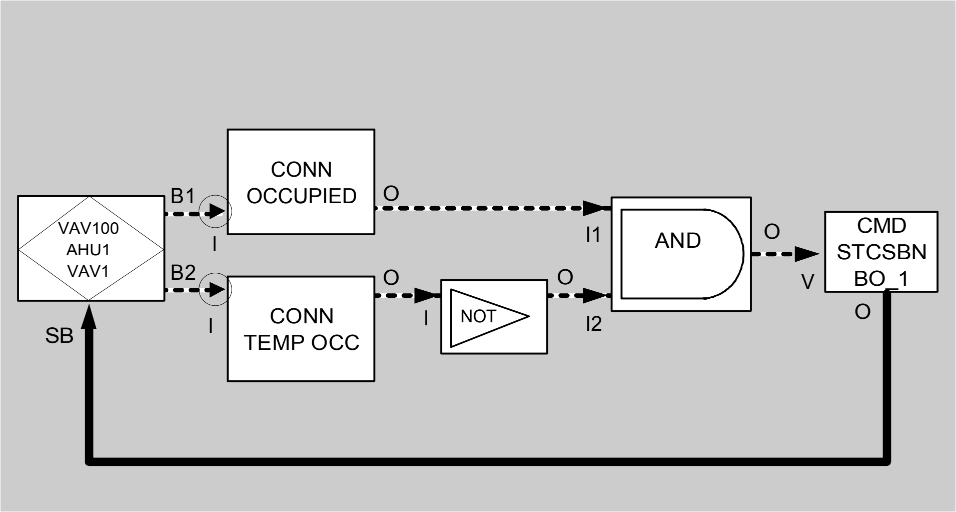
The REF block, since it is a reference to an object, matches the reliability of the object. Specifically, the particular output connection of the REF block is unreliable if the attribute referenced is unreliable.

Example

In the example shown in Figure 28, a REF block is used to reference a VAV100 Controller. The occupied mode (AHU1\VAV1\OCCUPIED) and temporary occupancy mode (AHU1\VAV1\TEMP_OCC) attributes from the controller are read. When the controller is in the occupied mode, but not in the temporary occupancy mode, the CMD block sends a STCSBN command to the controller to turn on the toilet exhaust fans. When the controller is in the unoccupied mode or the temporary occupancy mode, the fans are turned off.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. These include: AND, NOT, and CMD.



REFBLOK

Figure 28: REF Block Example

210A (C210A) Block

Category

Controllers

Purpose

Creates a software representation for a C210A Application Specific Controller.

Details

You must define one 210A object block for each C210A controller that you wish to connect to the Metasys Network. Within the database template, you define the specific applications of the controller. The application selections you make must match the field configuration exactly (i.e., how the C210A is configured with the Y199 Service Module). The template does not configure the controller, it merely matches the field configuration.

For more information on configuring the C210A, see the *Metasys Application Specific Controllers Technical Manual, C210A/C260A Controllers, C210A Controller Technical Bulletin*.

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
Application	Damper	TAB	NO DAMPER	NO DAMPER, PRESSURE DEP, or PRESSURE INP
	Heat	TAB	NO HEAT	NO HEAT or HEAT
	Fan	TAB	NONE	NONE, SERIES, or PARALLEL
	Setpoint	TAB	LOCAL	LOCAL or REMOTE
	Occupied/Unocc	TAB	NONE	NONE, LOCAL CONTACT, L2 COMMAND, or BOTH
	Warmup	TAB	NONE	NONE, L2 COMMAND, or SARE AND L2 COMMAND
	Shutdown	TAB	NONE	NONE, L2 COMMAND OPEN, LOCAL, or L2 CMD CLOSE and LOCAL
	Auxiliary Binary	BIN	N	Y (Yes) or N (No)
	Aux. Temp	BIN	N	Y (Yes) or N (No)
	Aux. Delta P	BIN	N	Y (Yes) or N (No)
	Aux. Humid	BIN	N	Y (Yes) or N (No)
	Aux. Sensor	BIN	N	Y (Yes) or N (No)

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Parameters	Display Value AI	ANA	1	1 to 6
	Display Units	STR	DEG F	6 characters
	Setpt for NT SP	ANA	7	1 to 12
Report Type	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Graphic Symbol No.	ANA	0	0 to 32767
	Operating Instr No.	ANA	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	END_TOT	CMD/2CMD	ET
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_210A	CMD	RL
	RES_TOT	CMD	RT
	ST210AAN	CMD	AN
	ST210ABN	CMD	BN
	ST210ASP	CMD	SP
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD
Continued on next page . . .			

Connection (Cont.)	Name	Description	Type	Label
Output (Attributes)	AXBI	Auxiliary Binary Input	BIN	AX
	CNWU	Central System Warmup	BIN	CW
	DFPR	Differential Press-Control	ANA	DR
	DPSP	Calculated DP Setpoint	ANA	DP
	HCPB	Htg/Clg Proportional Band	ANA	HB
	HLTC	Hardware Latch Point	BIN	HC
	HWSD	Hardware Shutdown	BIN	HD
	HWUO	Hardware Unoccupied	BIN	HO
	LTCH	Latched Internal Point	BIN	LT
	MNDP	Minimum DP Setpoint	ANA	MD
	MXDP	Maximum DP Setpoint	ANA	MP
	OFFLINE	Object Offline Flag	BIN	OL
	OVERRIDE	Software Override Flag	BIN	SO
	PRCL	Prop Cooling and Press DP Act	ANA	PL
	RZSP	Remote Zone Setpoint	ANA	RZ
	SARE	Supply Air Reheat Enable	BIN	SR
	SDBC	Shutdown w/Box Closed	BIN	SC
	SDBO	Shutdown w/Box Open	BIN	OS
	UNOC	Downloaded Unoccupied	BIN	UO
	WMUP	Warmup Enabled	BIN	WP
	ZNSP	Zone Setpoint	ANA	ZS
	ZNT	Zone Temperature	ANA	ZT

Reliability

The 210A object becomes unreliable when the C210A controller reports an unreliable status, or a device connected to the C210A goes offline. The following is a list of attributes that also become unreliable:

Name	Description	Name	Description
ACTC	Actuator Timing Constant	**OFRH**	On-Off Reheat
AUXI	Auxiliary 0-5 VDC Input	**PFOF**	Parallel Fan On/Off
AUXP	Auxiliary Diff Pressure	**PFSS**	Parallel Fan Start/Stop
AUXR	Auxiliary Relative Humidity	**PMAX**	Actual DP at Maximum
AUXT	Auxiliary Temp Sensor	**PRCL**	Prop Cooling and Press DP Act
AXBI	Auxiliary Binary Input	**PRRH**	Proportional Reheat
AXDP	Auxiliary DP Setpoint	**REHT**	Electric Reheat
CNWU	Central System Warmup	**RZSP**	Remote Zone Setpoint
DFPR	Delta Pressure-Control	**SARE**	Supply Air Reheat Enable
DISPLAY	ASCII Value	**SDBC**	Shutdown w/Box Closed
DMPP	Damper Position	**SDBO**	Shutdown w/Box Open
DPSP	Calculated DP Setpoint	**SETPOINT**	NT-chosen Setpoint
HCPB	Htg/Clg Proportional Band	**SFOF**	Series Fan On/Off
HLTC	Hardware Latch Point	**SFSU**	Series Fan Startup
HRTZ	50 Hz/60 Hz	**SUSB**	Setup/Setback
HWSD	Hardware Shutdown	**TCMD**	Temp Control Mode
HWUO	Hardware Unoccupied	**TCMD ASCI**	Temp Control Mode ASCII
HTDB	Heating Deadband	**UNOC**	Downloaded Unoccupied
INTE	Integer Error	**VALUE**	Value
INTG	Integral Gain	**WMUP**	Warmup Enabled
LTCH	Latched Internal Point	**WTMP**	Warmup Temp Diff
LV12	L2 Bus Only Switch	**ZNSP**	Zone Setpoint
MNDP	Minimum DP Setpoint	**ZNT**	Zone Temperature
MXDP	Maximum DP Setpoint		

The 210A object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to a 210A object block sends unreliable data to the object, the WRIT block will execute, but the 210A object will ignore the write and remain in its current state.

Example

In the example shown in Figure 29, the 210A block represents a VAV controller on the third floor (VAV1\3RDFLR), which in this process is used for temporary occupancy control. The controller monitors the state of a binary push button, which is located on a thermostat. An attribute called Latch (LT connection) represents the pushbutton. When the pushbutton is pressed, and the controller is in unoccupied mode (UO connection), two things happen: (1) the CMD block sends a command to set the C210A controller in the occupied mode, and (2) the DLAY block sets a delay time of two hours (02:00:00). The timer sets temporary occupancy control for two hours. After two hours, a command is sent to reset the C210A to unoccupied mode, and reset the Latch attribute to False. The latter action allows an occupant to press the pushbutton again.

A period of 00:10:00 is defined for this process, which means it will run once every 10 minutes.

Note: All operation blocks in this example must be placed in a process. They include: AND, both CMD blocks, and DLAY.

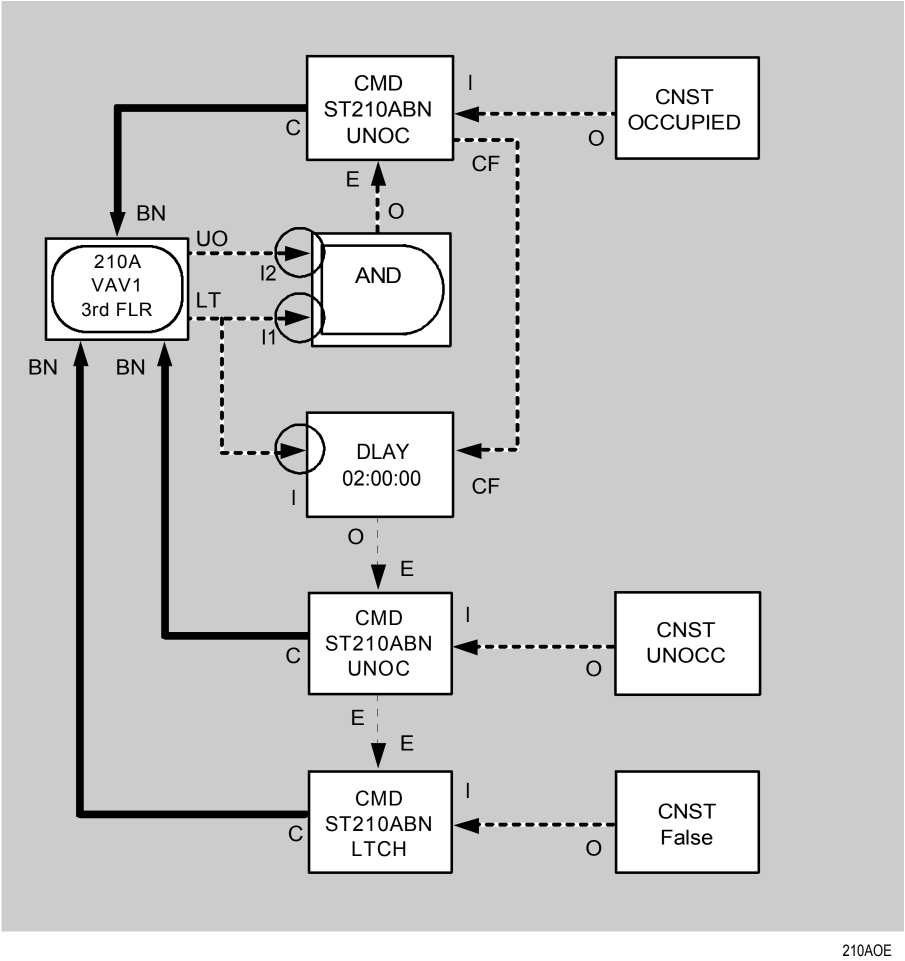


Figure 29: 210A Object Example

260A (C260A) Block

Category

Controllers

Purpose

Creates a software representation for a C260A Heat Pump controller.

Details

You must define one 260A object block for each C260A controller that you wish to connect to the Metasys Network. Within the database template, you define the specific applications of the controller. The application selections you make must match the field configuration exactly (i.e., how the C260A is configured with the Y199 Service Module). The template *does not* configure the controller, it merely matches the field configuration.

For more information on configuring the C260A, see the *Metasys Application Specific Controllers Technical Manual, C210A/C260A Controllers, C260A Controller Technical Bulletin*.

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	24 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
Application	Fan On/Comp Off	TAB	NONE	NONE, LOCAL CONTACT, L2 COMMAND, or BOTH
	Setpoint	TAB	LOCAL	LOCAL or REMOTE
	Occupied/Unocc	TAB	NONE	NONE, LOCAL CONTACT, L2 COMMAND, or BOTH
	Shutdown	TAB	NONE	NONE or L2 COMMAND
	Auxiliary Binary	BIN	N	Y (Yes) or N (No)
	Aux. Temp 1	BIN	N	Y (Yes) or N (No)
	Aux. Temp 2	BIN	N	Y (Yes) or N (No)
	Aux. Humid 1	BIN	N	Y (Yes) or N (No)
	Aux. Humid 2	BIN	N	Y (Yes) or N (No)
	Aux. Sensor	BIN	N	Y (Yes) or N (No)

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	N	Y (Yes) or N (No)
	Comm Disabled	BIN	N	Y (Yes) or N (No)
Parameters	Display Value AI	ANA	1	1 to 6
	Display Units	STR	DEG F	6 characters
	Setpt for NT SP	ANA	6	1 to 8
Report Type	Override	TAB	NONE	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Graphic Symbol No.	ANA	0	0 to 32767
	Operating Instr No.	ANA	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	BEG_TOT	CMD/2CMD	BT
	END_TOT	CMD/2CMD	ET
	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	REL_260A	CMD	RL
	RES_TOT	CMD	RT
	ST260AAN	CMD	AN
	ST260ABN	CMD	BN
	ST260ASP	CMD	SP
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD

Connection	Name	Description	Type	Label
Output (Attributes)	FLOW	Flow	BIN	FL
	HWUO	Hardware Unoccupied	BIN	HO
	OFFLINE	Object Offline Flag	BIN	OL
	OVERRIDE	Software Override Flag	BIN	SO
	RZSP	Remote Zone Setpoint	ANA	RZ
	SDWN	Shutdown	BIN	SN
	UNOC	Downloaded Unoccupied	BIN	UC
	ZNSP	Zone Setpoint	ANA	ZS
	ZNT	Zone Temperature	ANA	ZT

Reliability

The 260A object becomes unreliable when the C260A controller reports an unreliable status, or a device connected to the C260A goes offline. The following is a list of attributes that also become unreliable:

Name	Description	Name	Description
AUXT	Auxiliary Temperature 1	**LTCH**	Latched Internal Point
AUXR	Auxiliary Humidity 1	**LV12**	L2 Bus Only
AUXI	Auxiliary 0-5 VDC	**PE**	Proportional Error
AURH	Auxiliary Humidity 2	**PRCL**	Proportional Cooling
ATMP	Auxiliary Temp 2	**PRRH**	Proportional Reheat
CLPB	Cooling Prop Band	**RVAL**	Reverse Valve Configuration
CPCM	Compressor Cooling Cmd	**RZSP**	Remote Setpoint
DISPLAY	ASCII Value	**SETPOINT**	NT-chosen Setpoint
FLOW	Flow	**SDWN**	Downloaded Shutdown
FNON	Hardware Fan On/Comp Off	**STBK**	Setback
FOCO	Downloaded Fan On/Comp Off	**STUP**	Setup
HLTC	Hardware Latch Point	**TCMD**	Temp Control Mode
HRTZ	50 or 60 Hertz	**TCMD ASCI**	Temp Control Mode ASCII
HTCM	Heating Command	**UNOC**	Downloaded Occ/Unocc
HTDB	Heating Deadband	**VALUE**	Value
HTPB	Heating Proportional Band	**ZNSP**	Zone Setpoint
HWUO	Hardware Occupied/ Unoccupied	**ZNT**	Zone Temperature
INTG	Integration		

The 260A object ignores all commands that contain an unreliable parameter. Also, if a WRIT block that is connected to a 260A object block sends unreliable data to the object, the WRIT block will execute, but the 260A object will ignore the write and remain in its current state.

Example

In the example shown in Figure 30, the 260A block represents a heat pump controller that controls a fan and compressor that serves a conference room (AHU1\CONF. RM). In this process, the C260A is used for floating temperature control based on zone temperature. When occupancy mode is set (i.e., unoccupied mode attribute is not set), the zone temperature is examined. If zone temperature is less than 72.0°F, or greater than or equal to 75.0°F (with an applied deadband of 3.0°F) the CMD block commands the attribute Fan On/Compressor Off to Off, which allows for fan and compressor operation. If zone temperature is less than 72.0°F or greater than 75.0°F, the CMD block commands the Fan On/Compressor Off attribute to On, which allows for fan only operation.

A period of 00:10:00 is defined for this process, which means it will run once every ten minutes.

Note: All operation blocks in this example must be placed in a process. They include: AND, NOT, both CMD blocks, and DBCM.

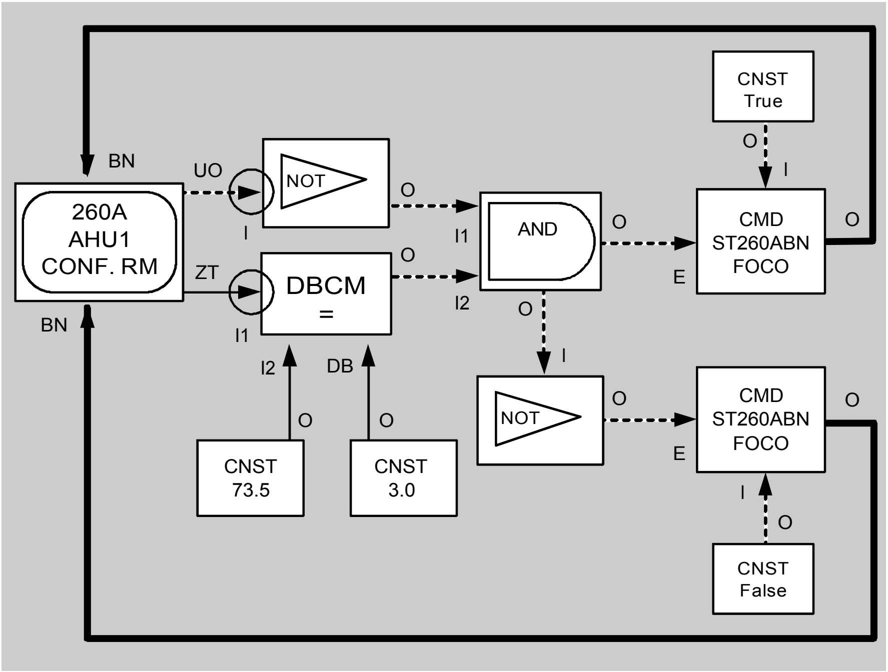


Figure 30: 260A Object Example

Zone (Fire Zone) Block

Category

Controllers

Purpose

Creates a software object representation of a single fire zone that is programmed into the IFC-2020 Fire Controller. For details, refer to the *Fire System Objects Technical Bulletin* in the *Metasys Network Technical Manual* under the *Objects* tab.

Details

Once defined and downloaded, the ZONE object block represents a ZONE object that resides in the NCM. The ZONE object tracks the status of the corresponding fire zone programmed into the IFC-2020, and reports that status as required throughout the Metasys Network. You can define up to 240 ZONE blocks per IFC-2020.

Note: When the ZONE object is downloaded to the NCM with the Operator Workstation, the data does not also download to the IFC-2020. To work around this situation, program the IFC-2020 manually or program the IFC-2020 online from the workstation.

The Fire Zone No. parameter assigns this block to a unique zone. The zone maps to a specific slot on the fire controller. Each zone must have a unique fire zone number.

The Panel No. and Point No. parameters both require a zero or non-zero number. An interfield error message displays if one is zero and the other is not.

As part of configuring the ZONE block, you may need to write an interlock statement in the block’s template. This is a control-by-event interlock statement that tells the fire system to take a specific action when a certain point reports an alarm. For example, if an initiating device goes into alarm, then horns will be sounded. An interlock statement is not required if the ZONE object is for a forward activated zone. If the object is for a reverse activated zone, you must enter at least a null set, such as OR ().

The interlock statement determines if this is a forward or reverse activated object. If the interlock statement is a list, a forward activated zone will be generated. If the interlock statement is an equation, a reverse activated zone will be generated. If the interlock statement is defaulted, a forward activated null set will be generated. For an explanation of forward activated and reverse activated lists, see the *IFC-1010/2020 Technical Manual (FAN 448).*

The interlock statement format almost directly follows the IFC-2020 standalone programming format for the control-by-event equation. All restrictions placed on the control-by-event equations apply. In addition, follow these special Metasys system requirements, which do not precisely match IFC-1010/2020 restrictions:

- Interlock statement must be an ASCII string of up to 70 characters.
- Items in the statement can be separated by spaces, but spaces are not required. (Items are defined as embedded operators, times, dates, zones, or devices.) Spaces after a left parenthesis or before a right parenthesis are optional.
- Statements containing a reverse activated equation must begin with an explicit operator and end with a right parenthesis.
- Each operator must include a left parenthesis immediately after the operator text. No spaces or implied parentheses are allowed. For example, using the implied parenthesis "(" will not be accepted to mean “OR(”, as it does on the IFC-1010/2020.
- Statements containing a forward activated list can begin and end with parentheses, but parentheses are not required. For example: (Z200 Z210 Z6M35). No embedded (additional) parentheses are allowed.
- Lower case characters will be treated the same as upper case characters.

- Only those operators known to the Metasys system can be used, which are: AND(, DEL(, NOT(, OR(, TIM(, and XZONE(.
- Time operators must have a colon between the hour, minute, and second values (e.g., 01:17:20) instead of a period.

When you press F10 to save the changes, the Editor checks the template entries and syntax of the interlock statement you entered. (However, if the Editor is in No Archive mode, no checks are made.) A caret (^) symbol below the statement indicates the approximate location of the error. The possible error messages with descriptions are as follows:

```
Address, time, or date value is out of
range
```

The value of the address, time, or date parameter is not within the valid range.

```
Cannot put detector on forward activated
list
```

A forward activated list cannot contain points that are detectors.

```
Day of week prefix incorrect or repeated
in TIM function
```

The day-of-week abbreviation entered is incorrect or repeated within the interlock statement. Valid abbreviations are SU, MO, TU, WE, TR, FR, and SA for Sunday through Saturday, respectively.

```
HR:MIN:SEC requires colon (:)
```

A colon (:) is required to separate the hour, minute, and second values.

```
Invalid operand
```

A legal parameter such as Z1 or L2D5 was expected but not found at the point of the error.

`Missing operator inside DEL (Delay) function`

The Delay function requires an operator.

`MM-DD-YY requires hyphen (-)`

A hyphen (-) is required to separate the month, day, and year values.

`No loop devices allowed as parameters for the XZONE function`

The XZONE operator only allows zones as parameters.

`NOT function operates on only one value`

Only one value can be specified with the NOT function. The value can come from a single parameter (e.g., Z5 or L1D3) or an equation that results in a single value (e.g., (OR(Z28 Z29) or OR(Z1 AND(Z3 Z4)). An example of an invalid function is: NOT(Z1 Z2), which has two parameters.

`Parentheses mismatch. Left and right parentheses do not match`

A required left or right parenthesis is missing.

`Stop time must be greater than start time within TIM function`

The stop time must be a later time than the start time. Crossing over midnight is not allowed.

`Text found past closing parenthesis`

A parameter or operator is found past the point where the forward activated list or reverse activated equation should end.

`Too many operands or operators`

The interlock statement is too large to be stored properly.

```
Type designation incorrect for module or
detector loop device
```

The type designation for this parameter is incorrect.
A parameter identified as a loop device must be either a module or a detector.

```
Unknown operator detected in the reverse
equation
```

The reverse equation contains an unknown operator.

Template Fields (First Screen)

Category	Field Name	Type	Default	Range/Choices
Identification	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Expanded ID	STR	Blank	20 characters
Hardware	System Name	STR/N	Blank	8 characters
	Object Name	STR/N	Blank	8 characters
	Fire Zone No.	INT/N	1	1 to 240
Fire Interlock	Interlock Statement	STR/N	Blank	70 characters
Annunciator	Panel No.	INT/N	0	0 to 32
	Point No.	INT/N	0	0 to 64

Template Fields (Second Screen)

Category	Field Name	Type	Default	Range/Choices
Flags	Auto Dialout	BIN	Y	Y (Yes) or N (No)
	Comm. Disable	BIN	N	Y (Yes) or N (No)
Report Type	Normal	TAB	STATUS	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Alarm	TAB	CRITICAL1	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Disable	TAB	CRITICAL2	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
	Trouble	TAB	CRITICAL3	NONE
				CRITICAL1
				CRITICAL2
				CRITICAL3
				CRITICAL4
				FOLLOW_UP
				STATUS
Messages	Alarm No.	INT	0	0 to 255
	Normal No.	INT	0	0 to 255
	Disable No.	INT	0	0 to 255
	Graphic Symbol No.	INT	0	0 to 32767
	Operating Instr No.	INT	0	0 to 32767

Connections

Connection	Name	Type	Label
Input (GPL)	COMMAND	CMD	C
	DUAL CMD	2CMD	2C
	WRITE	WRIT	WR
Input (Commands)	LOC_REP	CMD/2CMD	LR
	LOC_TRG	CMD/2CMD	LT
	UNL_REP	CMD/2CMD	UR
	UNL_TRG	CMD/2CMD	UT
Output (GPL)	READ	READ	RD
Output (Attributes)	ALARM	BIN	A
	OFFLINE	BIN	OL
	TROUBLE	BIN	TR

Reliability

The ZONE object becomes unreliable when the hardware it represents goes offline or reports an unreliable status. The following ZONE attributes also become unreliable: ALARM, DISABLED LOCALLY, TROUBLE, and VALUE. In addition, if no alarm has been recorded for this zone since the NCM was last downloaded, the last alarm time stamp will be unreliable.

If a WRIT block that is connected to a ZONE object block sends unreliable data to the object, the WRIT block will execute, but the ZONE object will ignore the write and remain in its current state.

Example

In the following example (Figures 31a-31d), several ZONE blocks provide a smoke control zone application in which mechanical fans are used to limit the spread of harmful smoke between floors (i.e., zones) in a building. The floor where the fire is detected is purged while the floors above and below are pressurized. These actions help isolate the smoke from the other floors. For this example, refer to the following table. The actions are performed within this example.

Zone That Detects Fire	Action
1 **(1st Floor)**	Zone 1 is purged.
	Zone 2 is pressurized.
2 **(2nd Floor)**	Zone 1 is pressurized.
	Zone 2 is purged.
	Zone 3 is pressurized.
3 **(3rd Floor)**	Zone 2 is pressurized.
	Zone 3 is purged.

Figure 31a shows three group compounds. Under each is a group compound that contains the control logic that determines which floor to purge and which to pressurize. The group compound for Floor 1 is shown in Figure 31b.

If a fire is detected on Floor 1 (AHU_1\FLOOR_1), the CMD block sends a command to the BO object (AHU_1\PURGE_1) to purge the floor (and perform any other control sequence, such as opening dampers). Also, another command is sent to pressurize Floor 2.

The logic for Floor 2 is shown in Figure 31c. If a fire is detected on Floor 2 (AHU_1\FLOOR_2), the CMD block sends a command to the BO object (AHU_1\PURGE_2) to purge the floor. Also, two other commands are sent to pressurize Floors 1 and 3.

Finally, the logic for Floor 3 is shown in Figure 31d. This is almost identical to Floor 1. If a fire is detected on Floor 3 (AHU_1\FLOOR_3), the CMD block sends a command to the BO object (AHU_1\PURGE_3) to purge the floor. Also, another command is sent to pressurize Floor 2.

A period of 00:00:00 is defined for the process since it needs to run only when the Alarm attribute of the ZONE block is triggered, indicating a fire has been detected.

Note: All operation blocks in this example are placed in a process. They include: all CNST, OR, NOT, AND, and 2CMD blocks.

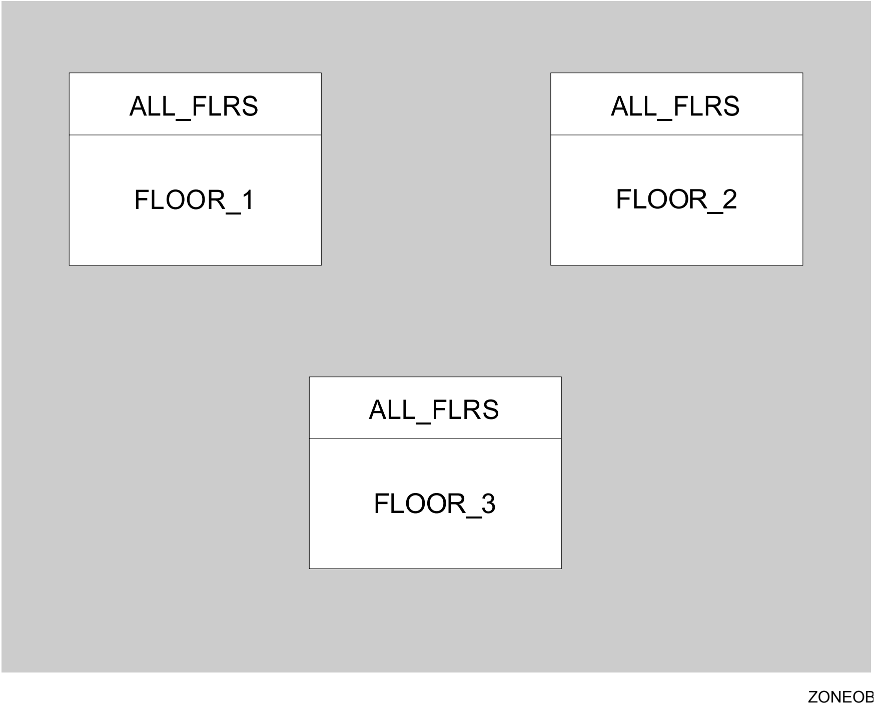


Figure 31a: ZONE Object Example

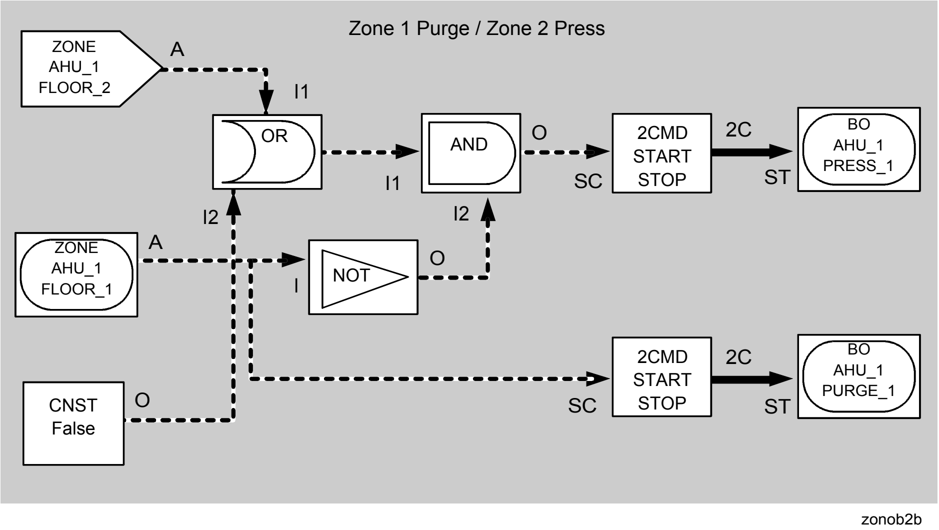


Figure 31b: ZONE Object Example

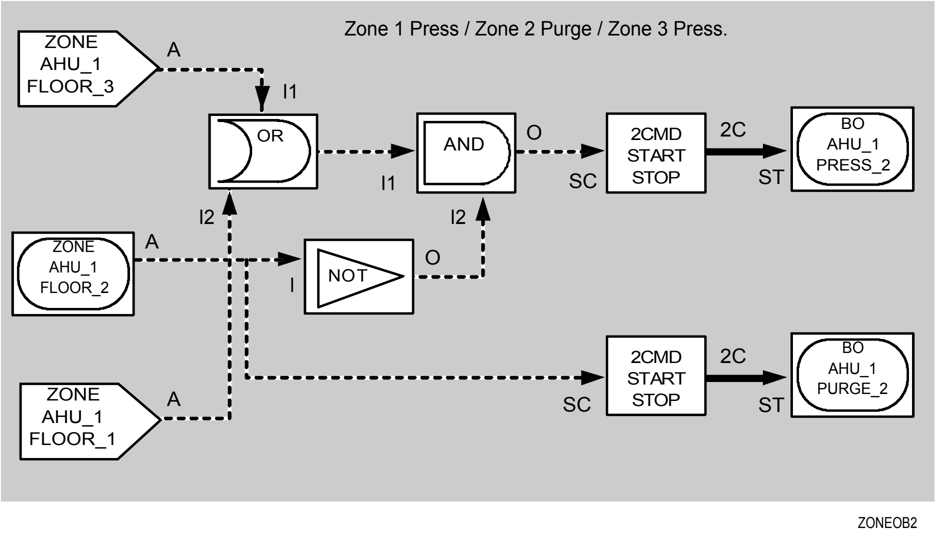
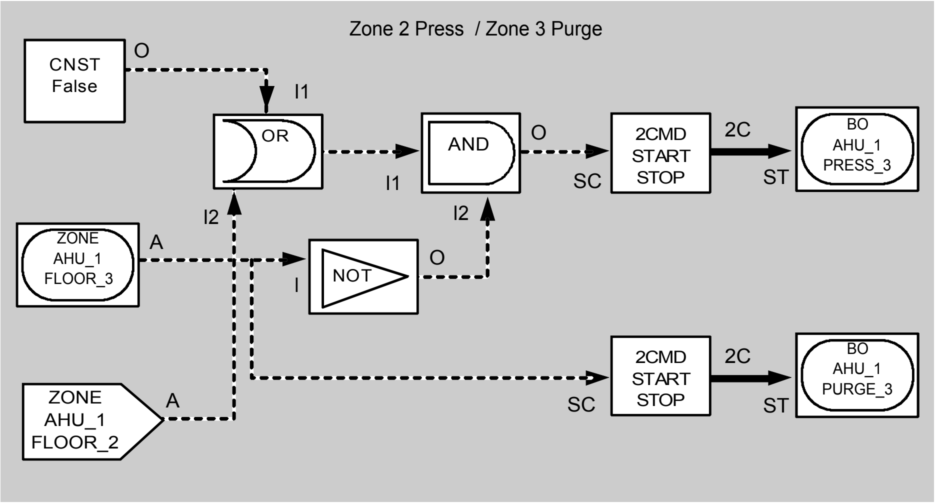


Figure 31c: ZONE Object Example



ZONEOB3

Figure 31d: ZONE Object Example

METASYS®

GPL Programmer's Manual

Operation and Special Blocks

* Indicates those sections where changes have occurred since the last printing.

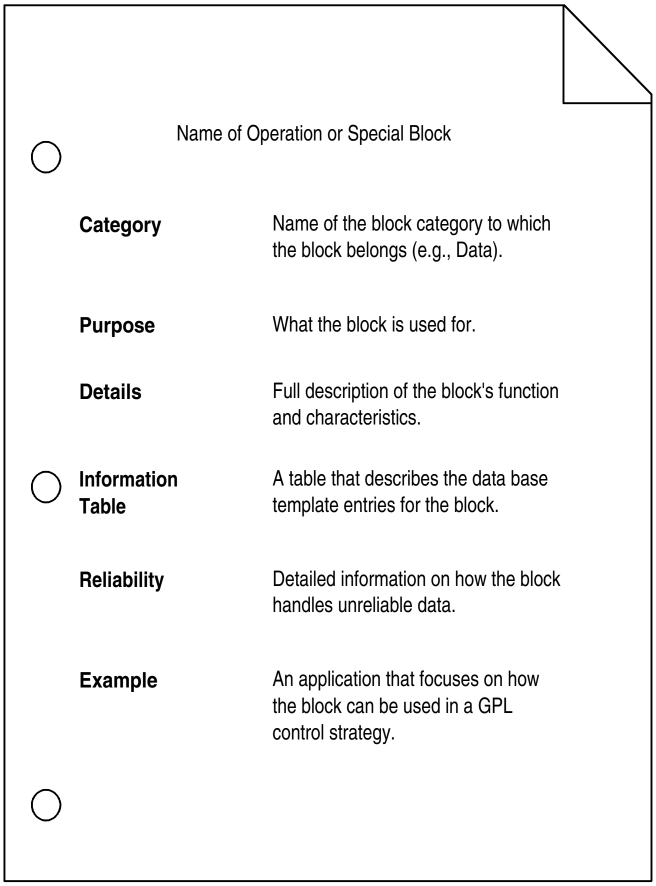
Code No. LIT-631090

* Indicates those sections where changes have occurred since the last printing.

Introduction

This section contains a detailed description of each operation and special function block available with the Graphic Programming Language. The blocks are organized alphabetically.

The description of each block in this section follows a standard format. Figure 32 shows this format.



Name of Operation or Special Block

Category Name of the block category to which the block belongs (e.g., Data).

Purpose What the block is used for.

Details Full description of the block's function and characteristics.

Information Table A table that describes the data base template entries for the block.

Reliability Detailed information on how the block handles unreliable data.

Example An application that focuses on how the block can be used in a GPL control strategy.

stdfmt2

Figure 32: Standard Format for Block Descriptions

Three of these headings require further explanation: category, information table, and example.

Category

All function blocks are organized in the GPL Editor by category. The operation and special blocks are under 13 different categories: Data, Control, Calculation, Psychometric Equation, Selector, Logic, Math, Report, Process Control, Object Control, Time, Reliability, and Miscellaneous.

Data Blocks

SVAR	CNST	CONN	VH	AD	BD	

These blocks are holders of data; data which can be used either in the same process or other processes. The SVAR and VH blocks are operation blocks, and the CNST and CONN blocks are special blocks.

Control Blocks

PIR	BSEQ	COMP	DFCM	DBCM		

These blocks are used for HVAC control and comparison applications.

Calculation Blocks

RAMP	SPAN	FILT				

These blocks perform calculations that can be used in processes.

Psychrometric Equation Blocks

ENRH	ENDP	WBRH	WBDP	DWPT	RH	

These blocks calculate enthalpy, wet bulb temperature, dew point, and relative humidity.

Selector Blocks

HSEL	LSEL	SWCH	SAMP	MSEL		

These blocks perform selection functions based on some criteria; for example, the HSEL block provides the highest value of two, three, or four inputs.

Logic Blocks

AND	OR	XOR	NOT	LTCH	PULS	DLAY

These blocks perform logical operations.

Math Blocks

ADD	SUB	MUL	DIV	AVG	EQN	

These blocks perform basic mathematical operations.

Report Blocks

PRNT	ADV					

These blocks send reports to operator devices.

Process Control Blocks

PERD	WAIT	STOP	ABRT			

These blocks control or affect the execution of processes.

Object Control Blocks

CMD	2CMD	READ	WRIT			

These blocks send commands to object blocks and allow an object block's attribute to be read or written to.

Time Blocks

TIME	RTOT	TTOT				

These blocks provide timing functions such as reading the system time and converting between real and time values. All Time blocks are operation blocks.

Reliability Blocks

FREL	UNRD					

The UNRD block tests for unreliable data, and the FREL block forces an output to be reliable. Both Reliability blocks are operation blocks.

Miscellaneous Blocks

TOT	USER	FILE				

These blocks provide various functions such as totalizing an object's value in minutes (TOT) and linking the files of multiple control strategies into one file (FILE). The TOT and USER blocks are operation blocks. The FILE block is a special block.

Information Table

The Information Table describes details about data base template entries and connection names.

Note: Some template entries may also be connection menu selections. For example, the Differential parameter of the DFCM block is in the block's template and in the block's input connection menu. You may enter a value in the template and connect the value to the block, *but the connected value will always take precedence over the template value.*

Those entries that are both in the template and in connection menus are indicated by an "X" marked in both the T column and the I (Input) or O (Output) column.

The following is a sample of the table with descriptions of its columns:

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB

Field Name

The name of the attribute or parameter as it appears in the data base template. An example is Block Name.

Note: For definitions of all attributes and parameters, refer to the section called *Template Field Descriptions.*

Connect Name

The name of the connection as it appears in the connection menus. Two examples are the INPUT and ENA OUT connections.

Type

The type of data that describes the template entry or connection. Included are:

ANA: Analog, real, or floating point fields or connections. Examples are the Setpoint field (PIR block) and the Deadband connection (DBCM block).

BIN: Binary (Boolean) fields or connections that take a Yes (Y) or No (N) entry or connection. Examples are the Default field (UNRD block) and the Input 1 connection (OR block).

CMD: Single command connections. An example is the command connection from a CMD block.

CTL: Control flow connections. An example is the control flow connection for a DBCM block.

2CMD: Dual command connections. An example is the command connection from a 2CMD block.

INT: Integer fields or connections that take any integer value. Examples are the Number of Inputs field (OR block) and the Priority connection (CMD block).

N: Fields that cannot be modified once you have added the block to the archive data base. Only the compound template for a Process and Restart compound has this type of field. An example is the Process System Name field for a process compound.

READ-ONLY: Read-only fields that cannot be edited. The GPL Editor enters values in read-only fields. An example is the Command field.

READ: Read attribute connections. An example is the Read connection (READ block).

STR: Character string fields that allow any combination of alphabetic characters (A-Z, a-z), international language characters, numbers (0-9), and the underscore (_). An example is the Block Name field (most blocks). Refer to *Appendix F* for a list of reserved names, international characters, and invalid symbols that cannot be used in STR fields.

TAB: Fields that toggle between different entries. A different entry displays each time you press the TAB key. Once you have tabbed through all the choices, the first choice reappears. To keep a selection, press Enter or an arrow key. An example is the I/O Connections field for the CONN block.

TIM: Time input connections that take values from 00:00:00 to 23:59:59. An example is the Time input connection of a SVAR block.

WRIT: Write attribute connections. Examples are the Write input and output connections of a WRIT block.

Default

The value that the GPL Editor automatically puts in the field. The default value is based on common applications; therefore, you do not have to change it in most cases.

Range

The range of acceptable values; or, the acceptable or available choices. The Char entry means characters. The Real and Time entries mean any number in these ranges:

Real: 99999999 to 0.000001, 0.0, -0.00001 to -9999999.

Time: 00:00:00 to 23:59:59.

Note: The GPL Editor converts all decimal numbers into binary format; therefore, only five of the eight possible digits can be displayed reliably. For example, if -49.4567 is entered, the Editor will display -49.4566, not -49.457. This is because rounding errors are introduced after the fifth digit. *Decimal number conversion is unpredictable after the fifth digit.*

Also, the Editor forces a fraction to have a 0 before the decimal point. The least significant digit cannot be displayed. For example, if you enter -.123456, the Editor displays -0.12345 (the number 6 is dropped).

RC (Required Connection)

Whether the block requires this connection ("Y" means Yes and "N" means No). Note that a required connection may actually be conditional; for example, a time connection is not required if an analog connection is made.

T (Template Entry)

Whether this attribute or parameter is an entry on the data base template. An "X" indicates the attribute or parameter is definable in the template.

I (Input Connection)

Whether this connection is an input. An "X" indicates an input connection.

O (Output Connection)

Whether this connection is an output. An "X" indicates an output connection.

LB (Label)

The one or two character abbreviation of the connection name. This label appears at the start or end of a connection line. Examples are I1 for Input 1 and V for Value.

Example

Each example indicates a process period, which is how often the process should execute. These period values are guidelines only. The actual process periods you choose should be based on the application and the type of controlled mechanical equipment.

ABRT (Abort) Block

Category

Process Control

Purpose

Ends a process abnormally, usually when some error condition is detected via graphic logic.

Details

This block places the process in the Error state. Once aborted, the process is not executed on its period or when triggered.

The process can be executed again by:

- a manual command (Enable command)
- another process (Process Enable command)
- redownloading the process in the Enabled state

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will abort a process only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not abort the process. If the Enable input is not connected, the block will abort the process each time it is executed.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 char		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

When this block is executed and the Enable input (if connected) is False or unreliable, the process is not aborted. If the Enable input is not connected, no reliability check is made.

Example

In this example (Figure 33), an ABRT block aborts the process, based on the condition of a fan, which is represented by a BO object (AHU1\FAN), and the status of a BI object (AHU1\FAN-STAT). When the fan starts, the process is triggered to execute. At this point, the output of the DLAY block is False, so the AND block evaluates to False and the process is not aborted.

After 10 minutes, the DLAY block timer expires, which triggers the process to execute again. Now, the output of the DLAY block is True. The XOR block reads the values of the BO and BI objects. If one is True and the other False, the XOR block will be True. Then, the AND block will be True, and the process will abort.

A period of 00:00:00 is specified for this process, since it only needs to run when a triggerable attribute of the BO object changes.

Note: All operation blocks in this example must be placed in a process. They include: ABRT, AND, DLAY, and XOR.

Also, the connections in this example dictate the order of execution. In some cases, you may want to use control flow lines to guarantee that the ABRT block executes when expected, and doesn't end execution before all desired blocks have executed.

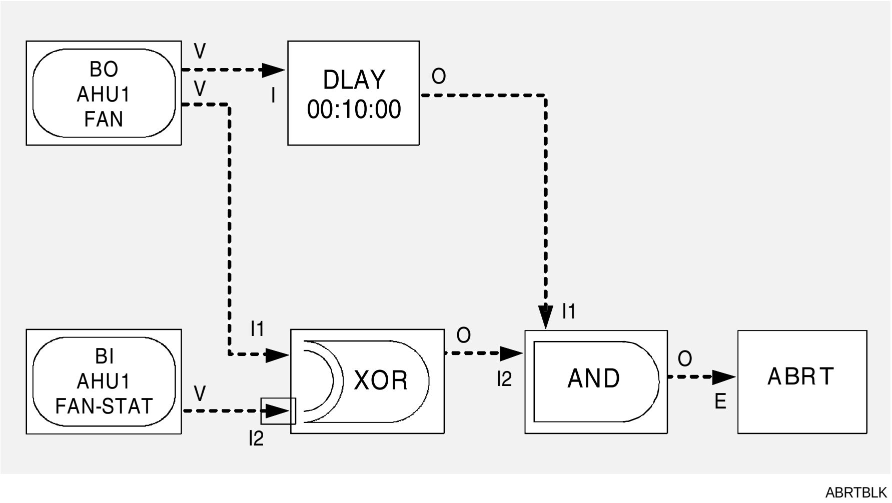


Figure 33: ABRT Block Example

ADD (Addition) Block

Category

Math

Purpose

Adds two inputs using the following equation:

Input 1 + Input 2 = Output

Details

The ADD block can accept either analog or time inputs. If you select analog in the Type field, real math is used. If you choose time, time math is used. Time math is performed in 24-hour format, for example:

10:00:00 + 11:17:10 = 21:17:10
11:00:00 + 15:00:00 = 02:00:00

You can specify constant values for Input 1 and Input 2 instead of connecting external inputs. Define the constant values in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, TIME		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
	TIME IN2	TIM	00:00:00	Time	N	X	X		T2
-	OUTPUT	ANA	0.000000	Real	Y			X	O
		TIM	00:00:00	Time	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the ADD block is unreliable if either or both of its inputs are unreliable.

Example

The ADD block in this example (Figure 34) adds the values of two flow sensors (CPLANT1\FLOW1 and CPLANT1\FLOW2), the sum of which is assigned to an analog SVAR block (TOT_FLOW).

A period of 00:02:00 minutes is defined for this process, which means it will run once every two minutes.

Note: The ADD and SVAR blocks, both operation blocks, must be placed in a process.

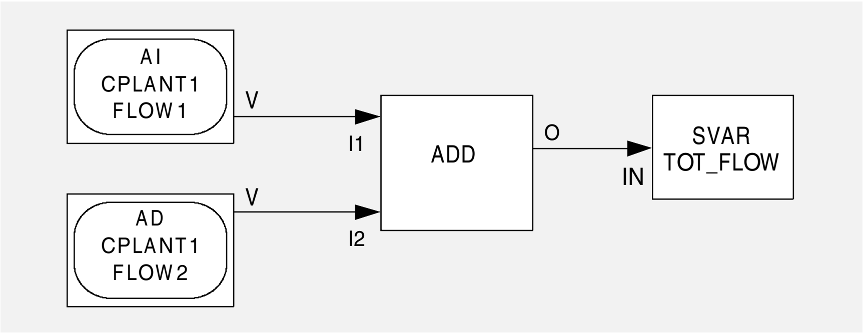


Figure 34: ADD Block Example

ADV Block

Category

Report

Purpose

Sends an advisory message to correct devices or files based on the access report groups. The system name of the process that contains the ADV block determines the access report group. Each time this block is executed, the advisory will be sent to the correct devices or files.

Details

You can configure this block to accept analog, binary, or time data. The default is analog. If you want to connect binary or time data, you must first make that selection in the template before making a connection.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will send the message only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not send the message. If the Enable input is not connected, the block will send a message each time it is executed.

You can append a value to the end of the message by connecting a data line from an object block to the ADV block. The value may be analog, binary, or time data. Here is a message example.

```
CRIT 1 ADVISORY LOOP IS SATURATED.  SETPOINT IS 55.0   6/26/90 16:48:14
```

In this example, the text entered in the template is “Loop is saturated. Setpoint is.” The type of data selected is Analog. The value 55.0 is 55.0°F, and comes from a duct sensor.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Advisory Type		TAB	STATUS	See Note A		X			
Text		STR	BLANK	50 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
-	INPUT	BIN	0	0,1	N		X		I
		ANA	0.000000	Real	N		X		IN
-	TIME IN	TIM	00:00:00	Time	N		X		TI
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note B	0,1	N			X	E
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note A: The Advisory Type field can be any of the following: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW_UP, or STATUS.

Note B: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Note: *Appendix F: Characters, Symbols, and Reserved Words* lists valid characters for use in messages.

Reliability

No advisory message is sent if the Enable input is False or unreliable. The advisory does not indicate whether the input value is reliable.

Example

In this example (Figure 35), the ADV block reads the value of the HI_SAT_FLAG attribute (“HF” connection) from a PIDL object (AHU1\PID-CLG). If the flag is True and Reliable, the ADV block sends a report to the Report Router feature in the NCM, which, in turn, sends a message to the Operator Workstation that the HI_SAT_FLAG is set. The message is:

```
THE COOLING PIDL IS HI SATURATED.
```

A period of 00:00:00 is defined for this process since it needs to run only when the triggerable attribute HI_SAT_FLAG changes.

Note: The ADV block, an operation block, must be placed in a process.

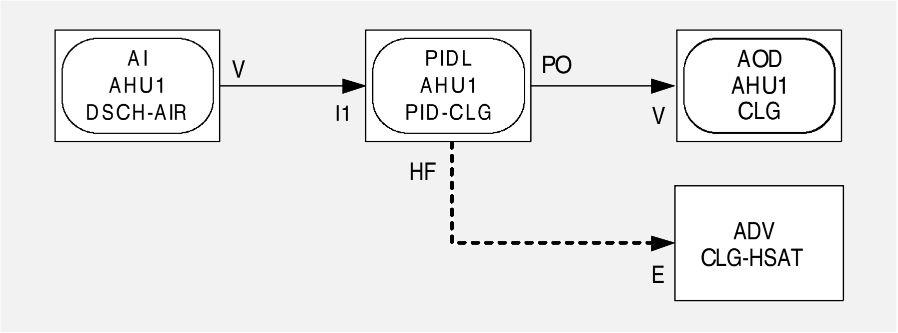


Figure 35: ADV Block Example

AND Block

Category

Logic

Purpose

Performs a logical AND operation on two, three, or four binary inputs.

Details

If all binary inputs into this block are True, then the output is True; otherwise, the output is False.

You can configure this block to have two, three, or four binary inputs. The default is two. If you want to connect more than two inputs, you must first specify the number of inputs in the template before making more than two connections.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Number of Inputs		INT	2	2,3,4		X			
-	INPUT 1	BIN		0,1	Y		X		I1
-	INPUT 2	BIN		0,1	Y		X		I2
-	INPUT 3	BIN		0,1	Y		X		I3
-	INPUT 4	BIN		0,1	Y		X		I4
-	OUTPUT	BIN	0	0,1	Y			X	O
-	CONTROL				N		X		CF
-	CONTROL				N			X	CF

Reliability

The output of the AND block is unreliable if any of its inputs are unreliable.

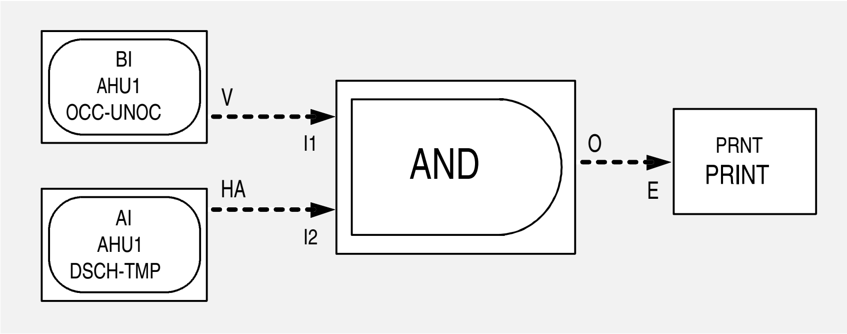
Example

In this example, (Figure 36), the AND block determines if an alarm message should be sent to the printer. The AND has as inputs a BI object block for occupied/unoccupied mode (AHU1\OCC-UNOC) and an AI object block for discharge air temperature (AHU1\DSCH-TMP). If the building is in occupied mode and the discharge air temperature is in high alarm, this message is printed:

```
AHU1\DSCH-TMP IS IN HIGH ALARM (OCCUPIED MODE).
```

A period of 00:00:00 is defined for this process since it should only run when triggered by the VALUE attribute of the BI object or the HI_ALARM attribute of the AI object.

Note: The AND and PRNT blocks, both operation blocks, must be placed in a process.



andblk

Figure 36: AND Block Example

AVG (Average) Block

Category

Math

Purpose

Averages up to four inputs using the following equation:

(Input 1 + Input 2 + Input 3 + Input 4)/n = Output

where n = number of configured inputs (n = 2, 3, or 4).

Details

You can configure this block to have two, three, or four inputs. The default is two. If you want to connect three or four inputs, you must specify the number of inputs in the template before making these connections.

Also, you can specify constant values for Input 1 and Input 2 instead of connecting external inputs. Enter the constant values in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
Number of Inputs		INT	2	2,3,4		X			
-	INPUT 3	ANA		Real	N*		X		I3
-	INPUT 4	ANA		Real	N*		X		I4
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL				N		X		CF
-	CONTROL				N			X	CF

* Input 3 is a required connection if you enter 3 in the Number of Inputs field. Input 4 is a required connection if you enter 4 in this field.

Reliability

The output of the AVG block is unreliable if any of its inputs are unreliable.

Example

The AVG block in the following example (Figure 37) averages three zone temperatures (AHU1\ZONE1,AHU1\ZONE-T2,AHU1\ZONE-T3). The SVAR block reads the result.

A period of 00:03:00 is defined for this process, which means the process will run once every three minutes.

Note: The AVG and SVAR blocks, both operation blocks, must be placed in a process.

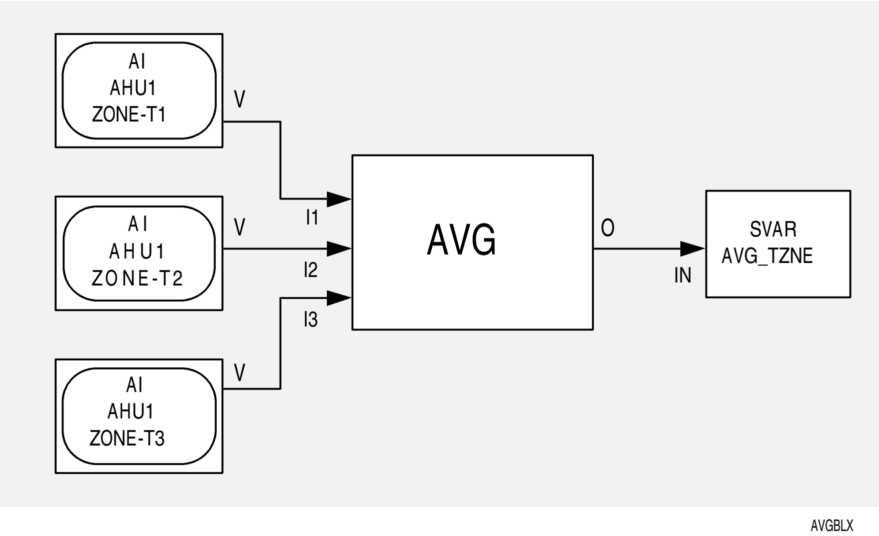


Figure 37: AVG Block Example

BSEQ (Binary Sequencer) Block

Category

Control

Purpose

Controls up to eight stages of heating, direct expansion cooling, or other similar control applications.

Details

The function of the BSEQ block is to increment and decrement through the configured stages. A stage is a configured set of binary values for the outputs. The number of binary values configured for each stage is equal to the configured number of outputs.

The functional operation of the BSEQ block is based on the reliability of the input, and the value and reliability of B-Enable. Their relationship to the block's operation is shown in the following table:

Inputs	Functional Operation
1. B-Enable is True and reliable or unconnected. **2. Input is reliable.**	BSEQ is in staging operation.
1. B-Enable is False and reliable. **2. Input is reliable.**	BSEQ is always at Stage 0.
1. B-Enable is Unreliable and/or **2. Input is unreliable.**	BSEQ stages to the Fail-safe Stage and remains there as long as the unreliable condition exists.

When the BSEQ block is in the staging operation, the input is compared to one of the setpoints, and the block will increment, decrement, or stay at the present stage, based on the following comparisons:

FOR < OPERATOR:	
INCREMENT STAGE	1. Delay On timer for next stage has expired; and 2. Input < Make Setpoint.
DECREMENT STAGE	1. Delay Off timer for current stage has expired; and 2. Input ≥ Break Setpoint.
NO STAGING	None of the above conditions exist.

FOR > OPERATOR:	
INCREMENT STAGE	1. Delay On timer for next stage has expired; and 2. Input > Make Setpoint
DECREMENT STAGE	1. Delay Off timer for current stage has expired; and 2. Input ≤ Break Setpoint
NO STAGING	None of the above conditions exist.

Note: When the BSEQ block evaluates to No Change, the block is evaluated again the next time the process is triggered.

The first time the BSEQ block executes, the Delay On timer for Stage 1 is set. Then, when a stage change occurs, the Delay On timer of the next highest stage and the Delay Off timer of the current stage are set. However, if the BSEQ block is currently at the highest configured stage, only the Delay Off timer for that stage is set. Similarly, if the BSEQ block is at Stage 0, only the Delay On timer for Stage 1 is set. If the stage changes and the opposite-direction timer has not expired, that timer is canceled.

You can configure the following for this block:

- Number of stages, from one to eight.
- Operation, either less than (<) or greater than (>). The operation you choose determines how the block will function (see tables above).
- Number of outputs, from one to eight.
- Fail-safe Stage, either Current Stage, or Stage 0 through Stage 8.
- Make Setpoints, Break Setpoints, Delay On timers, and Delay Off timers.

Follow the two rules below when defining the Make and Break Setpoints. If you do not follow these rules, the BSEQ block will not function properly.

> If you select the > operator: Define the Make Setpoints and Break Setpoints in ascending order (e.g., Make Setpoint 1=68.0 and Make Setpoint 2=69.0).
>
> If you select the < operator: Define the Make Setpoints and Break Setpoints in descending order (e.g., Make Setpoint 1=67.0 and Make Setpoint 2=65.0).

Note: The block will stage if you make all the Make Setpoints and Break Setpoints equal.

Here are some important characteristics of the BSEQ block:

- If you configure a Fail-safe Stage that is greater than the largest configured stage, the BSEQ block will use the largest configured stage.
- The BSEQ block increments and decrements only one stage at a time. You cannot configure it to skip a stage.
- The BSEQ uses the Fail-safe Stage when the analog input or the B-Enable input to the block is unreliable.
- The process in which the BSEQ is used is triggered when a Delay On or Delay Off timer expires.
- If the B-Enable is not connected, its value is always True and reliable.

Note: For most applications, place the BSEQ block in a process with a non-zero period that represents the response time required to respond to changes in the input when both timers have expired.

Information Table (First Screen)

Note: Number of rows and columns displayed will depend on the number of stages and the number of outputs specified on the first page of the template.

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Num. of Stages		INT	2	1 - 8		X			
Operation		TAB	<	< or >		X			
Num. of Outputs		INT	2	1 - 8		X			
Fail-safe Stage		TAB	CURRENT	CURRENT 0,1,...,8		X			
-	INPUT	ANA	0.000000	Real	N		X		IN
-	B-ENABLE	BIN	1	0,1	N		X		BE
-	CONTROL	CTL		N			X		CF
-	CONTROL	CTL		N				X	CF

Information Table (Second Screen)

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 0 OUTPUT 1	OUTPUT 1	BIN	0	0,1	Y			X	O1
STAGE 0 OUTPUT 2	OUTPUT 2	BIN	0	0,1	Y			X	O2
STAGE 0 OUTPUT 3	OUTPUT 3	BIN	0	0,1	Y			X	O3
STAGE 0 OUTPUT 4	OUTPUT 4	BIN	0	0,1	Y			X	O4
STAGE 0 OUTPUT 5	OUTPUT 5	BIN	0	0,1	Y			X	O5
STAGE 0 OUTPUT 6	OUTPUT 6	BIN	0	0,1	Y			X	O6
STAGE 0 OUTPUT 7	OUTPUT 7	BIN	0	0,1	Y			X	O7
STAGE 0 OUTPUT 8	OUTPUT 8	BIN	0	0,1	Y			X	O8

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 1 MAKE SETPOINT		ANA	68.00000	Real		X			
STAGE 2 MAKE SETPOINT		ANA	66.50000	Real		X			
STAGE 3 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 4 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 5 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 6 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 7 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 8 MAKE SETPOINT		ANA	0.000000	Real		X			
STAGE 1 BREAK SETPOINT		ANA	69.00000	Real		X			
STAGE 2 BREAK SETPOINT		ANA	68.00000	Real		X			
STAGE 3 BREAK SETPOINT		ANA	0.000000	Real		X			
STAGE 4 BREAK SETPOINT		ANA	0.000000	Real		X			
STAGE 5 BREAK SETPOINT		ANA	0.000000	Real		X			
STAGE 6 BREAK SETPOINT		ANA	0.000000	Real		X			
STAGE 7 BREAK SETPOINT		ANA	0.000000	Real		X			
STAGE 8 BREAK SETPOINT		ANA	0.000000	Real		X			

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 1 DELAY ON		TIM	00:02:00	Time		X			
STAGE 2 DELAY ON		TIM	00:03:00	Time		X			
STAGE 3 DELAY ON		TIM	00:00:00	Time		X			
STAGE 4 DELAY ON		TIM	00:00:00	Time		X			
STAGE 5 DELAY ON		TIM	00:00:00	Time		X			
STAGE 6 DELAY ON		TIM	00:00:00	Time		X			
STAGE 7 DELAY ON		TIM	00:00:00	Time		X			
STAGE 8 DELAY ON		TIM	00:00:00	Time		X			
STAGE 1 DELAY OFF		TIM	00:00:10	Time		X			
STAGE 2 DELAY OFF		TIM	00:00:15	Time		X			
STAGE 3 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 4 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 5 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 6 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 7 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 8 DELAY OFF		TIM	00:00:00	Time		X			
STAGE 1 OUTPUT 1		BIN	1	0,1		X			
STAGE 1 OUTPUT 2		BIN	0	0,1		X			
STAGE 1 OUTPUT 3		BIN	0	0,1		X			

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 1 OUTPUT 4		BIN	0	0,1		X			
STAGE 1 OUTPUT 5		BIN	0	0,1		X			
STAGE 1 OUTPUT 6		BIN	0	0,1		X			
STAGE 1 OUTPUT 7		BIN	0	0,1		X			
STAGE 1 OUTPUT 8		BIN	0	0,1		X			
STAGE 2 OUTPUT 1		BIN	1	0,1		X			
STAGE 2 OUTPUT 2		BIN	1	0,1		X			
STAGE 2 OUTPUT 3		BIN	0	0,1		X			
STAGE 2 OUTPUT 4		BIN	0	0,1		X			
STAGE 2 OUTPUT 5		BIN	0	0,1		X			
STAGE 2 OUTPUT 6		BIN	0	0,1		X			
STAGE 2 OUTPUT 7		BIN	0	0,1		X			
STAGE 2 OUTPUT 8		BIN	0	0,1		X			
STAGE 3 OUTPUT 1		BIN	0	0,1		X			
STAGE 3 OUTPUT 2		BIN	0	0,1		X			
STAGE 3 OUTPUT 3		BIN	0	0,1		X			
STAGE 3 OUTPUT 4		BIN	0	0,1		X			
STAGE 3 OUTPUT 5		BIN	0	0,1		X			
STAGE 3 OUTPUT 6		BIN	0	0,1		X			
STAGE 3 OUTPUT 7		BIN	0	0,1		X			
STAGE 3 OUTPUT 8		BIN	0	0,1		X			

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 4 OUTPUT 1		BIN	0	0,1		X			
STAGE 4 OUTPUT 2		BIN	0	0,1		X			
STAGE 4 OUTPUT 3		BIN	0	0,1		X			
STAGE 4 OUTPUT 4		BIN	0	0,1		X			
STAGE 4 OUTPUT 5		BIN	0	0,1		X			
STAGE 4 OUTPUT 6		BIN	0	0,1		X			
STAGE 4 OUTPUT 7		BIN	0	0,1		X			
STAGE 4 OUTPUT 8		BIN	0	0,1		X			
STAGE 5 OUTPUT 1		BIN	0	0,1		X			
STAGE 5 OUTPUT 2		BIN	0	0,1		X			
STAGE 5 OUTPUT 3		BIN	0	0,1		X			
STAGE 5 OUTPUT 4		BIN	0	0,1		X			
STAGE 5 OUTPUT 5		BIN	0	0,1		X			
STAGE 5 OUTPUT 6		BIN	0	0,1		X			
STAGE 5 OUTPUT 7		BIN	0	0,1		X			
STAGE 5 OUTPUT 8		BIN	0	0,1		X			
STAGE 6 OUTPUT 1		BIN	0	0,1		X			
STAGE 6 OUTPUT 2		BIN	0	0,1		X			
STAGE 6 OUTPUT 3		BIN	0	0,1		X			
STAGE 6 OUTPUT 4		BIN	0	0,1		X			

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
STAGE 6 OUTPUT 5		BIN	0	0,1		X			
STAGE 6 OUTPUT 6		BIN	0	0,1		X			
STAGE 6 OUTPUT 7		BIN	0	0,1		X			
STAGE 6 OUTPUT 8		BIN	0	0,1		X			
STAGE 7 OUTPUT 1		BIN	0	0,1		X			
STAGE 7 OUTPUT 2		BIN	0	0,1		X			
STAGE 7 OUTPUT 3		BIN	0	0,1		X			
STAGE 7 OUTPUT 4		BIN	0	0,1		X			
STAGE 7 OUTPUT 5		BIN	0	0,1		X			
STAGE 7 OUTPUT 6		BIN	0	0,1		X			
STAGE 7 OUTPUT 7		BIN	0	0,1		X			
STAGE 7 OUTPUT 8		BIN	0	0,1		X			
STAGE 8 OUTPUT 1		BIN	0	0,1		X			
STAGE 8 OUTPUT 2		BIN	0	0,1		X			
STAGE 8 OUTPUT 3		BIN	0	0,1		X			
STAGE 8 OUTPUT 4		BIN	0	0,1		X			
STAGE 8 OUTPUT 5		BIN	0	0,1		X			
STAGE 8 OUTPUT 6		BIN	0	0,1		X			
STAGE 8 OUTPUT 7		BIN	0	0,1		X			
STAGE 8 OUTPUT 8		BIN	0	0,1		X			

Reliability

The outputs of the BSEQ block are always reliable. If the input or B-Enable input is unreliable, the output will take on the state defined by the Fail-safe Stage, which can be either Current Stage, Stage 0, or Stage 1 to Stage 8.

Example

The example in Figure 38 contains a BSEQ block that has two stages of electric heat and two outputs configured. The BSEQ has a BI object for occupied mode (AHU1\OCC) as the B-Enable input. An AI object for discharge air temperature (AHU1\DIS-TEMP) is the analog input. The values specified in the BSEQ template are (default values):

Stage	Make Setpoint	Break Setpoint	Delay On	Delay Off	Outputs 12345678
0					00
1	68.0	69.0	00:02:00	00:00:10	10
2	66.5	68.0	00:03:00	00:00:15	11

The BSEQ block will operate when the building becomes occupied. When the discharge air temperature is less than 68.0°F, a Start command is sent to AHU1\HTG1, which turns on the first stage of heat. When the discharge air temperature is less than 66.5°F, the two BO objects AHU1\HTG1 and AHU1\HTG2 are sent Start commands, which turn on the first and second stages of heat on a rise in temperature. When the discharge air temperature is greater than or equal to 68.0°F, the BO objects are sent Stop commands, which de-energize the second stage of heat. The first stage of heat remains on a further rise in temperature, when the discharge air temperature is greater than or equal to 69.0°F, the BO objects are sent Stop commands, which de-energize the first and second stages (i.e., all stages of heat off).

A period of 00:03:00 minutes is defined for this process, which means it will run once every three minutes.

Note: All operation blocks in this example must be placed in a process. These include: BSEQ and both 2CMD blocks.

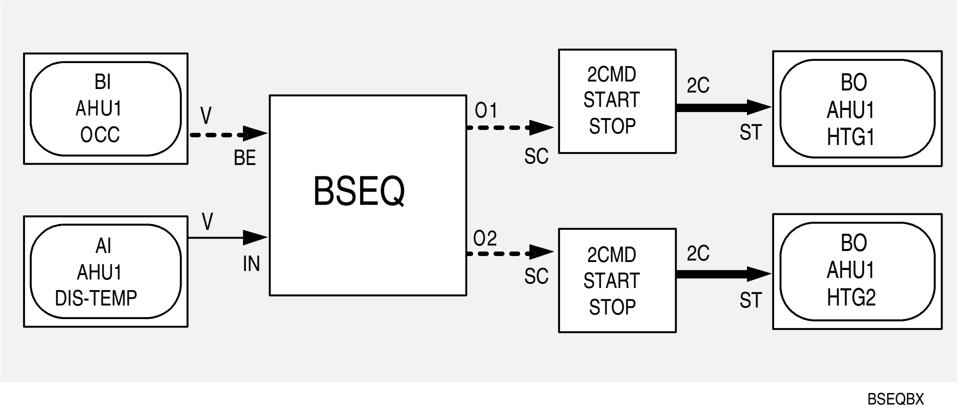


Figure 38: BSEQ Block Example

CMD (Command) Block

Category

Object Control

Purpose

Commands object blocks. This changes an object's attribute values and/or invokes an algorithm in the object (e.g., COS analysis). For example, you would use a CMD block to Start or Stop a binary output object.

Details

Some commands have parameters that must be specified with the command, such as High Limit, Low Limit, and Priority. You define these parameters in the command block's template. The CMD block is labeled with the name of the command (e.g., SET BD for the BD object).

These important factors relate to command blocks:

- *Before you can edit the CMD template or enter the block's parameters, you must connect the output of the CMD block to an object block or to a CONN block.*
- If you erase a command connection and try to connect a different command, you'll notice that the connection menu contains the previous command only.
- To change the type of command selected, you need to erase the CMD block from the work area, then paste down and define a new one. However, if this is an origin remote CMD block, you may select a different origin block by simply double-clicking left on the CMD block (with Connection icon selected) and selecting a different command. Yet, even in this case it is best to erase the origin remote connection and re-add it, since the Editor does not update the line label at the destination block or verify that you selected a valid command for that object. An invalid command is not detected until the file is translated and compiled.

- For some commands, you can delete a command parameter by entering the letter "D" in place of a value. When the command is sent, the current parameter value will be deleted. For example, placing a D in the High Alarm Limit field means that the next time the object is commanded, no High Alarm Limit will exist for the object.
- You may enter a blank in the parameter fields of some commands. This will avoid sending a new parameter value with the command. That is, the object will maintain its previous parameter value.
- The CMD block cannot display the underscore (_) that some commands contain (e.g., SET_AD). It displays a blank space instead (i.e., SET AD).
- You may use the CMD block to modify integer parameters and attributes. The NCM will convert the analog value to integer by rounding. The rounding rules are:

  0.00 - 0.49: Round off
  0.50 - 0.99: Round up

To command a REF block with the CMD block, keep in mind the following points:

- The selected command must be valid for the referenced object. For example, the REL_CS command should not be sent to a REF block that is configured for C260X. The Compiler verifies that the command is valid.
- The selected command must be valid for the attribute. (Refer to following table.) For example, even though the STCSAN command is valid for the CS object, it is not valid for the BI_1 attribute, since STCSAN is an analog-type command. Note that the Editor, Expert Checker, and Compiler do not check for invalid commands. If an object receives an invalid command, it simply ignores it.

- The selected attribute must be valid for the software object/model or hardware object. For example, if the AI_5 attribute is not configured in the model, it *should not be* used in GPL. The Compiler verifies that the attribute is valid.

The following table lists which attributes are valid for which REF block commands.

REF Block Command	**Valid Attributes**
STCSAN	AI_1 to AI_16 AO_1 to AO_16 AD_1 to AD_32 SP_1 to SP_32
STCSBN	BI_1 to BI_16 BO_1 to BO_16 BD_1 to BD_32
STCSMS	MS_1 and MS_2
REL_CS	AI_1 to AI_16 AO_1 to AO_16 AD_1 to AD_32 SP_1 to SP_32
	BI_1 to BI_16 BO_1 to BO_16 BD_1 to BD_32
	MS_1 and MS_2
ST500XAN	AI1VAL to AI6VAL AO1VAL to AO6VAL
ST500XBN	BI1VAL to BI5VAL BO1VAL to BO4VAL
ST500XSP	SP1VAL to SP16VAL
REL_500X	SP1VAL to SP16VAL
ST260XAN	AI1VAL to AI6VAL VO1VAL and AO2VAL
ST260XBN	BI1VAL to BI6VAL BO1VAL to BO5VAL
ST260XSP	SP1VAL to SP12VAL
REL_260X	SP1VAL to SP12VAL

BEG_TOT
END_TOT
BEG_TRND
END_TRND
	See Totalization and Trend features for valid attributes.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the command will be sent only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the command will not be sent. If the Enable input is not connected, the command will be sent each time the CMD block executes.

Information Tables

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command			See Note A	See Note A		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note B	0,1	N			X	E
-	COMMAND	CMD			Y			X	C
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note A: Refer to the following table to determine which of the various tables describe a particular command. For a list of which objects accept which commands, refer to the *Software Data Sheets* section in the *Metasys Network Technical Manual.*

Note B: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Table	Command(s)
1	ALARMS
2	AUX_DIS, AUX_ENA, LOC_REP, LOC_TRG, OFF, ON, PRC_DIS, PRC_ENA, RELEASE, RELEASE3, STARTUP, TRIGGER, UNL_REP, UNL_TRG, UNLATCH
3	BEG_TOT, BEG_TRND, END_TOT, END_TRND
4	REL_CS, REL_260X, REL_500X
5	REL_PIDL
6	REL_PRI
7	REL_210A
8	REL_260A
9	RES_TOT
10	SET_AD, SET_AOS
11	SET_AOD
12	SET_BD
13	SET_PIDL
14	START, STOP
15	STCSAN, STCSMS, ST260XSP, ST500XSP
16	STCSBN
17	ST210AAN
18	ST210ABN
19	ST210ASP
20	ST260AAN
21	ST260ABN
22	ST260ASP
23	ST260XAN, ST500XAN
24	ST260XBN, ST500XBN
25	TIMED_ON
26	WARNINGS
27	SET_MC, SET_MSD, SET_MSO

Table 1									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	ALARMS	8 Char		X			AL
Low Limit	LO LIMIT	ANA	40.00000	See Note	N	X	X		LL
High Limit	HI LIMIT	ANA	80.00000	See Note	N	X	X		HL
Differential	DIFF	ANA	1.000000	Real >0 See Note	N	X	X		DF

Note: This field can accept one of three entries: a real value, the letter "D" for Delete, or no entry at all (blank). Enter a D to delete the object's parameter when the command is sent. Enter a blank to keep the current parameter value in the object.

Table 2									
Field Name	**Connect Name**	**Type**	**Defaul t**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY		AUX_DIS		X			AD
				AUX_ENA		X			AE
				LOC_REP		X			LR
				LOC_TRG		X			LT
				OFF		X			OF
				ON		X			ON
				PRC_DIS		X			DS
				PRC_ENA		X			EN
				RELEASE		X			R
				RELEASE3		X			R3
				STARTUP		X			SU
				TRIGGER		X			TR
				UNL_REP		X			UR
				UNL_TRG		X			UT
				UNLATCH		X			UL

Table 3

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		BEG_TOT		X			BT
				BEG_TRND		X			BH
				END_TOT		X			ET
				END_TRND		X			EH
Attribute Name		STR	VALUE	8 Char		X			

Table 4

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		REL_CS		X			RL
				REL_260X		X			RL
				REL_500X		X			RL
Attribute		STR	Blank	8 Char		X			
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Table 5

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	REL_PIDL	8 Char		X			RP
Attribute		TAB	AUX_IN	AUX_IN		X			
				INPT1VAL		X			
				INPT2VAL		X			
				INPT3VAL		X			
				INPT4VAL		X			
				INPT5VAL		X			
				INTP6VAL		X			
				SETPOINT		X			
				OFFSET		X			
				HI_SAT_V		X			
				LO_SAT_V		X			
				SEL_INP		X			

Table 6									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	REL_PRI	8 Char		X			RP
Priority	PRIORITY	INT	7	2,4,5,6,7	N	X	X		PR

Table 7									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	REL_210A	8 Char		X			RL
Attribute		TAB	ZNSP	ZNSP		X			
				MXDP		X			
				MINDP		X			
				AXDP		X			
				HCPB		X			
				HTDB		X			
				RZSP		X			
				INTG		X			
				SUSB		X			
				WTMP		X			
				DPSP		X			
				SETPOINT		X			
Priority	PRIORITY	INT	2	2,3	N	X	X		PR

Table 8									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	REL_260A	8 Char		X			RL
Attribute		TAB	ZNSP	ZNSP		X			
				STUP		X			
				STBK		X			
				CLPB		X			
				HTPB		X			
				HTDB		X			
				RZSP		X			
				INTG		X			
				SETPOINT		X			
Priority	PRIORITY	INT	2	2,3	N	X	X		PR

Table 9

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	RES_TOT	8 Char		X			RT
Attribute		STR	VALUE	8 Char		X			
Value	VALUE	ANA	0.000000	Real ≥ 0 See Note	N	X	X		V

Note: The RES-TOT command has different ranges for Event, Analog, Pulse, and Runtime types of totalization. Make sure that the value you specify in the template is within the range for the particular totalization type.

Table 10

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		SET_AD		X			SA
				SET_AOS		X			SA
Value	VALUE	ANA	0.000000	Real	N	X	X		V
Priority	PRIORITY	INT	2	2,3	N	X	X		PR

Table 11

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	SET_AOD	SET_AOD See Note		X			SA
Value	VALUE	ANA	0.000000	Real	N	X	X		V

Note: The SET_AOD command is fixed at priority 2.

Table 12

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	SET_BD	8 Char		X			SB
Value	INPUT	BIN	0	0,1	N	X	X		I
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Table 13

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	SET_PIDL	8 Char		X			SP
Attribute		TAB	AUX_IN	AUX_IN		X			
				INPT1VAL		X			
				INPT2VAL		X			
				INPT3VAL		X			
				INPT4VAL		X			
				INPT5VAL		X			
				INPT6VAL		X			
				SETPOINT		X			
				OFFSET		X			
				HI_SAT_V		X			
				LO_SAT_V		X			
				SEL-INP		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Table 14

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		START		X			ST
				STOP		X			SP
Priority	PRIORITY	TAB	7	2,4,5,6,7	N	X	X		PR

Table 15

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		STCSAN		X			SA
				STCSMS		X			SA
				ST260XSP		X			SA
				ST500XSP		X			SA
Attribute		STR	Blank	8 Char		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Table 16

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	STCSBN	8 Char		X			SA
Attribute		STR	Blank	8 Char		X			
Value	VALUE	BIN	0	0 or 1	N	X	X		V
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Table 17

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	ST210AAN	8 Char		X			AN
Attribute		TAB	ZNT	ZNT		X			
				AUXT		X			
				AUXP		X			
				AUXR		X			
				DFPR		X			
				AUXI		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V

Table 18

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	SET210ABN	8 Char		X			BN
Attribute		TAB	UNOC	UNOC		X			
				SDBC		X			
				SDBO		X			
				LTCH		X			
				WMUP		X			
				SARE		X			
				HRTZ		X			
Value	INPUT	BIN	0	0,1	N	X	X		V

Table 19									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	ST210ASP	8 Char		X			SP
Attribute		TAB	MXDP	MXDP		X			
				MINDP		X			
				AXDP		X			
				HCPB		X			
				HTDB		X			
				ZNSP		X			
				RZSP		X			
				INTG		X			
				SUSB		X			
				WTMP		X			
				DPSP		X			
				SETPOINT		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V
Priority	PRIORITY	TAB	3	2,3	N	X	X		PR

Table 20									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command		READ ONLY	ST260AAN	8 Char		X			AN
Attribute		TAB	ZNT	ZNT		X			
				AUXT		X			
				AUXP		X			
				AUXH		X			
				AUXR		X			
				DFPR		X			
				AUXI		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V

Table 21

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	ST260ABN	8 Char		X			BN
Attribute		TAB	UNOC	UNOC		X			
				FOCO		X			
				LTCH		X			
				SDWN		X			
				HRTZ		X			
Value	INPUT	BIN	0	0,1	N	X	X		V

Table 22

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	ST260ASP	8 Char		X			SP
Attribute		TAB	STUP	STUP		X			
				STBK		X			
				CLPB		X			
				HTPB					
				HTDB		X			
				ZNSP		X			
				RZSP		X			
				INTG		X			
				SETPOINT		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V
Priority	PRIORITY	TAB	3	2,3	N	X	X		PR

Table 23

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		ST260XAN		X			SA
				ST500XAN		X			SA
Attribute		STR	Blank	8 Char		X			
Value	VALUE	ANA	0.000000	Real	N	X	X		V

Table 24

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY		ST260XBN		X			SB
				ST500XBN		X			SB
Attribute		STR	Blank	8 Char		X			
Value	VALUE	BIN	0	0 or 1	N	X	X		V

Table 25

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	TIMED_ON	8 Char		X			TO
Hours	Time	TIM	1.000000	0 to 99.9	N	X	X		T

Table 26

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	WARNINGS	8 Char		X			WC
Setpoint	SETPOINT	ANA	55.00000	See Note 1	N	X	X		SP
Normalband	NORMBAND	ANA	10.00000	See Note 1	N	X	X		NB
Warning Delay	WARNDLAY	ANA	3.0	0-255 min See Note 2	N	X	X		WD
Differential	DIFF	ANA	1.000000	Real > 0 See Note 1	N	X	X		DP

Note 1: This field can accept one of three entries: a real value, the letter “D” for Delete, or no entry at all (blank). Enter a D to delete the parameter. Enter a blank to keep the current parameter value.

Note 2: This field allows a blank, which directs the Translator and Simulator to leave this value unchanged. Also, this field allows whole numbers only.

Table 27

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command		READ ONLY	SET_MC	8 Char		X			SM
			SET_MSD	8 Char		X			SM
			SET_MSO	8 Char		X			SM
Value	INPUT	INTN	0	0,1,2,3	N	X	X		V
Priority	PRIORITY	INT	3	2,3	N	X	X		PR

Reliability

If any of the command parameters are unreliable, the unreliable value(s) and associated reliability flag(s) are sent with the command. For details refer to the *Software Data Sheets* section of the *Metasys Network Technical Manual.*

If an Enable line is connected and its value is unreliable, no command is sent.

Example

In this example (Figure 39), the HSEL block selects the highest of two analog input objects (AHU1\ZN-T1 and AHU1\ZN-T2). The CMD block reads this value and sends it to the AD object.

A period of 00:05:00 is defined for this process, which means it will run once every five minutes.

Note: The HSEL and CMD blocks, both operation blocks, must be placed in a process.

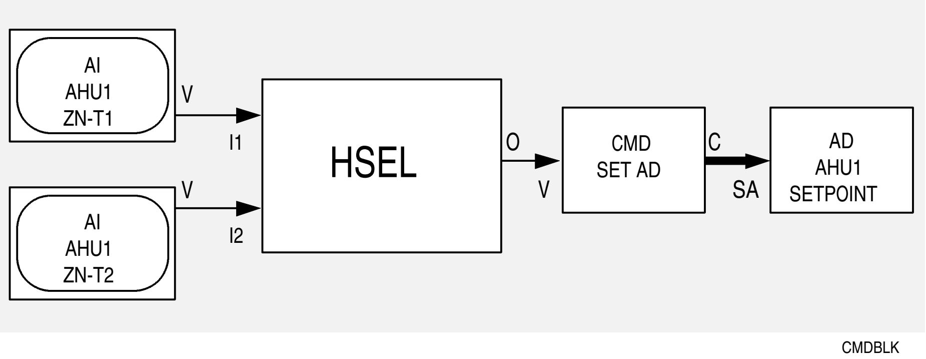


Figure 39: CMD Block Example

CNST (Constant) Block

Category

Data

Purpose

Provides a fixed value that can be an input to another block.

Details

You can configure this block for analog, binary, or time data. The default is analog. If you want to specify binary or time data, you must first select either binary or time in the template before making a connection.

The CNST block does not have to be in a process.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Description		STR	Blank	24 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
Value		ANA	0.000000	Real		X			
		BIN	0	0,1		X			
Time		TIM	00:00:00	Time		X			
	OUTPUT	ANA	See Note	Real	Y			X	O
		BIN	See Note	0,1	Y			X	O
		TIM	See Note	Time	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Note: The default value is the value specified in the template.

Reliability

The output of the CNST block is always reliable.

Example

The following example (Figure 40) shows two CNST blocks as inputs into a SWCH block. The SWCH block will switch between a discharge air temperature (AHU1\DSCH-TMP) High Alarm Limit of 68.0°F and 80.0°F, depending on whether the building is occupied (AHU1\OCC-UNOC). If the BI object is in the Occupied state, the CMD block will change the High Alarm Limit of the discharge air temperature to 68.0°F. If the BI is in the Unoccupied state, the SWCH block will send 80.0°F through, and the CMD block will change the High Alarm Limit to 80.0°F.

A period of 00:00:00 is defined for this process since it needs to run only when the triggerable attribute of the BI object changes.

Note: The SWCH and CMD blocks, both operation blocks, must be placed in a process.

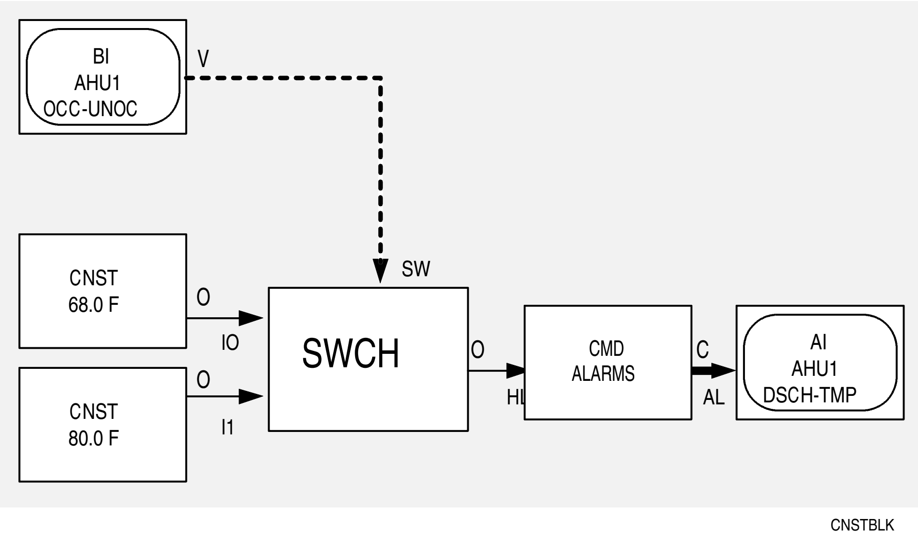


Figure 40: CNST Block Example

COMP (Compare) Block

Category

Control

Purpose

Compares two inputs, and outputs either a True or False state, based on that comparison.

Details

The output of this block is True if the comparison is True; the output is False if the comparison is False.

You can configure this block to compare analog or time inputs. The default is analog. If you want to compare time data, you must first select time in the template before making a connection.

The possible compare operations are:

Greater than (Input 1 > Input 2)
Less than (Input 1 < Input 2)
Equal to (Input 1 = Input 2)
Greater than or equal to (Input 1 ≥ Input 2)
Less than or equal to (Input 1 ≤ Input 2)
Not equal to (Input 1 <> Input 2)

Note: A COMP block that is configured for time and uses the Equal To (=) operator will compare hours and minutes, not seconds. Seconds are truncated before COMP makes the comparison.

Also, if the Equal To operator is used with real inputs, the output will almost always be False, since it is very unlikely that two real values will equal. This is especially true if a floating point operation was done prior to the compare. Similarly, if the Not Equal To (<>) operator is used with real inputs, the output will almost always be True.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, TIME		X			
Operation		TAB	<	<, >, <>, =, ≤, ≥		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
	TIME IN2	TIM	00:00:00	Time	N	X	X		T2
	OUTPUT	BIN	0	0,1	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if Input 1 or Input 2 is unreliable.

Example

The example in Figure 41 contains two COMP blocks that compare system time (TIME) with the fan Early Start/Late Stop times (AHU1\FAN1) to set Occupied mode (OCC). The process sets Occupied mode if the system time is greater than Early Start Time and less than Late Stop Time.

A period of 00:01:00 is defined for this process, which means it will run once every minute.

Note: All operation blocks in this example have to be placed in a process. These include: both COMP blocks, TIME, AND, and SVAR.

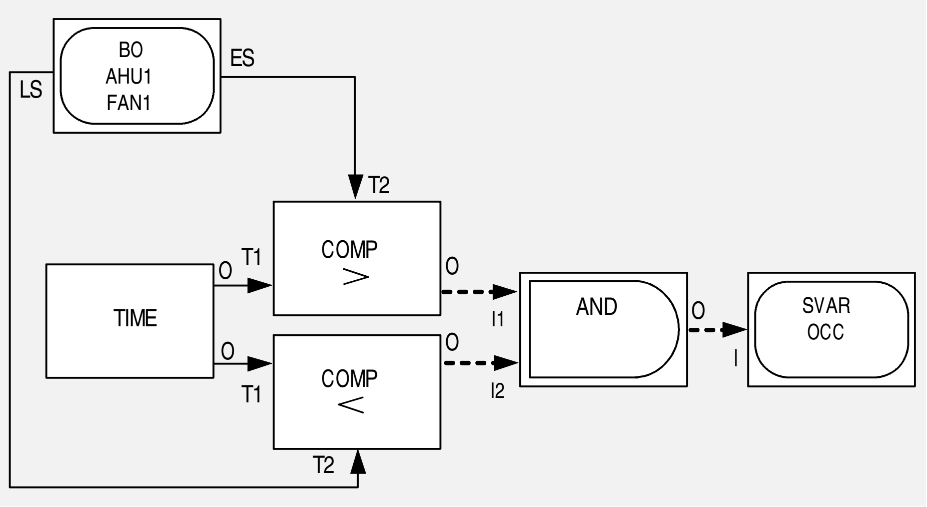


Figure 41: COMP Block Example

CONN (Connection) Block

Category

Data

Purpose

Labels a connection line on the screen.

Details

The CONN block is used primarily for labeling a line that comes from or exits to an external compound.

You can configure this block to accept analog, binary, time, control, command, dual command, read, or write connections. The default is analog. If you want to connect data other than analog, you must first select the data type in the template before making a connection.

With the I/O Connections parameter, you can specify whether this block must be connected to another block. The parameter has two tab choices: OPTIONAL and REQUIRED. If you select OPTIONAL, the output of the CONN block does not require a connection. If you select REQUIRED, the output requires a connection. Also, the letter "R" (colored blue) on the face of the block indicates it is configured as REQUIRED.

Note: You may connect two CONN blocks each in different compounds for the purpose of labeling the line in each compound.

These important factors relate to CONN blocks:

- If the input to an optional CONN block comes from a required output connection, the output of the optional CONN block is required. Similarly, if the output of an optional CONN block goes to a required input, the input of the optional CONN block is required.
- If an optional CONN block is connected to an optional input, and the input of the CONN block is not connected, the template's value in the destination block is used.

- If you need to change the type of command or dual command selected at a CONN block, you need to erase the CONN block and paste down a new one, then reconfigure it in the template.
- If you connect a CMD or 2CMD block to a CONN block, the output connection menu will list the commands of all objects.
- Once you have connected a CONN block that is configured for command or dual command to an object block, the input connection menu will only display the selected command.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	See Note		X			
I/O Connections		TAB	OPTIONAL	OPTIONAL REQUIRED		X			
	INPUT	BIN		0,1	**		X		I
		ANA		Real	**		X		IN
		CMD			**		X		C
		2CMD			**		X		2C
		READ			**		X		RD
		WRIT			**		X		WR
	TIME IN	TIM		Time	**		X		TI
	OUTPUT	BIN	0	0,1	**			X	O
		ANA	0.000000	Real	**			X	O
		TIM	00:00:00	Time	**			X	O
		CMD			**			X	C
		2CMD			**			X	2C
		READ			**			X	RD
		WRIT			**			X	WR
	CONTROL	CTL			**		X		CF
	CONTROL	CTL			**			X	CF

Note: The Type field has the following entries: BINARY, ANALOG, TIME, CONTROL, COMMAND, DUAL CMD, READ, and WRITE.

** A required connection if the I/O Connections field is set to Required. A required or optional connection if this field is set to Optional. Refer to the *Details* section for information.

Reliability

The CONN block passes reliability information along with the data. That is, the output is unreliable if the input is unreliable.

Example

In this example (Figure 42), three CONN blocks indicate data coming from an external diagram, and one CONN block shows data leaving the diagram.

The CONN block labeled OA-TEMP represents data from an AI object named AHU1\OA-TEMP (outdoor air temperature). The SETPOINT block is a setpoint value of an AD object block. The DIFF block is a differential value from an AI object block. All three of these CONN blocks are inputs into the DFCM block. The CONN block labeled SET BD is for a BD object on another diagram.

In this process, the DFCM compares the value of OA-TEMP with SETPOINT. If OA-TEMP is less than SETPOINT, given an applied differential (DIFF), a SET BD command is sent to the BD object.

A period of 00:03:00 is defined for this process, which means it will execute once every three minutes, sending a new command to the BD every three minutes.

Note: The DFCM and CMD blocks, both operation blocks, must be placed in a process.

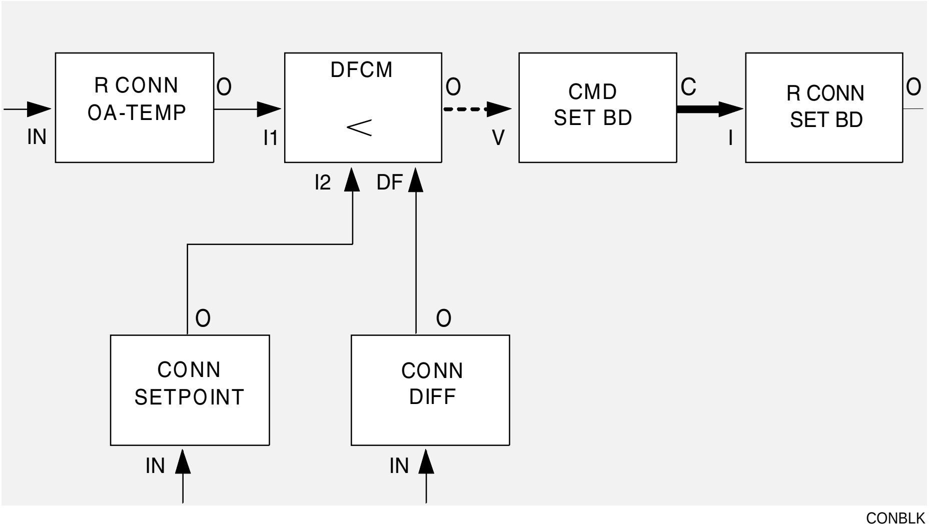


Figure 42: CONN Block Example

DBCM (Deadband Compare) Block

Category

Control

Purpose

Compares two analog inputs. It outputs either a True or False state, depending on the relational operator and the applied deadband.

Details

The two relational operators for this block are Equal To (Input 1 = Input 2) and Not Equal To (Input 1 <> Input 2). The operator determines the behavior of the block as it applies to the deadband. Figures 43 and 44 demonstrate these behaviors.

Note: If the Equal To operator is used with real inputs, the output will almost always be False, since it is very unlikely that two real values will equal. This is especially true if a floating point operation was done prior to the compare. Similarly, if the Not Equal To (<>) operator is used with real inputs, the output will almost always be True.

The deadband is a range within which Input 1 can vary about Input 2 without initiating any change in the output. One-half of the deadband value is on either side of Input 2. For example, if Input 2 is 70.0 and the deadband is 2.0, the deadband range will be from 69.0 to 71.0. With this example, Input 1 will be equal to Input 2 as long as its value is in the deadband range.

Figure 43 shows the function of the DBCM block for the Equal To operator as Input 1 changes. The output of the block is True as long as Input 1 is inside the deadband area (shaded). As Input 1 increases, the output becomes True when
Input 1 = Input 2 - Deadband/2. It changes to False when
Input 1 > Input 2 + Deadband/2. Then, as Input 1 decreases, the output becomes True again when
Input 1 = Input 2 + Deadband/2.

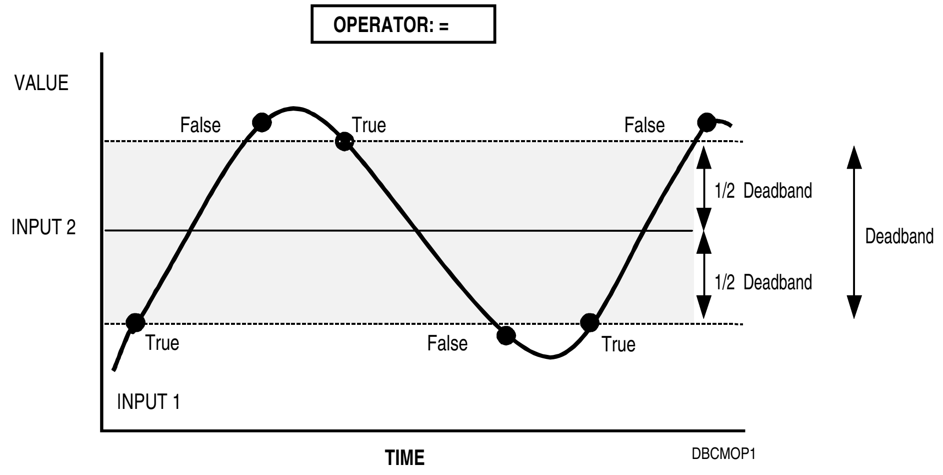


Figure 43: DBCM Operation: Equal To Operator

Figure 44 shows the function of the DBCM block for the Not Equal To operator. The output of the block will be True when the value of Input 1 is outside the deadband area (shaded). As Input 1 increases, the output becomes False when Input 1 ≤ Input 2 - Deadband/2. It changes to True when Input 1 > Input 2 + Deadband/2. Then, as Input 1 decreases, the output becomes False again when Input 1 = Input 2 + Deadband/2.

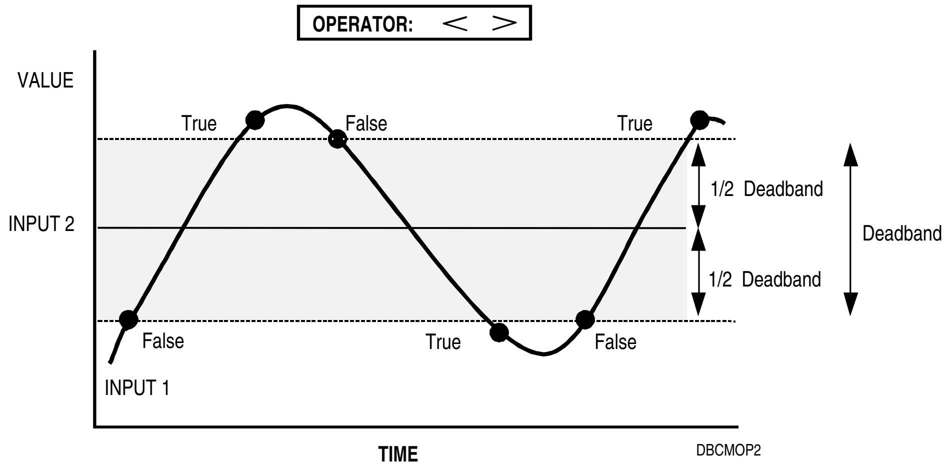


Figure 44: DBCM Operation: Not Equal To Operator

You can configure this block to compare analog inputs only. If you specify a deadband of 0.0, this block works the same as the COMP block. Also, you can specify a constant value for any of the inputs instead of connecting an external input. Enter a constant value in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Operation		TAB	=	=, <>		X			
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
Deadband	DEADBAND	ANA	0.000000	Real ≥ 0	N	X	X		DB
	OUTPUT	BIN	0	0,1	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if Input 1, Input 2, or the Deadband is unreliable.

Example

In the following example (Figure 45), the DBCM block is configured for Not Equal To operation. It compares discharge air temperature (AHU1\DSCH-AIR) with a setpoint (AHU1\SETPOINT) of 60.0°F, given a deadband of 10.0°F (AHU1\DEADBAND). If the DSCH-AIR is not equal to the SETPOINT and the deadband, the PRNT block sends a message to the printer. The message is:

```
DISCHARGE AIR TEMPERATURE IS OUT OF RANGE.
```

A period of 00:01:00 is defined for this process, which means it will run once every minute.

Note: The DBCM and PRNT blocks, both operation blocks, must be placed in a process.

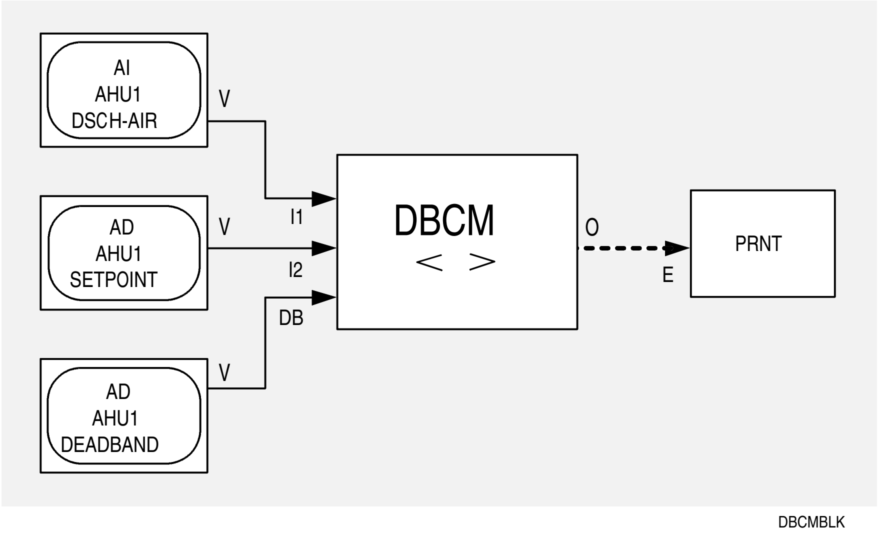


Figure 45: DBCM Block Example

DFCM (Differential Compare) Block

Category

Control

Purpose

Compares two analog inputs. It outputs either a True or False state, depending on the relational operator and the applied differential.

Details

The relational operators for this block are the following:

Less Than (Input 1 < Input 2)
Greater Than (Input 1 > Input 2)
Equal To (Input 1 = Input 2)
Less Than or Equal To (Input 1 ≤ Input 2)
Greater Than or Equal To (Input 1 ≥ Input 2)

The operator determines the behavior of the block as it applies to the differential. The differential is the range within which Input 1 can vary about Input 2 without initiating any change in the output. The differential range can be on either side of Input 2, or on both sides, depending on the operator selected.

Note: When a process containing a DFCM block executes for the first time, the differential of the DFCM block is not applied, and the block acts like a simple COMP block. First-time execution occurs when the NCM is rebooted or downloaded, or the process containing the DFCM block is downloaded or enabled.

Figures 46 to 50 illustrate the behaviors of the DFCM block according to the various relational operators.

Note: If the Equal To operator is used with real inputs, the output will almost always be False, since it is very unlikely that two real values will equal. This is especially true if a floating point operation was done prior to the compare. Similarly, if the Not Equal To (<>) operator is used with real inputs, the output will almost always be True.

Figure 46 shows the function of the DFCM block for the Less Than operator. The output is True when Input 1 < Input 2. As Input 1 increases, the output remains True through the differential. The output changes to False when Input 1 ≥ Input 2 + Differential.

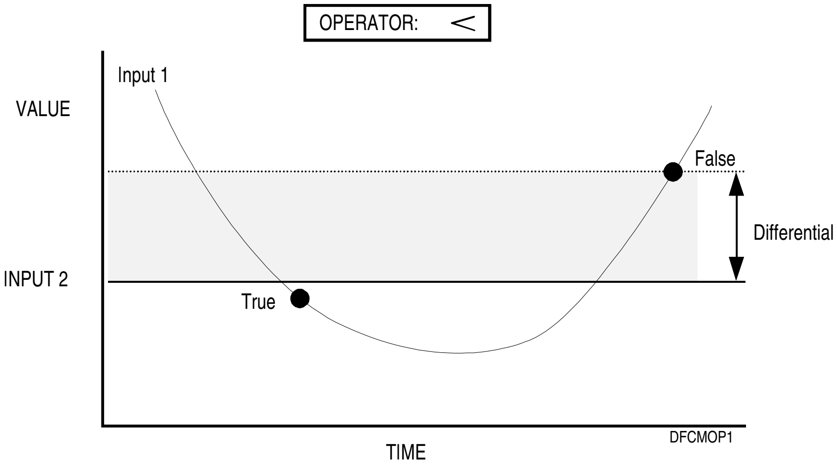


Figure 46: DFCM Operation: Less Than

Figure 47 shows the function of the DFCM block for the Less Than or Equal To operator. The output is True when Input 1 ≤ Input 2. As Input 1 increases, the output remains True through the differential. The output changes to False when Input 1 > Input 2 + Differential.

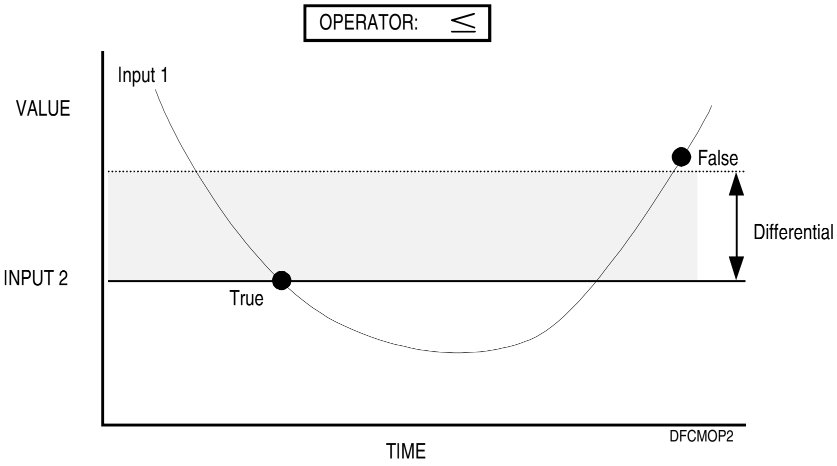


Figure 47: DFCM Operation: Less Than or Equal To

Figure 48 shows the function of the DFCM block for the Greater Than operator. The output is True when Input 1 > Input 2. As Input 1 decreases, the output remains True through the differential. The output changes to False when Input 1 ≤ Input 2 - Differential.

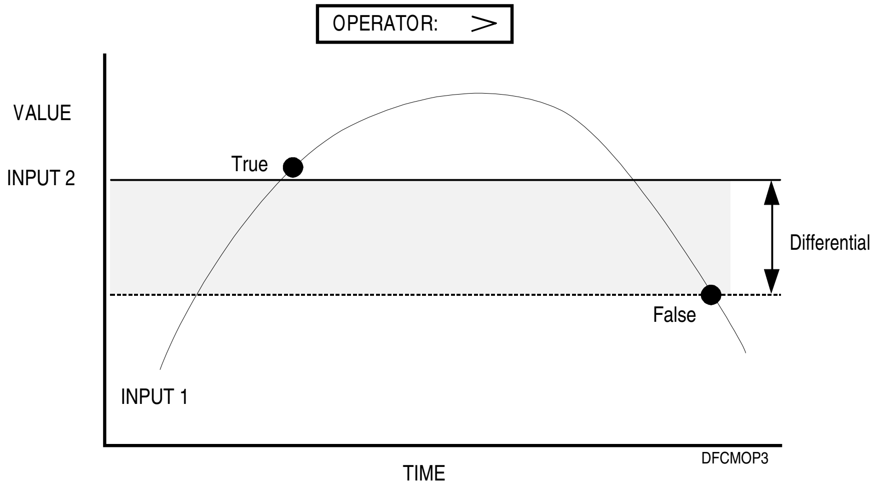


Figure 48: DFCM Operation: Greater Than

Figure 49 shows the function of the DFCM block for the Greater Than or Equal To operator. The output is True when Input 1 ≥ Input 2. As Input 1 decreases, the output remains True through the differential. The output changes to False when Input 1 < Input 2 - Differential.

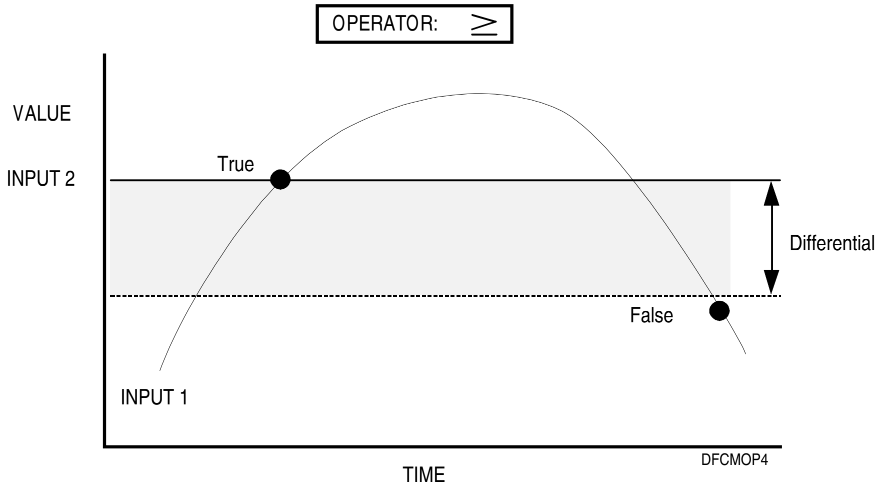


Figure 49: DFCM Operation: Greater Than or Equal To

Figure 50 shows the function of the DFCM block for the Equal To operator. For this operator, the differential is applied on both sides of Input 2. As Input 1 increases, the output remains False through the differential. The output changes to True when Input 1 = Input 2. The output returns to False near the point when Input 1 > Input 2 + Differential. It does not turn to False at an exact point because of floating point inaccuracies. As Input 1 decreases, the output remains False through the differential, until again Input 1 = Input 2, at which point it changes to True. The output returns to False at the exact point when Input 1 < Input 2 - Differential.

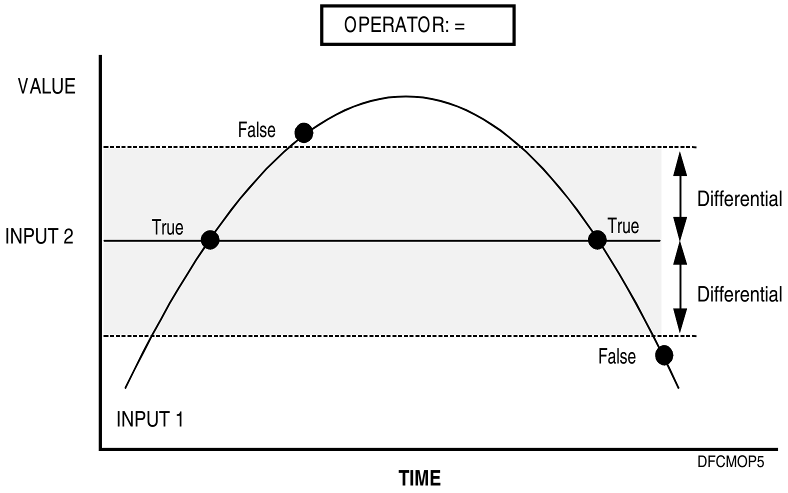


Figure 50: DFCM Operation: Equal To

You can configure this block to compare analog inputs only. Also, you can specify a constant value for an input instead of connecting an external input. However, you cannot connect the differential as an input.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Operation		TAB	<	<,>, =, ≤, ≥		X			
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
Differential	DIFF	ANA	0.000000	Real ≥ 0	N	X	X		DF
	OUTPUT	BIN	0	0,1	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if Input 1, Input 2, or the Differential input is unreliable. The output will be unreliable if the Differential input is less than 0.0. If it is, three things will occur: a user advisory will be sent to the Operator Workstation, the DFCM block will function like a COMP block, and the output of the block will be flagged unreliable.

Example

In this example (Figure 51), the DFCM block is configured for Less Than (<) operation. It is used to compare outdoor air temperature (AHU1\OA-TEMP) with an adjustable setpoint (AHU1\ECON-SPT) in order to determine economizer mode (ECON). If the outdoor air temperature (I1) is less than the setpoint (I2), economizer is set. If the outdoor air temperature is greater than or equal to the setpoint, plus a differential of 2.0°F, economizer is not set.

A period of 00:05:00 is defined for this process, which means it will execute once every five minutes.

Note: The DFCM and SVAR blocks, both operation blocks, must be placed in a process.

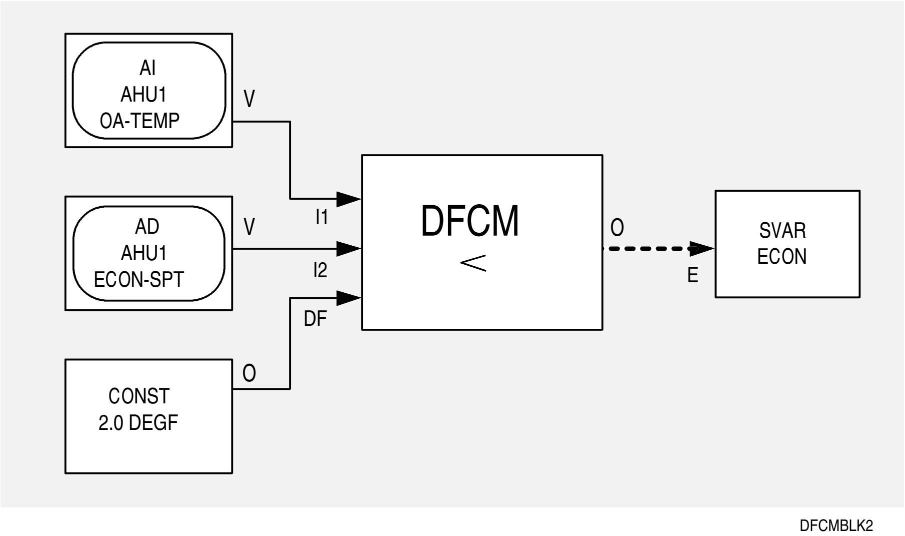


Figure 51: DFCM Block Example

DIV (Divide) Block

Category

Math

Purpose

Divides Input 1 by Input 2 using the following equation:

Input 1/Input 2 = Output

Details

The inputs into this block can only be analog. If Input 2 is zero (0.0), a divide-by-zero runtime error occurs and a user advisory is issued. If this happens, the result of the calculation will be:

3.4×10^{38} (if Input 1 ≥ 0.0), or
-3.4×10^{38} (if Input 1 < 0.0)

Note: The operator terminals cannot fully display the above numbers, since they are capable of displaying values of up to eight characters.

You can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Input 2	INPUT 2	ANA	1.000000	Real	N	X	X		I2
	OUTPUT	ANA	0.000000	Real	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

The output of the DIV block is unreliable if either of the inputs is unreliable or Input 2 is 0.0.

Example

In this example (Figure 52), the DIV block is part of a tonnage calculation. The standard tonnage equation is:

TONS = ΔT (GPM) / 24

where:

ΔT = Delta T, change in chilled water return temperature and chilled water supply temperature.

GPM = gallons per minute, a measured value.

To convert this equation into a GPL diagram, three math blocks are required:

SUB:	Subtracts two chilled water temperatures to find ΔT.
DIV:	Divides GPM by 24.
MUL:	Multiplies ΔT by GPM/24.

The example of Figure 52 performs three calculations:

- GPM is divided by 24. The GPM figure is obtained from an AI object that measures flow (AHU1\FLOW). This is Input 1 of the DIV block. Input 2 of the DIV block is 24, from a CNST block.
- ΔT is calculated. This value is obtained from two AI objects, one reporting chilled water return temperature (AHU1\CHWR-T), and the other chilled water supply temperature (AHU1\CHWS-T). Input 1 of the SUB block is AHU1\CHWR-T, and Input 2 is AHU1\CHWS-T.
- The result of the DIV and SUB blocks is multiplied. Input 1 of the MUL block is from the DIV block, and Input 2 is from the SUB block.

A command block reads the calculated value, and sends it to an AD object (AHU1\TONS), whose value can be read on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will execute once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: DIV, MUL, SUB, and CMD.

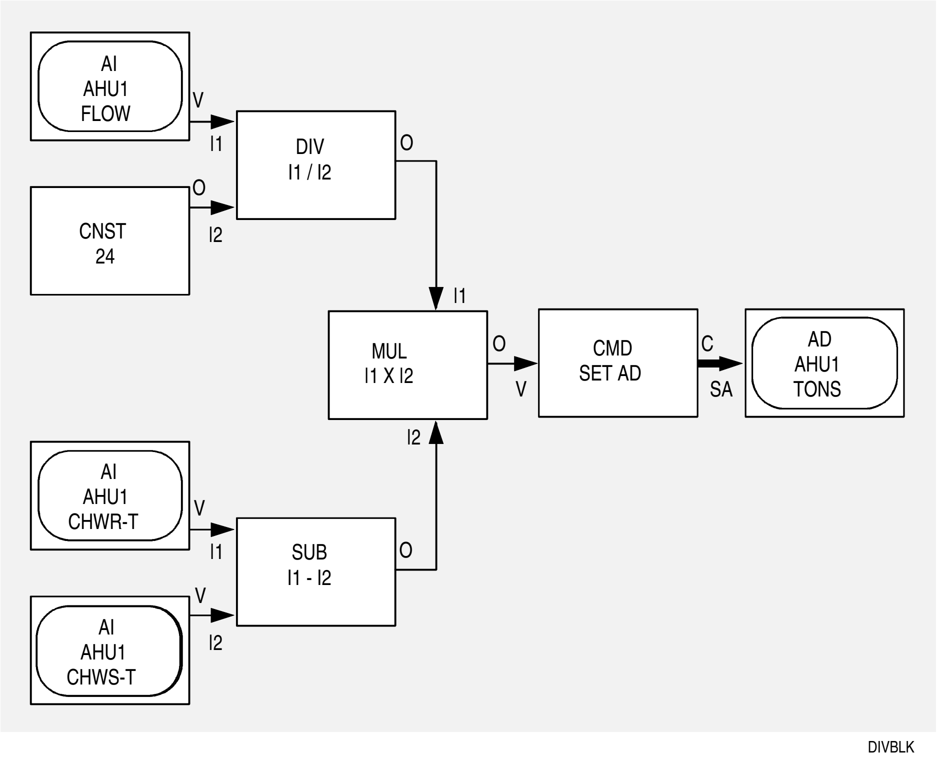


Figure 52: DIV Block Example

DLAY (Delay) Block

Category

Logic

Purpose

Delays the change of binary data to True for a period of time.

Details

This block can be configured as a Cancelable Delay or One-shot Delay. The Cancelable type delays binary change of data to True for a specified period of time, whereas the One-shot type delays the changing of data for one execution of the process.

Cancelable DLAY

The output of this type varies as shown in Figure 53 and according to the following table:

Cancelable Delay	
Change of Input	Output
False --> True	False for delay time specified, then True
True --> False	False

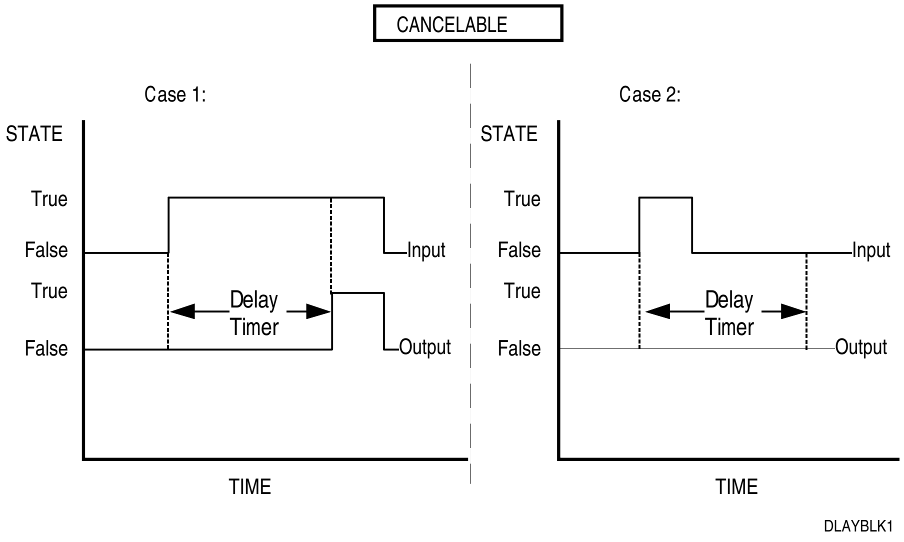


Figure 53: Cancelable DLAY Block

When the delay time for the Cancelable DLAY block has expired, the process that contains this block will be triggered. If the input returns to False before the timer expires, the output will stay False and the process will not be triggered.

Note: We recommend that you do not specify a time delay of 00:00:00 for a Cancelable Delay because it will require more processing time and memory. The DLAY block will always act as a One-shot delay if you do so. If you want the delay to be a One-shot, configure it as such in the Type field of the template.

For initialization applications, a Cancelable DLAY block can accept a constant of True as its input. In this case, the output varies as follows:

Cancelable Delay	
Constant Input	Output
True	False for first execution and until delay timer expires, then True for each subsequent execution until the NCM is warm started or the process is re-downloaded

One-Shot DLAY

The output of a One-shot DLAY block varies as shown in Figure 54 and according to this table:

One-shot Delay	
Change of Input	Output
False --> True	False for one execution of the process, then True
True --> False	False

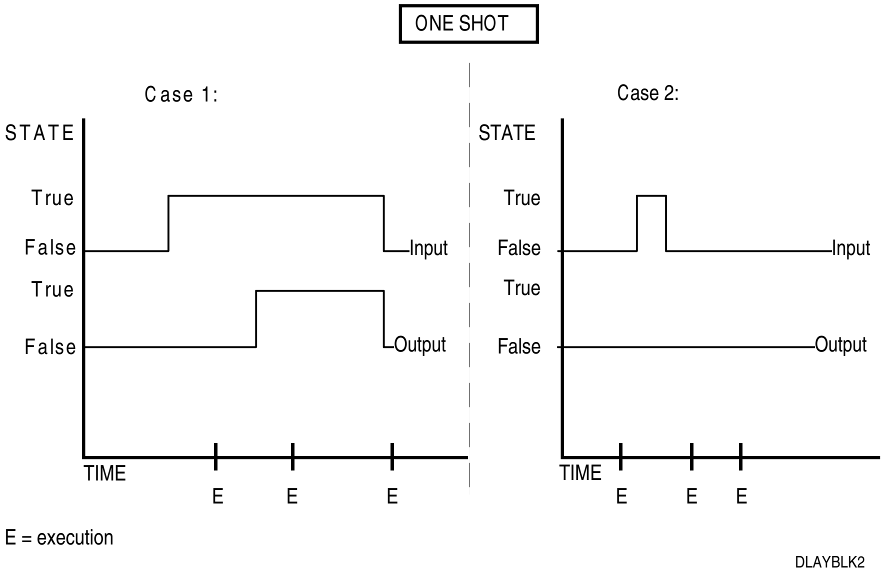


Figure 54: One-Shot DLAY Block

If the input of the One-shot DLAY block changes back to False before the next execution of the process, the output will remain False.

Note: The One-shot DLAY block will not trigger a process.

For initialization applications, a One-shot DLAY block can accept a constant of True as its input. In this case, the output varies as follows:

One-shot Delay	
CONSTANT INPUT	OUTPUT
True	False for first execution, then True for each subsequent execution

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	CANCEL	CANCEL, ONE-SHOT		X			
	INPUT	BIN		0,1	Y		X		I
Time	TIME IN	TIM/ POP-UP	00:00:05	Time > 00:00:00	N	X	X		TI
	OUTPUT	BIN	0	0,1	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

The reliability of the DLAY block depends on its configuration: Cancelable or One-shot.

Cancelable Delay

Input and Timer (If...)	Output (Then...)
If input changes from False to True, and input and timer are reliable. Or, if input changes from True to False and input is reliable.	**Reliable**
If input changes from False to True, and input and/or timer is unreliable. Or, if input changes from True to False, the input is unreliable, and the timer has not expired.	**Unreliable**
If input changes from True to False and input is unreliable. Input does not change, regardless of its reliability.	**No Change In Reliability**

One-Shot Delay

The output is unreliable if the input is unreliable.

Example

The DLAY block in the following example (Figure 55) is configured as a Cancelable delay. It delays a command to an exhaust fan (AHU1\EX-FAN1). The CMD block will command the exhaust fan to Start one minute (00:01:00) after the supply fan (AHU1\SFAN1-ST) starts.

If the AHU1\SFAN1-ST stops before the one minute delay timer expires, the start command is not sent.

A period of 00:00:00 is defined for this process, since its execution is dictated by the Cancelable operation of the block.

Note: The DLAY and CMD blocks, both operation blocks, must be placed in a process.

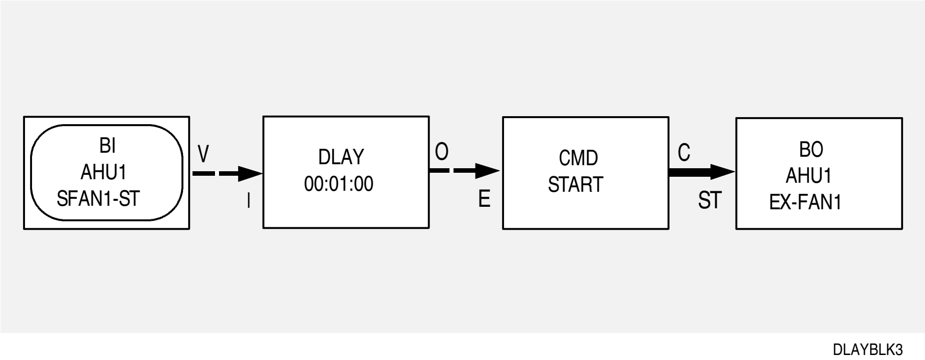


Figure 55: DLAY Block Example

DWPT (Dew Point) Block

Category

Psychrometric Equations

Purpose

Calculates dew point temperature based on dry bulb temperature and relative humidity. This block uses the ASHRAE formula for calculating dew point.

Details

The dew point calculation can be done in either English or Metric units.

Valid ranges for the calculated dew point temperature are:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Relative Humidity: 0 to 100%

Note: GPL converts relative humidity to an integer by rounding (e.g., 10 to 10.49999 = 10; 10.50000 to 10.99999 = 11).

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH METRIC		X			
	DRY BULB	ANA		Real	Y		X		DB
	REL HUMID	ANA		Real	Y		X		RH
	OUTPUT	ANA	0.000000	Real	Y			X	O
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Reliability

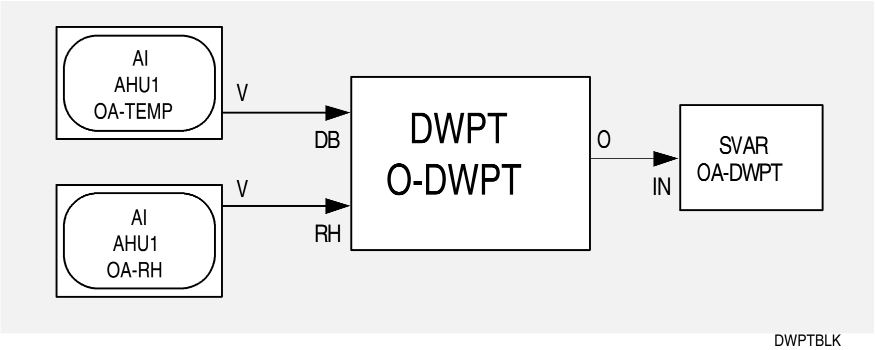
The output of this block is unreliable if the dry bulb temperature input or relative humidity input is unreliable. Also, if either of these values is less than the lowest part of their range, enthalpy is calculated using the lowest range value (e.g., for dry bulb temperature, -147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if either of these values is greater than the highest part of their range, enthalpy is calculated using the highest range value (e.g., for dry bulb temperature, -391.9°F), the output is unreliable, and a runtime error message is generated.

Example

In this example (Figure 56), the DWPT block calculates the dew point temperature of outside air. The block takes as inputs the outside air temperature (AHU1\OA-TEMP) and the outside air relative humidity (AHU1\OA-RH). These values are entered in the dew point algorithm, whose result is assigned to a SVAR block (AHU1\OA-DWPT).

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: The DWPT and SVAR blocks, both operation blocks, must be placed in a process.



DWPTBLK

Figure 56: DWPT Block Example

ENDP (Enthalpy Dew Point) Block

Category

Psychrometric Equations

Purpose

Calculates enthalpy based on dry bulb temperature, dew point temperature, and barometric pressure. This block uses the ASHRAE formula for calculating enthalpy.

Details

The enthalpy calculation can be done in either English or Metric units.

Valid ranges for calculated Enthalpy are:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Dew Point Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Barometric Pressure: 15.00 to 32.00 in. Hg (0.51 to 1.08 bar)

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH METRIC		X			
Barometric Press.	BARO PRES	ANA	29.00000	Real	N	X	X		BP
-	DRY BULB	ANA		Real	Y		X		DB
-	DEW POINT	ANA		Real	Y		X		DP
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the dry bulb temperature input, dew point temperature input, or barometric pressure input is unreliable. Also, if any of these values is less than the lowest part of their range, enthalpy is calculated using the lowest range value (e.g., for dry bulb temperature, -147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if any of these values is greater than the highest part of their range, enthalpy is calculated using the highest range value (e.g., for dry bulb temperature, -391.9°F), the output is unreliable, and a runtime error message is generated.

A runtime error message is also generated if dew point temperature is greater than dry bulb temperature.

Example

The following example (Figure 57) contains an ENDP block that calculates the enthalpy of outside air. The block takes as inputs the outside air temperature (AHU1\OA-TEMP) and the dew point of outside air (AHU1\OA-DWPT). These values are entered into the enthalpy algorithm. The CMD block reads the result, and commands an AD object (AHU1\OA-ENDP) to this value.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: The ENDP and CMD blocks, both operation blocks, must be placed in a process.

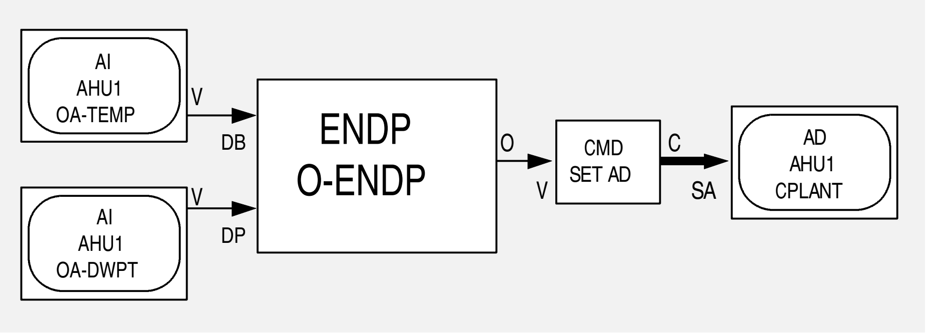


Figure 57: ENDP Block Example

ENRH (Enthalpy Relative Humidity) Block

Category

Psychrometric Equations

Purpose

Calculates enthalpy based on dry bulb temperature, relative humidity, and barometric pressure. This block uses the ASHRAE formula for calculating enthalpy.

Details

The enthalpy calculation can be done in either English or Metric units.

Valid ranges for calculated Enthalpy are:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Relative Humidity: 0 to 100%

Barometric Pressure: 15.00 to 32.00 in. Hg (0.51 to 1.08 bar)

Note: GPL converts relative humidity to integer by rounding (e.g., 10.00000 to 10.49999 = 10; 10.50000 to 10.99999 = 11).

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH METRIC		X			
Barometric Press.	BARO PRES	ANA	29.00000	Real	N	X	X		BP
-	DRY BULB	ANA		Real	Y		X		DB
-	REL HUMID	ANA		Real	Y		X		RH
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the dry bulb temperature input, relative humidity input, or barometric pressure input is unreliable. Also, if any of these values is less than the lowest part of their range, enthalpy is calculated using the lowest range value (e.g., for dry bulb temperature, -147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if any of these values is greater than the highest part of their range, enthalpy is calculated using the highest range value (e.g., for dry bulb temperature, -391.9°F), the output is unreliable, and a runtime error message is generated.

Note: The output of this block is also unreliable if the internally calculated dew point value is outside the range of -80.0 to 150.0°F (-60.0 to 70.0°C). (Internal dew point value is not provided to the programmer.)

Example

The following example (Figure 58) contains an ENRH block that calculates the enthalpy of outside air. The block takes as its inputs the outside air temperature (AHU1\OA-TEMP) and the relative humidity of outside air (AHU1\OA-RH). These values are entered into the enthalpy algorithm. The CMD block reads the result, and commands an AD object (AHU1\OA-ENRH) to this value.

A period of 00:05:00 is defined for this process, which means it will run once every two minutes.

Note: The ENRH and CMD blocks, both operation blocks, must be placed in a process.

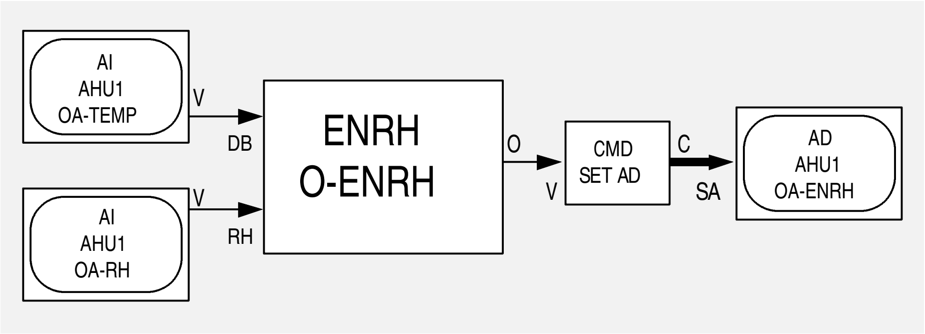


Figure 58: ENRH Block Example

EQN (Equation) Block

Category

Math

Purpose

Creates a user-defined equation.

Details

The equation can use up to four real inputs, four constants, and a variety of math functions. The available functions include the following:

+	addition	**SIN**	sine
-	subtraction	**COS**	cosine
*****	multiplication	**TAN**	tangent
/	divide	**ABS**	absolute value
^	exponentiation	**MIN**	minimum value
SQR	square root	**MAX**	maximum value
LOG	natural log	**AVG**	average
PI	PI		

You must follow this nomenclature:

I1, I2, I3, I4 for Inputs
C1, C2, C3, C4 for Constants

SQR(e), LOG(e), SIN(e),COS(e),TAN(e),ABS(e)

MIN(I1,I2,I3,...,I12)
MAX(I1,I2,I3,...,I12)
AVG(I1,I2,I3,...,I12)

PI

where e = numeric expression

Note: The angle for the trigonometric functions is assumed to be radians.

You may use the inputs, constants, and operations any number of times, as long as the total expression line is less than or equal to 50 characters. Also, you may use up to 20 sets of nested parentheses. Use parentheses to indicate the order of math evaluation.

Expressions in the EQN block are evaluated according to operator precedence. The following table shows the order of execution of the operators, where 1 is the highest and the first executed. In the case of equal rank, the operators are executed from left to right. The evaluation sequence is modified by parentheses, which have precedence over any operator.

Operator	Definition	Rank	Type
()	Group Operations	0	Any
^	Exponential	1	Arithmetic
-	Unary Minus	2	Arithmetic
+	Unary Plus	2	Arithmetic
***,/**	Multiply, Divide	3	Arithmetic
+,-	Add, Subtract	4	Arithmetic, Time
DIFF	Differential	5	Arithmetic
>	Greater Than	6	Relational
<	Less Than	6	Relational
≥	Greater or Equal to	6	Relational
≤	Less or Equal to	6	Relational
=	Equal to	6	Relational
<>	Not Equal	6	Relational
NOT	Negation	7	Logical
AND	Conjunction	8	Logical
OR	Disjunction	9	Logical
XOR	Exclusive OR	9	Logical

You can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

The Editor checks the equation you enter and outputs an error message if there is something wrong with it. A caret (^) symbol below the equation indicates the approximate location of the error. The possible error messages are:

```
Error ! Illegal Operator
```

The operator entered is not one of the following: +, -, *, /, ^.

```
Error ! Invalid identifier
```

An invalid identifier is entered. See the note below for all valid characters and symbols allowed.

```
Error ! Missing expression/operand
```

The function is missing an expression or operand.

```
Error ! Missing Operator
```

A required operator is missing.

```
Error ! Missing left parenthesis
```

A required left parenthesis is missing.

```
Error ! Missing right parenthesis
```

A required right parenthesis is missing.

```
Error ! No expression
```

The right parenthesis has no matching left parenthesis.

```
Error ! Too many parameters
```

The function has more than 12 parameters defined.

Note: You must enter an equation in the Equation (Output) field, or the Compiler will output an error. Also, all inputs that you connect to the EQN block must be represented in the equation. Otherwise, the Compiler will output an error.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
INPUT 1 (I1)	INPUT 1	ANA	0.000000	Real	N	X	X		I1
INPUT 2 (I2)	INPUT 2	ANA	0.000000	Real	N	X	X		I2
INPUT 3 (I3)	INPUT 3	ANA	0.000000	Real	N	X	X		I3
INPUT 4 (I4)	INPUT 4	ANA	0.000000	Real	N	X	X		I4
CONSTANT 1 (C1)		ANA	1.000000	Real		X			
CONSTANT 2 (C2)		ANA	1.000000	Real		X			
CONSTANT 3 (C3)		ANA	1.000000	Real		X			
CONSTANT 4 (C4)		ANA	1.000000	Real		X			
EQUATION (OUTPUT)		STR	Blank	50 Chars See Note		X			
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: Only use the following characters and symbols in the equation: I1-I4, C1-C4, +, -, *, /, (,), SQR, ^, LOG, SIN, COS, TAN, ABS, MIN, MAX, AVG, and PI.

Reliability

The output is unreliable if any of the inputs are unreliable. For example, divide-by-zero, math underflow, or overflow will cause unreliability.

Example

The example shown in Figure 59 uses an EQN block to calculate BTUs (British Thermal Units). The standard BTU equation is:

BTUs = GPM (ΔT) 500

where:

GPM = Gallons Per Minute

ΔT = Delta T, change of two temperatures

Applying the BTU equation in GPL requires four inputs and an output. The equation defined in the EQN template is:

Equation (Output) = I1 * (I2 - I3) * I4

where:

I1 = GPM
I2 = Temperature of chilled water return
I3 = Temperature of chilled water supply
I4 = 500 (Constant)

The value for Input 1 is from an AI object (AHU1\FLOW1) that measures water flow. Input 2 is from an AI object that measures the temperature of return water (AHU1\CHWR-T), and Input 3 from an AI object that measures supply water temperature (AHU1\CHWS-T). Input 4 is a constant, 500.0. The CMD block reads the output of the EQN block. The CMD block then commands an AD object (AHU\BTUs) to the value, which can be read on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: The EQN and CMD blocks, both operation blocks, must be placed in a process.

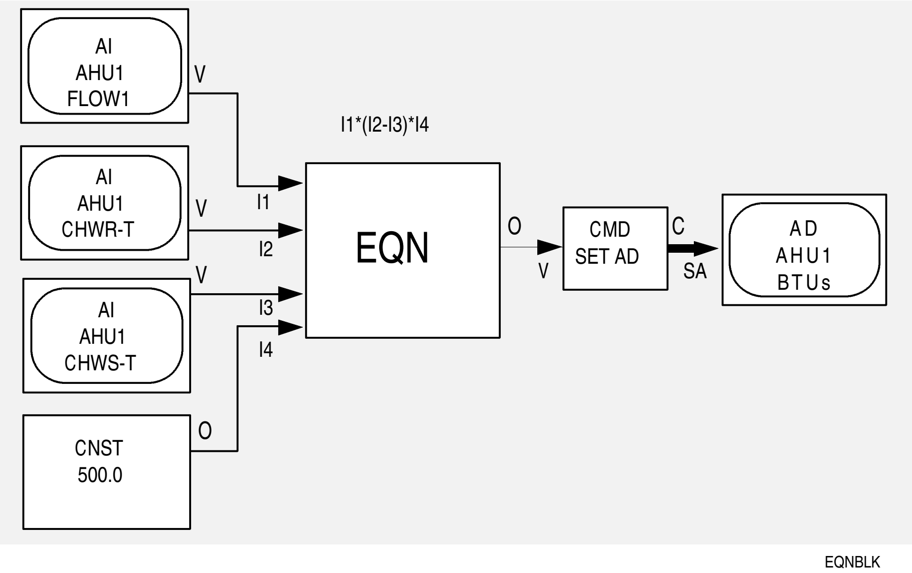


Figure 59: EQN Block Example

FILE Block

Category

Miscellaneous

Purpose

Links the files of multiple control strategies into one “master” file. You can access the external control strategy by double-clicking left on its FILE block.

Details

The primary purpose of the FILE block is to divide a large control strategy into smaller strategies. Another purpose is to create one global diagram for an NCM that contains a FILE block for each of the strategies in the NCM. If you translate a GPL strategy file that includes FILE blocks, the files referenced by the FILE blocks will also be translated.

When you translate a file that contains one or more FILE blocks, only one list file (.LST) is created (not one for each FILE block).

You may place FILE blocks inside of compounds. You may not connect blocks that are in different file blocks.

The name you assign to the FILE block must not be one of the reserved words or contain invalid symbols. (Refer to *Appendix F: Characters, Symbols, and Reserved Words*.) Also, the file name must match the file name of the control strategy it is to reference. For example, if the file name of the control strategy is AHU1, you must enter AHU1 in the FILE block’s template. Also, the file must be in the same directory. For example, if the full file name of the strategy is “C:\JOB123\AHU1”, the FILE block must be stored under the directory “C:\JOB123”. You do not have to specify a DOS extension in the file name.

You may have up to 30 nested FILE blocks in one control strategy file. Each nested file must be in the same directory.

After the FILE block is defined, double-click left on it to access the external control strategy. To return to the previous strategy, click left on the File Block icon (up arrow) that appears in the upper right corner of the work area.

The FILE block is different from operation blocks in that you cannot connect two FILE blocks or simulate a FILE block. However, you can simulate the contents of the FILE block.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
File Name		STR	Blank	8 Char		X			

Reliability

This block has no associated reliability.

Example

The example in Figure 60 shows four FILE blocks. Each FILE block represents control strategies for four different air handlers: AHU1, AHU2, AHU3, and AHU4. The control strategies can be accessed by double-clicking on one of the FILE blocks. (Query mode must be active.)

This example has no period defined since FILE blocks need not be placed in processes.

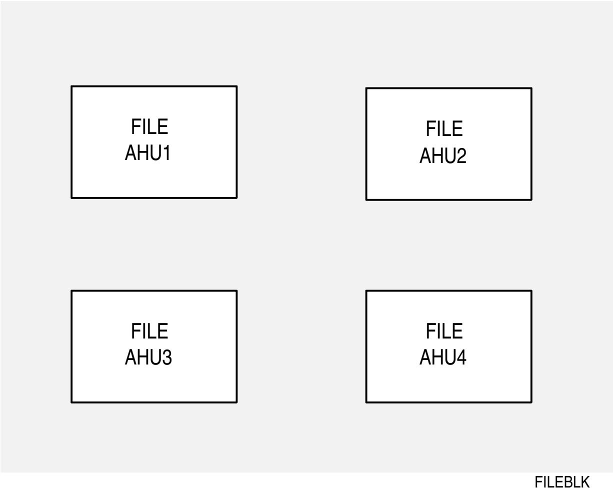


Figure 60: FILE Block Example

FILT (Filter) Block

Category

Calculations

Purpose

Smooths out a change in value by taking a percentage of change and adding to the previous value.

Note: This is a first-order exponential filter.

Details

The percentage the FILT block uses is based on the filter weight, a user-specified parameter. Figure 61 shows the effect of filter weight on an input.

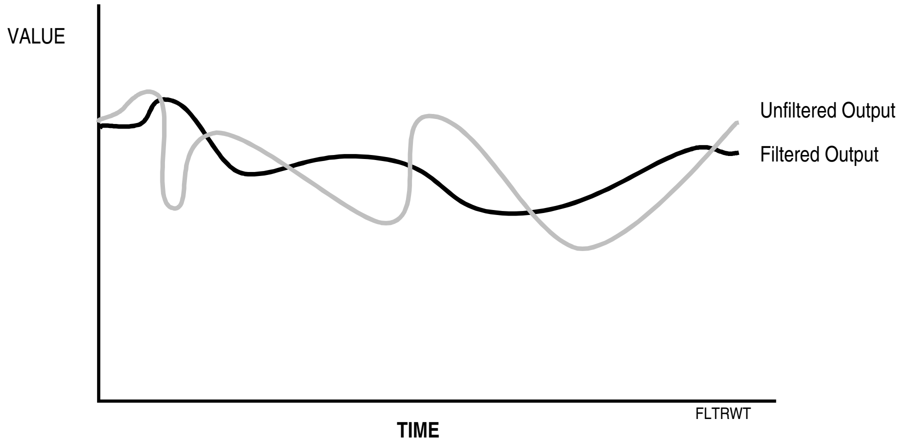


Figure 61: Effect of a Filter Weight

This block has a Reset input, which determines whether the output will be filtered. When the Reset input is not externally connected, its value is always false and reliable, and the input is always filtered. However, when the Reset input is externally connected and reliable, the output is as follows:

Reset Input	Output
True	Equals input value (not filtered).
False	Filtering occurs.

When the process that contains the FILT block runs for the first time, the output depends on the Reset input. The following chart illustrates:

Reset Input	First Output
Not Connected	Input/Filter Weight
True and Reliable	Input
False and Reliable	Input/Filter Weight
True or False and Unreliable	Input

For second and subsequent executions, the algorithm uses the previous output to calculate a new output.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Filter Weight	FILTER	ANA	1.000000	Real ≥ 1.0	N	X	X		F W
-	RESET	BIN	0	0,1	N		X		R
-	INPUT	ANA		Real	Y		X		IN
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the value of the input or filter weight is unreliable. If the Reset input is unreliable, the output will be the value of the input, flagged unreliable.

Example

This example (Figure 62) filters accumulated tons, whose values vary widely. Actual tons (ACT_TONS) are stored in a SVAR block and is input into the FILT block, which applies a filter of 20 (20.0). The CMD block then reads the result, which in turn sends the filtered tons value to an AD object (AHU1\FILT-TONS) for reading on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. These include: FILT, SVAR, and CMD.

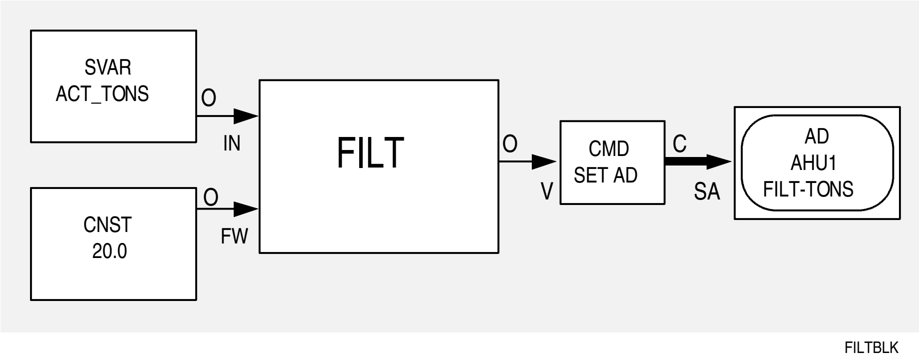


Figure 62: FILT Block Example

FREL (Force Reliable) Block

Category

Reliability

Purpose

Forces an output to be reliable. The output value is the same as the input value. The output is *always* reliable, regardless of the reliability of the input.

The block can be configured for binary, analog, or time values.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 CHAR		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
-	INPUT	ANA		Real	Y		X		IN
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	INPUT	BIN		0,1	Y		X		I
-	OUTPUT	BIN	0	0,1	Y			X	O
-	TIME IN	TIM		TIME	Y		X		TI
-	OUTPUT	TIM	00:00:00	TIME	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the FREL block is always reliable, regardless of the reliability of the input. Note that if any unreliable input comes into the process containing the FREL block, the process itself is still unreliable.

Example

In the example shown in Figure 63, the GPL process reads the Value attribute of the AI object. If the Value attribute is reliable, the Force Reliable block passes that value to its output. The ADD block adds 5.0 to the value, which the SVAR block then reads.

If the GPL process detects that the Value attribute of the AI object is unreliable, it uses the last known reliable value and flags it unreliable. The FREL block takes this value and flags it reliable, and the process continues as described above.

Note: All operation blocks in this example must be placed in a process. They include: FREL, ADD, CNST, and SVAR.

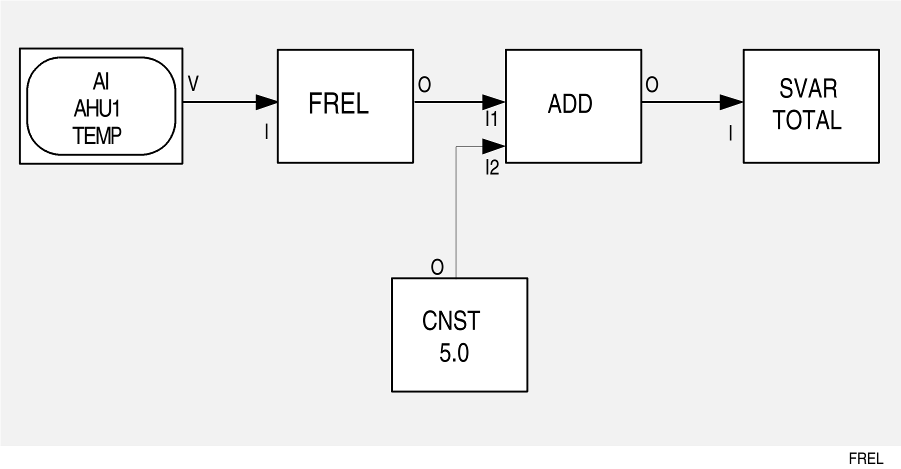


Figure 63: FREL Block Example

HSEL (High Select) Block

Category

Selectors

Purpose

Selects the highest value of two, three, or four real inputs.

Details

You can configure this block to accept two, three, or four inputs. The default is two. If you want to select between three or four inputs, you must first specify the number of inputs in the template before making the connection.

Also, you can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
Number of Inputs		INT	2	2,3,4		X			
-	INPUT 3	ANA		Real	N*		X		I3
-	INPUT 4	ANA		Real	N*		X		I4
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

* Input 3 is a required connection if you set the Number of Inputs field to 3.

Input 4 is a required connection if you set the Inputs field to 4.

Reliability

The output of this block is unreliable if any of the inputs are unreliable.

Example

In the example of Figure 64, the HSEL block chooses the highest of three multi-zone temperatures. The HSEL block takes as inputs temperatures as reported by three AI objects: MZU1\ZN-T1, MZU1\ZN-T2, and MZU1\ZN-T3. The CMD block then reads the highest of these values as selected by the HSEL, and sends the value to an AD object (MZU1\HI-ZN) for reading on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: The HSEL and CMD blocks, both operation blocks, must be placed in a process.

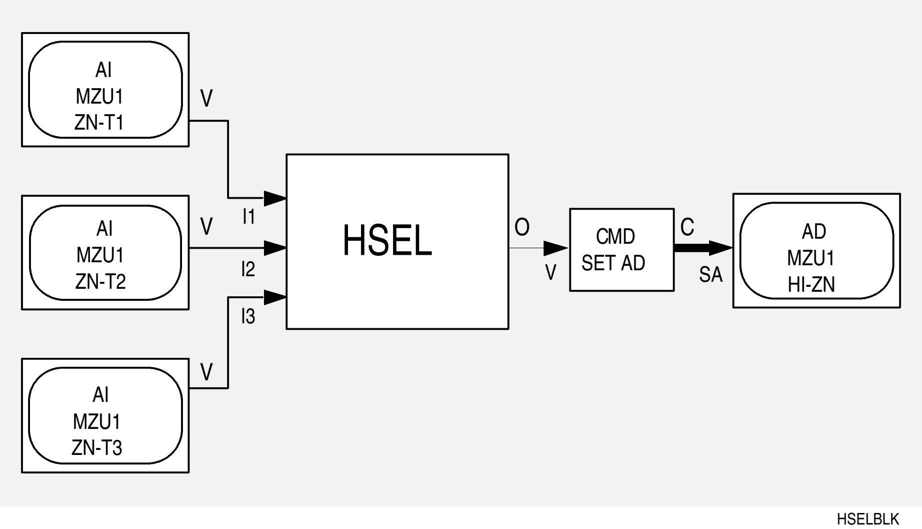


Figure 64: HSEL Block Example

LSEL (Low Selector) Block

Category

Selectors

Purpose

Selects the lowest value of two, three, or four inputs.

Details

You can configure this block to accept two, three, or four inputs. The default is two. If you want to select between three or four inputs, you must first specify the number of inputs in the template before making the connection.

Also, you can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
Number of Inputs		INT	2	2, 3, 4		X			
-	INPUT 3	ANA		Real	N*		X		I3
-	INPUT 4	ANA		Real	N*		X		I4
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

* Input 3 is a required connection if you set the Number of Inputs field to 3. Input 4 is a required connection if you set this field to 4.

Reliability

The output of this block is unreliable if either of the inputs is unreliable.

Example

In the example of Figure 65, the LSEL block chooses the lowest of four multi-zone temperatures to obtain a heating setpoint. The LSEL block takes as inputs temperatures as reported by four AI objects: MZU1\ZN-T1, MZU1\ZN-T2, MZU1\ZN-T3, and MZU1\ZN-T4. The CMD block then reads the lowest of these values as selected by the LSEL block, and sends the value to an AD object (MZU1\HT-STPNT) for reading on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: The LSEL and CMD blocks, both operation blocks, must be placed in a process.

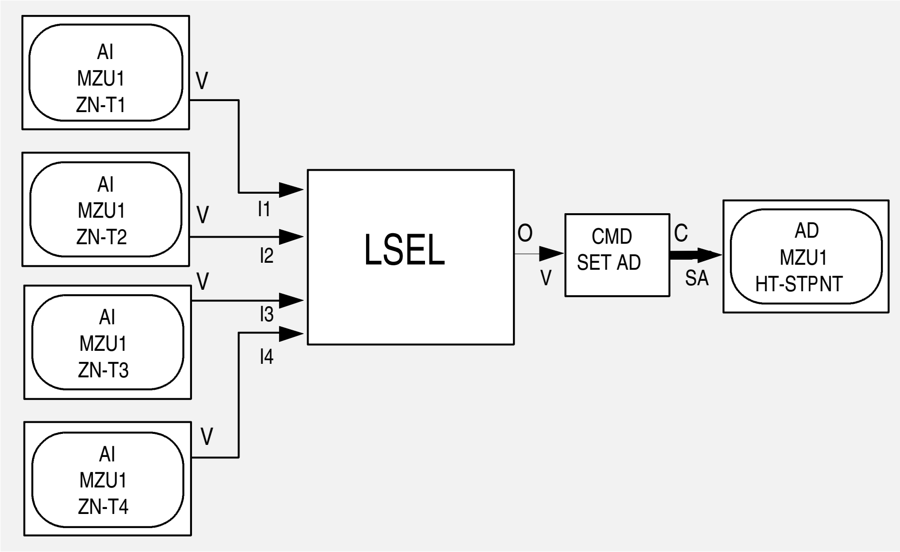


Figure 65: LSEL Block Example

LTCH (Latch) Block

Category

Logic

Purpose

Remembers the binary state that was last True, and passes this information to its output.

Details

Figure 66, and the tables that follow, help illustrate the Latch function.

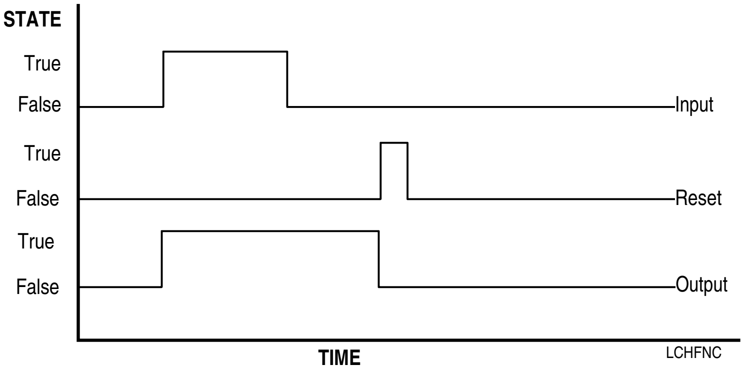


Figure 66: Latch Function

Truth Table (when reliable):

Input	Reset	Output
True	False	True
False	True	False
True	True	Last Output
False	False	Last Output

Transition Tables (when reliable):

Change Of Input	Reset	Output
True --> False	False	Last Output
False --> True	False	True (latched)
True --> False	True	False (latched)
False --> True	True	Last Output

Input	Change Of Reset	Output
True	True --> False	True (latched)
True	False --> True	Last Output
False	True --> False	Last Output
False	False --> True	False (latched)

Note: The initial condition of the output is False.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	INPUT	BIN		0,1	Y		X		I
-	RESET	BIN		0,1	Y		X		R
-	OUTPUT	BIN	0	0,1	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

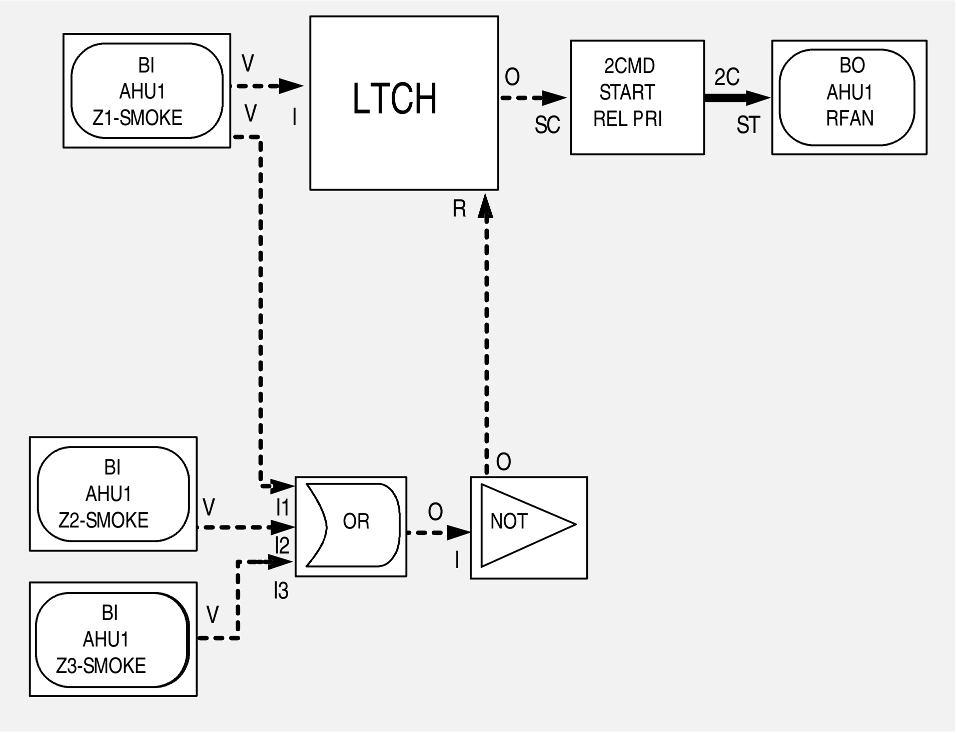
The output of this block is unreliable if the input or Reset input is unreliable. If either input is unreliable, the output value will be the previous value of the output, and will be unreliable.

Example

The example in Figure 67 contains a LTCH block that provides an interlock function for a return fan (AHU1\RFAN), based on the status of three zone smoke detectors (AHU1\Z1-SMOKE, Z2-SMOKE, Z3-SMOKE). When AHU1\Z1-SMOKE reports smoke, the 2CMD block issues a command to start the return fan. When all the smoke detectors return to normal, the 2CMD block sends a REL_PRI (release priority) command to the return fan.

A period of 00:00:00 is defined for this process since it needs to run only when a triggerable attribute of one of the BI objects changes.

Note: All operation blocks in this example must be placed in a process. They include: LTCH, OR, NOT, and 2CMD.



LTCHBLK

Figure 67: LTCH Block Example

MSEL (Mode Selector) Block

Category

Selectors

Purpose

Selects an input based on the analog Mode Input, and transfers the input's value to the output. The MSEL block is basically an analog multistate switch.

Details

You can configure the MSEL block as follows:

- To accept analog, binary, or time data--The default is analog. If you want to use binary or time data, you must specify the type of data in the template before making the connection.
- To accept two to eight modes--The default is two modes. If you want to select between three to eight modes, you must specify the number of modes in the template.
- To use a constant value as the input instead of an external input--Enter a constant in the data base template of the block.

This block has an input called Mode Input. The Mode Input provides two functions:

- Decides which input to send to the output.
- Decides which mode to select.

Since GPL requires real numbers, not integers, the Mode Input uses analog numbers in the range of 0.50 to 8.49. The MSEL block converts a value in this range to an integer by rounding (e.g., 0.50 to 1.49 = 1.0; 1.50 to 1.99 = 2.0). For example, if the Mode Input is 1.3, the MSEL block will convert it to 1.0. The following chart details the action of the Mode Input.

Mode Input	Mode Selected	Output
0.50 - 1.49	1	Input 1
1.50 - 2.49	2	Input 2
2.50 - 3.49	3	Input 3
3.50 - 4.49	4	Input 4
4.50 - 5.49	5	Input 5
5.50 - 6.49	6	Input 6
6.50 - 7.49	7	Input 7
7.50 - 8.49	8	Input 8

This block also has an Active Input and Active Output, which provide an OR function that can be used for an interlock application. The inputs to the OR function are Active Input and the Active Output Mode Flag. Each mode has its own Active Output Mode Flag, which is configured in the data base template. The following chart describes the OR function as it applies to any mode.

Active Input	Active Output Mode Flag	Active Output
True	Yes (True)	True
True	No (False)	True
False	Yes (True)	True
False	No (False)	False

If the Active Input is not connected, its value is False and reliable.

Information Table

The Number of Input and Mode fields that are offered on the template depends on what you specify in the Number of Modes field.

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Number of Modes		INT	2	2-8		X			
Type		TAB	ANA	ANALOG, BINARY, TIME		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
	INPUT 1	BIN	0	0,1	N	X	X		I1
	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
	INPUT 2	BIN	0	0,1	N	X	X		I2
	TIME IN2	TIM	00:00:00	Time	N	X	X		T2
Input 3	INPUT 3	ANA	0.000000	Real	N	X	X		I3
	INPUT 3	BIN	0	0,1	N	X	X		I3
	TIME IN3	TIM	00:00:00	Time	N	X	X		T3
Input 4	INPUT 4	ANA	0.000000	Real	N	X	X		I4
	INPUT 4	BIN	0	0,1	N	X	X		I4
	TIME IN4	TIM	00:00:00	Time	N	X	X		T4
Input 5	INPUT 5	ANA	0.000000	Real	N	X	X		I5
	INPUT 5	BIN	0	0,1	N	X	X		I5
	TIME IN5	TIM	00:00:00	Time	N	X	X		T5
Input 6	INPUT 6	ANA	0.000000	Real	N	X	X		I6
	INPUT 6	BIN	0	0,1	N	X	X		I6
	TIME IN6	TIM	00:00:00	Time	N	X	X		T6
Input 7	INPUT 7	ANA	0.000000	Real	N	X	X		I7
	INPUT 7	BIN	0	0,1	N	X	X		I7
	TIME IN7	TIM	00:00:00	Time	N	X	X		T7
Input 8	INPUT 8	ANA	0.000000	Real	N	X	X		I8
	INPUT 8	BIN	0	0,1	N	X	X		I8
	TIME IN8	TIM	00:00:00	Time	N	X	X		T8

Continued on next page . . .

Field Name (Cont.)	Connect Name	Type	Default	Range	RC	T	I	O	LB
Mode 1		BIN	N (No)	Y(Yes), N(No)		X			
Mode 2		BIN	N (No)	Y(Yes), N(No)		X			
Mode 3		BIN	N (No)	Y(Yes), N(No)		X			
Mode 4		BIN	N (No)	Y(Yes), N(No)		X			
Mode 5		BIN	N (No)	Y(Yes), N(No)		X			
Mode 6		BIN	N (No)	Y(Yes), N(No)		X			
Mode 7		BIN	N (No)	Y(Yes), N(No)		X			
Mode 8		BIN	N (No)	Y(Yes), N(No)		X			
-	MODE	ANA			Y		X		M
-	OUTPUT	ANA	0.000000	Real	Y			X	O
		BIN	0	0,1	Y			X	O
		TIM	00:00:00	Time	Y			X	O
-	ACTIVE IN	BIN	0	0,1	N		X		AI
-	ACTIVE	BIN	0	0,1	N			X	A
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the MSEL block is unreliable when the input value selected to be passed to the output is unreliable. An input can be unreliable if its value is unreliable or is outside the configured range. For example, if the block is configured for three modes, and the input is 3.5 or greater, the output will be unreliable and no advisory will be issued.

The Active Output is unreliable if either the Mode Input or Active Input is unreliable, or if both are unreliable.

Example

The example in Figure 68 is of a 100% outdoor air system. The MSEL block chooses between three damper settings based on the building's mode (BLD_MODE). The BLD_MODE values in the table below come from a time programming process that determines the building's mode of operation. See the following table.

BLD Mode	Mode Selected	Output
0.50 - 1.49	1	0% Open sent to AHU1\DPR-CTRL
1.50 - 2.49	2	10.0% Open (minimum) sent to AHU1\DPR-CTRL
2.50 - 3.49	3	100.0% Open (maximum) sent to AHU1\DPR-CTRL

The CMD block reads the selected mode and sends a SET_AOS command to the damper control (AHU1\DPR-CTRL).

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: SVAR, MSEL, and CMD.

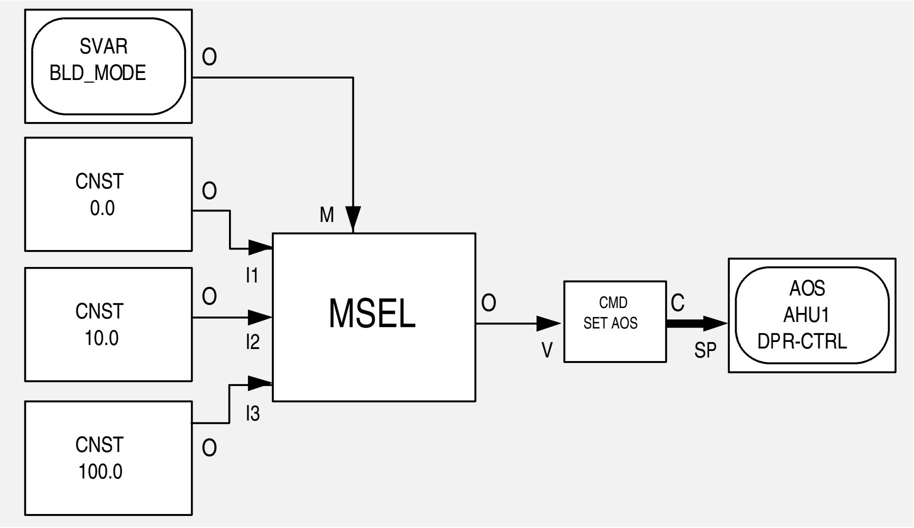


Figure 68: MSEL Block Example

MUL (Multiply) Block

Category

Math

Purpose

Multiplies two inputs using the following equation:

Input 1 * Input 2 = Output

Details

The inputs to the MUL block can be analog only.

You can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the MUL block is unreliable if any of its inputs are unreliable.

Example

In this example (Figure 69), two MUL blocks are used as part of the standard equation for BTUs (British Thermal Units). The BTU equation is:

BTUs/Hour = GPM (ΔT) 500

where:

GPM = Gallons Per Minute

ΔT = Delta T, change of two temperatures

To convert this equation into a GPL diagram, three math blocks are required:

SUB: Subtracts two chilled water temperatures to find ΔT;

MUL (two): Multiplies GPM by 500.0 by ΔT.

The example in Figure 69 performs the following:

1. GPM is multiplied by 500.0. The GPM figure is obtained from an AI object that measures flow (AHU1\FLOW). This is Input 1 of the MUL block. Input 2 of the MUL block is 500.0, from a CNST block.
2. ΔT is calculated. This value is obtained from two AI objects, one reporting chilled water return temperature (AHU1\CHWR-T), and the other chilled water supply temperature (AHU1\CHWS-T). Input 1 of the SUB block is AHU1\CHWR-T, and Input 2 is AHU1\CHWS-T.
3. The result of the MUL and SUB blocks is multiplied. Input 1 of the MUL block is from the first MUL block, and Input 2 is from the SUB block.

Then, a command block reads the calculated value, and sends it to an AD object (AHU1\BTU_HR), whose value can be read on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: both MUL blocks, SUB, and CMD.

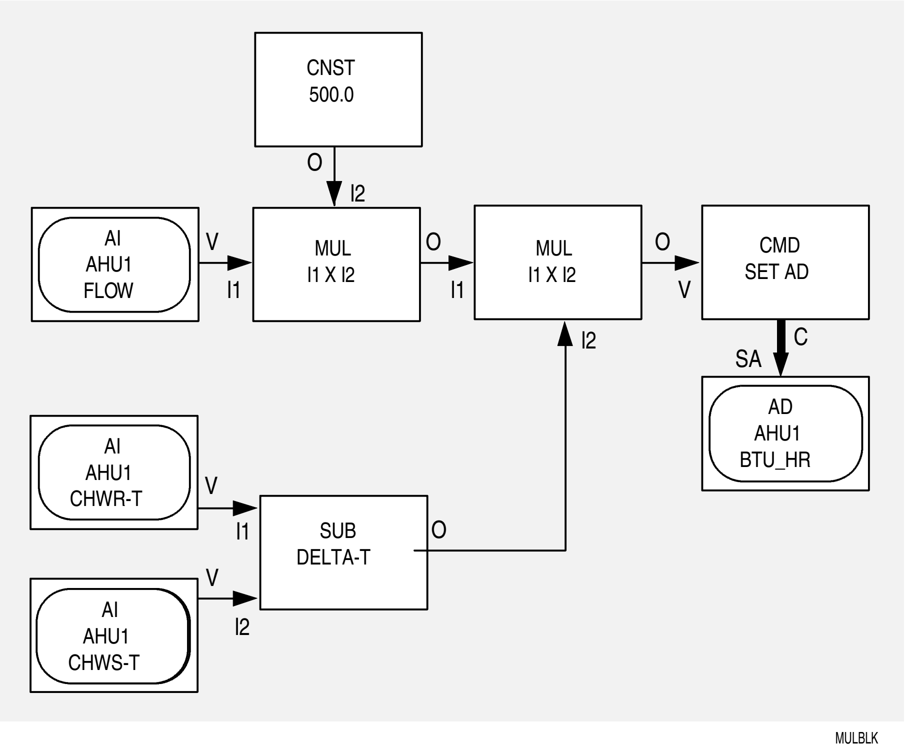


Figure 69: MUL Block Example

NOT Block

Category

Logic

Purpose

Performs a logical Not operation on an input. The output is equal to the opposite (logical negation) of the input.

Details

The NOT block functions as follows:

Input	Output
True	False
False	True

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	INPUT	BIN		0,1	Y		X		I
-	OUTPUT	BIN	0	0,1	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

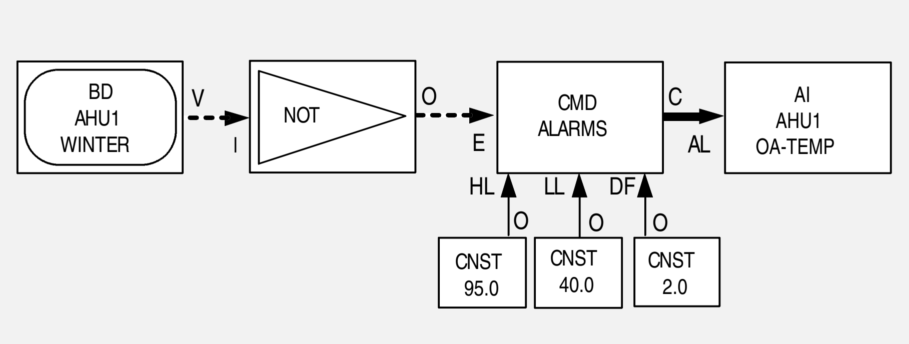
The output of the NOT block is unreliable if its input is unreliable.

Example

The NOT block in this example (Figure 70) aids in sending one of two commands to an AI object (AHU1\OA-TEMP), depending on the season of the year (AHU1\WINTER). When the mode changes from winter to summer, the CMD block sends summer alarm limits and differential to AHU1\OA-TEMP.

A period of 00:00:00 is defined for this process, since it needs to run only when a triggerable attribute of the BD object changes.

Note: The NOT and CMD blocks, both operation blocks, must be placed in a process.



NOTBLK

Figure 70: NOT Block Example

OR Block

Category

Logic

Purpose

Performs a logical OR operation on two, three, or four binary inputs. The output is either True or False.

If any of the inputs into this block are True, the output is True; otherwise, the output is False.

You can configure this block to have two, three, or four inputs. The default is two. If you want to select between three or four inputs, you must first specify the number of inputs in the template before making the connection.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Number of Inputs		INT	2	2, 3, 4		X			
-	INPUT 1	BIN		0,1	Y		X		I1
-	INPUT 2	BIN		0,1	Y		X		I2
-	INPUT 3	BIN		0,1	Y		X		I3
-	INPUT 4	BIN		0,1	Y		X		I4
-	OUTPUT	BIN	0	0,1			Y	X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the OR block is unreliable if any of its inputs are unreliable.

Example

In the example shown in Figure 71, the OR block aids in choosing to start an exhaust fan (AHU1\EXH-FAN), based on the status of two supply fans (AHU1\FAN1-ST and AHU1\FAN2-ST). If either of the supply fans is on, the OR block allows the CMD block to send a Start command to the exhaust fan. Otherwise, the CMD block sends a Stop command to the exhaust fan.

A period of 00:00:00 is defined for this process, since it needs to run only when a triggerable attribute of one of the BI objects changes.

Note: The OR and 2CMD blocks, both operation blocks, must be placed in a process.

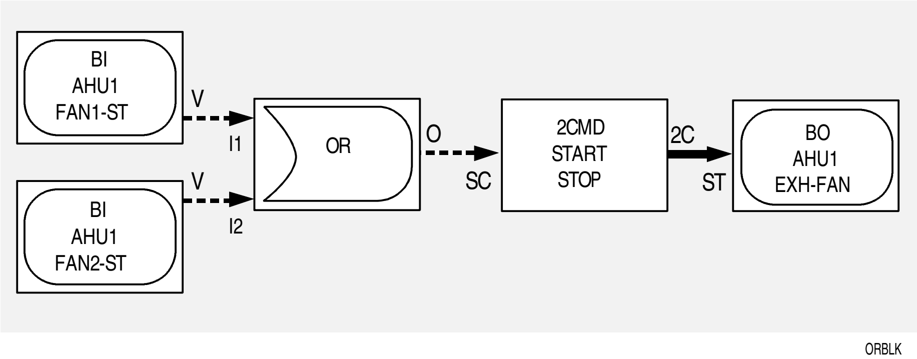


Figure 71: OR Block Example

PERD (Period) Block

Category

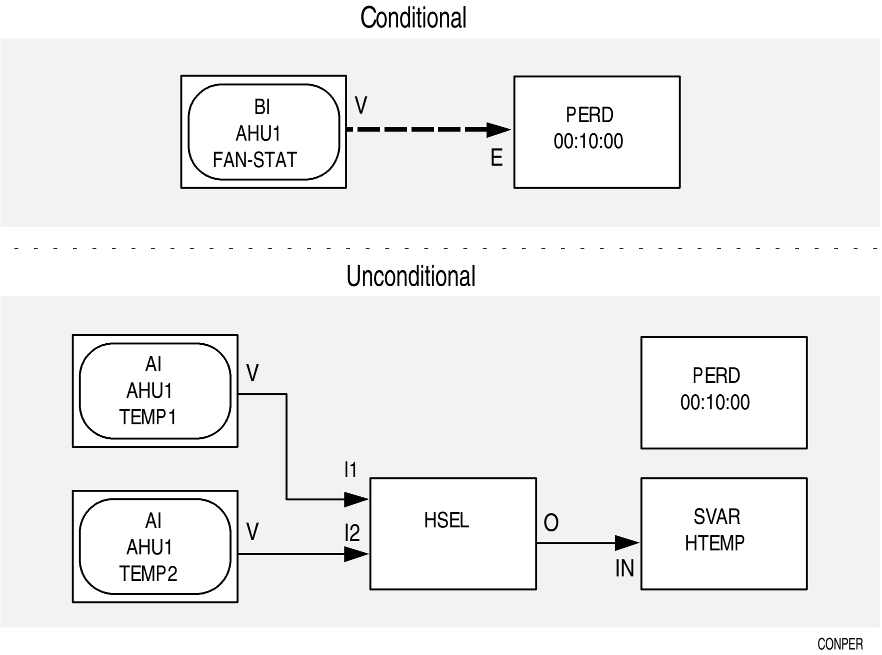
Process Control

Purpose

Changes the period of a process, either conditionally or unconditionally. The process that is affected is the one in which the PERD block is contained.

Details

The PERD block can provide a conditional or unconditional process period. A conditional PERD block changes the process period based on some binary status, such as fan status (Figure 72). An unconditional PERD block redefines the process period, in which the period of the PERD block is used, not the period as defined in the process compound template.


Conditional

Unconditional

Figure 72: Conditional and Unconditional Periods

This block has an Enable input, which provides conditional logic. If this optional input is connected, the process period will be changed only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, a period will not be changed. If the Enable input is not connected, the period will be changed each time the PERD block is executed.

When this block is executed and the Enable input is True, the time specified in the PERD block's template is used for the processes' period. However, if the Enable input is False, the period specified in the process compound template or in another PERD block is used.

More than one PERD block can be used in a single process. However, the last PERD block to be executed will define the period that the process actually uses.

A period of 00:00:00 will cause the process to run only when triggered by some other factor, such as by a change-of-state of an object's triggerable attribute.

A process will not execute exactly as the period states. The following external factors may affect execution:

- Some time is consumed in executing the process before the period is determined.
- The timer of a WAIT block, if used in the process, is added to the period timer.
- Higher priority processes may be running, preventing a lower priority process from executing.
- The process may be executed before the period expires by some other means (e.g., triggered by another process).

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Period	TIME IN	TIM	00:01:00	Time	N	X	X		TI
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

The PERD block will not set a period if the Enable input is False (reliable or unreliable), or True and unreliable. In this case, either the period as set by the last enabled PERD block to be encountered, or the period as defined in the processes' template, is used. The process period is also used if the process contains multiple PERD blocks that are all disabled. In addition, the Time In value of the PERD block is used regardless of its reliability.

Example

In this example (Figure 73), the PERD block directs this process to run either once every two minutes (00:02:00) or only when triggered (00:00:00). The period selected depends on the status of a supply fan (AHU1\SFAN-ST). If the supply fan is on, the SWCH block selects a period of 00:02:00 minutes, which means the process will run once every two minutes. However, if the supply fan is off, the SWCH selects a period of 00:00:00 minutes, in which case the process will run only when triggered by the BI object (AHU1\SFAN-ST). The lower portion of this diagram determines economizer mode.

Note: All operation blocks in this example must be placed in a process. They include: PERD, SVAR, SWCH, and DFCM.

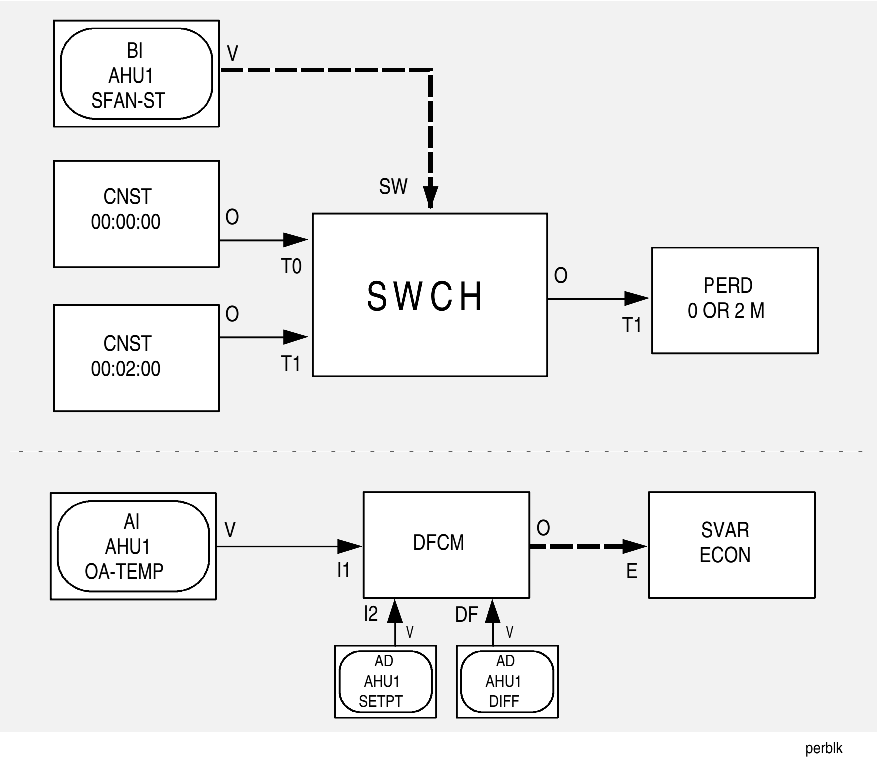


Figure 73: PERD Block Example

PIR (PI Reset) Block

Category

Control

Purpose

Resets a setpoint using a proportional plus integral algorithm.

Note: For network performance reasons, we recommend that the process in which this block runs has a period of at least one minute.

Details

The PIR block calculates a value based on the difference between the Setpoint, Input (feedback), Prop Band, and Integral Time. The output is a result of using a proportional plus integral control algorithm that is resident in the NCM.

The initial value of the output is the OFFSET if no error is present and it is the first time the block is executed.

PIR Block Algorithm

The PIR block uses the following algorithm:

calculate error:

```
error = SETPOINT - INPUT
if (abs(error) <= DEADBAND) then error = 0
if (error < DEADBAND) then error = error + DEADBAND
if (error > DEADBAND) then error = error - DEADBAND
```

calculate proportional term:

```
pterm = ((HI OUTPUT - LO OUTPUT) / PROPBAND) * error
```

calculate sample_time:

```
sample_time = current_time - last_time
```

calculate output:

```
OUTPUT = pterm + iterm + OFFSET;
if (OUTPUT > HI OUTPUT) then OUTPUT = HI OUTPUT
if (OUTPUT < LO OUTPUT) then OUTPUT = LO OUTPUT
```

calculate integral term:

```
if  (sample_time <= INT TIME)  then
   igain = sample_time / INT TIME
   iterm = igain * (OUTPUT - OFFSET) + (1 - igain) * iterm
```

Terms of Algorithm

Here are definitions of the terms used in the PIR block algorithm.

SETPOINT

Desired INPUT value.

INPUT

Feedback value.

PROPBAND (Proportional Band)

Change in feedback that will cause a full scale change in the value calculated by the function. If the PROPBAND is 0, a runtime error will result.

INT TIME (Integral Time)

Number of minutes needed for the integral response to equal the proportional response; must be greater than or equal to 0, or an error is generated and a default value of 5 set unreliable is used.

OFFSET

Initial value of the reset function if no error is present the first time the algorithm is executed.

DEADBAND

A value that is twice the acceptable value outside of setpoint that must be crossed before change of output occurs. Value must be greater than or equal to 0, or an error is generated and a default value of 1.0 is used.

LO OUTPUT

Lowest value that this function will calculate.

HI OUTPUT

Highest value that this function will calculate.

RESTART

Logical value that, when True, causes the function to reset the integral term (iterm) to zero.

Two PIR Block Configurations

The PIR block can be configured for two types of control: proportional only and proportional plus integral. To configure for proportional only, set integral time (INT TIME) to 0. To configure for proportional plus integral, integral time must be greater than 0, and the time between successive block executions must be less than integral time.

PI Configuration Only

The following information applies to a PIR block configured for proportional plus integral control.

This block has an optional input called Restart that resets the integration term (iterm) to zero (0), which in effect cancels the integral action of the PIR block. If the Restart input is not connected, the default state of False and the integration term (iterm) are used. If the Restart input is connected and its state is True and reliable, the integration term (iterm) is reset to zero (0), and the block functions as proportional only.

The integral action is zeroed (eliminated) the first execution after:

- The process that contains the PIR block is downloaded or enabled.
- The NCM that contains a process with a PIR block is warm started.
- The time between successive PIR block executions is greater than the Integral Time (INT TIME).

Subsequent executions of this block may cause the integral term (iterm) to change to zero.

The output of the PIR block is limited to between the High Output and Low Output. The integral action will not wind up beyond these limits.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Setpoint	SETPOINT	ANA	70.00000	Real	N	X	X		SP
Prop Band	PROPBAND	ANA	10.00000	Real ≠ 0	N	X	X		PB
Integral Time	INT TIME	ANA	20.00000	Real ≥ 0 (minutes)	N	X	X		IT
Offset	OFFSET	ANA	70.00000	Real	N	X	X		OS
Deadband	DEADBAND	ANA	1.000000	Real ≥ 0	N	X	X		DB
High Output	HI OUTPUT	ANA	85.00000	See Note	N	X	X		HO
Low Output	LO OUTPUT	ANA	55.00000	See Note	N	X	X		LO -
-	INPUT	ANA		Real	Y		X		IN
-	RESTART	BIN	0	0,1	N		X		RS
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: Enter a real number for this value. The value for Low Output must be less than or equal to High Output.

Reliability

The output of the PIR block is unreliable if any of its inputs are unreliable (or set unreliable because a value is out of range). In this case, the output is set to be the last output set unreliable, and the integration term (iterm) is set to zero (0). When the PIR block returns to reliable, the integration term (iterm) is again factored into the PIR algorithm.

Example

In the following example (Figure 74), the PIR block calculates a new setpoint for the AOS object (AHU1\SETPOINT), based on outside air temperature (AHU1\OA-TEMP). The CMD block sends the new setpoint value to the AOS object.

A period of 00:03:00 is defined for this process, which means it will execute once every three minutes.

Note: The PIR and CMD blocks, both operation blocks, must be placed in a process.

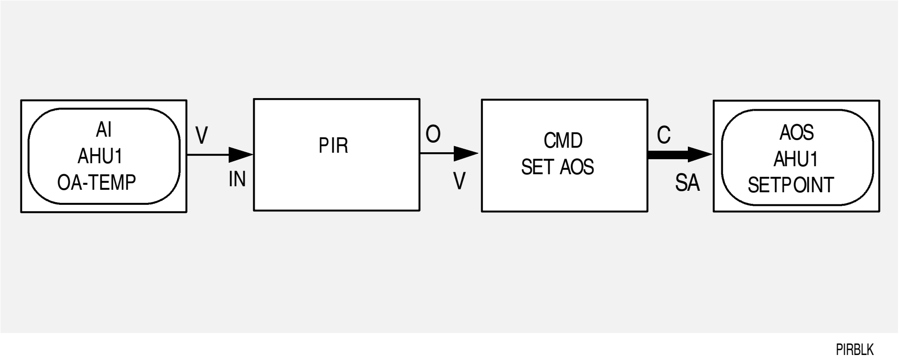


Figure 74: PIR Block Example

PRNT (Print) Block

Category

Report

Purpose

Prints a message to any printer on the network. Each time this block is executed (and the Enable input is True and reliable), the message will be sent to the printer identified in the block's data base template.

Details

The Object Name field in the PRNT template defines the printer to which the message is sent.

A value, the current date, and the current time may be appended to the end of the message. Here is an example:

```
Outside temperature is: 76.0  01/01/90 13:24:02
```

In this example, the text entered in the template is "Outside temperature is:." The type of data selected is Analog. The 76.0°F value comes from an outdoor air temperature sensor. The Date? and Time? fields are answered Yes. Notice that time is presented in 24-hour format.

You can configure this block to accept analog, binary, or time value connections. The default is analog. If you want to print binary or time data, you must first specify the type of data in the template before making a connection.

A carriage return and line feed is sent at the end of each print statement.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will print a message only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not print a message. If the Enable input is not connected, the block will print a message each time it is executed.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Object Name (Printer Name)		STR	Blank	8 Char		X			
Text		STR	Blank	50 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
Date ?		STR	Y (Yes)	Y(Yes), N(No)		X			
Time ?		STR	Y (Yes)	Y(Yes), N(No)		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	0	0,1	N			X	E
-	INPUT	BIN	0	0,1	N		X		I
		ANA	0.000000	Real	N		X		IN
-	TIME IN	TIM	00:00:00	Time	N		X		TI
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: *Appendix F: Characters, Symbols, and Reserved Words* lists valid characters for use in messages.

Reliability

No message is printed if the Enable input is False or unreliable. However, if the input is unreliable, the message is printed. The message does not indicate the value is unreliable.

Example

The example in Figure 75 contains a PRNT block that sends a message to the printer. If the building is occupied (AHU1\OCC-UNOC) and the discharge air temperature (AHU1\DSCH-TMP) is in high alarm, the PRNT block prints this message:

```
AHU1\DSCH-TMP IS IN HI ALARM. VALUE IS: 80.0
```

A period of 00:00:00 is defined for this process since it needs to run only when a triggerable attribute of the BI object changes.

Note: The AND and PRNT blocks, both operation blocks, must be placed in a process.

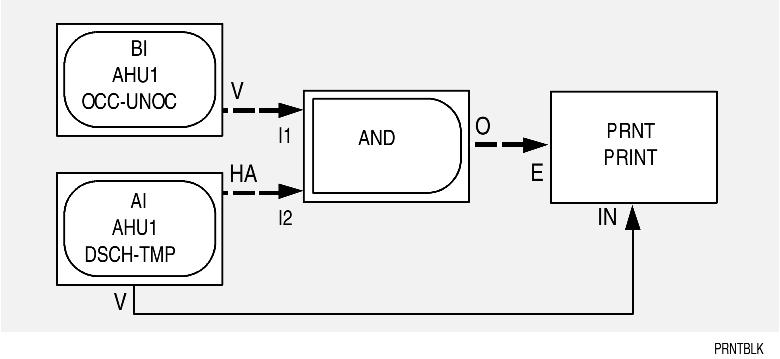


Figure 75: PRNT Block Example

PULS (Pulse) Block

Category

Logic

Purpose

Outputs a True state for a period of time, then returns to False.

Details

The PULS block has three different configuration types: Cancelable, Non-cancelable, or One-shot.

Cancelable PULS

The **Cancelable** type outputs a True state as long as the input remains True and the pulse timer has not yet expired. The output varies as shown in Figure 76 and according to the table that follows.

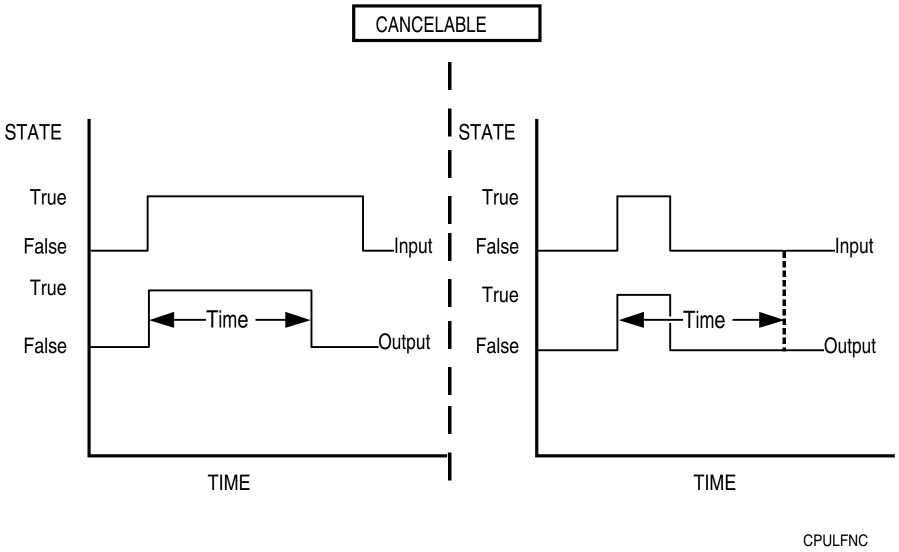


Figure 76: Cancelable Pulse Function

Cancelable Pulse	
Change of Input	**Output**
False --> True	True for pulse time specified, or until input returns to False
True --> False	False until input changes to True

When the pulse timer expires, the process that contains this block will be triggered. If the input returns to False before the timer expires, the process will not be triggered. In addition, the process is not triggered if you specify the timer as 00:00:00. In that case, the output is True for only one execution.

For initialization applications, a Cancelable PULS block can accept a constant of True for its input. In this case, the output varies as follows:

Cancelable Pulse	
Constant Input	**Output**
True	True for first execution and until pulse timer expires, then False for subsequent executions until NCM is warm started or process is redownloaded

Non-Cancelable PULS

The **Non-cancelable** type outputs a True state when the input is True, and remains True until the pulse timer expires, regardless if the input returns to False. The output varies as shown in Figure 77 and according to the table that follows.

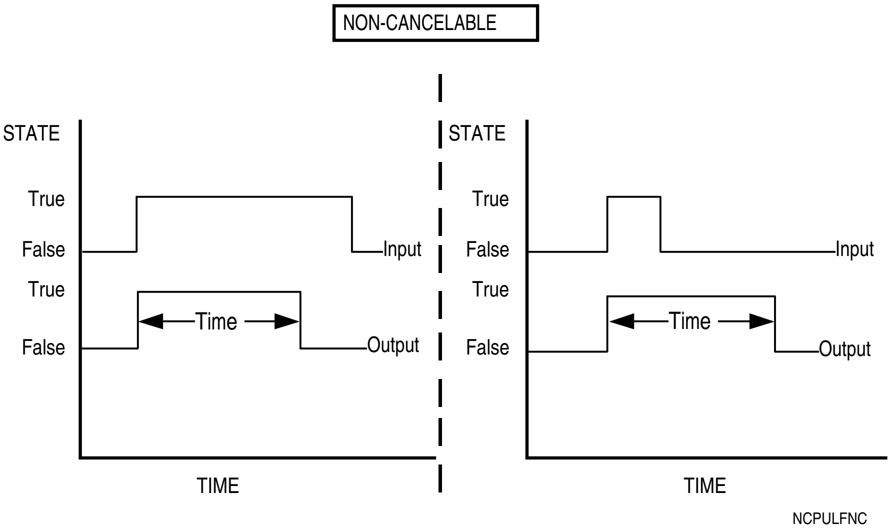


Figure 77: Non-Cancelable Pulse Function

Non-Cancelable Pulse	
Change of Input	**Output**
False --> True	True for pulse time specified.
True --> False	True until timer expires, then False.

When the pulse timer expires, the process that contains this block will be triggered, and the output will return to False. The process is not triggered if you specify the timer as 00:00:00. In this case, the output is True for only one execution.

For initialization applications, a Non-cancelable PULS block can accept a constant of True for its input. In this case, the output varies as follows:

Non-Cancelable Pulse	
Constant Input	**Output**
True	True for first execution and until pulse timer expires, then False for subsequent executions until NCM is warm started or process is re-downloaded.

One-Shot PULS

The **One-shot** type outputs a True state when the input changes to True, but for only one execution of the process. This type does not trigger a process. The output varies as shown in Figure 78 and according to the table that follows.

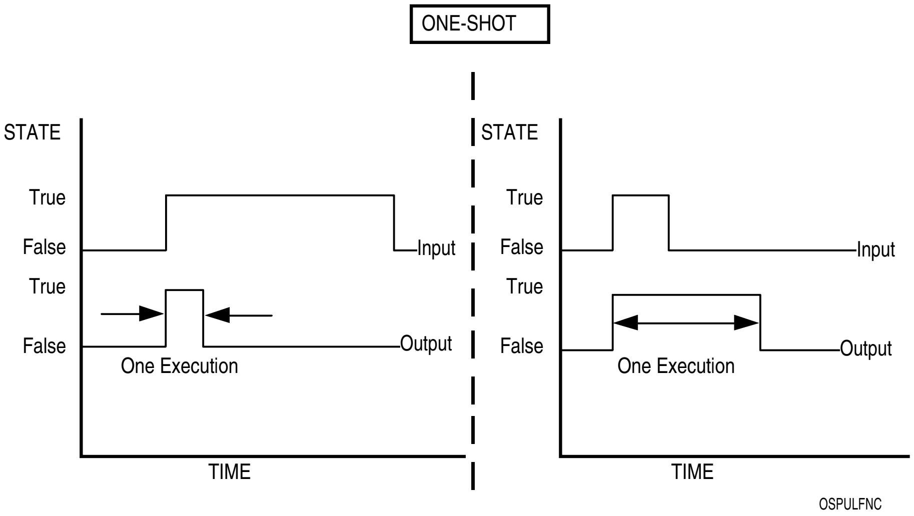


Figure 78: One-Shot Pulse Function

One-Shot Pulse	
Change of Input	**Output**
False --> True	True for one execution of the process, then back to False.
True --> False	False until input changes to True.

For initialization applications, a One-shot PULS block can accept a constant of True for its input. In this case, the output varies as follows:

One-Shot Pulse	
Constant Input	**Output**
True	True for first execution, then False for subsequent executions until NCM is warm started or process is re-downloaded.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	CANCEL	CANCEL, ONE SHOT, NON-CANC		X			
Time	TIME IN	TIM/ POP-UP	00:00:05	Time > 00:00:00	N	X	X		TI
-	INPUT	BIN		0,1	Y		X		I
-	OUTPUT	BIN	0	0,1	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The reliability of the PULS block depends on its type: Cancelable, Non-cancelable, or One-shot.

Cancelable

The reliability of the output cannot change unless the value of the input changes. The following table describes the reliability of the Cancelable PULS block.

Input And Timer (IF...)	Transition of Output (AND...)	Output (THEN...)
If the input changes from False to True, and both the input and timer are reliable.	False to True	**Reliable**
If the input changes from True to False reliably before the timer has expired.	True to False	
If the input changes from False to True, and either the input or timer is unreliable.	False to True	**Unreliable**
If the input changes from True to False unreliably and the timer has not expired.	True to False	
Either the input changes from True to False, regardless of reliability, and the timer has expired; or the input's value does not change, but its reliability does.	No transition of output	**No Change in Reliability**
If the input remains True, regardless of reliability, and the timer expires.	True to False	

Non-Cancelable

An unreliable output cannot change to reliable until the input makes a False to True transition. The following table describes the reliability of the Non-cancelable PULS block.

Input and Timer (IF..)	Output (THEN...)
If input changes from False to True, both the input and the timer are reliable, and the timer is not active.	**Reliable**
If the input changes from False to True, the input and/or the timer is unreliable, and the timer is not active.	**Unreliable**
If input changes from True to False or from False to True, and the timer is active.	**No Change in Reliability**

One-Shot

The output is unreliable if the input is unreliable.

Example

The example in Figure 79 uses a One-shot PULS block in an interlock sequence. When the supply fan starts (AHU1\SFAN-ST1), the PULS block enables the CMD block to send a Start command to the exhaust fan (AHU1\EX-FAN1).

A period of 00:01:00 is defined for this process, which means it will run once every minute or when a triggerable attribute of the BI object changes.

Note: The PULS and CMD blocks, both operation blocks, must be placed in a process.

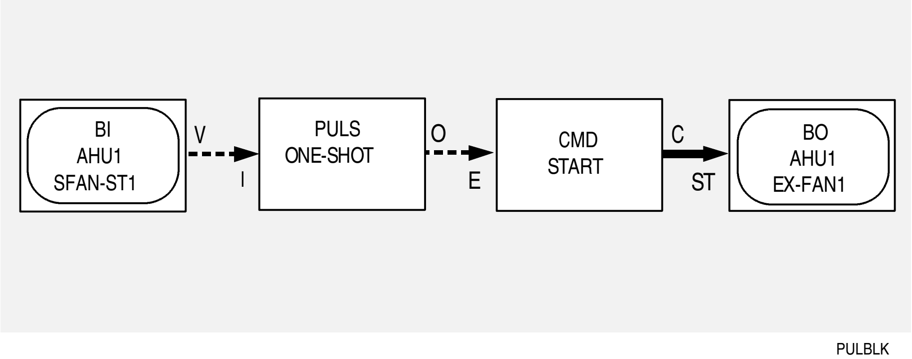


Figure 79: PULS Block Example

RAMP Block

Category

Calculations

Purpose

Varies an output at a predetermined linear rate, from one value to another. The block can be used to gradually start equipment.

Details

You can configure the RAMP block for direct acting or reverse acting control (Figure 80). A direct acting ramp increases the output of the RAMP block for each execution by adding the previous output value to the Step Size value. It continues until the Output is greater than or equal to the End value. Then, ramping stops. A reverse acting ramp decreases the output of the RAMP block for each execution by subtracting the Step Size value from the previous output value. It continues until the Output is less than or equal to the End value. Then, ramping stops.

To Define	Set
Direct acting ramp (ascending)	Start value < End value
Reverse acting ramp (descending)	Start value > End value

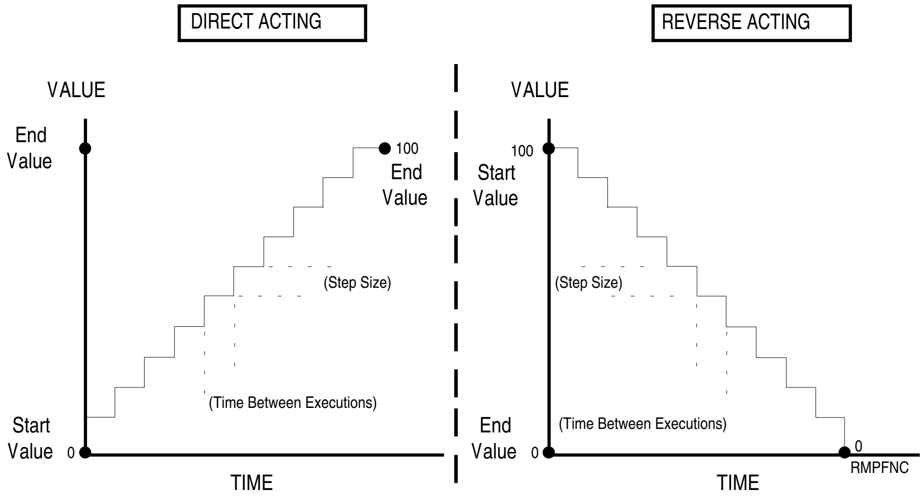


Figure 80: Ramp Function

This block has a Reset input. The RAMP block begins to execute when the Reset input changes from True to False.

For the first execution, the block’s output is equal to the Start value. For the second and subsequent executions, the output is equal to the last output plus (direct) or minus (reverse) the Step Size value. When the output reaches the End value, it remains there until the Reset input changes from True to False, and ramping starts again.

When Reset input is True and reliable, the output returns to the Start value, and remains at the Start value until the Reset input changes to False.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Step Size	STEPSIZE	ANA	5.000000	Real ≥ 0	N	X	X		SS
Start Value	START VAL	ANA	0.000000	Real	N	X	X		SV
End Value	END VAL	ANA	100.0000	Real	N	X	X		EV
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	RESET	BIN	0	0,1	N		X		R
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

If the Reset input is True and reliable, the output equals the Start value, flagged reliable. If the Reset input is unreliable, the output equals the Start value, flagged unreliable. If the Reset input is False and reliable, and the End value, Step Size, or last output is unreliable, the output will be the last output, flagged unreliable. Also, if an external input provides a Step Size value of less than 0, a runtime error message is generated.

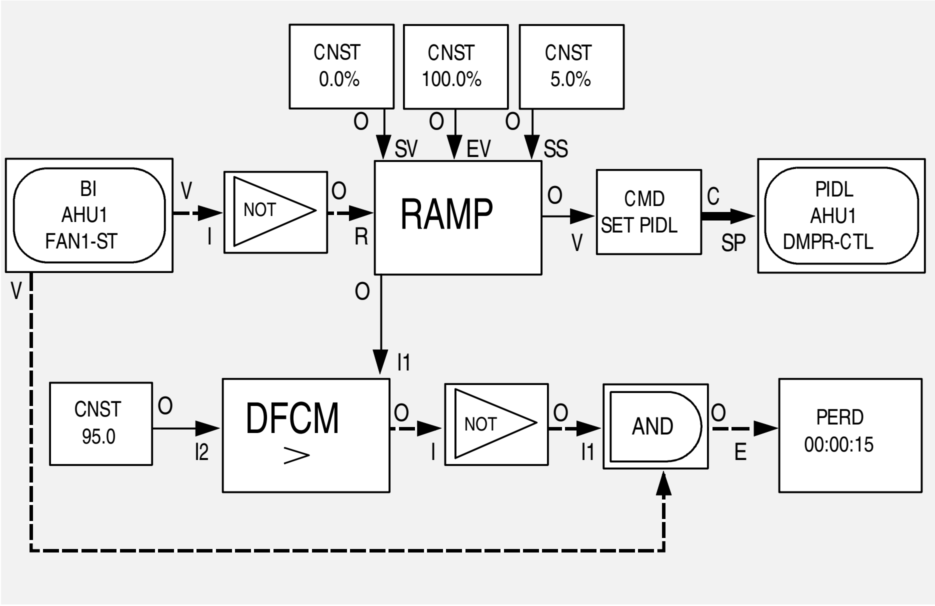
Example

In the following example (Figure 81), the RAMP block gradually opens a damper from 0 to 100% in 5% increments. When the fan turns on (AHU1\FAN1-ST), the reset on the RAMP block is set to 0, which activates the ramp. The CMD block sends the Start value of 0% to the PIDL loop that controls the dampers (AHU1\DMPR-CTL). Then, the DFCM block compares whether the output of the RAMP block is greater than 95.0%.

If the output is less than 95.0%, the CMD block sends the PIDL object a value of 5.0%, which is the Last Output value plus the Step Size (0.0 + 5.0% = 5.0%). Then, 15 seconds later (00:00:15), another comparison is made to see if the output is less than 95.0%. As long as the comparison is True, a command that is 5.0% greater than the last command is sent to the dampers until a command of 100.0% has been sent.

If the output is greater than 95.0%, the process period is set to 00:00:00, which stops the execution of the process.

Note: All operation blocks in this example must be placed in a process. They include: RAMP, DFCM, CMD, NOT, AND, and PERD.



RMPBLK

Figure 81: RAMP Block Example

READ (Read Attribute) Block

Category

Object Control

Purpose

Obtains the value of any readable attribute of an object, except string attributes. The purpose of the READ block is to access the readable attributes that do not appear on the connection menus. For a list of the readable attributes for each object, refer to *Appendix G: Attributes*.

Details

You may use the READ block to obtain the values of readable attributes that appear on the connection menus, but you should instead use a direct connection to access these.

You can configure this block to accept an analog, binary, or time attribute connection. The default is analog. If you want to read a binary or time attribute, you must first specify the type of attribute in the template before making a connection.

Note: To configure this block to read integer values, define its type as analog. The integer will be converted to an analog value, since GPL cannot use integers directly.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will read the attribute only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not read the attribute. If the Enable input is not connected, the attribute is read each time the block executes.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name (Attribute)		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	OUTPUT	BIN	0	0,1	N			X	O
		ANA	0.000000	Real	N			X	O
		TIME	00:00:00	Time	N			X	O
-	READ	REA D			Y			X	RD
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected.
The default is 0.

Reliability

The output of the READ block is unreliable if the input attribute is unreliable.

Example

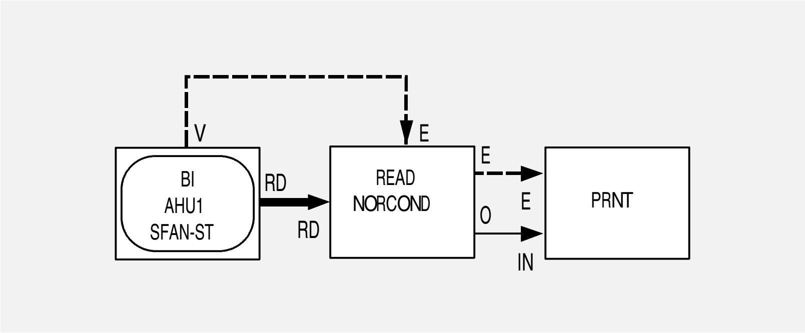
In the example shown in Figure 82, the READ block reads the NOR_COND attribute of the BI object (AHU1\SFAN1-ST), and sends its value to a printer. The NOR_COND attribute is an integer, but since GPL does not recognize integers, this value is configured as Analog in the READ data base template. An Enable line is drawn from the BI's value attribute to the READ block, and another Enable line is drawn from the READ block to the PRNT block. In this way, the value of NOR_COND will be printed only when it changes from off to on. The following statement is printed:

```
THE NORMAL CONTACT POSITION IS:<NOR_COND value>
01/01/90 13:24:02
```

When the value of NOR_COND changes from on to off, no statement is printed.

A period of 00:00:00 is defined for this process, since it only needs to run when a triggerable attribute of the BI object changes.

Note: The READ and PRNT blocks, both operation blocks, must be placed in a process.



READBLK

Figure 82: READ Block Example

RH (Relative Humidity) Block

Category

Psychrometric Equations

Purpose

Calculates relative humidity based on dry bulb temperature and dew point temperature.

Details

The relative humidity calculation can be done in either English or Metric units.

Valid ranges for the calculated relative humidity:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Dew Point Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH METRIC		X			
-	DRY BULB	ANA		Real	Y		X		DB
-	DEW POINT	ANA		Real	Y		X		DP
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the dry bulb temperature input or dew point temperature input is unreliable. Also, if either of these values is less than the lowest part of their range, relative humidity is calculated using the lowest range value (e.g., for dry bulb temperature,-147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if either of these values is greater than the highest part of their range, relative humidity is calculated using the highest range value (e.g., for dry bulb temperature, 391.9°F), the output is unreliable, and a runtime error message is generated.

A runtime error message is also generated if dew point temperature is greater than dry bulb temperature.

Example

In the example shown in Figure 83, the RH block calculates return air relative humidity, based on return air temperature (AHU1\RA-TEMP) and return air dew point (AHU1\RA-DWPT). The calculation is made only when the fan (AHU1\FAN1-ST) is on. The CMD block then sends the calculated result to an AD point (AHU1\RA-RH), which can be read at an operator device.

A conditional period of 00:02:00 is defined for this process, which means it will execute once every two minutes, as long as the fan is on. If the fan is off, a zero period is defined.

Note: All operation blocks in this example must be placed in a process. They include: RH, CMD, and PERD.

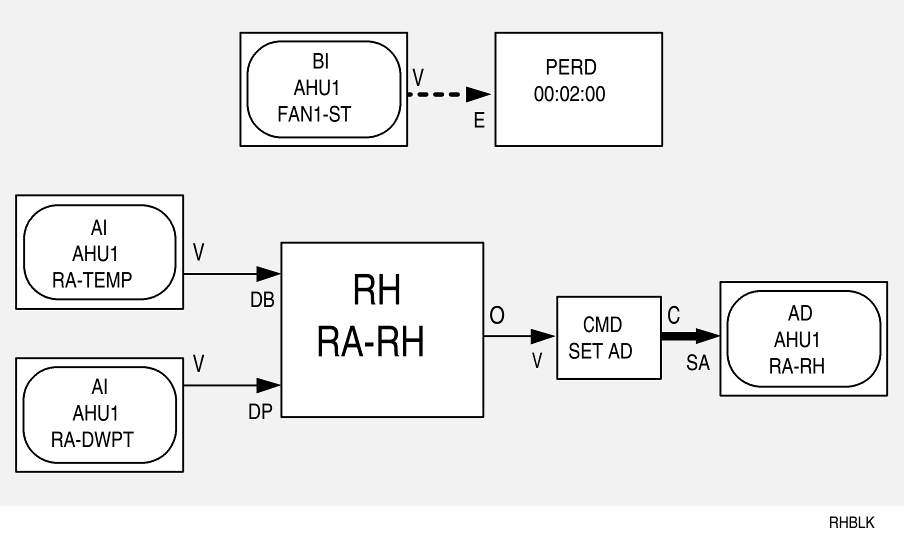


Figure 83: RH Block Example

RTOT (Real-to-Time) Block

Category

Time

Purpose

Converts a real value in minutes or seconds into a time value in 24-hour format.

Details

The RTOT block can be configured for either minutes or seconds resolution. You select either minutes or seconds in the Resolution tab field. Minutes is the default.

If minutes resolution is chosen, the input value represents a number of minutes. For example, an input value of 450 means 450 minutes:

450 minutes is converted to 07:30:00.

If seconds resolution is chosen, the input value represents a number of seconds. For example, an input value of 450 means 450 seconds:

450 seconds is converted to 00:07:30.

An analog value is rounded before it is converted to time. For example:

386.2 minutes is rounded to 386.0, then converted to 06:26:00.

386.5 seconds is rounded to 387.0, then converted to 00:06:45.

The time value is always represented in hours, minutes, and seconds (00:00:00).

Note: A runtime error is generated when the input of the RTOT block equals either less than 0 hours, or greater than or equal to 24 hours.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Resolution		TAB	MINUTES	MINUTES/ SECONDS		X			
-	INPUT	ANA		Real	Y		X		IN
-	OUTPUT	TIM	00:00:00	Time	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the RTOT block is unreliable when the conversion from real to time results in a value, after rounding, that is less than 0 hours or greater than or equal to 24 hours. If the result is less than 0 hours, the output is set to unreliable and given a value of 00:00:00. Similarly, if the result is greater than or equal to 24 hours, the output is set to unreliable and given a value of 23:59:59.

Example

In the example shown in Figure 84, the RTOT block converts a time in minutes to a time in 24-hour format, to provide for an adjustable delay. The minute value is an AD object (AHU1\P1-DELAY) that is defined as 10.5 minutes, which can be adjusted at an operator device. The RTOT block rounds 10.5 minutes to 11.0, and then converts to 00:11:00. The DLAY block reads this time value from the output of the RTOT block, and if the fan is on (AHU1\FAN1-ST), sets a delay of 00:11:00. When the delay timer expires, the process is triggered and the following statement is printed 11 minutes (00:11:00) after the fan has started:

```
RETURN AIR RELATIVE HUMIDITY IS <value>
```

A period of 00:00:00 is defined for this process, since it only needs to run when a triggerable attribute of the BI object changes.

Note: All operation blocks in this example must be placed in a process. They include: RTOT, PRNT, DLAY, and SVAR.

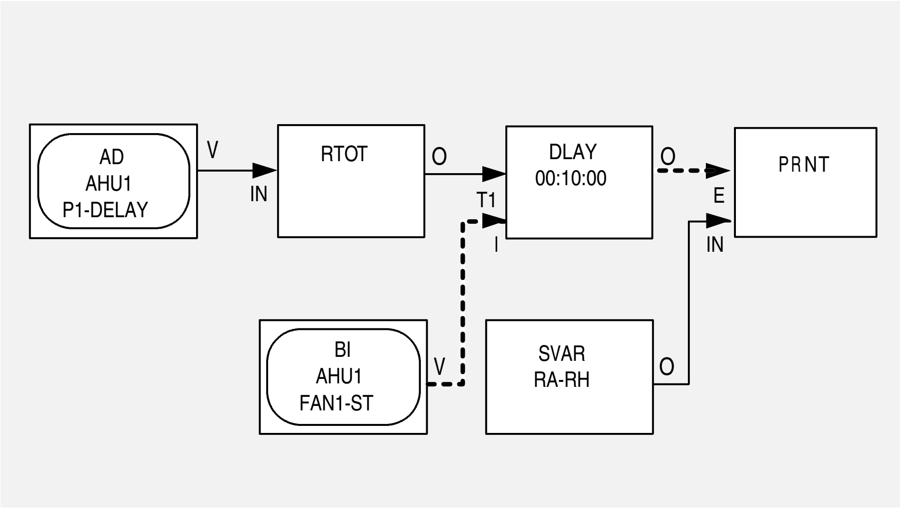


Figure 84: RTOT Block Example

SAMP (Sample and Hold) Block

Category

Selectors

Purpose

Reads and stores an analog value when the Hold input is True and reliable.

Details

The SAMP block reads and stores the value of the input when the Hold input changes from False to True. The value is held until the Hold input changes to False, at which point the output will equal the input.

The following table describes how the transition of the Hold input affects the output and its reliability.

Time	Hold Input Transition	Output	Reliability of Output
T0	False and reliable	Input	Input’s reliability
T1	True and reliable	Snapshot of Input at transition time	Input’s reliability
T2	True and unreliable	Snapshot of Input at T1	Unreliable
T3	True and reliable	Snapshot of Input at T1	Input’s reliability at T1
T4	False and unreliable	Input	Unreliable
T5	False and reliable	Input	Input’s reliability

The SAMP block can sample and hold analog values only.

Differences Between SAMP, SVAR, and VH Blocks

Though these three blocks are similar, they have the following differences.

Block	Input is Assigned to Output When:
SAMP	Hold input is False and reliable
SVAR	Enable input is True and reliable
VH	Enable input is True and reliable

The SVAR differs from the other blocks in that it can share data between processes on the same NC. In addition, more than one SVAR block with the same name cannot exist in the same process.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	HOLD IN	BIN		0,1	Y		X		H
-	HOLD OUT	BIN	0	0,1	N			X	H
-	INPUT	ANA		Real	Y		X		IN
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

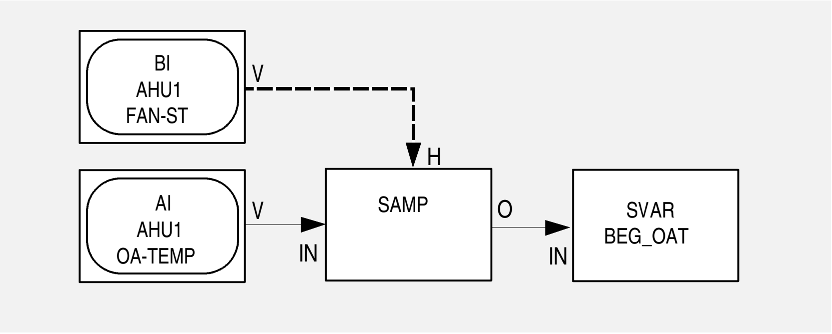
The output of this block is unreliable if the Hold input is unreliable. If the Hold input is reliable and False, the reliability of the output matches the reliability of the input. If the Hold input is reliable and True, the reliability of the output matches the reliability of the input when the sample was taken.

Example

In the following example (Figure 85), the SAMP block samples and holds the value of the outside air temperature (OA-TEMP) when the fan (AHU1\FAN-ST) starts. The SVAR block (BEG_OAT) reads this value, which can then be used in a different process. The SVAR block holds this value when the fan stops. When the fan starts again, outside air temperature is again read and sent to the SVAR block.

A period of 00:05:00 minutes is defined for this process, which means it will run once every five minutes, or when a triggerable attribute of the BI object changes.

Note: The SAMP and SVAR blocks, both operation blocks, must be placed in a process.



SAMPBLK

Figure 85: SAMP Block Example

SPAN Block

Category

Calculations

Purpose

Outputs a value that is in proportion to the input. The input and output are linear. For example:

Input Range	Input Default	Output Range	Output Default
0 - 100	50	0 - 10	5

Details

You can configure the SPAN block for direct acting or reverse acting control. The values of Output Range 1 and Output Range 2 determine the action. A direct acting span function increases the value of an output; a reverse acting span function decreases the value of an output. See the table and Figure 86 that follow.

To Define	Set
Direct acting span (ascending)	Output Range 1 = 0.0 Output Range 2 = 100.0
Reverse acting span (descending)	Output Range 1 = 100.0 Output Range 2 = 0.0

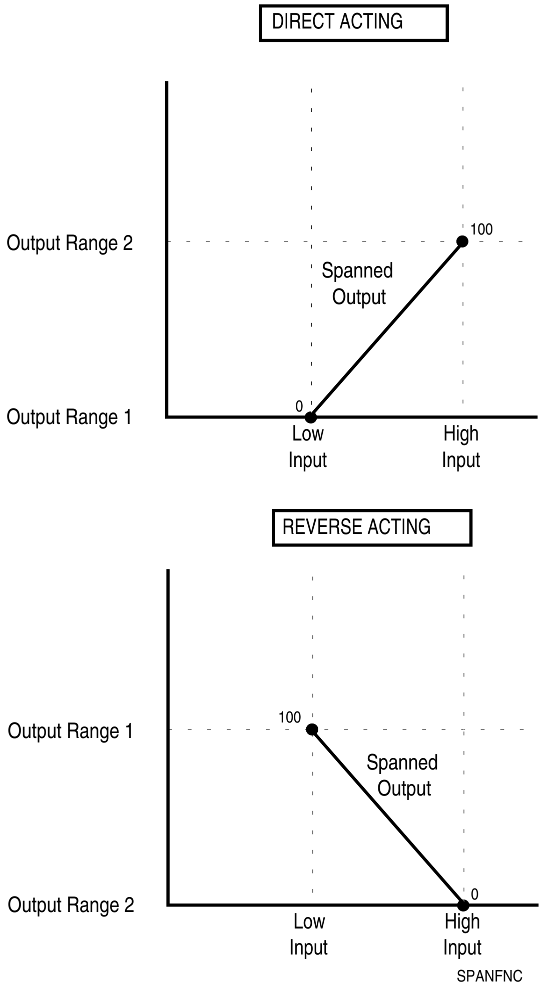


Figure 86: SPAN Functions: Direct Acting and Reverse Acting

Note: You must define a Low input that is less than the High input. Otherwise, the Translator will generate a fatal error.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Low Input	LO INPUT	ANA	0.000000	Real	N	X	X		LI
High Input	HI INPUT	ANA	100.0000	Real	N	X	X		HI
Output Range 1	OUT RG 1	ANA	0.000000	Real	N	X	X		R1
Output Range 2	OUT RG 2	ANA	100.0000	Real	N	X	X		R2
-	INPUT	ANA		Real	Y		X		IN
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if one of the following is unreliable: Input, High Input, Low Input, Output Range 1, or Output Range 2. If the input is greater than the High Input, the value of the High Input is used for spanning. Similarly, if the input is less than the Low Input, the value of the Low Input is used. In both cases, the output will be reliable.

Example

In the following example (Figure 87), the SPAN block performs a reset schedule on a setpoint (AHU1\SETPT), based on the outside air temperature (AHU1\OA-TEMP). The SPAN block has these parameters, all defined with CNST blocks:

High Input:	65.0°F
Low Input:	58.0°F
Output 1:	60.0°F
Output 2:	55.0°F

This process runs periodically when the fan (AHU1\FAN-ST) is on, or when the fan changes from on to off or from off to on. When the outside air temperature is 65.0°F, the CMD block will send a setpoint of 55.0°F to the PIDL object. When the outside air temperature is 58.0°F, the CMD block will send a setpoint of 60.0°F to the PIDL object. Refer to Figure 88.

A conditional period of 00:03:00 minutes is defined for this process, which means it will run once every three minutes, as long as the fan (AHU1\FAN-ST) is on.

Note: All operation blocks in this example must be placed in a process. They include: SPAN, CMD, and PERD.

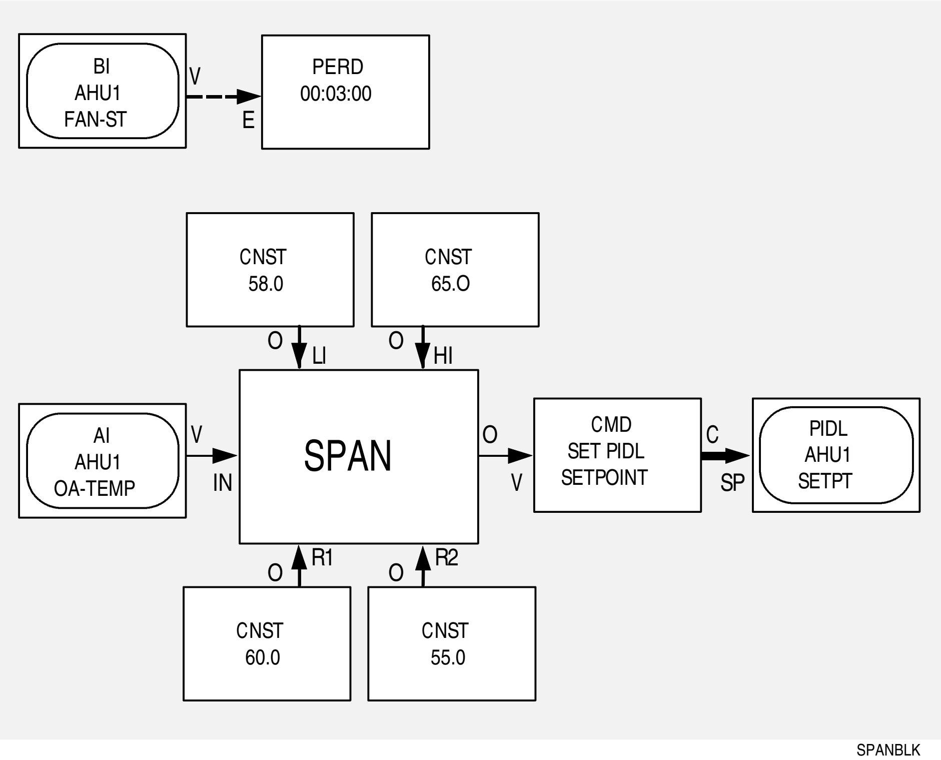


Figure 87: SPAN Block Example

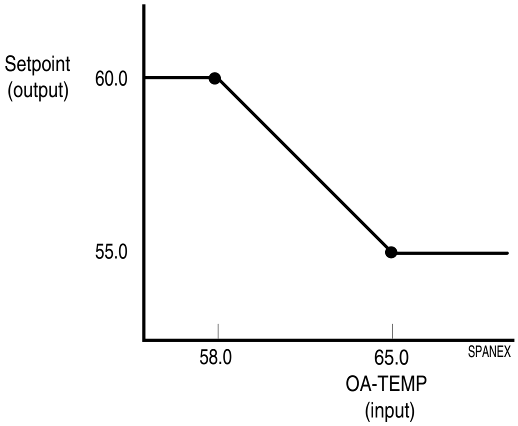


Figure 88: Graphing the SPAN Example

STOP Block

Category

Process Control

Purpose

Stops the current execution of a process.

Details

When this block is executed, the process ends just as if the entire process had completed normally. It executes again when the period of the process (if defined) expires, or when the process is triggered.

Note: The primary application of the STOP block is to prevent unnecessary processing. We do not recommend you use the STOP block for any other application, except under extreme circumstances.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will stop process execution only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not stop process execution. If the Enable input is not connected, process execution will be stopped each time the block executes.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

When this block is executed and the Enable input is False or unreliable, the execution of the process is not stopped.

Example

In this example (Figure 89), the STOP block is used to aid in improving NCU performance by allowing three commands to execute only when necessary.

In the example, if the fan status (AHU1\F-STATUS) is off, the current execution of the process is stopped after the STOP block is executed. However, if the fan status is on, the process executes normally, sending commands to three different BO objects two minutes apart. Also, a PRNT block will send a message to the printer indicating that the exhaust/kitchen fan interlock has been executed.

Note: This example does not require the STOP block to perform the interlock function. It is used for illustrative purposes only to show how NCU performance could be improved. We highly recommend that you do not use the STOP block for this purpose.

A period of 00:00:00 is defined for this process, since it needs to run only when a triggerable attribute of the BI changes.

Note: All operation blocks in this example must be placed in a process. They include: NOT, STOP, PRNT, both WAIT blocks, and the three CMD blocks.

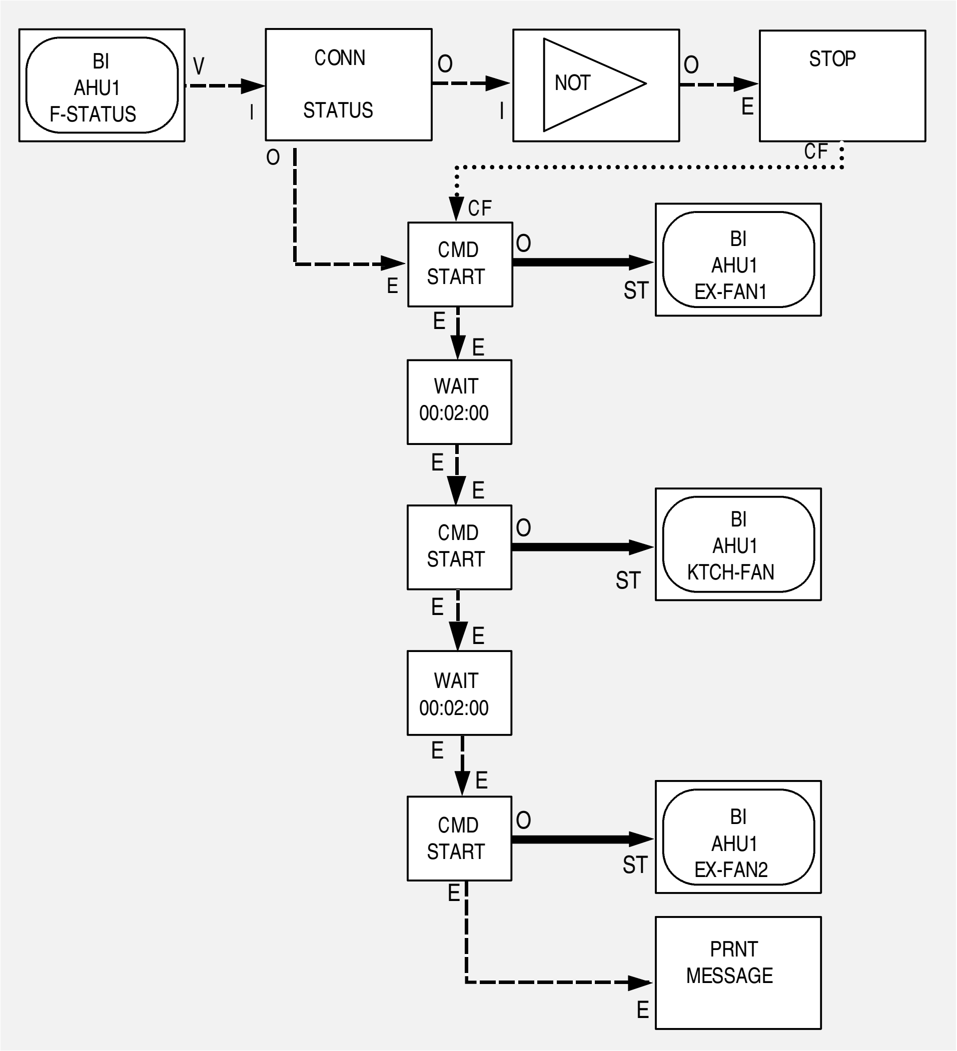


Figure 89: STOP Block Example

SUB (Subtraction) Block

Category

Math

Purpose

Subtracts two inputs using the following equation:

Input 1 - Input 2 = Output

Details

The inputs into this block can be either analog or time. Analog is the block's default, so to connect time data, configure the block for time in its data base template.

If you select analog in the Type field, the block uses real math. If you select time, the block uses time math.
Time math is performed in 24-hour format
(e.g., 3:17:10 - 6:00:00 = 21:17:10).

You can specify a constant value in the template instead of connecting an external input.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, TIME		X			
Input 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
Input 2	INPUT 2	ANA	0.000000	Real	N	X	X		I2
	TIME IN2	TIM	00:00:00	Time	N	X	X		T2
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	OUTPUT	TIM	00:00:00	Time	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the SUB block is unreliable if either of its inputs is unreliable.

Example

In this example (Figure 90), the SUB block is part of a tonnage calculation. The standard tonnage equation is:

TONS = ΔT (GPM / 24)

where:

ΔT = Delta T, change in chilled water return temperature and chilled water supply temperature.

GPM = gallons per minute, a measured value.

To convert this equation into a GPL diagram, three math blocks are required:

DIV: Divides GPM by 24

SUB: Subtracts two chilled water temperatures to find ΔT

MUL: Multiplies ΔT by GPM/24

The example in Figure 90 performs these calculations:

1. GPM is divided by 24. Input 1 of the DIV block is GPM, which is from an AI object (AHU1\FLOW1) that reports gallons per minute. Input 2 of the DIV block is 24, from a CNST block.
2. ΔT is calculated. Input 1 of the SUB block is chilled water return temperature (AHU1\CHWR-T). Input 2 is the chilled water supply temperature (AHU1\CHWS-T).
3. The result of the DIV and SUB blocks is multiplied. Input 1 of the MUL block is from the DIV block. Input 2 is from the SUB block. A command block reads the calculated value, and sends it to an AD object (AHU1\TONS), whose value can be read on a summary at an operator device.

A period of 00:02:00 is defined for this process, which means it will execute once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: DIV, MUL, CMD, and SUB.

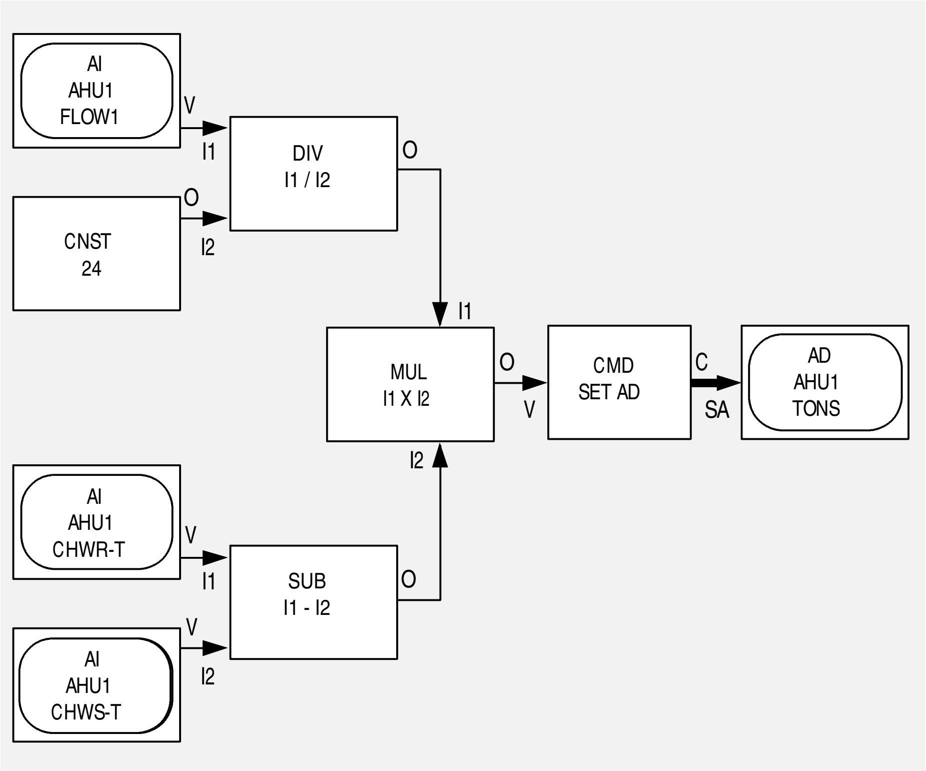


Figure 90: SUB Block Example

SVAR (Shared Variable) Block

Category

Data

Purpose

Names a variable that is used to pass data between processes, which can be in separate control strategy files for the same Network Control Module (NCM).

Details

The SVAR block has these primary characteristics:

- Can be configured for analog, binary, or time data. The default is analog. If you want to connect binary or time data, you must first specify the type of data in the template before making the connection.
- Can be further configured for input only, output only, or input and output. The block will store the value and reliability of the input. The output is the last value and its reliability.
- All shared variable blocks that have the same name and type share the same value, as long as they are in the same NCM.
- SVAR blocks retain their values in case of an NCM power failure (warm start).
- Only one shared variable with the same name can be used in the same process. However, the same shared variable name can be used in multiple processes in a single strategy file.
- SVAR blocks retain their values in case of a power failure (warm start).

When a binary shared variable changes state, the process in which it is read will be triggered, as long as the output line is not exempt from triggers and the variable is reliable.

The name you assign to a SVAR block must not be one of the reserved words or contain invalid symbols. (Refer to *Appendix F: Characters, Symbols, and Reserved Words.*) Also, the first character of a SVAR block must be a letter.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the variable will be assigned the input's value only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the variable will not be assigned. If the Enable input is not connected, the variable will be assigned the input's value each time the SVAR block executes.

Differences Between SAMP, SVAR, and VH Blocks

Though these three blocks are similar, they have the following differences.

Block	Input is Assigned to Output When:
SAMP	Hold input is False and reliable
SVAR	Enable input is True and reliable
VH	Enable input is True and reliable

The SVAR differs from the other blocks in that it can share data between processes on the same NC. In addition, more than one SVAR block with the same name cannot exist in the same process.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name (Variable Name)		STR	Blank	8 Char		X			
Description		STR	Blank	24 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	INPUT	BIN	0	0,1	N		X		I
-	INPUT	ANA	0.000000	Real	N		X		IN
-	TIME IN	TIM	00:00:00	Time	N		X		TI
-	OUTPUT	BIN	0	0,1	N			X	O
-	OUTPUT	ANA	0.000000	Real	N			X	O
-	OUTPUT	TIM	00:00:00	Time	N			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

The reliability of the last block to make the assignment to the SVAR block determines the reliability of the output. The value of the unreliable output is equal to the last reliable output.

Example

The example in Figure 91 shows a SVAR block used in diagrams that are in two different files. In the top diagram, the SVAR block represents economizer mode. The DFCM block compares the outside air temperature (AHU1\OA-TEMP) with the setpoint (AHU1\EC-SETPT). If the outside air temperature is less than the setpoint, given a differential of 2.0°F, economizer is set.

In the bottom diagram, the SVAR block is an input to a SWCH block. If SVAR is True, then a command of 100.0% is sent to the AD object AHU1\MA-DMPCT. If SVAR is False, then a command of 10.0% is sent to AHU1\MA-DMPCT.

A period of 00:02:00 minutes is defined for both processes, which means they will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: DFCM, SWCH, CMD, and both SVAR blocks.

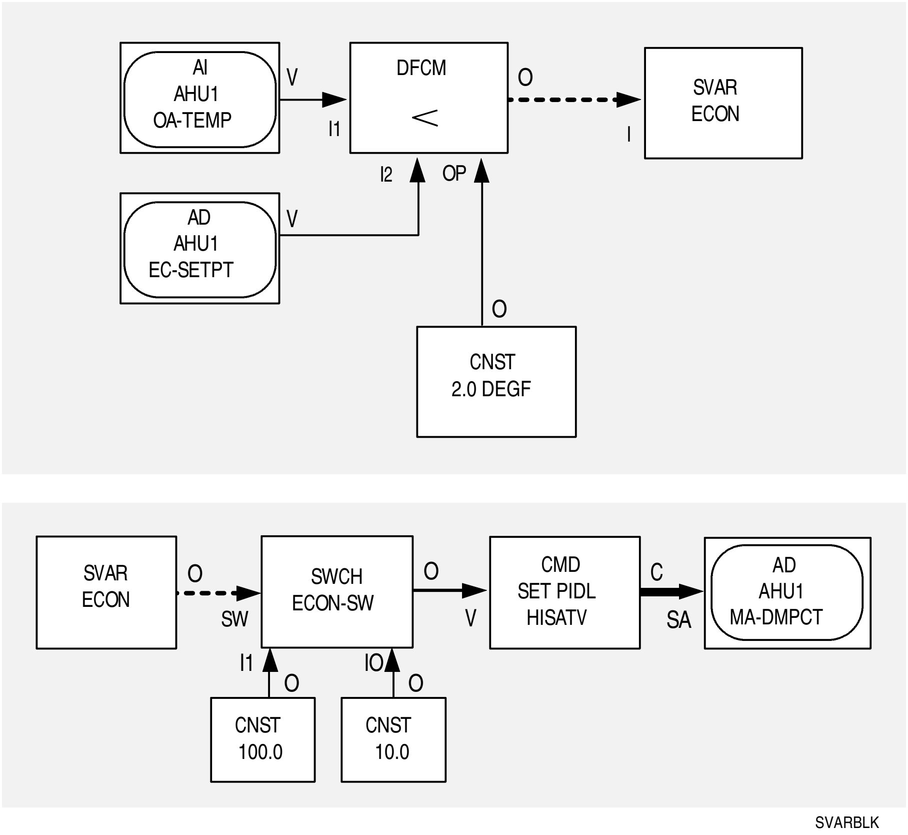


Figure 91: SVAR Block Example

SWCH (Switch) Block

Category

Selectors

Purpose

Chooses between two inputs. The SWCH block also provides an OR function through a binary output called Active output.

Details

You can configure this block to accept analog, binary, or time data. The default is analog. If you want to connect binary or time data, you must first specify the type of data in the template before making the connection.

Also, you can specify constant values for Input 0 and Input 1 instead of connecting external inputs. Enter a constant value in the data base template.

If the Switch input is True, then Input 1 is passed to the output. If the Switch input is False, then Input 0 is the output.

This block has an Active input and Active output, which provide an OR function inside the SWCH block (Figure 92). The inputs to the OR function are Active input and Switch input.

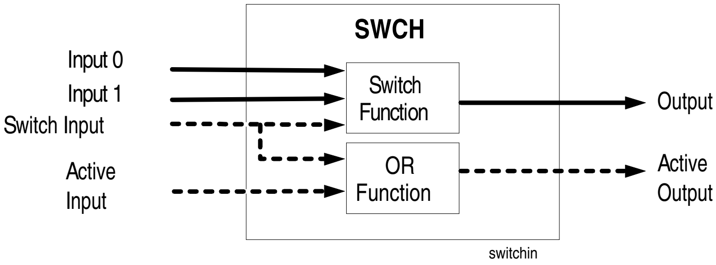


Figure 92: Active In/Active Out Provide OR Function Inside SWCH Block

The following chart describes the OR function:

Active Input	Switch Input	Active Output
True	True	True
True	False	True
False	True	True
False	False	False

If the Active input is not connected, its value is False and reliable.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
INPUT 0	INPUT 0	ANA	0.000000	Real	N	X	X		I0
	INPUT 0	BIN	0	0,1	N	X	X		I0
	TIME IN0	TIM	00:00:00	Time	N	X	X		T0
INPUT 1	INPUT 1	ANA	0.000000	Real	N	X	X		I1
	INPUT 1	BIN	0	0,1	N	X	X		I1
	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
-	OUTPUT	ANA	0.000000	Real	Y			X	O
		BIN	0	0,1	Y			X	O
		TIM	00:00:00	Time	Y			X	O
-	SWCH IN	BIN		0,1	Y		X		S W
-	SWCH OUT	BIN	0	0,1	N			X	S W
-	ACTIVE IN	BIN	0	0,1	N		X		AI
-	ACTIVE	BIN	0	0,1	N			X	A
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if either the selected input and/or the Switch input is unreliable. The Active output is unreliable if either the Switch input or Active input is unreliable.

Example

In this example (Figure 93), the two SWCH blocks are used to change the normal operation of a discharge air PID loop (AHU1\DA-C). During normal operation, the loop will modulate the heating valve based on discharge air. However, when the fan turns off or the outdoor air temperature falls below 35.0°F, PID control is overridden as follows:

- When the fan (AHU1\FAN-STAT) turns off, the CMD block sends a command of 0% to the heating valve (AHU1\H-VLV), closing it.
- When the outdoor air temperature (AHU1\OA-TEMP) is less than 35.0°F, the CMD block sends a command of 100% to the heating valve, opening it.

A period of 00:01:00 is defined for this process, which means it will run once every minute or when a triggerable attribute of the BI changes.

Note: All operation blocks in this example must be placed in a process. They include: DFCM, both SWCH blocks, CMD, and 2CMD.

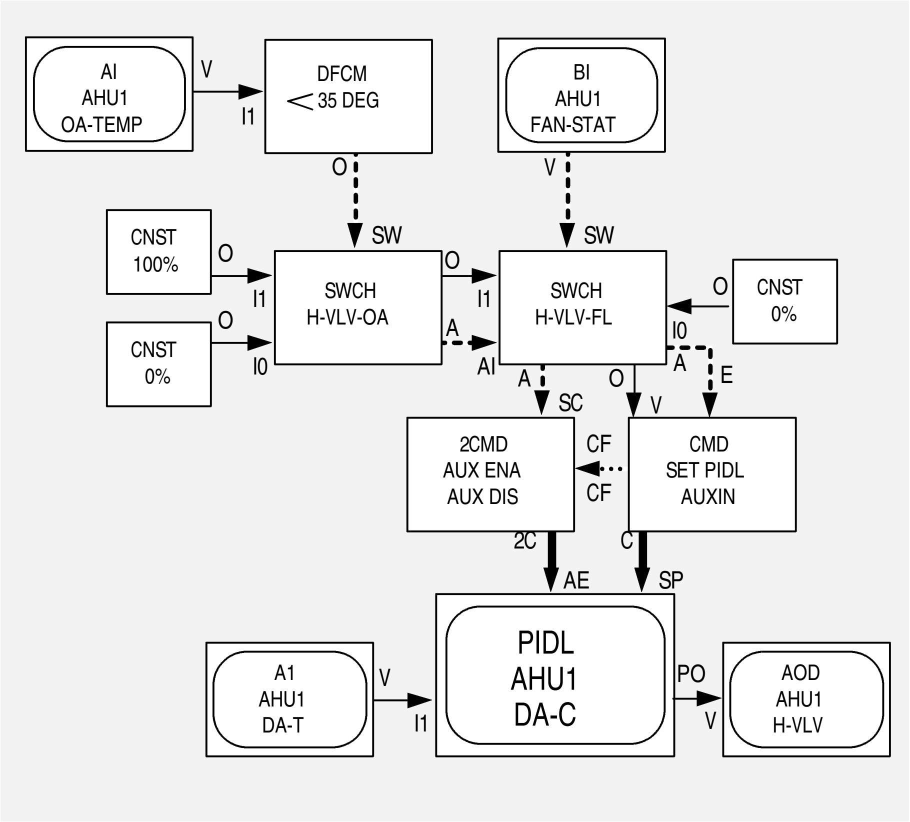


Figure 93: SWCH Block Example

TIME Block

Category

Time

Purpose

Obtains the current date information from the network operating system. The date includes the time, month, day, year, and operating day.

Details

These are the valid ranges for this block:

Time:	00:00:00 to 23:59:59
Month:	1.0 (January) to 12.0 (December)
Day:	1.0 to 31.0
Year:	1989.0 to 2099.0
Today:	1 to 15 (1-7 is Sunday through Saturday, 8-14 is alternate Sunday through Saturday, 15 is holiday)

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	TIME	TIM	00:00:00	TIME	N			X	T
-	TODAY	ANA	1.0	1.0-15.0	N			X	TD
-	DAY	ANA	1.0	1.0-31.0	N			X	D
-	MONTH	ANA	1.0	1.0-12.0	N			X	M
-	YEAR	ANA	1989.0	1989.0-2099.0	N			X	Y
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the TIME block is always reliable.

Example

The TIME block in this example (Figure 94) is used to determine occupancy mode. Two attributes of the BO object, Early Start Time (ES) and Latest Stop Time (LS), are inputs to the COMP blocks. If the current time is greater than or equal to the Early Start Time, and less than or equal to the Latest Stop Time, then occupancy mode (OCC) is set to True.

A period of 00:02:00 is defined for this process, which means it will run once every two minutes.

Note: All operation blocks in this example must be placed in a process. They include: both COMP blocks, AND, TIME, and SVAR.

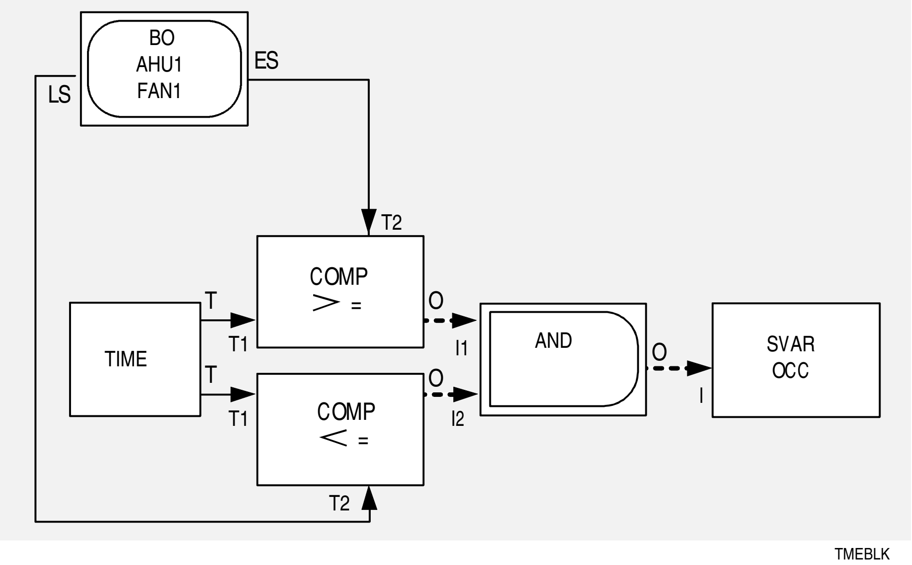


Figure 94: TIME Block Example

TOT (Totalization) Block

Category

Miscellaneous

Purpose

Obtains the totalized value in minutes of an object's attribute as determined by the Totalization feature.

Note: The TOT block *obtains* the totalized value only; it does not define Totalization for the object. Totalization must already be defined for the object (via online generation). For more information, see the *Totalization Technical Bulletin* in the *Metasys Network Technical Manual*.

Details

You can configure this block to obtain totalized data from the current period, the last period, or both periods. You can also configure it to obtain the values of analog or binary attributes. The default is analog. If you want to get totalized data of a binary attribute, you must first specify binary in the template before making the connection.

Note: You cannot use a READ block to read a totalized attribute. Only the object attributes that are available with a direct connection can be inputs to the TOT block.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY		X			
Totalization Typ		TAB	CURRENT	CURRENT LAST, CURRENT AND LAST (See Note)		X			
-	TOT IN	BIN			Y		X		I
-		ANA			Y		X		IN
-	CURTOT	ANA	0.000000	Real	N			X	CT
-	LASTTOT	ANA	0.000000	Real	N			X	LT
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: If CURRENT was chosen in the Totalization Type field, CUR TOT will display in the output connection menu. If LAST was chosen, LAST TOT will be displayed. And, if CURRENT AND LAST was chosen, both CUR TOT and LAST TOT will be displayed.

Reliability

The output of the TOT block is unreliable if there is no response to the request for totalized data, the attribute is not in the Totalization feature data base, or the value from the feature is unreliable. In either case, the input value and the unreliable status will be passed to the output.

Example

In this example (Figure 95), the TOT block obtains the current totalized value in minutes of an AI object attribute (AHU1\OA-TEMP) as determined by the Totalization feature. This value is then sent to a printer via the PRNT block. The message printed is:

```
THE CURRENT TOTALIZED VALUE OF AHU1\OA-TEMP IS
<value>
```

A period of 00:03:00 is defined for this process, which means it will run once every three minutes.

Note: The TOT and PRNT blocks, both operation blocks, must be placed in a process.

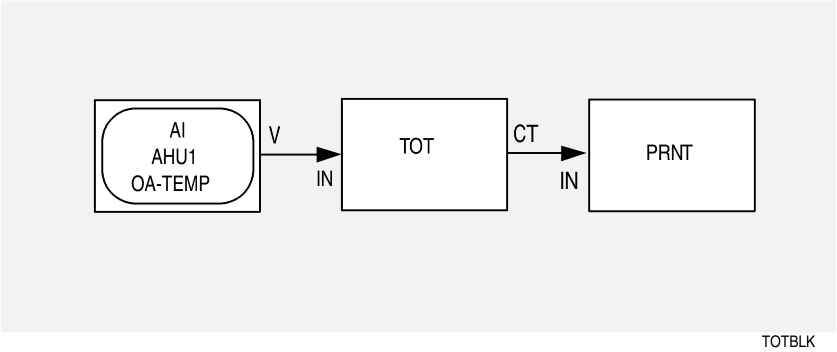


Figure 95: TOT Block Example

TTOR (Time-to-Real) Block

Category

Time

Purpose

Converts a time value in 24-hour format into a real value in minutes or seconds.

Details

The TTOR block can be configured for either minutes or seconds. You select either minutes or seconds in the Resolution tab field. Minutes is the default.

If minutes resolution is chosen, the output value represents a number of minutes. For example:

04:00:00 is converted to 240 minutes.

With minutes resolution, a time value that contains seconds is rounded before it is converted. For example:

04:00:29 is rounded to 04:00:00, then converted to 240 minutes.

04:00:30 is rounded up to 04:01:00, then converted to 241 minutes.

If seconds resolution is chosen, the output value represents a number of seconds. For example:

01:02:03 is converted to 3723 seconds.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Resolution		TAB	MINUTES	MINUTES/ SECONDS		X			
-	TIME IN	TIM		Time	Y		X		TI
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

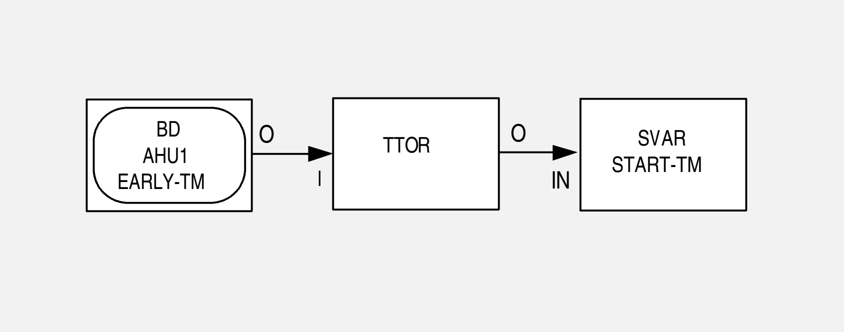
The output of the TTOR block is unreliable if the input is unreliable.

Example

In the example shown in Figure 96, the TTOR block converts the Early Start Time attribute of a BD object (AHU1\EARLY-TM) into a real value in minutes. The real value is read by the SVAR block (START-TM), which is used in an optimal start calculation in another process.

A period of 00:03:00 is defined for this process, which means it will run once every three minutes.

Note: The TTOR and SVAR blocks, both operation blocks, must be placed in a process.



TTORBLK

Figure 96: TTOR Block Example

2CMD (Dual Command) Block

Category

Object Control

Purpose

Sends one of two possible commands to an object block. The dual command changes the object's attribute values and/or invokes an algorithm in the object (e.g., COS analysis). For example, possible commands for a BO object include Start/Stop, Begin Totalization/End Totalization, and Lock/Unlock.

Details

The dual command may have parameter values that must be specified with the command (e.g., Priority). You specify these parameters in the 2CMD block's data base template.

When the 2CMD block is executed, one of two commands is sent to the object (as long as the Edge Trigger is set to No). Command 1 is sent when the Select Command input is True. Command 0 is sent when Select Command input is False.

Command execution is affected by how you configure the Edge Trigger parameter. See the following table.

For Edge Trigger?	Command 1 Sent	Command 0 Sent	No Command Sent
Yes	Only when Select Command input changes from False to True.	Only when Select Command input changes from True to False.	When Select Command input has not changed.
No	Always when Select Command input is True.	Always when Select Command input is False.	N/A

When Edge Trigger is Yes, if the Select Command input changes while the enable input is False, the command will be stored and then issued when the enable input goes to True.

These important factors relate to 2CMD blocks:

- Before you can edit the 2CMD template or enter the block's command parameters, you must connect the 2CMD block to an object block or to a CONN block.
- If you erase the command connection and try to connect to a different command set, you'll notice that the connection menu contains the previous commands only.
- You cannot change the commands selected without first erasing the 2CMD block, then pasting down a new one, and selecting a different command set. However, if this is an origin remote 2CMD block, you may select a different origin block by simply double-clicking left on the 2CMD block (with Connection icon selected) and selecting a different command set. Yet, even in this case, it is best to erase the origin remote connection and re-add it, since the Editor cannot update the line label at the destination block or verify that you selected a valid command for that object. An invalid command is not detected until the file is translated and compiled.

  You may use the 2CMD block to modify integer parameters and attributes. The NCM will convert the analog value to integer by rounding. The rounding rules are:

  0.00 - 0.49: Round off
  0.50 - 0.99: Round up

This block has an Enable input, which provides conditional logic. If this optional input is connected, a command may be issued only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, a command will not be sent. If the Enable input is not connected, a command could be sent each time the 2CMD block executes.

Information Tables

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command (1)			See Note A	See Note A		X			
Command (0)			See Note A	See Note A		X			
Edge Trigger		BIN	Y	Y(Yes), N(No)		X			
-	SEL CMD	BIN	1	0,1	Y		X		SC
-	ENA IN	BIN		0,1	N		X		EN
-	ENA OUT	BIN	See Note B	0,1	N			X	EN
-	DUAL CMD	2CMD			Y			X	2C
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note A: For the particular commands, refer to the following table which indicates the various tables that describe the commands.

Note B: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Table Number	Command(s)
1	AUX_DIS, AUX_ENA, LOC_REP, LOC_TRG, PRC_DIS, PRC_ENA, OFF, ON, RELEASE, RELEASE3, STARTUP, TRIGGER, UNL_REP, UNL_TRG, UNLATCH
2	BEG_TOT, BEG_TRND, END_TOT, END_TRND
3	START, STOP

Table 1

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Command (1/0)		READ ONLY		AUX_DIS		X			AD
				AUX_ENA		X			AE
				LOC_REP		X			LR
				LOC_TRG		X			LT
				PRC_DIS		X			DS
				PRC_ENA		X			EN
				OFF		X			OF
				ON		X			ON
				RELEASE		X			R
				RELEASE 3		X			R3
				STARTUP		X			SU
				TRIGGER		X			TR
				UNL_REP		X			UR
				UNL_TRG		X			UT
				UNLATCH		X			UL

Table 2

Field Name	Connect Name	Type	Default	range	RC	T	I	O	LB
Command (1/0)		READ ONLY		BEG_TOT		X			BT
				BEG_TRND		X			BH
				END_TOT		X			ET
				END_TRND		X			EH
Attribute Name		STR	VALUE	8 Char		X			

Table 3									
Field Name	**Connect Name**	**Type**	**Default**	**Range**	**RC**	**T**	**I**	**O**	**LB**
Command (1/0)		READ ONLY		START		X			ST
				STOP		X			SP
				REL_PRI		X			RP
Priority	CMD1 PRI	INT	7	2,4,5,6,7	N	X	X		P1
Priority	CMD0 PRI	INT	7	2,4,5,6,7	N	X	X		P0

Reliability

If a command parameter is unreliable, the unreliable value and the associated reliability flag are sent with the command to the object. Refer to the reliability information in the *Object Blocks* section of this chapter for details.

If an Enable line is connected and its value is unreliable, neither command will be sent. Also, no commands are sent if the Select Command input is unreliable.

In Figure 97, the 2CMD block sends a Start or Stop command to the BO object depending on which fan is on. If AHU1\FAN1-ST and AHU1\FAN2-ST are both on, the 2CMD block sends a Start command to AHU1\EXH-FAN, but only once. If either AHU1\FAN1-ST or AHU1\FAN2-ST is on (but not both), the 2CMD block sends a Stop command to AHU1\EXH-FAN; again, only once.

A period of 00:00:00 is defined for this process since it needs to run only when a triggerable attribute of one of the BI objects changes.

Note: The AND and 2CMD blocks, both operation blocks, must be placed in a process.

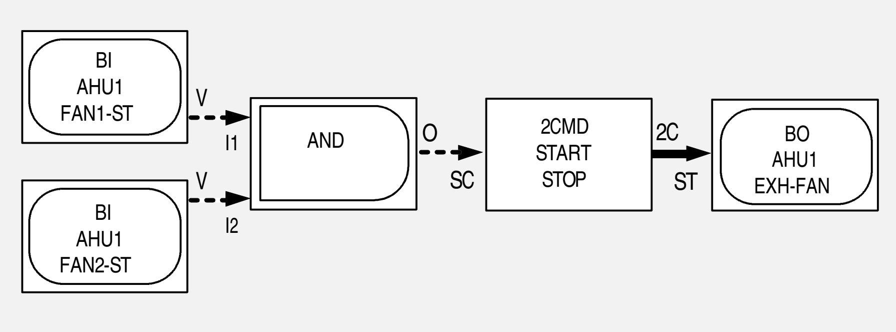


Figure 97: 2CMD Block Example

UNRD (Unreliable Data) Block

Category

Reliability

Purpose

Tests for unreliable data. You can use this block to ensure a block's value is never unreliable, and/or perform some logic based on the unreliable data test.

Details

The UNRD block passes to the output either the input value or a default value. If the input is reliable, the UNRD block passes the input value to the output. If the input is unreliable, the block passes the default value along with its reliability flag. The default value can be specified in the data base template or connected.

The UNRD block has an Unreliable output. The Unreliable output is set to True if the input is unreliable and False if the input is reliable.

The following chart summarizes the output and the Unreliable output.

Input	Output	Unreliable Output
Reliable	Input	False
Unreliable	Default value as specified in the data base template or from a connected input.	True

You may configure this block to act on analog, binary, or time input connections. The default is analog. If you want to connect binary or time data, you must first specify the type of data in the template before making a connection.

The default value can be configured in the data base template or connected.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	8 Char	Blank		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
Default	DEFAULT	BIN	0	0,1	N	X	X		DF
		ANA	0.000000	Real	N	X	X		DF
		TIM	00:00:00	Time	N	X	X		DF
-	INPUT	BIN	0	0,1	Y		X		I
		ANA	0.000000	Real	Y		X		IN
-	TIME IN	TIM	00:00:00	Time	Y		X		TI
-	OUTPUT	BIN	0	0,1	N			X	O
		ANA	0.000000	Real	N			X	O
		TIM	00:00:00	Time	N			X	O
-	UNREL OUT	BIN	0	0,1	N			X	UO
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the UNRD block is reliable, as long as the input is reliable. If the input is unreliable, the output and its reliability will be equal to the Default input. If the Default input is unreliable, the output will be unreliable.

Example

In the example shown in Figure 98, the UNRD block ensures that the value of an AI object called AHU1\OA-TEMP is always reliable. If AHU1\OA-TEMP is reliable, the UNRD block passes its value to the DFCM block. However, if AHU1\OA-TEMP becomes unreliable, the UNRD block sends a default constant of 55.0°F to the DFCM block. Also, the PRNT block that is connected to the UNRD block sends the following message to the printer:

```
OUTSIDE AIR TEMPERATURE UNRELIABLE. USING 55.0 DEGF.
```

This process also sets economizer mode: If AHU\OA-TEMP is less than the setpoint (AHU1\SETPT) with an applied differential, then economizer mode is set. Otherwise, the dampers are set to the minimum position (logic in another diagram).

A period of 00:03:00 is defined for this process, which means it will run once every three minutes.

Note: All operation blocks in this example must be placed in a process. They include: UNRD, DFCM, SVAR, and PRNT.

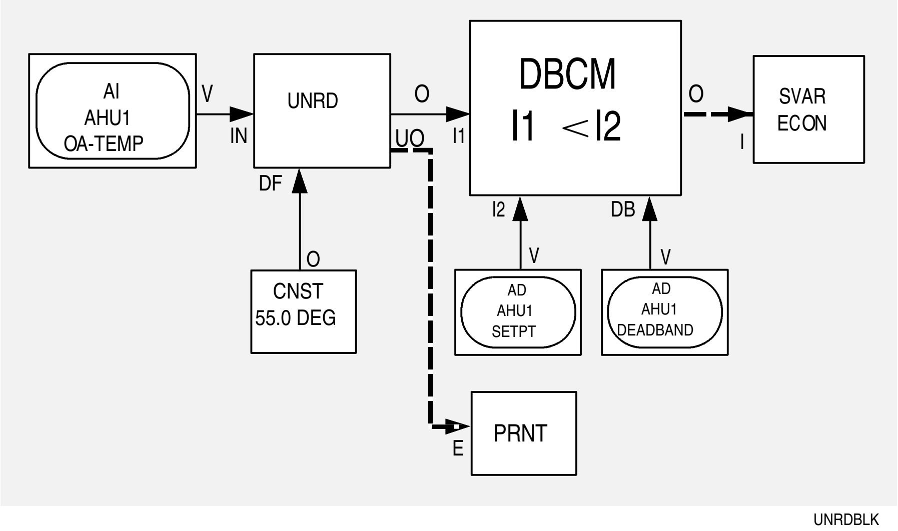


Figure 98: UNRD Block Example

USER Block

Category

Process Control

Purpose

Creates a generic operation block. You would make a USER block if the function you need to perform is not available with standard function blocks or compounds, or can be accomplished more efficiently with a USER block. You need to know JC-BASIC programming in order to create a USER block. Refer to the *JC-BASIC Programmer's Manual*.

Note: We recommend that you create a USER block only when the desired functionality cannot be achieved by using one or a combination of the standard function blocks and compounds.

Details

The USER block requires that you perform steps both outside of GPL and within GPL.

Outside of GPL

Using an ASCII text editor, write the macro file for the USER block, and store the file on disk. Assign the file a .MAC extension. For detailed instructions, refer to the section *Writing a USER Block File* in *Appendix E: External Functions.*

Within GPL

Follow these steps:

1. Paste down the USER block in the work area.
2. Click on the block with the Query mode selected to display the block's data base template. In the template, specify the root file name of the macro file for the USER block. Do not enter the .MAC extension.

3. Specify the number of inputs and outputs that are part of the USER block macro file. The block accepts the following inputs: 0 to 8 analog, 0 to 8 binary, and 0 to 4 time. The block accepts these outputs: 0 to 8 analog, 0 to 8 binary, 0 to 4 time, 0 to 8 single command, and 0 to 8 writes.
4. Save the template information by pressing F10.
5. Place the USER block inside a process compound, save the compound to disk, and save the compound to a strategy file. (By saving the USER block as a compound, you can use it again for another application whenever desired.)
6. Check the file with Expert Checker. Translate and compile.

Note: The Translator looks for your USER block macro file in the CUSTUMAC directory. If it cannot find it there, the Translator looks in the JCIUMAC file. What this means is if the name you assign a USER block macro file under the CUSTUMAC directory is the same as a block macro file under the JCIUMAC directory, the Translator will use the macro file in the CUSTUMAC directory.

Other Details

The name you assign to a USER block must not be one of the reserved words or contain invalid symbols. (Refer to *Appendix F: Characters, Symbols, and Reserved Words*.)

With the Query mode selected, you can view the macro file of the USER block by double-clicking on the block. The Editor can display the first 30 pages of the file. Use the Page Up and Page Down keys to scroll through the file. Use the Esc key or click left to display the previous diagram.

You cannot edit or print the macro file with the GPL Editor. You'll need the text editor to perform those tasks.

The USER block has connections called Command and Write. You can use these to command or write to an object without using a CMD or WRIT block. You cannot fan-out from these connections.

You can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

Once programmed and defined, you can use the USER block in any control strategy, just as any other operation block. Also, you can use multiple USER blocks in the same strategy.

Note: USER blocks are not simulated.

Information Table (First Screen)

Note: The number of inputs displayed will depend on the configured number of inputs on the template's first page.

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Type/File Name		STR	USER	8 Char		X			
Block Name		STR	Blank	8 Char		X			
Description		STR	Blank	24 Char		X			
Analog Inputs		INT	4	0-8		X			
Binary Inputs		INT	1	0-8		X			
Time Inputs		INT	1	0-4		X			
Analog Outputs		INT	1	0-8		X			
Binary Outputs		INT	1	0-8		X			
Time Outputs		INT	1	0-4		X			
Command Outputs		INT	1	0-8		X			
Write Outputs		INT	1	0-8		X			

Information Table (Second Screen)

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Input 1	ANA IN1	ANA	0.000000	Real	N	X	X		A1
Input 2	ANA IN2	ANA	0.000000	Real	N	X	X		A2
Input 3	ANA IN3	ANA	0.000000	Real	N	X	X		A3
Input 4	ANA IN4	ANA	0.000000	Real	N	X	X		A4
Input 5	ANA IN5	ANA	0.000000	Real	N	X	X		A5
Input 6	ANA IN6	ANA	0.000000	Real	N	X	X		A6
Input 7	ANA IN7	ANA	0.000000	Real	N	X	X		A7
Input 8	ANA IN8	ANA	0.000000	Real	N	X	X		A8

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Input 1	BIN IN1	BIN	0	0,1	N	X	X		B1
Input 2	BIN IN2	BIN	0	0,1	N	X	X		B2
Input 3	BIN IN3	BIN	0	0,1	N	X	X		B3
Input 4	BIN IN4	BIN	0	0,1	N	X	X		B4
Input 5	BIN IN5	BIN	0	0,1	N	X	X		B5
Input 6	BIN IN6	BIN	0	0,1	N	X	X		B6
Input 7	BIN IN7	BIN	0	0,1	N	X	X		B7
Input 8	BIN IN8	BIN	0	0,1	N	X	X		B8
Input 1	TIME IN1	TIM	00:00:00	Time	N	X	X		T1
Input 2	TIME IN2	TIM	00:00:00	Time	N	X	X		T2
Input 3	TIME IN3	TIM	00:00:00	Time	N	X	X		T3
Input 4	TIME IN4	TIM	00:00:00	Time	N	X	X		T4
-	ANA OUT1	ANA	0.000000	Real	N			X	A1
-	ANA OUT2	ANA	0.000000	Real	N			X	A2
-	ANA OUT3	ANA	0.000000	Real	N			X	A3
-	ANA OUT4	ANA	0.000000	Real	N			X	A4
-	ANA OUT5	ANA	0.000000	Real	N			X	A5
-	ANA OUT6	ANA	0.000000	Real	N			X	A6
-	ANA OUT7	ANA	0.000000	Real	N			X	A7
-	ANA OUT8	ANA	0.000000	Real	N			X	A8
-	BIN OUT1	BIN	0	0,1	N			X	B1
-	BIN OUT2	BIN	0	0,1	N			X	B2
-	BIN OUT3	BIN	0	0,1	N			X	B3

Continued on next page . . .

Field Name (Cont.)	connect Name	Type	Default	Range	RC	T	I	O	LB
-	BIN OUT4	BIN	0	0,1	N			X	B4
-	BIN OUT5	BIN	0	0,1	N			X	B5
-	BIN OUT6	BIN	0	0,1	N			X	B6
-	BIN OUT7	BIN	0	0,1	N			X	B7
-	BIN OUT8	BIN	0	0,1	N			X	B8
-	TIME OUT1	TIM	00:00:00	Time	N			X	T1
-	TIME OUT2	TIM	00:00:00	Time	N			X	T2
-	TIME OUT3	TIM	00:00:00	Time	N			X	T3
-	TIME OUT4	TIM	00:00:00	Time	N			X	T4
-	COMMAND1	CMD			N			X	C1
-	COMMAND2	CMD			N			X	C2
-	COMMAND3	CMD			N			X	C3
-	COMMAND4	CMD			N			X	C4
-	COMMAND5	CMD			N			X	C5
-	COMMAND6	CMD			N			X	C6
-	COMMAND7	CMD			N			X	C7
-	COMMAND8	CMD			N			X	C8
-	WRITE1	CMD			N			X	W1
-	WRITE2	CMD			N			X	W2
-	WRITE3	CMD			N			X	W3
-	WRITE4	CMD			N			X	W4
-	WRITE5	CMD			N			X	W5
-	WRITE6	CMD			N			X	W6
-	WRITE7	CMD			N			X	W7
-	WRITE8	CMD			N			X	W8
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

You must manage the passing of unreliable data entirely in the USER block’s macro file.

Example

This example demonstrates:

- Linking function blocks as inputs to and outputs from a USER block
- Linking a USER block's macro file (.MAC) to its associated block in GPL
- Defining and using arrays in a USER block

In this example (Figure 99), the USER block monitors the space temperature of occupied and unoccupied modes during the day. It stores 14 days of the highest and lowest values read from the space temperature sensor (AHU1\SPACETMP) during occupied and unoccupied modes (AHU1\OCC-UNOC). These values are stored in four different SVAR blocks (OCC_HI,OCC_LO,UNOCC_HI,UNOCC_LO).

A period of 01:00:00 is defined for this process, which means it will run once every hour. The BD object (AHU1\OCC-UNOC) connected to the USER block as a binary input type is a trigger for this process. This trigger will cause the process to execute when the mode changes from occupied to unoccupied, or vice versa, thereby capturing the initial space temperature of the mode.

In the USER block data base template, the name of the macro file (TEMP_MON) is entered in the Type/File Name field. Also, the following inputs and outputs are configured:

Analog inputs: 1
Binary inputs: 1
Time inputs: 0

Analog outputs: 4
Binary outputs: 0
Time outputs: 0
Commands: 0
Writes: 0

Note: All operation blocks in this example must be placed in a process. They include: USER and each SVAR block.

The macro file for this USER block is stored in the C:\CUSTUMAC directory under a file named: TEMP_MON.MAC (temperature monitoring). The following is the contents of the file:

```
REM The following array variables (OCC_H_TEMP,
REM UNOCC_H_TEMP, OCC_L_TEMP, and
REM UNOCC_L_TEMP) are defined as shared
REM variables instead of local variables
REM because on a warmstart, shared
REM variables maintain their values.

REM The following arrays (OCC_H_TEMP(14),
REM UNOCC_H_TEMP(14), OCC_L_TEMP(14),
REM and UNOCC_L_TEMP(14)) are for storage
REM of the occupied and unoccupied, high
REM and low space temperatures.

SHARED OCC_H_TEMP(14),UNOCC_H_TEMP(14),\
    OCC_L_TEMP(14),UNOCC_L_TEMP(14)

REM The following variables (DAY_POINTER%
REM and DAYY%) are defined as shared
REM variables instead of local variables
REM because on a warmstart, shared
REM variables maintain their values.

REM The shared variable DAY_POINTER% points
REM at day number 1 thru 14. The shared
REM variable DAYY% increments DAY_POINTER%.

SHARED DAY_POINTER%,DAYY%

REM The following lines initialize DAYY% and
REM then place initial values into the four arrays.

REM DAY is an integer function that provides
REM the same value as the Day output of the
REM TIME block.

IF DAYY% <> DAY THEN
    DAYY% = DAY
    IF DAY_POINTER% =14 \
        THEN DAY_POINTER% = 1 \
        ELSE DAY_POINTER%=DAYPOINTER% + 1
    OCC_H_TEMP(DAY_POINTER%)=-999.9
    UNOCC_H_TEMP(DAY_POINTER%)=-999.9
    OCC_L_TEMP(DAY_POINTER%)=999.9
    UNOCC_L_TEMP(DAY_POINTER%)=999.9
END IF

REM The next lines read in the space
REM temperature and occupied value
REM and compare the space
REM temperature
REM against the values in the arrays.

REM SPTEMP and OCC? are local variables.
REM They are used to provide consistent
REM values for the inputs and reduce the
REM number of accesses to the objects.

SPTEMP=[ANA IN1]
OCC?=[BIN IN1]
```

```
REM If either input is unreliable, the
REM temperature is not used to determine
REM the highest and lowest temperature.
REM If both inputs are reliable, compare
REM the space temperature to today’s
REM array values for the current building
REM mode, and store in the array value if a
REM new high and low is found.

IF NOT (UNRELIABLE(OCC?) OR UNRELIABLE(SPTEMP)) THEN
    IF OCC? THEN
        IF SPTEMP > OCC_H_TEMP(DAY_POINTER%) THEN \
            OCC_H_TEMP(DAY_POINTER%) = SPTEMP
        IF SPTEMP < OCC_L_TEMP(DAY_POINTER%) THEN \
            OCC_L_TEMP(DAY_POINTER%) = SPTEMP

    ELSE
        IF SPTEMP > UNOCC_H_TEMP(DAY_POINTER%) THEN \
            UNOCC_H_TEMP(DAY_POINTER%) = SPTEMP
        IF SPTEMP < UNOCC_L_TEMP(DAY_POINTER%) THEN \
            UNOCC_L_TEMP(DAY_POINTER%) = SPTEMP
    END IF
END IF

REM The following are four local variables:
REM OCC_HIGH, OCC_LOW, UNOCC_HIGH,
REM and UNOCC_LOW. They are used to collect
REM the highest and lowest values for the
REM occupied and unoccupied modes when checking
REM the 14 element arrays.

OCC_HIGH = OCC_H_TEMP(1)
OCC_LOW =  OCC_L_TEMP(1)
UNOCC_HIGH = UNOCC_H_TEMP(1)
UNOCC_LOW = UNOCC_L_TEMP(1)

REM Search the array for the highest and
REM lowest occupied and unoccupied space
REM temperatures.

FOR I% = 2 TO 14
    IF OCC_HIGH < OCC_H_TEMP(I%) THEN \
        OCC_HIGH = OCC_H_TEMP(I%)
    IF OCC_LOW > OCC_L_TEMP(I%) THEN \
        OCC_LOW = OCC_L_TEMP(I%)
    IF UNOCC_HIGH < UNOCC_H_TEMP(I%) THEN \
        UNOCC_HIGH = UNOCC_H_TEMP(I%)
    IF UNOCC_LOW > UNOCC_L_TEMP(I%) THEN \
        UNOCC_LOW = UNOCC_L_TEMP(I%)
NEXT I%

REM Assign the highest and lowest occupied
REM and unoccupied space temperatures
REM to the four respective outputs.

[ANA OUT1] = OCC_HIGH
[ANA OUT2] = OCC_LOW
```

```
[ANA OUT3] = UNOCC_HIGH
[ANA OUT4] = UNOCC_LOW
```

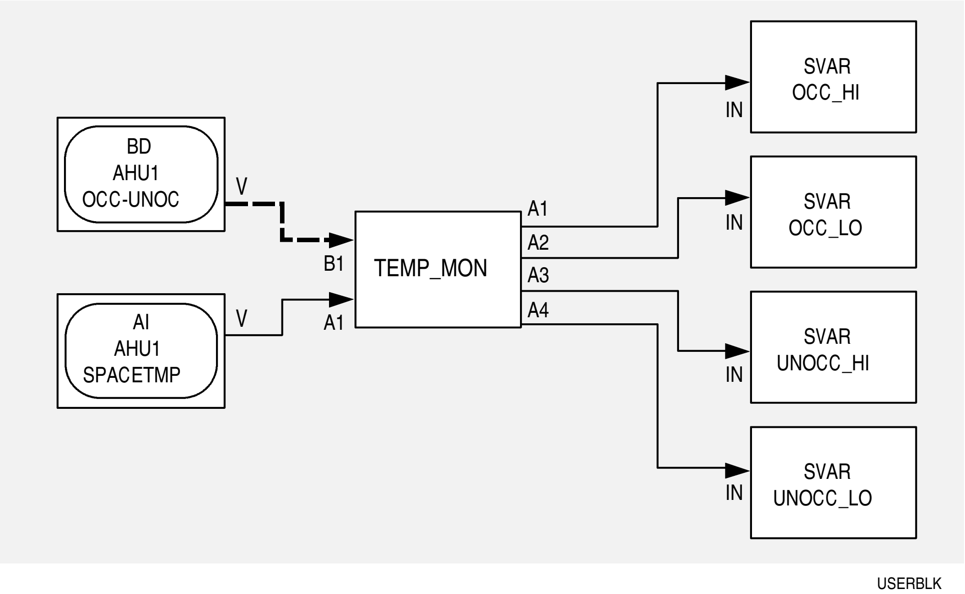


Figure 99: USER Block Example

VH (Value Holder) Block

Category

Data

Purpose

Reads and holds a value that can be read at a later time. The VH block is used in applications such as Average Over Time, Saving Network Reads, and First-In-First-Outs (FIFO). This block is also used to store a value until a specific condition occurs, at which time a new value is read and stored.

Details

The VH block reads and stores the value of the input. The VH block either assigns the input value to the output, or maintains the last assigned value as the output, depending on the Enable input. When the Enable input is True and reliable, the input value is assigned to the output. When the Enable input is False or unreliable, the input value is not assigned to the output. Instead, the last assigned value is maintained as the output. If the Enable input is not connected, the input is always assigned to the output. If the input has never been assigned to the output, the output is zero (0/0.0/00:00:00) and reliable.

The VH block can be configured for analog, binary, or time values.

Enable Input

The following table describes how the Enable input affects the output.

Enable Input	Output	Reliability of Output
True and reliable	Input	Input's reliability
False or unreliable	Last assigned value	Input's reliability at time of last assignment
Enable not connected	Input	Input's reliability

Differences Between SAMP, SVAR, and VH Blocks

Though these three blocks are similar, they have the following differences.

Block	Input is Assigned to Output When:
SAMP	Hold input is False and reliable
SVAR	Enable input is True and reliable
VH	Enable input is True and reliable

The SVAR differs from the other blocks in that it can share data between processes on the same NC. In addition, more than one SVAR block with the same name cannot exist in the same process.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG/ BINARY/ TIME		X			
	INPUT	ANA		Real	Y		X		IN
	OUTPUT	ANA	0.000000	Real	Y			X	O
	INPUT	BIN		0,1	Y		X		I
	OUTPUT	BIN	0	0,1	Y			X	O
	TIME IN	TIM		TIME	Y		X		TI
	OUTPUT	TIM	00:00:00	TIME	Y			X	O
	ENA IN	BIN		0,1	N		X		E
	ENA OUT	BIN	See Note	0,1	N			X	E
	CONTROL	CTL			N		X		CF
	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

The reliability of the output depends on the reliability of the last input value assigned to the output, at the time of the assignment. If the input is reliable at the time of the assignment, the output is reliable. Conversely, if the input is unreliable at the time of the assignment, the output is unreliable. If the input has never been assigned to the output, the output is zero (0/0.0/00:00:00) and reliable.

Example

In the following example (Figure 100), the VH blocks are used to average a value over time. Each time the process runs, the value of the VH2 block is shifted into VH3, the value of VH1 is shifted into VH2, and the value of the AI is shifted into VH1. The AVG block averages the three values and sends the result to the AD block.

A period of 00:00:10 seconds is defined for this process, which means it will run once every ten seconds.

Note: The VH, AVG, and CMD blocks, all operation blocks, must be placed in a process.

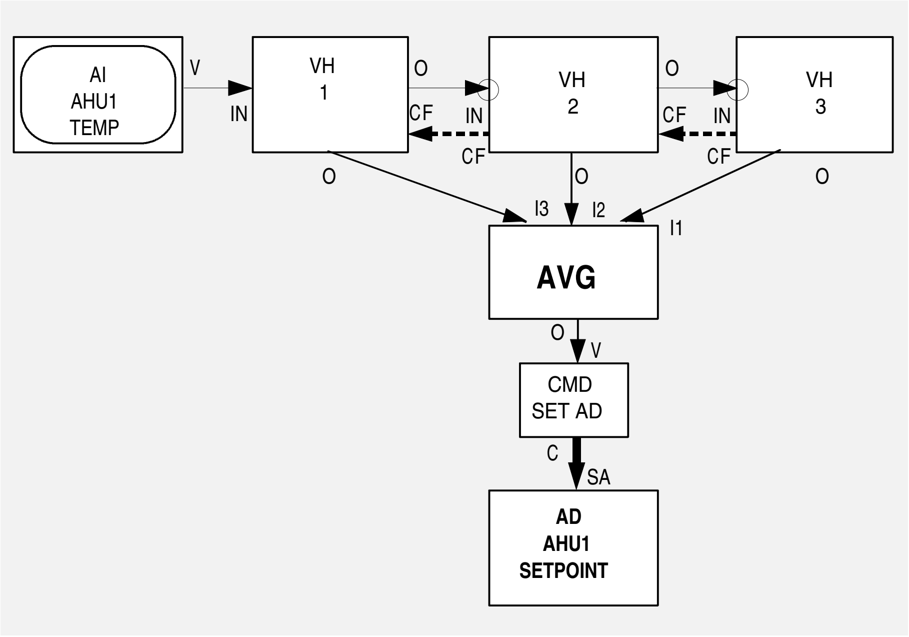


Figure 100: VH Block Example

WAIT Block

Category

Process Control

Purpose

Postpones the full execution of a process for a period of time. For example, the WAIT block can be used inside a Restart process to postpone the start of equipment. The process that is affected is the one in which the WAIT block is contained and executed.

Details

This block has an optional Enable input, which provides conditional logic.

When this block is executed and the Enable input is True, the process that contains the WAIT block is placed in a Waiting state. The NCM Interpreter then goes on to execute other processes. When the wait timer expires, the process resumes execution at the block after the WAIT block. When all blocks in the process have finished execution (or when a STOP block is encountered), the process period timer is set. The actual time between process executions is the sum of the process period timer, the wait timer, and the process execution time. (For details, refer to *PERD (Period) Block* in this chapter.) If the process is triggered before the wait timer has expired, the wait timer is canceled and the process executes from its beginning.

When this block is executed and the Enable input is False (reliable or unreliable) or True and unreliable, the process is not placed in a Waiting state.

Note: Do not specify 00:00:00 for wait timer. This will not provide a wait and will cause a runtime error.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Time	TIME IN	TIM	00:00:05	Time > 00:00:00	N	X	X		TI
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

The WAIT block still executes even if the Time input is unreliable, but will not execute if the Enable input is unreliable.

Example

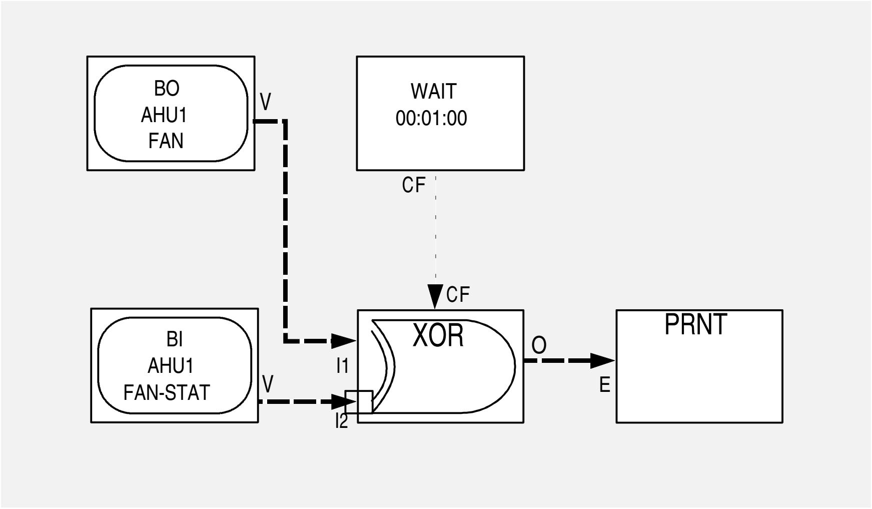
In this example (Figure 101), a WAIT block waits for one minute to verify a fan has actually started. When the fan (AHU1\FAN) is commanded to on, the process waits one minute. Then, it checks to see if the fan (AHU1\FAN-STAT) has indeed started. If the fan has not started, the PRNT block sends the following message to the printer:

```
AHU1\FAN DID NOT START
```

A control flow line is drawn from the WAIT block to the XOR block to make sure the wait is executed before the XOR block. The BI connection to the XOR block is exempted from triggers because only the BO object, not the BI object, should be allowed to trigger the process.

No period is defined for this process, since it only needs to run when the VALUE attribute of the BO object changes.

Note: All operation blocks in this example have to be placed in a process. They include: PRNT, XOR, and WAIT.



WAITBLK

Figure 101: WAIT Block Example

WBDP (Wet Bulb Dew Point) Block

Category

Psychrometric Equations

Purpose

Calculates wet bulb temperature based on dry bulb temperature, dew point temperature, and barometric pressure.

Details

The wet bulb calculation can be done in either English or Metric units.

Valid ranges for the calculated Wet Bulb Temperature:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Dew Point Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Barometric Pressure: 15.00 to 32.00 in. WG (0.51 to 1.08 bar)

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH, METRIC		X			
Barometric Press.	BARO PRES	ANA	29.00000	Real	N	X	X		BP
-	DRY BULB	ANA		Real	Y		X		DB
-	DEW POINT	ANA		Real	Y		X		DP
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the Dry Bulb Temperature input, Dew Point Temperature input, or Barometric Pressure input is unreliable. Also, if any of these values is less than the lowest part of their range, enthalpy is calculated using the lowest range value (e.g., for dry bulb temperature, -147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if either of these values is greater than the highest part of their range, enthalpy is calculated using the highest range value (e.g., for dry bulb temperature, -391.9°F), the output is unreliable, and a runtime error message is generated.

A runtime error message is also generated if dew point temperature is greater than dry bulb temperature.

Example

The following example (Figure 102) contains a WBDP block that calculates wet bulb temperature of the outside air based on dew point and outside air dry bulb temperature. It uses a barometric pressure of 29.00 in. Hg (specified in WBDP template).

The WBDP block takes as inputs the outside air temperature (AHU1\OA-TEMP) and the outside air dew point (AHU1\OA-DWPT). These values are entered into the WBDP block. The resultant value is read by the CMD block, which commands an AD object (AHU1\OA-WBDP) to this value.

A period of 00:02:00 is specified for this process, which means it will run once every two minutes.

Note: The WBDP and CMD blocks, both operation blocks, must be placed in a process.

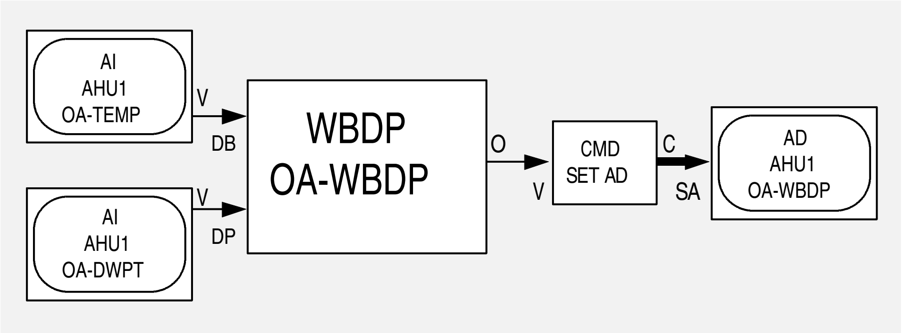


Figure 102: WBDP Block Example

WBRH (Wet Bulb Relative Humidity) Block

Category

Psychrometric Equations

Purpose

Calculates wet bulb temperature based on dry bulb temperature, relative humidity, and barometric pressure.

Details

The wet bulb calculation can be done in either English or Metric units.

Valid ranges for the calculated Wet Bulb temperature:

Dry Bulb Temperature: -147.9 to 391.9°F (-99.9 to 199.9°C)

Relative Humidity: 0 to 100% RH

Note: GPL converts relative humidity to integer by rounding (e.g., 10 to 10.49999 = 10; 10.50000 to 10.99999 = 11).

Barometric Pressure: 15.00 to 32.00 in. WG (0.51 to 1.08 bar)

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
Units		TAB	ENGLISH	ENGLISH, METRIC		X			
Barometric Press.	BARO PRES	ANA	29.00000	Real	N	X	X		BP
-	DRY BULB	ANA		Real	Y		X		DB
-	REL HUMID	ANA		Real	Y		X		RH
-	OUTPUT	ANA	0.000000	Real	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of this block is unreliable if the Dry Bulb Temperature input, Relative Humidity input, or Barometric Pressure input is unreliable. Also, if any of these values are less than the lowest part of their range, enthalpy is calculated using the lowest range value (e.g., for dry bulb temperature, -147.9°F), the output is unreliable, and a runtime error message is generated. Similarly, if either of these values is greater than the highest part of their range, enthalpy is calculated using the highest range value (e.g., for dry bulb temperature, -391.9°F), the output is unreliable, and a runtime error message is generated.

Note: The output of this block is unreliable if the internally calculated dew point value is outside the range of -80.0 to 150°F (-60 to 70°C). (Internal dew point value is not provided to the programmer.)

Example

The following example (Figure 103) contains a WBRH block that calculates wet bulb temperature of the outside air based on relative humidity and outside air dry bulb temperature. It uses a barometric pressure of 29.00 in. Hg (specified in WBRH template).

The block takes as inputs the outside air temperature (AHU1\OA-TEMP) and the outside air relative humidity (AHU1\OA-RH). These values are entered into the WBRH block. The resultant value is read by the CMD block, which commands an AD object (AHU1\OA-WBRH) to this value.

A period of 00:02:00 is specified for this process, which means it will run once every two minutes.

Note: The WBRH and CMD blocks, both operation blocks, must be placed in a process.

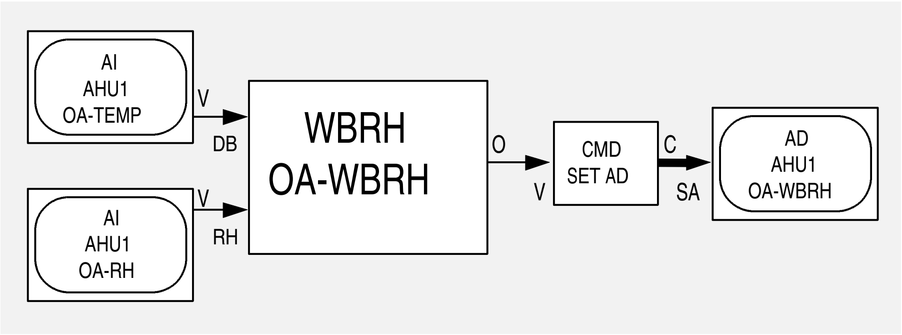


Figure 103: WBRH Block Example

WRIT (Write Attribute) Block

Category

Object Control

Purpose

Modifies an attribute of an object block. Its purpose is to write to those attributes that do not appear in the CMD and 2CMD block connection menus. For a list of all writable attributes for each object, refer to *Appendix G: Attributes.*

Details

You can configure this block for analog (the default), binary, or time attributes. If you want to write a binary or time attribute, you must first configure the block by specifying the type of attribute in the template before making a connection.

Note: You may use this block to modify an integer attribute. To do so, configure the block as analog. GPL will convert the analog value to integer by rounding. The rounding rules are:

0.00 - 0.49: Round off
0.50 - 0.99: Round up

You can specify a constant value for an input instead of connecting an external input. Enter a constant value in the data base template.

You may reassign an origin remote WRIT block to a different write attribute (or even change it to a command or dual command) that is available remotely. To do so, double-click left on the remote WRIT block (with Connection icon selected) and select the attribute or command. Be aware that when you use this method, the Editor cannot update the line label at the destination block or verify that you selected a valid attribute/command for the object. An invalid selection is not detected until the file is translated and compiled.

This block has an Enable input, which provides conditional logic. If this optional input is connected, the block will write an attribute only when the Enable input is True and reliable. If the Enable input is False (reliable or unreliable), or True and unreliable, the block will not write an attribute. If the Enable input is not connected, the attribute is written each time the WRIT block executes.

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Attribute Name (Block Name)		STR	Blank	8 Char		X			
Type		TAB	ANALOG	ANALOG, BINARY, TIME		X			
-	ENA IN	BIN		0,1	N		X		E
-	ENA OUT	BIN	See Note	0,1	N			X	E
INPUT	INPUT	BIN	0	0,1	N	X	X		I
		ANA	0.000000	Real	N	X	X		IN
-	TIME IN	TIM	00:00:00	Time	N	X	X		TI
-	WRITE	WRIT			Y			X	WR
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Note: The ENA OUT can be connected only if the ENA IN is connected. The default is 0.

Reliability

If the input into the WRIT block is unreliable, the block will not write to the attribute.

Example

In the following example (Figure 104), the WRIT block changes the value of the Unrel Dflt Resp (Unreliable Default Response) parameter of the PIDL object (defined in PIDL data base template) when the PIDL detects unreliable data. If the BI object (AHU1\OCC) reports that the building is in occupied mode, and the PIDL object detects unreliable data, the PIDL will output a value of 100, which will command the cooling valve (AHU1\CLG-VLV) to 100% open. If the BI object reports unoccupied mode, and the PIDL object detects unreliable data, the PIDL will output a value of 0, which will command the cooling valve (AHU1\CLG-VLV) to 0% open.

Note: This example requires that the Select Unrl Dflt parameter in the PIDL template is answered Y (Yes).

No period is specified for this process, since it only needs to run when triggered by the BI object.

Note: The SWCH and WRIT blocks, both operation blocks, must be placed in a process.

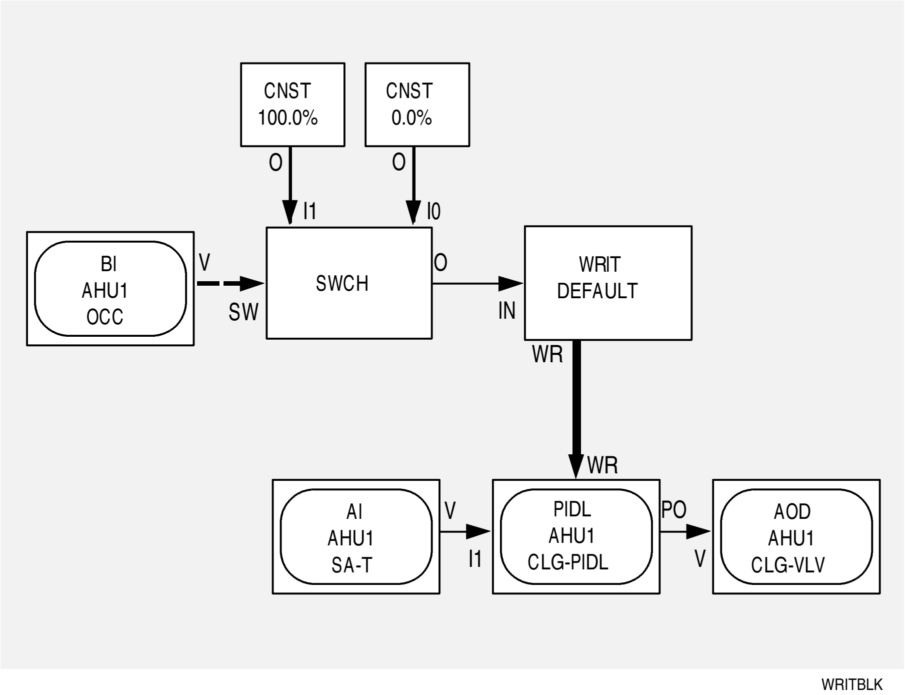


Figure 104: WRIT Block Example

XOR (Exclusive OR) Block

Category

Logic

Purpose

Performs a logical exclusive OR operation on two binary inputs.

Details

The following truth table describes the XOR block:

Input 1	Input 2	Output
True	True	False
True	False	True
False	True	True
False	False	False

Information Table

Field Name	Connect Name	Type	Default	Range	RC	T	I	O	LB
Block Name		STR	Blank	8 Char		X			
-	INPUT 1	BIN		0,1	Y		X		I1
-	INPUT 2	BIN		0,1	Y		X		I2
-	OUTPUT	BIN	0	0,1	Y			X	O
-	CONTROL	CTL			N		X		CF
-	CONTROL	CTL			N			X	CF

Reliability

The output of the XOR block is unreliable if either of its inputs is unreliable.

Example

In this example (Figure 105), an XOR block is used to verify a fan has actually started. When the fan (AHU1\FAN) is commanded to on, the process waits one minute. Then, it checks if the fan (AHU1\FAN-STAT) has indeed started. If the fan has not started, the PRNT block sends the following message to the printer:

```
AHU1\FAN DID NOT START
```

A control flow line is drawn from the WAIT block to the XOR block to make sure the wait is executed before the XOR block. The BI connection to the XOR block is exempted from triggers because only the BO object, not the BI object, should be allowed to trigger the process.

No period is defined for this process, since it only needs to run when the value attribute of the BO object changes.

Note: All operation blocks in this example have to be placed in a process. They include: PRNT, XOR, and WAIT.

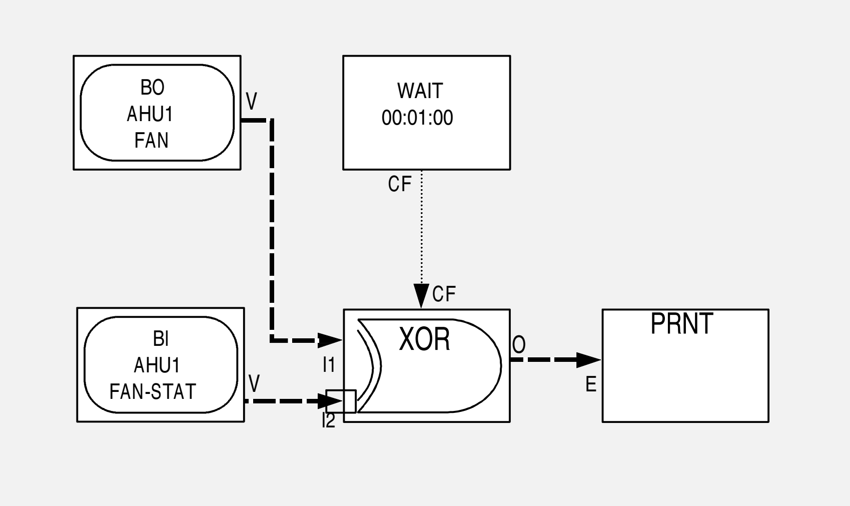


Figure 105: XOR Block Example

GPL Programmer's Manual

Template Field Descriptions

The following is a reference section that contains full descriptions of all attributes and parameters that appear in the GPL database templates. After the definition, the range or list of options for the attribute or parameter is given. (Refer to *Appendix F: Characters, Symbols, and Reserved Words* for a list of valid characters and symbols.) The descriptions are organized alphabetically.

Note: The definitions of the attributes and parameters for the object blocks are brief. For full details, refer to the particular object technical bulletin in the *Software Data Sheets* section of the *Metasys Network Technical Manual (FAN 636).*

Active Output Mode Flag

Specifies whether you want to enable the Active Output Mode Flag for this mode of the BSEQ block. The Active Output Mode Flag and the Active Input are placed in an OR function to determine the Active Output. See the following table.

Active Input	Active Output Mode Flag	Active Output
True	Yes	True
True	No	True
False	Yes	True
False	No	False

Choices: Y (Yes) or N (No).

Adjust Disabled

Specifies whether you want to disable the Priority 3 Adjust (or Set) commands. This prevents the associated input object's value from being overwritten by the Priority 3 commands. You can still issue the Override command if you need to change the object's value manually. Choices: Y (Yes=Adjust disabled) or N (No = Adjust enabled).

Advisory Type

Type of advisory that is assigned to the text that the ADV block will send to operator devices. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW_UP, and STATUS.

Alarm

The report type that an alarm COS will be reported as. Select NONE if you do not want a report generated when the object changes its status to Alarm. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW_UP, STATUS, and NONE.

Alarm #

Alarm Number. A user-defined reference number that identifies the particular text to be included with an alarm COS report. The text is displayed in the dialog box of a critical alarm report. Range: 0 to 255 (0 = No message).

Alarm Delay

Time period that the BI object may wait before reporting an alarm. The delay is useful in preventing nuisance alarms when the BI object is used as a feedback to a BO object. When Delay All Alarms is enabled, all COS transitions from the normal to the alarm state will also be delayed.
Range: 0 to 255 seconds (0 = no alarm delay).

Alarm if Set

Determines whether or not to activate a local annunciation relay when the binary alarm point is set into alarm.
Choices: Y (Yes) or N (No).

Analog Con Units

Analog Consumption Units. Units that help make values in the Demand Limiting feature more meaningful in that they describe the type of value that the ACM object will report (e.g., kWh). Range: one to six valid characters and symbols.

Analog Connections Attribute (A1 to A8)

Name of the analog attribute that will be an output connection for the REF block (e.g., AI_2). This name should match exactly with the corresponding attribute that was specified in the DDL file or at the Operator Workstation. Choices: all valid analog readable attributes.

Analog Connections Description (A1 to A8)

Description of the analog attribute (e.g., Dis_Temp). If the attribute is used in a software model or hardware object, we recommend that you match the attribute’s description in the REF block with its description in the model or object. In that way, you will be able to easily identify the attribute in GPL and at the Operator Workstation. Range: 1 to 8 valid characters and symbols.

Analog Inputs

Number of configured analog data inputs for the USER block. Range: 0 to 8.

Analog Outputs

Number of configured analog data outputs for the USER block. Range: 0 to 8.

Analog Type

The type of analog device that this object represents. Choices: 1K ohm, 100 ohm, VOLT/AMP, V/A LOW END REL.

Analog Units

Engineering units that describe the type of value of the object, such as DEG F (Degrees Fahrenheit) or DEG C (Degrees Celsius). These units are displayed on Change-of-State (COS) reports and summaries that report the condition of the object. Range: 1 to 6 valid characters and symbols.

Associated Input Attribute Name

Name of the attribute, such as VALUE, that the AD or BD is sampling.

Associated Input Object Name

Name of the system, such as AHU1, from which the AD (or BD) object is getting its attribute sample. Choices: all valid object names.

Associated Input System Name

Name of the object, such as OAT1 (for AD) or RETFAN (for BD), that the AD (or BD) object is sampling. Choices: all valid system names.

Attribute Name (Output 1-8 of PIDL Block)

Name of the object attribute to which that the output will map. If the destination object is an AOD block, select VALUE. If the destination object is another PIDL block, select one of the following: INP1VAL, INP2VAL, INP3VAL, INP4VAL, INP5VAL, INP6VAL, SETPOINT, OFFSET, HI_SAT_V, LO_SAT_V, AUX_IN, SEL_INP. If no output is to be mapped, leave this field blank.

Auto Dialout

Whether you want critical reports (CRITICAL 1-CRITICAL 4) to force a dialup to a remote Operator Workstation. Choices: Y (Yes) or N (No).

Auto Restore

Whether you want an object to revert to its last commanded condition when communication is restored (after a failure), or when the NCM is restarted. Choices: Y (Yes) or N (No).

Aux Delta P

Auxiliary Delta Pressure. Whether the C210A controller has configured the auxiliary delta pressure point called AUXP. Choices Y (Yes) or N (No).

Aux Humid

Auxiliary Humidity. Whether the C210A controller has configured the auxiliary relative humidity point called AUXR. Choices: Y (Yes) or N (No).

Aux Humid 1

Auxiliary Humidity 1. Whether the C260A controller has configured the auxiliary relative humidity point called AUXR1. Choices: Y (Yes) or N (No).

Aux Humid 2

Auxiliary Humidity 2. Whether the C260A controller has configured the auxiliary relative humidity point called AUXR2. Choices: Y (Yes) or N (No).

AUX Sensor

For 210A Block:

Auxiliary Sensor. Whether the C210A controller has configured the auxiliary input sensor point called AUXI. Choices: Y (Yes) or N (No).

For 260A Block:

Auxiliary Sensor. Whether the C260A controller has configured the auxiliary input sensor point called AUXI. Choices: Y (Yes) or N (No).

Aux Switch Ena

Auxiliary Switch Enable. Whether you want the PIDL object to use the auxiliary switch input instead of the PID algorithm. The auxiliary switch provides an input to the PID loop that bypasses the PID algorithm. If you want the auxiliary switch input to be passed to the auxiliary switch output, answer Y (Yes). If you want the result of the PID algorithm to be passed to the auxiliary switch output, answer N (No).

Aux Switch Input Object Name

Name of the AI object, if any, whose value is used for the Aux Switch Input value. The AI object and PIDL object must be on the same DCM. Choices: all valid object names.

Aux Switch Input Reference Select

Which value the PIDL object will use for the Auxiliary Switch input. If you want the PIDL to use the Auxiliary Switch input value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM. Choices: NORMAL or REFERENCE.

Aux Switch Input System Name

Name of the system that contains the AI object, if any, whose value is used for the Aux Switch Input value. Choices: all valid system names.

Aux Switch Input Value

Value that provides a means for some other algorithm or a default value to have control of the PIDL object output. Range: all real numbers.

Aux Temp

Auxiliary Temperature. Whether the C210A controller has configured the auxiliary temperature sensor point called AUXT. Choices: Y (Yes) or N (No).

Aux Temp 1

Auxiliary Temperature 1. Whether the C260A controller has configured the auxiliary temperature sensor point called AUXT. Choices: Y (Yes) or N (No).

Aux Temp 2

Auxiliary Temperature 2. Whether the C260A controller has configured the auxiliary temperature sensor point called AUXP. Choices: Y (Yes) or N (No).

Auxiliary Binary

For 210A Block:

Whether the C210A controller has configured the auxiliary latching binary input called AXBI. Choices: Y (Yes) or N (No).

For 260A Block:

Whether the C260A controller has configured the auxiliary latching binary input points called LTCH and HLTC. Choices: Y (Yes) or N (No).

Barometric Press.

Barometric Pressure. Value of barometric pressure for the Psychrometric Equations operation blocks (included are ENDP, ENRH, WBDP, and WBRH). Range: all real numbers.

BI Point Number

The binary alarm point address (terminal connection) at the Card Reader. Range: 1 to 8.

Binary Connections Attribute (B1 to B8)

Name of the binary attribute that will be an output connection for the REF block (e.g., BI_2). This name should match exactly with the corresponding attribute that was specified in the DDL file or at the Operator Workstation. Choices: all valid binary readable attributes.

Binary Connections Description (B1 to B8)

Description of the binary attribute (e.g., Fan_Stat). If the attribute is used in a software model or hardware object, we recommend that you match the attribute’s description in the REF block with its description in the model or object. In that way, you will be able to easily identify the attribute in GPL and at the Operator Workstation. Range: 1 to 8 valid characters and symbols.

Binary Inputs

Number of configured binary data inputs for the USER block. Range: 0 to 8.

Binary Outputs

Number of configured binary data outputs for the USER block. Range: 0 to 8.

Binary Type

The type of binary device that this object represents.
BI choices: BIN101, BIF101, BIS101, SST101, SST102;
BO choices: SST101, SST102.

Blnk at Turn OFF

Blink at Turn Off. Whether the group of lights that is represented by the LCG object should blink five minutes and two minutes before turning off. The purpose is to notify occupants that lights are soon to turn off. Choices: Y (Yes) or N (No).

Block Label

A user-specified unique character that is part of the REF block’s identification. It differentiates between multiple REF blocks that have the same system\object name.
Range: 1 valid character.

Block Name

Name that identifies the operation block. Range: 1 to 8 valid characters and symbols.

Block Name (Attribute Name)

Name of the attribute you wish to read from or write to. You must spell the attribute name correctly. Refer to *Appendix G: Attributes* for the correct spelling of all readable and writable attributes. Choices: valid attribute names.

Block Name (Variable Name)

Name assigned to the variable of the SVAR block.
Range: 1 to 8 valid characters and symbols.

Break Setpoint 1-8

Parameter of the BSEQ block that helps decide whether the block is to increment or decrement through the stages. If you select the Greater Than (>) operator, the Break Setpoints must be descending order (e.g., Break Setpoint 1=68.0, Break Setpoint 2=67.0). However, if you select the Less Than (<) operator, the Break Setpoints must be in ascending order (e.g., Break Setpoint 1=67.0, Break Setpoint 2=68.0). The Break Setpoint can be equal to the Make Setpoint.
Range: all real numbers.

Cleaning Crew Switches

Whether you want to enable the cleaning crew switch function for the lighting group. This function allows a switch to turn on a lighting group temporarily while the cleaning crew works in the area served by the group. Choices: Y (Yes) or N (No).

Comm Disabled

Communications Disabled. Whether you want to disable communications between the object and the associated controller (DCM or XM). When an object has its communication disabled, it cannot trigger control processes, send COS reports to operator devices, or accept commands (except for Enable). Choices: Y (Yes) or N (No).

Command

The name of the command that the CMD block will send to the object. This is a read-only field (cannot be modified). Choices: all possible commands shown on the connection menu.

Command (0)

The name of the command that the 2CMD block will send to the object when the Select Command input is False (0). This is a read-only field (cannot be modified). Choices: all possible commands shown on the connection menu.

Command (1)

The name of the command that the 2CMD block will send to the object when the Select Command input is True (1). This is a read-only field (cannot be modified). Choices: all possible commands shown on the connection menu.

Command Outputs

Number of configured command outputs for the USER block. Range: 0 to 8.

Constant 1 (C1)-Constant 4 (C4)

Constant values 1 to 4 that are used in the equation defined by the EQN block. Range: all real numbers.

Damper

Which damper configuration is defined at the C210A controller. Select a configuration according to the following table:

Points Defined	Select This Configuration
ACTC, DMPR	PRESSURE DEP (pressure dependent)
ACTC, DMPR, DFPR, MXDP, AXDP, DPSP, and PMAX	PRESSURE INP (pressure independent)
No Damper Points Defined	NO DAMPER

Date?

Whether you want the current date appended to the statement that the PRNT block sends to the printer. Choices: (Yes) or N (No).

Deadband

For DBCM Block:

A range for the DBCM block within which Input 1 can vary about Input 2 without initiating any change in the output (Figure 106). One-half of the deadband is on either side of Input 2. Range: all positive real numbers.

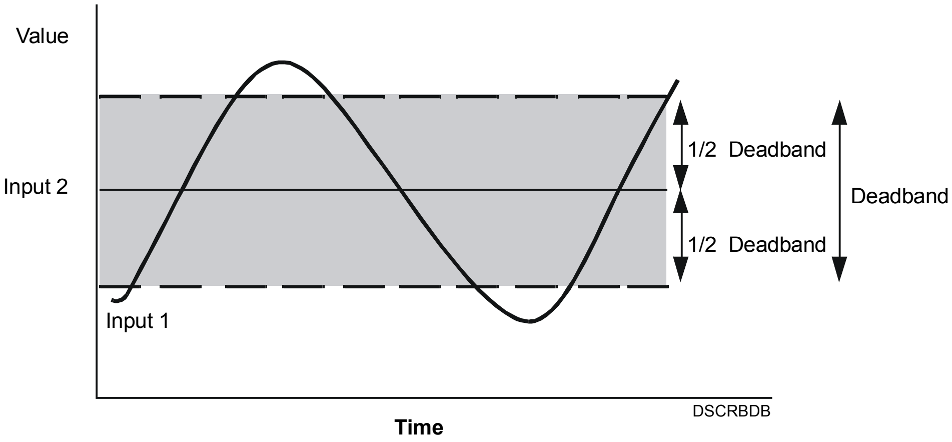


Figure 106: Describing the Deadband

For PIDL and PIR Blocks:

Specifies the range of error (setpoint minus feedback) that will cause no change in the block's output. If the error is within the deadband (1/2-Deadband ≤ Error ≤ 1/2-Deadband), the error is 0. If the error is outside the deadband, the error is the measured error minus one-half the deadband (positive error) or plus one-half the deadband (negative error).

Debounce Filter

The period of time in seconds or milliseconds that the software waits before reading the value of a fluttering contact.
Range for DCM: 1 to 255 seconds.
Range: 12 to 3060 msec (multiples of 12 only).

Decimal Position

Parameter that defines the number of digits that are displayed to the right of the decimal point. For example, a decimal position of "2" would display the number 72 as "72.00." This parameter applies to all numerical attributes of the object. Range: 0 to 3.

Default

Value that the UNRD block will pass to its output if its input is unreliable. Range: all real numbers (analog), 0 or 1 (binary), or 00:00:00 to 23:59:59 (time).

Delay All Alarms

Determines whether or not the alarm delay of the object is used only for feedback alarm delay or every time the object goes into the alarm state.

Delay Off

Timer used in the BSEQ block to aid in determining when a configured stage should run. When the Delay Off timer expires, the BSEQ checks to see if it needs to move to the next lowest stage. Range: 00:00:00 to 23:59:59.

Delay On

Timer used in the BSEQ block to aid in determining when a configured stage should run. When the Delay On timer expires, the BSEQ checks to see if it needs to move to the next highest stage. Range: 00:00:00 to 23:59:59.

Derivative Wgt

Derivative Weight. Value that controls the sensitivity of the PID algorithm to the derivative (the rate of change) of the feedback value. If you want to remove the derivative action from the PID control algorithm, enter 0.0. The larger the derivative weight, the larger is the effect of the derivative on the overall control. A value greater than 4.0 is not recommended. Range: all positive real numbers.

Description

A descriptor that further explains the purpose of the CNST or REF block. Range: up to 24 valid characters and symbols.

Differential

For AI Block:

An optional software setting that prevents excessive alarm and warning reports, resulting from fluctuations above and below the limits (Figure 107). When the current value of the AI crosses over a limit and a report is generated, the value must again cross over the limit and its differential before a new report is issued. Range: 0.000001 to 99999999 or blank (not defined).

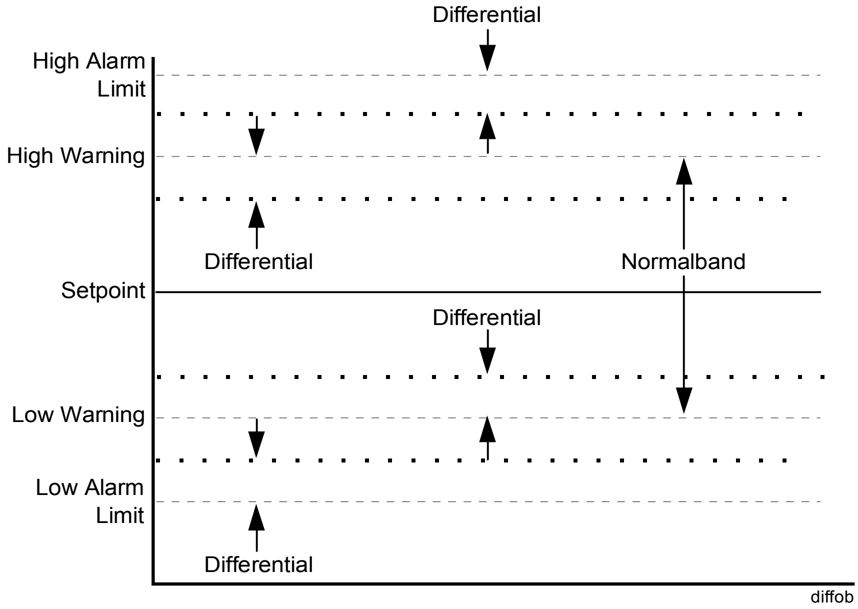


Figure 107: Differential for AI Object

For DFCM Block:

The range within which Input 1 can vary about Input 2 without initiating any change in the output. Range: all positive real numbers.

Disable #

Zone Disabled Number. A user-defined reference number that identifies the particular text to be included with a zone disabled COS report. The text is displayed in the dialog box of a critical zone disabled report. Range: 0 to 255 (0 = no message).

Display Units

Engineering units that describe the type of value of the CS object, such as DEG F (Degrees Fahrenheit). These units are displayed on COS reports and summaries that report the condition of the object. Range: 1 to 6 valid characters and symbols.

Display Value AI

Which analog input (AI-1 to AI-6) will be assigned to the Value attribute of the C210A object. The value of this input can then be viewed at the Operator Workstation and the Network Terminal. Choices: 1 to 6 (i.e., AI-1 to AI-6).

Edge Trigger

Whether you want to enable the Edge Trigger function for the 2CMD block. This function affects command execution. If you answer Y (Yes) to Edge Trigger, Command 1 is sent only when the Select Command input changes from False to True. Similarly, Command 0 is sent only when the Select Command input changes from True to False. If you answer N (No) to Edge Trigger, Command 1 is always sent when the Select Command input is True; and, Command 0 is always sent when the Select Command input is False.

Enable PT Hist.

Enable Point History. Whether you want the NCM to collect historical information on the object automatically. When enabled, this parameter causes the NCM to sample and store AI information, such as its current value and alarm status. This information can be viewed on an operator device or sent to a printer. Choices: Y (Yes) or N (No).

End Value

The value at which the RAMP block will stop ramping. If you want the block to ramp up, define an End Value that is greater than the Start Value. If you want the block to ramp down, define an End Value that is less than the Start Value.
Range: all real numbers.

Equation (Output)

Mathematical equation that the EQN block will evaluate.
You may use the following characters: I1-I4, C1-C4, +, -, *, ?, (,), SQR, ^, LOG, SIN, COS, TAN, ABS, MIN, MAX, AVG, and PI. Range: up to 50 characters.

Exempt All?

Whether you want to exempt all binary objects from triggering the process. Choices: (Yes) or N (No).

Expanded ID

An expanded version of the object name that more clearly identifies the object. For example, Outside Air Temp for OAT. Range: 1 to 24 valid characters and symbols.

Fail-Safe Stage

Stage that the BSEQ block will assume if the Binary Enable input or analog input is unreliable. Choices: CURRENT stage, or stage 0 to 8.

Fan

The fan configuration that is defined at the C210A controller. Choices: NONE (no fan defined), SERIES, or PARALLEL.

Fan On/Comp Off

Fan On/Compressor Off. The Fan On/Compressor Off operation that is configured at the C210A controller. Select an operation according to the following table:

Point Defined	Select This Operation
FNON	LOCAL CONTACT
FOCO	L2 COMMAND
FNON, FOCO	BOTH
No Fan On/Compressor Off Point Defined	NONE

Feedback (Closed for Start)

Which state a feedback object will be set to when a Start command is issued. Choices: Y (Yes; Start command will close feedback input) or N (No; Start command will open feedback input).

Feedback Object Name

Name of the object, if any, whose setpoint value or Normal State is mapped to the BO or AOS Feedback attribute. Choices: all valid object names.

Feedback System Name

Name of the system that contains the feedback object, if any. Choices: all valid system names.

Feedback Units

Engineering units associated with the feedback of a PID loop, such as DEG F (Degrees Fahrenheit). These units are displayed on COS reports and summaries that report the condition of the loop. Range: 1 to 6 valid characters and symbols.

File Name

Name assigned to the control strategy file. Range: one to eight valid characters and symbols.

Filter Tolerance

The amount that a monitored quantity must change before the FPU processes and reports the change as a change of status. For example, a filter tolerance of 1.0% for a humidity point means that a change of 1% is required for FPU processing. Also known as filter increment. Range: 0.2 to 100.0%.

Filter Weight

For ACM and AI Blocks:

An optional parameter that instructs the software to smooth out erratic input readings, such as spikes and analog noise. Its purpose is to reduce false alarms and unnecessary control process triggering. Range for ACM and AI: all real numbers or blank (not defined). Range for AD: 1.000001 to 99999999 or blank (undefined)

For FILT Block:

Value that specifies the amount of filtering that this block will apply to the input. The filtering action smoothes out changes in the input’s value. Range: 1.0 to 99999999.

For PIDL Block:

An optional parameter that filters out short duration variations in the output of the PID algorithm. To specify no filtering, enter 1.0. Note that a large filter weight can greatly distort the signal coming from the PID algorithm, to the extent that the controller is simply controlling the filter rather than the process. Range: 1.0 to 99999999.

Fire Zone #

Fire Zone Number. An identification number assigned to a particular zone in a fire controller. Range: 1 to 240.

Flow Coefficient

Optional value that is used in the square root calculation on a ranged AI value. It converts a flow sensor input into engineering units, such as CFM or GPM. If you specify a flow coefficient that is greater than 0.0, the square root function is performed on the filtered value and multiplied by the flow coefficient. However, if you specify a flow coefficient that is equal to 0.0, the square root function is not performed, and the filtered value is passed to the output. Range: 0.000001 to 99999999 or blank (not defined).

Graphic Symbol #

Graphic Symbol Number. A reference number that identifies the particular graphic symbol used to represent the object in Operator Workstation graphic summaries. Range: 0 to 32,767 (0 = No graphic).

Group Number

Number assigned that identifies the lighting group. Range: 1 to 32.

Hardware Object Name

Name of the hardware device that the object is mapped to. The object name must exist under the hardware system name defined. Choices: all valid hardware object names.

Hardware System Name

Name of the system that represents the control panel or NCM that is handling the object. Choices: all valid hardware system names.

Hardware Reference

See HW Reference.

Hardware Type

See HW Type.

Heat

Whether the C210A controller has configured the preheat mode. Preheat mode requires that these point names are defined: HTDB, OFRH, PRRH, and REHT. Choices:
NO HEAT (none of these names are defined) or HEAT.

Heavy Equip Dlay

Heavy Equipment Delay. Time period that equipment represented by the BO object will delay in starting.
Range: 0 to 255 seconds (0 = no delay).

High Alarm Limit

Optional parameter that determines when an alarm for an AI or AD object is generated. If the current value of the AI or AD object exceeds the High Alarm Limit, an alarm will be generated (Figure 108). Once the value decreases below the High Alarm Limit and Differential, the alarm clears.
Range: any positive real number or blank (not defined).

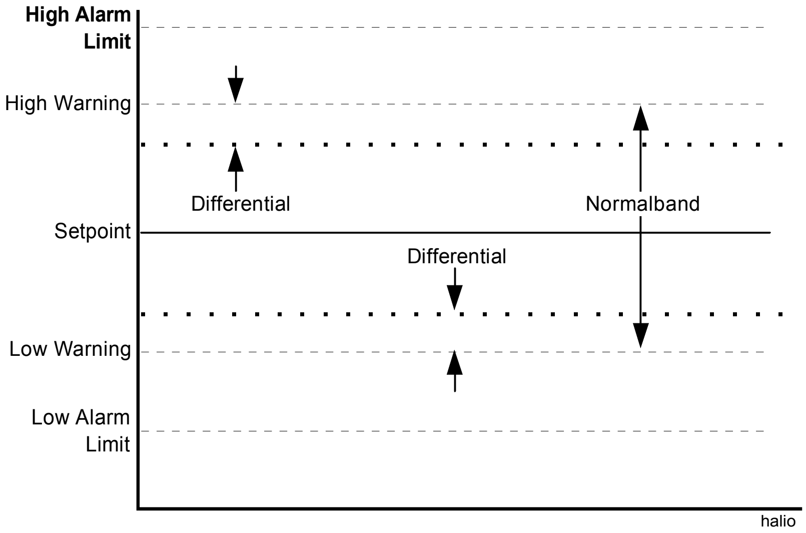


Figure 108: High Alarm Limit for AI Object

High Input

The highest value that the SPAN block can accept as an input. Enter a High Input that is greater than a Low Input.
Range: all real numbers.

High Output

The highest value that the PIR block can output. Enter a High Output that is greater than the Low Output.
Range: all real numbers.

High Saturation Object Name

Name of the AI object, if any, whose value is used for the High Saturation Limit. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

High Saturation Reference Select

Which value the PIDL object will use for the High Saturation Limit. If you want the PIDL to use the High Saturation Limit entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM.
Choices: NORMAL or REFERENCE.

High Saturation System Name

Name of the system that contains the AI object, if any, whose value is used for the High Saturation Limit. Choices: all valid AI system names.

High Saturation Value

The maximum allowed output of the PID algorithm.
Range: all positive real numbers.

HW Reference

The point type and number for the LCP, TC9100, DX9100, LON, DX91ECH, DC9100, DR9100, XT9100, or XTM the object is mapped to.

Choices per object:

Objects	Hardware Reference	
ACM	DC9100:	Total1-2
	DX9100, DX91ECH:	CNT1-8 PM1-12AC1-8 XT1-8CNT1-8
	XT9100, XTM:	CNT1-8
AI	DR9100:	AI1-4
	DC9100, LCP:	AI1-8, Total1-2[1] (for DO9100 only)
	DX9100:	AI1-8, XT1-8AI1-8, CNT1-8[1], PM1-12AC1-8[1], XT1-8CNT1-8[1]
	DX91ECH:	AI1-8, XT1-8AI1-8, CNT1-8[1], PM1-12AC1-8[1], XT1-8CNT1-8[1]
	TC9100:	AI1-4
	XT9100, XTM:	AI1-8, CNT1-8[1]
	LON[2]	Definition not available.

[1] Restrictions apply to all AI objects mapped to these hardware references. Please refer to AI documentation.

[2] GPL cannot show hardware references used by software objects on LONWORKS compatible devices.

Continued on next page . . .

Objects (Cont.)	Hardware Reference	
AOS	DC9100, LCP:	OUT1-8, ACO1-4
	DX9100:	OUT1-14, ACO1-8, XT1-8AO1-8
	DX91ECH:	OUT1-14, ACO1-8, XT1-8AO1-8
	XT9100, XTM:	AO1-8
	LON[2]:	Definition not available.
BI	DR9100:	WIN, OCC, AIRQ
	DC9100, LCP:	DI1-8, LCM1-4
	DX9100:	DI1-8, XT1-8DI1-8, LRS1-32
	DX91ECH:	DI1-8, XT1-8DI1-8, LRS1-64
	TC9100:	WIN, OCC, AIRQ, ALM, AFM
	XT9100, XTM:	1DI1-8, 2DI1-8
	LON[2]:	Definition not available.
BO	DR9100:	DO3-7, STUP, SOFF
	DC9100, LCP:	DO3-8, STUP, SOFF, DCO1-4
	DX9100:	DO3-8, STUP, SOFF, DCO1-32
	DX91ECH:	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	TC9100:	DO1-7, STUP, SOFF
	XT9100, XTM:	1DO1-8, 2DO1-8
	LON[2]:	Definition not available.
MSI	DR9100:	WIN, OCC, AIRQ
	DC9100, LCP:	DI1-8, LCM1-4
	DX9100:	DI1-8, XT1-8DI1-8, LRS1-32
	DX91ECH:	DI1-8, XT1-8DI1-8, LRS1-64
	TC9100:	WIN, OCC, AIRQ, ALM, AFM
	XT9100, XTM:	1DI1-8, 2DI1-8
	LON[2]:	Definition not available.
MSO	DR9100:	DO3-7, STUP, SOFF
	DC9100, LCP:	DO3-8, STUP, SOFF, DCO1-4
	DX9100:	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	DX91ECH:	DO3-8, STUP, SOFF, DCO1-32, XT1-8DO1-8
	TC9100:	DO1-7, STUP, SOFF
	XT9100, XTM:	1DO1-8, 2DO1-8
	LON[2]:	Definition not available.

[1] Restrictions apply to all AI objects mapped to these hardware references. Please refer to AI documentation.

[2] GPL can not show hardware references used by software objects on LONWORKS compatible devices.

Continued on next page . . .

Objects (Cont.)	Hardware Reference	
MSO Local Contact	DR9100:	WIN, OCC, AIRQ
	DC9100, LCP:	DI1-8, LCM1-4
	DX9100:	DI1-8, LRS1-32
	DX91ECH:	DI1-8, XT1-8DI1-8; LRS1-64
	TC9100:	WIN, OCC, AIRQ, ALM, AFM
	XT9100, XTM:	1DI1-8, 2DI1-8

HW Type

Hardware Type. The type of hardware that the sensor or device is connected to for this object. (The abbreviation 2X means Multistate XMs.)

Choices per object:

Objects	Hardware Type
ACM	XBN, XRM/XRL/2X, AHU, VAV, UNT, OTHER*, DC9100, DX9100, XT9100, XTM, DX91ECH, FPU, DSC8500
AI	DCM, DCM140, FPU, DSC8500, AHU, VAV, VMA, UNT, OTHER[2], DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
AOD	DCM, DCM140
AOS	DCM, DCM140, FPU, DSC8500, AHU, VAV, VMA, UNT, OTHER[2], DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, LON[1]
BI	DCM, DCM140, FPU, DSC8500, XBN, XRM/XRL/2X, AHU, VAV, VMA, UNT, OTHER[2], D600, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
BO	DCM, DCM140, FPU, DSC8500, XRM/XRL/2X, AHU, VAV, VMA, UNT, OTHER[2], DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
MSI	DCM, DCM140, XBN, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
MSO	DCM, DCM140, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100, LON[1]
MSO Local Contact	DCM, DCM140, XBN, XRM/XRL/2X, DR9100, DC9100, LCP, DX9100, XT9100, XTM, DX91ECH, TC9100

[1] GPL cannot show hardware references used by software objects on LONWORKS compatible devices.

[2] OTHER must be the N2OPEN hardware type as used in DDL.

Hysteresis Comp. Hysteresis Compensation. Value that compensates for the error that occurs when the direction of a device is driven open or closed by an actuator. The value is either added to or subtracted from the output of the PID algorithm. Range: 0.0 to 100.0%.

Identification Object Name Name assigned to identify the point object, such as AIR_FLOW (Air Flow switch). The AI object and PIDL object must be on the same DCM. Choices: all valid object names.

Identification System Name Name assigned to identify a system in the network, such as AHU1. Choices: all valid system names.

Initial Value The value of the object when it is first defined. Range for AD and AOD: all real numbers. Range for BD and BO: STATE 0 or STATE 1.

Input A constant value for the input of the WRIT block. The block uses this value if you do not make an external input connection. Ranges: all real numbers (analog), 0 or 1 (binary), or 00:00:00 to 23:59:59 (time).

Input 0 A constant value for Input 0 of the SWCH block. The block uses this value for Input 0 if you do not make an external connection for Input 0. Ranges: all real numbers (analog), 0 or 1 (binary), or 00:00:00 to 23:59:59 (time).

Input 1 A constant value for Input 1 of the operation block. The block uses this value for Input 1 if you do not make an external connection for Input 1. Ranges: all real numbers (analog), 0 or 1 (binary), or 00:00:00 to 23:59:59 (time).

Input 1 (I1)-4 (I4) Constant values for Input 1 through Input 4 of the EQN block. These values are placed into the I1 to I4 terms as defined by the equation. Range: all real numbers.

Input 1 (I1)-8 (I8) Constant values for Input 1 through Input 8 of the MSEL block. The block uses these values if you do not make external connections for Input 1-8. Range: all real numbers.

Input 1-6 Object Name

Name of the AI object, if any, whose value is used for the Input *n* value. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

Input 1-6 Reference Select

Which value the PIDL object will use for Input *n*. If you want the PIDL to use the Input *n* value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM. Choices: NORMAL or REFERENCE

Input 1-6 System Name

Name of the system that contains the AI object, if any, whose value is used for the Input n value. Choices: all valid system names.

Inputs 1-6 Value

Specifies the inputs that are used in the PID algorithm. Range: all real numbers.

Input 2

A constant value for Input 2 of the operation block. The block uses this value for Input 2 if you do not make an external connection for Input 2. Ranges: all real numbers (analog),
0 or 1 (binary), or 00:00:00 to 23:59:59 (time).

Input Function

Determines the feedback value for the PID algorithm. If you select SUM (summation), the feedback value is the summation of the PID loop’s six inputs multiplied by the six scalars.

If you select MIN, the feedback value is the lowest product of the PID loop’s six inputs multiplied by the six scalars. If you select MAX, the feedback value is the highest product of the PID loop’s six inputs multiplied by the six scalars.

The choices are: SUM, MIN, and MAX.

Input Type

Type of input defined for the BI object. Choice: 2-STATE.

I/O Connections

Whether the CONN block must be connected to another block. Choices: Optional and Required.

Integral Time

Value in seconds that controls the sensitivity of the PID control algorithm to the integral of the error. If you want to remove the integral action from the PID control algorithm, enter 0.0. Range: 0.000000 to 99999999 seconds.

Interlock Statement

Control-by-event statement that tells the fire system to take a specific action when a certain point reports an alarm. Only use characters that are valid for interlock statements. [Refer to the *IFC-1010/2020 Technical Manual (FAN 448).*]
Range: up to 70 characters.

Latching Point

Whether you want the BD object to remain in alarm when it changes to an Alarm state. Choices: Y (Yes) or N (No).

LED ON when CLO

LED On when Closed. Whether the LED is On when the relay of the BI object is closed. This value defines the normal state of the LED associated with the normal condition state. Choices: Y (Yes; LED is On when Closed) or N (No; LED is Off when Closed).

Linear. Parm. 1

The first linearization parameter used in the software calculation that ranges the analog-to-digital counts. The value is defined and depends on the STD Range Type selected in the field above. However, you may also define your own value for this parameter by entering “0” in the STD Range Type field. Range: 1.0000 to 999999.

Linear. Parm. 2

The second linearization parameter used in the software calculation that ranges the analog-to-digital counts. The value is defined and depends on the STD Range Type selected in the field above. However, you may also define your own value for this parameter by entering “0” in the STD Range Type field. Range: 1.0000 to 999999.

Linear Parm. 3

The third linearization parameter used in the software calculation that ranges the analog-to-digital counts. The value is defined and depends on the STD Range Type selected in the field above. However, you may also define your own value for this parameter by entering “0” in the STD Range Type field. Range: 1.0000 to 999999.

Linear. Parm. 4 The fourth linearization parameter used in the software calculation that ranges the analog-to-digital counts. The value is defined and depends on the STD Range Type selected in the field above. However, you may also define your own value for this parameter by entering “0” in the STD Range Type field. Range: 1.0000 to 999999.

Local Control Whether the controller or N2 device can assume control of its points from the NCM. Choices: Y (Yes; controller or N2 device is controlling its points) or N (No; NCM is controlling points).

Note: Not used for LONWORKS devices.

Logical Pnt Nmbr Logical Point Number. A unique number associated with the logical point type that identifies the DSC8500 point. Range: 1 to 255.

Logical Pnt Type Logical Point Type. An abbreviation of the type of DSC8500 point that, with the logical point number, identifies the point. Choices are:

Object	Choices
ACM	TOT
AI	ASP, ADP, LTD, FUL, RAT, TOT, INC
AOS	ASP, ADP, INC
BI	BSP, BDP, CON, MAN
BO	BSP, BDP, MOM, MAN, BOF

Low Alarm Limit Optional parameter that determines when an alarm for an AI object is generated. If the current value of the AI object is lower than the Low Alarm Limit, an alarm will be generated (Figure 109). Once the value increases above the Low Alarm Limit and the Differential, the alarm clears. Range: any positive real number or blank (not defined).

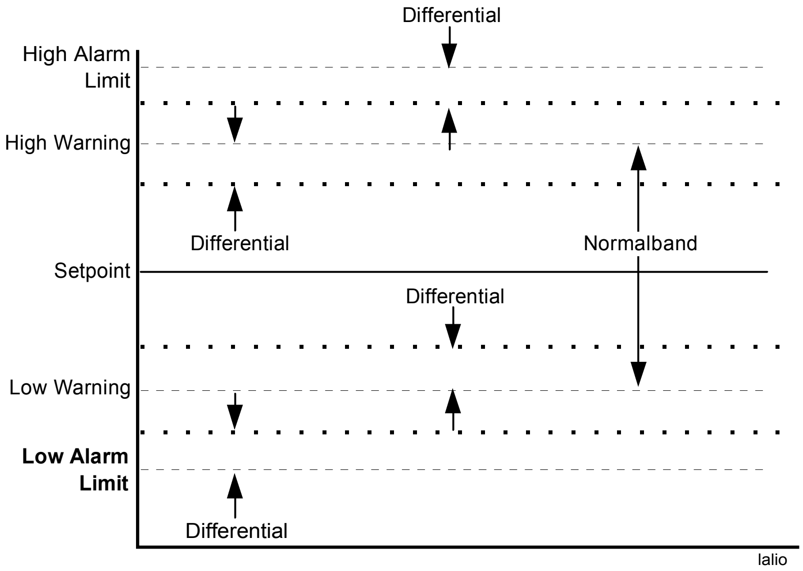


Figure 109: Low Alarm Limit for AI Object

Low Input

The lowest value that the SPAN block can accept as an input. Enter a Low Input that is less than a High Input.
Range: all real numbers.

Low Output

The lowest value that the PIR block can output. Low Output must be less than the High Output. Range: all real numbers.

Low Saturation Object Name

Name of the AI object, if any, whose value is used for the Low Saturation Limit. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

Low Saturation Reference Select

Which value the PIDL object will use for the Low Saturation input. If you want the PIDL to use the Low Saturation value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM.
Choices: NORMAL or REFERENCE.

Low Saturation System Name

Name of the system that contains the AI object, if any, whose value is used for the Low Saturation Limit. Choices: all valid system names.

Low Saturation Value

The minimum allowed output of the PID algorithm.
Range: all positive real numbers.

Make Setpoint 1-8

Parameter of the BSEQ block that helps decide whether the block is to increment or decrement through the stages. If you select the Greater Than (>) operator, the Make Setpoints must be in ascending order (e.g., Make Setpoint 1=68.0, Make Setpoint 2=69.0). If you select the Less Than (<) operator, the Make Setpoints must be in descending order (e.g., Make Setpoint 1=67.0, Make Setpoint 2=65.0).
The Make Setpoint can be equal to the Break Setpoint.
Range: all real numbers.

Max Starts/Hour

Maximum Starts per Hour. The maximum number of State 1 commands that the BO object can issue in any hour.
Range: 1 to 255.

Min OFF Time

Minimum Off Time. The minimum time in seconds that the load controlled by the BO object must stay off, after it is turned off. The period protects the load from being damaging by short cycle commands. Range: 0 to 255 seconds (0 = no minimum Off time).

Min ON Time

Minimum On Time. The minimum time in seconds that the load controlled by the BO object must stay on, after it is turned on. The period protects the load from being damaging by short cycle commands. Range: 0 to 255 seconds (0 = no minimum On time).

NC Name

The name of the Network Control Module on which the AD or BD object is defined. This is a read-only field, which means that GPL enters it for you.

Normal

The report type that a normal COS will be reported as. Select NONE if you do not want a report generated when the object changes to Normal. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW-UP, STATUS, and NONE.

Normal #

Normal Number. A user-defined reference number that identifies the particular text to be included with a normal COS report. The text is displayed in the dialog box of a return-to-normal report. Range: 0 to 255 (0 = no message).

Normalband

Optional parameter that is the width of the normal operating range for the object's current value. Normalband is centered on the setpoint (Figure 110), and is used to compute the high and low warning limits. Range: 0.000001 to 99999999 or blank (undefined).

Note: Either both the Normalband and Setpoint must be defined, or both must be undefined (blank).

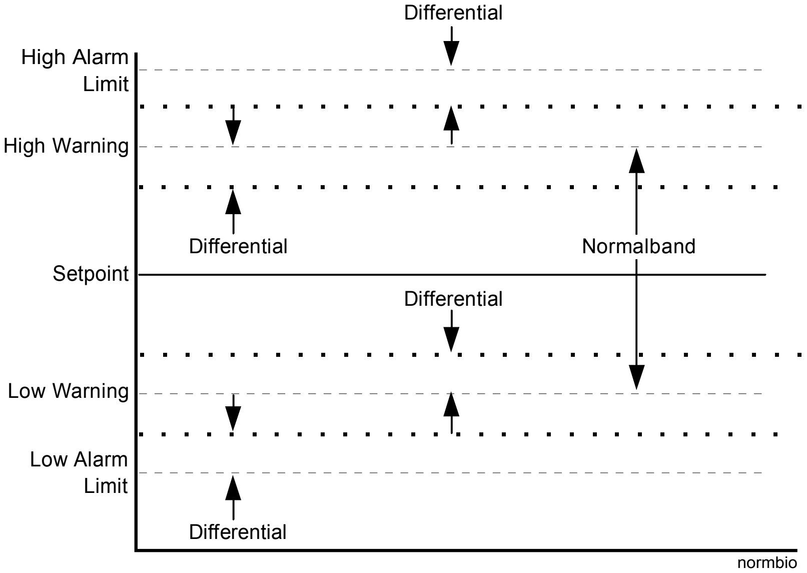


Figure 110: Normalband for AI Object

Normal State

A user-defined normal contact value for the BD or BI object, such as On or Off. If you do not want to define a normal state for the object, select NONE, which means it will never go in alarm. Choices: STATE 0, STATE 1, or NONE (no state defined).

Num. of Outputs

Number of Outputs. The number of configured outputs for the BSEQ block. Range: 1 to 8 outputs.

Num. of Stages

Number of Stages. The number of configured stages for the BSEQ block. Range: 1 to 8 stages.

Number of Inputs

The number of configured inputs for the operation block. This figure determines how many inputs can be connected to the block. Choices: 2, 3, or 4.

Number of Modes

The number of configured modes for the MSEL block. Range: 2 to 8 modes.

Object Name (Printer Name)

Name assigned to the printer to which the PRNT block will send the message. Choices: all valid printer object names.

Occupied/Unocc

For 210A Block:

Occupied/Unoccupied. Which occupied/unoccupied mode is configured at the C210A controller. Select a mode according to the following table:

Points Defined	Select This Mode
SUSB, HWUO	LOCAL CONTACT
SUSB, UNOC	L2 COMMAND
SUSB, HWUO, UNOC	BOTH
No Occupied/Unoccupied Points Defined	NONE

For 260A Block:

Occupied/Unoccupied. Which occupied/unoccupied mode is configured at the C260A controller. Select a mode according to the following table:

Points Defined	Select This Mode
STUP, STBK, HWUO	LOCAL CONTACT
STUP, STBK, UNOC	L2 COMMAND
STUP, STBK, HWUO, UNOC	BOTH
No Occupied/Unoccupied Points Defined	NONE

Off Switch Inp #

Off Switch Input Number. The number of the hardware input that is used to turn the group Off. This parameter applies only to a double momentary switch, which requires separate inputs for its On and Off pushbuttons. Range: 0 to 32 (0 = no Off input).

Offset

Initial value used by the PIR block the first time it executes error-free. Range: all real numbers.

Offset Object Name

Name of the AI object, if any, whose value is used for the PIDL object Offset Value. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

Offset Reference Select

Which value the PIDL object will use for the Offset input. If you want the PIDL to use the Offset value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM. Choices: NORMAL or REFERENCE.

Offset System Name

Name of the system that contains the AI object, if any, whose value is used for the PIDL object Offset Value. Choices: all valid AI system names.

Offset Value

A value that is added to the results of the PID algorithm. If all PIDL input scalars are 0.0, then Offset is the output of the PID algorithm. Range: all real numbers.

Operating Instr #

Operating Instruction Number. A reference number that identifies the particular text provided when Help on an object is requested at the Operator Workstation. Range: 0 to 32,767 (0 = No instructions).

Operation

The type of configured relational operator for the operation block. Choices: <, >, -, -, =, and <> (depends on block).

Output 1-8 Object Name

Name of the AOD or PIDL object, if any, that contains the attribute that will take on the value of the PIDL object's Current Output attribute. The object you enter must exist under the output system name defined. The AOD or PIDL object and the other PIDL object must be on the same DCM. Choices: all valid AOD or PIDL object names.

Output 1-8 System Name

Name of the system containing the attribute that will take on the value of the PIDL object's Current Output attribute. Choices: all valid AOD or PIDL system names.

Output Range 1

The lowest or highest value that the SPAN block can output. For a direct acting SPAN block, make Output Range 1 less than Output Range 2. For a reverse acting SPAN block, make Output Range 1 greater than Output Range 2.
Range: all real numbers.

Output Range 2

The lowest or highest value that the SPAN block can output. For a direct acting SPAN block, make Output Range 2 greater than Output Range 1. For a reverse acting SPAN block, make Output Range 2 less than Output Range 1. Range: all real numbers.

Output Relay (Closed for Start)

Which state the Output Relay will be set to when the Start command is issued. Choices: Y (Yes; Start command will close relay) or N (No; Start command will open relay).

Override

The report type that an override COS will be reported as. If you do not want a report generated when the object changes its status to Override, select NONE. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW_UP, STATUS, and NONE.

Ovrd Def Delay

Override Default Delay. The delay time in hours that is used for a Timed On command to a lighting group that has no assigned time value. When the timer expires, the lighting group will be commanded Off. Range: 0 to 99.9 hours
(0 = no delay).

Panel #

Panel Number. The addressing number assigned to a particular annunciator panel. Range: 1 to 32.

Period

How often a process will execute. Range: 00:00:00 to 23:59:59.

PID Loop Number

Integer value between 1 and 20 that identifies the PID Loop. The DCM can have up to 16 separate PID Loops. The DCM140 can have up to 20 separate PID Loops. The selected PID Loop must not currently have another PIDL object assigned to it. Range: 1 to 20.

Point #

Point Number. The addressing number assigned to a particular annunciator point. Range: 1 to 64.

Point Address

The hardware reference number assigned to the particular ASC point as listed in the symbol table. Ranges:

Object	Controller	Ranges
ACM	AHU	7 or 8
	VAV, UNT	4
	OTHER	1 to 256
AI	AHU	1 to 8
	VAV, UNT	1 to 6
	OTHER	1 to 256
	VMA	1 to 5
AOS	AHU and AO type	1 to 8
	AHU and ADF, ADI type	129 to 256
	VAV, UNT and AO type	1 to 8
	OTHER	1 to 256
	VMA and AO type	1 to 2
	VMA, VAV, UNT and ADF, ADI type	1 to 256
BI	AHU	1 to 8
	VAV, UNT	1 to 5
	OTHER	1 to 256
	VMA and BI type	1 to 3
BO	AHU and BO type	1 to 10
	VAV, UNT, and BO type	1 to 8
	AHU and BD type	193 to 256
	VAV, UNT, and BD type	225 to 256
	OTHER	1 to 256
	VMA and BO type	1 to 5
	VMA and BD type	65 to 256

Point Type

The type of input required or output provided by the associated field device. Choices:

Object	Hardware Device	Tab Choices
ACM	XMs	SINGLE or FORM C
	AHU, VAV, UNT, OTHER	BI only
AI	DCM140	AI, MAI
	AHU, VAV, VMA, UNT, OTHER	AI only
AOD	DCM	PROP or INCR
	DCM140	PROP, INCR, MAO
AOS	DCM	PROP or INCR
	DCM140	PROP, INCR, MAO
	AHU, VAV, VMA, UNT, OTHER	AO, ADF, ADI
BI	DCM, DCM140	BI or MBI
	XMs	SINGLE or FORM C
	AHU, VAV, VMA, UNT, OTHER	BI only
BO	DCM, DCM140	MAINTAINED or LATCHED
	AHU, VAV, VMA, UNT, OTHER	BO, BD

Priority

For Commands:

The relative priority level that is assigned to a command. The priority dictates whether the object will accept the command. For example, if the object's current command is at priority 4, a command at priority 3 or higher is required to release the priority 4 command. Range for Start/Stop commands: 2 to 7 (2=highest). Range for all other commands: 2 or 3 (2=highest).

For Processes:

The relative priority level that is assigned to a process. The priority dictates when a process will run in relation to other processes that are in the queue. The lower the priority number, the higher the priority. Range: 1 to 4 (1=highest).

Process Obj Name Process Object Name. Object name assigned that identifies the process. The name appears inside the process compound block. For a RESTART process, the process object name is fixed at RESTART. Range: all valid process object names.

Process Sys Name Process System Name. System name assigned that identifies the process. The name appears inside the process compound block. Choices: all valid system names.

Proportional Bnd Proportional Band. The change in the error that is required to cause a change of 100 in the PID algorithm’s output value. The error is defined as the difference between the feedback value and the setpoint value. For direct acting control, enter a negative value. For reverse acting control, enter a positive value. Range: all real numbers.

Prop Band Proportional Band. The change in the error that is required to cause the output to change from its low limit to its high limit, or vice versa. The error is defined as the difference between the feedback value and the setpoint value. For direct acting control, enter a negative value. For reverse acting control, enter a positive value. Range: all real numbers.

Protected? Whether you want to protect the contents of a compound from viewing. Choices: Y (Yes) or N (No).

PT Enabled Point Enabled. Determines whether the binary alarm point (associated with the BI) is enabled or disabled at the D600 controller. Choices: Y (Yes) or N (No).

Pulse Constant

A constant for converting a single ACM pulse into the quantity of energy or material that the pulse represents. For example, 0.034 kWh per pulse. If 0 is defined, no pulse constant is used in the calculation. Range: all real numbers or blank (not defined).

Pulse Duration

The length of time, in milliseconds, that the output to a latching field device will be pulsed for State 0 and State 1 commands. The range of values depends on whether the device is from a DCM or XRM/XRL. Range for DCM: 20 to 5100 msec (in multiplies of 20 msec only). Range for XRM/XRL: 12 to 3060 msec (in multiplies of 12 msec only).

Quiet if Reset

Determines whether or not to automatically silence the local annunciation relay when a binary alarm point is reset to its normally closed position. Choices: Y (Yes) or N (No).

Rate Constant

The time unit in the engineering units for the ACM object. Choices: HOUR (hours), MIN (minutes), and SEC (seconds).

Reader Number

The address the D600 controller uses to poll the card reader and its associated binary alarm points. Range: 1 to 16.

Ref. Object Type

Reference Object Type. The type of object to which the REF block is associated. Choices: CS, C260X, C500X, DSC-1000, XM (XBN, XRM, XRL, 2X), DCM, DCM140, LCD, AHU, VAV, UNT, LCP, DR9100, DC9100, DX9100, XT9100, FIRE, FPU, DSC8500, D600, READER, MIG, PHX, DX91ECH, TC9100, MC, NDM, XTM, VND, LON, OTHER.

Relay Outputs

The assignments for the relay outputs that the LCG object will control. Up to 40 relays can be controlled by a single LCG object. Choices for each output: Y (Assign relay/No blink), N (No assignment), or B (Assign relay/Yes blink).

Resolution

Time units used when converting an integer to a time value or a time value to integer in RTOT and TTOR blocks. Choices: SECONDS, MINUTES.

Sample Period

The time in seconds between consecutive calculations of the PID control algorithm. Range: 1 to 32,767 seconds.

Saturation Size

The pulse duration in seconds used to resynchronize the hardware point to 0% to 100%. The parameter applies only to an AOD or AOS object that is configured as an INCR point type. If you want to disable the saturation size function, enter 0. Range: 0 to 255 seconds.

Save PT History

Save Point History. If you answer Y (Yes), historical information for the object will be automatically sent from the NCM to an archive file on an Operator Workstation. If you select N (No), the information is only buffered at the NCM, and will be overwritten with new data when the file is full.

Scalar

Value that is multiplied by Input 1-6 to obtain the actual input to the PID loop. If you want to disable the input to the PID loop, enter 0.0. If you enter 0.0 for all scalar values, the PID algorithm is not invoked and the value of the PID loop Offset Port is used as the output of the PID calculation.
Range: all real numbers.

Select Unrl Dflt

Select Unreliable Default. If you answer Y (Yes), the Reliability Switch uses the Unreliable Default Response attribute as an output when the Reliable flag is equal to No. If you answer N (No), the last reliable value will be sent to the output.

Selector Input Object Name

Name of the AI object, if any, whose value is used for the Selector Input value. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

Selector Input Reference Select

Which value the PIDL object will use for the Selector input. If you want the PIDL to use the Selector input value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM. Choices: NORMAL or REFERENCE.

Selector Input System Name

Name of the system that contains the AI object, if any, whose value is used for the Selector Input value. Choices: all valid AI system names.

Selector Input Value

A minimum or a maximum limit that can be substituted for the filtered PID algorithm output value. Range: all real numbers.

Selector Type = HI

Whether you want the High/Low Signal Selector to choose the higher or lower input value and pass its value to the output of the PID loop. If you want the selector to choose the higher input value, answer Y (Yes). If you want the selector to choose the lower input value, answer N (No).

Setpoint

For ACM, AD, AI, and PIR Blocks:

The center point (axis) of the normal operating range for the object's current value. Setpoint is also used by the software to compute high and low limits. You must define a setpoint for a PIR block. Range: all real numbers or blank (not defined).

Note: Either both the Normalband and Setpoint must be defined, or both must be undefined (blank).

For 210A and 260A Blocks:

Which setpoint is configured at the C210 or C260A controller. If the point name ZNSP is defined, select LOCAL. If the point name RZSP is defined, select REMOTE.

Setpoint Object Name

Name of the AI object, if any, whose value is used for the PIDL object's Setpoint value. The AI object and PIDL object must be on the same DCM. Choices: all valid AI object names.

Setpoint Reference Select

Which value the PIDL object will use for the Setpoint input. If you want the PIDL to use the Setpoint value entered in the template, select NORMAL. If you want the PIDL to obtain the value from the object entered in the template, select REFERENCE. The referenced AI object and the PIDL object must be on the same DCM.

Setpoint System Name

Name of the system that contains the AI object, if any, whose value is used for the PIDL object's Setpoint value.
Choices: all valid AI system names.

Setpoint Value

The desired value of the control loop feedback signal. The PID algorithm will work to cause the Feedback value to be equal to the Setpoint value. Range: all real numbers.

Setpt for NT SP

Setpoint for Network Terminal Setpoint. Which setpoint will be the default setpoint of the 210A object that the Network Terminal can modify and time program. The NT can modify and time program only one of the 12 setpoints.
Choices: 1 to 12 (SP-1 to SP-12).

Shutdown

For 210A Block:

Which shutdown mode is configured at the C210A controller. Select a mode according to the following table:

Points Defined	Select This Mode
SDBO	L2 COMMAND OPEN
HWSD	LOCAL
HWSD, SDBC	L2 COMMAND CLOSED and LOCAL
No Shutdown Points Defined	NONE

For 260A Block:

Which shutdown mode is configured at the C260A controller. If the point name SDWN is defined, select L2 COMMAND. If no shutdown points are defined, select NONE.

Slot Number

Number that represents the Function Module (FM) slot number or input address. For the DCM, this number represents the FM slot where the field device is connected. For the XRLs/XRMs, this number usually means the input address (terminal block location) where the field device is connected. (The abbreviation 2X means Multistate XMs, available only in the European market.) If an IUN is used, the input slot number represents the FM slot number. The selected slot on the given hardware object must not be currently assigned to any other object. Ranges per hardware type are:

Hardware Type	Ranges
DCM	1 to 10
XBN	1 to 32
XRL/XRM/2X	1 to 8
FPU	1 to 16

Span High Input

Optional parameter that sets the upper bound for an analog value. If this parameter is defined, the analog value will not report an input higher than the Span High Input. This allows you to provide restrictions on the maximum displayed value of the analog input. It also allows you to apply an offset and/or gain adjustment to the input value. For example, a humidity sensor might operate in a range of 30-80%. A Span High Input of 80% would mean no value higher than 80% would be input. Range: all real numbers or blank (not defined).

Note: Span High Input must be greater than the Span Low Input. Also, if you define this parameter, you must also define the other three span parameters.

Span High Output

Optional parameter that sets the upper bound for an analog value. If this parameter is defined, the analog value will not report an output higher than the Span High Output. This allows you to provide restrictions on the maximum displayed value of the analog output. It also allows you to apply an offset and/or gain adjustment to the output value.
For example, a humidity sensor might operate in a range of 30-80%. A Span High Output of 80% would mean no value higher than 80% would be output. Range: all real numbers or blank (not defined).

Note: Span High Output must be greater than the Span Low Output. Also, if you define this parameter, you must also define the other three span parameters.

Span Low Input

Optional parameter that sets the lower bound for an analog value. If this parameter is defined, the analog value will not report an input lower than the Span Low Input. This allows you to provide restrictions on the minimum displayed value of the analog input. It also allows you to apply an offset and/or gain adjustment to the input value. For example, a humidity sensor might operate in a range of 30-80%. A Span Low Input of 30% would mean no value lower than 30% would be input. Range: all real numbers or blank (not defined).

Note: Span Low Input must be less than the Span High Input. Also, if you define this parameter, you must also define the other three span parameters.

Span Low Output

Optional parameter that sets the lower bound for an analog value. If this parameter is defined, the analog value will not report an output lower than the Span Low Output. This allows you to provide restrictions on the minimum displayed value of the analog output. It also allows you to apply an offset and/or gain adjustment to the output value. For example, a humidity sensor might operate in a range of 30-80%. A Span Low Output of 30% would mean no value lower than 30% would be output. Range: all real numbers or blank (not defined).

Note: Span Low Output must be less than the Span High Output. Also, if you define this parameter, you must also define the other three span parameters.

Stage 1-8/Delay Off

The Delay Off timers for Stages 1 to 8. For details, see *Delay Off*. Range: 00:00:00 to 23:59:59.

Stage 1-8/Delay On

The Delay On timers for Stages 1 to 8. For details, see *Delay On*. Range: 00:00:00 to 23:59:59.

Stage 1-8/ Outputs 1-8

The binary values of Outputs 1 to 8 for Stages 1 to 8. Choices: 0 or 1.

Start Value

The value at which the RAMP block will start ramping. If you want the block to ramp up, define a Start Value that is greater than the End Value. If you the block to ramp down, define a Start Value that is less than the End Value.
Range: all real numbers.

State 0 (Units)

Characters that represent the open (Off) contact position of a binary object. Range: up to six valid characters and symbols.

State 1 (Units)

Characters that represent the closed (On) contact position of a binary object. Range: up to six valid characters and symbols.

STD Range Type

Standard Range Type. Defines the type of field device that you are using, such as a 0-10 VDC transmitter. It is a number that represents a standard sensor device. Entering a number changes the linearization parameters to the predefined values. You may also enter customized linearization values by entering a “0” in the field. Ranges per hardware type are:

Hardware Type	Ranges
DCM	0 to 25, 34 to 111 (0=customized values)
FPU	0, 26 to 33 (0=customized values)

Step Ratio

Conversion factor used for incremental type points. The factor converts a commanded value change in percent to a pulse duration in seconds. The parameter applies only to an AOD or AOS object that is configured as an INCR point type.
Range: all real numbers.

Step Size

Value that the output of the RAMP block will change for each step up or step down. Range: all positive real numbers.

Subslot Number

Number that describes which input of the Function Module is represented by the object. Choices: 1 or 2.

Suppress TZ

Alarm Suppression Time Zone number. The Alarm Suppression Time Zone suppresses COS reporting for the binary alarm point during a specified time period.
Range: 0 to 8.

Switch Input #

The number of the hardware input that will be controlled by the LCG object. Range: 0 to 32 (0 = no switch input assignment).

Switch Type

The type of switch that characterizes the hardware input that will be controlled by the LCG object. Choices: MAINTAINED, MOMENTARY, DBL (double) MOMENTARY.

Text

The message that will be printed via the ADV or PRNT block. Range: up to 50 valid characters and symbols.

Time

For CNST Block:

A constant value for the time input of the CNST block. The block uses this value for its input if you do not make an external input connection. Range: 00:00:00 to 23:59:59.

For DLAY, PULS, and WAIT Blocks:

The value of the period timer that manipulates the value of the output. Range: 00:00:01 to 23:59:59.

Time?

Whether you want the PRNT block to append the current time of day to the message it will print. Choices: Y (Yes) or N (No).

Time Connections Attribute (T1 to T8)

Name of the time attribute that will be an output connection for the REF block (e.g., AD_2). This name should match exactly with the corresponding attribute that was specified in the DDL file or at the Operator Workstation. Choices: all valid time readable attributes.

Time Connections Description (T1 to T8)

Description of the time attribute (e.g., Sta_Time). If the attribute is used in a software model or hardware object, we recommend that you match the attribute's description in the REF block with its description in the model or object. In that way, you will be able to easily identify the attribute in GPL and at the Operator Workstation. Range: 1 to 8 valid characters and symbols.

Time Inputs

Number of configured time data inputs for the USER block. Range: 0 to 4.

Time Outputs

Number of configured time data outputs for the USER block. Range: 0 to 4.

Totalization Typ

Totalization Type. The type of totalization data that the TOT block will obtain. Choices: CURRENT, LAST, or CURRENT AND LAST.

Trouble

The report type that a trouble COS will be reported as. If you do not want a report generated when the object changes to Trouble, select NONE. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW-UP, STATUS, and NONE.

Tune Chng Factor

Tune Change Factor. How quickly the automatic self-tuning algorithm may change the Proportional Band (PBand) and the Integral Time (ITime) attributes. To give the self-tuning algorithm full control of the PBand and ITime values, enter 1.0. To disable self-tuning control, enter 0.0. Range: 0.0 to 1.0.

Tune Noise Band

The range of values within which the feedback can vary before tuning is attempted (Figure 111). The Tune Noise Band is compared to the difference between the setpoint and the input-conditioned feedback. If the difference is greater than twice the Tune Noise Band, tuning may take place. The value of Tune Noise Band must be greater than the Deadband. Range: any positive real number.

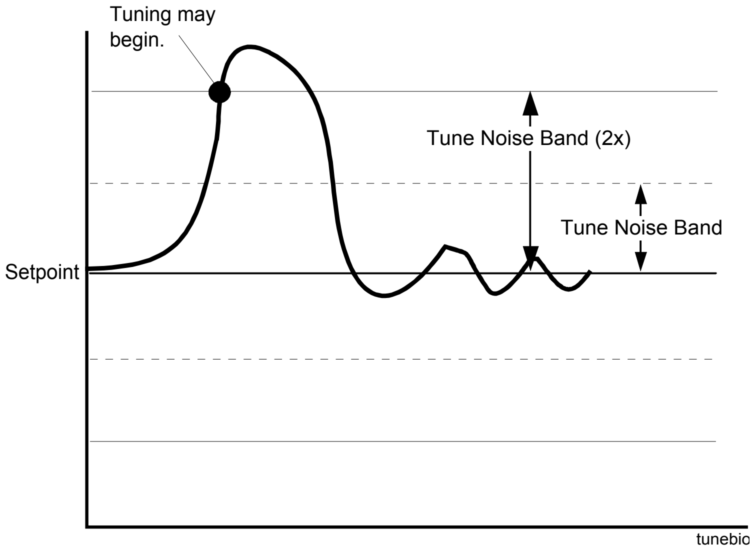


Figure 111: Describing Tune Noise Band

Type

The type of data that the block will use. Choices: ANALOG, BINARY, TIME, CONTROL, COMMAND, DUAL CMD, READ, or WRITE (depends on block).

Type/File Name

The DOS name assigned to the file that contains the code for the USER block. Do not enter the DOS extension as part of the type/file name. Range: up to eight valid characters and symbols.

Type Name

A user-specified name that describes the type of REF block. It appears on the outside of the block. Usually, the user matches it to the Ref. Object Type entry or to the CS type to indicate the model name. Range: 1 to 8 valid characters and symbols.

Units

The type of temperature units for the operation block. Choices: ENGLISH or METRIC.

Unrel Dflt Resp

Unreliable Default Response. A PIDL object output that may be used when the Reliable flag equals 0. Range: all real numbers.

Value

A constant value for the analog or binary input of the CNST block. The block uses this value for its input if you do not make an external connection. Range: all real numbers (analog) or 0 or 1 (binary).

Warmup

Which warmup mode is configured at the C210A controller. Which shutdown mode is configured at the C210A controller. Select a mode according to the following table:

Points Defined	Select This Mode
WMUP, SARE	L2 COMMAND
AUXI, WTMP, CNWU, WMUP, SARE	SARE AND L2 COMMAND
No Warmup Mode Points Defined	NONE

Warning

The report type that a warning will be reported as. Select NONE if you do not want a report generated when the object changes its status to Warning. Choices in descending order: CRITICAL1, CRITICAL2, CRITICAL3, CRITICAL4, FOLLOW_UP, STATUS, and NONE.

Warning #

Warning Number. User-defined reference number that identifies the particular text to be included with a high or low warning COS report. The text is displayed in the dialog box of a critical warning report. Range: 0 to 255 (0 = No message).

Warning Delay

The amount of time in minutes that the object will wait before issuing a COS report. Range: 0 to 255 minutes (0 = no delay).

Write Outputs

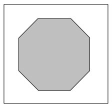
Number of configured write outputs for the USER block. Range: 0 to 8.

Appendix A

This appendix gives a brief description of each icon and option in the GPL Editor. It consists of two sections: major icons and minor icons. The major icons include those on the left side of the screen and in the option menus. The minor icons are those in the connection, command, and find menus. For more detailed information on all icons, refer to the *Editor* chapter.

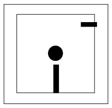
Major Icons

The following are brief descriptions of the major icons, which are used on the left side of the screen and in option menus.



Exit Icon (Stop Sign)

Exits GPL, either to DOS or the Metasys FMS.



File Icon (Disk)

Displays the File option menu.

UP
DWN

UP and DWN Options

UP: Displays the previous page of a directory.

DWN: Displays the next page of a directory.

SAVE

SAVE Option

Saves a control strategy to a disk drive.

LOAD

LOAD Option

Loads a control strategy into the work area.

Description Icon

Displays a description file for a control strategy.

DRIV
DIR

DRIV and DIR Options

DRIV: Changes the currently selected disk drive that is being used to save and load control strategies.

DIR: Creates a new DOS directory.

Delete File Icon (Trash Can)

Deletes a control strategy file.

Compound Icon

Displays the Compound option menu.

UP
DWN

UP and DWN Options

UP: Displays the previous page of a directory.

DWN: Displays the next page of a directory.

MAKE
EDIT

MAKE and EDIT Options

MAKE: Creates a new compound.

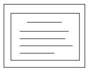
EDIT: Edits an existing compound.

LOAD
XPND

LOAD and XPND Options

LOAD: Loads a compound block into the work area.

XPND: Loads the expanded version of a compound into the work area.



Description Icon

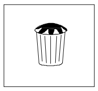
Displays a description file for a compound.

DRIV
DIR

DRIV and DIR Options

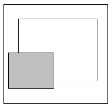
DRIV: Changes the currently selected disk drive that is being used to save and load compounds.

DIR: Creates a new DOS directory.



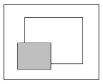
Delete File Icon (Trash Can)

Deletes a compound.



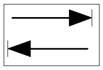
Zoom and Pan Icon

Single click: Zooms a diagram.
Double click: Displays the Zoom and Pan option menu.



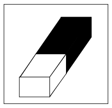
Zoom Icon

Zooms a diagram.



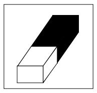
Pan Icon

Moves up, down and across a diagram.



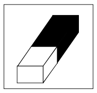
Erase Icon (Eraser)

Single click: Erases any item or deletes an object on a diagram.
Double click: Displays the Erase option menu.



Erase Icon (Eraser)

Erases any item or deletes an object on a diagram.



Erase Group Icon (Framed Eraser)

Erases a group of items on a diagram.

CLR
ALL

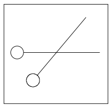
CLR ALL Option

Clears the entire work area, which, in effect, clears all data in RAM memory.

UNDO

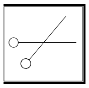
UNDO Option

Restores to the diagram any item previously erased.



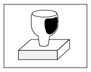
Move Icon (Scissors)

Single click: Moves and resizes an item on the diagram.
Double click: Displays the Move option menu.



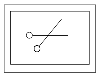
Move Icon (Scissors)

Moves and resizes an item on the diagram.



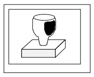
Copy Icon (Stamp)

Copies an item on the diagram.



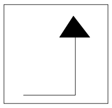
Move Group Icon (Framed Scissors)

Moves and resizes a group of items on a diagram.



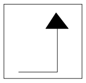
Copy Group Icon (Framed Stamp)

Copies a group of items on a diagram.



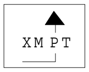
Connection Icon (Arrow)

Single click: Draws a connection line between two function blocks.
Double click: Displays the Connection option menu.



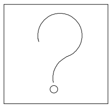
Connection Icon (Arrow)

Draws a connection line between two function blocks.



Exempt Connection Icon (XMPT Arrow)

Makes an existing binary line exempt from triggering a process.



Query Icon (Question Mark)

Single click: Displays a database template, displays the contents of a compound, and queries a connection.
Double click: Displays the contents of a compound or displays the Query option menu.

?

Query Icon (Question Mark)

Single click: Displays a database template, displays the contents of a compound, and queries a connection. Double click: Displays the contents of a compound or displays the Query option menu.

FIND

FIND Option

Locates places in the file where a particular function block is used.

VIEW

VIEW Option

Displays the list file.

READ

READ Option

Reads the archive database to synchronize (match) the GPL object block database with the archive object database.

Print Icon (Page)

Displays the Print option menu.

UP
DWN

UP and DWN Options

UP: Displays the previous page of the print queue.

DWN: Displays the next page of the print queue.

PRT
ONE

PRT ONE Option

Creates a print file for the displayed diagram.

PRT
MANY

PRT MANY Option

Creates a print file for multiple nested diagrams in a control strategy.

START
OUTPUT

START OUTPUT Option

Sends all print files in the queue to the printer.

Delete File Icon (Trash Can)

Deletes a print file from the print queue.

Tools Icon (Hammer)

Single click:	Turns the grid on and off.
Double click:	Displays the Tools option menu.

▦ *Grid Icon*

Turns the grid on and off.

TEXT *TEXT Option*

Types comments, notes and labels on a diagram.

0.00 *0.00 Option*

Pastes down analog displays used to show values during simulation.

✓ *Expert Checker Icon (Check Mark)*

Runs the GPL Expert Checker.

SIM *SIM Option*

Runs the GPL Simulator.

TRAN *TRAN Option*

Runs the GPL Translator.

Minor Icons

The following are brief descriptions of the minor icons, which are used in the connection, command, and find menus.

UP
DWN

UP and DWN Options

UP: Displays the previous page of a list.
DWN: Displays the next page of a list.

ANA

ANA Option

Filters list to show all analog connections.

BIN

BIN Option

Filters list to show all binary connections.

CMD1

CMD1 Option

Highlighted when list shows the commands that can be sent when the Select Input is True (1).

CMD0

CMD0 Option

Highlighted when list shows the commands that can be sent when the Select Input is False (0).

OBJ

OBJ Option

Filters list to show all system names.

BLK

BLK Option

Filters list to show all operation, special, and group compound block names.

TOP
BOT

TOP and BOT Options

TOP: Displays the first page (top) of the list.

BOT: Displays the last page (bottom) of the list.

Appendix B: Summary of Keys

This appendix provides descriptions of the keys that have specific functions for the GPL Editor and Simulator.

GPL Editor Keys

The following list explains the keystrokes used in the Editor.

The GPL Editor uses the following special keys:

Key	Function
Alphanumeric Keys	Enter information such as names, values, and descriptions.
Arrow Keys	Move cursor between parameter fields.
Backspace	Moves cursor back one space within a parameter field.
CTRL/Arrow	Moves cursor within a parameter field.
Esc	Ignores changes made to a template and closes template. Exits a help screen. Clears most items in the work area, such as some option menus and description files.
F1	Displays a help screen.
F2	Changes the function block size to the large default size.
F3	Changes the function block size to the small default size.
F4	Changes the function block size to the standard default size.
F10	Saves changes made to a template and closes template.
Page Up	Displays the next page of a template, help screen, or file.
Page Down	Displays the previous page of a template, help screen, or file.
Insert	Shifts characters one space to the right for you to add a character.
Delete	Deletes a character in a parameter field.

GPL Simulator Keys

The following tables explain the mouse actions and keystrokes used in the three windows of the Simulator.

From the Main Window

Key	Function
Alt Key and Highlighted Letter of Option	Performs option (e.g., press ALT/Y for Yes and ALT/T for CLR TOP).

From the Control Window

Key	Function
Alt Key and Highlighted Letter of Option	Performs option (e.g., press ALT/Q to Quit).
Esc Key	Clears dialog box and cancels previously selected control function.

From the Block Window

Key	Function
Alphanumeric Character Set	Specifies values in fields.
Arrow Keys	Move between data fields to select a field.
Enter	Enters new data in currently selected field.
F5	Triggers execution of process displayed in Block window.
F4	Toggles between Automatic and Manual modes.
PageUp/PageDn	Displays multiple pages of data fields.
Tab	Scrolls through possible entries in some data fields.

Appendix C: Summary of File Names and Extensions

This appendix lists all files that the GPL utilities create and use. The file names for the extensions listed here is the control strategy or compound name
(e.g., C:\FMS\DATA\JOBS123\GPL\AHU1.DB).

File Description	File Type	Extension
Control Strategy	Database	.DB
	Connection	.CI
	Text	.TX
	Database Backup	.ODB
	Connection Backup	.OCI
	Text Backup	.OTX
Compound	Compound	.CMP
	Compound Backup	.OCM
Text Description	Diagram	.DDS
	Compound	.CDS
Print	Diagram	.PRD
	Template	.PRT
List	List	.LST
USER Block	Source Code	.MAC
Intermediate Source	Source Code	.BAS
Process Object	Object Code	.OBJ

The Editor uses two files for online help, which are accessed when you press F1. These files are stored under the C:\{FMSPATH} directory.

File Description	File Type	File Name
Help Screens	Icon	ICON.HLP
	Block	BLOCK.HLP

Appendix D: Capabilities

The following table outlines the capabilities of the GPL Editor.

Item	Maximum
Block database size	64 Kb (approximately)
Connection database size	32 Kb (approximately)
Connections per file	2200
Function blocks per file	999
GPL processes per NCM	255
Line segments per direct connection	14
Line segments per remote connection	7
Nested compound levels	30
Nested FILE blocks	30
Connection penetration levels	10
Files in a subdirectory	100
Sequential connections	120
Text database size (includes analog displays)	40 Kb (approximately)
Text fields	800
Viewable pages for description file	30
Viewable pages for list file	30
Viewable pages for USER block macro file	30
Viewable pages for the block help screens	5
Viewable pages for the icon help screens	3

Appendix E: External Functions

This appendix describes those functions you need to perform outside of the GPL Editor.

Restoring Backup Files and Compounds

The GPL Editor creates a set of backup files each time you edit a control strategy or compound. These files are the version of the strategy or compound before it is updated. You would use the backup files if the originals are inadvertently changed, lost, or damaged.

Notes: The GPL Editor creates backup files automatically; the backup procedure cannot be inhibited.

Any software utility that can copy DOS files and delete DOS directories can be used as an alternate to the procedures that follow.

Restoring the Control Strategy Files

A control strategy has three backup files: .ODB, .OCI, and .OTX. Each must be restored individually. You must be in DOS to perform this procedure.

1. To restore the .ODB file to a .DB file, rename the file by typing in the following at the DOS prompt:

   ```
   rename c:\[path]\[filename].odb
   c:\[path]\[filename].db
   ```

2. To restore the .OCI and .OTX files, repeat Step 1 above, specifying the .OCI/.CI and .OTX/.TX file extensions.

The old version of the control strategy can be loaded as the original. To load the strategy for editing, refer to the *Directory and Control Strategy Functions* section in the *Editor* section.

Restoring a Compound File

A compound has only one backup file: .OCM. You must be in DOS to perform this procedure.

To restore the .OCM file to a .CMP file, rename the file by typing in the following at the DOS prompt:

```
rename c:\[path]\[filename].ocm
c:\[path]\[filename].cmp
```

The old version of the compound has replaced the original. To load the compound for editing, refer to the *Compound Functions* section in the *Editor* chapter.

Deleting a Directory

A directory can be created but not deleted with the GPL Editor. You must be in DOS to do so.

Notes: Do not delete the GPL directory; the archive database requires it.

Refer to your DOS manual for detailed instructions on deleting directories.

To delete a directory, simply type in the following at the DOS prompt:

```
rd c:\[directory name]
```

Writing a Text Description File

Each control strategy and compound can have an associated description file. The purpose of the description files is to explain the strategy or compound. You can display these files in the GPL Editor work area by clicking left on the Description icon.

Follow these rules in creating a description file:

- Create the file outside of GPL with a text editor that can provide a file in ASCII format (i.e., no control characters).
- Define a left margin of 0 (minimum) and a right margin of 69 (maximum). Your file can have a right margin of more than 69, but when viewed with the Editor, the text that is after column 69 will wrap to the next line. The file can be of any size, though only the first 30 pages (900 lines) can be viewed with the GPL Editor.

- Store the control strategy description file in the same directory and assign it the same name as the control strategy, with a .DDS (Diagram Description) extension. Similarly, store the compound description file in the same directory and assign it the same name as the compound, with a .CDS (Compound Description) extension.

Writing a USER Block File

You program each USER block by writing a macro file with a text editor. The macro file contains the JC-BASIC statements and connection names that define the function of the USER block. The connection names specified in the file will be available in the block's input and output connection menus. The valid connection names are:

ANA IN1 to ANA IN8

BIN IN1 to BIN IN8

TIME IN1 to TIME IN4

ANA OUT1 to ANA OUT8

BIN OUT1 to BIN OUT8

TIME OUT1 to TIME OUT4

COMMAND1 to COMMAND8 (output)

WRITE1 to WRITE8 (output)

Creating

Follow these rules in creating a USER block macro file:

- Create the file outside of GPL with a text editor that can provide a file in ASCII format (i.e., no control characters).
- Define a left margin of 0 (minimum) and a right margin of 69 (maximum). Your file can have a right margin of more than 69, but when viewed with the Editor, the text that is after column 69 will wrap to the next line. The file can be of any size, though only the first 30 pages (900 lines) can be viewed with the GPL Editor.

- Store the USER block macro file in the directory that is specified by the SET GPLUMAC command in the GPL.BAT file (e.g., SET GPLUMAC=C:\CUSTUMAC). Assign a .MAC (Macro) extension to the file.

You link the macro file to the pasted down USER block via the Type/File field in the database template. For details, refer to the *Function Blocks* section.

Programming

Follow these rules in writing the JC-BASIC macro file for the USER block:

- Follow the syntax rules for creating JC-BASIC processes. The file may contain any valid combination of JC-BASIC functions and compiler directives that are required to achieve your control objectives. For details on how to write in JC-BASIC, refer to the *JC-BASIC Programmer's Manual*.
- The connection names you define and use in the macro file must match the connection names in the block. Valid connection names are:

ANA IN1 to ANA IN8	COMMAND1 to COMMAND8
ANA OUT1 to ANA OUT8	TIME IN1 to TIME IN4
BIN IN1 to BIN IN8	TIME OUT1 to TIME OUT4
BIN OUT1 to BIN OUT8	WRITE1 to WRITE8

  The connection names must be enclosed in square brackets ([ ]). For example, to represent Analog Input 1, you would specify: [ANA IN1]. However, the connection names for COMMAND1 to COMMAND8 must be enclosed in single parentheses and brackets; for example: '[COMMAND1]'. The Translator will substitute the names of the graphically connected inputs and outputs into the macro file.

- Do not use the PROCESS and END PROCESS statements in the USER block macro file.
- We recommend that you do not use line labels in the USER block macro file. If you were to inadvertently duplicate the USER block in the same process, the second and subsequent uses of the line label would cause a compiler error.

For an application example of a USER block, refer to the *Function Blocks* section.

Appendix F: Characters, Symbols, and Reserved Words

This appendix lists the valid characters and symbols, invalid symbols, and reserved words that apply to character strings, including names of directories, files, objects, blocks, attributes, and disk drives.

The check for the proper directory, file, and disk drive names occurs when you click left on an icon or option that is under the Disk or Compound menu. The check for proper object and block names occurs when you press the F10 key to save the block's database template. The check for proper attribute names occurs when the file is compiled.

International Language Characters

Certain international language characters can be entered with the GPL Editor. The Editor supports French, Canadian French, German, Spanish, and Portuguese international characters. The following table lists the supported international characters.

Ä,Å,à,á,â,ä,å	í,î
ß	Ñ,ñ
Ç,ç	Ö,ó,ô,ö
É,è,é,ê,ë	Ü,ù,ú,û,ü

Note: Do not use international language characters in the following: attribute names, directory/file names, disk drive names, SVAR, FILE, or USER block names.

Character and Symbol Table

The following table lists all valid characters and symbols that can be specified in GPL. An “x” in the column means that the character or symbol is valid for the particular entry. For example, the @ symbol can be used in an object block name.

Symbol	Description	Directory Name or File Name	SVAR Block Name or FILE Block Name	Print and Advisory Block Text Fields	System, Object, Attribute, Compound, USER Block Name or Other Text Fields (See Note a.)
a-z	Lower case letters	x	x	x	x
A-Z	Upper case letters	x	x	x	x
0-9	Numbers	x	x	x	x
See previous table.	International characters		x	x	x
&	Ampersand			x	x
*****	Asterisk			x	
x	At symbol			x	x
****	Backslash			x	
{ and }	Braces			x	x
^	Caret			x	x
:	Colon			x	
,	Comma			x	
$	Dollar			x	x
" and "	Double quotes			x (see Note b)	
=	Equal			x	
!	Exclamation			x	x
-	Hyphen or dash			x	x
> and <	Greater or less than			x	
(and)	Parentheses			x	x
%	Percent			x	x
.	Period			x	
+	Plus			x	
#	Pound			x	x
?	Question mark			x	
;	Semicolon			x	
' and '	Single quotes			x	
/	Slash			x	

Continued on next page . . .

Symbol (Cont.)	Description	Directory Name or File Name	SVAR Block Name or FILE Block Name	Print and Advisory Block Text Fields	System, Object, Attribute, Compound, USER Block Name or Other Text Fields (See Note a.)
( )	Space			x	
[and]	Square brackets			x	
~	Tilde			x	x
_	Underscore	x	x	x	x
|	Vertical bar				

a Other text fields: expanded IDs, REF block description, operation block names, REF block label, block descriptions, etc.

b Double quotes (") must be used in pairs (otherwise, one double quote mark is perceived as a termination character).

Reserved Words

The following list of names cannot be used as directory names, file names, or SVAR, FILE, or USER block names, since they are reserved as DOS device names.

AUX
AUXIN
CLOCKS
COM1
COM2
COM3
COM4
CON
LPT1
LPT2
LPT3
LPT4
NUL
PRN

The following list of names cannot be used as directory names, file names, or SVAR, FILE, or USER block names, since they are reserved as Metasys subdirectory names.

COS
INSTRUCT
DEVICES
GPL
DDL
PROCESS
CAL1

The following list of names cannot be used as SVAR block names since they are JC-BASIC keywords.

ABORT
ABS
ADVISORY
ALARM
ALL
AND
AVG
CANC_PULSE
CLOSED
COS
CRITICAL1
CRITICAL2
CRITICAL3
CRITICAL4
DATE$
DAY
DEF
DELETE
DEWPT
DIFF
DIM
ELSE
END
ENGLISH
ENTHALPYDP
ENTHALPYRH
EXEMPT
EXIT
FALSE
FILTER
FIRST
FOLLOW-UP
FOR
FORCE_REL
FORCE_UNREL
GO
GOSUB
GOTO
HI_ALARM
HI_WARNING
IF
INT_TO_TIME
LASTTOT
LET
LO_ALARM
LO_WARNING
LOG
MAX
METRIC
MIN
MONTH
N_CANC_PULSE
NAME
NEXT
NORMAL
NOT
OFF
ON
OPEN
OR
PASS
PERIOD
PI
PI_RESET
PRINT
PRIORITY
PROCESS
RAMP
REL_HUM
REM
RESET
RETURN
SET
SHARED
SPAN
SQR
STATUS
SIN
STEP
STOP
SUB
TAN
TELL
THEN
TIME
TIME$
TIME_TO_INT
TO
TODAY
TOTAL
TROUBLE
TRUE
UNRELIABLE
USING
WAIT
WETDP
WETRH
WHILE
XOR
YEAR

#INCLUDE
#REPLACE
#REMARKS
#NOLIST
#LIST
#NOREMARKS

Appendix G: Attributes

This appendix contains a list of all readable, writable, and triggerable attributes for each object. The readable and writable attributes are those that you can specify with the READ and WRIT function blocks, respectively. The triggerable attributes are those that may cause a process to execute. The letter “x” in a column indicates which attribute is readable, writable, or triggerable. For more details on these and all attributes, refer to the technical bulletins for the objects in the *Metasys Network Technical Manual*.

ACM Object

Attribute Name	Type	Readable	Writable	Triggerable
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
COS_DEL	Binary	x		
DB_FILT	Integer	x		
DEADBAND	Float. Pt.	x		
DELAY	Integer	x		
DIAL_UP	Binary	x	x	
DIFF	Float. Pt.	x		
DISCONCT	Binary	x		x
FBK_PROB	Binary	x		x
FILTER	Float. Pt.	x		
FORMAT	Integer	x		
GRAPHIC	Integer	x	x	
HI_ALARM	Binary	x		x
HI_LIMIT	Float. Pt.	x		
HI_WARN	Binary	x		x
HI_WARNL	Float. Pt.	x		
HISTORY	Binary	x		
INSTRUCT	Integer	x	x	
LED_STAT	Binary	x		
LO_ALARM	Binary	x		x
LO_LIMIT	Float. Pt.	x		
LO_WARN	Binary	x		x
LO_WARNL	Float. Pt.	x		
LOC_CNTL	Binary	x		
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
NORMBAND	Float. Pt.	x		
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PPM	Float. Pt.	x		
PREFIX	Binary	x		
PT_TYPE	Integer	x		
RATE_K	Integer	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		

Continued on next page . . .

Attribute Name (Cont.)	**Type**	**Readable**	**Writable**	**Triggerable**
SCAN	Binary	x		
SETPOINT	Float. Pt.	x		
SLOT	Integer	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
TOTAL	Integer	x		
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		
WARN_MSG	Integer	x	x	
WARN_RPT	Integer	x	x	

AD Object

Attribute Name	Type	Readable	Writable	Triggerable
ADJ_DIS	Binary	x		
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
ASS_ATTR	Integer	x		
CMD_PRI	Integer	x		
COS_DEL	Binary	x		
DELAY	Integer	x		
DIAL_UP	Binary	x	x	
DIFF	Float. Pt.	x		
DISCONCT	Binary	x		x
FEATURE	Integer	x		
FILTER	Float. Pt.	x		
FORMAT	Integer	x		
GRAPHIC	Integer	x	x	
HI_ALARM	Binary	x		x
HI_LIMIT	Float. Pt.	x		
HI_WARN	Binary	x		x
HI_WARNL	Float. Pt.	x		
HISTORY	Binary	x		
INITIAL	Float. Pt.	x	x	
INSTRUCT	Integer	x	x	
LO_ALARM	Binary	x		x
LO_LIMIT	Float. Pt.	x		
LO_WARN	Binary	x		x
LO_WARNL	Float. Pt.	x		
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
NORMBAND	Float. Pt.	x		
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SETPOINT	Float. Pt.	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		
WARN_MSG	Integer	x	x	
WARN_RPT	Integer	x	x	

AHU, UNT, VAV, MIG, PHX, VND, NDM Hardware Objects

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x		

AI Object

Attribute Name	Type	Readable	Writable	Triggerable
AD_COUNT	Integer	x		
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
COS_DEL	Binary	x		
DELAY	Integer	x		
DIAL_UP	Binary	x	x	
DIFF	Float. Pt.	x		
DISCONCT	Binary	x		x
FILTER	Float. Pt.	x		
FLOW_K	Float. Pt.	x		
FLTR_VAL	Float. Pt.	x		
FORMAT	Integer	x		
GRAPHIC	Integer	x	x	
HI_ALARM	Binary	x		x
HI_LIMIT	Float. Pt.	x		
HI_WARN	Binary	x		x
HI_WARNL	Float. Pt.	x		
HISTORY	Binary	x		
INSTRUCT	Integer	x	x	
LO_ALARM	Binary	x		x
LO_LIMIT	Float. Pt.	x		
LO_WARN	Binary	x		x
LO_WARNL	Float. Pt.	x		
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
NORMBAND	Float. Pt.	x		
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PRE_VAL	Float. Pt.	x		
PREFIX	Binary	x		
RANGE_1	Float. Pt.	x		
RANGE_2	Float. Pt.	x		
RANGE_3	Float. Pt.	x		
RANGE_4	Float. Pt.	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
SCAN	Binary	x		
SETPOINT	Float. Pt.	x		
SLOT	Integer	x		
SPAN_IN1	Float. Pt.	x		
SPAN_IN2	Float. Pt.	x		
SPAN_OT1	Float. Pt.	x		
SPAN_OT2	Float. Pt.	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
STDRANGE	Integer	x		
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		
WARN_MSG	Integer	x	x	
WARN_RPT	Integer	x	x	

AOD Object

Attribute Name	Type	Readable	Writable	Triggerable
CMD_PRI	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
FEATURE	Integer	x		
FORMAT	Integer	x		
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
HOA	Binary	x		x
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
PT_TYPE	Integer	x		
REPORT	Binary	x		
RESTORE	Binary	x	x	
SAT	Integer	x	x	
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SLOT	Integer	x		
SPAN_IN1	Float. Pt.	x		
SPAN_IN2	Float. Pt.	x		
SPAN_OT1	Float. Pt.	x		
SPAN_OT2	Float. Pt.	x		
STATDISP	Integer	x		
STEP	Float. Pt.	x	x	
TRIGGER	Binary	x		

AOS Object

Attribute Name	Type	Readable	Writable	Triggerable
CMD_PRI	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
FB_SET	Binary	x		
FEATURE	Integer	x		
FORMAT	Integer	x		
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
HOA	Binary	x		x
INITIAL	Float. Pt.	x		
INSTRUCT	Integer	x	x	
LOC_CNTL	Binary	x		x
LOC_ELIG	Binary	x		
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
PT_TYPE	Integer	x		
REPORT	Binary	x		
RESTORE	Binary	x	x	
SAT	Integer	x	x	
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SLOT	Integer	x		
SPAN_IN1	Float. Pt.	x		
SPAN_IN2	Float. Pt.	x		
SPAN_OT1	Float. Pt.	x		
SPAN_OT2	Float. Pt.	x		
STATDISP	Integer	x		
STEP	Float. Pt.	x	x	
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		

BD Object

Attribute Name	Type	Readable	Writable	Triggerable
ADJ_DIS	Binary	x		
ALARM	Integer	x		x
ALR_GEN	Binary	x	x	
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
ASS_ATTR	Integer	x		
CMD_PRI	Integer	x		
COS_DEL	Binary	x		x
DELAY	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
EARLY_TM	Time	x		
FEATURE	Integer	x		
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
INITIAL	Binary	x		
INSTRUCT	Integer	x	x	
LATCH	Binary	x		
LATCHING	Binary	x		
LATE_TM	Time	x		
NOR_COND	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Binary	x		x

BI Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Integer	x		x
ALR_GEN	Binary	x	x	
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
COS_DEL	Binary	x		x
DB_FILT	Integer	x		
DELAY	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
INP_TYPE	Integer	x		
INSTRUCT	Integer	x	x	
LATCH	Binary	x		
LATCHING	Binary	x		
LED_STAT	Binary	x		
NOR_COND	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
POINT_NO	Integer	x		
PREFIX	Binary	x		
PT_ENA	Binary	x		
PT_TYPE	Integer	x		
REPORT	Binary	x		
RLAY_RST	Binary	x	x	
RLAY_SET	Binary	x	x	
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SLOT	Integer	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
STI_NO	Integer	x		

Continued on next page . . .

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
SUB_SLOT	Integer	x		
SUPP_TZ	Integer	x	x	
TRIGGER	Binary	x		
TROUBLE	Binary	x		x
VALUE	Binary	x		x

BO Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Integer	x		
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
CMD_ACTN	Binary	x		
CMD_PRI	Integer	x		
COS_DEL	Binary	x		x
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
EARLY_TM	Time	x		
FB_ACTN	Binary	x		
FB_SET	Binary	x		
FBK_PROB	Binary	x		
FEATURE	Integer	x		
FEEDBACK	Binary	x		x
GRAPHIC	Integer	x	x	
HE_DELAY	Integer	x	x	
HISTORY	Binary	x		
HOA	Binary	x		x
INITIAL	Binary	x		
INSTRUCT	Integer	x	x	
LATE_TM	Time	x		
LED_STAT	Binary	x		
LOC_CNTL	Binary	x		x
LOC_ELIG	Binary	x		
LOCK	Binary	x		
LSTATUS	Binary	x		
LMIN_OFF	Integer	x	x	
LMIN_ON	Integer	x	x	
LOAD	Binary	x		
MAX_OFF	Integer	x	x	
MAX_STA	Integer	x	x	
MIN_OFF	Integer	x	x	
MIN_ON	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
PREFIX	Binary	x		
PT_TYPE	Integer	x		
PULSE	Integer	x		
RATE	Float. Pt.	x		
REPORT	Binary	x		
RESTORE	Binary	x	x	
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SLOT	Integer	x		
STATDISP	Integer	x		
STATE_0	Binary	x		x
STATE_1	Binary	x		x
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Binary	x		x

C210A Object

Attribute Name	Type	Readable	Writable	Triggerable
ACTC	Float. Pt.	x		
AINUM	Integer	x	x	
AUXAAPP	Binary	x		
AUXBAPP	Binary	x		
AUXI	Float. Pt.	x		
AUXIOVR	Integer	x		
AUXP	Float. Pt.	x		
AUXPOVR	Integer	x		
AUXR	Float. Pt.	x		
AUXROVR	Integer	x		
AUXT	Float. Pt.	x		
AUXTOVR	Integer	x		
AXBI	Binary	x		
AXDP	Float. Pt.	x		
AXDPOVR	Integer	x		
CNWU	Binary	x		
DAMPAPP	Integer	x		
DFPR	Float. Pt.	x		
DFPROVR	Integer	x		
DIAL_UP	Binary	x	x	
DMPR	Float. Pt.	x		
DPSP	Float. Pt.	x		
DPSPOVR	Integer	x		
FANAPP	Integer	x		
GRAPHIC	Integer	x	x	
HCPB	Float. Pt.	x		
HCPBOVR	Integer	x		
HEATAPP	Integer	x		
HLTC	Binary	x		
HRTZ	Binary	x		
HRTZOVR	Integer	x		
HTDB	Float. Pt.	x		
HTDBOVR	Integer	x		
HWSD	Binary	x		
HWUO	Binary	x		
INTE	Float. Pt.	x		
INTG	Float. Pt.	x		
INTGOVR	Integer	x		

Continued on next page . . .

Attribute Name (Cont.)	**Type**	**Readable**	**Writable**	**Triggerable**
INSTRUCT	Integer	x	x	
LTCH	Binary	x		
LTCHOVR	Integer	x		
LV12	Binary	x		
MNDP	Float. Pt.	x		
MNDPOVR	Integer	x		
MXDP	Float. Pt.	x		
MXDPOVR	Integer	x		
OCCAPP	Integer	x		
OFFLINE	Binary	x		x
OFRH	Float. Pt.	x		
OVERRIDE	Binary	x		
OVF_SETP	Binary	x		
OVR_RPT	Integer	x	x	
PMAX	Binary	x		
PFOF	Binary	x		
PFSS	Binary	x		
PRCL	Float. Pt.	x		
PREFIX	Binary	x		
PRRH	Float. Pt.	x		
REHT	Binary	x		
REPORT	Binary	x		
RZSP	Float. Pt.	x		
RZSPOVR	Integer	x		
SARE	Binary	x		
SAREOVR	Integer	x		
SCAN	Binary	x		
SDBC	Binary	x		
SDBCOVR	Integer	x		
SDBO	Binary	x		
SDBOOVR	Integer	x		
SETPAPP	Integer	x		
SETPOINT	Float. Pt.	x		
SFOF	Binary	x		
SFSU	Binary	x		
SHTDAPP	Integer	x		
SPNUM	Integer	x	x	
STATDISP	Integer	x		
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
SUSB	Float. Pt.	x		
SUSBOVR	Integer	x		
TCMD	Integer	x		
TRIGGER	Binary	x		
UNOC	Binary	x		
UNOCOVR	Integer	x		
VALUE	Float. Pt.	x		
WMUP	Binary	x		
WMUPAPP	Integer	x		
WMUPOVR	Integer	x		
WTMP	Float. Pt.	x		
WTMPOVR	Integer	x		
ZNSP	Float. Pt.	x		
ZNSPOVR	Integer	x		
ZNT	Float. Pt.	x		
ZNTOVR	Integer	x		

C260A Object

Attribute Name	Type	Readable	Writable	Triggerable
AINUM	Integer	x	x	
ATMP	Float. Pt.	x		
ATMPOVR	Integer	x		
AURH	Float. Pt.	x		
AURHOVR	Integer	x		
AUXAAPP	Binary	x		
AUXI	Float. Pt.	x		
AUXIOVR	Integer	x		
AUXR	Float. Pt.	x		
AUXROVR	Integer	x		
AUXT	Float. Pt.	x		
AUXTOVR	Integer	x		
AXBI	Binary	x		
AXDP	Float. Pt.	x		
AXDPOVR	Integer	x		
CLPB	Float. Pt.	x		
CLPBOVR	Integer	x		
CNWU	Binary	x		
CPCM	Float. Pt.	x		
DAMPAPP	Integer	x		
DIAL_UP	Binary	x	x	
FANAPP	Integer	x		
FLOW	Binary	x		
FNON	Binary	x		
FOCO	Binary	x		
FOCOOVR	Integer	x		
GRAPHIC	Integer	x	x	
HCPB	Float. Pt.	x		
HCPBOVR	Integer	x		
HLTC	Binary	x		
HRTZ	Binary	x		
HRTZOVR	Integer	x		
HTCM	Float. Pt.	x		
HTDB	Float. Pt.	x		
HTDBOVR	Integer	x		
HTPB	Float. Pt.	x		
HTPBOVR	Integer	x		
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
HWSD	Binary	x		
HWUO	Binary	x		
INTE	Float. Pt.	x		
INTG	Float. Pt.	x		
INTGOVR	Integer	x		
INSTRUCT	Integer	x	x	
LTCH	Binary	x		
LTCHOVR	Integer	x		
LV12	Binary	x		
MNDP	Float. Pt.	x		
MNDPOVR	Integer	x		
MXDP	Float. Pt.	x		
MXDPOVR	Integer	x		
OCCAPP	Integer	x		
OFFLINE	Binary	x		x
OVR_SETP	Binary	x		
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PE	Float. Pt.	x		
PFOF	Binary	x		
PFSS	Binary	x		
PRCL	Float. Pt.	x		
PREFIX	Binary	x		
PRRH	Float. Pt.	x		
REHT	Binary	x		
REPORT	Binary	x		
RVAL	Binary	x		
RZSP	Float. Pt.	x		
RZSPOVR	Integer	x		
SCAN	Binary	x		
SDWN	Binary	x		
SDWNOVR	Integer	x		
SETPAPP	Integer	x		
SETPOINT	Float. Pt.	x		
SFOF	Binary	x		
SFSU	Binary	x		
SHTDAPP	Integer	x		
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
SPNUM	Integer	x	x	
STATDISP	Integer	x		
STUP	Float. Pt.	x		
TCMD	Integer	x		
TRIGGER	Binary	x		
UNOC	Binary	x		
UNOCOVR	Integer	x		
VALUE	Float. Pt.	x		
ZNSP	Float. Pt.	x		
ZNSPOVR	Integer	x		
ZNT	Float. Pt.	x		
ZNTOVR	Integer	x		

C260X Object

Attribute Name	Type	Readable	Writable	Triggerable
AINUM	Integer	x	x	
AISUSED	Integer	x		
AI1OVR - AI6OVR	Binary	x		
AI1VAL - AI6VAL	Float. Pt.	x		
AOSUSED	Integer	x		
AO1OVR - AO2OVR	Binary	x		
AO1VAL - AO2VAL	Float. Pt.	x		
BISUSED	Integer	x		
BI1OVR - BI4OVR	Binary	x		
BI1VAL - BI4VAL	Binary	x		
BOSUSED	Integer	x		
BO1OVR - BO5OVR	Binary	x		
BO1VAL - BO5VAL	Binary	x		
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x		
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
OVR_SETP	Binary	x		
PREFIX	Binary	x		
REPORT	Binary	x		
SCAN	Binary	x		
SETPOINT	Float. Pt.	x		
SPNUM	Integer	x	x	
SPSUSED	Integer	x		
SP1OVR - SP13OVR	Binary	x		
SP1VAL - SP13VAL	Float. Pt.	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		

C500X Object

Attribute Name	Type	Readable	Writable	Triggerable
AINUM	Integer	x	x	
AISUSED	Integer	x		
AI1OVR - AI6OVR	Binary	x		
AI1VAL - AI6VAL	Float. Pt.	x		
AOSUSED	Integer	x		
AO1OVR - AO6OVR	Binary	x		
AO1VAL - AO6VAL	Float. Pt.	x		
BISUSED	Integer	x		
BI1OVR - BI5OVR	Binary	x		
BI1VAL - BI5VAL	Binary	x		
BOSUSED	Integer	x		
BO1OVR - BO4OVR	Binary	x		
BO1VAL - BO4VAL	Binary	x		
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x		
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
OVR_SETP	Binary	x		
PREFIX	Binary	x		
REPORT	Binary	x		
SCAN	Binary	x		
SETPOINT	Float. Pt.	x		
SPNUM	Integer	x	x	
SPSUSED	Integer	x		
SP1OVR - SP16OVR	Binary	x		
SP1VAL - SP16VAL	Float. Pt.	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
VALUE	Float. Pt.	x		

CS Object

Attribute Name	Type	Readable	Writable	Triggerable
ADADJEN	Binary	x		
ADADJS	Binary	x		
ADOVREN	Binary	x		
ADOVRS	Integer	x		
AD_1 - AD_32	Float. Pt.	x		
AIADJEN	Binary	x		
AIADJS	Binary	x		
AIOVREN	Binary	x		
AIOVRS	Integer	x		
AD_1 - AI_16	Float. Pt.	x		
AOADJEN	Binary	x		
AOOVREN	Binary	x		
AOOVRS	Integer	x		
AO_1 - AO_16	Float. Pt.	x		
BDADJEN	Binary	x		
BDADJS	Binary	x		
BDOVREN	Binary	x		
BDOVRS	Integer	x		
BD_1 - BD_32	Binary	x		x
BIADJEN	Binary	x		
BIOVREN	Binary	x		
BIOVRS	Integer	x		
BI_1 - BI_16	Binary	x		x
BOADJEN	Binary	x		
BOADJS	Binary	x		
BOOVREN	Binary	x		
BOOVRS	Integer	x		
BO_1 - BO_16	Binary	x		x
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
FORMAT	Integer	x	x	
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
MSADJEN	Binary	x		
MSADJS	Binary	x		
MSVALUE0 - MSVALUE4	Integer	x		
MSOVREN	Binary	x		

Continued on next page . . .

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
MS_1 - MS_2	Integer	x		
NTCMDADJ	Binary	x		
NTCMDATR	Integer	x	x	
NUMAD	Integer	x		
NUMAI	Integer	x		
NUMAO	Integer	x		
NUMBD	Integer	x		
NUMBI	Integer	x		
NUMBO	Integer	x		
NUMMS	Integer	x		
NUMSP	Integer	x		
OBJVAL	Integer	x	x	
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
REPORT	Binary	x		
SCAN	Binary	x		
SPADJEN	Binary	x		
SPADJS	Binary	x		
SPOVREN	Binary	x		
SPOVRS	Integer	x		
SP_1 - SP_32	Float. Pt.	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		

D600 Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
ACDWNEED	Binary	x		
AC_FAIL	Binary	x		
AC_TAMP	Binary	x		
BAFE	Long	x		
BAT_LOW	Binary	x		
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
DL_IN_PR	Binary	x		
ENCODE	Integer	x		
G01 through G64	Binary	x		x
GLO_ACC	Binary	x		
GRAPHIC	Integer	x	x	
ILK_TIME	Integer	x	x	
INSTRUCT	Integer	x	x	
INXIT	Binary	x		
MAG	Long	x		
NCRYPT	Long	x		
NOREPVCD	Binary	x		
OFFLINE	Binary	x		x
OP_CHNG	Binary	x		
PIN_5	Binary	x		
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
REP_ALM	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x		
TZ_CHEK	Binary	x		
VALUE	Binary	x		
WIEG	Long	x		

DC9100/DR9100 Hardware Objects

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

DCM/DCM140 Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT*	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

* The DISCONCT attribute is not supported by the DCM140.

DSC-1000
Hardware Objects

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_TYPE	Integer	x		
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

DSC8500
Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
DL_PROG	Binary	x		
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
PREFIX	Binary	x		
S2_TYPE	Integer	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

DX9100/DX91ECH Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

Fire Controller Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
AC_FAIL	Binary	x		x
ALARM	Binary	x		x
ALR_RPT	Integer	x	x	
ALRE_RPT	Integer	x	x	
BAT_LOW	Binary	x		
DB_MATCH	Binary	x		
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
DL_IN_PR	Binary	x		
DL_REQR	Binary	x		
DNLD_ENB	Binary	x		
EARL_MSG	Integer	x	x	
ETBL_MSG	Integer	x	x	
EVT_RPT	Integer	x	x	
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
LALR_MSG	Integer	x	x	
LTBL_MSG	Integer	x	x	
NOR_RPT	Integer	x	x	
NORE_RPT	Integer	x	x	
OFFLINE	Binary	x		x
OPER_ENB	Binary	x	x	
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
QY_IN_PR	Binary	x		
REPORT	Binary	x		
SCAN	Binary	x		
SGNL_SIL	Binary	x		x
STATDISP	Integer	x		
STATUS	Integer	x		x
TBL_RPT	Integer	x	x	
TBLE_RPT	Integer	x	x	
TRIGGER	Binary	x		
TRUNK	Integer	x	x	
VALUE	Binary	x		x
ZONE_BND	Integer	x		

FPU Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
PREFIX	Binary	x		
S2_TYPE	Integer	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

LCD Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
ALM_STAT	Binary	x		
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x	x	
IN_STAT	Binary	x		
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
OUT_STAT	Binary	x		
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

LCG Object

Attribute Name	Type	Readable	Writable	Triggerable
ASSOC_IN	Integer	x	x	
BLNK_FLG	Binary	x	x	
CC_FEAT	Integer	x		
CTOT_HR	Integer	x	x	
CTOT_MIN	Integer	x	x	
DIAL_UP	Binary	x	x	
EVNT1TYP	Integer	x	x	
EVNT1STH	Integer	x	x	
EVNT1STM	Integer	x	x	
EVNT1SPH	Integer	x	x	
EVNT1SPM	Integer	x	x	
EVNT2TYP	Integer	x	x	
EVNT2STH	Integer	x	x	
EVNT2STM	Integer	x	x	
EVNT2SPH	Integer	x	x	
EVNT2SPM	Integer	x	x	
EVNT3TYP	Integer	x	x	
EVNT3STH	Integer	x	x	
EVNT3STM	Integer	x	x	
EVNT3SPH	Integer	x	x	
EVNT3SPM	Integer	x	x	
EVNT4TYP	Integer	x	x	
EVNT4STH	Integer	x	x	
EVNT4STM	Integer	x	x	
EVNT4SPH	Integer	x	x	
EVNT4SPM	Integer	x	x	
GRAPHIC	Integer	x	x	
GRP_NUM	Integer	x		
HISTORY	Binary	x		
INSTRUCT	Integer	x	x	
IN_TYPE	Integer	x	x	
NOR_RPT	Integer	x	x	
OFF_INPT	Integer	x	x	
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_DLY	Integer	x	x	
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		

Continued on next page . . .

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
PTOT_HR	Integer	x	x	
PTOT_MIN	Integer	x	x	
REPORT	Binary	x		
SAVE_HIS	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
VALUE	Binary	x		x

LCP Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

LON Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DIAL_UP	Binary	x	x	
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

MC Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Binary	x		x
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
ASS_ATTR	Integer	x		
CMD_PRI	Integer	x		
CNTL_FEA	Binary	x		
COS_DEL	Binary	x		x
DELAY	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
EARLY_TM	Time	x		
FEATURE	Integer	x		
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
INITIAL	Integer	x		
INSTRUCT	Integer	x	x	
LATCH	Binary	x		
LATCHING	Binary	x		
LATE_TM	Time	x		
LMIN_OFF	Integer	x	x	
LMIN_ON	Integer	x	x	
LOAD_PRI	Integer	x		
LOCK	Binary	x		
LSTATUS	Binary	x		
MAX_OFF	Integer	x	x	
NOR_COND	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
RATE_1	Real	x		
RATE_2	Real	x		
RATE_3	Real	x		
REL_LEFT	Integer	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		

Continued on next page . . .

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
SCAN	Binary	x		
SENDING	Binary	x		x
SHD_LEFT	Integer	x		
SHED_STA	Integer	x		
SL_COUNT	Integer	x		
STATDISP	Integer	x		
STATE_0	Binary	x		x
STATE_1	Binary	x		x
STATE_2	Binary	x		x
STATE_3	Binary	x		x
STATES	Integer	x		
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Integer	x		x

MSD Object

Attribute Name	Type	Readable	Writable	Triggerable
ADJ_DIS	Binary	x		
ALARM	Binary	x		x
ALR_GEN	Binary	x	x	
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
ASS_ATTR	Integer	x		
CMD_PRI	Integer	x		
COS_DEL	Binary	x		x
DELAY	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
EARLY_TM	Time	x		
FEATURE	Integer	x		
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
INITIAL	Integer	x		
INSTRUCT	Integer	x	x	
LATCH	Binary	x		
LATCHING	Binary	x		
LATE_TM	Time	x		
NOR_COND	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
STATE_0	Binary	x		x
STATE_1	Binary	x		x
STATE_2	Binary	x		x
STATE_3	Binary	x		x
STATES	Integer	x		
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Integer	x		x

MSI Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Binary	x		x
ALR_GEN	Binary	x	x	
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
COS_DEL	Binary	x		x
DB_FILT	Integer	x		
DELAY	Integer	x	x	
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
HISTORY	Binary	x		
INSTRUCT	Integer	x	x	
LATCH	Binary	x		
LATCHING	Binary	x		
LED_STAT	Binary	x		
NOR_COND	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
PT_TYPE	Integer	x		
REPORT	Binary	x		
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SEQUENCE	Integer	x		
SLOT	Integer	x		
STATDISP	Integer	x		
STATE_0	Binary	x		x
STATE_1	Binary	x		x
STATE_2	Binary	x		x
STATE_3	Binary	x		x
STATES	Integer	x		
STATUS	Integer	x		x
SUB_SLOT	Integer	x		
TRIGGER	Binary	x		
VALUE	Integer	x		x
WIRED_0	Binary	x		

MSO Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Binary	x		x
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
AUTO_MAN	Binary	x		
CMD_ACTN	Binary	x		
CMD_PRI	Integer	x		
COS_DEL	Binary	x		x
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
EARLY_TM	Time	x		
FB_ACTN	Binary	x		
FB_SET	Binary	x		
FBK_PROB	Binary	x		
FEATURE	Integer	x		
FEEDBACK	Integer	x		x
GRAPHIC	Integer	x	x	
HE_DELAY	Integer	x	x	
HISTORY	Binary	x		
HOA	Binary	x		x
INITIAL	Integer	x		
INSTRUCT	Integer	x	x	
LATE_TM	Time	x		
LC_DB_FI	Integer	x		
LC_HOA	Binary	x		x
LC_LED	Binary	x		
LC_PROB	Binary	x		x
LC_PTTYP	Integer	x		
LC_SBSLT	Integer	x		
LC_SEQ	Integer	x		
LC_SET	Binary	x		x
LC_SLOT	Integer	x		
LC_WRD_0	Binary	x		
LED_STAT	Binary	x		
LMIN_OFF	Integer	x	x	
LMIN_ON	Integer	x	x	
LOAD	Binary	x		
LOC_CNTL	Binary	x		x
LOC_ELIG	Binary	x		
Continued on next page . . .				

Attribute Name (Cont.)	**Type**	**Readable**	**Writable**	**Triggerable**
LOCK	Binary	x		
LSTATUS	Binary	x		
MAX_OFF	Integer	x	x	
MAX_STA	Integer	x	x	
MIN_OFF	Integer	x	x	
MIN_ON	Integer	x	x	
NOR_RPT	Integer	x	x	
NORMAL	Binary	x		x
OFFLINE	Binary	x		x
OVERRIDE	Binary	x		
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
PT_TYPE	Integer	x		
PULSE	Integer	x		
RATE_1	Float. Pt.	x		
RATE_2	Float. Pt.	x		
RATE_3	Float. Pt.	x		
REPORT	Binary	x		
RESTORE	Binary	x	x	
SAVE_HIS	Binary	x		
SCAN	Binary	x		
SEQUENCE	Integer	x		
SLOT	Integer	x		
STATDISP	Integer	x		
STATE_0	Binary	x		x
STATE_1	Binary	x		x
STATE_2	Binary	x		x
STATE_3	Binary	x		x
STATES	Integer	x		
STATUS	Integer	x		x
TRIGGER	Binary	x		
VALUE	Integer	x		x

PIDL Object

Attribute Name	Type	Readable	Writable	Triggerable
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
AUX_ENA	Binary	x	x	
AUX_IN	Float. Pt.	x		
AUX_PRI	Integer	x		
AUX_REF	Binary	x		
DEADBAND	Float. Pt.	x	x	
DEFAULT	Float. Pt.	x	x	
DIAL_UP	Binary	x	x	
DWEIGHT	Float. Pt.	x	x	
FEEDBACK	Float. Pt.	x		
FILTER	Float. Pt.	x	x	
FLTR_VAL	Float. Pt.	x		
FORMAT	Integer	x	x	
GRAPHIC	Integer	x	x	
HI_SAT_F	Binary	x		x
HI_SAT_V	Float. Pt.	x		
HISAT_RF	Binary	x		
HSAT_PRI	Integer	x		
HYST_CMP	Float. Pt.	x	x	
IN_FUNC	Integer	x	x	
INP1_PRI - INP6_PRI	Integer	x		
INP1REF - INP6REF	Binary	x		
INP1VAL - INP6VAL	Float. Pt.	x		
INSTRUCT	Integer	x	x	
INT_TIME	Float. Pt.	x	x	
LO_SAT_F	Binary	x		x
LO_SAT_V	Float. Pt.	x		
LOOP_NUM	Integer	x		
LSAT_PRI	Integer	x		
LSAT_RF	Binary	x		
NOR_RPT	Integer	x	x	
OFFLINE	Binary	x		x
OFFSET	Float. Pt.	x		
OFS_PRI	Integer	x		
OFS_REF	Binary	x		
OUT1_ATT - OUT8_ATT	Integer	x		
OVERRIDE	Binary	x		
Continued on next page . . .				

Attribute Name (Cont.)	Type	Readable	Writable	Triggerable
OVF_HSAT	Binary	x		
OVF_IN1 - OVF_IN6	Binary	x		
OVF_LSAT	Binary	x		
OVF_OFFS	Binary	x		
OVF_SEL	Binary	x		
OVF_SETP	Binary	x		
OVF_SWCH	Binary	x		
OVR_RPT	Integer	x	x	
PERIOD	Integer	x	x	
PID_CALC	Float. Pt.	x		
PIDA_REL	Binary	x		x
PREFIX	Binary	x		
PROPBAND	Float. Pt.	x	x	
RELIABLE	Binary	x		x
REPORT	Binary	x		
SCALAR1 - SCALAR6	Float. Pt.	x	x	
SCAN	Binary	x		
SEL_FLAG	Binary	x		x
SEL _INP	Float. Pt.	x		
SEL_OUT	Float. Pt.	x		
SEL_PRI	Integer	x		
SEL_REF	Binary	x		
SELTYPE	Binary	x	x	
SETPOINT	Float. Pt.	x		
STATDISP	Integer	x		
STP_PRI	Integer	x		
STP_REF	Binary	x		
SWCH_OUT	Float. Pt.	x		
TRIGGER	Binary	x		
TUNE_BND	Float. Pt.	x	x	
TUNEMODE	Integer	x		
TUNE_WT	Float. Pt.	x	x	
UNR_TYPE	Binary	x	x	
VALUE	Float. Pt.	x		

READER Object

Attribute Name	Type	Readable	Writable	Triggerable
ACC_INT	Integer	x	x	
ACC_SEC	Binary	x		x
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
AL_SHUNT	Integer	x	x	
ANTI_PAS	Binary	x	x	
ANTI_TAI	Binary	x	x	
AUX_ACC	Binary	x	x	
CARD_TYP	Integer	x	x	
DIAL_UP	Binary	x	x	
FAC_BAK	Binary	x	x	
GLO_ACC	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
NOR_RPT	Integer	x	x	
OFFLINE	Binary	x		x
OVER_SCH	Binary	x		
OVER_TZ	Integer	x	x	
PASS_INT	Integer	x	x	
PIN_BAK	Binary	x	x	
PIN_TZ	Integer	x	x	
PREFIX	Binary	x		
RDR_TYP	Integer	x	x	
RDR_TZ	Integer	x	x	
REPORT	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
VALUE	Binary	x		

TC9100 Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

XM Hardware Objects (XBN, XRM, XRL)

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_TYPE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

XT9100/XTM Hardware Object

Attribute Name	Type	Readable	Writable	Triggerable
DEV_ADDR	Integer	x	x	
DEV_CODE	Integer	x		
DIAL_UP	Binary	x	x	
DISCONCT	Binary	x		x
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
OFFLINE	Binary	x		x
POLL_PRI	Integer	x	x	
PREFIX	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
TRIGGER	Binary	x		
TRUNK	Integer	x	x	

ZONE Object

Attribute Name	Type	Readable	Writable	Triggerable
ALARM	Binary	x		x
ALR_MSG	Integer	x	x	
ALR_RPT	Integer	x	x	
ANN_PANL	Integer	x		
ANN_PNT	Integer	x		
APPLICAT	Integer	x		
DIAL_UP	Binary	x	x	
DIS_MSG	Integer	x	x	
DISABLED	Binary	x		
DNLD_ENB	Binary	x		
GRAPHIC	Integer	x	x	
INSTRUCT	Integer	x	x	
NOR_MSG	Integer	x	x	
NOR_RPT	Integer	x	x	
OFFLINE	Binary	x		x
OVR_RPT	Integer	x	x	
PREFIX	Binary	x		
REPORT	Binary	x		
SCAN	Binary	x		
STATDISP	Integer	x		
STATUS	Integer	x		x
TBL_RPT	Integer	x	x	
TIME_STP	Hexadecimal	x		
TRIGGER	Binary	x		
TROUBLE	Binary	x		x
VALUE	Binary	x		
ZONE_NO	Integer	x		

Appendix H: GPL Advanced Concepts

Metasys advanced concepts documents are created to help continuously improve existing Metasys skills.

Who Should Read this Document?

GPL programming satisfies life/safety applications and/or very complex HVAC systems. If the programs go beyond standard HVAC applications, programmers need to expand programming skills into full network integration. Programmers implement Priority schemes of processes and objects to ensure proper sequence of operations between standard HVAC applications and life/safety applications.

This appendix suggests ways to improve efficiency when executing in the NCM memory and help control traffic on the N1 and N2 networks.

The information provided here should increase GPL programming performances by:

- reducing programming time
- reducing traffic on the N1 and N2 busses
- reducing the time it takes to execute a GPL process in the NCM
- maximizing the effectiveness of AD and BD objects
- maximizing the use of the NCM Memory

GPL Efficiency Issues

Consider the how to address the following three GPL efficiency issues:

- a single GPL process
- all the GPL processes in an NCM
- GPL in the network

Efficiency of a Single GPL Process

There are trade-offs to efficiency with each application having a different set of criteria. The most important criteria may be saving memory in the NCM, faster execution time, or response time of certain processes.

The NCM execution queue cannot begin until the current process is finished. Only one process is executed at a time and if a lower Priority process is executing when a Priority 1 process is triggered, the lower Priority process must complete before the Priority 1 process begins. The following sections provide suggestions for improving single GPL process efficiency.

Use Binary Change-of-State (COS) Triggers Instead of Periods

If all the attributes read by a process are triggerable, do not use a period. Any change to an attribute causes the process to execute and recalculate any changes to the control, which wastes processor time.

All period timers are supported by the operating system in the NCM. Every second the NCM's operating system must check the period timers to see if they have expired. Although the operating system is extremely efficient, over time (years) the wasted processor time from a single unnecessary period timer can add up.

If some attributes are analog and do not trigger, a period may not be necessary. If the analog value is only used when a binary trigger changes, then a period is unnecessary. If an analog value is only necessary when a binary is in a certain state (for example, room temperature only considered if in occupied mode) then the process should be programmed to conditionally set the period to a non-zero time when the binary is in a certain state (occupied mode). Conditional periods can be used to vary the period.

Example

A process may need to execute once a minute in occupied mode, but only once every 10 minutes in unoccupied mode. If the period is exceptionally long (12-24 hours) weekly schedule the process to run once or twice a day. Scheduling a process to execute at a time when there is less demand on the NCM can be a good alternative to a period. Weekly schedule a process that needs to execute once a day to execute when a building is unoccupied.

Eliminate Triggers Unnecessary for Process Execution

If a process runs periodically because of analog data, consider eliminating all or most of the triggers.

Examples

If a process is monitoring room temperature every 5 minutes and when room temperature is outside the comfort level, the process checks the status of certain equipment to determine if additional equipment should be turned on or off. The status attributes of the additional equipment should not trigger the process, their change of value only cause unnecessary process execution. Therefore, exempt all the triggers to the process.

If a process needs to open a damper when two fans are on and the fans are sequenced by another process such that when Fan 1 goes on Fan 2 is commanded on 10 minutes later, there is no need to trigger the process until Fan 2 is on. Therefore, exempt Fan 1 as a trigger. If Fan 1 triggered the process before Fan 2 is on, the damper would not be opened anyway, and the processor time used to execute the process would be wasted.

Set the Period to a Reasonable Value

Set the process period for as long as possible.

Example

A process that needs to react to a change in outdoor air temperature probably doesn't need to execute much faster than once every 15 minutes.

A process that only needs to run once every 15 minutes with a period of 5 minutes is executing three times as often as necessary. That makes it three times as likely to be executing when a critical process needs to execute. That means it uses up three times the NCM processor time.

Reduce the Number of Attribute Reads a Process Must Execute

For every data line out of an object block or a reference block an attribute read is done during the execution of the process.

Each attribute causes the process execution to pause. The length of the pause varies based on many factors including:

- the amount of available acquired memory
- where the data must be retrieved from (e.g., across N1, from N2 device, and traffic on either/both)
- number of messages queued up at object manager
- other activity on the NCM
- if the object does not respond within 1 minute, the process will use the last known reliable value for the attribute, flagged unreliable, and continue execution.

Note: If an NC's N2 Bus is very busy, this message may return after the time-out. This will require additional processor time by the JCB-Interpreter, put an error message in the NC error log, and may affect the results of another attribute read.

There are a few rare cases where an attribute must be reread by a single execution of a process (after a command to the object or after a Wait block). For most applications, if an attribute value is needed more than once in a process, then read it once, store in a Value Holder, and fan-out of the Value Holder where needed. The Value Holder read is faster than the attribute read is to execute.

Use the Value Holder (VH) Block Instead of the Shared Variable (SVAR) Block

In general, a Value Holder block should be used when the data it contains is used only inside a single process. A shared variable is 6 bytes bigger in size and 2 bytes bigger per reference than a Value Holder. Each reference to a shared variable takes a few nanoseconds longer than a Value Holder reference. In most cases 2 bytes and a few nanoseconds are insignificant, but it is worth considering getting into good programming habits. Do not redo a GPL process in a well running NCM to use Value Holder blocks in place of shared variables.

Reduce the Number of Commands a Process Executes

For every data line out of a command block, a command is sent to an object when the line is executed. If the command block's enable input is true or there is no enable input, a command will be sent every execution of the process. Similarly if a 2CMD block has no edge trigger and the enable in is true (or there is no enable input), then a command will be sent every execution of the process. The amount of time it takes a process to execute the command depends on several factors. However, it takes longer to create and send a command than it does to process an edge trigger on a 2CMD or a one shot pulse into the enable input of a command.

Every command sent requires the following:

- memory acquired for the command message
- object receiving the command must process it
- message may have to travel across the N1
- message may cause N2 command (although in most cases a repeated command will not cause N2 traffic)--with the SYS91 and DX91xx N2 devices, it may now be more common for the command to be repeatedly sent. When a command is received by the NC object manager, it goes out on the N2 and gets the bit state. If the bit does change then the command goes out on the N2; if it doesn't change, there is "no" additional N2 traffic. So every command to the DX is at least one and maybe two messages on the N2.

There are a few, rare, cases where a command must be re-sent for each execution of a process even though the command does not change. In general, if a command is sent once, it does not need to be sent again until the command itself changes. When a command must be resent, determines if the process is in competition with another process, or scheduled commands, for control of the object.

Remove Fixed Commands from the Process into an MCO (Multiple Command Object)

If a process sends a lot of commands, consider using an MCO to issue the commands. The MCO provides automatic edge triggers.

Positioning a single GPL process in the NCM with the majority of the inputs (attribute reads) and MCOs in the NCMs with the objects to be commanded reduces N1 traffic from commands to a single command per MCO. In fire applications, sending out one command to an MCO in each NC provides faster response (damper commands) than a single process commanding objects in all the NCMs.

Example

For damper control, a process can perform the logic of which state the dampers should be in and send a single command to an MCO to issue the commands to all the dampers. For a fire application, an MCO in each NC would be more efficient than a single GPL process.

If a network has 5 NCs and each NC has 8 dampers to command in the event of a fire a single fire process would have to send out 40 commands to all the dampers (32 across the N1). With 5 MCOs, the fire process would only be sending 5 commands (4 across the N1), the MCOs would be commanding their subordinates (dampers) in parallel and all commands would be issued quicker. The GPL process would also execute faster allowing other processes to execute in the NCM.

Note: Don’t use the MCO in a GPL process when there is only one or two subordinates to be commanded because there may be more commands to the subordinate objects than if you would use standard GPL blocks and command the objects straight from GPL.

Use an MCO to trigger processes. If you want to order the execution of processes as a result of the mode change, an MCO could do that with subordinate order or delay times and you could create more finely tuned process prioritization.

The MCO provides edge triggering of commands because it only sends out the commands when the MCO changes state, therefore if a process sends the same command (same state) to the MCO every execution the MCO gets the command and remains at its state, if the state changes the MCO immediately starts sending out its commands for the new state.

Note: The MCO can send commands at the same time the process is commanding other objects or ending execution or another process is starting execution. This could cause many reads on the N2 or N1 LAN.

Efficiency of All GPL Processes in an NCM

The following sections provide ways processes in an NCM can be designed to work together to make an NCM more efficient.

Reduce the Number of Attribute Reads the Processes Execute

In GPL, for every data line out of an object block or a reference block, an attribute read is performed. Five data lines out of a single object block into five different processes means five processes reading the same value. Examine the application, if the attribute value does not need to be absolutely current, choose one of the processes (periodic for an analog value or triggered by the attribute for a binary or integer value) to be the only process that actually reads the attribute. The chosen process should write to a shared variable that all the other processes use to get the attribute value.

Reading an attribute from only one process in an NCM does all of the following:

- reduces execution time of all processes using a shared variable instead of doing read of the attribute
- reduces N2 traffic on NCM when the attribute must be read from N2 device
- reduces N1 traffic to and from NCM if object is on different NCM
- increases total NCM process memory usage by approximately 34 bytes

If several processes read the same attributes and do the same calculation on the values, the process designated to do the reads should do the common calculations and share the results in SVARs. This reduces execution time of the other processes.

Break Large Processes Down into Smaller Processes

Any event which causes a process to be executed places the process on the queue (e.g., change of a triggerable non-exempt attribute or shared variable; wait, period, delay or pulse timer expiration; manual, scheduled, or programmed trigger command, etc.). The length of time the process spends on the queue is based on three things:

- process's Priority
- processes of higher Priority in the queue
- execution time of the process currently executing

Breaking up large processes into smaller processes reduces the time a process of higher Priority waits. If a large process of Priority 3 is broken into three processes of Priority 3, any one of the smaller processes executes faster than the original. A Priority 1 process, which is queued during the execution of one of the smaller processes, waits less time to execute.

Smaller processes are easier to understand and troubleshoot. However, the trade-off is memory; each process has a fixed amount of overhead.

When breaking apart large processes, look at triggers and periods. If a process has both trigger and periods, see if the process can be broken apart so that the smaller processes have either triggers **or** periods. This reduces the executions of the smaller processes.

Combine Small Processes Into Large Processes

Note: This is the exact opposite of the efficiency described above. Larger processes are more difficult to understand, to troubleshoot, and require more acquired memory to upload and download.

Larger process are harder to debug and understand. The single large process has a longer execution time during which a higher Priority process must wait on the queue.

Reasons to combine processes would be:

- if they were performing duplicate reads and calculations
- if the NCM has reached the 255 process limit
- If you decide to combine small processes into larger processes consider the following when doing so:
  - combine processes with the same period
  - combine processes with the same inputs

Efficiency of the Entire Network

The following is a list of ways processes can be designed to improve the entire FMS Network.

Select Either AD or BD in a Process Input or Value

Using both AD and BD in a process input or value is a waste of resources (memory and processor time). Do not define an AD or BD with an associated input and using a process to command its value (SET_BD or SET_AD) at Priority 3. Every 30 seconds an AD polls its associated input and resets its Value. Every 4 seconds (may vary on a very busy system), a BD polls its (non-triggerable) associated input attribute and resets its Value. Thus, if a process changed an AD's value with a Priority 3 SET_AD command, the value sent by the process will be lost in less than 30 seconds. If an AD or BD is commanded at Priority 1 or 2, the value of the AD or BD will be the highest Priority commanded value. If an AD or BD with an associated input is commanded at Priority 1 or 2, the AD or BD continues to poll for its associated input.

If the Associated Inputs for BDs use triggerable attributes, then the BD will change its value when the Associated Input triggers. The Key here is if the input is triggerable. Most CS object binary attributes do not trigger (as well as other non-triggerable attributes), therefore they will be polled at the 4-second rate.

Proper NCM Placement of a Process can Reduce N1 Traffic

An attribute read from a process to an object on a different NCM requires two messages across the N1 (one to request the data, one to respond to the request). A process output to an object (command or write) on a different NCM requires one message across the N1. If a process inputs and outputs to multiple NCMs total up the messages in and out of each NCM and place the process in the NCM, that minimizes the N1 messages.

Read the AD or BD Value Directly

If a process that reads a specific attribute resides on the same NCM as an AD or BD associated to that attribute, read the value from the AD or BD instead of from the object. The reasons include:

- ADs and BDs never poll the N2 in response to a read attribute request
- reading an attribute from an object on the same NCM eliminates two messages across the N1

Replace the Associated Input of an AD or BD with a Process

Writing a process to replace the associated input of an AD or BD can reduce N1 and/or N2 traffic.

The following describes how an AD or BD to get its associated input.

Note: A BD with an associated input initiates this every 4 seconds. An AD with an associated input initiates this every 30 seconds.

1. A message is sent to the object for the attribute's value:

   The requesting task (AD or BD manager) acquires memory for the message, fills in the message with the appropriate information, and sends the message to the object.

   If the object is on a different NCM, the message is sent across the N1 (via network manager). The memory acquired for the message is released. When the message arrives at the proper NCM, that NCM's network manager acquires memory for the message.

2. Waits for a return message from the object or 1 minute, whichever occurs first.

   - If the information requires an N2 read the object manager passes the message to the N2 poll task. The N2 poll task polls the N2 device for the information, the N2 device responds to the poll.
   - The object manager (or the N2 poll task if the message was passed on) usually uses the request message to respond putting all requested attribute value information in the message. Once the response message is filled in the object manager (or the N2 poll tasks), it returns the message to the requesting task.

- If the object is on a different NCM, the message is sent back across the N1 (via network manager). The memory acquired for the message is released. When the message arrives at the requesting NCM, that NCM's network manager acquires memory for the message. And then sends the message back to the requesting task.

3. If a message was returned it assumes the returned value as its Priority 3 value and releases the memory acquired for the message. If the message "times-out" (no return after 1 minute) it maintains the last Priority 3 value it had.

   In the case of an AD, a simple process written to read the attribute from the object and command the AD to the value could greatly reduce N1 traffic if the period of the process is set to a value greater than 30 seconds.

Example

Assume an AD is associated to an AI whose value averages a 1% change every 10 minutes. The AD sends a message every 30 seconds requesting the AI's value and the AI returns the message, or two request messages per minute and two returned messages per minute. A simple process that reads the AI and commands the AD to the value read, with a period of 10 minutes in place of the associated input, can greatly reduce the messages sent. The process will request the attribute value once every 10 minutes the object will return the attribute value once every 10 minutes, and the process will command the AD once every 10 minutes.

A total of three messages every 10 minutes, depending on the NCM placement of the process one, two, or three messages across the N1. Depending on the attribute that is associated with the AD, this could also reduce N2 traffic.

There are drawbacks to this approach:

- Memory. The simple process takes up considerably more memory than the associated input.
- Complexity. When viewing the AD on a Focus window, it is no longer obvious where the value is coming from extra engineering time to create the processes.
- This adds another process, which needs to be executed.

- When a process reads an unreliable value, it uses the last known reliable value for the attribute, flagged unreliable. So, the AD would be commanded to the last known reliable value of the attribute, not the attribute's unreliable value. The use of the AD should be considered when contemplating this approach.

Reducing N1 traffic can increase the efficiency of other NCMs as well. Consider how many messages the NCM with outdoor air temperature must handle in a system with many NCMs all needing outdoor air temperature. Every N1 message eliminated is one less message that needs to be acquired.

4. Writing a process to replace the associated input of multiple ADs or BDs can reduce N1 and/or N2 traffic. Writing a single process that reads an attribute and sends (set commands) values to multiple ADs or BDs can significantly reduce N1 and N2 traffic. See the earlier outdoor air temperature example.

   The drawbacks to this approach are the same as Step 3 above.

   Using an AD or BD in the same NC could eliminate GPL code. An AD or BD can be assigned an initial value; the initial value could eliminate special code written to determine if an NC is responding. If the GPL process does an attribute read and never gets a response, it uses 0 or false as the default. Using an Initial value can provide a better value to use in case NC to NC communication is unavailable.

GPL Libraries

HLIB - GPL HVAC Library

If you need information about a compound in the GPL HVAC library refer to the appropriate application notes in the *Metasys Network Technical Manual (FAN 636)*.

The same information in the text of the application notes is available through the Description icon. Highlight the compound name and click on the Description icon. Page Up and Page Down keys move you through the text.

The text for the example files is available in the same way.

Recommendations

We recommend when you modify or add any compound or example files to the GPL HVAC library, you also modify or add the corresponding descriptor file. It may be helpful to make additions to the GPL HVAC library or create a new library to create a standard for applications that are unique.

Using the Metasys GPL HVAC Library

The easiest way to build a GPL library is that every time you make a compound, save it to the disk instead of the screen. Then load the compound into your strategy file when you need it.

Building Your Own Library of GPL Compounds

When you make a process, keep the template a Group compound. Give it a name that helps describe the compound and save it to the disk. Then when you need it, load it into your strategy file and change it to a process compound.

Don’t forget to share your library compounds with others in the company.

Prioritization of Processes

In order to understand process prioritization, some background information on the process queues is needed. There are five process queues in the NCM, one queue for each of the four priorities and a fifth for TIME SLICED processes (see the *Process Statuses* section for an explanation of Time Sliced processes). A queue is nothing more than list. In this case the lists are of processes requesting execution. With two exceptions, whenever an event occurs that requires a process to be executed the process is placed at the bottom of its Priority queue. The two exceptions are:

- when the process is already on its Priority queue or Time Sliced.
- when the process is in a state (Disabled, Error) that can not be executed.

If a trigger is received for a process on its Priority queue in a Wait state the process maintains its position on the queue, however, its Wait state is canceled.

Processes are removed from the queues one process at a time and executed.

The process that has been on the Priority 1 queue the longest (i.e., the one on the top of the list) executes next. If there is no process on the Priority 1 queue then the one that has been on the Priority 2 queue the longest executes. If there is no process on the Priority 2 queue then the one that has been on the Priority 3 queue the longest executes and the same is true for Priority 4. The last to execute is any process that has been Time Sliced.

The order in which processes are removed from the queues explains the effects of process prioritization. In a NCM with processes of all priorities, a Priority 2 process spends less time on the queue between the event that caused it to be queued and its execution than an identical process with a Priority of 3. Process Priority should be assigned based on the application. The most critical and ideally the least frequent processes should have the highest Priority.

Process Priority cannot guarantee that a critical process will execute immediately or at a constant period. When any event causes a process to be queued the only way it will be executed immediately is if there is no process currently executing. Only one process at a time is executed.

Each process must finish execution before another process can begin execution. Once a process has started, even if it is at Priority 4, it will continue until it finishes. This is true even if a Priority 1 process was added to the queue immediately after the Priority 4 started. Finish execution means all blocks executed, or a STOP, ABRT, or Wait block is executed, or the process is ERROR or Time Sliced, or the process is commanded DISABLED. Therefore, if there is a process that absolutely, positively must execute immediately, then it absolutely, positively must be the only ENABLED process (other than the RESTART process) in the NCM.

Types of Memory

There are two types of memory in the NCM: allocated and acquired. Allocated memory is used for long-term or permanent storage. This includes the NCM programs themselves, processes, databases, etc. Acquired memory on the other hand is used for short term storage such a sending messages between tasks in the NCM, sending or receiving messages on the N1 or N2, sending commands to objects, etc. During process execution a process may acquire memory. When a process acquires memory it is for the purpose of communication. The communication is to obtain information or send information. The communication may be to objects and features in the same NCM or a different NCM, it may be to a printer or OWS. The following are all the times a process will acquire memory:

- to read an attribute value (data line out of object block or REF block)
- to read a totalization value (TOT block)
- to time a period or wait (Wait blocks, STOP blocks or all blocks executed)
- to send an advisory message (ADV block)
- to send a print message (PRNT block)
- to time a pulse or delay (PULS, DLAY, and BSEQ blocks)
- to send a command (CMD or 2CMD blocks)
- to write an attribute value (WRIT block)
- to trigger for a binary shared variable change (trigger other processes)

When a process acquires memory, it affects the NCM that the process is executing in, but it may also affect other NCMs on the network. The following may cause memory to be acquired in another NCM when the object read/commanded or the printer written to is associated with another NCM:

- to read an attribute value
- to read a totalization value
- to send an advisory message
- to send a print message
- to send a command
- to write an attribute value (WRIT block)

A process has only one of the following acquired at a time:

- to read an attribute value
- to read a totalization value
- to time a period or wait

A process acquires the following and may have many of these acquired at a time:

- to send an advisory message
- to send a print message
- to time a pulse or delay (PULS, DLAY and BSEQ blocks)
- to send a command (CMD and 2CMD blocks)
- to write an attribute value (WRIT block)
- to trigger for a binary shared variable change (this does not affect other NCMs)

The difference between the two categories listed above is the *one at a time* category issues a message and does not proceed until the message is returned. The *many at a time* category issues a message and continues with the potential to immediately issue another.

Example

A *one at a time* read attribute value sends the message and pauses execution until the message is returned; during the pauses, it is not executing, it is not asking for any other messages. An example of the *many at a time* is a command sent to an object with process execution continuing, the process may immediately send another command. Several commands may be sent by a single process before the first commanded object has processed the command.

Note: An NCM has a single processor and each piece (object/feature) must share the use of the processor, thus an executing process may send many messages before an object gets its turn.

If an NCM is running out of (or running low on) acquired memory, the following blocks in a GPL process may be contributing to the problem:

Note: Processes are not the only users of acquired memory, however, this document deals strictly with processes.

- ADV (advisory) block
- PRNT (print) block
- PULS (pulse) block (except one-shot)
- DLAY (delay) block (except one-shot)
- BSEQ (binary sequencer) block
- CMD block
- 2CMD block
- WRIT block
- binary SVAR blocks with any non-exempted output connections

Advisory and print messages are particularly large and a printer is a slow device. A process was not designed to be a report generator. Use print messages with extreme care. The timer messages (period, wait, pulse, delay, and binary sequencer) stick around for the length of the timer, or until they are canceled. The other messages only exist until they are processed by their receiver. In the case if Print Messages, they may accumulate in the NCM that has the printer.

Shared Variables

A shared variable is a data storage location in a NCM. A shared variable's purpose is to give a name to a specific data location so the data in the data storage location can be shared between processes on the same NCM. In GPL there are two ways to create this data storage location:

- a SVAR block
- by drawing a data connection line that connects two operation blocks across two different processes. In JC-BASIC this data storage location is created by declaring a variable SHARED.

In GPL or in JC-BASIC a shared variable should be used when data (usually calculated) in one process needs to be used by another process.

A binary shared variable that is not exempted in a process that reads its value triggers the process each time the value changes. Therefore, when using a binary SVAR block to reduce attribute reads exempt the shared variable in the process that reads the attribute and assigns the attribute's value to the SVAR.

Using Shared Variables

The following is a list of ways a shared variable can be used to increase process and NCM efficiency. If the data does not need to be shared with another process, use a Value Holder block instead of the Shared Variable block for greatest efficiency. In JC-BASIC, if the data does not need to be shared with another process use a local variable.

1. Instead of doing a fan-out from an object or REF block, drag a single line out to an SVAR block and do the fan-out from the SVAR block to reduce N1 traffic and/or N2 traffic.

   This does a single read of the attribute value, at most one poll of the N2, and at most one read and reply on the N1. By using the SVAR block the attribute value does not change during the process, every comparison uses the same value of the attribute.

Example

If a process wants to start a fan with the temperature between 60 and 80 degrees, there are two compare blocks both needing the temperature value. If a shared variable is used to store the temperature, both compare blocks will get the identical temperature value; if both compares have a line direct to the object block, two reads of the object (probably to the N2 device) will occur, and chances are the value may vary slightly between reads. In this example, a few tenths or hundredths of a degree difference may not matter; however, you may have an application where it does matter. Also, if the attribute value is binary, the change in value could make a dramatic difference. The first half of a process could execute as if the BI were open and the second half could execute as if the BI were closed.

2. A shared variable takes up less NCM memory than an AD or BD object. Therefore, if a calculated value does not need to be seen by the customer or used by another NCM’s process but does need to be used by multiple processes on a single NCM, put it in a shared variable to save memory. The shared variable will have an added bonus; in this case, it is faster to access a shared variable than to access an AD or BD attribute.

3. Use a shared variable as described above to decrease process execution time and reduce N1/N2 traffic. Each attribute read involves a pause of up to 1 minute while message passing occurs to retrieve the data. Reducing the number of reads (even on the same NCM) decreases process execution time.

Value Holder Block vs. Shared Variable Block

If the data does not need to be shared with another process, use a Value Holder block instead of the Shared Variable block for greatest efficiency. In JC-BASIC if the data does not need to be shared with another process use a local variable.

Some facts that may help you make wise choices concerning when to use a Value Holder block (local variable in the NCM) and when to use a Shared Variable block (shared variable in the NCM) follow:

1. In the NCM, a local variable and a shared variable do not use the same amount of memory to store the variable data (name, value, reliability, array information). However, a shared variable must store extra information to provide its extra capabilities. Each shared variable has 6 bytes more of information per variable than a local variable (2 bytes for location information, 4 bytes that signify whether the variable is a trigger to any process).

   If the shared variable is binary and triggers any process, it uses an additional 32 bytes (regardless of how many processes it triggers) to store the information on which processes to trigger. The shared variable has a small amount of overhead that a local variable does not, the overhead averages 1/4 byte per shared variable.

2. Each NCM process object contains references to shared and local variables. The reference to a shared variable is a long pointer, the reference to a local variable is a near pointer. The ramifications this has to you is a long pointer is 4 bytes, a near pointer is 2 bytes. A long pointer takes a few nanoseconds longer to execute than a short pointer. In most cases, 2 bytes and a few nanoseconds are insignificant, but it is worth considering getting into good programming habits. There is a variable reference in the NCM process object every time the variable value is read or written, in GPL this is every line in or out of a SVAR or VH block. In JC-BASIC, this is every time a local or shared variable name is used.

3. If NCM memory is extremely full shared variables can be used instead of local variables to save memory. Instead of having a VH block in every process, reusing a SVAR could save memory. The memory saved is the memory each local variable would take up vs. the overhead of a shared variable. There are some important considerations:

   a. Space **must** be more important than speed.

   b. Space **must** be more important than program readability.

   c. The VH block **must** be assigned a new value before it is used elsewhere for every process substituting the common SVAR. Another way of saying this is that the value assigned to the Value Holder now will not be used for the next execution of the process, the value is reassigned (either by a calculation or attribute read).

   d. There are enough processes in the NCM to justify this substitution (if the type is binary all processes must **exempt** it as a trigger) the VH block has less than 12 connections (12 or more connections no longer saves memory because the process object code required for an access of a shared variable is 2 bytes per access larger than the process object code for VH’s local variable) **extreme caution** should be used if making this substitution, and this is not recommended as a good programming practice. Other options should be explored before attempting this.

4. A shared variable can be used to decrease N1/N2 traffic and increase an NCM’s performance as follows:

   - Analyze all processes, which read the same attribute.

   - Determine the acceptable age of the attribute’s value that will provide proper operation. Acceptable age means how old can the value be, for example, since outdoor air usually doesn’t change very fast, an acceptable age may be 15-20 minutes.

- Choose a process accessing the attribute value whose period is less than or equal to the acceptable age. For the outdoor air example, if three processes use the variable, with periods of 50 seconds, 2 minutes and 5 minutes, then choose the 5 minute process since its period falls closest to the acceptable age.
- In the chosen process, read the attribute value and store in a shared variable in the chosen process and all other processes. In the outdoor air example (assuming outdoor air on another NCM), this reduces N1 traffic to read outdoor air from an average of 3.4 messages per minute to an average of 1 message per 2.5 minutes (a read is 2 messages, a request and a reply). In the outdoor air example (regardless of the NCM containing outdoor air) this reduces N2 traffic generated by these processes from 1.7 polls per minute to 0.4 polls per minute for outdoor air. In addition all processes not reading the attribute will reduce their execution time by the time that they previously spent creating the message, waiting for the response, and cleaning up after the message.

Process Statuses

The process compound object in the archive database is referred to in this document as a disk process. The process compound object in the online database of the NCM is referred to in this document as a Field process. The terms Disk and Field are taken from the compound’s System name process window.

A disk process has two states: enabled and disabled. A process in the disabled state (disk) is in the disabled state (field) when download is complete. A process in the enabled state (disk) downloads into the Ready state (field). A process in a single process downloads into the Held state.

A process has eight statuses that can appear in the field process focus window:

1. **Not fully downloaded**

   This status would appear in the focus window while a process is being downloaded. It will also appear when an error occurred during download and the process cannot be executed because of the error.

2. **Error**

   A process can be in the Error state for two reasons:

   - the process executed an abort (ABRT block in GPL or an ABORT statement in JC-BASIC)
   - an unrecoverable error, usually floating point, occurred. Whenever an unrecoverable error places the process in the Error state an advisory is generated by the process.

   A process in the Error state does not execute, as long as the process is in the Error state. No trigger will cause a process in the Error state to execute. A process can be changed out of the Error state, by an enable command (manual or programmed),or by downloading a process of the same name (system\object name).

3. **Disabled**

   A process can be in the Disabled state for two reasons:

   - the process received a disable command (manual or programmed)
   - the process was downloaded in the disabled state

   A process in the Disabled state will not execute, as long as the process is in the Disabled state. No trigger will cause a process in the Disabled state to execute. A process can be changed out of the Disabled state by an enable command (manual or programmed), or by downloading a process of the same name in the enabled state (disk).

4. **Held**

   A process can be held for four reasons:

   - A process is placed in the Held state when a full download of the NCM has completed, the process was downloaded in the enabled state (disk), and a restart process was downloaded.
   - A process is placed in the Held state when the NCM it resides in resets, the process was in the Ready, Executing, Waiting, Time Sliced, or Held state when the NCM reset and a restart process exists in the NCM.

- A restart process cannot be placed in the Held state. If the NCM does not have a Restart process, no processes in the NCM can be in the Held state.
- When a restart process exists in the NCM, no other process will execute until the restart process has successfully completed. The restart process completes successfully when it has executed a stop (STOP block in GPL or stop statement in JC-BASIC) or has reached the end of the process (all blocks executed in GPL, end process statement in JC-BASIC). If the restart process is in the disabled or not fully downloaded state when a full download completes or if the restart process is in the disabled, error, or not fully downloaded state when the NCM resets no other process executes. All processes placed in the Held state remain in the Held state. If the restart process finishes unsuccessfully and is placed in the Error state (see above), all processes placed in the Held state remain in the Held state.

5. **Ready**

A process in the Ready state is available for execution when triggered. The only process in the NCM in the Ready state that cannot be triggered is the restart process. A process is placed in the Ready state when:

- the process finishes execution by executing a stop (GPL STOP block or JC-BASIC stop statement) or by executing a end process (all GPL blocks executed or JC-BASIC end process)
- the process is disabled and receives an enable command (manual or programmed)
- the process is downloaded (single process download) successfully in the enabled state (disk). The restart process has finished if a restart process exists on the NCM.
- the process is in the Waiting state and the process is triggered

6. **Executing**

   The process is in the Executing state when the process is being executed. The Interpreter is the task in the NCM that executes a process. When the process is removed from the queue for execution the interpreter changes the status to executing. A process must be in the Ready, Waiting, or Time Sliced state to be on the queue. When the process is done executing it is placed in the Ready or Waiting state if the process executed normally. The process is placed in the Error state if the process executed abnormally. The process is placed in the Time Sliced state if the process is in an infinite loop or too large (see below).

   The process is placed in the Disabled state if a disable command (manual or programmed) is received for the process, in this case where a process is executing, when a disable is received the process, halts wherever it is, in the code. Care should be taken when disabling a process that does two or more commands that if the first command occurs the second command must be sent. For example, if the pump is started the damper must be commanded open, consider that if the disabled command is received after the pump is started but before the camper is commanded the damper command will not be issued.

   Only one process per NCM can be in the Executing state at a time.

7. **Waiting**

   The process is placed in the Waiting state when the process executes a wait (GPL Wait block or JC-BASIC wait statement). When the wait timer expires, the process is put on its Priority queue. The process remains in the Waiting state until:

   - the wait timer has expired and the process is removed from the queue for execution, at which time the process state is changed to Executing.
   - a trigger is received for the process. The process state is changed to Ready. If the wait timer has not expired, the wait timer is canceled and the process is placed on its Priority queue. If the wait timer has expired, the process stays on its Priority queue where it had been placed by the wait timer expiration.

Any of the following can trigger the process:

- non-exempt triggerable attribute read by the process
- manual trigger command
- programmed trigger command
- scheduled trigger command
- binary shared variable read by the process
- expiration of the pulse timer (BSEQ, PULS, and DLAY blocks or JC-BASIC canc_pulse and n_canc_pulse functions)

The following removes the process from the Wait state:

- the process receives a disable command (manual or triggered) and the process is changed to the Disabled state. If the wait timer has not expired, the timer is canceled, or if the wait timer has expired, the process is removed from the queue.

8. Time Sliced

Each time a process's status is set to Executing, an op-code counter is set to 32767. For every op-code executed in the process, the counter is decremented. If the counter reaches zero, the process is Time Sliced.

When a process is GPL translated, it is converted to JC-BASIC source code. When a process is compiled, it is converted from JC-BASIC source code to process op-codes that the NCM's Interpreter task can execute. The number of op-codes compiled per GPL block or JC-BASIC statement varies greatly. The number 32768 is large enough that any reasonable application can execute without being Time Sliced. The maximum size of a process is 32 Kb; only a portion of the 32 Kb can be op-codes. If a process exceeds its op-code count there is a loop where it is executing some of the op-codes more than once. If a process executes more than 32768 op-codes, it may be in an infinite loop.

Once a process exceeds it 32768 op-codes its state is changed to Time Sliced and it is placed on the Time Sliced queue and an advisory message is issued. A process is removed from the Time Sliced queue and execution continues at the exact op-code it left off whenever there are no processes on the Priority queues.

Each time execution of a Time Sliced process is begun, the op-code counter is divided by 2 until the op-code counter reaches 500 or less (i.e., if the process is removed from the Time sliced queue for the first time it will start with a op-code counter of 16383, if the process is removed from the Time sliced queue for the fifth consecutive time the op-code counter will start at 1027). A process with a status of Time Sliced cannot be triggered.

When a Time Sliced process has been removed from the queue for execution its status is changed to Executing, except for the reduced op-code counter it is executed the same as any other process.

A disable command (manual or programmed) will change a Time Sliced process to the disabled state.

Process status is a readable attribute of a process object. The process status attribute is an integer attribute. The status attribute has a range of 1-8. Listed below is the integer value of the process status and the corresponding state the process is in:

Attribute Value	Process Status (field)
1	Not Fully Downloaded
2	Error
3	Disabled
4	Held
5	Ready
6	Executing
7	Waiting
8	Time Sliced

JC-BASIC allows reading any readable integer attribute. A GPL user’s block can be used to access the status attribute of any field process. The attribute name is STATUS.

When to Use the Restart Process

Some processes may go unreliable the first time they run after the NCM starts up because some of the field objects being read by the processes are initially offline/unreliable. This happens when the field objects have not reported to the NCM at the time the processes run, but eventually come online.

If you have this situation, then create a Restart process that includes a Wait block with a wait time of xx seconds. If a Restart process already exists, add the Wait block to it. This Wait block should be placed in the process so it executes before any objects are read. The amount of wait time needed will be job dependent but you may wish to start at 30 seconds and decrease the value if no problems occur.

How a Process Period Works

The process period timer is set when the execution of the process is complete and the period is not 00:00:00. The process period timer is not set if the process goes into a Wait state (a Wait block in GPL or a Wait statement in JC-BASIC). The process period timer is not set if the process is put into the Time Sliced or Error states. When the process period timer expires the process is placed on the queue and will be executed as soon as possible based on process Priority.

A fixed process period does not guarantee that the process will execute at that rate. For example, a process with a 5-minute period and no triggers will execute every 5+x minutes where x is variable and is the sum of the process execution time, the time spent on the queue, and any time spent Waiting. The process execution time varies based on the size and complexity of the process object code, the response time of attribute reads, and the other NCM activity while the process is executing. The time spent on the queue is based on the Priority of the process and the execution time of all processes ahead of it on the queue (either higher Priority or same Priority but queued first).

A process period timer is canceled whenever the process is triggered. Therefore, a process with a 5 minute period timer that has been timing for 2.5 minutes will be placed on the queue immediately if any trigger for the process is received. A process period need not be fixed. The period can be calculated. If a process has more than one Period statement (block) the last one executed determines the length of the process’s period to be started at the end of the current execution.

A process period of 00:00:00 does not start the period timer at the end of execution, therefore a process with a period of 00:00:00 will not execute again until it is triggered.

When set, the process period timer is set to the value last assigned to the period (process template’s period or PERD block in GPL, PERIOD statement in JC-BASIC). The timer has a range of 1 second to 23 hours-59 minutes-59 seconds. When the period timer expires, the process is placed on a queue and executes in order based on process Priority and time on the queue.

The process period timer is canceled and the process is immediately queued if any event occurs that requires a process to be executed (change of a triggerable, non-exempted attribute or shared variable; delay or pulse timer expires; manual, scheduled, or programmed trigger command). The process period timer is canceled and the process is not queued for execution if the process is Disabled.

Frequently Asked Questions

Question

If a process is executing and encounters a wait or delay block, say 10 minutes, will other processes waiting to be executed go ahead and run during the 10-minute delay or will they set until the first process has fully executed?

Answer

The wait and delay blocks have different actions. The Wait block when executed immediately stops the current process execution (as long as wait is not 00:00:00), sets the process status to Waiting and lets the other processes on the queue execute. A Wait block stops process execution immediately, means that all blocks whose execution follows the Wait block in the process will not execute until the Wait timer expires.

When the wait timer expires, execution of the process begins at the block immediately following the WAIT block. When a delay or pulse block timer expires, the execution of the process starts at the beginning of the process.

The delay and pulse blocks do not stop the execution of the process. When they set their timer, the process continues to execute the process. A process can have multiple delay and pulse timers active at the same time (one for each delay and pulse block). Expiration of a delay timer does not cancel the timer of other delay or pulse timers (expiration of a delay or pulse timer will cancel the WAIT timer).

Only one process can be executed at a time in the NCM. The process that is executing has a process status of EXECUTING. Whenever the Interpreter changes the process from EXECUTING to any other status, the Interpreter is immediately available to execute another process.

Question

Will a CS object (Reference Block) trigger a GPL process, or must you use a process timer?

Answer

Some attributes of a CS object are triggerable. (See the CS object attribute list in *Appendix G: Attributes* for triggerable attributes). When using the other CS object attributes, a process timer is necessary. If in doubt about the triggers of a GPL process, check the trigger table in the compile listing.

Question

Is last known value table cleared when NCM reboots?

Answer

First, let’s redefine reboot to be coldstart and warmstart.
A coldstart is hitting the reset button or performing a full download, and warmstart is after a powerfail where the battery has maintained memory.

On a coldstart, the last known value table is built by the process object manager as processes are downloaded. As the last known value table is built the last known values are set to 0.0 for real, 0 for integer, 00:00:00 for time, and False for Boolean.

On a warmstart, the last known value table is maintained.
The last known values are not changed.

Likewise, shared variables are initialized on a coldstart as they are downloaded, and shared variable values are maintained on a warmstart. In contrast, Value Holders and other local variables are initialized on both a coldstart and on a warmstart.

Question

What is the maximum number of commands an NCM can execute per minute?

Answer

There are 600 Commands per minute.

These commands come from the GPL processes executing as well as features that produce reads and writes to objects.

For the DX, every command requires three messages on the N2. For every DX command, the bits are read (one message to the DX to read the bits, one message back with the data); if the command changes the bits then a third message with the new settings is sent on the N2.

GPL Programmer's Manual

Glossary

ANA Option

Analog option that displays on the Connection menu. Clicking on it displays analog connections only.

Analog Display

Numeric field (0.00) that is on the Tools option menu. Clicking on it brings a numeric display to the screen, which can be associated with a block's output to indicate its value during simulation.

Archive Database

Database stored at an Operator Workstation for a particular Network Control Module. It contains process objects, and databases of hardware objects, point objects, and feature software.

Associated Input Timer

Timer that the Simulator uses to execute Analog Data and Binary Data blocks. You define the parameter by running the SYSGEN program. It has a value of 1 to 255 seconds (default is 30 seconds).

Attribute

Data element of an object.

BIN Option

Binary option that displays on the Connection menu. Clicking on it displays binary connections only.

BLK Option

Option that displays on the Remote Connection and Find option menus. Clicking on it displays the names of all named operation and special blocks.

Block Category

See *Function Block Category*.

Block Class

See *Function Block Class*.

Block Directory

See *Function Block Directory*.

Block Name A user-specified name for a function block or compound block.

Block Type Name A fixed name given to a function block that identifies it. For example, each Binary Input block is labeled with a “BI” block type name.

Block Window A section of the GPL Simulator that indicates various information about a function block, such as its name and output value.

CAE Translator A program that translates DDL+ files into GPL control strategy diagrams.

Calculation Blocks A category of function blocks. These blocks perform common control calculations; included are RAMP (Ramp), SPAN (Span), and FILT (Filter).

.CI extension The strategy file name extension for the file that contains connection information (.CI=Connection Information).

Click Left Pressing the left mouse button (two or three button mouse) once to perform an operation.

Click Right/Middle Pressing the right button (two button mouse) or middle button (three button mouse) once to perform an operation.

Clicking Pressing the mouse button to perform an operation.

CLR ALL Option GPL Editor field that is on the Erase option menu. Clicking on it clears the control strategy currently in memory. The display cannot be returned by the Undo function.

CMD0 Option Command option that displays on the connection menu for the 2CMD operation block. CMD0 is the command sent when the input is False (logical 0). When this option is highlighted, the menu shows all commands that can be sent when the input of the block is False.

CMD1 Option Command option that displays on the connection menu for the 2CMD operation block. CMD1 is the command sent when the input is True (logical 1). When this option is highlighted, the menu shows all commands that can be sent when the input of the block is True.

.CMP extension The strategy file name extension for the file that contains compound data (.CMP=Compound).

Compiler GPL utility that compiles intermediate source code into object code that an NCM can execute. The compiler also adds the code to the archive database.

Compound A user-created block that represents a grouping of many underlying function blocks. For example, a number of blocks can be interconnected and then grouped under a Heating compound.

Compound Icon GPL Editor icon that offers several Compound functions. These include: making, editing, loading, and deleting compounds, loading an expanded compound, loading a compound description file, changing disk drives, and changing compound directories. Clicking on this icon also shows the names of all GPL compound directories and file names.

Conditional Execution The execution of a block or process that occurs only when specified conditions are satisfied.

Configuration Attribute An attribute of an object that must be defined in the object's database template before the object can be added to the archive database. The attribute's value is stored in the archive database.

Configuration Connection A line drawn between two blocks that is not actually a data or control flow line, but a representation of an "association" made between the blocks on the database template of the destination block.

Connection A line drawn between two blocks on a GPL diagram to represent data or control flow. The types of connections include binary, analog, time, control flow, command, dual command, read, write, and configuration.

Connection Icon GPL Editor icon that enables you to draw data and control flow lines between function blocks. Double-clicking on this icon displays the Connection option menu.

Consultant Version Version of GPL that is specially designed for engineering consultants and educators. Includes GPL Editor, Expert Checker, and Simulator. Excludes archive database interface and translator/compiler.

Control Blocks A category of function blocks. These blocks perform analog or binary control functions; included are PIR (PI Reset), BSEQ (Binary Sequencer), COMP (Simple Compare), DFCM (Differential Compare), and DBCM (Deadband Compare).

Control Flow The order of block execution. The order is specified by the data and control flow lines.

Control Flow Connection A line drawn between two function blocks that helps to determine which block is to execute first.

Control Strategy A GPL file that is composed of compounds, function blocks, objects, connections, and text. The control strategy defines the software objects and defines the control logic for the Network Control Module, based on the sequence of operations. Static pressure control and dry bulb economizer are examples of control strategies. A strategy is actually stored in a set of three files, which have these DOS file name extension names: .TX, .DB, and .CI.

Control Strategy File Storage location of a GPL control strategy. The file is actually composed of three DOS files with these extensions: .TX, .DB, and .CI. The file name can be up to eight alphanumeric characters long.

Controllers A category of function blocks. These blocks represent various control functions or field controllers; included are PIDL (PID Loop), LCG (Lighting Control Group), 210A (C210A), 260A (C260A), and ZONE (Fire Zone).

Copy Group Icon GPL Editor icon that is on the Move option menu. Clicking on it lets you copy a group of blocks, connections, analog displays and text.

Copy Icon GPL Editor icon that lets you copy a single block, connection, analog display, or text.

Cursor Coordinates The x- and y-coordinates of the cursor’s current position. The coordinates are displayed on the bottom left side of the GPL Editor screen.

Database Manager (DBM) Software program that GPL uses when you add, change, or delete objects to or from the archive database.

Data Blocks A category of function blocks. These blocks store data; included are SVAR (Shared Variable), CNST (Constant), CONN (Connection), VH (Value Holder), AD (Analog Data), and BD (Binary Data)

Data Flow The passing of data from one function block to another. Data that can be passed between blocks includes analog and time values, and binary states.

Data Definition Language (DDL) A Metasys software language in ASCII text syntax used to create system databases.

.DB extension The strategy file name extension for the file that contains database information (.DB=Database).

DBM (Database Manager) Software program that GPL uses when you add, change, or delete objects to or from the archive database.

DDL (Data Definition Language) A Metasys software language in ASCII text syntax used to create system databases.

Defined Block A function or compound block that has all required template fields specified. Or, an object or process compound block that has been successfully added to the archive database. In the GPL Editor, the borders of a defined block are solid and colored brown.

Delete Icon GPL Editor icon that is on the File, Compound, and Print option menus. It lets you delete control strategy files, compounds, and items on the print queue.

Description Icon GPL Editor icon that displays a text description file associated with a control strategy or a compound.

Destination Block The block into which a connection line ends.

Destination Remote Connection A connection that references the destination block, in which the remote connection accepts an input from an origin block.

Diagram A single screen in the GPL Editor that may be composed of connected function blocks, text, and analog displays.

DIR Option GPL Editor field that is on the File and Compound option menus used for creating directories.

Direct Connection A continuous line between two function blocks that are contained in the same file. The line passes data, specifies control, or directs a command.

Double Click Quickly pressing a mouse button twice without moving the mouse.

Drag Moving the mouse while holding down one of its buttons, then releasing.

DWN Option GPL Editor field that is on the print queue and on the following menus: File, Compound, Connection, and Find. Clicking on it displays the next page of the queue or menu.

DRIV Option GPL Editor field that is on the File and Compound option menus. Clicking on it allows you to access a different disk drive.

EDIT Option GPL Editor field that is on the Compound option menu. Clicking on it brings the specified compound to the work area for editing.

Editor GPL utility that you use to create, modify, and delete control strategies. The editor is the central “hub” from which all other GPL utilities are available (i.e., Expert Checker, Simulator, and Translator).

Erase Icon GPL Editor icon that enables the erase and delete operations. Double-clicking on this icon displays the Erase option menu.

Exempt To prevent an object’s binary attribute or a binary shared variable from triggering a process.

Exempt Connection Icon GPL Editor icon that enables the exempt function, which lets you exempt an attribute of a particular object from triggering a process. An exempted connection is indicated by an open square over the arrowhead.

Exit Icon GPL Editor icon that lets you exit GPL to return to DOS or the FMS.

Expanded Compound The full contents of a compound; that is, all of its blocks, connections, analog displays, and text. The expanded compound can be pasted down on the screen by clicking left on the XPND option (located on Compound option menu).

Expert Checker GPL utility that checks a control strategy for completeness and correctness, and outputs any errors it finds to the list file.

Expert Checker Icon GPL Editor icon that is on the Tools option menu. Clicking on it runs the Expert Checker.

Fan In Multiple output connections fed into one input.

Fan Out One output connection fed into multiple inputs.

Field An area on the screen that contains or can hold information.

FILE Block A function block that represents and is a throughway to an external GPL control strategy. The block is pasted down on a diagram of a control strategy, and its contents can be accessed by double-clicking on it.

FILE Block Icon An arrow pointing up that displays in the upper right corner of the work area when you have accessed a FILE block. Clicking left on it returns you to the parent file.

File Icon GPL Editor icon that lets you perform directory and file functions. These include: saving, loading and deleting control strategies, changing disk drives, changing directories, displaying a strategy description file, and listing the directory. Clicking on this icon also shows the names of all GPL directories and files.

FIND Option GPL Editor field that is on the Query option menu. Clicking on it enables the Find feature. Find is similar to a Search feature on a word processor. It locates each place in a control strategy where a function block name is used.

Function Block Category A grouping of blocks that have some functions in common, such as mathematical equations or logic operations. GPL features 16 block categories.

Function Block Class A grouping of function blocks that have similar definition requirements. GPL features three block classes: object, operation, and special.

Function Block Directory The row of function blocks on the bottom of the GPL Editor screen. Clicking on a block selects it for pasting on the work area.

Function Blocks Rectangles that represent Metasys software objects, process objects, and operation or special blocks. There are three classes of function blocks: object, operation, and special.
A block is pasted down on the work area, and connection lines are drawn to it and from it to show data and control flow.

Global Database The central database that is common for all NCMs on the Metasys Network.

GPL See *Graphic Programming Language*.

GPL Consultant Version See *Consultant Version*.

GPL Editor See *Editor*.

GPL Expert Checker See *Expert Checker*.

GPL HVAC Application A defined, tested control strategy that is provided by the factory. Static pressure control and dry bulb economizer are examples of GPL HVAC applications.

GPL Simulator See *Simulator*.

GPL Translator See *Translator*.

Graphic Programming Language (GPL) A unique graphics-oriented language used for writing control logic for HVAC and lighting control equipment based on a sequence of operation.

Grid An invisible background grid on the work area of the GPL Editor. When the grid is on, blocks and connections “snap” into place, which helps you line up and orient these items on the screen.

Group Compound A functional grouping of blocks, connections, and text that is represented by a block.

Help Screens Brief descriptions and instructions that are available for any icon, option, function block, or screen position on the GPL Editor. Help screens are context sensitive and are displayed by pressing the F1 key.

Icon A graphic symbol that represents a function. You activate the function by clicking or double-clicking on the icon.

Input Connection Menu Menu that shows all available input connection names for a selected function block.

Input/Output Blocks A category of function blocks. These blocks represent input and output software and hardware objects; included are BI (Binary Input), BO (Binary Output), AI (Analog Input), AOS (Analog Output: Setpoint), AOD (Analog Output: Digital), ACM (Accumulator), and REF (Generic Object Reference).

Interfield Error An error that the GPL Editor finds when you try to save the changes made to a database template. The error involves the interaction between two fields. For example, if the high limit is lower than the low limit, the Editor will notify you with an error message.

Intermediate Source File File that the GPL Translator creates that contains textual statements of the translated GPL diagrams. To retain this file, you select the “Save Translated Source” option in the Translator submenu.

JC-BASIC (JCB) Johnson Controls BASIC language, a high-level, textual programming language developed to implement control strategies.

Library Field GPL Editor field that you click on to show all the function block categories. The categories display down the right side of the work area. Clicking inside the red checkbox next to a category changes the block directory to those blocks that belong to the category.

Line Segment A single section of a connection line.

List File A file in ASCII format that contains all errors that the Expert Checker, Session Read, Translator, and Compiler found. The error file is viewed by clicking on the VIEW option. Its name is the file name of the control strategy with a .LST extension.

LOAD Option GPL Editor field that is on the File and Compound option menus. Clicking on it loads a control strategy or compound block into the work area.

Logic Blocks A category of function blocks. These blocks perform logical operations; included are AND (And), OR (Or), XOR (Exclusive Or), NOT (Not), LTCH (Latch), PULS (Pulse), and DLAY (Delay).

Loop Two or more function blocks feeding input and output connections into each other. If a loop is formed, the GPL Editor cannot determine which block to execute first. All loops in GPL must be arbitrated (i.e., you must choose which block will execute first by designating the Loop-master block).

Loop-Master Block The first block to execute in a loop. A Loop-master block is indicated by a white circle at its input connection.

Macro File JC-BASIC source statements and connection names that the GPL Translator uses to translate operation blocks.

Main Window The upper part of the screen that is displayed when the GPL Simulator begins execution. The blocks and connections of the GPL diagram are visible in this window.

MAKE Option GPL Editor field that is on the Compound option menu. Clicking on it lets you create a compound. A white enclosing box comes to the screen with which you enclose the area of the diagram you wish to compound.

Math Blocks A category of function blocks. These blocks perform mathematical operations; included are ADD (Addition), SUB (Subtraction), MUL (Multiply), DIV (Division), AVG (Average), and EQN (Equation).

Memory Usage Figure Figure in percent that indicates how large the loaded file is in relation to its maximum allowable size. The figure appears in the lower left corner of the GPL Editor.

Miscellaneous Blocks A category of function blocks. These blocks perform various functions not related to the other blocks; included are TOT (Totalization), USER (User), and FILE (File).

Modifiable Fields Template fields that you can change by typing the desired value before or after the object has been added to the archive database.

Move Group Icon GPL Editor icon that is on the Move option menu. Clicking on it lets you move a group of blocks, connections, analog displays and text.

Multistate Blocks A category of function blocks that require special NCM software that is currently available in the European market only; included are MSD (Multistate Data), MSI (Multistate Input), MSO (Multistate Output).

NCM Process Object A process object that has been downloaded into and is being executed by the NCM.

Network Name field GPL Editor field that displays which network archive database is selected.

No Archive Mode Mode of the GPL Editor in which no archive database interaction occurs. Among other advantages, this mode allows you to create control strategies more efficiently.

Non-Modifiable Fields Template fields that cannot be changed once the object has been added to the archive database.

OBJ Option Object option that displays on the Remote Connection and Find menus. When clicked on, this option shows the names of all defined system names for the objects.

Object A software database record that corresponds to and characterizes a field device, a data point, an N2 address, or a control function.

Object Blocks A class of function blocks. These blocks represent software objects in the Network Control Module. They generate a database that the NCM uses.

Object Code See *Process Object Code*.

Object Control Blocks A category of function blocks. These blocks control objects; included are CMD (Command), 2CMD (Dual Command), READ (Read Attribute), and WRIT (Write Attribute).

.OCI extension The strategy file name extension for the file that contains previous connection information (.OCI=Old Connection Information). This is a backup file that can be used in case the original file is accidentally erased or unintentionally changed.

.OCM extension The strategy file name extension for the file that contains previous compound information (.OCM=Old Com-pound). This is a backup file that can be used in case the original file is accidentally erased or unintentionally changed.

.ODB extension The strategy file name extension for the file that contains previous database information (.ODB=Old Database). This is a backup file that can be used in case the original file is accidentally erased or unintentionally changed.

One-Shot Execution Single execution of a process.

Online Help Screens

See *Help Screens*.

Operation Blocks

A class of function blocks. These blocks instruct the Network Control Module to perform an operation, such as selecting between two values, executing some logic or calculation, or issuing an advisory. They also provide process control functions, such as wait and stop. GPL translates operation blocks into code.

Option Menu

Menu that provides additional functions. The option menu is displayed by clicking or double-clicking left on an icon or option.

Optional Connection

A connection line that is not required by the block’s algorithm and may have an equivalent template field for its value.

Origin Block

A block from which a connection line exits.

Origin Remote Connection

A connection that references the origin block, in which the remote connection enters a destination block.

.OTX extension

The strategy file name extension for the file that contains previous text information (.OTX=Old Text). This is a backup file that can be used in case the original file is accidentally erased or unintentionally changed.

Output Connection Menu

Menu that shows all available output connection names for a selected function block.

Pan Icon

GPL Editor icon that lets you move across, up, and down a zoomed diagram.

Parameter

A value in the database template that can be viewed and modified.

Pasting Down

Procedure of placing down an item on the work area.

Place Holder Compound An empty compound that you intend to fill in later. Used in a top-down system design.

Print Icon GPL Editor icon that lets you perform printing functions. These include: print one or more diagrams and templates, start printing process, and delete item from print queue.

Print Queue Listing in the Print option menu of diagrams and templates that are ready to be sent to the printer.

Print Submenu Menu that allows you to create print files for diagrams or templates.

Process A self-sufficient, modular block of computer instruction for an NCM. Processes are used to implement control strategies.

Process Compound A grouping of blocks, connections, and text that generates a process.

Process Control Blocks A category of function blocks. These blocks control or affect the execution of processes; included are PERD (Period), WAIT (Wait), STOP (Stop), and ABRT (Abort).

Process Object A control object that can be downloaded to and executed by the NCM.

Process Object Code A downloadable process object that has been compiled from the intermediate source file.

Protected Compound A protected grouping of blocks, connections, and text. You cannot edit or view the diagrams under this type of compound, because they contain proprietary information.

PRT MANY Option GPL Editor field that is on the Print option menu. Clicking on it displays the Print submenu with which you select whether to print multiple diagrams or templates. The function then prints to the queue the currently displayed diagram or templates and all of its lower level diagrams or templates.

PRT ONE Option GPL Editor field that is on the Print option menu. Clicking on it displays the Print submenu with which you select whether to print a diagram or template. The function then prints to the queue the currently displayed diagram or templates of the diagram.

Psychrometric Equation Blocks A category of function blocks. These blocks perform psychrometric calculations; included are ENRH (Enthalpy: Relative Humidity), ENDP (Enthalpy: Dewpoint), WBRH (Wetbulb: Relative Humidity), WBDP (Wetbulb: Dewpoint), DWPT (Dewpoint), and RH (Relative Humidity).

Reliability Blocks A category of function blocks. These blocks control or affect the reliability of outputs; included are FREL (Force Reliable) and UNRD (Unreliable).

Query Icon GPL Editor icon that lets you view, read, and modify an object in the archive database (single-click), edit a function or compound block template (single-click), and view a connection description (single-click). It also lets you view the contents of a compound (double-click on com-pound block). Double-clicking on the icon displays the Query option menu.

READ Option GPL Editor field that is on the Query option menu. Clicking on it performs a Session Read, which synchronizes a control strategy database with the archive database.

Remote Connection A noncontinuous line between two function blocks in the same file. Most often, a remote connection is used to conveniently connect two blocks that are on different diagrams, but can also connect two blocks on the same diagram.

REPL Option GPL Editor field that is on the Query option menu. Clicking on it allows you to globally replace system names of object blocks in a strategy file.

Report Blocks A category of function blocks. These blocks send reports to I/O devices; included are PRNT (Print) and ADV (Advisory).

Required Connection A connection line that is required by the block’s algorithm and may have an equivalent template field for its value.

Restart Process Special process created for a Network Control Module that performs required startup processing. Selecting a compound type of “Restart” in the compound template creates a Restart process.

SAVE Option GPL Editor field that lets you save a control strategy or compound to a file.

Scissors Icon GPL Editor icon that lets you move and size blocks, and move connection lines, text, and analog displays to any open area of the screen.

Segment A single section of a connection line.

Selector Blocks A category of function blocks. These blocks select between two or more inputs; included are HSEL (High Select), LSEL (Low Select), SWCH (Switch), SAMP (Sample and Hold), and MSEL (Mode Selector).

Session Read Process of synchronizing a control strategy database with the archive database for the purpose of matching the databases. See also READ Option.

SIM Option GPL Editor field that is on the Tools option menu. Clicking on it executes the Simulator. The Simulator screen replaces the GPL Editor screen.

Simulator GPL utility that emulates a process. Used to verify control logic before downloading a process to a Network Control Module.

Single-Click Pressing the mouse button once to perform an operation.

Special Blocks A class of function blocks. These blocks perform special functions; included are CONN (Connection), CNST (Constant), and FILE (File).

START OUTPUT Option GPL Editor field that is on the Print option menu. Clicking on it starts the printing of all diagrams and templates in the print queue.

Strategy See Control Strategy.

Strategy File See Control Strategy File.

SYSGEN Program A program (SYSGEN.EXE) that defines several parameters: the type of mouse used with GPL, whether the Operator Workstation or PC has a Drive B, whether to allow printing to any Windows printer, and the value of the Associated Input Timer.

System/Object Name A user-specified name that identifies an object.

Tab Fields Template fields with two or more entries that are displayed by pressing Tab.

Template An entry form of database information for a function or compound block. This information defines the characteristics of the block.

Template Field An open area on the template into which you enter information.

TEXT Option GPL Editor field that is on the Tools option menu. Clicking on it allows you to type comments on a diagram.

Time Blocks A class of function blocks. These blocks perform timing functions; included are TIME (Time), RTOT (Real to Time), and TTOR (Time to Real).

Tools Icon GPL Editor icon that offers several miscellaneous functions. These include: grid on/off, type text, analog display, Expert Checker, Simulator, Translator, and Compiler. Clicking on the Tools icon turns the grid on and off; double-clicking on the icon displays the Tools option menu.

TRAN Option GPL Editor field that is on the Tools option menu. Clicking on it executes the Translator. The Translator screen replaces the GPL Editor screen.

Translator GPL utility that translates a control strategy into intermediate source code.

Triggering An action that causes a process to run.

Turn Any point where a connection line changes direction.

.TX Extension The strategy file name extension for the file that contains text information (.TX=Text).

Undefined Block An object block that has not yet been added to the archive database. The border of an undefined block is dashed and colored magenta.

UNDO Option GPL Editor field under the Erase icon that lets you restore items (blocks, connections, text, and analog displays) on the screen that you previously erased.

Unreliable Data Data that is outside the prescribed range.

UP Option GPL Editor field that is on the File and Compound icon option menus, Connection and Find menus, and the print queue. Clicking on it displays the previous page of the menu.

USER Block A generic operation block that can be configured and linked to user block macro code to create a customized function.

VIEW Option GPL Editor field that shows the errors that the Session Read, Expert Checker, Translator, or Compiler found. The error file is in ASCII format and can be up to ten pages.

Work Area Area of the GPL Editor screen used to draw and edit diagrams.

X-Coordinate Position of the cursor on the x axis (range=0 to 1000). The current x-coordinate is shown on the GPL Editor in the lower left corner.

XPND Option GPL Editor field that lets you bring into the work area the expanded version (entire contents) of a compound.

Y-Coordinate Position of the cursor on the y axis (range=0 to 1000). The current y-coordinate is shown on the GPL Editor in the lower left corner.

Zoom Icon GPL Editor icon that lets you zoom a diagram up to four times. The amount of zoom is inversely proportional to the size of the white enclosing box (i.e., the smaller the white box, the larger the zoom). Double-clicking on this icon displays the Zoom and Pan option menu.