Foxboro® by Schneider Electric

Foxboro Evo™
Process Automation System

Control Processor 270 (CP270) and Field Control Processor 280 (FCP280) Integrated Control Software Concepts

B0700AG   *S*

B0700AG

Rev S
July 10, 2015

Contents

Figures

Tables

Preface

This document is intended for use by process control and applications engineers. It provides theoretical, descriptive, and conceptual information for process control and process control database configuration including block processing cycles, phasing, equipment control blocks, compounds, and blocks. You should read this document prior to attempting any control configuration. Complete block descriptions and lists of their parameters are found in *Integrated Control Blocks Descriptions* (B0193AX) and *Integrated Control Block Descriptions for FOUNDATION fieldbus Specific Control Blocks* (B0700EC).

This document describes control software concepts for the Field Control Processor 280 (FCP280) and the Control Processor 270 (CP270). It includes information on new alarming features, control blocks, Fieldbus Modules, and other information.

— NOTE

For information on control software concepts for the Control Processor 60 (CP60) or earlier control processors, refer to *Integrated Control Software Concepts* (B0193AW).

Revision Information

For this revision of the document (B0700AG-S), the following changes were made:

Appendix B “Fieldbus Modules”

- Updated “FBM203, FBM203b, FBM203c and FBM203d (RTD Inputs)” on page 219 and “FBM245 – Redundant 0 to 20 mA I/O Interface Module with HART® Support” on page 243 with changes related to RoHS compliance
- Added the Compact FBM247 to “FBM247 – Current/Voltage Analog/Digital/Pulse I/O Configurable Channel Interface Module with HART® Support” on page 244.

Reference Documents

Refer to the following documents for additional information:

- *Field Control Processor 280 (FCP280) User's Guide* (B0700FW)
- *Field Control Processor 280 (FCP280) Sizing Guidelines and Excel Workbook* (B0700FY)
- *Control Network Interface (CNI) User's Guide* (B0700GE)
- *Field Control Processor 270 (FCP270) User's Guide* (B0700AR)
- *Field Control Processor 270 (FCP270) Sizing Guidelines and Excel Workbook* (B0700AV)
- *Z-Module Control Processor 270 (ZCP270) User's Guide* (B0700AN)
- *Z-Module Control Processor 270 (ZCP270) Sizing Guidelines and Excel Workbook* (B0700AW)

- *Field Device System Integrators (FBM230/231/232/233) User's Guide* (B0700AH)
- *FOUNDATION fieldbus User's Guide for the Redundant FBM228 Interface* (B0700BA)
- *FOUNDATION fieldbus H1 Interface Module (FBM220/221) User Guide* (B0400FD)
- *PROFIBUS-DP™ Communication Interface Module (FBM223) User's Guide* (B0400FE)
- *HART® Communication Interface Modules User's Guide* (B0400FF)
- *Modbus Communication Interface Module (FBM224) User's Guide* (B0400FK)
- *DCS Fieldbus Modules for APACS+ Systems User's Guide* (B0700BK)
- *DCS Fieldbus Modules for Westinghouse WDPF Systems User's Guide* (B0400BA)
- *DCS Fieldbus Modules for ABB MOD300 Direct I/O Systems with HART I/O Capability User's Guide* (B0700AE)
- *Integrated Control Block Descriptions* (B0193AX)
- *Integrated Control Block Descriptions for FOUNDATION fieldbus Specific Control Blocks* (B0700EC)
- *Object Manager Calls* (B0193BC)
- *PLC™ Interface Block Descriptions* (B0193YQ)
- *Process Operations and Displays* (B0700BN)
- *High Level Batch Language (HLBL) User's Guide* (B0400DF)
- *PLC Interface Block Descriptions* (B0193YQ)
- *Time Synchronization User's Guide* (B0700AQ)
- *System Manager* (B0750AP)
- *System Management Displays* (B0193JC)
- *Control Core Service V9.x System Error Messages* (B0700AF)
- *Measurement Integration* (B0193RA)
- *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA)
- *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ)

For information on the I/A Series Configuration Component (IACC) software, refer to the following documents:

- *I/A Series Configuration Component (IACC) User's Guide* (B0700FE)
- *Learning to Use IACC* (B0400BT)
- *Intelligent Design Studio (IDS) Library for IACC* (B0400BQ)

For information on the Integrated Control Configurator (ICC) software, refer to the following document:

- *Integrated Control Configurator* (B0193AV)

Most of these documents are available on the Foxboro Evo Electronic Documentation media (K0174MA). The latest revisions of each document are also available through our Invensys® Global Customer Support at *https://support.ips.invensys.com.*

For information on the Foxboro Evo™ Control Software (formerly known as Foxboro® Control Software, or FCS), refer to the following documents:

- *Foxboro Evo Process Automation System Deployment Guide* (B0750BA) - Overview, Read First
- *Appearance Object Editor User's Guide* (B0750AE)
- *Bulk Data Editor User's Guide* (B0750AF)
- *Common Graphical Editor Features User's Guide* (B0750AG)
- *Block Configurator User's Guide* (B0750AH)
- *Control Database Deployment User's Guide* (B0750AJ)
- *PLB Ladder Logic Editor User's Guide* (B0750AK)
- *Sequence Block HLBL Editor User's Guide* (B0750AL)
- *Sequence Block SFC Editor User's Guide*(B0750AM)
- *Strategy Editor User's Guide* (B0750AN)
- *Configuration Utilities User's Guide* (B0750AZ)
- *Logic Block Editor and Troubleshooting Tool* (B0750BL)
- *Scripting with Direct Access User's Guide* (B0750BM)
- *Implementing PROFIBUS Networks in Foxboro Evo Control Software Applications* (B0750BE)
- *Implementing a DeviceNet Network on the Foxboro Evo Core Services Applications* (B0750CH)
- *Using HART Instrumentation on Foxboro Evo Core Services with the Control Software Field Device Manager* (B0750CM)
- *Implementing FOUNDATION fieldbus in Foxboro Evo Core Services Applications* (B0750DA)

These documents are available on the Foxboro Evo Control Software installation media kit (such as K0203AW, shipped with Foxboro Evo Control Software v6.0). The latest revisions of each document are also available through our Invensys Global Customer Support at *https://support.ips.invensys.com.*

Conventions

The following conventions are used throughout this document:

- The Field Control Processor 280 (FCP280), Field Control Processor 270 (FCP270) and the Z-module Control Processor 270 (ZCP270) are generally referred to as controllers.
- The term "CP270" refers to either the FCP270 or the ZCP270.
- A bold sans-serif font is used to denote a command, for example: **Show**.
- A plain sans-serif font is used to denote a code, for example: `SUBROUTINE SUBR3()`.
- Compounds, Blocks, and their parameters are denoted by UPPERCASE text.

Definitions and Acronyms

This section provides a convenient summary of the definitions, acronyms, and abbreviations used in this document.

Name	Meaning
AW	Application Workstation
BPC	Block Processing Cycle of the controller is the smallest resolution of time in which the Compound Processor task can be scheduled to run
Control Core Services	See “Foxboro Evo Control Core Services” below.
control network	Formerly known as “The Mesh control network”, The Foxboro Evo Control Network is a switch network that facilitates communications among Foxboro Evo workstations/servers and other stations.
CSA	Compound Summary Access
DCI	Distributed Control Interface
DCS Fieldbus Modules	Foxboro-provided Distributed Control System (DCS) FBMs - control and interface modules to third-party control solutions, such as Fisher's PROVOX® Series 20 or Honeywell® TDC 2000 systems.
ECB	Equipment Control Block
FBM	Fieldbus Module
FCM	Fieldbus Communication Module
FCP270	Field Control Processor 270
FCP280	Field Control Processor 280
FDSI	Field Data System Integrator
Foxboro Evo Control Core Services	Core software environment, formerly known as “I/A Series® (Intelligent Automation Series) software”. A workstation which runs this software is known as a “Foxboro Evo Control Core Services workstation”.
Foxboro Evo Control Editors	Formerly known as “FCS Configuration Tools”, “InFusion® Engineering Environment”, or “IEE”, these are the Control software engineering and configuration tools built on the ArchestrA® Integrated Development Environment (IDE).
Foxboro Evo Control Software	Formerly known as “Foxboro Control Software (FCS)” and “InFusion”, a suite of software built on the ArchestrA Integrated Development Environment (IDE) to operate with the Foxboro Evo Control Core Services.
Foxboro Evo Process Automation System	An overall term used to refer to a system which may include either, or both, Foxboro Evo Control Software and Foxboro Evo Control Core Services.
GPS	Global Positioning System
HDLC	High-level Data Link Control protocol is the Master/Slave Protocol used on top of several physical layers for FBM communication
IACC	I/A Series Configuration Component
ICC	Integrated Control Configurator

Name	Meaning
MTK	Master TimeKeeper
OM	Object Manager
PIO	Peripheral I/O bus
STK	Slave TimeKeeper
SMDH	System Management Display Handler, the user interface for equipment status and change actions
SOE	Sequence of Events
The Control Software	See “Foxboro Evo Control Software” above.
UTC	Universal Coordinated Time
ZCP270	Z-module Field Control Processor 270

1. Compounds and Station Blocks

This chapter gives an overview of compounds and blocks, including compound functions, attributes, and access, and compound/block alarming, phasing, and parameters. This chapter also covers the station compound or block and shadow parameters.

Process control for Foxboro Evo™ Process Automation Systems is based on the concepts of compounds and blocks. A compound is a logical collection of blocks that performs a control strategy. A block is a member of a set of algorithms that performs a certain control task within the compound structure. Figure 1-1 shows the compound/block relationship.

The compound provides the basis for the integration of:

- Continuous control
- Ladder logic
- Sequential control.

Within this structure, any block in any compound can be connected to any other block in any other compound in the system. The entire compound structure can be viewed through the workstation FoxView™ display.

The block contains parameters that have values of the types: Real, Boolean, Packed Boolean, Boolean Long, Integer, or String.

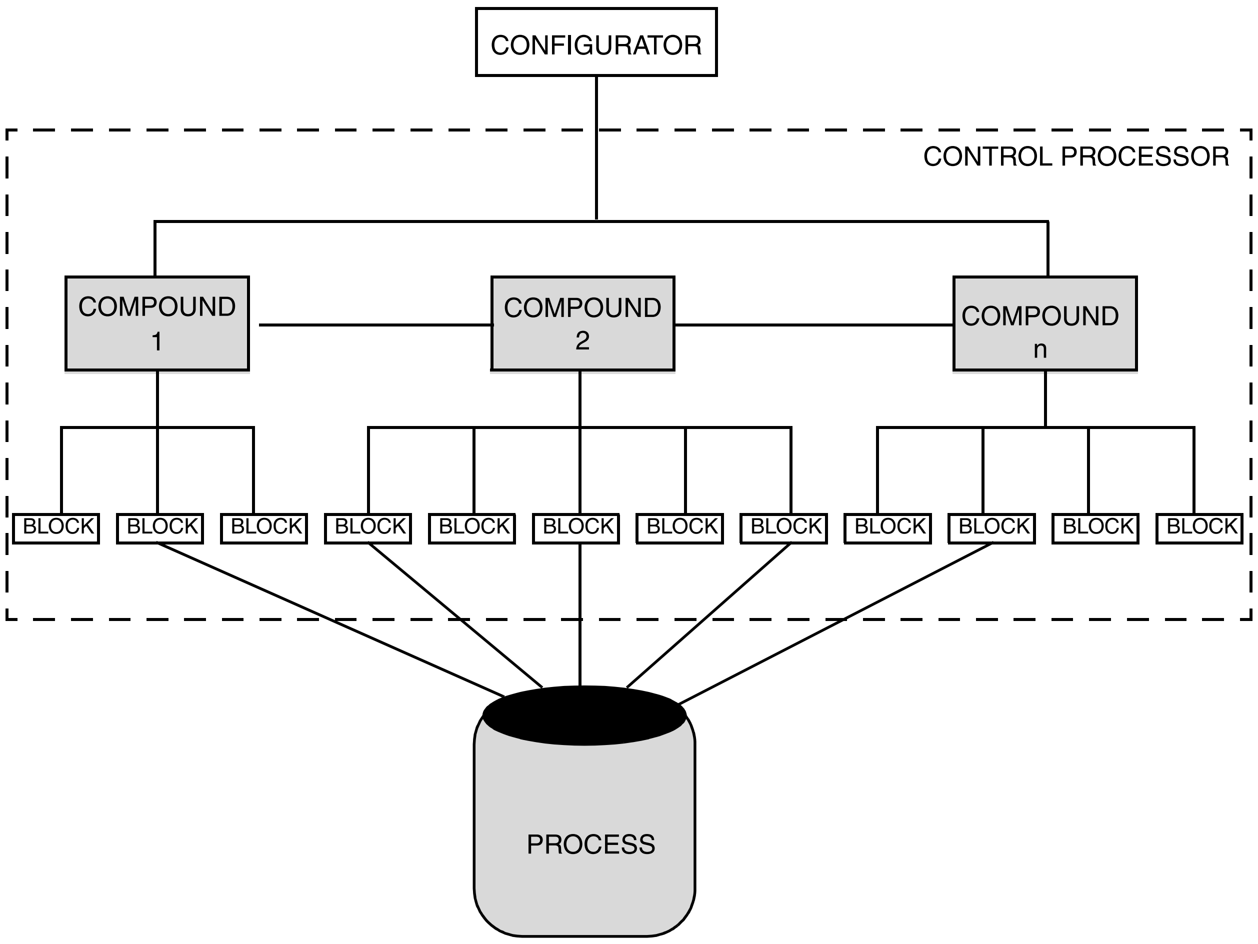


Figure 1-1. Compound/Block Relationship

Compound Functions

The compound supports the following functions for the related blocks:

- Process alarm priority, alarm inhibiting, and alarm grouping
- Sequence status notification (see "Sequential Control Block States" on page 144)
- Phasing for execution load leveling at execution time.

The compound rules are:

- Multiple compounds can be executed within the same station
- A single compound cannot cross station boundaries
- Blocks in different compounds can be interconnected across station boundaries
- Every compound must have a unique name.

Compound/Block Process Alarming

Alarms and status messages are generated by specific alarm blocks and by alarm options in selected blocks.

Alarms have five levels of priority, 1-5, (where 1 = highest priority) that enable you to quickly focus on the most important plant alarm conditions. An alarm priority of 0 indicates the absence of any alarm.

These are summarized in a single alarm summary parameter for each compound. This parameter contains the priority of the highest current alarm in that compound.

To reduce nuisance alarms, alarms can be inhibited at the compound level on a priority level basis. Alarms can also be inhibited at the block level, on either an alarm type basis, or an overall basis.

Alarms are initiated by the blocks within the compound. Alarm messages are then sent to groups of stations or applications (for example, workstations, historians, printers) according to configured alarm groups. The UNACK alarm acknowledge output parameter allows you to propagate alarm acknowledge actions to all blocks in a compound.

Stations, applications, and devices corresponding to various alarm destination groups are configured at the compound level or at the station level in the case of station compounds.

Group numbers for individual block alarm types are configured at the block level.

Compound/Block Phasing

A user-defined phase number can be assigned to each compound using a range of integer values that varies with assigned period.

Phasing allows the starting time of one compound/block to lead or lag the starting time of another compound/block, thereby leveling the block processor load.

Compound Attributes

The compound has the following attributes:

Name:	User-defined name that must be system-unique and no more than 12 characters in length. The name can be any mix of numerics (0 to 9), upper case alphabetics (A to Z), and the underscore (_).
Descriptor:	32-character field for user-defined identification.
On/Off:	Parameter that enables or disables the execution of all blocks within the compound, where: 1 = on; 0 = off.

Compound Naming for Interconnected Foxboro Evo Systems

When two Foxboro Evo or I/A Series systems are connected (using the Control Network Interface, as described in *Control Network Interface (CNI) User's Guide* (B0700GE)), all compound names must be unique across all inter-connected systems. If you have any compound names which exist in both systems, you must rename them to ensure all compound names are unique.

If any compounds in two or more interconnected Foxboro Evo systems share the same name, when a request is made to that compound, the system will return data from the local compound with that name.

Compound Access

Both compounds and blocks have a set of parameters that comprise the user interface. To access a compound parameter value, use the following convention:

Compound.Parameter

where:

- Compound has (up to) a 12-character name
- Parameter has (up to) a 6-character name.

Compound/Block Parameters

Compound and block parameters contain values that are of one of the types Real, String, Integer, Short Integer, Long Integer, Boolean, Packed Boolean, Packed Long, or Character.

Additionally, parameters are defined as being configurable, and either connectable/settable, not connectable/not settable, or a combination that is dependent upon the compound, block, and state.

Configurable Parameters

Configurable parameters are those parameters that can be defined through a control configurator. They can be displayable only, or displayable and editable.

NOTE

For I/A Series software v8.4-v8.8 and Foxboro Evo Control Core Services v9.0 (hereinafter referred to as the Control Core Services) or later, any non-zero value entered for a Boolean parameter will result in the parameter value being set to one, without exception. The Boolean value in the controller will be set to zero only when the user specifically configures a zero.
Also be aware that the ICC workfile value will contain the value entered by the user, so that if an “illegal” value (any value other than zero or one) is entered, there will be a mismatch between the content of the workfile and the value in the controller. This will not affect the process.

Connectable Parameters

Connectable parameters are those parameters of the user interface in which secured, change-driven connections may be made between network stations, or as local direct connections within the same station.

Each connection consists of a connectable source and a connectable sink. Output parameters (all outputs are connectable) are sources, while input parameters may be a sink or a source, or both.

Certain parameters that may be considered functional inputs, such as SPT in the PID blocks, and RATIO in the RATIO block, are settable but not connectable.

A connectable parameter has a value record that contains the parameter's value, its status, and its designated value type (Real, Boolean, or Integer). Its status consists of the following Boolean attributes:

Out-of-Service (OOS)	Defines the validity of the data. This flag is set and reset by the block algorithm. The OOS status usually originates from I/O type blocks (for example, AIN, COUT) which detect abnormal I/O conditions, for example, that the Fieldbus Module (FBM) is out-of-service, or that the compound containing the block of the I/O parameter is turned OFF.
Secure	Defines the conditional settability status of the parameter. The secure flag is set and reset by the block algorithm. A settable parameter can only be written if it is not secured. For example, a remote set point, RSP, is unsecured and settable when it is not connected. When connected, the parameter is secured and not settable.
Bad	Defines the validity of the data. This flag is set and reset by the block algorithm. The Bad status usually originates from I/O type blocks (for example, AIN, COUT) which detect abnormal I/O conditions such as a bad FBM, type mismatch, bad channel status, or out-of-range conditions.

You can access certain status bits of a parameter value record as explicit connections in control schemes, by using Boolean connection extensions. Certain CALC and LOGIC block instructions also have this capability, and user tasks can access these variables implicitly within their specific algorithms.

In addition, the BAD status of an I/O block's value record is made available as a unique Boolean-type connectable output parameter. This value can be accessed explicitly by any other block or task.

Input Parameters

Input parameters are connectable types that are the receivers of data from other connectable parameters via a path connection.

If no source path is specified during configuration, then the resident data of the value record is the actual "source" of data. It can be either the initial default or configured value, or a new value through a SET call to the input parameter.

If a source path is specified, then the data value is an output parameter of the same or another block, or a shared variable, thereby securing the input. By linking a shared variable to a block input during configuration, you can establish a long-term secured connection between a remote application program and the block input.

Output Parameters

All output parameters are connectable data sources that have value records. There are two types: settable and nonsettable.

The settability of a settable output is controlled by the secured status of the value record. The secured status is dependent on whether the block's operational mode is in Auto or Manual.

In either Auto or Manual, nonsettable output parameters cannot be written by any other source under any conditions.

Settable outputs may be conditionally released by the block algorithm in the Manual mode.

In Manual, the block unsecures settable output parameters. They can then be written by other tasks via SET calls. When the block switches to Auto, the block secures and updates its own output parameter(s).

Nonconnectable Parameters

Nonconnectable parameters have no value records and are not linkable. They mainly consist of string-type variables like NAME, or nonsettable parameters that are used in the configurator only, for example, block options. Local algorithm variables are also nonconnectable.

Nonconnectable parameters are generally accessible through GET calls.

There is also a class of nonconnectable input parameters that comprise the block user interface which can be manipulated through SET calls. An example is an alarm deadband.

Compound Parameters

Refer to *Integrated Control Block Descriptions* (B0193AX) and *PLC Interface Block Descriptions* (B0193YQ) for a detailed description of specific compounds and their parameters.

The Station Compound/Block

A Station compound containing one Station block for each control station in a system, is installed in a station automatically when a database is downloaded, even an "empty" database. This block provides global data storage for station system functions. You can view the station block by calling up its default display.

A Station Block provides information about the station's resources. The station block for all control stations are identical. Station blocks are used for each control station.

Each Station compound and block has a unique name in any Control Core Services network, determined by embedding the station letterbug in the compound name as follows:

- Compound name is: <*letterbug*>_STA
- Block name is: STATION
- Full pathname is: <*letterbug*>_STA:STATION.

The Station compound and block have the following restrictions:

- The compound cannot be deleted or turned off.
- The block cannot be deleted.
- User-created blocks cannot be added to the compound.
- The compound is not run periodically by the Control Processor Task (CPT).

Refer to *System Manager* (B0750AP) and *Process Operations and Displays* (B0700BN) for station block operational procedures and to *Integrated Control Block Descriptions* (B0193AX) for a detailed description of the station block and its parameters.

Dynamic Loading Calculations

The station, when enabled, performs several Dynamic Control Processor loading calculations. The first is the I/O scan load. The second calculation is for the loading for continuous block processing. The third calculation is for sequence processing.

Controller load is the total control block load, that is, the load value for continuous blocks, sequence blocks, and I/O. When the station initializes, the load calculation switch is automatically set to false and remains in that state under normal operation. You enable (or disable) the loading calculations at the Station Block display by toggling the **ACTIVE** pick.

You may synchronize the calculations to start at a specified phase number, LODPHS, or to start at the current phase. You request synchronization, at the block display, by setting the LODSYN input to true and toggling the STATION block to INACTIVE then ACTIVE.

LODPHS is specified in phase numbers corresponding to one minute and normalized to the system BPC; for example, the default system BPC is 0.5 seconds which normalizes the one minute period to 120 frames, giving LODPHS a domain of 0 to 119 (modulo 120). Thus, each load output is computed in the BPC frame equal to LODPHS plus their index.

You may also specify the sampling period, LODPER. The value limit is 10 * BPC to 3600 seconds. Default is 10 * BPC.

The station calculates separate load values for ten consecutive BPCs. On the tenth cycle, the station calculates two separate averages of the ten most recent cycles – one to calculate the load for continuous blocks and the other for the station load (continuous and sequence blocks).

At this time, all 22 load values are copied to the Station block output parameters (connectable, nonsettable). After the tenth cycle, the station suspends calculation until the sampling period expires, when the calculation cycles begin again. The station makes these calculations every sampling period until the loading calculations are disabled.

Since the real-time clock has a resolution of 10 ms, the load value accuracies are as follows:

Table 1-1. Load Value Accuracy

BPC	Maximum Error	Usefulness
0.1 s	10.0%	not very
0.2 s	5.0%	marginal
0.5 s	2.0%	useful
1.0 s	1.0%	useful

Dynamic Overrun Variables

The Station block provides an output parameter, OVRRUN, that is set to true (or false) each cycle to indicate when the Control Processor Task (CPT) processing does (or does not) overrun.

The station increments a separate overrun counter, CUMOVR, when an overrun cycle is detected. CUMOVR is available as an output of the Station block. You reset this counter by setting the block's momentary input parameter, RESOVR, to true.

Dynamic Free Memory Variable

The Station block indicates the number of bytes of dynamic free memory available for the control database.

The station updates this value a minimum of every thirty seconds.

Peer-to-Peer Status

The Station Block supplies status and performance data on the station's peer-to-peer communications. The performance and status information provided is as follows:

- the current number of peer-to-peer control block input connections configured in the control data base.
- a counter that contains the current number of peer-to-peer connections that have never been made.
- a counter that contains the current number of peer-to-peer connections whose source blocks or compounds have been deleted via a control configurator.
- a counter that contains the current number of peer-to-peer connections that have been disconnected due to a loss of peer-to-peer communications with the source station.

Deleted connection errors are temporary. When a station is checkpointed following the deletion of any of its control blocks, the status of any peer-to-peer sink connections to these blocks in other stations are changed from deleted to not found, and the station updates the not found and deleted counters accordingly.

The station software updates counters every two minutes.

Database Security

The Station block contains a database security parameter, Configuration Option (CFGOPT). When CFGOPT is true, database changes to any active block or Equipment Control Block (ECB) are disabled. The block is active if the compound in which it resides is ON, and the ECB is active if it is ON.

Time/Date

The Station block contains five parameters, YEAR, MONTH, DAY, HOUR, and MINUTE which provide user access to the system clock. These parameters are updated every 30 seconds by the station block software.

Station Block Parameters

Refer to the *Integrated Control Block Descriptions* (B0193AX) document for a detailed description of the station block and its parameters.

Time Synchronization

The Foxboro Evo Process Automation System supports time synchronization using either an externally maintained source of Universal Coordinated Time (UTC) from Global Positioning System (GPS) satellites or an internal source using proprietary software. UTC is the international time standard (commonly referred to as Greenwich Mean Time or GMT).

Time synchronization within a Control Core Services system synchronizes controllers/control processors to provide accurate timestamps for event and data reporting throughout the system. Time stamping is used for Sequence of Events (SOE) evaluation, transient data recording (TDR), and alarm messages.

Overview

A Master TimeKeeper (MTK), residing in an Application Workstation (AW), maintains the time source and distributes the system time to all other stations on the control network. A Slave Time-Keeper (STK) receives time information from the MTK and keeps itself synchronized with the MTK, and thus with all other stations in the control network. STKs reside in all controller and FCM100Et modules in the control network.

The MTK determines the time for synchronizing all slave stations by using either the AW's real-time clock (internal time source) or the optional GPS receiver and time strobe generator (external time source).

For complete information on time synchronization, refer to *Time Synchronization User's Guide* (B0700AQ).

Internal Source Time Synchronization

For internal source time synchronization (standard), the MTK station uses time from the internal clock in the hosting PC. The MTK distributes time as UTC to all stations on The Foxboro Evo Control Network (hereinafter the control network). This time is displayed as local time.

You enter the date and time in the MTK using the System Management Set Date and Time display. At runtime, you can also change the time using the Set Date and Time display or allow the time to continue to run on its internal clock.

For procedures on how to set the date and time using System Management, refer to *System Management Displays* (B0193JC).

External Source Time Synchronization

For external source time synchronization (optional), the MTK station uses an externally maintained source of Universal Coordinated Time (UTC) from GPS satellites. Equipment to support this option includes a GPS receiver and time strobe generator.

The MTK uses a hardware connection to the controller and FCM100Et modules to increase the synchronization accuracy by providing a time strobe pulse, which is sent continuously by the MTK at a precise time interval. The controller and FCM100Et modules have built-in hardware to receive the sync pulses generated by the MTK.

When using GPS time synchronization, the ZCP270 plays the master timekeeper role for the FCM100Ets in its I/O network. The FCM100Ets use their synchronized time to synchronize the FBMs.

2. Blocks

This chapter defines the path for block parameters, describes the common block parameters, lists all block types, and lists the control stations that host each type of control block.

A block has one or more inputs/outputs and performs a predefined process function that has been prespecified by an algorithm.

There are continuous, sequence, and ladder logic block function types that can be mixed and matched to satisfy your integrated control needs.

This section briefly defines the block set. However, detailed information can be found in *Integrated Control Block Descriptions* (B0193AX).

Block Attributes

A block has the following attributes:

NAME	Name is a user-defined string that must be unique within the compound and up to 12 characters in length. The name can be any mix of numerics (0 to 9), upper case alphabetics (A to Z), and the underscore (_).
TYPE	Type is a system-defined name (up to six characters) that identifies the algorithm control function. In the Integrated Control Configurator, for example, you can enter the block-type string (for example, MAIN) or you can select the desired type from a block type list. To display the block type list, select SHOW from the menu-bar and select BLOCK TYPE NAMES from the SHOW menu. Type is entered as a string, but is stored in the database as an indexed integer. (The Object Manager (OM) Get command retrieves this integer value, **not** the string.) Therefore, to be consistent with the database, the parameter tables in *Integrated Control Block Descriptions* (B0193AX) list the data type as an integer.

Block Access

To access a block parameter value from outside the resident compound, you must use the entire path:

Compound:Block.Parameter

Connectivity between blocks in different compounds is through this same convention, whether the compounds are in the same or different stations.

To connect to either a block parameter value from another block within the same compound, or to a parameter value from within the same block, you can use:

:Block.Parameter

Since block names do not have to be unique across compounds, the following example shows two compounds, REFLUX and EN_BAL, each having a block called F100.

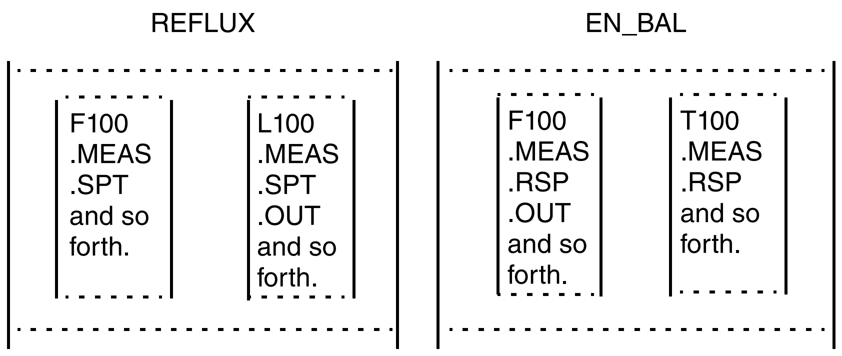


Figure 2-1. Block Access

Approximate Block Sizes

Each control block uses a different amount of memory in a Control Processor. The type and number of control blocks determines the Control Processor loading. Refer to the following documents for complete and precise memory and loading calculations:

- *Field Control Processor 280 (FCP280) Sizing Guidelines and Excel Workbook* (B0700FY)
- *Field Control Processor 270 (FCP270) Sizing Guidelines and Excel Workbook* (B0700AV)
- *Z-Module Control Processor 270 (ZCP270) Sizing Guidelines and Excel Workbook* (B0700AW)

Block Parameters

Like compounds, each block contains select parameters that serve as the inputs and outputs of their respective functions. These parameters follow the same conventions as compound parameters regarding their value types and connectability/settability.

— NOTE

For I/A Series software v8.4-v8.8 and Control Core Services v9.0 or later, any non-zero value entered for a Boolean parameter will result in the parameter value being set to one, without exception. The Boolean value in the controller will be set to zero only when the user specifically configures a zero.
Also be aware that the ICC workfile value will contain the value entered by the user, so that if an “illegal” value (any value other than zero or one) is entered, there will be a mismatch between the content of the workfile and the value in the controller. This will not affect the process.

For information on accessing block parameters from a user task, refer to *Object Manager Calls* (B0193BC).

Common Parameters

All blocks, with exceptions as noted, have the following common parameters:

NAME	Name is a user-defined string of up to 12 characters, which must be unique within the compound, used to access the block and its parameters. The string can be any mix of numerics (0 to 9), upper case alphabetics (A to Z), and the underscore (_).
TYPE	Type is a system-defined name (up to six characters) that identifies the algorithm control function. Type is entered as a string, but is stored in the database as an indexed integer. (The Object Manager Get command actually retrieves this integer value, *not* the string.) Therefore, to be consistent with the database, the parameter tables in *Integrated Control Block Descriptions* (B0193AX) list the TYPE parameter's data type as an integer.
DESCRP	Description is a user-defined string of up to 32 characters that describe the block's function (for example, "PLT 3 FURNACE 2 HEATER CONTROL").
PERIOD	Period is an indexed, nonconnectable, input parameter that specifies the block's execution time base. Data Variable blocks (for example, BOOL, LONG, and so forth.) do not have the PERIOD parameter.
PHASE	Phase is an integer input that causes the block to execute at a specific BPC within the time determined by the PERIOD. For instance, a block with PERIOD of 3 (2.0 sec) can execute within the first, second, third, or fourth BPC of the 2-second time period, assuming the BPC of the Control Processor is 0.5 sec. "Block Phasing" on page 64 provides further details.
MA	Manual/Auto is a connectable Boolean input that controls the Manual/Automatic operating state (0 = False = Manual; 1 = True = Auto). The BLNALM and Data Variable blocks do not have the MA parameter.
INITMA	Initialize Manual/Auto specifies the desired state of the MA input during initialization: 0 = Manual 1 = Auto 2 = No change, except if a reboot, use the MA state specified in the checkpoint file The BLNALM and Data Variable blocks do not have the INITMA parameter.
LOOPID	Loop Identifier is a configurable string of up to 32 characters which identify the loop or process with which the block is associated. It is displayed on the detail display of the block, immediately below the faceplate. Data Variable blocks do not have the LOOPID parameter.
OWNER	Owner is a settable string of up to 32 ASCII characters which are used to allocate control blocks to applications. Attempts to set OWNER are successful only if the present value of OWNER is the null string, an all-blank string, or identical to the value in the set request. Otherwise the request is

	rejected with a LOCKED_ACCESS error. OWNER can be cleared by any application by setting it to the null string; this value is always accepted, regardless of the current value of OWNER. Once set to the null string, the value can then be set as desired. The EVENT, FBTUNE, FFTUNE, and Data Variable blocks do not have the OWNER parameter.
LOCKRQ	Lock Request is a boolean input which can be set true or false only by a set command from the LOCK U/L toggle key on workstation displays. When LOCKRQ is set true in this fashion a workstation identifier accompanying the set command is entered into the LOCKID parameter of the block. Thereafter, set requests to any of the block's parameters are honored (subject to the usual access rules) only from the workstation whose identifier matches the contents of LOCKID. LOCKRQ can be set false by any workstation at any time, whereupon a new LOCKRQ is accepted, and a new ownership workstation identifier written to LOCKID. The MSG and DCI output blocks do not have the LOCKRQ parameter.
LOCKID	Lock Identifier is a string identifying the workstation which has locked access to the block via a successful setting of LOCKRQ. LOCKID has the format LETTERBUG:DEVNAME, where LETTERBUG is the 6-character letterbug of the workstation and DEVNAME is the 1-6 character logical device name of the Display Manager task. The MSG and DCI output blocks do not have the LOCKID parameter.

Editing Parameters

You may edit block parameters with a control configurator. Refer to the user document associated with control configurator you are using for a description of compound and block editing functions.

Block Function Types

The following list gives a brief overview of all the Integrated Control software block function types. For a detailed description of each, refer to *Integrated Control Block Descriptions* (B0193AX), with the exception of the blocks for FOUNDATION™ fieldbus (FF) devices, which are described in *Integrated Control Block Descriptions for FOUNDATION fieldbus Specific Control Blocks* (B0700EC).

ACCUM (Accumulator)	The ACCUM block integrates a real input (rate or pulse count) signal and scales it to produce a real output quantity of the running total. Inputs are provided to let you clear, preset, and hold the accumulator output.
AI[1, 2] (Analog Input)	The AI block connects to an AI function block in a Foundation fieldbus (FF) H1 device via an FBM228 to read an analog measurement input and status data. The AI block supports AI function block configuration through parameters that are downloaded by the FBM228 into the AI function block in the FF H1 device. The AI block provides absolute and bad point alarming of the analog input, and other standard input block alarm functions. It also provides a simulation option.
AIN (Analog Input)	The AIN block supports a single input from an analog-type FBM. Provisions exist to condition, scale, clamp, and filter the input, and alarm the hardware status and output value.
AINR (Redundant Analog Input)	The AINR block supports a single input point from two redundant analog-type FBMs. Provisions exist to condition, scale, clamp, and filter the selected input, and alarm the hardware status and output value.
ALMPRI (Alarm Priority Change)	The ALMPRI block is used to dynamically reassign the priority of an alarm point.
AO[1, 2] (Analog Output)	The AO block connects to an AO function block in a Foundation fieldbus (FF) H1 device via an FBM228 to write an analog output to the AO function block. The AO block supports AO function block configuration through parameters that are downloaded by the FBM228 into the AO function block in the FF H1 device. The AO block provides bad point alarming of the analog output readback value, and other standard output block alarm functions. It also provides failsafe and simulation options, and supports cascade initialization.
AOUT (Analog Output)	The AOUT block provides an auto/manual with bias (AMB) function. It biases, clamps, and conditions the input and drives a single output point within an analog-type FBM. Provisions exist to alarm the hardware status of the connected FBM.
AOUTR (Redundant Analog Output)	The AOUTR block provides an auto/manual with bias (AMB) function. It biases, clamps, and conditions the input and drives a single output point via a dual pair of redundant analog-type FBMs. Provisions exist to alarm the hardware status when both FBMs have bad status.
BIAS (Bias)	The BIAS block produces an output that is the sum of the two input values, MEAS and BIAS, each of which can be scaled independently. The block supports measurement alarm messages. It does not support output alarm messages. The BIAS block supports cascade initialization.
BIN (Binary Input)	The BIN block receives one binary value from an external device via a Distributed Control Interface (DCI).

BINR (Binary Input, Redundant)	The BINR block receives one binary value from an external device. The source of the value can be specified as either two or three redundant inputs. The redundant inputs can either be in the same device or in different devices. The block's selection algorithm determines which of the two or three input values is presented to the control strategy as the block output. BINR supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
BLNALM (Boolean Alarm)	This block provides independent state-change alarm messages for each of eight Boolean-type inputs.
BOOL (Boolean Data Variable)	This block provides the capability of creating a settable and configurable boolean data value for use by other control blocks.
BOUT (Binary Output)	The BOUT block sends one binary value to an address in an external device. It also continuously reports, to the Foxboro Evo Process Automation System, any changes made by the device to the value at this address. BOUT supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
BOUTR[1] (Binary Output, Redundant)	The BOUTR block can send one binary value to either two or three redundant outputs. The outputs may be in the same or different external devices. An arbitration algorithm determines which of the two or three readback values is to be used. The BOUTR block supports connectivity of Foxboro control stations only to DCS FBMs for migration of APACS+ process automation systems.
CALC (Calculator)	The Calculation block provides up to 50 sequentially executed arithmetic and logical operations. It has the capability of a programmable scientific calculator.
CALCA (Advanced Calculation Block)	The CALCA block adds dual-operand efficiency to many mathematical and logical calculation operations.
CHARC (Characterizer)	This block converts a real input to a real output using a table lookup of piecewise linear segments. Up to 20 segments can be used.
CIN (Contact Input)	The CIN block supports a single input point from a digital input type FBM. The block also provides an input inversion option.
CINR[1] (Contact Input, Redundant)	The CINR block receives redundant input values for a single digital input process point from two digital input type FBMs. Based on the quality of the two inputs and the user specification of a default selection, one of the inputs is chosen for use in the control strategy. The CINR block provides bad point and state alarming of the digital input, and other standard Control Core Services block alarm functions. The block also provides input inversion and simulation options. The CINR block supports connectivity of Foxboro control stations to DCS FBMs for migration of APACS+ process automation systems.
COUT (Contact Out)	The COUT block writes single output to a digital type FBM. The block also provides an output pulsing option.

COUTR[1] (Contact Out Redundant)	The COUTR block drives a single contact output point to two digital FBMs. It provides output pulsing and simulation mode options. Bad point alarming is indicated when both FBMs are bad or the primary and secondary contact output readback point values are bad. Failsafe is indicated when both outputs have been driven to failsafe. The block also provides an option to use the last good value for the contact output when the contact input is in error, bad, or out of service. The COUTR block supports connectivity of Foxboro control stations to redundant FBM240 modules and DCS FBMs for migration of APACS+ process automation systems.
DEP (Dependent Sequence)	This block contains user-programmable statements that can manipulate compound or block parameters, or shared variables. It can also activate other sequence and monitor blocks, along with sending messages to displays and historians. A Dependent Sequence block's execution is automatically delayed while any Exception Sequence block in the same compound is running.
DGAP (Differential Gap)	This block outputs two discrete output values that can be used for on/off control of bi-state or tri-state actuated final actuator devices. The outputs depend on the difference between the measurement, set point, and adjustable error GAP. The DGAP block does not support cascade initialization.
DI[1, 2] (Digital Input)	The DI block connects to a DI function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to read a digital input and status data. The DI block supports DI function block configuration through parameters that are downloaded by the FBM228 into the DI function block in the FF H1 device. The DI block provides bad point alarming of the digital input and other standard Control Core Services block alarm functions. It also provides a simulation option.
DO[1, 2] (Digital Output)	The DO block connects to a DO function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to write a digital output to the DO function block. The DO block supports DO function block configuration through parameters that are downloaded by the FBM228 into the DO function block in the FF H1 device. The DO block provides bad point alarming of the digital output readback value, and other standard output block alarm functions. It also has failsafe and simulation options, and supports cascade initialization.
DTIME (Deadtime)	This block delays the input a specific length of time before making it available at the output. It is typically used to simulate process transport delay and to compensate feed-forward signals.
EVENT (Sequence of Events)	The EVENT block provides message reporting for a sequence of state-change events detected in a contact input FBM. The connected FBM must be an input-only type.

EXC (Exception Sequence)	This block contains user-programmable statements that can manipulate compound or block parameters, or shared variables. It can activate other sequence and monitor blocks, along with sending messages to displays and historians. When it activates, all Dependent Sequence blocks in the same compound delay executing until the Exception Sequence block finishes its execution.
FBTUNE (Feedback Self-Tuner)	The FBTUNE block is used to adaptively tune the proportional band, the integral time, derivative time, dead time, and the set-point-filter lead-lag ratio of the PIDA block.
FFTUNE (Feedforward Self-Tuner)	The FFTUNE block is used to adaptively tune the feedforward compensators for the PIDA block.
GDEV (General Device)	This block provides Open/Close control of motor- or air-operated valves, and Run/Stop control of 2-wire, or 3-wire, motor circuits.
IIN (Integer Input)	The IIN block receives one integer value from an external device. The actual receipt and processing of this value is subject to the conditions established by the Simulation Option and the Auto/Manual mode of the IIN block. IIN supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
IINR[3] (Integer Input Redundant)	The IINR block reads one set of redundant integer input values from an external device. The actual receipt and processing of this value is subject to the conditions established by the Simulation Option and the Auto/Manual mode of this block. IINR supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI). It accepts redundant inputs from a single device hosted by a DCI FBM (single or redundant), from separate devices hosted by the same DCI FBM (single or redundant) or different non-redundant DCI FBMs, or from the same device with redundant connections to different non-redundant DCI FBMs. It provides a milliseconds-since-midnight timestamp for its values from FOUNDATION fieldbus, dual or triple modular redundancy, and a simulation mode of operation.
IND (Independent Sequence)	This block contains user-programmable statements that can manipulate compound or block parameters, or shared variables. It can also activate other sequence and monitor blocks, along with sending messages to displays and historians. An Independent Sequence block does not affect the execution of other sequence blocks nor does the execution of other blocks affect the operation of Independent Sequence blocks.
IOUT (Integer Output)	The IOUT block sends one integer value to a field device. It also continuously reports any changes made by the device. These reports are made to the value at the same address. IOUT supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
LIM (Limiter)	The Limiter block provides a position and velocity limiter function for a real input signal.
LLAG (Lead-Lag)	This block performs dynamic signal compensation by making the output dynamically lead or lag the input.

LOGIC **(Logic)**	The Logic block provides 15 sequentially-executed logical functions.
LONG (Long Integer Data Variable)	This block provides the capability of creating a settable and configurable long integer data value for use by other control blocks.
MAI[1, 2] (Multiple AI)	The MAI block connects to a MAI function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to read multiple analog measurement inputs and status data. The MAI block supports MAI function block configuration through parameters that are downloaded by the FBM228 into the MAI function block in the FF H1 device. The MAI block provides absolute and bad point alarming of the analog inputs, and other standard input block alarm functions. It also provides a simulation option. It provides up to eight milliseconds-since-midnight timestamps for its values from FOUNDATION fieldbus (one per analog input and output).
MAIN (Multiple Analog Input)	The MAIN block supports up to 8 inputs from an analog-type FBM. An internal channel for a temperature reference sensor is also provided.
MAO[1, 2] (Multiple AO)	The MAO block connects to a MAO function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to write multiple analog outputs to the MAO function block. The MAO block supports MAO function block configuration through parameters that are downloaded by the FBM228 into the MAO function block in the FF H1 device. The MAO block provides bad point alarming of the analog output readback values, and other standard output block alarm functions. It also provides failsafe and simulation options, and supports cascade initialization. It provides up to eight milliseconds-since-midnight timestamps for its values from FOUNDATION fieldbus (one per analog input and output).
MATH (Mathematics)	The MATH block provides a set of mathematics functions for specialized control needs.
MCIN (Multiple Contact Input)	The MCIN block supports up to 32 inputs from digital input type FBMs.
MCOUT (Multiple Contact Out)	The MCOUT block supports up to 16 digital outputs to a digital type FBM.
MDI[1, 2] (Multiple Discrete Input)	The MDI block connects to a MDI function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to read multiple discrete measurement inputs and status data. The MDI block supports MDI function block configuration through parameters that are downloaded by the FBM228 into the MDI function block in the FF H1 device. The MDI block provides absolute and bad point alarming of the discrete inputs, and other standard input block alarm functions. It also provides a simulation option. It provides up to eight milliseconds-since-midnight timestamps for its values from FOUNDATION fieldbus (one per discrete input and output).

MDO[1, 2] (Multiple Discrete Output)	The MDO block connects to a MDO function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228 to write multiple discrete outputs to the MDI function block. The MDO block supports MDO function block configuration through parameters that are downloaded by the FBM228 into the MDO function block in the FF H1 device. The MDO block provides bad point alarming of the discrete output readback values, and other standard output block alarm functions. It also provides failsafe and simulation options, and supports cascade initialization. It provides up to eight milliseconds-since-midnight time-stamps for its values from FOUNDATION fieldbus (one per discrete input and output).
MEALM (Measurement Alarm)	The MEALM block serves as an alarm annunciator to activate the Foxboro alarm mechanism upon alarm conditions detected by an external source.
MON (Monitor Sequence)	This block monitors up to 16 process conditions. It monitors parameter values and Boolean expressions and triggers Exception, Dependent, or Independent blocks.
MOVLV (Motor-Operated Valve)	This block operates two related output contacts which open or close a valve on an incremental basis. It supports optional feedback from one or two contacts (limit switches) for mismatch alarming.
MSG (Message Generator)	The MSG block generates state change messages upon transitions of its Boolean inputs.
MTR (Motor Controller)	This block performs both 2- and 3-wire motor control functions.
OUTSEL (Output Select)	The OUTSEL block controls strategies that require the higher or lower of two input signals to be selected as the final output signal to the process, while providing the appropriate handshake data to prevent integral action from "winding up" in the block containing the unselected signal. The block also provides separate cascade initialization each of the two upstream blocks.
PACK (Packed Long Data Variable)	This block provides the capability of creating a settable and configurable packed long data value for use by other control blocks.
PAKIN (Packed Input)	The PAKIN block reads up to 32 bits of discrete data from an external device. Each bit represents a binary value having opposing states, such as ON and OFF, or START and STOP. The PAKIN block supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).

PAKINR[3] (Packed Input Redundant)	The PAKINR block reads one set of redundant Packed Boolean inputs, each up to 32 bits of discrete data from an external device. Each bit represents a binary value having opposing states, such as ON and OFF, or START and STOP. The PAKINR block supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI). It accepts redundant inputs from a single device hosted by a DCI FBM (single or redundant), from separate devices hosted by the same DCI FBM (single or redundant) or different non-redundant DCI FBMs, or from the same device with redundant connections to different non-redundant DCI FBMs. It provides a milliseconds-since-midnight timestamp for its values from FOUNDATION fieldbus, dual or triple modular redundancy, and a simulation mode of operation.
PAKOUT (Packed Output)	The PAKOUT writes up to 32 bits of discrete data to an external device. Each bit represents a binary value having opposing states, such as ON and OFF, or START and STOP. The PAKOUT block supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
PATALM (Pattern Alarm)	This block compares the relationship of up to 16 Boolean inputs to a user-specified pattern.
PATT (Pattern Match)	The PATT block provides pattern matching capability for 16-bit patterns.
PID (Proportional, Integral, Derivative)	The PID block provides the functions of the traditional interacting 3-term controller. The PID block supports cascade initialization. The PIDA (with FBTUNE and FFTUNE when necessary) is recommended for use in all PID applications. The PIDA block has all of the functionality of the older PID algorithms plus additional functionality.
PIDA (Proportional, Integral, Derivative, Advanced)	The PIDA block implements continuous PID or dead-time feedback and additive and multiplicative feedforward control of an analog loop, providing advanced features beyond those of the PID and PIDX blocks.
PIDE (PID with EXACT™)	This block provides the same capability as the PID block with the addition of the EXACT Self-Tuning algorithm.
PIDFF[1, 2] (FOUNDATION fieldbus Proportional, Integral, Derivative)	The PIDFF block acts as an interface to a PID function block in a FOUNDATION fieldbus (FF) H1 device via an FBM228. The PID function block provides the functions of the traditional interacting 3-term controller and supports cascade initialization.
PIDX (PID Extended)	This block provides the same capability as the PID block, with optional capability for nonlinear gain compensation, sampling mode, and batch control preload.
PIDXE (PID Extended with EXACT)	This block adds the EXACT algorithm to the PIDX block.
PLB (Programmable Logic Block)	The Programmable Logic Block supports ladder logic executing in an FBM. The block provides 32 input and 32 output parameters.

PLSOUT (Pulse Output)	The PLSOUT block allows the control strategy or operator to output ON and OFF, or START and STOP, type commands through momentary pulsed outputs on two separate lines, one for each state. PLSOUT supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
PTC (Proportional Time Controller)	This block performs the functions of a proportional-time on/off controller. The PTC block does not support cascade initialization.
RAMP (Ramp)	This block performs a general purpose ramp function.
RATIO (Ratio)	This block computes an output that is the scaled multiplication of a measurement input with a ratio set-point input. The RATIO block supports cascade initialization.
REAL (Real Data Variable)	This block provides the capability of creating a settable and configurable real data value for use by other control blocks.
REALM (Real Alarm)	The REALM block optionally supports three types of alarming: ♦ High-low absolute alarming on the measurement. ♦ Rate-of-change alarming on the measurement. ♦ High-low deviation alarming on the difference between measurement and set point.
RIN (Real Input)	The RIN block receives one real value from an external device. It presents that value, after input processing, at parameter RINP. The RIN block supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
RINR (Real Input, Redundant)	The RINR block receives one real value from an external device. The source of the value can be specified as either two or three redundant inputs. The redundant inputs can either be in the same device or in different devices. Each of the redundant inputs is independently scaled, limited, and converted into engineering units before the block's selection algorithm is invoked to determine which of the two or three inputs is set into parameter RINP. The RINR block supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
ROUT (Real Output)	The ROUT block sends one real value to an address in an external device. It also continuously reports any changes made by the device to the value at this address. ROUT is used in a Distributed Control Interface (DCI). DCI blocks support connectivity of Foxboro control stations to various FBMs via a general purpose interface.
ROUTR[1] (Real Output, Redundant)	The ROUTR block can send one real value to either two or three redundant outputs. The outputs may be in the same or different external devices. The output value is clamped or limited, and then converted into engineering units before being sent to the redundant outputs. An arbitration algorithm determines which of the two or three readback values is used. The ROUTR block supports connectivity of Foxboro control stations only to DCS FBMs for migration of APACS+ process automation systems.

SIGSEL (Signal Selector)	This block examines up to eight inputs and produces an output dependent upon a relational selection option.
STALM (State Alarm)	The STALM block serves as an alarm annunciator to activate the Control Core Services alarm mechanism upon alarm conditions detected by an external source.
STATE (State)	The STATE block outputs selected 16-bit patterns.
STRIN (String Input)	The STRIN block receives one string value from an external device. STRIN supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
STRING (String Data Variable)	This block provides the capability of creating a settable and configurable string data value for use by other control blocks.
STROUT (String Output)	The STROUT block sends a string output to an external device. STROUT supports connectivity of Foxboro control stations to various FBMs via a Distributed Control Interface (DCI).
SWCH (Switch Position Selector)	This block selects either of two independent inputs. Each input can be a real, integer, or Boolean variable. The SWCH block supports cascade initialization.
TIM (Timer Sequence)	This block keeps track of time while control strategies are executing.
UNIVFF[1, 2] (Universal FOUNDATION fieldbus)	The UNIVFF block provides an interface between the control processor and a resource, transducer or device function block operating in a FOUNDATION fieldbus (FF) H1 device. The function block type can be any FOUNDATION fieldbus device block type not supported by a dedicated FOUNDATION fieldbus function block such as AI or AO. (See B0700EC for the list of dedicated FOUNDATION fieldbus function blocks.) Communications are performed via an FBM228.
VLV (On-Off Valve Control)	This block is typically used to operate two related output contacts which open or close a valve on an ON/OFF basis.

1. New blocks for use in the FCP280, FCP270 and ZCP270.
2. New blocks for use with FOUNDATION fieldbus, and described in *Integrated Control Block Descriptions for FOUNDATION fieldbus Specific Control Blocks* (B0700EC).
3. New blocks for use in FDSI.

Supervisory Set Point Control (SSC)

Supervisory Set Point Control (SSC) is supported in all the PID control blocks, RATIO, AOUT, and AOUTR control blocks (Figure 2-2).

The station block Supervisory Groups display allows you to enable or disable control blocks assigned to each of eight supervisory groups. In addition, the station block shows the fallback status and fallback request status for each group, and the (enabled/disabled) running time and set point for each fallback timer.

The supported control blocks include an option to allow the control processor to close supervisory cascades automatically when the supervisory input is written without requiring an explicit acknowledgement from an SSC-related application. This allows applications unable to send the acknowledgement to use the Control Core Services SCC options without hindrance.

From the station block Supervisory Groups, you can:

- enable or disable SSC for any group
- request fallback status for a group,
- enable or disable a fallback timer.

SSC is supported in the FCP280, FCP270 and ZCP270 control processors.

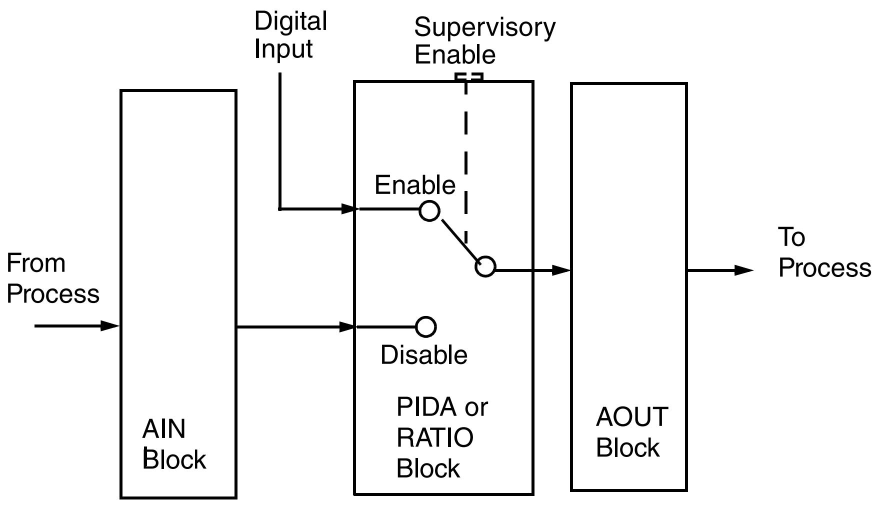


Figure 2-2. Supervisory Set Point Control Concept

3. Block Characteristics

This chapter describes common aspects of block characteristics, including implicit and explicit block initialization, PRIBLK configuration, configuring cascade schemes, and error handling, as well as clamping, scaling, secure/release, control block fail-safe strategy, and the PID hold constraint.

Block Initialization

The need for block initialization exists when:

- ♦ Compounds switch from OFF to ON
- ♦ Blocks recover from re-established connections or from bad process inputs
- ♦ Blocks change modes (for example, from Manual to Auto)
- ♦ Controller blocks in cascade schemes return from open loop conditions
- ♦ The Control Processor is rebooted
- ♦ A block or compound is modified or added during on-line configuration while the compound is ON
- ♦ The Control Processor recovers from a power failure.

Initialization within controller-type blocks is performed without bumping the process. Achieving proper initialization depends upon the features available in each type of block and how blocks are interconnected in a particular control scheme.

There are two types of initialization, referred to as implicit and explicit. Implicit initialization is the execution of block-specific start-up and validation logic which occurs whenever any block begins execution within the control strategy, and not thereafter. Explicit initialization is the bumpless return to control following any disruption of control. It is a process involving multiple blocks within a cascade, and is a response to exception situations which can occur at any time. All blocks support implicit initialization, but only controller and output-type blocks, which require the bumpless transfer function, support both implicit and explicit initialization.

Implicit Initialization

All blocks perform implicit initialization when requested to start up. This occurs automatically when the compound is first switched on, after a block is modified, on a reboot, after a power fail recovery, or when a block is added to an existing compound through the on-line configuration process.

Validation logic is performed by the block to check the integrity of its block record. Critical, "configuration only" (that is, nonconnectable/nonsettable) parameters are checked to make certain that they are within acceptable values prior to start-up.

If any of these inputs are not within acceptable limits, the block is declared undefined and enters the Undefined state (refer to "Block States" on page 39). In this state, the block is not processed and the status of all its connectable parameters is set to Release and Bad.

INITMA sets a block's MA parameter during the execution cycle in which the block implicitly initializes. The default is 1 (Auto), but INITMA is overridden if the MA parameter has an established linkage. Note that INITMA overrides the MA value set by a checkpoint file or by a power failure recovery.

Explicit Initialization

Introduction

Controller-type and output-type blocks must initialize to the current state of the process. This means that output block reads back the current state of the device and causes the upstream blocks control action to initialize to the present state of the process and/or manipulated variable.

The order in which cascaded blocks initialize is in the opposite order that they normally execute. When controller-type and output-type blocks are cascaded in a control scheme, the open-loop effects of any downstream block action must be propagated to upstream block(s). This ensures bumpless return to the current state when the downstream block returns to closed-loop operation.

Blocks that support explicit cascade initialization have two connectable parameters. They are BCALCI (back calculation in) and BCALCO (back calculation out) which are real input/output parameters. These parameters have two status bits (INITU - Initialize Unconditional, and INITC - Initialize Conditional) which are used to notify upstream blocks of the need for initialization and a PRIBLK acknowledgment request bit (INITC) to ensure that the upstream block does not initialize until the downstream block acknowledges that it is initialized.

When a block sees one of these status bits (INITU or INITC) set in its BCALCI parameter, it performs the specified initialization algorithm and sets one of the status bits in its BCALCO parameter. The BCALCO parameter is connected to the BCALCI of an upstream block as shown in Figure 3-1.

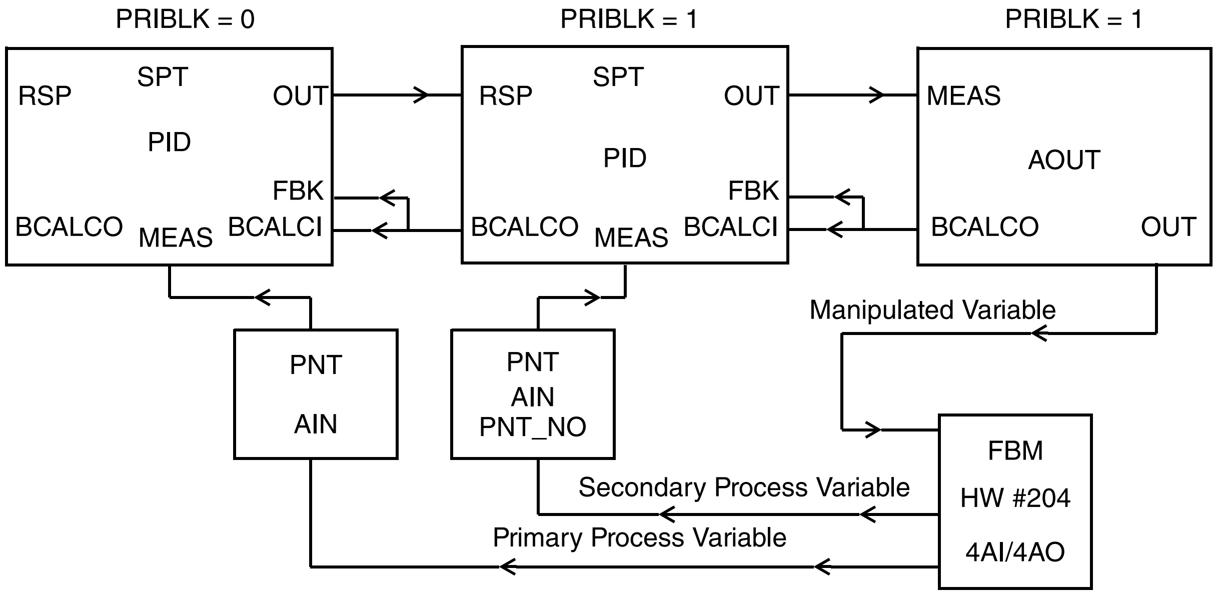


Figure 3-1. PRIBLK Cascade Standard Configuration

When an elaborate initialization scheme is required, you can connect the INITU or INITC status bits to Boolean logic.

To provide the bumpless transfer to normal, the BCALCO value of the downstream block is passed to the BCALCI value of the upstream block when the latter executes.

PRIBLK is a block parameter that allows cascade schemes to initialize properly without bumping the process during initial start-up and any return to closed-loop operation. These open loop conditions can occur at any point within the cascade.

PRIBLK extends the Explicit Initialization logic to allow cascades to be configured between compounds, between CPs, or between blocks having different block periods. Thus, there are no boundary restrictions that affect the PRIBLK connections. Also, there is no limitation to the block period assignment to each level of the cascade, other than process sampling considerations. PRIBLK is supported by the:

- PID, PIDE, PIDX, PIDXE, PIDA, RATIO, BIAS, SWCH, OUTSEL, AOUT, and AOUTR control blocks.
- AO, DO, and PIDFF FOUNDATION fieldbus blocks.

Configuring PRIBLK to a 0 initializes all blocks of the cascade in the same control processor, at the configured BPC period for the block. For example, a three-level cascade as shown in Figure 3-1containing a 30-second period primary block, a 10-second period intermediate block, and a 1-second period secondary block, results in a delay of up to 40 seconds before the primary block initializes after the secondary block, for example, switched from Manual to Auto.

Configuring PRIBLK to a 1, initializes blocks only one BPC later than the next lowest downstream block in a cascade, provided the upstream blocks are in the same control processor. In the three-level cascade shown in Figure 3-1, the primary block initializes in 2 BPCs. This is referred to as “fast initialization”, and is accomplished by forcing execution of the upstream block immediately when cascade initialization is under way, regardless of its period and phase.

PRIBLK Parameter

General

The PRIBLK parameter initializes a cascade scheme at start-up and any return to closed-loop operation. A block with PRIBLK = 1 responds to each of these operational events by performing two unique types of Explicit Initialization.

The types of Explicit Initialization action that PRIBLK enables in response to start-up and return to closed-loop operation are referred to as Unconditional Initialization and Conditional Initialization, respectively.

PRIBLK initialization is based on a tight coupling between each adjoining block of a cascade. At any point along the cascade after the first controller, a given block can be viewed as a secondary to its upstream block, and as a primary to its downstream block. The roles of primary and secondary exchange as one moves along the cascade.

The standard connections shown in Figure 3-1 maintain the control/data flow of information that allows initialization to be coordinated along the entire cascade.

Both types of Explicit Initialization progress from a secondary block to its upstream primary block. Initialization may begin at any point along the cascade and proceeds to the next primary block. The most primary block must have PRIBLK set to zero. How far the rippling proceeds depends on the type of initialization and the mode of each primary block. Thus, the direction in which initialization is performed is opposite to the forward signal path of the cascade.

The amount of time that it takes for explicit initialization to ripple between any two blocks within a cascade is dependent on the primary block's PERIOD parameter and the transit time across a station-to-station interface that may be between the two blocks. Thus, initialization occurs naturally (that is, from the FBM to the beginning of the cascade) and is governed by the configuration.

PRIBLK affects the manner in which the incoming demand signal (for example, RSP for PIDs, REMRAT for RATIO, RBIAS for BIAS, and MEAS for an AOUT or AOUTR) is handled. It also affects the behavior of the BCALCO output, so that the proper initialization value is made available to the primary block.

Start-up Initialization (Unconditional Initialization)

Unconditional Initialization occurs whenever a PRIBLK undergoes block initialization, which occurs when:

- The compound makes a transition to ON
- The controller reboots and the compound is ON
- The controller recovers from a power fail and the compound is ON
- The CIO Configurator performs an Add/Modify/Un-delete/LOADALL operation with the compound ON.

Note that in response to block initialization, the block's INITMA option asserts the Manual/Auto state of the block before unconditional initialization occurs.

A block performs unconditional initialization by forcing its OUT parameter to take on the value of its BCALCI input. The OUT value may then be subjected to output clamping, depending on the Manual/Auto mode and the configured MCLOPT option. For example, if the block initializes in Manual, then it performs clamping only if MCLOPT is configured to 1. The block always performs clamping when it initializes in Auto.

Regardless of the state of its PRIBLK option, a block always performs unconditional initialization whenever it starts up or receives, via the INITU bit of its BCALCI parameter, an unconditional initialization request from its downstream secondary. The block then requests initialization to its upstream primary block via the status bits of its BCALCO parameter (INITU), for one block period when PRIBLK = 0, or until the upstream primary block confirms initialization with an ACK status bit passed to the downstream block's set point inputs when PRIBLK = 1. Unconditional initialization continues to ripple backwards, until it terminates at the beginning of the cascade.

Since the upstream block specifically acknowledges receipt and action of the BCALCO status bit from downstream, you must not connect BCALCO to multiple connections (to more than one upstream block).

If all secondary blocks have their PRIBLK parameters configured to one, and a unconditional initialization sequence encounters either a block that is out-of-service or one in which the BCALCI connections are in error (see Security Aspects), then the unconditional initialization request remains pending at the last primary block that successfully completed initialization. When the next primary block returns to normal operation, then the pending request continues to ripple toward the beginning of the cascade. Therefore, unconditional initialization is always destined for completion and takes precedence over any operational state of a block, including Manual.

Transfer-of-Control Initialization (Conditional Initialization)

Conditional Initialization is the action taken by a secondary block when control is transferred back to its primary block; that is, a return to closed-loop operation. A transition to closed-loop operation occurs whenever a secondary block that has its PRIBLK parameter configured to 1:

- Transfers from Manual to Auto, with its set point selected to Remote
- Transfers from Local to Remote set point, with the output in Auto and Controlling
- Experiences both of the above events simultaneously
- Undergoes a return-to-normal transition from a Bad I/O status (AOUT or AOUTR only).

A secondary block with its PRIBLK option configured to 1 attempts to return to closed-loop operation by requesting conditional initialization to its primary block. While waiting for its primary block to respond, it continues controlling, using the Local set point value in SPT. While in this state, it ignores its remote set point RSP, even though the controller is switched to Remote. In addition, it switches the live back-calculated set point output BCALCO to the held SPT value. Thus, it maintains local control and can minimize any load upsets, while it waits for the primary to initialize.

The action taken by the primary block, in response to a conditional initialization request, depends on its Manual/Auto mode. If it is in Manual, it does not honor the request. It only honors the request if it is in Auto by performing the same action as described for the case of unconditional initialization. Thus, the primary block sets its output to the held SPT of the secondary block.

Transfer to closed-loop operation is completed when the secondary receives acknowledgment, via the BCALCO/BCALCI parameter, from the primary. When acknowledgment occurs, the secondary begins tracking the RSP demand signal into its local set point SPT. In addition, the BCALCO output switches from the held SPT value to the live back-calculated set point.

Therefore, transferring control to a primary block in Manual forces the secondary's set point to the Manual value. At the moment of transfer, any proportional action is transferred to the integral term of the controller.

Transferring control to a primary block in Auto is performed bumplessly from the held set point value.

On a return to closed-loop transition, the RSP connection must not have any detectable errors; that is, the status bits of RSP must indicate that the primary is not Out-of-Service, Bad, or connected with any OM scan errors. If any of these errors are detected, then the secondary block with PRIBLK set to one continues to use the held local set point. When the error condition returns to normal, the secondary block reasserts conditional initialization request to the primary, if it is still switched to Remote and on Control.

Operation of the AOUT or AOUTR block during a transfer of control is analogous to the behavior of a PID block, except that the roles of the MEAS and OUT are interchanged with those of the RSP and SPT.

A transfer from Manual to Auto, Local to Remote, or Bad to Normal sets the appropriate BCALCO status output to one. If PRIBLK is set to zero, the block resets the BCALCO status bit to zero and resumes normal forward calculation at the next block period without waiting for upstream blocks to initialize. However, an upstream block that is in Auto, and has the same period, initializes properly.

Set Point Behavior

A secondary block, with its PRIBLK set to one and its LR input set to Remote, does not track the remote set point RSP to the local set point SPT while it waits for initialization acknowledgment from its primary. Thus, the set point remains in a “pseudo” remote mode until the primary performs either unconditional or conditional initialization. This is not indicated at the display. However, the graphics portion of the Select displays indicate the different values between the RSP and SPT parameters, while the initialization is pending. The two parameters begin tracking as soon as initialization is completed.

Security Aspects

The block checks the BCALCI input for open loop status and connection errors of the downstream block. This ensures proper handling of connections across compound and station interfaces. The upstream block goes to the Hold state when the downstream block is either Bad, Out-of-Service, or Disconnected.

If the secondary with PRIBLK detects an error condition on its RSP connection while it is closed loop, the local set point holds the last good set point value until the connection returns to normal. This action prevents possible process upsets under disruptive conditions, such as, configuration modification of an upstream block, compound initialization of an upstream block, or reset of a remote station. Upon return to normal, the downstream block requests conditional initialization.

— NOTE

If a primary controller detects an error in its BCALCI connection, then it does not honor any initialization requests from its secondary block.

Initializing PID Blocks in Auto/Manual

The PID blocks honor explicit initialization requests (that is, the BCALCI initialization status bits set true) while in the Auto or Manual state, provided PRIBLK = 1 in the downstream block. The process output initializes to the old value referenced by their BCALCI input parameter. The sequence of events is as follows:

1. On the first execution cycle the block initializes and sets its OUT parameter to the value of BCALCI. However, at this point the value of BCALCI from the downstream block is left over from the last time the compound ran and thus, does not represent the actual process output.
2. The downstream block is now processed and updates its BCALCO parameter to the current process output value, setting its BCALCO initialization status bit to the PID to request explicit unconditional initialization for the next BPC.
3. The PID runs on the next cycle, updates its output to the new BCALCI value, and sets the Acknowledge bit in its output status.
4. The downstream block runs, detects the acknowledgment via its Measurement connection, and bumplessly drives its output to the existing value.

Configuring Cascade Schemes

Standard Cascade Configuration

A cascade scheme consists of two or more blocks, with at least one having PRIBLK = 1, configured in the manner depicted in Figure 3-2. Any number and consecutive combination of PID, BIAS and RATIO blocks may be configured in a cascade. The cascade should have an AOUT block at the end of the cascade. Figure 3-2 shows a typical two-loop PID cascade that illustrates the standard connections required to configure a cascade.

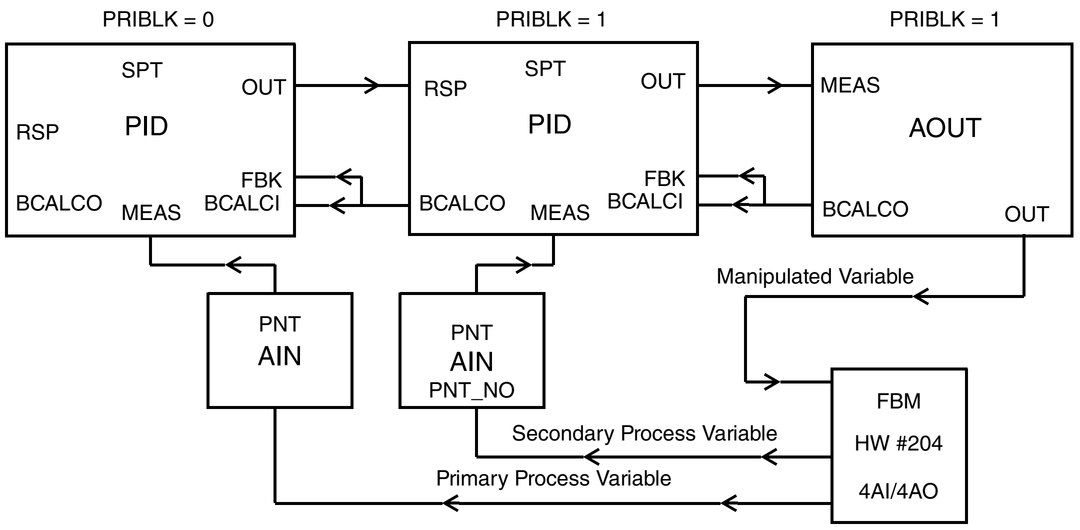


Figure 3-2. PRIBLK Cascade Standard Configuration

— **NOTE**

For examples of cascades for each of the FOUNDATION fieldbus blocks, refer to *Integrated Control Block Descriptions for FOUNDATION fieldbus Specific Control Blocks* (B0700EC).

Each secondary block must be configured with its PRIBLK parameter set to one. The first controller block at the beginning of the cascade has its PRIBLK set to zero. The AIN blocks have no PRIBLK option.

Only two standard cascade connections between a primary (p) and a secondary (s) block are mandatory:

OUT(p)	---> RSP(s)
BCALCO(s)	---> BCALCI(p)

In the case of a RATIO block, the RSP demand signal connection is replaced by OUT(p) ---> REMRAT(s); for an AOUT block, it is replaced by OUT(p) ---> MEAS(s); and for an BIAS block, it is replaced by OUT(p) ---> RBIAS(s).

You set PRIBLK = 1 in all blocks after the primary controller, including the AOUT block. You set PRIBLK = 0 in the primary controller. In each PID-type block, you connect FBK to the same signal as BCALCI.

Figure 3-3 shows the block connections for a normal cascade control scheme with feedback control. The AOUT block and the PID block interfacing to it have their PRIBLK option configured true.

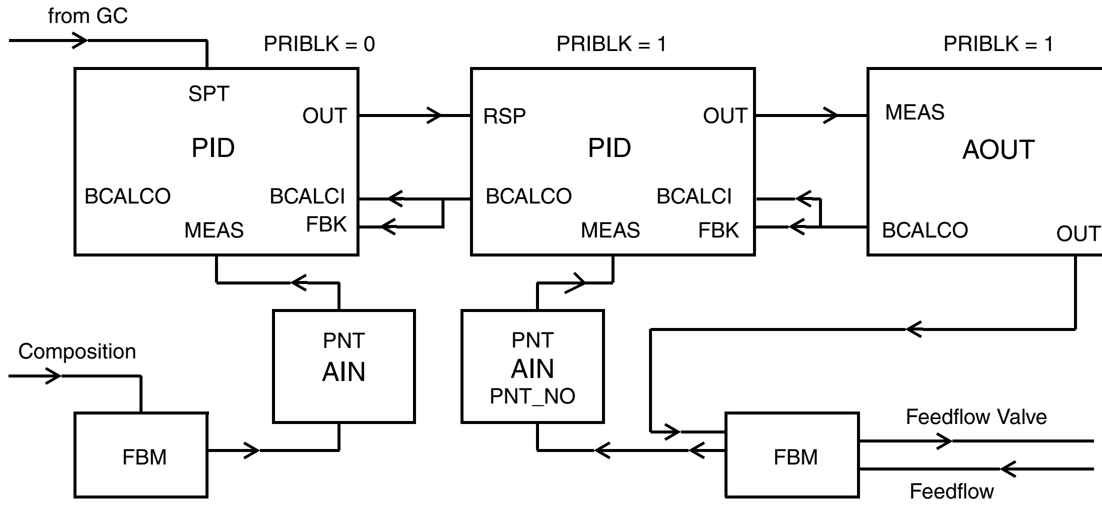


Figure 3-3. Cascade Block Strategy with Feedback Control

When the AOUT block goes back on control (from either a Manual or Bad state) it initializes and sets its BCALCO - INITC status bit.

When the downstream PID is processed, its BCALCI - INITC input status bit is set, so it performs its own initialization and back calculation and sets its BCALCO - INITC output status bit. The upstream block is processed in the same manner. The BCALCO parameter of the PID block at the top of the cascade is not connected because there is no need for further upstream initialization.

Note that the feedback input can be connected to the block's output. However, you should connect it to the same variable as BCALCI.

When PRIBLK is set to zero in all blocks, bumpless initialization occurs when all blocks start in Auto and have the same period.

Like any connectable parameter, the BCALCI and BCALCO parameters can pass data between blocks in the same compound, in different compounds, or in different stations. You should be aware that connections between different stations adds to the time required for communications.

Figure 3-4 shows how initialization occurs between blocks in compounds residing in separate stations. Communication between compounds occurs when the output (or source) change exceeds the DELTI parameter of the sink. To avoid offset or limit cycle behavior, it may be necessary to set DELTI to zero, or a very small value, in the sink block. For example, in Figure 3-5, the DELTI2 parameter in the RATIO block would be set to zero, or a small value, to avoid steady state control error.

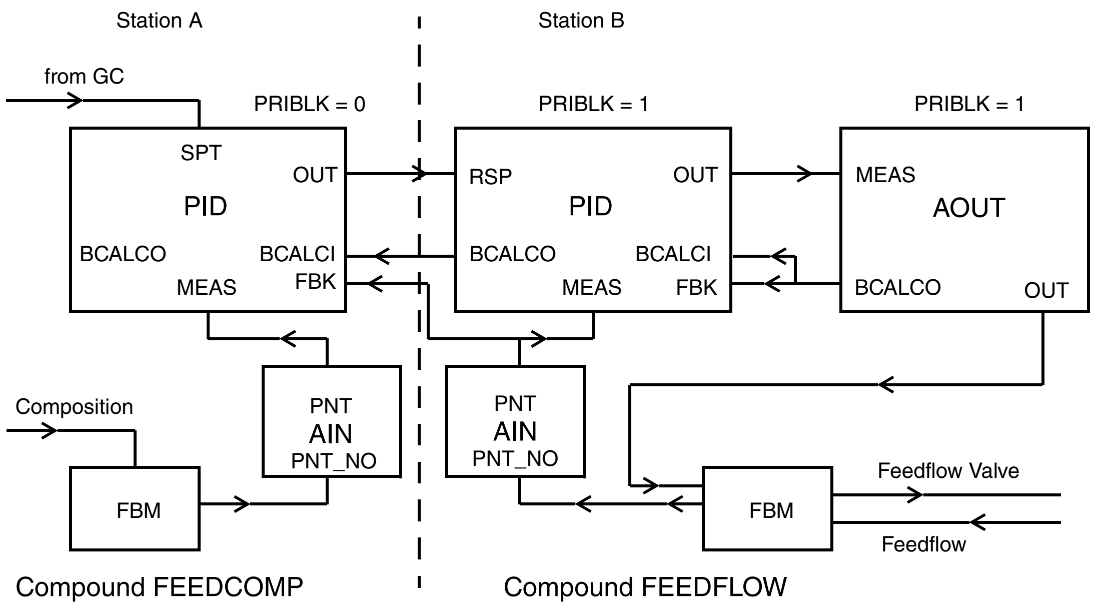


Figure 3-4. Cascade Handling Between Two Compounds Across Stations

Again, as in Figure 3-3, all the blocks except the first (primary) controller have their PRIBLK option configured true.

Figure 3-5 shows the initialization connections for a cascade control scheme with feedforward control.

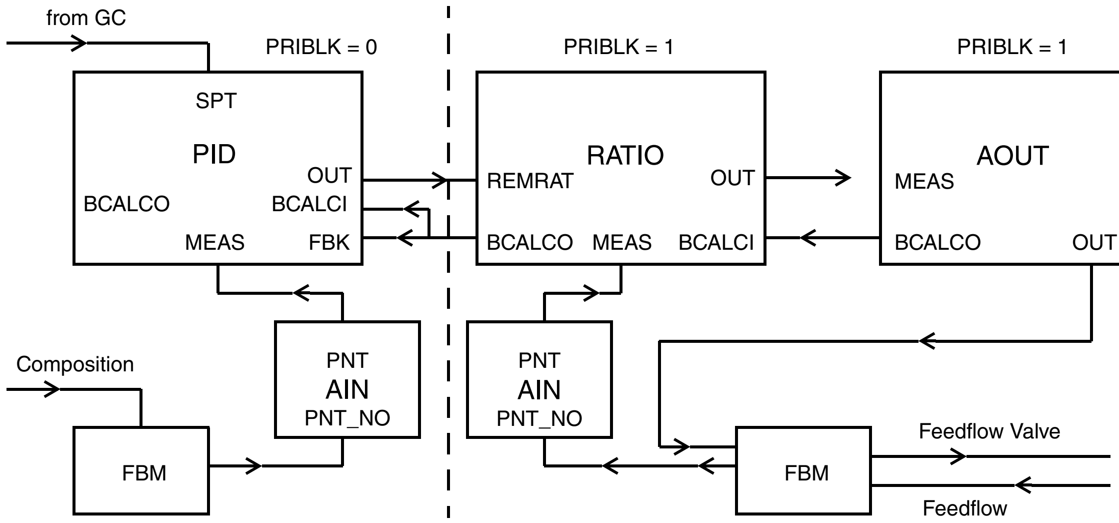


Figure 3-5. Cascade Block Strategy with Feedforward Control

External Integral Feedback

The external integral feedback input FBK of a PID controller is decoupled from the BCALCI input to provide the maximum flexibility for controlling PID integral action and anti-windup, without having to compromise initialization requirements. Under almost all conditions, FBK and BCALCI should be linked to the same source variable.

Figure 3-2 shows the recommended FBK connection for obtaining external integral feedback, or PID integral action. With this configuration, a PID controller provides integral action only as long as its integral feedback loop remains closed and tracks its own output. Therefore, in a cascade scheme, integral action at the primary block occurs only while the secondary controller's measurement tracks its remote set point.

When the integral feedback loop becomes open, the integral action of the primary block stops. This behavior is an old analog scheme to prevent integral windup of the primary controller. It is especially useful in configurations where multiple primary controllers are fanned into a single secondary controller using a SIGSEL block. When an OUTSEL block is used in conjunction with PIDA blocks, integral windup is prevented by logic signals transmitted with the BCALCO/BCALCI connections, provided LIMOPT in the PIDA blocks is set to 1 or 2.

If you desire to eliminate the external integral feedback behavior entirely, then connect FBK to the PID's output. This connection provides classic integral action, since the external integral feedback loop is closed when the controller output is not limited. This configuration is not recommended because it provides no primary integral windup protection for a non-PIDA or for a PIDA with LIMOPT = 3 when the secondary output is at its limit. Also, the primary loop cannot be tuned as tightly.

Integral windup is not a real concern when FBK and BCALCI are tied together in cascades that utilize Explicit Initialization or the PRIBLK feature, since BCALCI forces the primary block to track (in Auto) and initialize properly.

Error Handling

In general, error handling is resolved at the application level. It is accomplished through explicit user configuration of alternate control strategies based upon specific error conditions.

These error conditions are contained on a per-parameter basis within the status record for connectable parameters and shared variables.

In the status record three variables provide information with regard to the validity of the data and the validity of the connection. These are called Bad, Connect, and Out-of-Service.

The Bad status is set and reset by the block algorithm dependent upon block application.

The Out-of-Service status is set and reset by all blocks to indicate the unavailability of input data or data dependent upon unavailable inputs.

The Connect status indicates whether any problem exists in regard to the source of a connected parameter. Such problems include deleted source blocks, nonexistence of source compounds, or peer-to-peer path failures.

Implicit error handling is performed at the block level according to the following:

- If the data or the connection to the data is bad, or if the data is not updated, then certain controller type blocks (for example, PID, PTC) perform appropriate error handling. This is dependent upon the specific parameter and algorithm application.
- Most control and I/O blocks support the Propagate Error Option (PROPT). When true, this option sets the ERROR status of the primary output when the input is in error.
- I/O-type blocks also have a connectable BAD parameter for explicit error-handling purposes.

- For some block types, the error status is not propagated to the output parameters. The user can explicitly program downstream blocks to react using the AIN output parameters BAD, LOR, and HOR.
- Most DCI (Distributed Control Interface) blocks do not provide any alarm detection or reporting capabilities. For a list of blocks that support alarming, refer to Table 7-7 on page 102.

Implicit error handling is coupled with the initialization parameters defined for the I/O and Control-type blocks. The initialization status bits of BCALCO are set according to several algorithm states, one of which is the error state.

In Figure 3-3, if the flow measurement becomes bad, then the AIN block does not update its output and sets the corresponding bad status. If the LASTGV parameter in the AIN block is configured true, the last good value and the error status are picked up by the flow PID measurement.

The PID algorithm performs its error logic, which consists of setting the INITU or INITC status bit in the BCALCO parameter and transferring to a Hold or Manual state dependent upon the option. The BCALCO parameter then explicitly propagates the error notification to the upstream block.

Clamping

Clamping is performed on certain parameters dependent on the specific function. Clamping is performed as follows:

- For any parameter that has a system-specified range, the value written to the parameter is clamped by the block within the specified range. The clamped value is overwritten into the parameter and used by the algorithm.
- In general, clamping is only performed on block outputs. The actual clamped value is at range ± output span variance (OSV). The default settings are 0-100% for the range, and 2% for OSV. For the default settings, the actual clamp occurs between -2% and 102%. The maximum allowable value of OSV is 25%.
- The clamping of the OUT parameter in the following blocks is performed in identical manner: PID, PIDE, PIDX, PIDXE, PIDA, BIAS, RATIO, AOUT, and AOUTR. The algorithm is as follows:

  ```
  If LOLIM < LSCO1 - OSV, set LOLIM to LSCO1 - OSV
  Else if LOLIM > HSCO1 + OSV, set LOLIM to HSCO1 + OSV
  ```

  After LOLIM is thereby brought into range, HOLIM is prevented from being less than LOLIM:

  ```
  If HOLIM < LOLIM, set HOLIM equal to LOLIM
  ```

  Finally, HOLIM is prevented from exceeding the high end of the range:

  ```
  If HOLIM > HSCO1 + OSV, set HOLIM to HSCO1 + OSV
  ```

Scaling

Real-type data represents continuous-time process variable signals that relate to physical units and range. Real-type parameters have associated user-specified range and unit parameters.

Certain blocks like AIN, PID, RATIO, BIAS, RIN and others require scaling when the output and input engineering units differ.

Scaling factor parameters are incorporated into these blocks to enable the algorithm to account for range differences. You can enter the appropriate scaling factors during configuration. Some are user-specified, others are calculated by the block.

Ranges for associated parameters are always stated. For example, set point must have the same units and range as measurement; and feedback the same units and range as output.

Secure/Release

A secure/release mechanism is supported on a per-parameter basis. For connectable input parameters (real, Boolean, integer), the secure/release mechanism is governed by the type of connection.

If the parameter has an established linkage, it is automatically secured. If there is no linkage (connection), it is available to all users until someone secures it.

It is the responsibility of the higher level task or program to arbitrate securing any parameter at the source end. This is done automatically when parameters are opened in OM Write lists.

Outputs are secured and released by the block algorithm according to certain modes (for example, M/A, L/R, and so forth) and definitions. Some parameters are never settable, therefore they are always secured. All primary outputs are governed by the Manual-Auto function.

Some specific actions involving block input parameters include:

- ♦ Unlinked inputs are released during controller reboot to allow for user access
- ♦ Linked inputs are secured when a block initializes and can no longer be released by the user
- ♦ A Controller block which has its LOCSP option configured true, secures the LR parameter when the block initializes. LOCSW and REMSW overrides have higher precedence, but LR stays secured when LOCSW and REMSW are no longer asserted.

Inputs to ECBs are not secured. If two AOUT blocks are configured to drive the same FBM output channel, a warning message is displayed at the control configurator and/or the default display, for each of the blocks. The blocks are not prevented from executing, however, since the duplicated output channel may be an intentional part of the control strategy. (The two blocks may be in compounds which are never ON at the same time, for example.) If the duplication of the output channel is unintentional and the warning message is ignored, unpredictable results ensue, including possible process upset.

Control Block Fail-safe Strategy

Fail-safe Support is implemented in the following blocks:

- ♦ AOUT or AOUTR block
- ♦ AO (FF) block
- ♦ Analog controller blocks (PIDs, PIDFF, RATIO, BIAS)
- ♦ Propagation through SWCH blocks.

When the block processing logic detects that an FBM containing an analog output has asserted Fail-safe, an FS Boolean output is set in any AOUT (AOUTR) block connected to that FBM. Also the FS block status bit (bit 24 of BLKSTA) is set in the AOUT (AOUTR) block.

The Fail-safe status can be propagated upstream in a cascade to analog controller blocks via an FS status bit (BCALCO.FS) in the BCALCO-BCALCI connection in analog controller blocks or in a SWCH block.

The Manual Fail-safe (MANFS) option can be configured in AOUT (AOUTR) blocks and in analog controller blocks. If the MANFS option is set, the block is set to the Manual mode when Fail-safe is detected.

The FS BLKSTA bit is set in the analog controller blocks under these conditions:

- The analog controller block is in the Auto mode
- The analog controller block is in the Manual mode and its MANFS option is set.

If PRIBLK is configured, the AOUT (AOUTR) block also requests the unconditional initialization of the upstream blocks via its BCALCO output when Fail-safe is detected.

Once set, the FS BLKSTA bit and the FS Boolean output remain set in the AOUT (AOUTR) block until one of these events occurs:

- The block is in Manual, and the output parameter is changed by the you
- The block makes a transition from Manual to Auto
- PRIBLK is not configured, and the block is in Auto
- PRIBLK is configured, and the initialization acknowledgment is received from the upstream block.

Once set, the FS status bit remains set in the analog controller block until one of these events occurs:

- The block is in Manual, and the output parameter is changed by the operator
- The block is in Auto and the downstream cascade closes
- The block makes a transition from Manual to Auto.

PID Hold Constraint

A PID block may optionally be driven to the Hold state, in which the block's OUT parameter is not changed by block action and is secured against setting, by two methods:

- Explicit Hold action, produced by the true state of the HOLD parameter. The block goes into this explicit Hold state whenever HOLD is true, the block is in Auto, and parameter MBADOP = 0.
- Implicit Hold action, which depends on the value of CEOPT. When CEOPT = 0, there is no Implicit Hold action. When CEOPT = 1, the block goes into the Hold state whenever a basic input parameter has Bad, Out-of-Service, or off-scan status. When CEOPT = 2, the block goes into the Hold state whenever a basic input parameter has Bad, Out-of-Service, off-scan, or Error status. The basic input parameters are MEAS, FBK, and BCALCI. As in the case of Explicit Hold, the action requires that the block be in Auto, with MBADOP = 0.

Bad status for MEAS, when the PID block's immediate upstream connection is an AIN, AINR, or MAIN block, is in turn dependent on several factors, as follows:

- If the FBM has Bad status, the output PNT or PNT_x has Bad status, and therefore MEAS has Bad status.

- ♦ If the connected point in the FBM has Bad status, PNT or PNT_x and MEAS has Bad status.
- ♦ If the measurement at the analog input point is determined by the AIN, AINR, or MAIN block to be out-of-range, then HOR or LOR is set true, and, in some circumstances, PNT or PNT_x will have Bad status. Parameter BADOPT in the AIN, AINR, or MAIN block governs whether range violation causes Bad status, in accordance with the following rules:
  - ♦ BADOPT = 0: Neither HOR nor LOR causes Bad status.
  - ♦ BADOPT = 1: LOR causes Bad status.
  - ♦ BADOPT = 2: HOR causes Bad status.
  - ♦ BADOPT = 3: Both HOR and LOR cause Bad status.

In general, the signal values at which the input block declares the point status LOR or HOR, are approximately lower range value for the low end, and approximately upper range value for the high end, depending on the value of OSV. However, the actual determination of HOR/LOR status depends on the details of the signal conditioning in use for the point. [See *Integrated Control Block Descriptions* (B0193AX).]

For FBM types 1, 4, 201, 204 and 208 there is much greater latitude on the low end of the signal span when you use elevated-zero signal conditioning (that is, 4 to 20 mA with an SCI of 3, 5, or 7). However, there is very little headroom on the high end of the signal span, regardless of elevated-zero signal conditioning.

In the AIN, AINR, and MAIN blocks the Bad status of PNT or PNT_x is always identical to the block's BAD output parameter.

4. Block States

This chapter describes various block states, including the shutdown, define, bad, and manual/auto states, as well as the block status parameter (BLKSTA) and status indicators and events.

Block states comprise both System states and Application states. System states result from:

- ♦ Mismatching or undefined FBM/ECB identifiers
- ♦ Incorrect or out-of-range parameter values
- ♦ Off-line/On-line switching
- ♦ Any other action other than Manual/Auto switching.

The System states are:

- ♦ Shutdown
- ♦ Define
- ♦ Bad.

The Application states under block control are:

- ♦ Manual/Auto
- ♦ Local/Remote.

Shutdown State

Shutdown is the state the block exhibits when its compound is off-line, that is, it is turned off by the compound processor (via Compound On/Off). It is essentially a wait state until the compound goes back on-line.

This event causes a state transition out of the Shutdown state to one of the application states, Manual or Auto. The block initializes when it asserts this state transition.

Conversely, when the compound is turned off, it causes the block to make a transition out of its present application state back to the Shutdown state.

In this state:

- ♦ All connectable parameters are released
- ♦ The last values, or block history, is saved and maintained
- ♦ The Out-of-Service status of all value records is set true.

This action notifies other users, which have connections to this block, that the block has been taken off-line. At Shutdown, all previous history is saved.

Define State

During initialization, the block validates critical configuration parameters prior to performing a state transition out of the shutdown state. If validation is successful, the block reverts to an Application state, Manual or Auto, depending on the value in INITMA. However, if any of these critical parameters are not within their legal range, the block will avert going on-line by performing a state transition back to the Shutdown state and declaring itself Undefined.

During installation, a block will be installed and placed in the Undefined state when an input connection, specified to a source block residing in the same station, experiences any of the following errors:

- ♦ The source parameter is nonconnectable.
- ♦ The source parameter name is invalid for its block type.

Bad State

The Bad state results from I/O-related errors, for example, mismatching FBM and ECB identifiers, or bad inputs and handling, or non-operational FBMs.

This Bad status is included as a quality attribute (bad) of the output parameter of input-type blocks.

In the Bad state:

- ♦ The AOUT or AOUTR block's BCALCO initialization status bit is set
- ♦ Outputs are marked Bad and Secured
- ♦ The last driven state is retained.

When the data is validated and the block is not Bad, the outputs are reset, marked good, and the block returns to either Manual or Auto.

Bad Detection

The first checks for a bad input take place in the FBM. Part of the FBM error detection includes a "bad channel status" test that includes a check of the hardware signal to determine if that signal exceeded the range of the A/D converter (that is, the signal value was at zero or at its maximum value). Also included as part of the "bad channel status" test is a rate of change check (assuming that the ROC parameter of the associated ECB block is a non-zero value) which detects when a measurement value changes faster then the configured limit and consequently marks the input channel bad. The "bad channel status" bit that is sent with the raw data value to the analog input block (via the ECB) is an indicator of the results of these tests which includes other internal FBM diagnostic checks for erroneous raw data conversion.

The ECB receives the raw data and the channel status, and adds an "ecb_status" which is resolved at the ECB interface and indicates the availability of the FBM. "ecb_status" is set when the FBM is not available because it is either not operational, or the path to it [the Peripheral I/O (PIO) bus] is in error. These three entities, raw data, channel_status, and ecb_status are directed to an analog input block (AIN, AINR, or MAIN). AIN, AINR, and MAIN blocks handle Bad detection identically.

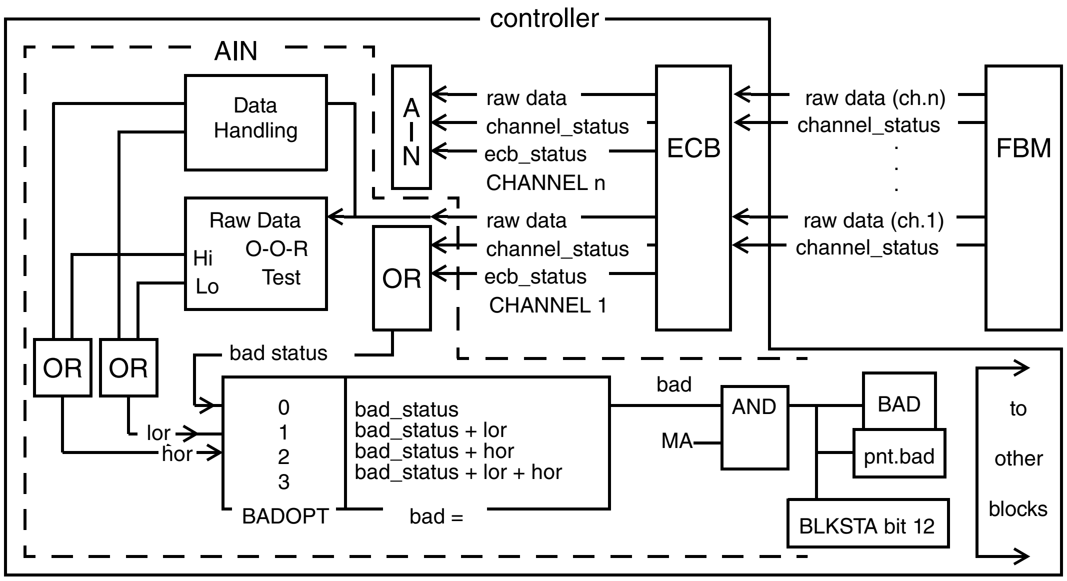


Figure 4-1. Bad Detection

The AIN or AINR block uses the logical OR of the two status bits, channel_status and ecb_status, to produce the internal signal, bad_status. If bad_status is true, the input is Bad for all values of BADOPT, and, when the block is in Auto, the BAD output parameter, the PNT.Bad status bit, and the BAD bit (bit 12) of the BLKSTA output are all set true.

The raw data goes to two functions.

The raw data goes to the data handling algorithm to produce an output value determined by the input and the configuration. If the converted value exceeds the normalized signal span specified by the HSCO, LSCO, and OSV parameters, output clamping occurs. This output clamping activates one of two signals to indicate that the output is clamped at the high, or low, end of the span.

The raw data also undergoes a raw data out-of-range test to ensure that the data, as received by the AIN or AINR block, is within the raw signal range in those cases (for example, square root extraction or table lookup linearization) where an out-of-range input might not produce a limited output. If the data is out-of-range, the block activates one of two signals to indicate that the input is outside the high, or low, end of the input range.

The results of these two functions at each end of the span are combined to yield two out-of-range signals. The signal that indicates the output is clamped at the high end, is OR'd with the signal that indicates the input was too high, to produce the internal signal "hor" (high out-of-range). Likewise, the two low indicators are OR'd, to produce the internal signal "lor" (low out-of-range).

One, or both, of these signals ("hor" or "lor") may be included with the bad_status signal (depending on the configuration of the BADOPT parameter) in the determination of a Bad input. BADOPT (Bad and Out-Of-Range option) is a nonconnectable, nonsettable, integer input that determines the conditions that, when the block is in Auto, make the BAD parameter output, the PNT.Bad status bit, and bit 12 (BAD) of the BLKSTA output, all true.

The BADOPT value ranges from 0 to 3 and map to the following conditions:

0	Bad_status
1	Bad_status or Low Out-of-range
2	Bad_status or High Out-of-range
3	Bad_status or Low Out-of-range or High Out-of-range.

When an out-of-range condition causes a Bad input condition, the block takes the following actions:

- ♦ Sets the appropriate HOR or LOR parameter.
- ♦ Sets the point Bad status, the BAD output parameter, and the BAD bit in the BLKSTA parameter.
- ♦ Activates the BAD alarm, if configured.

The default value of BADOPT is zero. The High and Low out-of-range conditions will set the BAD output when the BADOPT parameter is defaulted.

Manual/Auto States

Manual/Auto are Application states. Most blocks support the parameter MA for their primary outputs. MA is a Boolean input that controls the Manual/Auto operational state of the block's output(s). The nonsettable outputs of a block are not under MA control.

In Manual, the output is unsecured, which makes it an input or an independent variable from any external process. In Manual, the block:

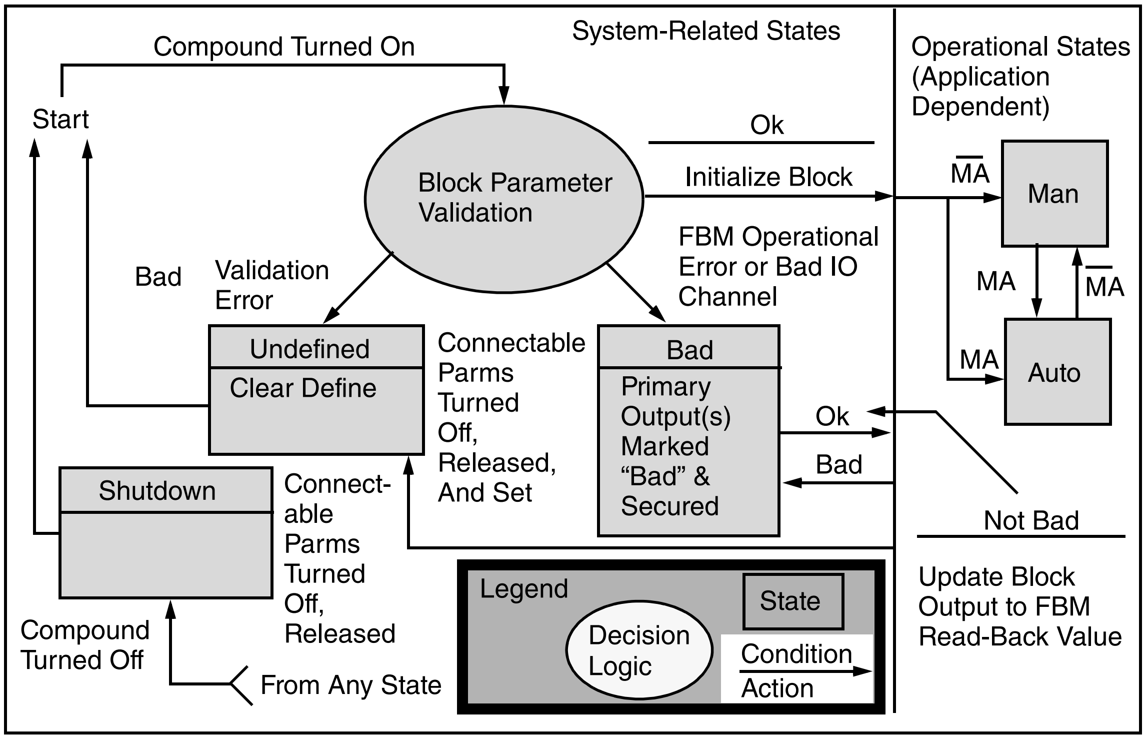
- ♦ Unsecures the settable output parameter(s)
- ♦ Retains real type output values from last values while in Auto. (Exceptions: Man clamp and Man alarm options.)
- ♦ Clears Boolean-type outputs on initial transition.

In Manual, any task or process is allowed to write to settable outputs through SET calls. The outputs become available to all users.

In Auto, the block secures its settable outputs, which makes them dependent variables that are determined by the substate of Auto (for example, Hold, Track, Control). On a transition to Manual, the output is held for reals but is cleared for Booleans. In Auto the block:

- ♦ Secures its primary settable output parameter(s)
- ♦ Updates them according to the algorithm.

The following, Figure 4-2, shows a block state transition diagram.



System-Related States
Compound Turned On
Start
Block Parameter Validation
Ok
Initialize Block
FBM Operational Error or Bad IO Channel
Bad
Validation Error
Undefined
Clear Define
Connectable Parms Turned Off, Released, And Set
Bad
Primary Output(s) Marked “Bad” & Secured
Ok
Bad
Shutdown
Connect-able Parms Turned Off, Released
Compound Turned Off
From Any State
Legend
Decision Logic
State
Condition
Action
Operational States (Application Dependent)
$\overline{\text{MA}}$
Man
MA
$\overline{\text{MA}}$
MA
Auto
Not Bad
Update Block Output to FBM Read-Back Value

Figure 4-2. Block State Transition Diagram

BLKSTA – the Block Status Parameter

Block Status (BLKSTA) is a connectable, 32-bit output that is bit mapped to indicate the block operational states at the time the block status is read.

Each individual block description in *Integrated Control Block Descriptions* (B0193AX) identifies the bits that make up the BLKSTA parameter for that block.

A list of all the bits that can be assigned to the BLKSTA parameter and their indicated block state follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31	B32

See Table 4-1 for BLKSTA bit assignments.

Table 4-1. BLKSTA-Assignable Bits

Bit Number[1]	Indicated Operational State	Boolean Connection Extension
31 (MSB)	FLB Supervisory Control Fallback State	BLKSTA.B1
30	SC Supervisor Control	BLKSTA.B2
29	SE Supervisor Enabled	BLKSTA.B3

Table 4-1. BLKSTA-Assignable Bits (Continued)

Bit Number[1]	Indicated Operational State	Boolean Connection Extension
28	HOL High Output Limit (Clamped)	BLKSTA.B4
27	LOL Low Output Limit (Clamped)	BLKSTA.B5
26	MAO Manual/Auto Override	BLKSTA.B6
25	LRO Local/Remote Override	BLKSTA.B7
24	FS Fail-safe	BLKSTA.B8
23	DSB Disabled; MO Manual Override (MDACT, DPIDA)	BLKSTA.B9
22	DSR DSR Mismatch (GDEV); DF Device Fail (MDACT); LLO Downstream Limited Low (BIAS, PID, PIDA, PIDX, PIDE, PIDXE, RATIO)	BLKSTA.B10
21	ILK Interlock; LHI Downstream Limited High (BIAS, PID, PIDA, PIDX, PIDE, PIDXE, RATIO)	BLKSTA.B11
20	WLCK Workstation Lock	BLKSTA.B12
19	SBX Executing SBX Statements (Sequence blocks); TEST Test Mode (PLB); HRQ Hold Request (MDACT); DEV Redundant MEAS input deviation limit (DPIDA); Set Point Ramp Up (PIDA)	BLKSTA.B13
18	PAUS Paused (DEP); SIM Simulation Mode (PLB); INER Input Error (OUTSEL)	BLKSTA.B14
17	TRIP Tripped (MON); FTN Feedforward Tune Active (PIDA, DPIDA); LM2 Limit Switch 2 On (GDEV); RAMP Ramp Option (RAMP)	BLKSTA.B15
16	ACT Active/Inactive (Sequence blocks); FTNI Feedforward Tune Inactive (PIDA, DPIDA); REPT Repeat Option (RAMP); LM1 Limit Switch 1 On (GDEV)	BLKSTA.B16
15	ON Compound On	BLKSTA.B17
14	UDEF Undefined	BLKSTA.B18
13	(Unused)	BLKSTA.B19
12	BAD Bad I/O (I/O Blocks only - block in BAD state)	BLKSTA.B20
11	MA Manual(= false)/Auto(= true)	BLKSTA.B21
10	LR Local(= false)/Remote(= true)	BLKSTA.B22
9	STRK Set Point Tracking; FBMR Redundant FBM Fail (AINR, AOUTR)	BLKSTA.B23
8	FBM FBM Failure	BLKSTA.B24
7	HLD Holding	BLKSTA.B25
6	TRK Tracking	BLKSTA.B26
5	CTL Controlling; ACC Accumulating (ACCUM); RMP Ramp (RAMP)	BLKSTA.B27

Table 4-1. BLKSTA-Assignable Bits (Continued)

Bit Number[1]	Indicated Operational State	Boolean Connection Extension
4	FOL Follow; ASP Alternate Set Point (MDACT); DEV DEV Redundant I/O Deviation (DPIDA)	BLKSTA.B28
3	(Unused)	BLKSTA.B29
2	PTN Pre-Tune Mode (EXACT tuning algorithm)	BLKSTA.B30
1	STN Self-Tune Mode (EXACT tuning algorithm); BADR Bad Redundant I/O (AINR, AOUTR)	BLKSTA.B31
0	MTN Manual Tune Mode (EXACT tuning algorithm); SEL Redundant I/O Selected (AINR)	BLKSTA.B32

1. Bit 0 is the least significant bit (starting from the right).

BLKSTA provides bit-mapped status indication of bad I/O, Manual/Auto, and other block states.

1. HOL and LOL are mutually exclusive, set false on a transition to manual, and updated when the block output is clamped.
2. ON is true whenever the compound that includes this block has its ON parameter set true.
3. UDEF indicates that, during initialization, the block was unable to validate critical configuration parameters, and returned to the shutdown state.
4. BAD is set true in an I/O block when the block detects a bad input channel or, depending on the BADOPT configuration, data that is out-of-range.
5. MA indicates the Auto/Manual state of the block (true = 1 = Auto; false = 0 = Manual).
6. LR and STRK are mutually exclusive, initialized to the appropriate set point state, and updated when state transitions occur.
7. FBM is set true when the FBM is found to have bad status.
8. HLD, TRK, CTL, and FOL are mutually exclusive substates of the Auto state, initialized to the appropriate block state and updated when block state transitions occur.
9. PTN, STN, and MTN are mutually exclusive, initialized to the appropriate tune state and updated when tune state transitions occur.

The BLKSTA parameter is set to zero when a block initializes.

BLKSTA can be accessed by application programs. It can also be accessed by other blocks having long (32 bits) integer inputs that can be handled as packed Booleans (for example, the MCIN block when the IOMOPT option is configured false, or the CALC block).

Table 4-1 indicates, in alphabetical order, the literal names assigned to the individual bits of BLKSTA. These names may be used in Boolean connection extensions (for example, BLKSTA.LOL).

Block Status

This section discusses two aspects of block status: indicators and events.

Control and I/O blocks are equipped with status indicators to:

- ♦ Indicate the current block status
- ♦ Indicate the nature of the event that caused that status
- ♦ Provide inputs to the control scheme that may initiate an appropriate response to that event.

Status Indicators

Three types of status indicators that Control and I/O blocks use are:

1. The BAD parameter – a dedicated boolean output of the AIN, AINR, AOUT, AOUTR, CIN, CINR, COUT, COUTR, EVENT, MAIN, MCIN, MCOUT, MDACT, MTR, MOVLV, and VLV blocks. Control blocks typically use BAD as a source parameter for determining control strategy.
2. Bits from a parameter's value record status word. The only bits available as named sources in Control Configurator connections are:

PARM..B	(Bad)
PARM..E	(Error)
PARM..O	(Out-of-Service)
PARM..D	(Disconnected)

(PARM represents the name of any parameter.) See "Connections to Source Status" on page 53.
In addition, user-written software applications may access any of the 16 bits of the value record status word. These bits are defined as follows:

Table 4-2. Parameter Status Bits

Bit	Status	Signal Value
0-3	TYPE	
4	Used only in `setval` call	
5-7	OM	
8	BAD	1= Bad I/O
9	SECURE/FS	1= Secure/Fail-safe
10	INITC/ACK	1= InitC Request/Acknowledge
11	OOS	1= Out-Of-Service
12	SHADOW	1= Shadow Parameter
13	LHI	1= Limited High
14	LLO	1= Limited Low
15	ERROR/INITU	1= Error/InitU Request

3. Bits from the 32-bit packed long status parameters, Block Status (BLKSTA) and Alarm Status (ALMSTA).

The following paragraphs provide some examples of the events that activate some of these indicators.

1. BAD parameter – Set TRUE when:

   The FBM has failed, or the input point has bad status.

2. Parameter Value Record Status Bit – PARM..Bad

   Set TRUE when:

   a. An input block detects a bad I/O channel.
   b. An AIN, AINR, or MAIN block detects an out-of-range value if BADOPT is configured to respond to it.
   c. An MCIN block receives an invalid BCD input.

   Set FALSE when:

   A compound owning an input/output parameter is turned ON.

3. Parameter Value Record Status Bit – PARM..Oos (Out-of-Service)

   Set TRUE when:

   a. A compound owning the input/output parameter is turned OFF.
   b. The FBM is out-of-service.
      Set FALSE when:
      A compound owning an input/output parameter is turned ON.

4. BLKSTA Bit 8 – FBM Fail
   Set TRUE when:

   a. The block loses FBM communications or the FBM has failed.

5. BLKSTA Bit 12 – Bad I/O

   Set TRUE when:

   a. An input block detects a bad I/O channel.
   b. An MCIN block receives an invalid BCD input.

Status Events

Among the events that activate these indicators are:

- Loss of FBM communications (permanent or temporary)
- Compound turns OFF
- Compound turns ON with, or without, errors
- Block initializes with fatal validation errors
- An I/O block has an unresolved ECB linkage
- An I/O block receives a Bad ECB status
- An I/O block receives a Bad channel status
- An I/O block receives an out-of-range value
- Block has a source connection resolved

- Block has a source connection deleted
- Block installation
- Block modification with, or without, specified linkage
- Download with resolved, or unresolved, linkage.

The following provide examples of the block indications shown at some of these events.

1. Loss of FBM communications.
   a. The I/O blocks connected to the lost FBM set their BAD parameter true.
   b. The I/O blocks connected to the lost FBM set their BLKSTA Bit 8 (FBM Fail) true.
   c. The OOS bits in the input/output parameters of any connected blocks are set true.
   d. The output I/O blocks connected to the lost FBM set the BCALCO initialization status bit true.
2. Loss of FBM communications – temporary (for example, PIO bus switch).
   a. The I/O blocks connected to the lost FBM set their BLKSTA Bit 8 (FBM Fail) true.
   b. The OOS bits in the input/output parameters of any connected blocks are set true.
   c. The output I/O blocks connected to the lost FBM set the initialization status bit of BCALCO true.
3. Compound turned OFF.
   The OOS bits in the input/output parameter records of each block in the compound are set true.
4. Compound turns ON (no block errors).
   a. The OOS bits in the input/output parameter record of each block in the compound are reset.
   b. The BAD bit in each input/output parameter record is reset.
5. Compound ON and I/O block has a bad channel status.
   BLKSTA Bit 12 (Bad I/O) is set.
6. Compound ON and AIN, AINR, or MAIN block receives out-of-range data.
   BLKSTA Bit 12 (Bad I/O) is set if BADOPT configured correctly.
7. Source connection broken – for example, source deleted, source resides in a nonexistent compound, or there is a peer-to-peer path failure.
   The OOS bit in the sink parameter record is set.
8. Source compound turned OFF.
   The OOS bit in the sink parameter record is set.
9. MCIN block receives an invalid BCD input.
   Bit 12 (Bad I/O) in the block status parameter BLKSTA is set.

5. Connections

This chapter describes various aspects of process connections, including shared variables, linkage syntax, boolean and packed boolean connection extensions, default values for integer and real connections, and mixed data types.

Process connections between blocks, or blocks and shared variables, are long-term secured connections that are established during control configuration.

A block output to an FBM point is not a secured connection. You must confirm that no more than one output block is attached to an FBM point at a time.

A local “block-to-block” parameter linkage is a direct connection from the source parameter to the receiver in the local database and does not involve the Object Manager.

A remote connection between two parameters in different stations is established by the Object Manager, which maintains a change-driven read connection from the source to the receiver.

Linkages from blocks in the compound database to application programs is through shared variables.

Short-term connectionless access from application programs is established through GET/SET calls. Access security is determined at the host or source by setting the Secure flag in the shared variable status (GET/SET calls).

GET calls to any parameter in the database, including local algorithm variables, are allowed.

SET calls are monitored for validity of data type, range, setability attribute, and secured/ released status.

If necessary, the data value may undergo algorithm validation, for example, clamping.

For more information on GET/SET calls, refer to *Object Manager Calls* (B0193BC).

Shared Variables

A shared variable acts as a unidirectional linkage between an application and the control database.

By linking a shared variable to a block input, the configurator can establish a long-term secured connection between a remote application program and the compound processor database.

The actual connection is made by the Object Manager which establishes a change-driven read connection to the block input from the application program via the shared variable.

The application program sets and resets the secure, bad, and on/off flags in the shared variable status.

Linkage Syntax

The general format of linkage specifications entered during control configuration is, as noted in "Blocks" on page 11:

```
Compound:Block.Parameter (Cname:Bname.Pname)
```

or, if internal to one compound or one block:

```
:Block.Parameter (:Bname.Pname)
```

Boolean Connection Extensions

If the sink is of Boolean type, then the format:

```
Cname:Bname.Pname.Extension
```

may be used to refresh the sink with a wide variety of logical functions of bits within the source parameter. The source parameter may be of any data type. Both local and peer-to-peer connections of this type may be made. Although certain restrictions apply based on the type of the source, the following functions may be requested in the general case, by appropriate "extension" syntaxes:

- ♦ Extract a specific bit from the source and copy it to the sink.
- ♦ Bitwise AND the source with a specific mask and copy the result to the sink.
- ♦ Bitwise XOR the source with a specific mask and copy the result to the sink.
- ♦ Perform the AND or XOR on only the high order integer of a Long or Packed Long source.
- ♦ Clear specific bits of the sink.
- ♦ Copy specific bits from the status word of the source value record.
- ♦ Copy the logical OR of various status bits from the source status word.
- ♦ Perform similar actions with named bits of source parameters BLKSTA, ALMSTA, or INHSTA.
- ♦ Invert the extracted result before using it.
- ♦ In connection with any of these options, specify a default/fallback value for the sink in the event the connection is unresolved or the source is BAD or OOS. When the block is installed, the sink value will be initialized to this default (0/1) value.

— NOTE

By connecting the boolean input to itself and specifying a default value, you can create a boolean constant of 0 or 1 that cannot be changed without reconfiguring the connection. In addition, this feature may be used to pre-configure deterministic fallback states for any boolean input that is connected peer-to-peer to a source parameter in a different station.

If you enter invalid extension information, the connection will be marked Unresolved, the block will be set Undefined, and the OM Scan status of the input will be set to 0.

The general format of connections to Boolean inputs is one of the following:

`Cname.Pname.Extension`	(Connection to a compound parameter.)
`Cname:Bname.Pname.Extension`	(Connection to a parameter in a different compound.)
`:Bname.Pname.Extension`	(Connection to a parameter in another block of the same compound or to another parameter of the same block, or to itself.)

The general format of the Extension field in the above is:

```
   [ {{0,1} [.~], ~, [~]Bitmask, [~]{A,X}[H]mask,  [.][~] symbol}]
```

where:

[ ] = optional

{,} = select one

The syntax of the Extension field is best understood by noting the specific examples in the following paragraphs. These examples are subdivided into the categories Connections to Source Data, Connections to Source Status, and Connections to Status Parameters.

NOTE

All user documentation refers to the status bits as `parameter.<name of status bit>`. That is, a reference to a status bit does not use the boolean extension syntax.

Connections to Source Data

The optional characters 0 or 1. at the beginning are used to specify an initial default and fallback value for the sink. They may be used alone or in conjunction with other fields. When used alone, the terminal period is deleted.

Examples:

`Cname:Bname.Pname.1`	Set default result = 1. If connection broken, set result = 1. Else set result = source. The tilde (~) is used to invert the Boolean result.

Examples:

`Cname:Bname.Pname.~`	If source = 0, set result = 1. If source non-zero, set result = 0.
`Cname:Bname.Pname.1.~`	Set default result = 1. If connection broken, set result =1. Else if source = 0, set result = 1, and if source is non-zero, set result = 0.

The selected option “Bbit” causes the result to be set equal to the specified bit of the source, where B1 is the high-order bit. The “bit” specification indicates which bit is selected. There is no space between “B” and “bit”. “Bit” must be a 1-or 2-digit decimal number which is in range for the data type of the source (that is, no higher than 16 if the source is an Integer or Packed Boolean, and no higher than 32 if it is a Long or Packed Long).

The only allowable source types are Integer, Long, Packed Boolean, or Packed Long.

The result is subject to optional inversion.

Examples:

`Cname:Bname.Pname.B12`	Set result = Boolean 12 of source.
`Cname:Bname.Pname.0.B12`	Set default result = 0. If connection broken, set result = 0. Else set result = Boolean 12 of source.
`Cname:Bname.Pname.~B12`	Set result = inverse of source Boolean 12.
`Cname:Bname.Pname.0.~B12`	Set default result = 0. If connection broken, set result = 0. Else set result = inverse of source Boolean 12.

The selected option "Amask" or "Xmask", where mask is a hexadecimal mask of 1 to 4 hexadecimal digits, causes the sink to be set to the bitwise AND of the source value with the mask, or the bitwise XOR of the source value with the mask, respectively. There is no space between the "A" or "X" and the mask. The source must be of type Integer, Long, Packed Boolean, or Packed Long. If the source is Long or Packed Long, the mask is applied to the low-order 16 bits. The result is subject to optional inversion.

Examples:

`Cname:Bname.Pname.A30A0`	Bitwise AND the low-order 16 bits of source with the hexadecimal mask 30A0; if the result is non-zero, set the sink to 1, else set the sink to 0.
`Cname:Bname.Pname.~A30A0`	Invert the result of the previous example, and set the sink accordingly.
`Cname:Bname.Pname.X30A0`	Bitwise XOR the low-order 16 bits of source with the hexa-decimal mask 30A0; if the result is non-zero, set the sink to 1, else set the sink to 0.
`Cname:Bname.Pname.~X30A0`	Invert the result of the previous example, and set the sink accordingly.

The option "H", when it follows "A" or "X", indicates that the specified operation is to be performed on the high-order 16 bits of the source value. The only valid source types are Long or Packed Long.

Examples:

`Cname:Bname.Pname.~XH30A0`	Bitwise XOR the high-order 16 bits of the source with hexadecimal mask 30A0. If the result is non-zero, set the result = 0, else set the result = 1.
`Cname:Bname.Pname.1.~XH30A0`	Set the default result = 1. If the connection is broken, set the result = 1. Else set the result as in the previous example.

Connections to Source Status

You may connect to certain bits within the status of the source parameter by use of the syntax:

```
    [Cname]:Bname.Pname.[{0,1}.].[~]symbol
```

(Pname cannot be BLKSTA, ALMSTA, or INHSTA when using this format.)

"Symbol" is a 1- to 4-letter field which specifies various bits within the status field of the source parameter. The sink is set to the logical OR of the specified bits. The meaning of the letters in Symbol is as follows:

B	1 = Bad I/O
O	1 = Out-of-Service
D	1 = OM Off-Scan ("Disconnected")
E	1 = Propagated Error

The results may be optionally inverted.

Examples:

`Cname:Bname.Pname..B`	Sink = Bad status bit of source.
`Cname:Bname.Pname..~B`	Sink = Inverse of Bad status bit of source.
`Cname:Bname.Pname..BOD`	Sink = Logical OR of Bad, OOS, and OM Off-Scan status bits of source.
`Cname:Bname.Pname..~BOD`	Sink = Inverted result of previous example.

Connections to Status Parameters

You may connect to any bits, or combinations of bits, within the status parameters BLKSTA, ALMSTA, and INHSTA by using the syntax:

```
    [Cname]:Bname.{BLKSTA, ALMSTA, INHSTA}.[{0,1}.]
    {[~]Bbit,[~]{A,X}[H]mask, [~]symbol}
```

The meanings of B, A, X, and H are as already described. "Symbol", in this format, refers to one of the 2- to 4-letter symbolic names assigned to the bits within the status parameters.

Examples:

`Cname:Bname.BLKSTA.BAD`	Result = Value of the BAD bit within the source BLKSTA.
`Cname:Bname.BLKSTA.~BAD`	Result = Invert of the previous example.
`Cname:Bname.ALMSTA.LDA`	Result = Value of the LDA bit within the source ALMSTA.
`Cname:Bname.ALMSTA.~LDA`	Result = Invert of the previous example.
`Cname:Bname.INHSTA.HOA`	Result = Value of the HOA bit within the source INHSTA.
`Cname:Bname.INHSTA.~HOA`	Result = Invert of the previous example.

`Cname:Bname.BLKSTA.B12`	Result = Boolean B12 of BLKSTA.
`Cname:Bname.BLK-STA.AH30A0`	Result = Bitwise AND of the high-order 16 bits of BLKSTA with hexadecimal 30A0. (If the result is nonzero, set sink = 1. Else set sink = 0.)

Table 5-1. Block Status Symbols

Symbol	Meaning	Bit	Bbit
ACC	1 = Accumulating	Bit 5	B27
ACT	1 = Active	Bit 16	B16
ASP	1 = Alternate Set Point	Bit 4	B28
BAD	1 = Bad I/O	Bit 12	B20
BADR	1 = Bad Redundant I/O	Bit 1	B31
CTL	1 = Control	Bit 5	B27
DEV	1 = Redundant I/O Deviation	Bit 4	B28
DF	1 = Device Fail	Bit 22	B10
DSB	1 = Disabled	Bit 23	B9
DSR	1 = DSR Mismatch	Bit 22	B10
FBM	1 = FBM Failure	Bit 8	B24
FBMR	1 = Redundant FBM Fail	Bit 9	B23
FLB	1 = Supervisory Control Fallback	Bit 31	B1
FOL	1 = Follow	Bit 4	B28
FS	1 = Fail-safe output	Bit 24	B8
FTN	1 = Feedforward Tune Active	Bit 17	B15
FTNI	1 = Feedforward Tune Inactive	Bit 16	B16
HLD	1 = Hold	Bit 7	B25
HOL	1 = Hi Output Limit	Bit 28	B4
HRQ	1 = Hold Request	Bit 19	B13
ILK	1 = Interlocked	Bit 21	B11
INER	1 = Input Error	Bit 18	B14
LM1	1 = Limit Switch 1 On	Bit 16	B16
LM2	1 = Limit Switch 2 On	Bit 17	B15
LOL	1 = Lo Output Limit	Bit 27	B5
LR	1 = Remote, 0= Local	Bit 10	B22
LRO	1 = LR Override	Bit 25	B7
MA	1 = Auto, 0= Manual	Bit 11	B21
MAO	1 = MA Override	Bit 26	B6
MO	1 = Manual Override	Bit 23	B9
MTN	1 = Manual Tune	Bit 0	B32
ON	1 = On	Bit 15	B17

Table 5-1. Block Status Symbols (Continued)

Symbol	Meaning	Bit	Bbit
PAUS	1 = Paused	Bit 18	B14
PTN	1 = Pre-Tune	Bit 2	B30
RAMP	1 = Ramp Option	Bit 17	B15
RED	1 = Redundant Inputs	Bit 21	B11
REPT	1 = Repeat Option	Bit 16	B16
RMP	1 = Ramp	Bit 5	B27
SBX	1 = Exec SBX statements	Bit 19	B13
SC	1 = Supervisor Control	Bit 30	B2
SE	1 = Supervisor Enabled	Bit 29	B3
SEL	1 = Redundant I/O Selected	Bit 0	B32
SIM	1 = PLB Simulation Mode	Bit 18	B14
SPDN	1 = Set Point Ramp Down	Bit 19	B13
SPUP	1 = Set Point Ramp Up	Bit 18	B14
STN	1 = Self Tune	Bit 1	B31
STRK	1 = Set Point Track	Bit 9	B23
TEST	1 = PLB Test Mode	Bit 19	B13
TRIP	1 = Tripped	Bit 17	B15
TRK	1 = Track	Bit 6	B26
UDEF	1 = Undefined	Bit 14	B18
WLCK	1 = Workstation Lock	Bit 20	B12

Example:

The Manual/Auto state of a block is presented in Bit 11 of BLKSTA, where Bit 0 is the low-order bit. Boolean connection extensions refer to this bit as either BLKSTA.MA or BLKSTA.B21. (Note that Bit 31, the high-order bit, is referred to in Boolean connection extensions as B1.) When you use the Amask or Xmask format, you must refer to the third column ("Bit") and when you use the Bbit format, you must refer to the fourth column ("Bbit"). In order to prevent parameter BLKSTA from becoming unnecessarily long, certain bits are used in a block-dependent fashion, thereby necessitating "alias" names for some bit positions. See, for example, ACT and CTL.

Table 5-2. Alarm Status Symbols

Symbol	Meaning	Bit	Bbit
BAD	1 = I/O Bad Alarm	Bit 22	B10
HDA	1 = Hi Deviation Alarm	Bit 21	B11
HHA	1 = Hi Hi Abs Alarm	Bit 25	B7
HMA	1 = Hi Meas Alarm	Bit 17	B15
HOA	1 = Hi Output Alarm	Bit 19	B13

Table 5-2. Alarm Status Symbols (Continued)

Symbol	Meaning	Bit	Bbit
INH	1 = Alarms Inhibited	Bit 29	B3
LDA	1 = Lo Deviation Alarm	Bit 20	B12
LLA	1 = Lo Lo Abs Alarm	Bit 24	B8
LMA	1 = Lo Meas Alarm	Bit 16	B16
LOA	1 = Lo Output Alarm	Bit 18	B14
OPER	1 = Operational Error Alarm	Bit 27	B5
OOR	1 = Out of Range Alarm	Bit 28	B4
PNT1	1 = Point 1 Alarm	Bit 15	B17
PNT2	1 = Point 2 Alarm	Bit 14	B18
PNT3	1 = Point 3 Alarm	Bit 13	B19
PNT4	1 = Point 4 Alarm	Bit 12	B20
PNT5	1 = Point 5 Alarm	Bit 11	B21
PNT6	1 = Point 6 Alarm	Bit 10	B22
PNT7	1 = Point 7 Alarm	Bit 9	B23
PNT8	1 = Point 8 Alarm	Bit 8	B24
PTRG	1 = Pre-Target Alarm	Bit 19	B13
ROC	1 = Rate of Change Alarm	Bit 23	B9
STA	1 = State Alarm	Bit 26	B6
TARG	1 = Target Alarm	Bit 25	B7
TRIP	1 = Trip Alarm	Bit 31	B1
UNAK	1 = Unacknowledged	Bit 30	B2

The boolean extension symbols for the inhibit status parameter INHSTA are the same as those for the ALMSTA parameter, except for the UNAK symbol.

Packed Boolean Connection Extensions

If the sink is a Packed Boolean, then the syntax:

`Cname:Bname.Pname.mask` [where *mask* is a four-digit (xxxx) hexadecimal number]

may be used to set and reset specific bits of the sink.

If Pname is a boolean source parameter with a value of FALSE when the connection is refreshed, then the bits of the sink corresponding to the 1-bits of the mask are cleared.

All other bits of the sink are unchanged. If Pname is TRUE at this time, the bits of the sink corresponding to the 1-bits of the mask are set, and all other sink bits are unchanged.

If Pname is an integer, long integer, packed boolean or packed long, the bits of the low-order 16 bits of the source which correspond to 1-bits of the mask are copied into the sink. All other bits of the sink are unchanged.

If Pname is a short integer, those bits of the source which correspond to 1-bits in the low-order byte of the mask are copied to the sink. All other bits of the sink are unchanged.

If Pname is a real, then Pname is converted to an unsigned integer before the mask is applied.

You cannot set any sink bits which are specified in the mask for setting, clearing, or copying.

All other bits of the sink are settable.

Defaults for Integer and Real Connections

The section "Boolean Connection Extensions" on page 50 explains how default values may be specified for inputs of boolean type. It is also possible to specify such default values for inputs of integer or real type by configuring the connection according to the following format:

```
Cname:Bname.Pname.n
```

where *n* is the desired default value.

If the sink is of integer type, n is an integer.

Example:

```
EVAPORATOR:F1200.HOLIM.7000
```

If the sink is of real type, then n is a real value in floating point or scientific notation.

Examples:

```
EVAPORATOR.F1200.HOLIM.124.5
EVAPORATOR.F1200.LOLIM.-3.2E-2
```

The default value specified will only be used until the source value is available (that is, the connection is resolved and the source status is not BAD or OOS). When that happens, the source value will be copied into the input parameter, and the default value will be lost. The connection is secured while the specified default is in use.

If the input parameter is connected to itself, however, the default value specified in the connection will be preserved until the parameter is reconfigured. You may use this method of specifying defaults to override the standard default for any parameter, as stated in the parameter tables.

Mixed Data Types

Local connections are input parameter connections to source parameters in the same controller station. You can connect an input parameter to a source parameter of any data type in any block in the same controller station.

The controller converts the data from the source format into the sink format when the data is copied into the sink connection. This is done each block execution cycle prior to processing the block algorithm.

In most cases, data conversions conform to the standard "C" programming language conventions used in assignment statements containing mixed data types.

For the cases shown in Table 5-3, however, the converted value is clamped at the specified maximum positive or minimum negative value. Table 5-4 summarizes the results of the conversion algorithms.

Table 5-3. Data Conversion Clamping

Sink Value	Source Value	Maximum	Minimum
Short Integer	Integer	127	-128
Short Integer	Long integer	127	-128
Short Integer	Real value	127	-128
Integer	Long integer	32767	-32768
Integer	Real value	32767	-32768
Long Integer	Real value	2147483647	-2147483648

Table 5-4. Data Conversion Results

Sink Value	Source Value	Results
Boolean	Short integer, integer, long integer, real, packed boolean	If source not = 0, result = 1 Else result = 0
Short Integer	Boolean, packed boolean, packed long	Low-order byte of source
	Integer, long integer	Clamped source
	Real	Converted clamped source
Integer	Boolean	Unsigned low-order byte of source
	Short integer	Extended low-order byte of source
	Long integer	Clamped source
	Real	Converted clamped source
	Packed boolean	Source
	Packed long	Unsigned low-order integer of source
Long Integer	Boolean	Unsigned low-order byte of source
	Short integer	Extended low-order byte of source
	Packed boolean	Unsigned low-order integer of source
	Integer	Extended low-order integer of source
	Real	Converted clamped source
	Packed long	Source
Real	Boolean, short integer, integer	Converted source
	Long integer, packed long	Converted truncated high-order source
	Packed boolean	Converted unsigned integer of source

Table 5-4. Data Conversion Results (Continued)

Sink Value	Source Value	Results
Packed Boolean	Boolean	If source = 0, mask bits cleared in sink If source = 1, mask bits set in sink (unmasked bits are unchanged)
	Short integer	Masked unsigned low-order byte of source (unmasked bits are unchanged)
	Integer, long integer, packed boolean, packed long	Masked unsigned low-order integer of source, (unmasked bits are unchanged)
	Real	Masked converted clamped integer of source (unmasked bits are unchanged)
Packed Long	Boolean	If source = 0, clear all sink bits If source = 1, set all sink bits
	Short integer	Unsigned low-order byte of source
	Integer, packed boolean	Unsigned low-order integer of source
	Real	Converted clamped long of source
	Long integer	Source

Peer-to-peer connections are input parameter connections to source parameters in different controller stations.

Peer-to-peer connections support a mixture of the following data types. The same conversions specified for the mixed local connections in Table 5-4 are performed for the combinations of mixed peer-to-peer connections listed in Table 5-5.

Table 5-5. Peer-to-Peer Connections

Source Value	Sink Value
Long Integer	Integer, short integer, boolean, packed boolean, packed long
Integer	Long integer, short integer, boolean, packed boolean, packed long
Short Integer	Long integer, integer, boolean, packed long

Peer-to-peer connections do **not** support a mixture of real and integer data types, or a mixed connection to a boolean source.

6. Block Processing

This chapter describes the compound processor's actions in the block processing cycle (BPC). It describes scanning, I/O blocks, and input and output signal conditioning.

The compound processor is a task scheduled by the operating system to run every block processing cycle (BPC).

The *BPC* is the smallest resolution of time in which the compound processor can be scheduled to run. It presently defaults to 0.5 second. Periods are resolved in multiples of the BPC.

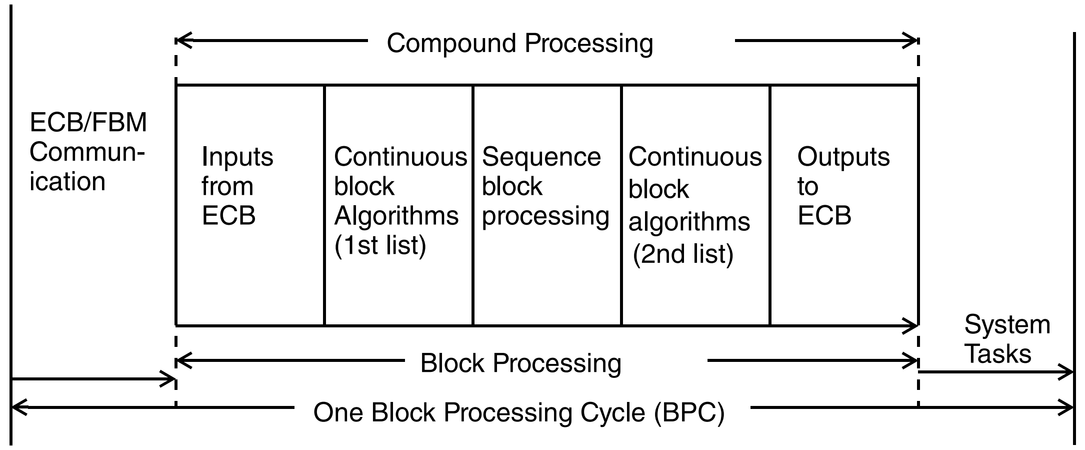

Compound Processing
ECB/FBM Commun-ication
Inputs from ECB
Continuous block Algorithms (1st list)
Sequence block processing
Continuous block algorithms (2nd list)
Outputs to ECB
System Tasks
Block Processing
One Block Processing Cycle (BPC)

Figure 6-1. The Block Processing Cycle

Within each BPC, the compound processor task processes the blocks in the following order:

1. The first list of continuous blocks.
2. Monitor blocks and timer blocks.
3. Exception blocks.
4. Dependent and independent blocks.
5. The second list of continuous blocks.

See the diagram above for a BPC representation.

Ladder Logic processing takes place within the Fieldbus Modules.

After the first list of continuous blocks is processed, the sequence blocks are processed before the second list of continuous blocks. Refer to "Sequential Control Blocks" on page 139 for details on sequential control block processing.

Scan Period

For control processors, the scan period defined by the block processing PERIOD/PHASE of the block. This includes:

- The block processing cycle (BPC) set for the control processor (one of the following: 0.05 sec, 0.1 sec, 0.2 sec, 0.5 sec, or 1.0 sec)
- The PERIOD parameter (0-13) set for the control block.

NOTE

The scan period is NOT defined by the block (or database) scanning cycle of the OM scanner.

Table 6-1 lists the allowable user-specified scan periods for CP60s, FCP270s, ZCP270s, and FCP280s, based on these settings. For these CPs, PERIOD can be 0-13.

Table 6-1. Scan Periods for CP60, FCP270, ZCP270, FCP280

PERIOD	BPC = 0.05 sec	BPC = 0.1 sec	BPC = 0.2 sec	BPC = 0.5 sec	BPC = 1.0 sec
0	0.1 sec	0.1 sec	0.2 sec	0.5 sec	1.0 sec
1 (default)	0.5 sec	0.5 sec	0.6 sec	0.5 sec	1.0 sec
2	1.0 sec	1.0 sec	1.0 sec	1.0 sec	1.0 sec
3	2.0 sec	2.0 sec	2.0 sec	2.0 sec	2.0 sec
4	10 sec	10 sec	10 sec	10 sec	10 sec
5	30 sec	30 sec	30 sec	30 sec	30 sec
6	1.0 min	1.0 min	1.0 min	1.0 min	1.0 min
7	10 min	10 min	10 min	10 min	10 min
8	-1[1]	60 min	60 min	60 min	60 min
9	0.2 sec	0.2 sec	0.2 sec	0.5 sec	1.0 sec
10	5.0 sec	5.0 sec	5.0 sec	5.0 sec	5.0 sec
11	0.6 sec	0.6 sec	0.6 sec	0.5 sec	1.0 sec
12	5.0 sec	5.0 sec	5.0 sec	5.0 sec	5.0 sec
13	0.05 sec	0.1 sec	0.2 sec	0.5 sec	1.0 sec

1. This is a non-existent period.

Integrator and Gateway blocks have different period values than shown here. Their values are defined only by the PERIOD parameter, as shown in Table 6-2. For these Integrators and Gateways, PERIOD can be 0-12.

Table 6-2. Allowable Scan Periods for Integrator and Gateway Blocks

PERIOD	Scan Period for Integrators and Gateways
0	0.1 second

Table 6-2. Allowable Scan Periods for Integrator and Gateway Blocks (Continued)

PERIOD	Scan Period for Integrators and Gateways
1	0.5 second
2	1 second
3	2 seconds
4	4 seconds
5	8 seconds
6	16 seconds
7	32 seconds
8	64 seconds
9	128 seconds
10	256 seconds
11	512 seconds
12	1024 seconds

NOTE

For the PERIOD for an Allen-Bradley Integrator 30B (AB3B) station, refer to “Period Table” in *Integrators for Allen-Bradley Controllers* (B0193RG).
For the PERIOD for a Modbus Integrator 30 Style B (MG3B) station, refer to “Period Table Used by the Integrators” in *Integrators for Modbus and Modbus Plus Devices* (B0193RL).
For the PERIOD for the FDSCAN, FDMSBL blocks for a Device Integrator 30B (FD3B) station, refer to “FDMSBL Block Data Elements” in *Device Integrator 15 and Device Integrator 30 User's Guide* (B0193RH).
For other integrators, refer to the applicable document for your integrator.

In general, a control block cannot be processed any faster than the BPC of the CP, so if the PERIOD is configured faster than the BPC, then it will default to BPC.

Also, the configured PERIOD must be a multiple of the BPC. For example, if the BPC is 0.2 seconds and the PERIOD is 1 (0.5 seconds), you will not be able to get the scan rate of 0.5 seconds. 0.5 seconds is not evenly divisible by the BPC of 0.2 seconds; the closest is 0.6 seconds, which will be the scan period for the block.

To set the block processing cycle (BPC) for a control processor:

- In Foxboro Evo Control Editors (hereinafter referred to as Control Editors), set the parameter called “Basic Proc Cycle” in the Software tab for the control processor. Refer to “Configuring Controllers - Software” in *Hardware Configuration User's Guide* (B0750BB).
- In IACC, set the BPC through the Parameter Editor invoked for the desired CP. Refer to “Parameter Editor” in *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).
- In System Definition, select the desired CP and set the parameter called “Basic Proc Cycle x 10” from the Parameter Definition screen. Refer to “Assigning Parameter Definitions” in *System Definition: A Step-By-Step Procedure* (B0193WQ).

To set the PERIOD parameter for a control block:

- ♦ In the Control Editors, set the parameter "Block Sample Time" in the General tab for the block. Refer to "Opening the Block Configurator" in *Block Configurator User's Guide* (B0750AH).
- ♦ In IACC, set the parameter "Period" in the Properties dialog box for the block. Refer to "Properties Dialog Box" in *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).
- ♦ In ICC, set the parameter "PERIOD" in the Block Parameters window. Refer to "Edit Block Parameters Window" in *Integrated Control Configurator* (B0193AV).

Compounds process in the order in which you install them (from top to bottom) in the control configurator (Control Editors, IACC, or ICC), subject to variations due to differences in their periods.

Within the compound, blocks also process in the order in which you install them. As with compounds, you can modify the execution sequence by means of the period and phase parameters.

Block execution is initiated by the compound processor when a compound is turned on.

The compound period should be set to the Block Processing Cycle. A compound period of zero implies an execution period equal to the BPC.

Phases (PHASE) and periods (PERIOD) can be adjusted in individual blocks to alleviate potential overrun situations.

NOTE

Be aware that the fastest allowed ECB period for the HART™ FBMs is 100 milliseconds (PERIOD = 0). However, it is recommended that you refer to the *Sizing Guidelines and Excel Workbook* appropriate for your Control Processor to determine the optimal BPC for this ECB in order to prevent overloading.

Scan Overrun

A Scan Overrun occurs when a processor has more blocks than it can process within a single scan cycle.

Block Phasing

A block can never be executed more frequently than the minimum Block Processing Cycle (BPC) of the controller. A block with a period of zero (0) is executed as if the period were 0.5 second if the BPC is 0.5 second.

The BPC default value is 0.5 second. However, the BPC can be set to 0.1 second or other values with System Configurator/Definition.

Phasing is the distribution of the block processing over multiple BPCs. The object is to balance the number of blocks scheduled for processing in any given BPC.

When the block period is equal to the BPC, the block is processed every cycle and there are no phasing options.

However, when the block period is greater than the BPC (processed less frequently), the block can be phased to execute at some time other than the very beginning of the period. This feature is useful in avoiding overruns. Overruns occur when too many blocks are scheduled for execution at the same BPC.

Relationship Between Block Period and Phase

Block periods greater than the BPC are always equal to an integral number of BPCs (for example, when the BPC is 0.5 second, a block period of 2.0 seconds equals 4 BPCs).

Each block uses this integral multiple, N, to set up a modulo N counter that counts BPCs and triggers block execution every Nth BPC. In the example where N equals 4, the block's modulo counter cycles through the counts 0-1-2-3-0-1-2-3-0-1-2-3-0-... , triggering block execution every fourth cycle.

The block's Phase value determines whether block execution occurs at the 0 count, the 1 count, the 2 count, or the 3 count. A Phase value greater than three in this example exceeds the maximum modulo count and the block will never be executed.

Generally, the legal phase values are the integers from 0 to N-1, where N equals the number of BPCs in the Period. Table 6-3 lists the phase values for a controller that has a BPC of 0.5 seconds.

Table 6-3. Legal Phase Values (BPC = 0.5 second)

PERIOD		PHASE	PERIOD		PHASE
Value	Length	Values	Value	Length	Values
0	0.1 sec	Not allowed	7	10 min	[0 - 1199]
1	0.5 sec	[0] default	8	60 min	[0 - 7199]
2	1.0 sec	[0,1]	9	0.2 sec	Not allowed
3	2.0 sec	[0,1,2,3]	10	5.0 sec	[0 - 9]
4	10 sec	[0 - 19]	11	0.6 sec	[0] default
5	30 sec	[0 - 59]	12	5.0 sec	[0 - 9]
6	1 min	[0 - 119]	13	0.05 sec[1]	Not allowed

1. Available in FCP280, FCP270 and ZCP270, when permitted by the selected BPC.

Thus, a block having a PERIOD/PHASE assignment of 3/1 executes every 2.0 seconds in the second BPC frame of the two second period.

WARNING

Do **not** assign a block phase value that is not shown in Table 6-3. Such an improper assignment causes a warning message in the control configurator and, if not corrected, causes the block to be undefined. This results in the block **not** being processed.

Assume a group of blocks all having a PERIOD value of 1, 2, or 3, and a PHASE value of 0. Execution of these blocks will follow the patterns shown below.

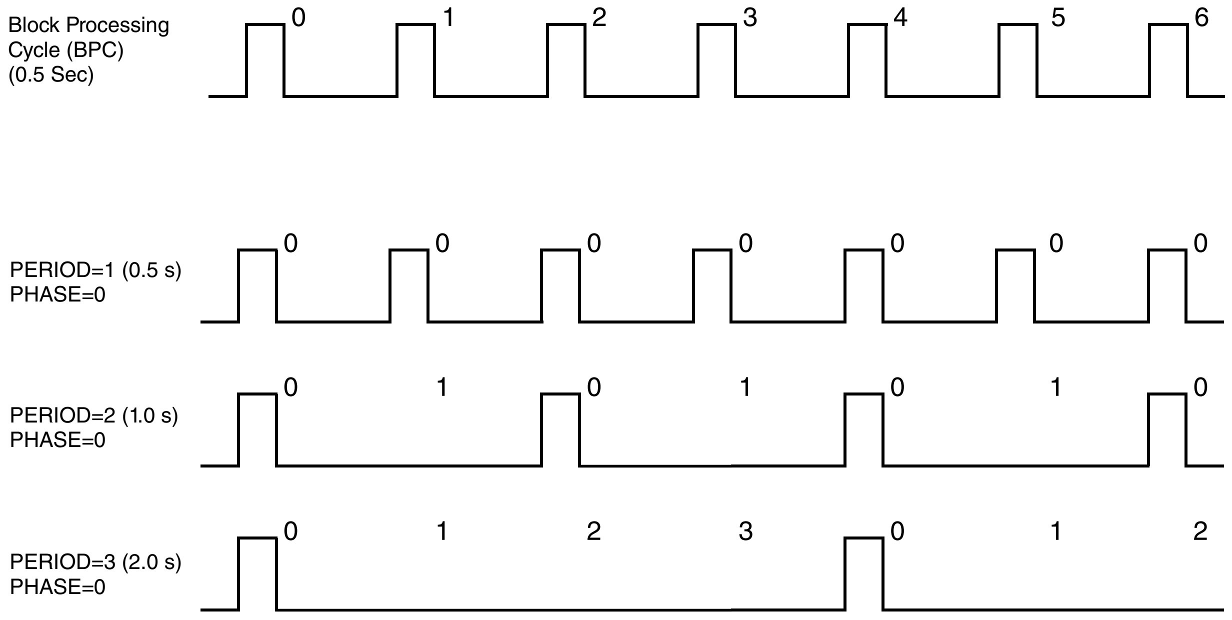


Figure 6-2. Example of Phased Execution

A processor can be overloaded with more blocks than it can process in a single BPC. To prevent overloading, you can balance the block load within a single phase by changing a block(s) execution to another phase.

Scan Overload

When blocks are configured initially, they default to a phase of zero. As shown in Figure 6-2, many blocks can be designated for execution in Phase 0 of a single BPC. When all blocks in a processor have a phase of zero, they are processed during a single BPC. In such a case, the processor must be able to handle all the blocks in a single BPC, or scan overrun may occur.

For example, if the blocks in Figure 6-2 were equal in load, the controller's total execution time for each BPC would take the pattern shown in Figure 6-3. BPC0 and BPC4 begin to show the effects of a Phase 0 pileup; the amount of time required to process the blocks increases as more blocks are added.

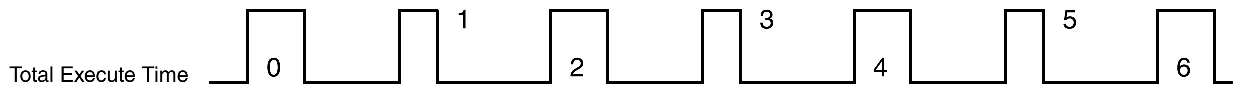


Figure 6-3. Phase Zero Overload

A Scan Overrun occurs when a processor has more blocks that it can process within a single scan cycle. During a scan overrun, the Compound Processor continues to process blocks even after the allocated period for the cycle is over. It simply extends the scan cycle into the next BPC until it has completed processing the remaining blocks. However, the processor does not perform any of the operations scheduled for this overrun BPC; to prevent the scan cycles from being offset, the overrun BPC is ignored. When finished, the Compound Processor waits until the beginning of the next scheduled BPC to begin the next execution.

This situation is illustrated in Figure 6-4. If you add CMP1, a large compound of blocks having a PERIOD value of 3 and PHASE 0, to the blocks described in Figure 6-3, the PHASE 0 Total Execute Time is large enough to cause an overrun and the PHASE 1 operation is delayed. The next processing operation occurs at PHASE 2.

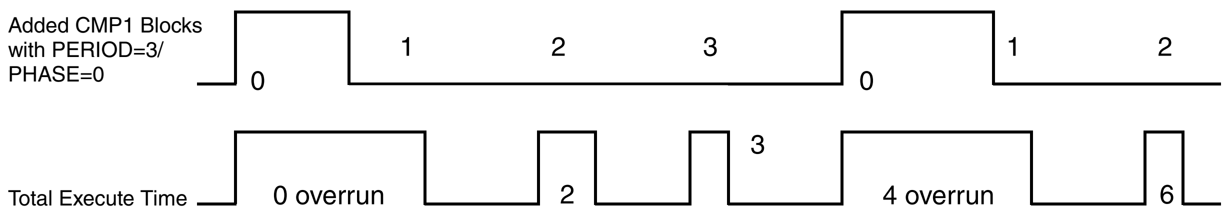


Figure 6-4. Example of Overrun

— NOTE

When the blocks are configured at longer periods, the Compound Processor may still overrun intermittently due to a pileup at certain Phase 0 frames. Adjust the phases to alleviate the scan overrun.

An overrun guarantees that the control operation goes to completion, but it also results in the loss of a processing cycle and the skewing of succeeding cycles. An occasional overrun may not affect control, but repeated occurrences eventually degrade process control.

To alleviate a potential scan overrun situation, the block phasing should be adjusted for the extra blocks causing the scan overrun. Configuring block phasing balances out the load on the controller's database by spreading block processing evenly throughout the BPCs.

Using the same example, you can configure CMP1, the large added compound executing at PHASE 0, to execute at other times (for example, Phases 1 and 3) and avoid the overruns that occurred at BPC0 and BPC4 (PHASE 0).

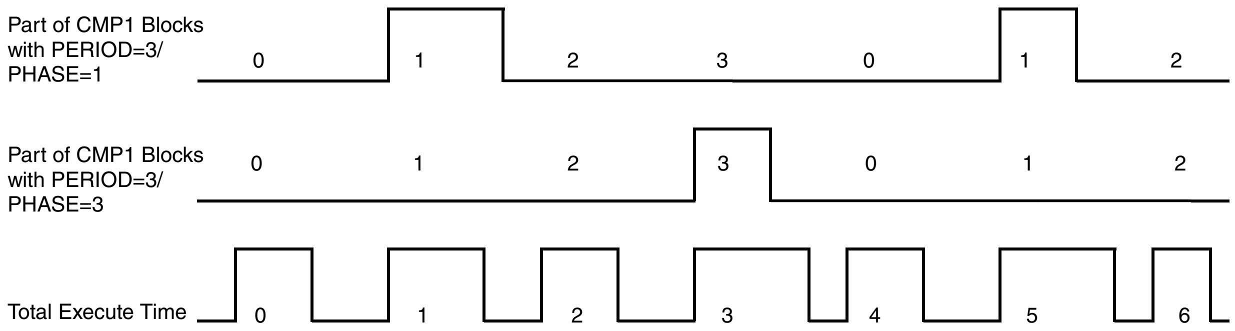

Part of CMP1 Blocks with PERIOD=3/ PHASE=1
0
1
2
3
0
1
2
Part of CMP1 Blocks with PERIOD=3/ PHASE=3
0
1
2
3
0
1
2
Total Execute Time
0
1
2
3
4
5
6

Figure 6-5. Avoiding Overrun

The following describes how to avoid scan overrun by configuring block phasing.

Block Phasing: Cautions and Guidelines

Phase configuration should take into account a block's position in the control scheme, its work load, the number of active options, and so on. Some guidelines follow.

Compound Phasing

- The system defaults the Compound Period to the system BPC, and the Compound Phase to 0.

ECB (Equipment Control Block) Phasing

- ECBs are configured with their own periods and phases. The "execution" of an ECB at its configured period and phase refers only to the capture and storage of input data from its associated FBM. This need not be done any more frequently than the input data is required by the connected control block(s).
- ECBs can be configured as part of any ordinary compound or as part of the station's ECB compound. Within any one BPC, the Compound Processor guarantees that the scheduled ECBs are processed prior to the scheduled I/O blocks in that compound. When the I/O block is processed, it receives the input data from the current contents of the ECB, whether or not those contents were refreshed in the current BPC. Note that the writing of output data from an output block to its destination ECB is always controlled by the output block. After all the compound's I/O control blocks have been processed, those ECBs which have received fresh output data from the control blocks transmit the data to their connected FBMs, in the current BPC. Therefore, output activity is not under control of periods and phases.
- Output blocks such as AOUT or COUT write their values to their destination FBM via their ECB during the BPC frame assigned to the block after the output block executes. However, the readback of the FBM's output registers is initially scanned by the ECB before the output block executes. This is used to calculate the BCALCO.

Control Block Phasing

- All block phasing is on a global basis. Phase 0 is common to all blocks, compounds and ECBs.
- When you configure phasing, Block execution order and controller loading are the major considerations. Block execution order in a control loop should be set up so that the measurement input blocks execute immediately before the control blocks, and the output blocks immediately follow the control blocks. If all blocks in a control loop are in the proper order and have the same period and phase, then block initializations and bumpless transfers work properly, even when PRIBLK = 0.
- In cascade strategies, the primary blocks should execute before the secondary blocks. Execution of feedforward blocks should be sequential in the data flow direction.
- If a phase value, outside of the legal phase values for the assigned Period value, is configured, the block, compound, or ECB is never processed. A warning message is displayed in the control configurator and the block, compound, or ECB is set undefined. If you reduce the Period value, the existing Phase value is verified to ensure that it is within the legal range for the new Period value.

Input/Output (I/O) Blocks

I/O blocks provide the application interface between the physical process inputs and outputs for the system. They process as part of any compound.

The I/O blocks relate logical names (FLWTRNS100, for example) to physical hardware point addresses, which are identified by the FBM device ID and point number.

All control references to I/O points (such as from displays or data logging) use the block name rather than the physical address.

Input blocks have manual modes to enable you to disconnect control schemes from live FBMs for simulation and checkout purposes.

Output blocks have Manual modes to enable operator intervention in controlling the process.

Analog input blocks convert raw data read from the FBMs to a real value in engineering units. They also provide signal conditioning.

Piecewise linear characterization is performed for thermocouple conversion (see "Signal Conditioning Indexes" on page 207).

Different AIN blocks can be connected to the same FBM to provide conditioning to different engineering units.

Redundant I/O blocks provide high availability by utilizing two separate FBMs to support a single process point. (See "Redundant I/O Blocks" on page 70).

I/O Block Processing

I/O processing takes place both in the Control Processors (FCP280, FCP270 and ZCP270) and the Fieldbus Modules (FBMs). FBMs interface to the control system through Equipment Control Blocks (ECBs), which are "holding places" for the FBM data.

Each Fieldbus Module has an accompanying ECB. ECBs are created through the control configurator in the same manner as blocks.

The system monitor and the compound processor are responsible for maintaining this active interface at the ECB level.

"Control Processor/Fieldbus Application Interface" on page 167 provides I/O Block, ECB, and FBM application interface information. "Equipment Control Blocks (ECBs)" on page 182 and "200 Series Fieldbus Module Types (FBMs)" on page 217 provides detailed information on ECBs and FBMs.

I/O Block Status

A bad FBM or bad channel status combine to form a bad data status. This bad status is included in the output value record of input type blocks.

For outputs, a check is made for a non-operational FBM, or loss of communication. These are combined in a BAD output status which is available as a Boolean output for connection into control strategies.

Output blocks also perform a readback of the FBM raw output values and this information is made available in special output parameters. This allows live process values to be fed back into control schemes for bumpless initialization purposes.

I/O Block Validation

At the system level, a process input/output (PIO) maintenance task checks the security status of each installed FBM. This task runs periodically.

It establishes the operational status of each FBM and its related ECB.

At the application level, process input values are checked against fixed limits through various alarm options at the block level.

Analog Inputs:

- High/low sensor range check
- High/low alarm checks in engineering units
- Rate-of-change detection and alarming.

Analog Outputs:

- Bad alarming.

I/O Fail-safe Status

The desired fail-safe value of a particular FBM output channel can be configured.

An output point in an FBM can be configured on a per-type (analog or digital) basis to fail to a fallback value or to maintain its current value. An analog output fails to a hold value or fails to a configured value. A digital output fails to zero, one, or to a hold value. The individual FBM stores the fail-safe values.

I/O fail-safe mode of operation can be caused by the loss of process I/O communications or by internal FBM diagnostics. An FBM is reset from fail-safe status by an automatic I/O reset request from the Control Processor.

Redundant I/O Blocks

Redundant I/O blocks are provided for applications that require higher system availability.

Redundant Input Block (AINR)

The Redundant Analog Input Block accepts inputs from two separate and separately named FBMs. The user designates a primary and a secondary signal. Both signals are converted and validated but only the primary signal is presented to the block's point output, unless a failure is detected.

If a failure is detected and the secondary signal is healthy, that signal is used on the block's point output.

When either signal fails or when they differ by more than a user-defined parameter, alarms are generated.

Redundant Output Block (AOUTR)

The Redundant Analog Output Block connects to any pair of FBMs providing analog output, providing they are of the same hardware and software types. For FBCs or any output FBMs between FBM01 and FBM46, redundant operation is achieved when a special Termination Cable Assembly (TCA) is installed. For DIN rail mounted FBMs, a Redundant Adapter is installed on the Baseplate linking the redundant FBMs. Special TCAs or Redundant Adapters attach to the pair of FBMs and use diodes to combine the pairs outputs into a single analog signal. There are redundant TCAs available for FBMs 4, 5, 6, 37, and 39 (FBM04 and FBM06 use identical TCAs). Redundant Adapters are available for FBM205, FBM208, and FBM237.

As noted in "Control Processor/Fieldbus Application Interface" on page 167, the special TCA and Redundant Adapters provide redundancy correctly only when the fail-safe mask and fail-safe fallback values in both of the ECBs are configured to the required values.

Alarms can be generated when either output fails.

Redundant Contact Input Block (CINR)

The Redundant Contact Input (CINR) block receives redundant input values for a single digital input process point from two digital input type Fieldbus Modules (FBMs). The CINR block is currently qualified for use only with DCS FBMs for Migration to APACS+ systems. Based on the quality of the two inputs and the user specification of a default selection, one of the inputs is chosen for use in the control strategy.

The CINR block offers Auto/Manual control and simulation mode capability. It also provides bad point and state alarming of the digital input, and other standard block alarm functions. The block also offers an input inversion option.

Redundant Contact Output (COUTR)

The Redundant Contact Output block (COUTR) provides the control strategy with redundant digital output values for a single digital output point in any DCS FBM containing such points. The COUTR block is currently qualified for use only with FBM240 and DCS FBMs for Migration to APACS+ systems. You can select a sustained output that follows the block input or a pulsed output with a selectable pulse width. The block provides Auto/Manual/Auto Hold control modes and simulation mode capability. It also provides bad alarming of the contact outputs read back from the FBMs.

To provide bumpless transfer from Manual to Auto in cascade control loops, the COUTR block alerts upstream blocks to open loop conditions via an initialization request output. It also indicates when the contact outputs from both FBMs are in the failsafe state.

Input Signal Conditioning

The analog type I/O blocks support a set of signal conditioning algorithms, which are specified by a set of index numbers.

The AIN reads the analog input point data from the FBM in raw counts. After status validation, the raw counts are conditioned and converted to engineering units in floating point, according to the specified conversion index and engineering units range. See Figure 6-6 for an example of input signal conditioning.

Note that most raw data is normalized in positive valued counts from 0 to 64,000 and can have a legal converted value in the range of 0 to 65,535 counts.

The conditioned value is converted to the specified units using the high and low sensor range. The data is then optionally filtered and made available at the output parameter.

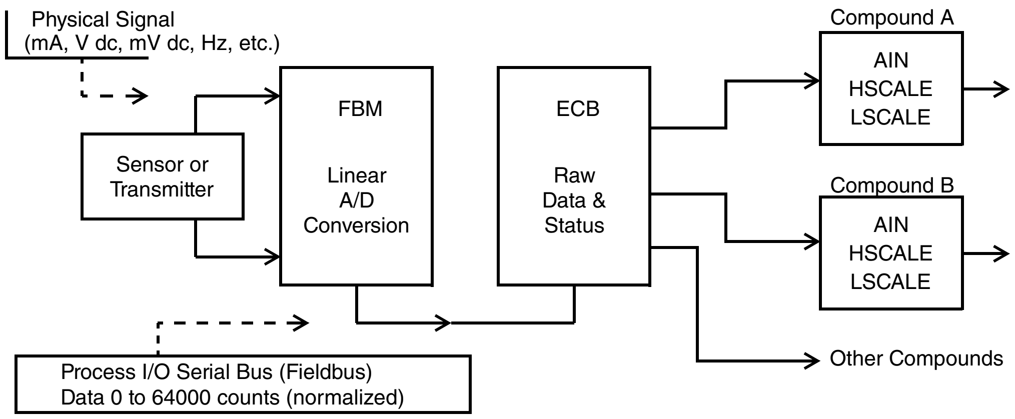


Figure 6-6. Input Signal Conditioning

Measurement values are linearized using linear/square root conversion, standard curves, and/or piecewise-linear interpolation. This maintains the accuracy of the raw data over the entire range. The following types of signal conditioning are provided (for details, see "Input Signal Conditioning" on page 207):

- Default = 0, no conditioning
- 0 to 64000 raw counts linear (Analog Input 0 to 20 mA)
- 1600 to 64000 raw counts linear (Analog Input 0 to 10 V dc)
- 12800 to 64000 raw counts linear (Analog Input 4 to 20 mA)
- 0 to 64000 raw counts square root (Analog Input 0 to 20 mA)
- 12800 to 64000 raw counts square root (Analog Input 4 to 20 mA)
- 0 to 64000 raw counts square root with low cutoff (Analog Input 0 to 20 mA)
- 12800 to 64000 raw counts square root with low cutoff (Analog Input 4 to 20 mA)
- 1600 to 64000 raw counts linear with low cutoff (Analog Input 0 to 10 V dc)
- 12800 to 64000 raw counts linear with low cutoff (Analog Input 4 to 20 mA)
- 14080 to 64000 raw counts linear (Analog Input 2 to 10 V dc)
- 14080 to 64000 raw counts square root with low cutoff (Analog Input 2 to 10 V dc)
- type E thermocouple
- type E thermocouple EA-2
- type J thermocouple
- type K thermocouple
- type R thermocouple
- type S thermocouple
- type T thermocouple
- type N thermocouple
- nickel RTD

- platinum RTD (SAMA)
- platinum RTD (DIN)
- platinum RTD (IEC)
- copper RTD.

Every RTD/thermocouple “SCIX” (Signal Conditioning Index) has a table associated with it (see Appendix A “Signal Conditioning Indexes” on page 207).

The CHARC block can be used to apply user defined piecewise-linear characterization to non-standard inputs. This is accomplished by connecting the EXTBLK parameter of the input block to the BLKSTA parameter of the CHARC block, and setting the EXTOPT parameter of the CHARC block to 1. The CHARC block segments must be configured as required.

You can also use “special thermocouple” signal conversion for nonstandard thermocouple inputs when the cold-junction compensation must be accommodated by the CHARC block. To do this, you should connect the EXTBLK parameter as described, but the EXTOPT parameter of the CHARC block should be set to 2. First, the cold junction compensation temperature is converted by the CHARC block from degrees Celsius into millivolts, using reverse interpolation. Then, the millivolt equivalent of the cold junction reference is added to the millivolt analog input point value, and the sum is passed to the CHARC block for forward interpolation. This results in the generation of a compensated output value in degrees Celsius.

Filtering

First or Second-Order

First- or second-order filter is provided on an optional basis:

Option:

0 = No filtering
1 = First order
2 = Second order Butterworth

Contact Filtering

Contact bounce is filtered by the FBM.

Output Signal Conditioning

Before an output block writes a value to an appropriate type FBM, it is converted to raw counts.

The analog value, which is written to an FBM point, is converted from engineering units to counts, using the high and low output range. The processing is the inverse operation of signal conditioning described for AIN blocks. This value is conditioned according to the applied conditioning index. See “Signal Conditioning Indexes” on page 207. The result is clamped between the high and low actuator limit value.

See Figure 6-7 for an example of output signal conditioning.

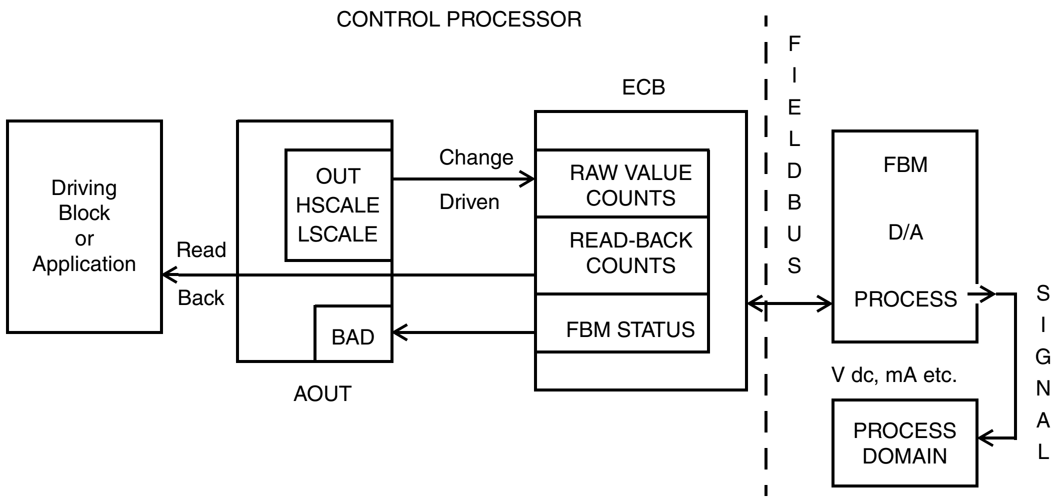


Figure 6-7. Output Signal Conditioning

The following output Signal Conditioning options are applicable for analog output I/O blocks (for details, see "Output Signal Conditioning" on page 213):

- Default = 0, no conditioning
- 0 to 64000 raw counts linear (Analog Output 0 to 20 mA)
- 1600 to 64000 raw counts linear (Analog Output 0 to 10 V dc)
- 12800 to 64000 raw counts linear (Analog Output 4 to 20 mA)
- 0 to 64000 raw counts square root (Analog Output 0 to 20 mA)
- 12800 to 64000 raw counts square root (Analog Output 4 to 20 mA)
- 14080 to 64000 raw counts linear (Analog Output 2 to 10 V dc)
- 14080 to 64000 raw counts square root with low cutoff (Analog Output 2 to 10 V dc).

7. Block/Process Alarming

This chapter provides details on block alarm processing, alarm states, and data recorded in alarm messages.

Alarming Overview

Block alarming, or process alarming, is used to detect when a specific event or condition with potentially hazardous consequences has occurred. This event or condition involves a change in the operation of a monitored field device or block.

When an alarm is triggered in a block, notification is sent through the station block to the Alarm Manager, which alerts the operator via a message on the process display, and/or a horn on an annunciator keyboard. In most cases, the station block can also send the alarm alert to an alarm printer as well. The alarming system is shown in Figure 7-1.

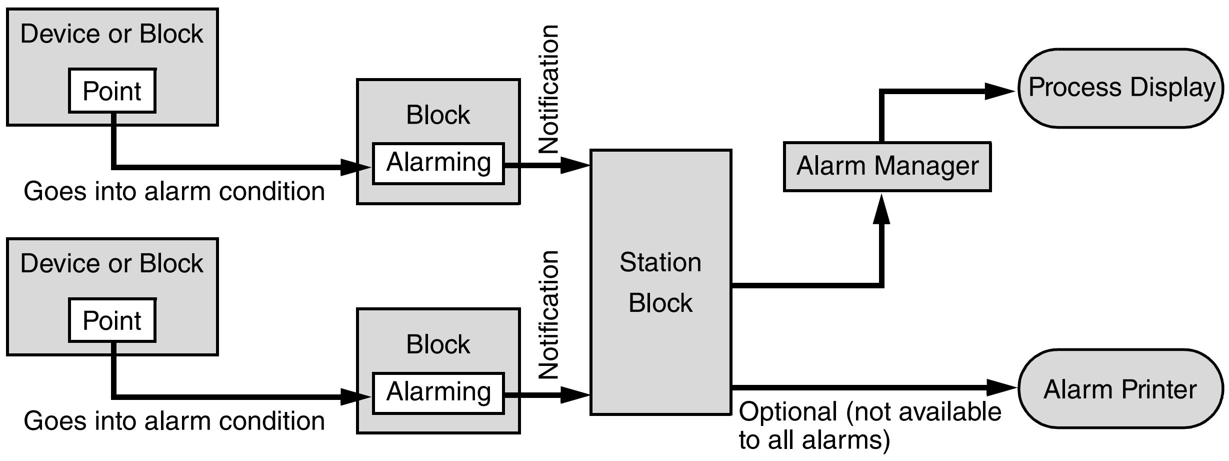


Figure 7-1. Block Alarming

There are a number of different types of process alarming, each specific to a particular event or condition. To enable each type of process alarm, you must determine and configure the following information (via alarm parameters specific to each block):

- ♦ which types of alarms you want to enable (alarm options), and which types of alarms you want to inhibit
- ♦ the text that appears in the process display if the alarm is initiated
- ♦ the device group that you want each alarm message sent to
- ♦ the priority level you want to assign to each alarm
- ♦ any parameters specific to the alarm type, such as the limit at which the alarm is initiated and the deadband that determines when the alarm condition returns to normal.

Configuration procedures for process alarms are available in *Process Operations and Displays* (B0700BN).

Some blocks have alarming as their primary function, however, most blocks offer alarming on an optional basis. Some blocks do not offer alarming features.

Alarming provides the basis for:

- Alarm detection via alarm output indicators (see next section)
- Message generation, related to the value of certain parameters
- Guidance notification.

The following section explains how process alarms are generated.

Process Alarm Generation

Blocks typically generate process alarms by setting a bit in the alarm status parameter (ALMSTA), which is a packed long value. Additionally, a user-defined message may be sent to the process displays, depending on whether this feature is inhibited or not. User-defined messages are not sent to the alarm printer.

Most blocks have a Boolean parameter for each alarm which acts as an alarm indicator; that is, a parameter set to true when the alarm is active, and false when it is not. Other continuous blocks, sequence blocks, and applications can connect to output alarm indicators, to pass the alarm on to the operator and control software.

Refer to the next section to learn more about ALMSTA, alarm inhibition, and other features common to all process alarms.

Common Process Alarm Features

The following sections detail the common features that block alarms share:

- "Common Process Alarm Parameters" on page 77 - alarm parameters that typically exist in every block that supports alarms
- "Alarm Messages" on page 78 - reports on alarms
- "Alarm Status Parameter (ALMSTA)" on page 79 - reports the states of alarms
- "Alarm Option Parameter (ALMOPT)" on page 81 - indicates availability of alarms
- "Criticality and Priority Type" on page 82 - rate each alarm based on the system importance of the alarm
- "Alarm Inhibition" on page 83 - prevents alarm from occurring or being displayed.

Detailed descriptions of the available types of process alarms are provided in "Process Alarm Types" on page 88.

Common Process Alarm Parameters

Some examples of specific parameters related to the process alarm functions that are held in most blocks that support alarming, are provided below:

Per Alarm Point:

[Alarm Indicator][1]	Set to true when a specific alarm point is in alarm. It can be connected into control strategies.
[Group Number][1]	Allows you to assign a group number from 1 to 8 for alarm device destination and distribution. You can specify up to eight destination devices for each group between group number 1 and group number 3. You do this by configuring the GRx and DVy parameters in the compound containing the block. You can specify up to 16 destination devices for each group between group number 4 and group number 8. You do this by configuring the GRx and DVy parameters of the station block for the station containing the block.
[Priority Level of Alarm][1]	A priority level of 1-5 assigned for display and annunciation purposes, where: 1 = highest priority (0 = no alarm) The block does not use this parameter. The alarm message processor uses it as a filter for the Compound Inhibit (CINHIB) function.
[Alarm Text]	A string that is sent to a process display when a specific alarm point is in alarm (if uninhibited). In some blocks, a “return-to-normal” message may also be available for when the point goes out of alarm. This string is sent with the alarm message (see “Alarm Messages” on page 78).

One Per Block:

INHIB	All alarms in the block are inhibited. The specific action taken depends on the value of INHOPT in that block.
INHALM	Contains packed Boolean values that represent alarm inhibit requests for each type of alarm configured in the block. The specific action taken depends on the value of INHOPT in that block.
INHOPT	Defines the behaviors applied to inhibited and, optionally, uninhibited alarms.

— NOTE

More information on the INHIB, INHALM, and INHOPT parameters is available in “Alarm Inhibition” on page 83.

CRIT	Indicates the highest priority of all active alarms within the block.

1. Name varies, depending on the block in which it is used.

UNACK	A Boolean output parameter which is automatically set true (1) when a block alarm becomes active. It is resettable, by an operator “acknowledge” pick on a default or user display, or via a user task. It may also be reset automatically when the alarm condition clears if that option is specified by the INHOPT parameter.
AMRTIN	Alarm Regeneration Timer is a configurable, non-settable integer that specifies the time interval for an alarm condition to exist continuously, after which a new unacknowledged alarm condition and its associated alarm message is generated. The parameter value ranges from zero (default, no regeneration) to 32767 seconds.
NASTDB	Alarm Deadband Timer is a configurable, non-settable long integer. Depending on the value of NASOPT, it either specifies the deadband time interval that must elapse before an alarm condition is allowed to return to normal, or the length of a delay-on timer which specifies the amount of time between an alarm’s detection and the announcement of the alarm. The parameter value ranges from zero (default, no delay) to 2147483647 ms.
NASOPT	Alarm Suppression Option is a configurable, non-settable Boolean that specifies the method for delaying alarms. Alarms can be delayed by either delaying the return-to-normal condition (default), or delaying detection of the alarm itself, by the length of time specified in NASTDB.

Additional information is sent to the process display in each alarm message which is not user-configured. This information is detailed in the next section.

Alarm Messages

When an alarm is generated, an alarm message is sent to the configured device group. This alarm message consists of the following data:

— NOTE

The following terms are used internally with the blocks, and are not parameter names.

TIME	Time of the message
DATE	Date of the message
CNAME	Compound name (for example, FURNACE)
BNAME	Block name (for example, HEATXCHFLW)
PNAME	Parameter name (for example, MEAS)
TYPE MSG	Text indicating the type of check violated (for example, HI DEV) or Fail Text (for example, PUMP FAILURE) Be aware that this is not the alarm text configured for each block, but is derived from a separate list.

LIMIT	Alarm limit value (usually derived from a user-configurable parameter)
VALUE	Alarmed variable value
GROUP #	Destination device (usually derived from a user-configurable parameter)

Alarm messages are shown on a process display, and can also be sent to a historian and/or a printer. Alarm message data is derived from block parameters which have differing names specific to each block. Refer to the "Station Block" chapter of *Integrated Control Block Descriptions* (B0193AX) for information on configuring the station block to send alarm messages to their proper device group.

The next section details the parameter that the blocks use to generate alarms.

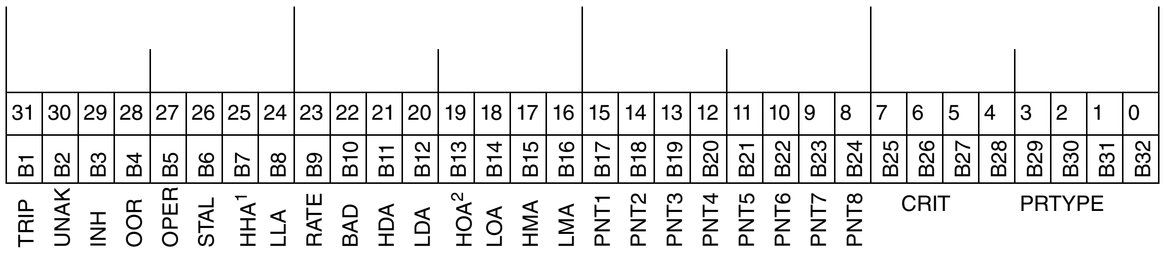
Alarm Status Parameter (ALMSTA)

When an alarm condition is detected, the block sets the appropriate bits of the Alarm Status (ALMSTA) parameter to true. When the alarm condition clears, the block resets this bit to false.

ALMSTA is a 32-bit block output parameter that is bit mapped to indicate the following block alarm states as they apply to those blocks that have alarming capability.

Blocks which do not have alarming capability have no ALMSTA parameter. Some blocks, such as the DCI blocks, only support alarming on systems with I/A Series software v8.4-v8.8 and Control Core Services v9.0 or later.

The remaining blocks have one or more applicable bits. Each individual block description in *Integrated Control Block Descriptions* (B0193AX) identifies the bits that make up the ALMSTA parameter for that block.



1. TARG in ACCUM block.
2. PTARG in ACCUM block.

A list of all the bits that can be assigned to the ALMSTA parameter and their indicated alarm state is shown in Table 7-1:

Table 7-1. ALMSTA-Assignable Bits

Bit Number[1]	Indicated Alarm State	Boolean Connection Extension
31 (MSB)	TRIP Trip Alarm	ALMSTA.B1
30	UNACK (unacknowledged)	ALMSTA.B2
29	INH Inhibit Alarm	ALMSTA.B3

Table 7-1. ALMSTA-Assignable Bits (Continued)

Bit Number[1]	Indicated Alarm State	Boolean Connection Extension
28	OOR Out of Range Alarm	ALMSTA.B4
27	OPER Sequence Operational Error Alarm	ALMSTA.B5
26	STAL State Change Alarm	ALMSTA.B6
25	HHA High High Absolute (TARG in ACCUM block)	ALMSTA.B7
24	LLA Low Low Absolute Alarm	ALMSTA.B8
23	RATE Rate of Change Alarm	ALMSTA.B9
22	BAD Input/Output Bad (BAD output of block)	ALMSTA.B10
21	HDA High Deviation Alarm	ALMSTA.B11
20	LDA Low Deviation Alarm	ALMSTA.B12
19	HOA High Output Alarm (PTARG in ACCUM block)	ALMSTA.B13
18	LOA Low Output Alarm	ALMSTA.B14
17	HMA High Measurement Alarm	ALMSTA.B15
16	LMA Low Measurement Alarm	ALMSTA.B16
15 to 8	PNT1 to PNT8 – Points in STATE Alarm (BLNALM)	ALMSTA.B17 – ALMSTA.B24
7 to 5	CRIT – Criticality (Range 0 to 5)	ALMSTA.B25 – ALMSTA.B27
4 to 0	PRTYPE – Priority Type (Range 0 to 9, 25)	ALMSTA.B28 – ALMSTA.B32

1. Bit 0 is the least significant bit (starting from the right).

Conditions that determine how these bits are updated include:

- CRIT and PRTYPE are updated when any alarm change occurs in the block.
- HMA and LMA are mutually exclusive, and are updated when High Absolute or Low Absolute alarm status changes occur.
- HDA and LDA are mutually exclusive, and are updated when High Deviation or Low Deviation alarm status changes occur.
- HOA and LOA are mutually exclusive, and are updated when High Output or Low Output alarm status changes occur.
- HHA is updated when High High alarm status changes occur.
- RATE is updated when Rate alarm status changes occur.
- BAD is updated when I/O bad alarm changes occur.
- STAL is updated when State alarm changes occur.
- UNACK is updated every processing cycle. UNACK is set automatically when a new alarm condition is detected. It is reset by the operator.
- TRIP is updated when a Trip alarm change occurs in an EVENT block.

- ♦ OOR is updated when a Range alarm change occurs in an AIN block.

All block types update the alarm status each processing cycle in order to track user changes in the UNACK status.

The ALMSTA parameter is set to zero when a block initializes.

ALMSTA for all block types except AOUT, COUT, and MCOUT is set to zero on an Auto to Manual transition. ALMSTA can be accessed by application programs. It can also be accessed by other blocks having long (32-bit) integer inputs that can be handled as packed Booleans (for example, the MCIN block when the IOMOPT option is configured false, or the CALC block).

Table 7-1 indicates the literal names assigned to the individual bits of ALMSTA. These names can be used in Boolean connection extensions (for example, ALMSTA.HHA).

As you will read later, each block only uses certain bits of ALMSTA, depending on the alarms available in the block. In order for other blocks and applications to determine which alarms are available in each block, the Alarm Option parameter is used.

Alarm Option Parameter (ALMOPT)

Alarm Option (ALMOPT) is a 32-bit block output parameter that is bit mapped to indicate which alarms are configured for a particular block. ALMOPT is a non-settable, non-connectable parameter. Each individual block description in *Integrated Control Block Descriptions* (B0193AX) identifies the specific bits that make up the ALMOPT parameter for that block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31	B32

Table 7-2 lists of all the bits that can be assigned to the ALMOPT parameter and their indicated configured alarms.

Table 7-2. ALMOPT-Assignable Bits

Bit Number[1]	Configured Alarm Option When True	Boolean Connection (B32 to B1)
28	Out-of-Range Alarm	ALMOPT.B4
27	Operational Error Alarm	ALMOPT.B5
26	State Alarm	ALMOPT.B6
25	High-High Absolute Alarm	ALMOPT.B7
24	Low-Low Absolute Alarm	ALMOPT.B8
22	Bad I/O Alarm	ALMOPT.B10
21	High Deviation Alarm	ALMOPT.B11
20	Low Deviation Alarm	ALMOPT.B12
19	High Output Alarm	ALMOPT.B13
18	Low Output Alarm	ALMOPT.B14
17	High Absolute Alarm	ALMOPT.B15

Table 7-2. ALMOPT-Assignable Bits (Continued)

Bit Number[1]	Configured Alarm Option When True	Boolean Connection (B32 to B1)
16	Low Absolute Alarm	ALMOPT.B16
7	Alarm Group 1 in Use	ALMOPT.B25
1	Alarm Group 7 in Use	ALMOPT.B31
0	Alarm Group 8 in Use	ALMOPT.B32

1. Bit 0 is the least significant bit (starting from the right).

There are no mnemonic names for the individual bits of ALMOPT.

The alarms available in each block are ranked according to their importance in the system using the criticality and priority type parameters which are described in "Criticality and Priority Type" below.

Discard Alarm Messages with Alarm Group 0

Foxboro Evo Control Software and Control Core Services normally only support Alarm Groups 1-8. Older configurators, such as ICC and IACC, did not check blocks to see if their alarm groups were configured outside of the valid 1-8 range. Also, alarm group parameters are settable, i.e. they can be changed by the user while the block is running. Therefore, logic in the control blocks clamp the alarm group range automatically to prevent a range check violation from stopping the block execution. If Alarm Group 0 were set, it would be clamped to Alarm Group 1, and corresponding messages would be sent to Alarm Group 1 or would be discarded if Alarm Group 1 were undefined.

Later, it was discovered that by not defining an alarm destination for Alarm Group 1, setting an alarm group parameter to 0 resulted in discarding the corresponding alarm message while continuing to perform the alarm detection. In some systems, this technique was preferable to using the alarm inhibit function to discard the messages, since the alarms would not show up as inhibited in any alarm display or summary.

In I/A Series software v8.6/InFusion v2.5, the control block logic was modified to add an option to the CFGOPT parameter in the Station block (Bit 11) to specify discarding alarm messages for any alarm whose alarm group is 0. If this option is not specified, using Alarm Group 0 will continue to send the alarm messages to all devices in Alarm Group 1 if Alarm Group 1 is defined.

To maintain backward compatibility, this feature is optional.

Criticality and Priority Type

It is important for the operator to be aware of which alarms take precedent over others. Each block uses the criticality and priority type parameters to make this determination automatically for the operator.

Criticality (CRIT) is an integer output that indicates the priority of the block's highest currently active alarm. The integer is a value of 1 through 5 with 1 being the highest priority. An output of zero indicates the absence of alarms.

Priority Type (PRTYPE) is an indexed output parameter that indicates the alarm type of the highest priority active alarm. The PRTYPE output includes the alarm types in Table 7-3.

Table 7-3. PRTYPE Indicators

Bit Number[1]	Description When True
25	Out of Range Alarm
9	State Alarm
8	Bad I/O Alarm
7	Rate of Change Alarm
6	Low Deviation Alarm
5	High Deviation Alarm
4	Low Low Absolute Alarm
3	High High Absolute Alarm
2	Low Absolute Alarm
1	High Absolute Alarm
0	No Active Alarm

1. Bit 0 is the least significant bit (starting from the right).

Together, PRTYPE and CRIT record the level and type of the highest alarm currently active. For example, assume the Bad and State alarms are both active. If the Bad alarm has priority 3 and the State alarm has priority 2, then CRIT = 2 and PRTYPE = 9. If the Bad alarm has priority 1 and the State alarm has priority 4 then CRIT = 1 and PRTYPE = 8. If both alarms have priority 2, then CRIT = 2 and PRTYPE = 8.

The compound processor maintains a compound criticality parameter in the compound header as it is processing the blocks. This connectable parameter stores the criticality of the highest priority, currently-active alarm in the compound, for the Alarm Manager to display.

Despite the priority level of each alarm, any alarm can be inhibited by the operator. The following section details alarm inhibition.

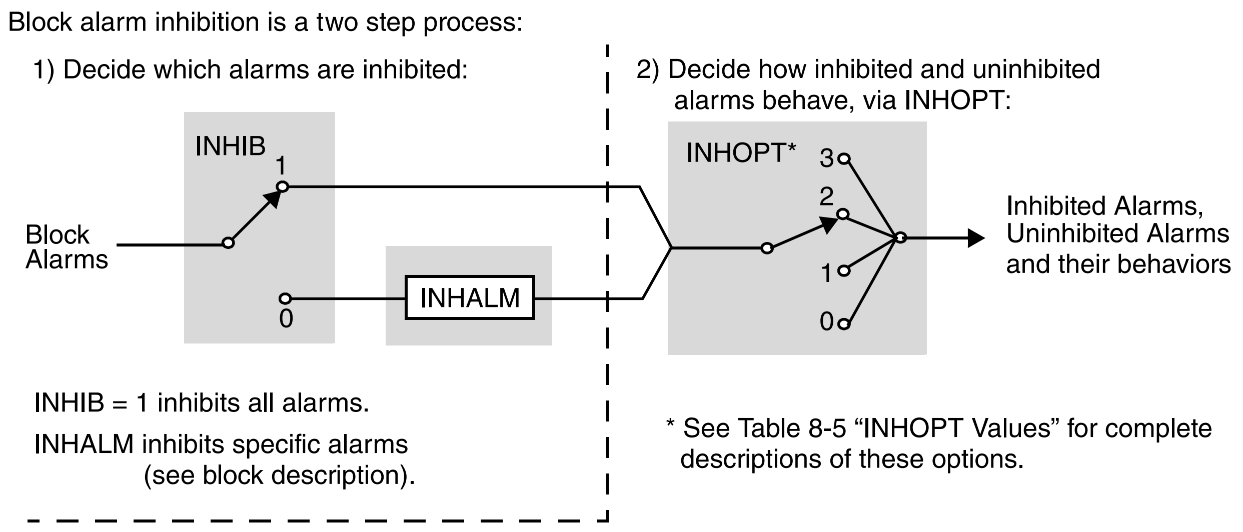
Alarm Inhibition

Block alarm inhibition is the process by which a block establishes a set of behaviors for specific alarms, such as preventing the delivery of alarm messages to the process displays and, optionally, disabling alarm detection.

The following parameters play key roles in inhibiting alarms:

- ♦ INHIB, INHALM, and INHOPT define the behaviors for the inhibited (and in some cases, uninhibited) block alarms (see Figure 7-2)
- ♦ CINHIB (in the CMP block) inhibits alarm messaging and detection at the compound level
- ♦ INHSTA records which alarms have been inhibited for a particular block.

Figure 7-2 illustrates how INHIB, INHALM, and INHOPT define block alarm inhibition.


Block alarm inhibition is a two step process:
1) Decide which alarms are inhibited:
2) Decide how inhibited and uninhibited alarms behave, via INHOPT:
INHIB
1
0
Block Alarms
INHALM
INHOPT*
3
2
1
0
Inhibited Alarms, Uninhibited Alarms and their behaviors
INHIB = 1 inhibits all alarms.
INHALM inhibits specific alarms (see block description).
* See Table 8-5 "INHOPT Values" for complete descriptions of these options.

Note: To perform these changes, see "Controlling Alarm States" in *Process Operations and Displays* (B0700BN).

Figure 7-2. INHIB, INHALM, and INHOPT Behavior

These inhibiting parameters are described in detail below:

- "INHALM" in next section
- "INHIB" on page 85
- "INHOPT" on page 85
- "INHPRT" on page 86
- "CINHIB" on page 86
- "INHSTA" on page 87.

INHALM

INHALM is a 16-bit packed Boolean input parameter that is supported in the continuous blocks that perform alarm detection for multiple alarm types or points. The packed Boolean values specify the alarm types or points to inhibit in the block. INHALM is a configurable, settable, and connectable parameter that is used in conjunction with the CINHIB compound parameter and the INHIB block parameter to determine which alarm types and points to inhibit in the block.

INHALM is not supported in AOUT, COUT, CHARC, PATALM, or in sequential blocks (DEP, IND, EXC, MON), since these blocks contain a single alarm that you inhibit by setting the INHIB parameter in the block. See *Integrated Control Block Descriptions* (B0193AX) to identify the specific bits that make up the INHALM parameter for that block.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B1	B2	B3	B4	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15

A list of all the bits that can be assigned to the INHALM parameter and their indicated inhibited alarms is shown in Table 7-4.

Table 7-4. INHALM-Assignable Bits

Bit Number[1]	Description When True	Boolean Connection (B16 to B1)
15	Trip Alarm	INHALM.B1
14	Unacknowledged	INHALM.B2
13	Inhibit Alarm	INHALM.B3
12	Inhibit Out-of-Range Alarm	INHALM.B4
11	Sequence Operational Error Alarm	INHALM.B5
10	State Change Alarm	INHALM.B6
9	Inhibit High-High Absolute Alarm	INHALM.B7
8	Inhibit Low-Low Absolute Alarm	INHALM.B8
7	Rate of Change Alarm	INHALM.B9
6	Inhibit Bad I/O Alarm	INHALM.B10
5	High Deviation Alarm	INHALM.B11
4	Low Deviation Alarm	INHALM.B12
3	High Output Alarm	INHALM.B13
2	Low Output Alarm	INHALM.B14
1	Inhibit High Absolute Alarm	INHALM.B15
0	Inhibit Low Absolute Alarm	INHALM.B16

1. Bit 0 is the least significant bit (starting from the right).

There are no mnemonic names for the individual bits of INHALM.

INHIB

Inhibit is a Boolean input. When true, it inhibits all block alarms; the alarm handling and detection functions are determined by the INHOPT setting. Alarms can also be inhibited based on INHALM and the compound parameter CINHIB.

INHOPT

Inhibit Option specifies the actions in Table 7-5 that apply to all block alarms. Refer to Figure 7-2 for more information on the role of INHOPT in alarm inhibition.

Table 7-5. INHOPT Values

Value	Alarm Inhibit Options
0	Inhibit alarm messages when alarms are detected. If an alarm occurs, flash the current alarm display and the appropriate detail displays, and sound the annunciator horn, if present.
1	Inhibit the detection of process alarms and the generation of alarm messages. All existing alarm states in the block are cleared automatically when the alarm is inhibited.

Table 7-5. INHOPT Values (Continued)

Value	Alarm Inhibit Options
2	Inhibit alarm messages when alarms are detected. If an alarm occurs, flash the current alarm display and the appropriate detail displays, and sound the annunciator horn, if present (same as Value = 0) **plus** enable automatic acknowledgement of alarms when they return to normal.
3	Inhibit the detection of process alarms and the generation of alarm messages. All existing alarm states in the block are cleared automatically when the alarm is inhibited (same as Value =1) **plus** enable automatic acknowledgement of alarms when they return to normal.

The INHOPT parameter also has the following attributes:

- ♦ The INHOPT parameter is configurable, but it is not settable or connectable.
- ♦ If INHOPT = 0 or 2, and alarms are inhibited, alarm detection is performed, but alarm messages are disabled.
- ♦ If INHOPT = 1 or 3, and alarms are inhibited, alarm detection is disabled.
- ♦ If alarm detection is being performed and INHOPT = 0 or 1, alarms must be acknowledged by an explicit user ACK command to reset the UNACK parameter to 0. This must be done even after the alarm has returned to normal.
- ♦ If alarm detection is being performed and INHOPT = 2 or 3, alarms are acknowledged automatically when all alarms return to normal. If all alarms have not returned to normal, they may also may be acknowledged by an explicit user command to set UNACK to 0. This means that the UNACK parameter will be cleared and the UNACK Boolean reset in block status when all alarms in the block have returned to normal. At that time, an ALARM_ACK message is sent to all reporting alarm devices configured in the block and compound.

INHPRT

In the Station block, the Inhibit printer option (INHPRT) inhibits printing of station's system messages.

CINHIB

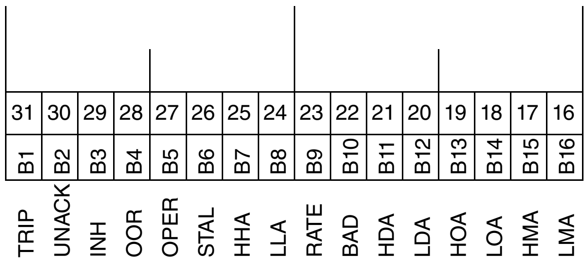
Compounds also support an alarm-inhibit parameter for the entire compound. The CINHIB parameter is a connectable integer that sets the suppression level of all messages at and below the specified priority level within the compound. The levels of each alarm are based on the value of CRIT (see "Criticality and Priority Type" on page 82).

Compound Inhibit specifies the priority levels of alarm inhibition within the compound, where:

- ♦ 0 = no inhibit
- ♦ 1 = inhibit all inclusive
- ♦ 2 = inhibit levels 2-5 inclusive
- ♦ 3 = inhibit levels 3-5 inclusive
- ♦ 4 = inhibit levels 4-5 inclusive
- ♦ 5 = inhibit level 5.

INHSTA

Inhibit Status (INHSTA) is a 32-bit block output parameter that is bit mapped to indicate the status of inhibited alarm options for a particular block. See *Integrated Control Block Descriptions* (B0193AX) to identify the specific bits that make up the INHALM parameter for that block.



A list of all the bits that can be assigned to the INHSTA parameter and their alarm status follows:

Table 7-6. INHSTA-Assignable Bits

Bit Number[1]	Name	Description When True	Boolean Connection (B32 to B1)
31	TRIP	Trip Alarm Inhibited	INHSTA.B1
30	UNACK	Unacknowledged Inhibited	INHSTA.B2
29	INH	Inhibit Alarm Inhibited	INHSTA.B3
28	OOR	Out-of-Range Alarm Inhibited	INHSTA.B4
27	OPER	Sequence Operational Error Alarm Inhibited	INHSTA.B5
26	STAL	State Change Alarm Inhibited	INHSTA.B6
25	HHA	High-High Absolute Alarm Inhibited	INHSTA.B7
24	LLA	Low-Low Absolute Alarm Inhibited	INHSTA.B8
23	RATE	Rate of Change Alarm Inhibited	INHSTA.B9
22	BAD	Bad I/O Alarm Inhibited	INHSTA.B10
21	HDA	High Deviation Alarm Inhibited	INHSTA.B11
20	LDA	Low Deviation Alarm Inhibited	INHSTA.B12
19	HOA	High Output Alarm Inhibited	INHSTA.B13
18	LOA	Low Output Alarm Inhibited	INHSTA.B14
17	HMA	High Absolute Alarm Inhibited	INHSTA.B15
16	LMA	Low Absolute Alarm Inhibited	INHSTA.B16

1. Bit 0 is the least significant bit (starting from the right).

Quality Status Parameter (QALSTA)

Quality Status parameter (QALSTA) is a connectable packed boolean output parameter that is provided as a single source for the following information:

- ♦ The value status of the primary measurement/input of the input block or control block
- ♦ A subset of block status (BLKSTA)
- ♦ The Manual status of the block - the inverse of the Manual/Auto status bit in the block status parameter BLKSTA.
- ♦ Alarm status (ALMSTA) data relevant to the primary measurement/input

QALSTA is non-configurable and non-settable, and is updated each block cycle after the block is processed. It is only provided in the analog input blocks (AIN, AINR, RIN, and RINR), discrete input blocks (CIN, CINR, BIN, and BINR), analog controllers (PID family, BIAS, and RATIO), discrete controllers (DGAP, PTC, and MDACT), and the REALM block.

User written blocks (e.g. CALC, CALCA, MATH, LOGIC) may connect to the QALSTA parameter in the AIN, AINR, RIN, RINR, CIN, CINR, BIN, and BINR blocks.

Refer to the individual block description in *Integrated Control Block Descriptions* (B0193AX) to identify the specific bits that make up the QALSTA parameter for that block. The available bits vary depending on block type.

Process Alarm Types

The following types of process alarms are used in the Foxboro Evo Process Automation System. A block may have multiple types of process alarms, depending on the block:

- ♦ Absolute alarming (includes output alarming, pretarget alarming, and target alarming)
- ♦ Bad input/output (I/O) alarming - see page 91
- ♦ Deviation alarming - see page 91
- ♦ Mismatch alarming - see page 93
- ♦ Out of range alarming - see page 93
- ♦ Rate of change alarming - see page 94
- ♦ Sequence operational error alarming - see page 95
- ♦ State change alarming - see page 95
- ♦ Trip alarming - see page 96.

NOTE

Other forms of alarming are available as well, but only the following types of alarming are explicitly initiated by blocks. For examples of other types of alarming, such as authorized/unauthorized movement alarming, density rate-of-change alarming, and flow rate alarming, refer to *Measurement Integration* (B0193RA).

Absolute Alarming

Absolute alarming produces alarms when the measurement has exceeded or dropped below a user defined high or low limit. There are four types of predefined limits that absolute alarming can check for:

- High-High Absolute Alarming/Target Alarming[2]
- High Absolute Alarming/High Output Alarming[2]
- Low Absolute Alarming/Low Output Alarming[2]
- Low-Low Absolute Alarming.

Figure 7-3 illustrates the principle of absolute alarming.

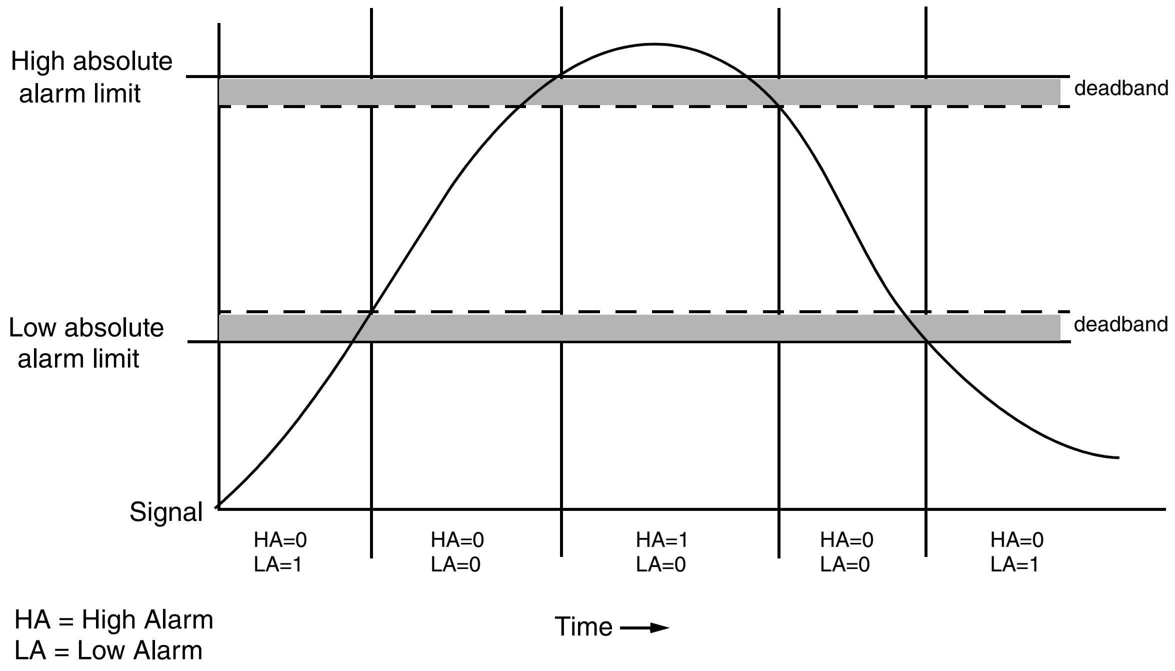


Figure 7-3. Absolute Alarming

There are two user defined settings that you need to configure: high and low alarms limits. Each set of limits must also have a deadband setting associated with it. The deadband is an absolute engineering unit that filters out signal jitter, preventing the signal from bouncing in and out of an alarm state. As the signal passes the defined limits, the alarm is set accordingly.

These four types of absolute alarming are described below.

High Absolute Alarming/High Output Alarming

High absolute alarming is initiated after the signal has climbed above a predefined threshold.

High-high absolute alarming is generally initiated after a high absolute alarm to provide a warning that the signal has continued to climb after it crossed the high absolute threshold.

---

[2]. Different names for the same sub-type of alarming, see below for details.

NOTE

In ACCUM, this type of alarming is called target alarming.

NOTE

Certain blocks, such as BIAS, perform high output alarming, which is identical in function to high absolute alarming. However, high output alarming has its own corresponding bits in ALMSTA, ALMOPT, INHALM, and INHSTA.
In ACCUM, this type of alarming is called pretarget alarming, and is completely identical to high absolute alarming.

In this type of alarming, the signal is compared to the high absolute (or output) alarm limit set for the block. If the signal is greater than this limit, the block:

- ♦ sets its high absolute (or output) alarm indicator to true,
- ♦ generates the alarm via the alarm status parameter, and
- ♦ outputs an alarm message that includes the user-defined high absolute (or output) alarm text.

When the signal falls to, or below, the alarm limit minus the signal deadband, the block sets the appropriate high alarm indicator to false, resets the appropriate bit in the alarm status parameter, and outputs a return-to-normal message.

Low Absolute Alarming/Low Output Alarming

Low absolute alarming is initiated after the signal has climbed below a predefined threshold.

Low-low absolute alarming is generally initiated after a low absolute alarm to provide a warning that the signal has continued to drop after it crossed the low absolute threshold.

NOTE

Certain blocks, such as BIAS, perform low output alarming, which is identical in function to low absolute alarming. However, low output alarming has its own corresponding bits in ALMSTA, ALMOPT, INHALM, and INHSTA.

In this type of alarming, the signal is compared to the low absolute (or output) alarm limit set for the block. If the signal is less than this limit, the block:

- ♦ sets its low absolute (or output) alarm indicator to true,
- ♦ generates the alarm via the alarm status parameter, and
- ♦ outputs an alarm message that includes the user-defined low absolute (or output) alarm text.

When the signal rises to, or above, the alarm limit (plus the signal deadband, in some cases), the block sets its low alarm indicator to false, resets the appropriate bit in the alarm status parameter, and outputs a return-to-normal message.

Bad Input/Output Alarming

Bad I/O alarming is initiated if the device to which the block is connected becomes non-operational (BAD). It is generated via the alarm status parameter.

NOTE

There is some confusion about the difference between a Bad I/O alarm, and a BAD status. For a given block, BAD is the status of a parameter which has its BAD bit set. Status is separate from alarming; it is not an alarm condition. See “Connectable Parameters” on page 4 for more information on BAD (or IO BAD) status.

For blocks which receive redundant signals, such as CINR and DPIDA, the block generates a separate alarm message when any signal is bad, but it sets the Bad I/O Alarm state and BAD parameter only when all of the signals are bad.

Additional events may occur in some blocks. For example, in MTR, the block enters the Bad state and holds the outputs at the last known driven state of the device.

Bad alarming occurs when the Bad Alarm Option (BAO) is set and one or more FBM inputs are bad. Bad alarm messages are generated for each input value independently when its status is Bad.

When the FBM input value becomes bad, a bad alarm message is sent to all devices in the bad alarm group specified by the BAG parameter. This message contains a text string to identify the bad input value. This message also contains the descriptive text in the BAT parameter and the loop identifier in the LOOPID parameter.

When the FBM input value becomes good, a corresponding return-to-normal message is sent to all devices in the bad alarm group.

Bad alarm status information, however, is generated only when the FBM input (or all redundant FBM inputs) are bad. For these cases, the bad parameter (BAD) is set and the BAD bit is set in the alarm status parameter (ALMSTA.BAD). If a State alarm of higher priority does not exist, the CRIT parameter and its corresponding ALMSTA.CRIT field are set to the BAP parameter value, and the PRTYPE parameter and its corresponding ALMSTA.PRTYPE field are set to the Bad alarm type.

When the FBM input (or all redundant FBM inputs) have returned to good status, and if a State alarm of higher priority does not exist, BAD, CRIT, PRTYPE and their corresponding fields in ALMSTA are cleared.

Deviation Alarming

Deviation alarming checks if the difference between setpoint and measurement (deviation) has exceeded a predefined limit. There are two types of predefined limits for which deviation alarming can check:

- High Deviation Alarming
- Low Deviation Alarming

Figure 7-4 illustrates the principle of deviation alarming.

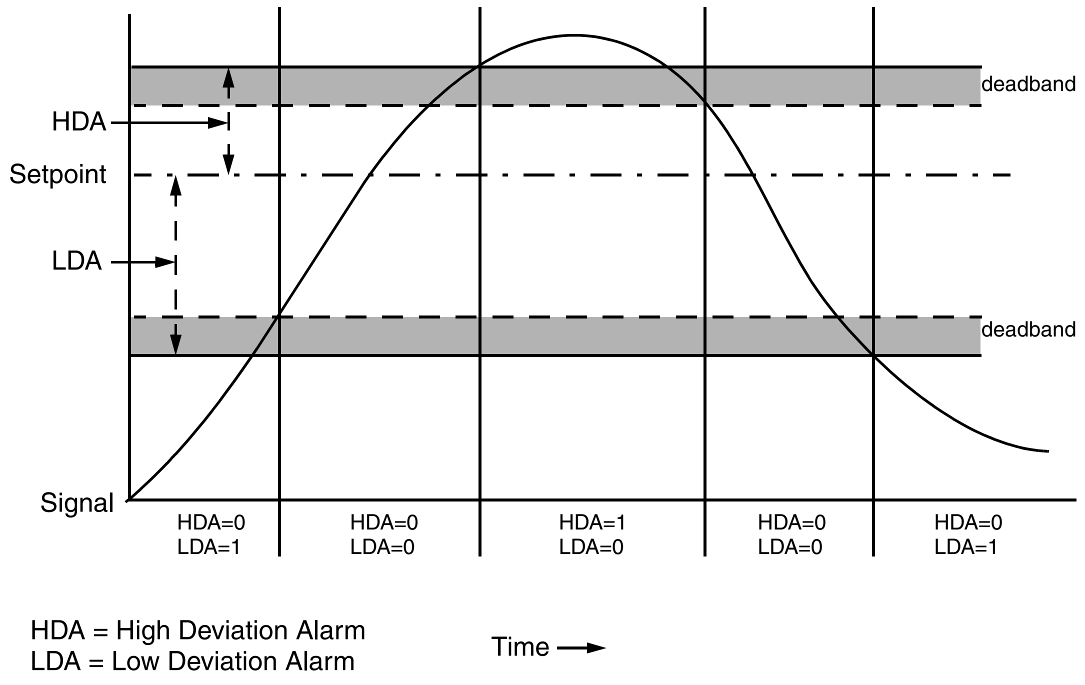


Figure 7-4. Deviation Alarming

In addition to setting the setpoint, you need to set both high and low limits and the deadband. The deadband is an absolute engineering unit that filters out signal jitter, preventing the signal from bouncing in and out of an alarm state. As the signal passes the defined limits, the alarm is set accordingly.

The two forms of deviation alarming are described below.

— NOTE

Note that Deviation = MEAS - SPT and Error = SPT - MEAS.

High Deviation Alarming

In high deviation alarming, the deviation (signal minus the set point) is compared to the high deviation alarm limit set for the block. If the deviation is greater than this limit, the block:

- sets its high deviation alarm indicator true,
- reports the alarm via the alarm status parameter, and
- outputs an alarm message that includes the user-defined high deviation alarm text.

When the deviation falls to, or below, the high deviation alarm limit minus the deadband, the block sets its high deviation alarm indicator to false, resets the appropriate bit in the alarm status parameter, and outputs a return-to-normal message.

The alarm value field of the messages reports the deviation in absolute engineering units at the instant the alarm condition changes to active.

Low Deviation Alarming

In low deviation alarming, the deviation (signal minus the setpoint) is compared to the low deviation alarm limit set for the block. If the deviation is less than this limit, the block:

- sets its low deviation alarm indicator true,
- reports the alarm via the alarm status parameter, and
- outputs an alarm message that includes the user-defined low deviation alarm text.

When the deviation rises to, or above, the low deviation alarm limit plus the deadband, the block sets its low deviation alarm indicator to false, resets the appropriate bit in the alarm status parameter, and outputs a return-to-normal message.

The alarm value field of the messages reports the deviation in absolute engineering units at the instant the alarm condition changes to active.

Mismatch Alarming

Mismatch alarming initiates when the requested state of a device fails to match the actual state of a device within a time interval specified at the block level. The time at which the mismatch timeout begins varies, depending on the block (see the MTR and VLV blocks for examples).

NOTE

Mismatch alarming is reported by the block’s own logic, and not in the ALMSTA parameter.

When a mismatch alarm occurs, the mismatch alarm option parameter is set to true, causing the AUTOPN or MANOPN parameter in the block to be reset to its original state. This allows you to retry the original request action, without having to toggle the request parameter in the wrong direction, by creating a leading edge for the timeout to begin again. Also, an alarm report is generated, using the text in the Name 0 and Name 1 parameters.

When the alarm is acknowledged by the operator, or the field input indicates that the device has changed state as requested, the mismatch alarms are cleared, and return-to-normal messages are generated.

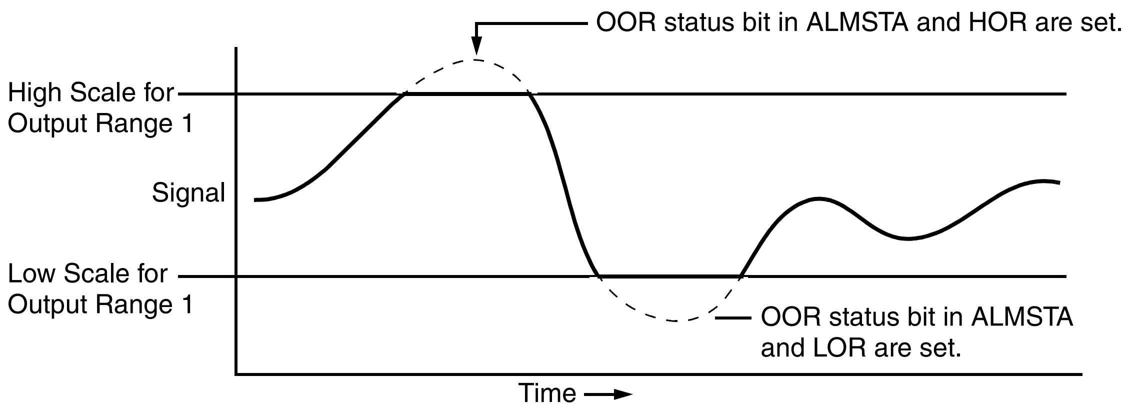
Out of Range Alarming

Out of range alarming initiates when the output exceeds or drops below the specified range set for the block. This type of alarming creates a single alarm when initiated.

However, it can be initiated if:

- the signal exceeds the high scale for output range value, and the high out of range Boolean (HOR) is set, or
- the signal drops below the low scale for output range value, and the low out of range Boolean (LOR) is set.

Figure 7-5 illustrates the principle of out of range alarming.



OOR status bit in ALMSTA and HOR are set.
High Scale for Output Range 1
Signal
Low Scale for Output Range 1
OOR status bit in ALMSTA and LOR are set.
Time

Figure 7-5. Out of Range Alarming

When initiated, this alarm:

- ♦ reports the alarm via the alarm status parameter, and
- ♦ outputs an alarm message to the configured alarm group that includes the user-defined alarm text, and the alarm priority.

When the signal returns between the high and low alarm limits, the block resets the appropriate bit in the alarm status parameter, and outputs a return-to-normal message.

Rate Of Change Alarming

When enabled, rate of change alarming occurs when the magnitude of signal change per period exceeds a configured delta in the rate-of-change limit for the configured ROCTIM. Figure 7-6 shows how the rate of change is determined for a signal value.

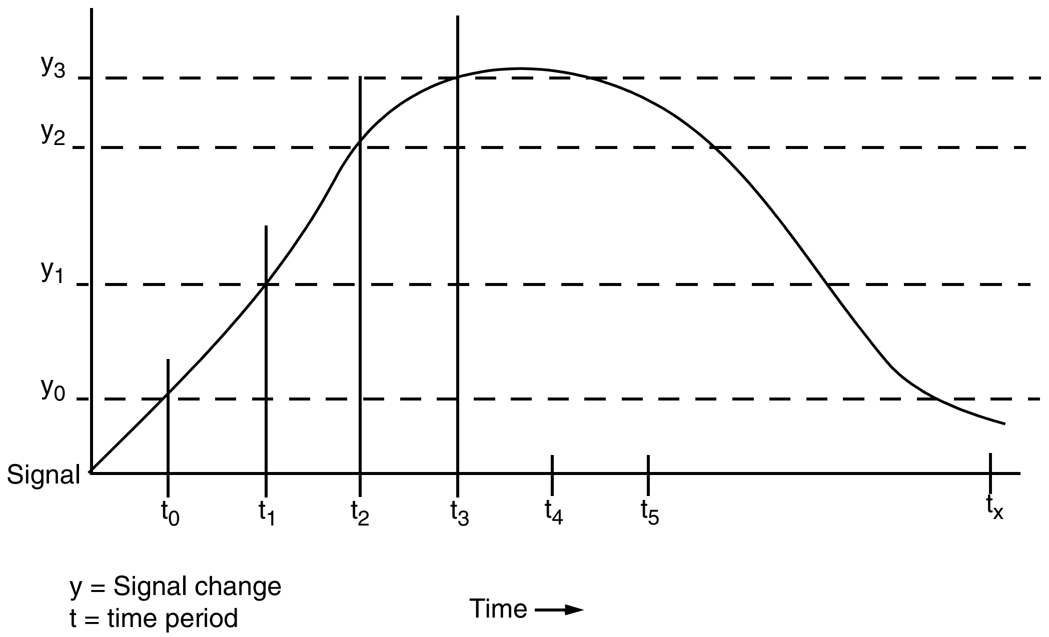


Figure 7-6. Rate of Change Alarming

If the rate of change exceeds the rate-of-change limit over a consecutive time period greater than the ROCTIM time, the block:

- sets the rate of change indicator to true
- reports the alarm via the alarm status parameter, and
- outputs an alarm message to the configured alarm group that includes the user-defined alarm text, and the alarm priority.

When the rate of change no longer exceeds the rate-of-change limit over a consecutive period greater than the ROCTIM time, the rate of change indicator is set to false and the block generates a return-to-normal message.

If ROCTIM = 0 then the block compares changes over 1 time period. For example:

If $|y_1 - y_0| > ROCLIM$ then set the alarm. If $|y_2 - y_1| > ROCLIM$ then alarm stays on.

The alarm returns to normal if $|y_3 - y_2| < ROCLIM$

If ROCTIM = 1 then the block compares changes over 2 time periods. For example:

If $|y_1 - y_0| > ROCLIM\ and\ |y_2 - y_1| > ROCLIM$ then set the alarm.

The alarm returns to normal if $|y_3 - y_2| < ROCLIM$

This type of alarming is performed in blocks such as MEALM.

Sequence Operational Error Alarming

Sequence operational error alarming is performed in sequence blocks, such as MON. MON contains up to 16 user-defined Boolean expressions referred to as monitor cases. Monitor cases can be defined such that when they are evaluated, either to True or to some BAD marked value, the MON block activates a named sequence block, which can be an EXC, IND, or DEP block, or even another MON block.

Sequence operational error alarming is initiated if a monitor case becomes tripped, but the sequence block associated with the monitor case cannot be found. When initiated, the block:

- raises an operational error, and sets the operational error indicator to true
- reports the alarm via the alarm status parameter, and
- outputs an alarm message to the configured alarm group that includes the user-defined alarm text, and the alarm priority.

The alarm is discontinued and a return-to-normal message is issued when the required sequence block activates.

State Change Alarming Using STALM Block

State change alarming is the monitoring of state changes in another block. State change alarming is performed by the STALM block and is initiated when the monitored indicator moves into an alarm state. For example, STALM uses state change alarming to serve as alarm annunciator to activate the Control Core Services alarm mechanism upon alarm conditions detected by an external source.

When initiated, the block:

- reports the alarm via the alarm status parameter, and
- outputs an alarm message to the configured alarm group that includes the user-defined alarm text, and the alarm priority.

The alarm is discontinued and a return-to-normal message is issued when the block detects a transition out of alarm.

State Alarming

State alarming is available for the CIN and CINR contact input blocks. State alarming occurs when the State Alarm Option (SAO) is set and the FBM input transitions from 0 to 1, or 1 to 0 if INVALM is set to 1.

When the input value is in the alarm state, a state alarm message is sent to all devices in the alarm group specified by the SAG parameter. This message also contains the descriptive text in the NM1 parameter and the loop identifier in the LOOPID parameter.

When the input value is no longer in the alarm state, a corresponding return-to-normal message is generated and sent to all devices in the state alarm group (SAG). This message contains the descriptive text in the NM0 parameter.

When the state alarm exists, the SA bit is set in the alarm status parameter (ALMSTA.SA). If a Bad alarm condition of a higher priority does not also exist, the CRIT parameter and its corresponding ALMSTA.CRIT field are set to the SAP parameter value, and the PRTYPE parameter and its corresponding ALMSTA.PRTYPE field are set to the State alarm type.

When the state alarm condition returns to normal status, and if a Bad alarm condition of higher priority does not exist, ALMSTA.SA, CRIT, PRTYPE and their corresponding fields in ALMSTA are cleared.

Trip Alarming

Trip alarming invokes a specific response which differs from block to block when a certain state change occurs.

For EVENT, trip alarming indicates a state change in a Boolean output that is set true when the block reads a new (unreported) event record from its associated FBM.

For MON, trip alarming indicates the BAD bit of a monitor case (see “Sequence Operational Error Alarming” on page 95) is set.

Other blocks have alternate behaviors for trip alarming.

When initiated, the block:

- performs a specific block-dependent action (refer to the block description of the block for more information)
- reports the alarm via the alarm status parameter, and
- outputs an alarm message to the configured alarm group that includes the user-defined alarm text, and the alarm priority.

The alarm is discontinued and a return-to-normal message is issued when the block detects a transition to the original state.

Priority-Change-Based Alarm Message Regeneration

Alarms generated from the control block re-alarm when the alarm priority (PRTYPE) of a block alarm changes (up or down) while it is still active (acknowledged or not). The acknowledgement status of an alarm is preserved if the alarm is changed from a higher to a lower priority. See “Preservation of Alarm Acknowledgement following Re-Alarming” on page 103.

You enable alarm message regeneration based on alarm priority change by setting an option bit (bit 3: 0x08) in the CFGOPT parameter in the Station block in the control processor.

Time-Based Alarm Message Regeneration

Alarms generated from the control block can optionally re-alarm when their re-alarm timer expires while an alarm is still active (acknowledged or not). When the re-alarm message is sent, the block returns to an unacknowledged alarm status. Alarm message regeneration is enabled or disabled for the entire block, and the same time interval applies to all alarms in the block.

To support alarm message regeneration, the alarm message regeneration timer interval (AMRTIN) parameter exists in most blocks that support alarming. This parameter and its attributes are defined in "Common Process Alarm Parameters" on page 77. The user-specified AMRTIN value is rounded to the nearest value that is a multiple of the station BPC for the block. A value of zero disables alarm message regeneration in the entire block for all alarm types.

For blocks supporting multiple alarm types, a separate alarm message regeneration timer based on AMRTIN is maintained for each alarm type listed in Table 7-7 to provide alarm message regeneration by alarm type. For each block type, Table 7-7 specifies the alarm types for which alarm message regeneration is supported.

Time Stamping of Re-Alarmed Messages

There is an option defined in the Station block (CFGOPT Bit 6 (0x40)). If set, this option specifies that re-alarmed messages contain the original timestamp when:

- ♦ an alarm priority is changed. In this case, re-alarming is done only if the Station Block CFGOPT bit 3 (0x08) is also set.
- ♦ a re-alarm timer (AMRTIN) expires.

When Bit 6 of the CFGOPT parameter is not set, the original timestamp of the alarm occurrence is used in all re-alarmed messages sent in reply to a Current State Update (CSU) request.

In all other cases, re-alarmed messages contain new timestamps.

Nuisance Alarm Suppression – Analog Alarms

Nuisance alarm suppression is an optionally enabled feature that allows you to:

- ♦ specify a time interval within which changes in the state of an analog alarm are ignored, and
- ♦ specify the method of delaying the alarm.

Nuisance alarm suppression is implemented at the block level. To support nuisance alarm suppression, the blocks specified in Table 7-7 now include two new parameters:

- ♦ the nuisance alarm suppression time deadband (NASTDB) - Depending on the value of NASOPT, either specifies the deadband time interval that must elapse before an alarm condition is allowed to return to normal, or the length of a delay-on timer which specifies the amount of time between an alarm's detection and the announcement of the alarm.
- ♦ the nuisance alarm suppression option (NASTOPT) - Specifies whether the nuisance alarms will be suppressed by delaying the return-to-normal condition (NASOPT=0) or by delaying the alarm detection (NASOPT=1) by the length of time specified in NASTDB. NASTOPT is ignored when NASTDB is not greater than zero (0).

These parameters and their attributes are defined in "Common Process Alarm Parameters" on page 77.

NOTE

With NASOPT set to 0, nuisance alarm suppression is in addition to and somewhat analogous to the current alarm deadband function. Prior to I/A Series software v8.4, the alarm suppression was done only by delaying the return-to-normal.

You enable the nuisance alarm suppression by configuring NASTDB to a value greater than 0 ms. This value is rounded to the nearest value that is a multiple of the station BPC for the block. Suppression is enabled or disabled for the block as a whole.

For blocks that support multiple alarm types, a separate nuisance alarm message suppression timer based on NASTDB and NASOPT is maintained for certain alarm types to provide nuisance alarm message suppression by alarm type. The same time interval (NASTDB) applies to all supported alarm types in the block. Table 7-7 specifies the alarm types for which these NASTDB and NASOPT features are supported for each block type.

Figure 7-7 shows the time history of a typical high level alarm without nuisance alarm suppression. In this scenario, four alarm active messages are sent. The active alarm messages at times T_2 and T_3 can be considered nuisance messages.

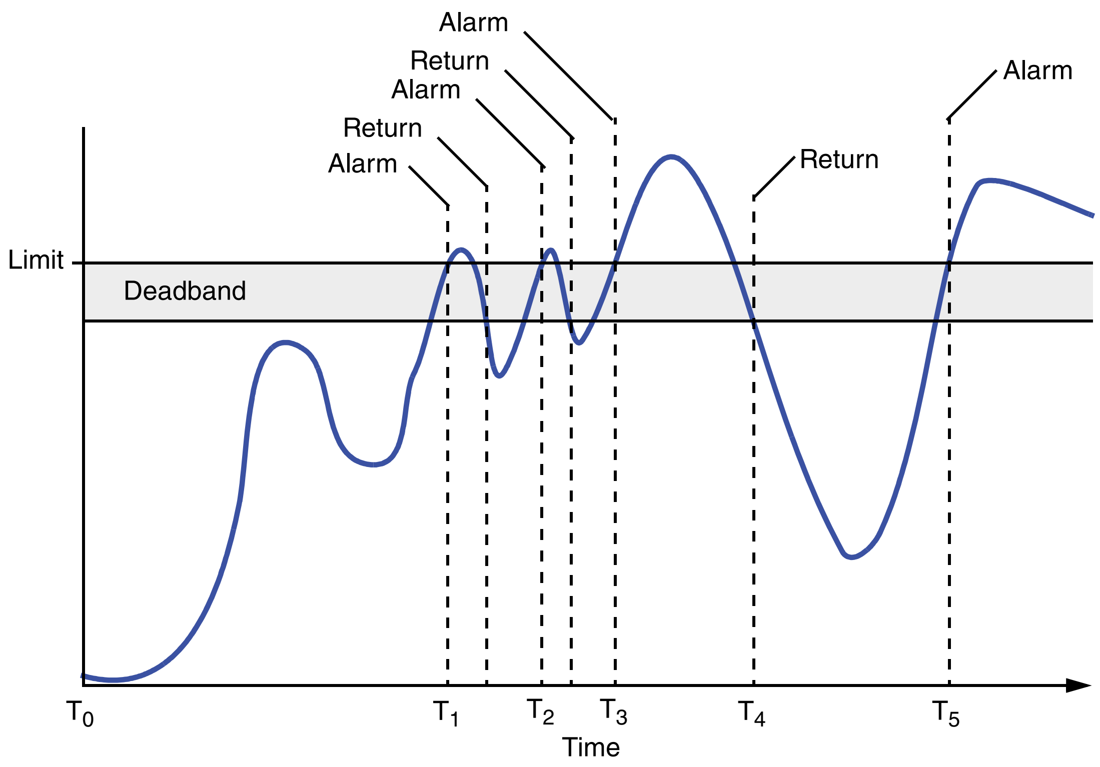


Figure 7-7. Nuisance Alarm Suppression Disabled (NASTDB = 0, NASOPT is Ignored)

Figure 7-8 and Figure 7-9 show the effects of enabling nuisance alarm suppression.

For Figure 7-8, nuisance alarms are suppressed by delaying the return-to-normal condition (NASOPT=0). The suppression deadband is set to a time indicated by the NASTDB parameter. The nuisance messages are suppressed and only the two active messages at times T_1 and T_5 are sent. Changes in the alarm state are ignored until the alarm condition has cleared continuously for the specified NASTDB deadband time.

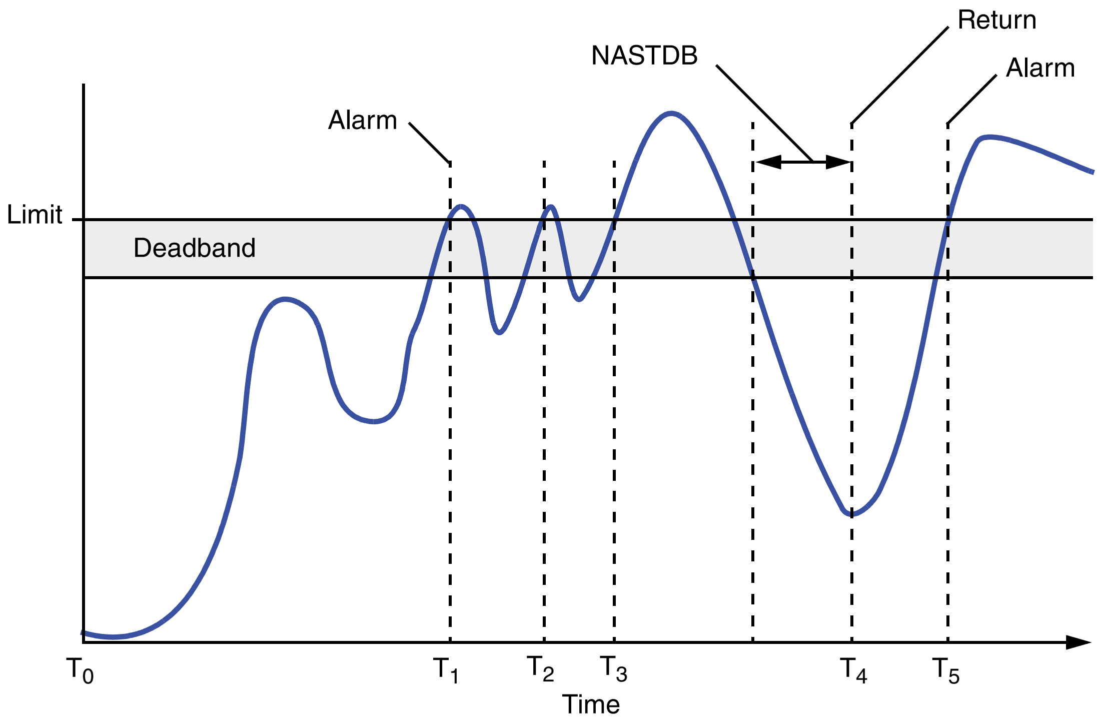


Figure 7-8. Nuisance Alarm Suppression Enabled (NASTDB > 0)

For Figure 7-9 nuisance alarms are suppressed by delaying the alarm detection (NASOPT=1). The original analog alarm at time T_1 remains undetected until the alarm condition exists continuously for the specified NASTDB deadband time. When this happens at time T_3, the alarm state is created and the alarm message generated. The nuisance alarm messages are suppressed, and only the two active messages at times T_3 and T_6 and the return-to-normal message at T_4 are sent.

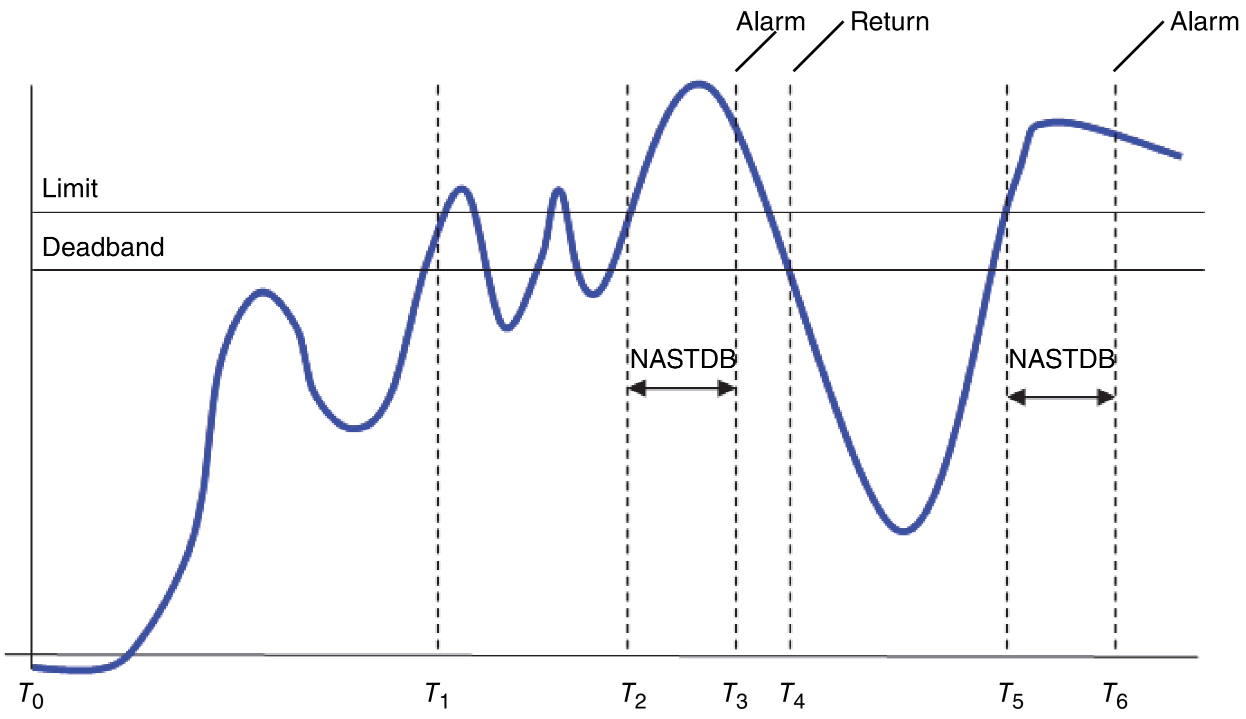


Figure 7-9. Delayed Analog Alarming (NASTDB >0, NASOPT=1)

Alarm Flutter Suppression – Contact Alarms

Nuisance alarms can be suppressed on all contact input alarms. The contact alarm flutter suppression is equivalent to the nuisance analog alarm suppression, as the same two parameters determine the length of delay and the delay mechanism:

- the nuisance alarm suppression time deadband (NASTDB) - Depending on the value of NASOPT, either specifies the deadband time interval that must elapse before an alarm condition is allowed to return to normal, or the length of a delay-on timer which specifies the amount of time between an alarm's detection and the announcement of the alarm. NASTDB also allows for flutter suppression of contact input alarms.
- the nuisance alarm suppression option (NASTOPT) - Specifies whether the nuisance alarms will be suppressed by delaying the return-to-normal condition (NASOPT=0) or by delaying the alarm detection (NASOPT=1) by the length of time specified in NASTDB. NASTOPT is ignored when NASTDB is not greater than zero (0).

These parameters and their attributes are defined in "Common Process Alarm Parameters" on page 77.

You enable contact alarm flutter suppression by configuring the NASTDB parameter to a value greater than 0 ms. This value is rounded to the nearest value that is a multiple of the station BPC in the block. Suppression is enabled or disabled for the block as a whole.

For blocks supporting multiple alarm types, a separate alarm flutter suppression timer is maintained for each alarm type. Table 7-7 specifies the alarm types for which this feature is supported for each block type.

Figure 7-10 and Figure 7-11 show the effects of enabling nuisance alarm suppression.

For Figure 7-10, nuisance alarms are suppressed by delaying the return-to-normal condition (NASOPT=0). In this example, an alarm message is sent at time T_1. The subsequent fluctuations do not cause additional changes in the alarm state until the alarm condition has cleared continuously for the specified NASTDB deadband time. When this happens at time T_4, the alarm state is cleared and a return-to-normal message is generated. In the example, the alarm reactivates at time T_5.

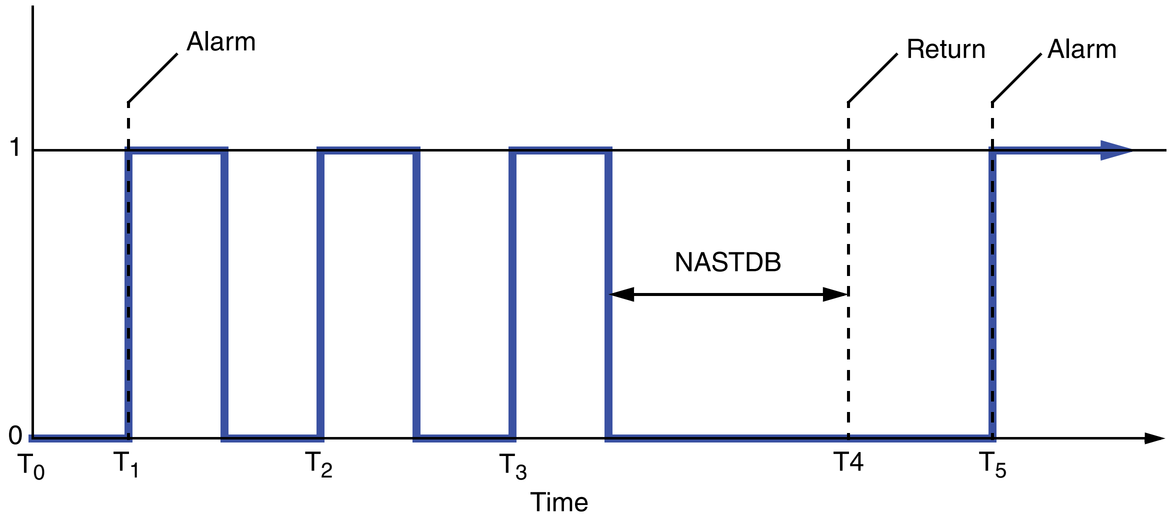


Figure 7-10. Alarm Flutter Suppression Enabled (NASTDB > 0)

For Figure 7-11 nuisance alarms are suppressed by delaying the alarm detection (NASOPT=1). Figure 7-11 shows the effect of enabling alarm flutter suppression by delaying the detection of a contact alarm. The original alarm at time T_1 remains undetected until the alarm condition exists continuously for the specified NASTDB deadband time. When this happens at time T_3, the alarm state is created and the alarm message generated. In the example, the alarm reactivates at time T_5. Note that the alarms that occur at time T_1 and T_2 will be suppressed.

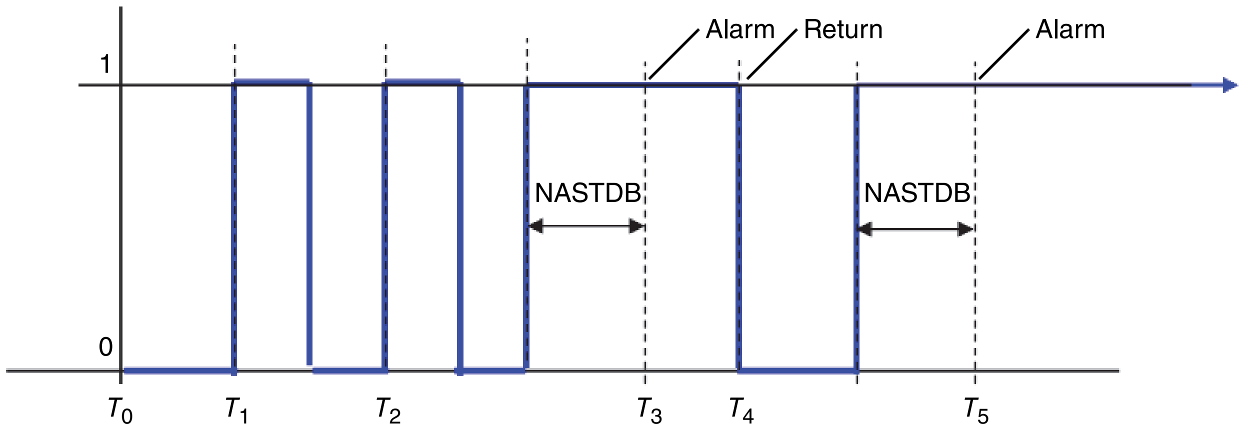


Figure 7-11. Delayed Contact Alarming (NASTDB >0, NASOPT=1)

Summary of Block Alarm Message Regeneration and Suppression

Table 7-7. Block Alarm Message Regeneration and Suppression

Block Type	Alarm Regeneration		Alarm Suppression (NASOPT, NASTDB)
	Priority-Change Based (PRTYPE)	Time-Based (AMRTIN)	
ACCUM	HA (Pre-target Alarm), HHA (Target Alarm)	HA, HHA	----
AI	HHA/LLA, HA/LA, BAD	HHA/LLA, HA/LA, BAD	HHA/LLA, HA/LA
AIN, AINR	HHA/LLA, HA/LA, BAD, OOR	HHA/LLA, HA/LA, BAD, OOR	HHA/LLA, HA/LA
AO	BAD	BAD	----
AOUT, AOUTR	BAD	BAD	----
BIAS	HHA/LLA, HA/LA	HHA/LLA, HA/LA	HHA/LLA, HA/LA
BIN, BINR	SA, BAD	SA, BAD	SA
BLNALM	SA	SA(8)	----
BOUT, BOUTR	BAD	BAD	----
CHARC	OOR	OOR	----
CIN, CINR	SA, BAD	SA, BAD	SA
COUT, COUTR	BAD	BAD	----
DGAP, PTC	HHA/LLA, HA/LA, HDA/LDA	HHA/LLA, HA/LA, HDA/LDA	HHA/LLA, HA/LA, HDA/LDA
DI	BAD, SA	BAD, SA	SA
DO	BAD	BAD	----
EVENT	BAD, TRIPA	BAD, TRIPA	----
GDEV	BAD, SA	BAD, SA	----
IIN, IINR	BAD	BAD	----
IOUT	BAD	BAD	----
MAI, MAO	BAD	BAD	----
MDACT	HHA/LLA, HA/LA, HDA/LDA	HHA/LLA, HA/LA, HDA/LDA	HHA/LLA, HA/LA, HDA/LDA
MEALM	BAD, HHA/LLA, HA/LA	BAD, HHA/LLA, HA/LA	HHA/LLA, HA/LA
MOVLV, MTR, VLV	BAD, SA	BAD, SA	----
PATALM	SA	SA	----
PID, PIDA, PIDE, PIDX, PIDXE, DPIDA	HHA/LLA, HA/LA, HDA/LDA, HOA/LOA	HHA/LLA, HA/LA, HDA/LDA, HOA/LOA	HHA/LLA, HA/LA, HDA/LDA, HOA/LOA
RATIO	HA/LA, HA/LA,	HA/LA, HA/LA,	HHA/LLA, HA/LA
REALM	HA/LA, HA/LA, HDA/LDA, ROC	HA/LA, HA/LA, HDA/LDA, ROC	HHA/LLA, HA/LA, HDA/LDA

Table 7-7. Block Alarm Message Regeneration and Suppression (Continued)

Block Type	Alarm Regeneration		Alarm Suppression (NASOPT, NASTDB)
	Priority-Change Based (PRTYPE)	Time-Based (AMRTIN)	
RIN, RINR	HHA/LLA, HA/LA, BAD, OOR	HHA/LLA, HA/LA, BAD, OOR	HHA/LLA, HA/LA
ROUT, ROUTR	BAD	BAD	----
STALM	BAD, SA	BAD, SA	----

Note:

BAD = Bad Alarm	PTA = Pre-Target Alarm (High Absolute)
HA/LA = High/Low Absolute Alarm	ROC = Rate-of-Change Alarm
HDA/LDA = High/Low Deviation Alarm	SA = Discrete State Alarm
HHA/LLA = High-High/Low-Low Absolute Alarm	TA = Target Alarm (High-High Absolute)
HOA/LOA = High/Low Output Alarm	TRIPA = Trip Alarm
OOR = Out-of-Range Alarm	

Preservation of Alarm Acknowledgement following Re-Alarming

In the FCP280 or CP270 database, every block that supports alarms contains a single Alarm Acknowledge status. When a new alarm condition arises, this status is set to Unacknowledged (UNACK = 1). When the operator acknowledges the alarm, this status is set to Acknowledged (UNACK = 0).

Re-alarmed messages caused by a Current State Update (CSU) request preserve the Alarm Acknowledge status of the block at the time of the CSU request. If the block had been Acknowledged prior to the CSU request, it remains Acknowledged following the CSU request.

Re-alarmed messages caused by the expiration of the Alarm Message Regeneration Timer (AMRTIN) are treated as new alarms and cause the Alarm Acknowledge status to be set to Unacknowledged (UNACK = 1). If the block had been Acknowledged prior to the expiration of the timer, it becomes Unacknowledged following the timer expiration.

Re-alarmed messages caused by reprioritizing an alarm are treated as a new alarm and cause the Alarm Acknowledge status to be set to Unacknowledged if the priority is raised (for example, changed from 5 to 2). If the block had been Acknowledged prior to the reprioritization, it becomes Unacknowledged following the reprioritization.

Re-alarmed messages caused by reprioritizing an alarm preserves the Alarm Acknowledge status of the block if the priority is lowered (for example, changed from 1 to 3). If the block had been Acknowledged prior to the reprioritization, it remains acknowledged following the reprioritization.

For example, if an alarm has been generated and acknowledged, lowering its priority preserves the Acknowledged block alarm state, but raising its priority sets the block alarm state to Unacknowledged.

Preservation of Alarm Acknowledgement for Multiple Alarm Priority Types

If more than one alarm priority type is enabled for a single block, there are cases where an acknowledged alarm can require an additional operator acknowledgement if an alarm priority is lowered. This behavior occurs because a block has only one boolean acknowledgement status parameter but can have multiple alarms with different priorities.

AIN Block Example

For example, take a situation where an AIN block has multiple active acknowledged alarms (UNACK=0). Raising one alarm's priority causes the block to go to the unacknowledged state (UNACK=1) as expected. Then, if you lower the priority of another alarm from the same block, the unacknowledged state of the block that existed before alarm reprioritization is preserved, UNACK=1. Therefore, changing the priority of the second alarm in this case has no effect on the block alarm state; the block remains in the unacknowledged state and an operator acknowledgement is required. The event history follows:

1. The AIN block has the following alarms tripped:
   - ♦ High Absolute Alarm Indicator (HAI)
   - ♦ High-High Absolute Alarm Indicator (HHAIND)
   - ♦ High Out-of-Range Alarm (HOR)
   - ♦ Bad input (BAD).
2. Alarms are acknowledged.
3. Out-of-Range alarm priority (ORAP) is raised from 3 to 2. This action causes the block alarm state to be set to unacknowledged.
4. High/Low alarm priority (HLPR) is lowered from 2 to 3. This action has no effect on the block alarm state, which remains unacknowledged.

When ORAP was raised, the unacknowledged state was set. Lowering HLPR does not change the unacknowledged state of the block, which will be preserved.

8. Ladder Logic Concepts

This chapter discusses ladder logic and ladder logic diagrams in depth, as well as fanned outputs, zone control logic, programmable logic (PLB) block operation, and gives a PLB editor overview.

Ladder Logic Overview

Ladder logic diagrams use symbols traditionally associated with relay circuits. These symbols, consisting of contacts, timers, counters, and coils, are used to build rungs. The rungs are built into ladders.

Ladder logic capability is a configurable option in digital Fieldbus Modules, where the Ladder diagrams are processed. A Programmable Logic Block (PLB) in the control processor supports ladder logic operation, integrating it with control block processing. See below for a simplified diagram showing the interaction of the main Ladder Logic components.

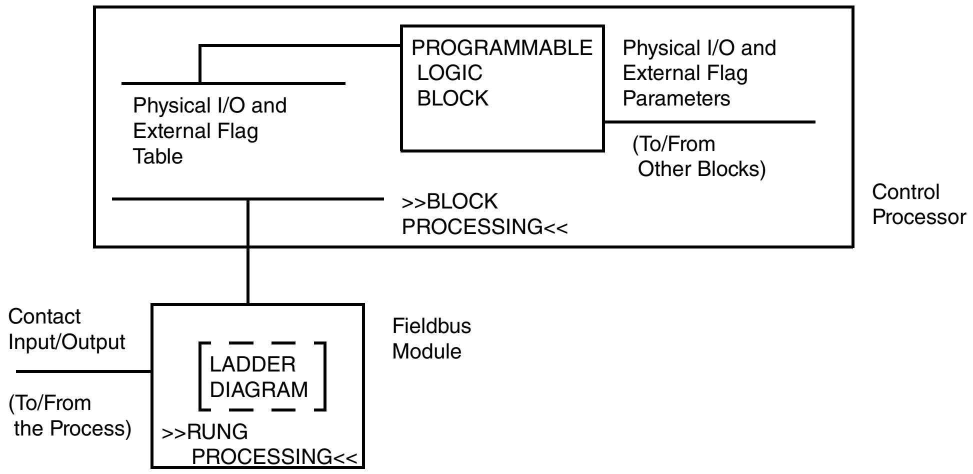


Figure 8-1. Ladder Logic Components (Simplified)

Ladder Diagram Functions

Ladder diagram rungs sense the status of digital inputs and control the status of digital outputs.

Inputs to ladder logic come from physical input points connected to the field terminals of the Fieldbus Module or from external input flags supplied by control processor blocks. Physical inputs present process status information from devices such as limit switches, push buttons, and pressure switches. External input flags allow continuous and sequential blocks and user tasks to initiate or control ladder logic action.

Outputs from ladder logic go to physical output points connected to the field terminals of the Fieldbus Module or to physical output parameters and external output flags supplied to control processor blocks. Physical outputs control devices such as solenoids, motor starters, and indicator lamps. Physical output parameters and external output flags allow continuous and sequential blocks and user tasks to monitor ladder logic action.

PLB Functions

A PLB block connects user tasks, other blocks, and other ladder diagrams with:

- ♦ A ladder diagram's external I/O flags
- ♦ A digital Fieldbus Module's physical inputs and outputs.

Each PLB block defines a ladder diagram.

In the Integrated Control Configurator (ICC), creating a PLB establishes a ladder diagram source file. Using this source file, the PLB ladder logic editor lets you construct a ladder diagram, check it for syntax errors, and produce a printed copy for documentation. You can choose to have the ladder diagram code installed in a digital Fieldbus Module or save the source file for later use. You can develop a library of ladder diagrams and retrieve (copy) them for installation in multiple Fieldbus Modules. You can also save ladder diagram source files (as part of a compound) on diskette.

To create ladder logic and assign it to a PLB using the I/A Series Configuration Component (IACC), refer to *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

The ladder diagrams remain "generic", independent of hardware, until the associated PLB block is assigned to a specific Fieldbus Module. The elements then become bound to that hardware component.

Through the PLB block's detail (default) display, you can monitor the status of ladder logic contacts, timers, counters, and coils. Alternatively, you can create your own graphic displays. Displays that you generate access the status of ladder logic elements through external flag parameters. The PLB's detail display allows you to force contacts and coils on or off to verify correct operation of the logic under simulated process conditions.

The ladder logic editor configures and checks at the block level; not at the FBM level. Since up to eight programmable logic blocks (PLBs) can be configured to run in the same FBM, you must consider certain interactions and constraints. These include:

- ♦ All ladders in a given FBM use a single coil table or table of parameter values. The available ladder components or "technical identifiers" constitute a fixed pool of resources associated with the final concatenated ladder. Ensure that one ladder does not conflict with the outputs of another, for example, turning off an output that another ladder has just turned on.
- ♦ FBM state control applies to all ladders in the FBM.
- ♦ The FBM's 16 timer/counters are available to all ladders in the FBM. Plan for the consistent use and selection of these 16 components across ladder boundaries. Always reset timers and counters prior to use.
- ♦ Once you configure a timer/counter with a specific technical ID, you cannot associate that ID with a different timer/counter.
- ♦ The total number of ladder logic lines for all ladders in a given FBM must be less than 98.
- ♦ The FBM can accommodate approximately 850 bytes of object code representing the ladder logic. This can support about 390 user-entered ladder symbols. (Straight line horizontal and vertical segments can be considered as using no memory.)

Ladder Diagram Description

Input parameters are represented as contact symbols (Boolean inputs) on the ladder diagram. Output parameters are represented as coil symbols, including relays, timers, counters, and conditional coils.

Typical symbols:

——|/|——	Normally Closed Contact
——| |——	Normally Open Contact
——( )——	Coil

A ladder logic diagram is made up of rungs. A ladder rung is a logical element with one or more input conditions and a single output value. (Outputs can be paralleled for fanout.)

When drawn, the logic consists of two conceptual vertical rails – power source on the left and return on the right. (These rails do not appear visibly in ladder displays.) Coils, timers, and counters attach directly to the right-hand rail. Contacts complete a power flow path from the left-hand rail to the timer, counter, or coil

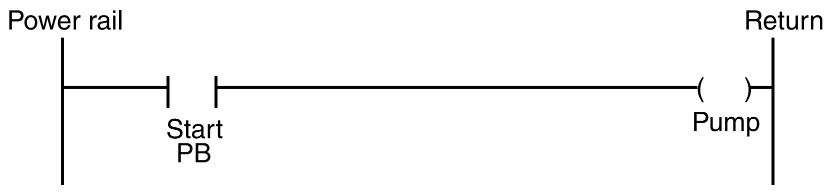


Figure 8-2. Simplified Ladder Logic Diagram

Ladder logic is executed rung by rung from top to bottom of the ladder. Ladders are executed at an average of 300 ladder logic symbols in 20 to 25 ms. During execution, each contact symbol in the rung is examined and compared with the Boolean value referenced by its technical identifier. Power flow is enabled through the contact in the following situations:

- The contact is represented as normally-open on the ladder diagram and the referenced input is true (present or active).
- The contact is represented as normally-closed on the ladder diagram and the referenced input is false (not present or inactive).

Ladder logic flows from left to right, mimicking power flow from source to destination.

As each ladder logic rung is executed, if the enabled contacts provide a complete path from the left margin (power rail) to the output coil (or paralleled coils) connected to the right margin (return rail), then the coil is “activated” (that is, the Boolean value for its associated technical identifier is set true). The coil remains active until the next cycle when that rung is scanned and the inputs are again analyzed. If the power flow path no longer exists, the coil is deactivated, unless the coil is a latching type.

Latching coils, once energized, remain active until an unlatching coil with the same technical identifier is energized.

You place logic symbols strategically on the ladder rungs to depict a combination of conditions or sequence of events. This placement determines the control logic.

The contacts can be connected in series in a simple circuit where all conditions must be met for the path to be completed (ANDing of conditions). A normally open contact provides a path when the condition for which it is named is true. A normally closed contact provides a path when the condition for which it is named is false. In this series network, when all the conditions represented by normally open contacts are true and all the conditions represented by normally closed contacts are false, the path is complete.

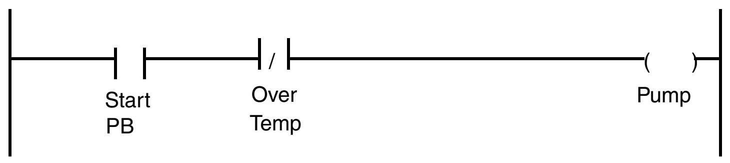


Figure 8-3. Ladder Logic Contacts

A row contains up to seven contact symbols; the last position is reserved for a coil symbol. Horizontal "connector" symbols provide power flow through positions in a row where no contact is configured.

Rows may be interconnected through a vertical connector symbol. A ladder rung comprises all rows (branches) connecting to a given coil, timer, or counter, or fanout of these. You can scroll vertically to view a ladder diagram containing more rungs than the screen can display at one time.

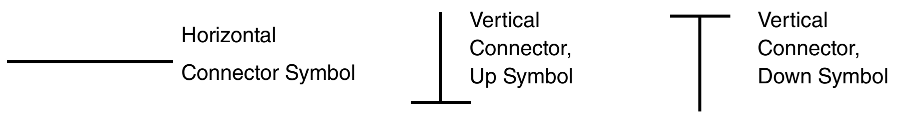


Figure 8-4. Ladder Logic Connectors

The contacts can be connected in a series, parallel circuit where the parallel portion provides an alternate set of conditions that allow the path to be completed (ORing of conditions). For example, the main leg could provide for activating a coil, and the parallel leg provide a path to keep it activated. When logic continuity exists in at least one path, the rung is true or has a GO condition.

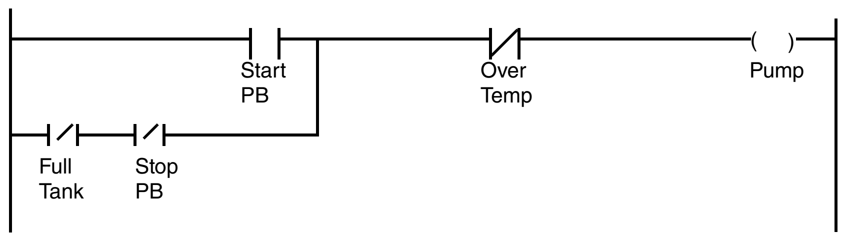


Figure 8-5. Parallel Circuits in Ladder Logic

Ladder Diagram Display

As displayed, the ladder diagram omits the left and right rails; their existence is implied.

The following figure shows a typical ICC ladder diagram display.

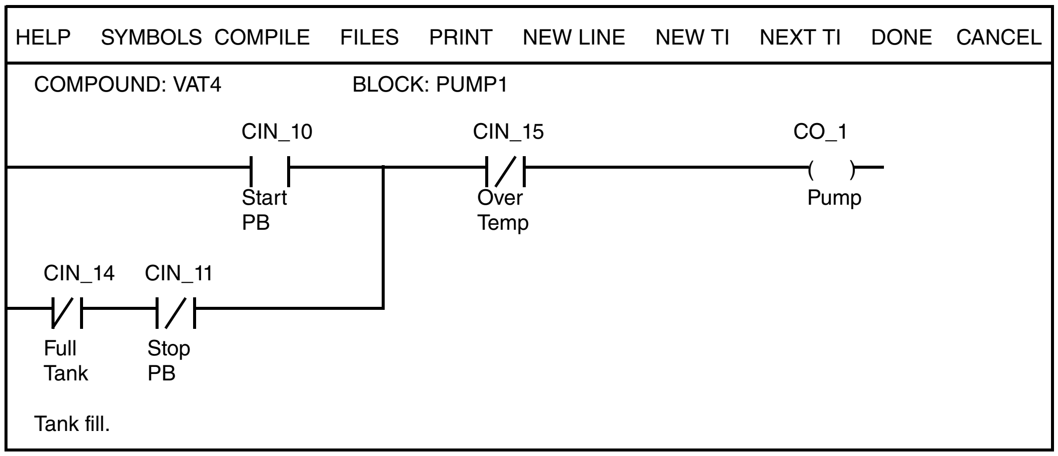


Figure 8-6. Typical ICC Ladder Diagram Display

Ladder Diagram Execution

The Fieldbus Module ladder logic executive performs logical operations based on the placement of the contacts and coils in the ladder diagram.

Rungs are executed in sequence, in an infinite loop, from beginning to end of the ladder. Each execution of the loop is termed a scan. As each rung is processed, contact status is evaluated and if a path exists, the rung output is activated. The actual state of the physical input associated with the contact is recognized when the rung is executed.

To allow for replacing the program without taking the control off line for an extended period, the Fieldbus Module provides two application program spaces. This allows an existing program to execute from one while a revised program is loaded into the other. When the download completes, the FBM software, at the end of a scan through the ladder, switches to the new logic.

Ladder Diagram Construction Using the ICC

— NOTE

To create ladder logic using the IACC, refer to *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

In the ICC, you build your ladder diagram in a work area occupying all of the screen below the menu bar, except for a column at the right for timer/counter preset and reset values. Figure 8-7 shows the work area that is reserved for the ladder rungs and the logic symbols.

The ladder logic editor lets you create ladder diagrams from a set of predefined symbols and identifiers, preset and reset values for counters and timers, and your choice of text labels for symbols and rungs. A ladder diagram row accepts up to seven series-contacts and a coil.

From the menu bar you can select SYMBOLS to display the symbols and function key assignments which create these or HELP to display the instruction set and technical identifiers that can be used for constructing a ladder.

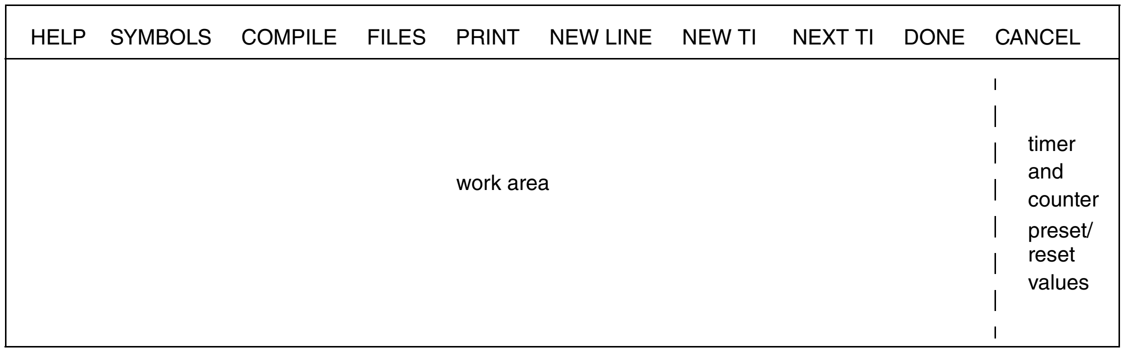


Figure 8-7. ICC Ladder Diagram Work Area

Symbols

Table 8-1 below shows the ladder instruction set used to implement ladder logic.

Table 8-1. Ladder Symbols

Symbol	Name	Description
—| |—	Normally Open Contact	Provides power flow from left to right when the named signal is present (true state).
—|/|—	Normally Closed Contact	Provides power flow from left to right when the named signal is absent (false state).
———	Connector	Provides power flow through a symbol position.
┬	Vertical Connector (down)	Joins two rows when used together with a matching vertical connector (up).
┴	Vertical Connector (up)	Joins two rows when used with a vertical connector (down). Used in pairs, vertical connectors provide power flow vertically within a ladder diagram.
—( )—	Energize Coil	Sets the Boolean value representing coil status true if the rung has a power flow path. If the path is lost, the Boolean value is set false.
—(/)—	Write Not Coil	Sets the Boolean value representing coil status false if the rung has a power flow path. If the path is lost, the Boolean value is set true.
—(L)—	Latch Coil	Sets the Boolean value representing coil status true if the rung has a power flow path. If the path is lost after the coil is set, the Boolean value remains true until the associated Unlatch Coil is set.
—(U)—	Unlatch Coil	Unlatches an output that was previously set by a Latch Coil instruction.

Table 8-1. Ladder Symbols (Continued)

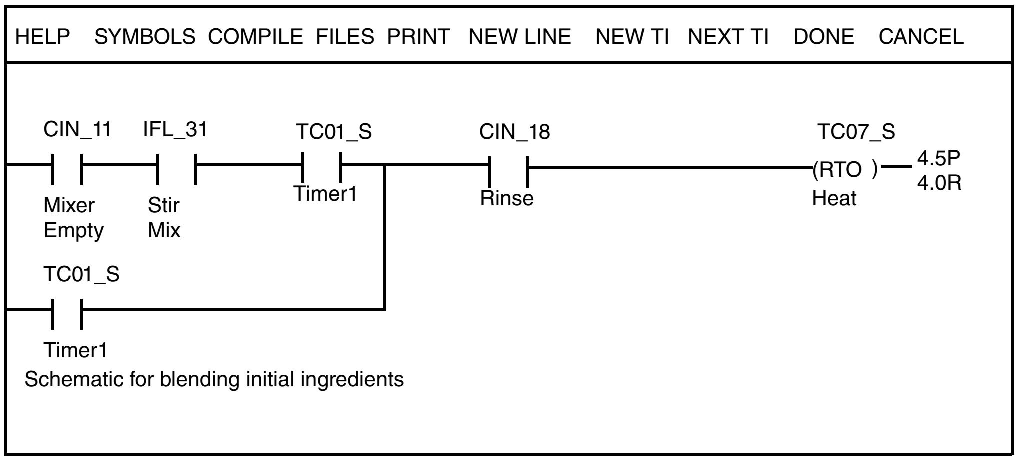
Symbol	Name	Description
—(CTU)—	Up Counter	Increments a counter on off-to-on transitions.
—(CTD)—	Down Counter	Decrements a counter on off-to-on transitions.
—(RST)—	Counter/Timer Reset	Resets a counter or timer having the same technical identifier as this symbol.
—(MCR)—	Master Control Relay	Enables the rungs between this symbol and the NCR symbol to execute normally if the MCR rung condition is true. If MCR is false, the area rungs are not executed and the nonretentive outputs within the area are de-energized.
—(NCR)—	End of Master Control Relay Zone	Marks the end of the MCR conditional group of rungs. This symbol must be alone on its rung.
—(ZCL)—	Zone Control Logic	Enables the rungs between this symbol and the NCL symbol to execute normally if the ZCL rung condition is true. If ZCL is false, the rungs in the zone are not executed and the outputs are held at their last state.
—(NCL)—	End of Zone Control Logic	Marks the end of the ZCL conditional group of rungs. This symbol must be alone on its rung.
—(RTO)—	Retentive timer-on delay	Indicates the rung has been true (POWER TRUE) for at least the specified delay period.
—(RTF)—	Retentive timer-off delay	Indicated the rung has been false (POWER FALSE) for at least the specified delay period.
—(TON)—	Non-retentive timer-on delay	Same as RTO except that when the rung is false at any time, the accumulated time is zeroed.
—(TOF)—	Non-retentive timer-off delay	Same as RTF except that when the rung is true at any time, the accumulated time is zeroed.

Text and Numeric Entries

Two types of text entries identify each contact, coil, and timer or counter in a ladder diagram. A technical identifier is placed above each symbol and a label for it is optionally placed below the symbol. An optional rung descriptor can be placed beneath each rung as shown in Figure 8-8 (the rung descriptor is “Schematic for blending initial ingredients”).

A set of numeric entries is associated with each counter (CTU or CTD) or timer symbol (RTO, TON, RTF, or TOF). These entries, located in the right hand column of the ladder diagram consist of a preset value identified by a “P” suffix, a reset value identified by an “R” suffix; and an accumulated value identified by an “A” suffix. Accumulated values are not entered or displayed by the PLB editor.

Table 8-2 describes the text and numeric elements briefly. Table 8-3 lists the technical identifiers.



Schematic for blending initial ingredients

Figure 8-8. ICC Sample Ladder Diagram

Table 8-2. Text and Numeric Elements

Text and Numeric Elements	Description
Technical Identifier	A single row of up to six upper case characters that identifies an element by type and number (see Table 8-3).
Label	Two rows of up to seven characters each, beneath the element.
Preset Value	For timer or counter symbol, maximum value, up to 65,535. Timer values represent tenths of seconds for a maximum of 6553.5 seconds; counter values represent counts.
Reset Value	For timer or counter symbol, minimum value, up to 65,535, set in at reset time. Timer values represent tenths of seconds for a maximum of 6553.5 seconds; counter values represent counts.
Rung Descriptor	Up to three lines of text with up to 60 characters in each. Used for documentation as the previous diagram shows. Each ladder can have a maximum of twenty rung descriptors.

Inputs to and outputs from ladder logic are stored as Boolean values referenced by technical identifiers. These identifiers are chosen from a list of valid names (see Table 8-3).

Table 8-3. Technical Identifiers

Technical Identifiers	Meaning
CIN_1 through CIN_32	Physical Inputs
CO_1 through CO_16	Physical Outputs
IFL_1 through IFL_32	External Input Flags
OFL_1 through OFL_32	External Output Flags
TC01_S through TC16_S	Timer/Counter Coils
TC01_S through TC16_S	Timer/Counter Status Contacts

Table 8-3. Technical Identifiers (Continued)

Technical Identifiers	Meaning
TC01_O through TC16_O	Timer/Counter Overflow Contacts
INT_01 through INT_32	Internal Flags
INIT	Initialize Flag
POWERF	Power Fail Flag
COMMF	Communications Failure Flag
FAILSF	Fail-Safe Flag

The user-defined label consists of two text fields that may contain any comments (seven characters per field) to enhance the readability of the ladder diagram. Each label is associated with the technical identifier, not with the symbol, as may be seen by the following rung example.

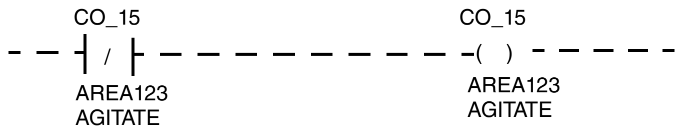


Figure 8-9. User-Defined Labelling

Ladder Diagram Elements

Inputs (Contacts)

Ladder diagram inputs are represented as (relay type) contacts on the ladder diagram. These inputs include the following:

- ♦ External input flag
- ♦ Internal coils
- ♦ Contact input
- ♦ External output flag
- ♦ Contact output
- ♦ Timer/Counter flag

Within the ladder diagram, an input can be represented as a normally open (NO) or a normally closed (NC) contact. The technical identifier assigned to the contact indicates the source of the input, as shown in Table 8-4. The input (with its technical identifier) can be used in the ladder as many times as desired in NO and NC states.

Table 8-4. Ladder Inputs

Type of Input	Applicable Instructions	Contact Symbols	Technical Identifier	Source
External Input Flag	Normally Open Normally Closed	-| |- -|/|-	IFL_nn	Ladder Logic Interface Block (PLB Block).

Table 8-4. Ladder Inputs (Continued)

Type of Input	Applicable Instructions	Contact Symbols	Technical Identifier	Source
Contact Input	Normally Open Normally Closed	-| |- -|/|-	CIN_nn	Fieldbus Module Physical Input.
Contact Output	Normally Open Normally Closed	-| |- -|/|-	CO_nn	Energize Coil, Write Not Coil, and Latch Coil instructions.
Internal Flag	Normally Open Normally Closed	-| |- -|/|-	INT_nn	Energize Coil, Write Not Coil, and Latch Coil instructions.
External Output Flag	Normally Open Normally Closed	-| |- -|/|-	OFL_nn	Energize Coil, Write Not Coil, and Latch Coil instructions.
Timer/ Counter	Normally Open Normally Closed	-| |- -|/|-	TCnn_S TCnn_O	Up Counter, Down Counter, Retentive Timer On-Delay, Retentive Timer Off-Delay, Non-Retentive Timer On-Delay, and Non-Retentive Timer Off-Delay

The control processor can obtain the status of a ladder logic physical input through the CIN_nn parameter of the PLB block.

Outputs (Coils)

Outputs are represented as coils (see Table 8-5). The symbol and technical identifier assigned to the coil determines the type of output provided:

- Contact output
- External output flag
- Internal flag
- Timer
- Counter.

Contact outputs control the status of Fieldbus Module output channels.

External output flags allow control processor blocks to monitor the status of a ladder rung.

Internal flags allow you to create AND conditions of more than seven elements in a single ladder rung by cascading INT_nn elements.

Timer/counter flags allow you to condition rung outputs on the passage of a specified length of time or the occurrence of a specified number of events.

Within the ladder diagram, any of these outputs can be represented as a normally open or a normally closed contact and used in both forms as many times as needed. However, a coil provides only one physical output (CO_nn). To send output to an additional Fieldbus Module channel, you would configure an additional coil, with its own technical identifier, in parallel with the first.

Table 8-5. Ladder Coils

Type of Output	Applicable Instructions	Coil Symbols	Technical Identifier	Primary Destination
Contact Output	Energize Coil Write Not Coil Latch Coil	-( )- -(/)- -(L)-	CO_nn	Fieldbus Module physical output.
External Flag Output	Energize Coil Write Not Coil Latch Coil	-( )- -(/)- -(L)-	OFL_nn	Ladder Logic Interface block (PLB Block).
Internal Flag	Energize Coil Write Not Coil Latch Coil	-( )- -(/)- -(L)-	INT_nn	Internal
Timer	Retentive Timer-On Delay Retentive Timer-Off Delay Non-retentive Timer-On Delay Non-retentive Timer-Off Delay	-(RTO)- -(RTF)- -(TON)- -(TOF)-	TCnn_S	Internal
Counter	Up Counter Down Counter	-(CTU)- -(CTD)-	TCnn_S	Internal

Counters and Timers

The ladder logic Fieldbus Module has timer/counter structures numbered 1 through 16. To implement a timer or counter, you configure it as one of the following types:

Table 8-6. Counters and Timers

Category	Type	Symbol
Transition Counters	Up Counter Down Counter	--(CTU)-- --(CTD)--
Retentive Timers	On Delay Off Delay	--(RTO)-- --(RTF)--
Non-retentive Timers	On Delay Off Delay	--(TON)-- --(TOF)--

— NOTE

All timers and counters use a "TC" technical identifier prefix, followed by a number from 01 through 16, inclusive. For example, TC01 or TC16.

Table 8-7. Counter/Timer State Terminology

Term	Description
Ladder Rungs	Evaluate to true or false.
Counters/Timers	Are reset (contain the configured reset value), counting or timing, count satisfied or delay timed out (at or above the configured preset value), or overflow (at or beyond the maximum or minimum count or time value)
Flags	Are either true (raised, set, or 1-state) or false (lowered, reset, cleared, 0-state).
Coils	Are either Energized (activated) or De-energized (deactivated).

Associated with each timer/counter are two flags for communicating with the ladder and three registers for controlling timer/counter operation.

Each timer or counter has a status flag and an overflow flag. The status flag is set when the pre-specified count or time delay has been satisfied. The overflow flag is set when the maximum or minimum count or delay time has been exceeded, and wrap has occurred.

Each timer or counter has three 16-bit registers, supporting counts from 0 to 65,535 and delay times from 0 to 6,553.5 seconds. The registers store an initial value, a final value, and a current, accumulated value. You specify initial and final values during configuration. Refer to Table 8-8.

Table 8-8. Timer/Counter Registers and Flags

Category	Type	Description
Registers	Accumulated Count/Time	The number of counts or ticks since the last reset (not displayed) in the ladder editor.
	Preset Value	The target value for timers and counters; set through ladder configuration.
	Reset Value	The initial value for timers and up counters; set through ladder configuration and loaded by an RST operation.
Flags	Status Flag	Set if the accumulated value is equal to or greater than the preset value for a timer or up counter, or less than the preset value for a down counter. The status flag remains set until the timer/counter is reset.
	Overflow Flag	Set on transition from 65,535 to 0 when counting up (overflow) or from 0 to 65,535 when counting down (underflow). When overflow is set, counts continue to accumulate via wrapping, but the status flag retains the value it had when the overflow or underflow occurred. The overflow flag remains set – and the status flag frozen – until the timer/counter is reset.

Transition Counters

Up counters operate on rung transitions from false to true. Starting from a configured reset value, an Up counter increments on each transition toward a configured preset value.

Down counters operate on rung transitions from false to true. Down counters are SET immediately following a reset signal to reset its value. When the count decrements and counts past the preset value, the status is set FALSE. Once the count has been satisfied, the counter's status flag is set and remains so until the Reset coil for that counter is energized.

Retentive Timers

A retentive timer indicates, through its status flag, that its rung has been in a particular state for at least the specified delay period. The delay period is determined as the difference between the preset and reset values configured for the timer. Starting from the reset value, the timer counts clock pulses or ticks that occur every tenth of a second. This type of timer retains the accumulated count during intervals when its rung state is false.

An on-delay timer counts ticks whenever its rung state is true. As long as the delay is in effect, the timer's status flag is false. When the accumulated count equals the preset value, the timer's status flag is set true.

An off-delay timer counts ticks whenever its rung state is false. As long as the delay is in effect, the timer's status flag is true. When the accumulated count equals the preset value, the timer's status flag is set false.

Once the delay has been satisfied, the timer's status flag retains its state until the timer's reset rung is set true. Regardless of timer rung state, the count for either type of timer is forced to the reset value and the timer's status flag held false whenever the timer's reset rung is true.

Table 8-9 for On-Delay and Table 8-10 for Off-Delay timers show the cause and effect relationships among Reset coil state, timer rung state, accumulated tick count, and reset/preset values. This section has two figures (for the On-Delay and the Off-Delay timers) that show the counting action with various changes in timer rung state.

Table 8-9. RTO Truth Table

Reset Rung (RST Coil)	On-Delay Timer Rung (RTO Coil)	Accumulator Register	Status Flag (_S Coil)	Timer Advance	Overflow Flag (_O Coil)
True*	Don't care	= Reset Value*	False*	No	False*
False	True*	> Reset Value	False	Yes*	False
False	False*	> Reset Value	False	No*	False
False	True*	> Reset Value	False	Yes*	False
False	True	= Preset Value*	True*	Yes	False
False	True	> 6,553.5*	True	Yes	True*

* Note that in each row, the emphasized entry initiates change in other asterisked entries in the same row.

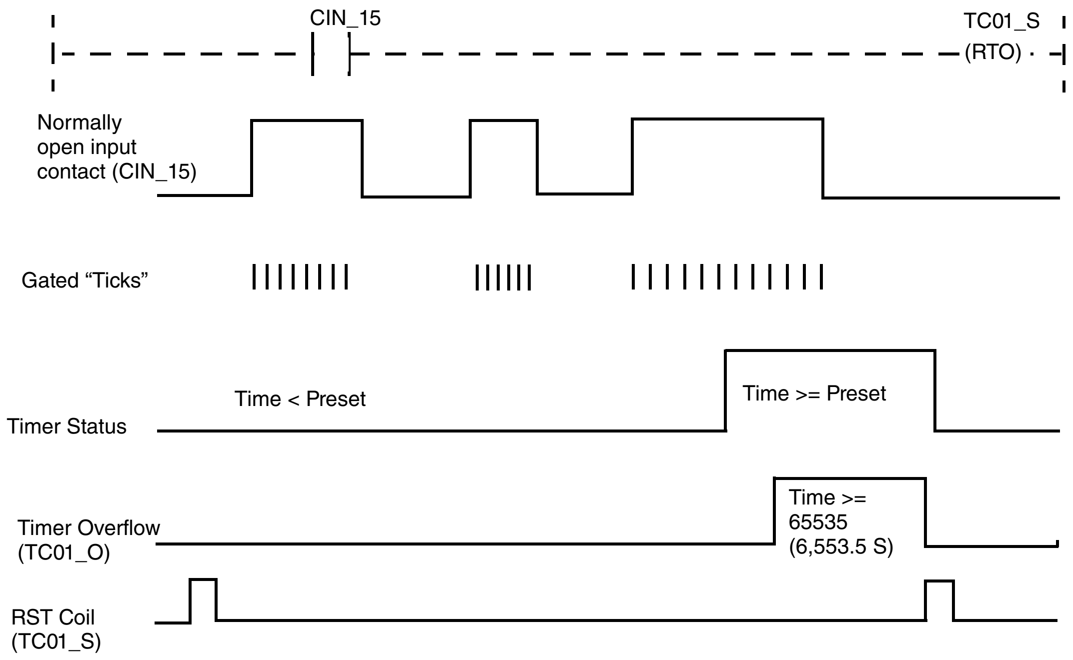


Figure 8-10. Retentive Time-On Delay Timing Chart

Table 8-10. RTF Truth Table

Reset Rung (RST Coil)	Off-Delay Timer Rung (RTF Coil)	Accumulator Register	Status Flag (_S Coil)	Timer Advance	Overflow Flag (_O Coil)
True*	don't care	= Reset Value*	False*	No	False*
False	True*	= Reset Value	True*	No	False
False	False*	> Reset Value	True	Yes*	False
False	True*	> Reset Value	True	No*	False
False	False	= Preset Value*	False*	Yes	False
False	False	> 6,553.5*	False	Yes	True*

* Note that in each row, the emphasized entry initiates change in other asterisked entries in the same row.

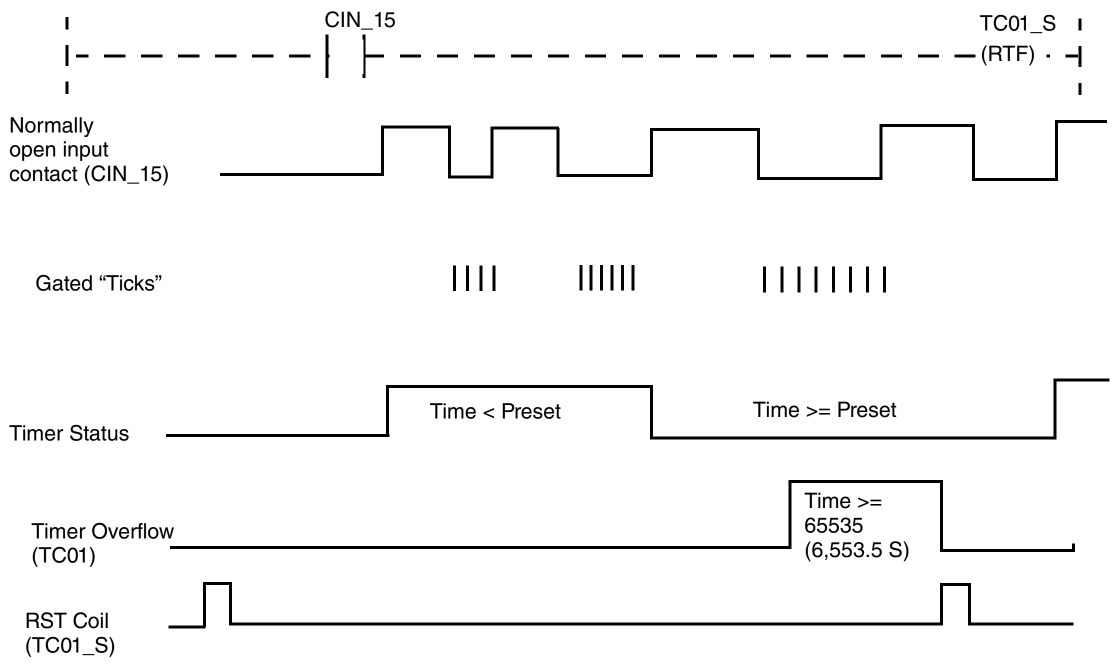


Figure 8-11. Retentive Time-Off Delay Timing Chart

The coil symbol that you insert in the ladder determines whether a specific structure is treated as a counter or a timer. You complete the definition by associating a technical identifier with the coil symbol. To use a timer or counter to control a rung, you insert a contact symbol in the rung and assign it the timer or counter identifier, together with an "_S" for the status flag or "_O" for the overflow flag. To control the reset of a counter or timer, you build a rung that outputs to a Reset coil that you give the same technical identifier as that of the counter or timer.

Counting, timing, and reset operations, including related changes in the status and overflow coils, are performed when the corresponding symbols are encountered during ladder solution.

The tick interval is 0.1 seconds. This clock tick is a global Fieldbus Module function that is not synchronized with ladder scans in any way. This means that turning a timer on and off rapidly to accumulate short times may produce unpredictable results.

Non-Retentive Timers

The non-retentive on-delay timer (TON) is identical in operation to the RTO coil, except that any period of false rung state is treated as if a Reset (RST) has occurred, that is, timing accumulation does not bridge false-rung periods. Similarly, non-retentive off-delay timers (TOF) treat periods of true rung state as if a Reset had occurred.

Timer/Counter Reset (RST)

The timer/counter Reset instruction loads the configured reset value into a counter's or timer's accumulator and resets the status and overflow flags. When configured, the timer/counter Reset coil is given the same reference (technical identifier) as the timer or up/down counter coil it is to reset. The referenced counter/timer is prevented from counting or timing as long as the reset rung is true.

To ensure predictable operation, you should configure a reset value that is less than the preset value, except for CTD coils.

⚠ WARNING

Without a Reset (RST) instruction the counter's accumulator starts at a zero value. For the down counter, this means the first count rolls the counter over to 65,535 and sets the overflow flag. You should always configure a Reset for twice the counter structures you plan to use.

Connector Symbols

There are two types of symbols providing only power flow: horizontal connectors and vertical connectors.

Horizontal connector symbols provide power flow through symbol positions where no symbol is configured.

Two different connector symbols are needed to connect two ladder rows in the form of a branch. A down vertical connector must be attached to the upper ladder row. An up vertical connector is inserted as the final element of the row beneath it. The vertical connectors must meet one another to form a single vertical line. The connectors are inserted equidistant between two symbol's positions.

Blank Positions

A blank in a symbol position interrupts rung power flow.

Program Flow Control

The program flow control instructions – Master Control Relay and Zone Control Logic – provide a means of conditionally skipping execution of one or more ladder logic rungs within a ladder logic diagram. These instructions come in pairs. The first one has a conditional input which defines whether a zone of ladder rungs is to be processed. The second one (no conditional input) is inserted to mark the end of the conditional group of rungs.

Each of these instructions when false causes a zone of ladder logic to be skipped. However, the effect on outputs, counters, and timers differs according to the instruction used. The following Table 8-11 summarizes the effects that Master Control Relay and Zone Control Logic have on the operation of ladder logic.

Table 8-11. Effects of MCR and ZCL on Ladder Operation

Ladder Element	Effect When MCR False	Effect When ZCL False
Non retentive outputs Retentive outputs	Set false Do not change	Do not change Do not change
Counters	Do not count transitions	Still count transitions
Counter/timer registers and flags	Do not change	Status and overflow reflect Counter/Timer changes

NOTE

1. MCR is Master Control Relay; ZCL is Zone Control Logic.
2. MCR transitions do not cause counters to count.
3. MCR can affect ZCL rungs.
4. When MCR is false, the only coil that can be written true is MCR itself.

Master Control Relay (MCR, NCR)

You can use Master Control (MCR) and End of Master Control (NCR) coils to divide a ladder diagram into several master control zones. An MCR coil marks the beginning of the zone and an NCR coil marks the end.

MCR is a coil that is referenced by all output operations as they are encountered during the ladder solution. MCR is set true at the top of each scan. At any point in the ladder, MCR can be written in the same way as any other coil.

Ordinary contacts determine whether to set the MCR coil value true when the first rung in the MCR zone is processed. The state of the MCR coil has the following effects:

- ♦ MCR true – The rungs situated between the MCR coil and the End Master Control Relay (NCR) coil are executed normally.
- ♦ MCR false – The rungs situated between the MCR coil and the NCR coil are not executed. The following additional effects take place:
    - ♦ The states of non-retentive outputs within the zone are set false. Retentive outputs within the zone are not affected.
    - ♦ Counters do not count transitions.
    - ♦ MCR transitions do not cause counters to count.
    - ♦ Counter/timer registers and flags are not affected.
    - ♦ The only coil that can be written true is MCR itself.
    - ♦ A ZCL zone nested in a MCR is affected by the MCR as follows: if both zones are off, the ZCL will not turn on until the MCR is true.

The ladder logic implementation allows nested and overlapping ZCLs and MCRs in any combination.

MCR does not affect rungs inside a zone that is bypassed.

Zone Control Logic (ZCL, NCL)

You can use Zone Control (ZCL) and End of Zone Control (NCL) coils to divide a ladder diagram into several zones. A ZCL coil marks the beginning of the zone and an NCL coil marks the end.

Ordinary contacts determine whether to set the ZCL coil value true when the first rung in the ZCL zone is processed. The state of the ZCL coil has the following effects:

- ♦ ZCL true – the rungs situated between the ZCL coil and the NCL coil are executed normally.
- ♦ ZCL false – the rungs situated between the ZCL coil and the NCL coil are skipped (not executed) and rung outputs within the zone do not change state.

If the state of a timer rung is such that the timer is ticking, if the zone transitions to false, the timers cease operation.

The ladder logic implementation allows nested and overlapping ZCLs and MCRs in any combination.

Special Purpose Coils

The special purpose instructions include the following coils:

- ♦ Initialization
- ♦ Communication Failure
- ♦ Power fail
- ♦ Fail safe.

Initialization (INIT)

NOTE

The INIT coil is global within the same FBM; that is, if there are multiple ladders resident in a given FBM, activating the INIT coil in any one of those ladders initializes all the ladders in the FBM.

The Initialization (INIT) coil has two functions:

1. When your program writes the INIT coil true, INIT forces the next scan to be an initialization cycle. This forces a reset of all timers, counters, and coils at the start of the next scan. The INIT coil remains true for one scan.
2. Following cold start or receipt of an initialize command, INIT is true for one scan. You can reference INIT to initialize certain coils prior to beginning operation.

Communication Failure (COMMF)

Communications software sets Communication Failure (COMMF) false during exchanges with the control processor as long as the communications timer has not timed out. If communications fail, COMMF is set true and remains true until communications are restored.

COMMF does not indicate the type of messages received, control processor state, or whether replies actually reach the control processor.

Use the COMMF contact to have your ladder respond to a communication failure between the FBM and the control processor. For instance, the rung diagrammed below will cause the ladder to fall back to the fail-safe mode. (In addition all other ladders in the FBM will go to fail-safe – see Note under Fail-safe.)

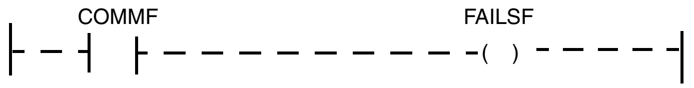


Figure 8-12. COMMF Contact

Power Fail (POWERF)

The Power Fail (POWERF) coil is false as long as the execution of the Fieldbus Module ladder application is uninterrupted.

POWERF is set true following a warm start caused by either of the following conditions:

- Recovery from a power interruption.
- A command to the Fieldbus Module to go offline and back to online.

POWERF is true for exactly one scan following recovery, and then becomes false. You cannot set or reset POWERF. If failure occurs in the middle of the processing of a rung, recovery starts at the beginning of the ladder with POWERF true, and all input and output values are as they were at the end of the last complete scan before the failure.

Fail-safe (FAILSF)

— NOTE

The FAILSF coil is global within the same FBM; that is, if there are multiple ladders resident in a given FBM, activating the FAILSF coil in any one of those ladders puts all the ladders in the FBM into the configured fail-safe mode.

Writing the Fail-safe (FAILSF) coil true forces physical outputs to the preconfigured fail-safe state. It does not affect values in the coil table. During the output exchange portion of the scan cycle, the value of the output coil is checked and the physical outputs are handled based on the values of the fail-safe, connect, and hold flags. Writing the FAILSF coil true will not override a control processor specification of disconnect-hold.

Fanned Outputs

Fanned outputs are supported when implemented as a single vertical column branching down from the first line of the rung, immediately after the seventh element position. See Figure 8-13.

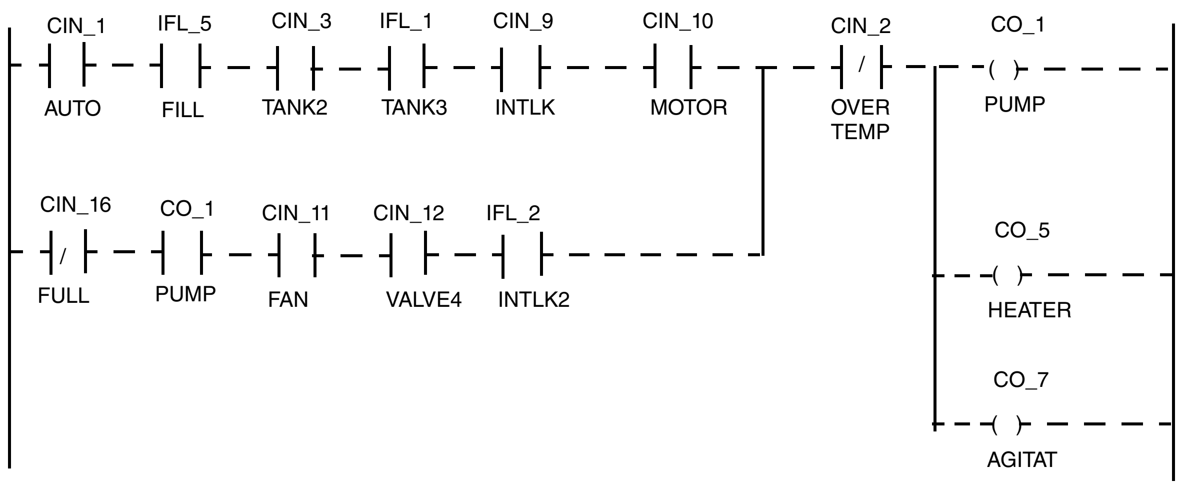


Figure 8-13. Fanned Outputs

Simple and Nested ZCLs

Figure 8-14 through Figure 8-16 illustrate the use of simple Zone Control Logic, and Figure 8-17 through Figure 8-18 show the inclusion of a nested “Fast” zone within the “Manual” zone.

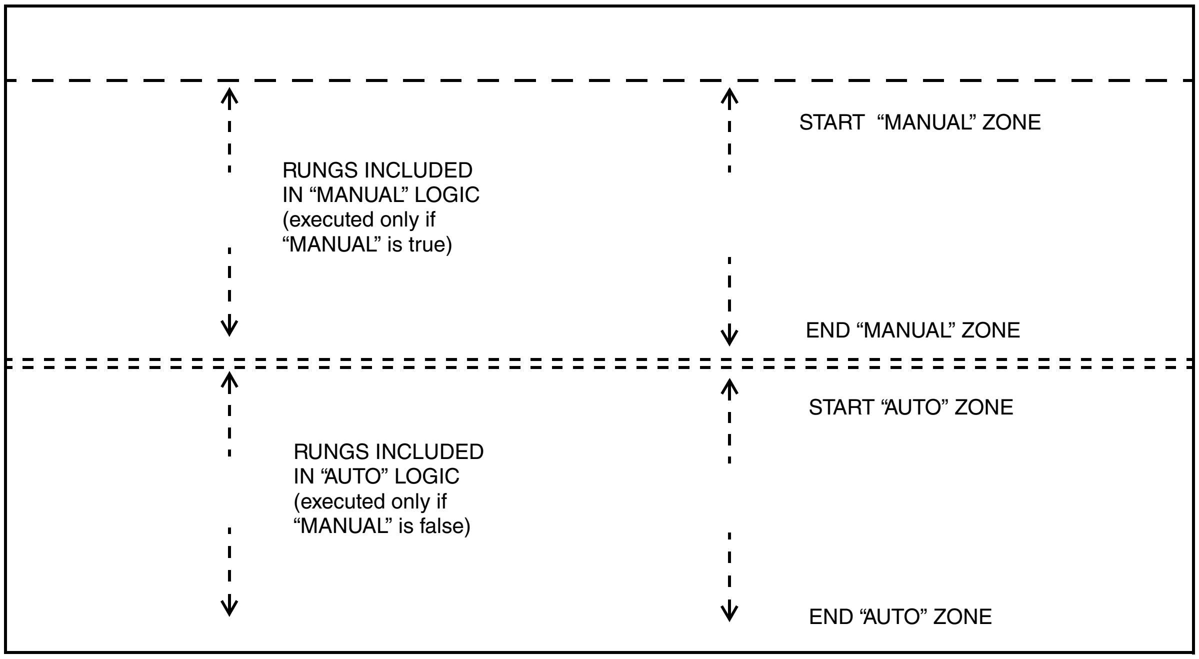

START “MANUAL” ZONE
RUNGS INCLUDED IN “MANUAL” LOGIC (executed only if “MANUAL” is true)
END “MANUAL” ZONE
START “AUTO” ZONE
RUNGS INCLUDED IN “AUTO” LOGIC (executed only if “MANUAL” is false)
END “AUTO” ZONE

Figure 8-14. Zone Control Logic

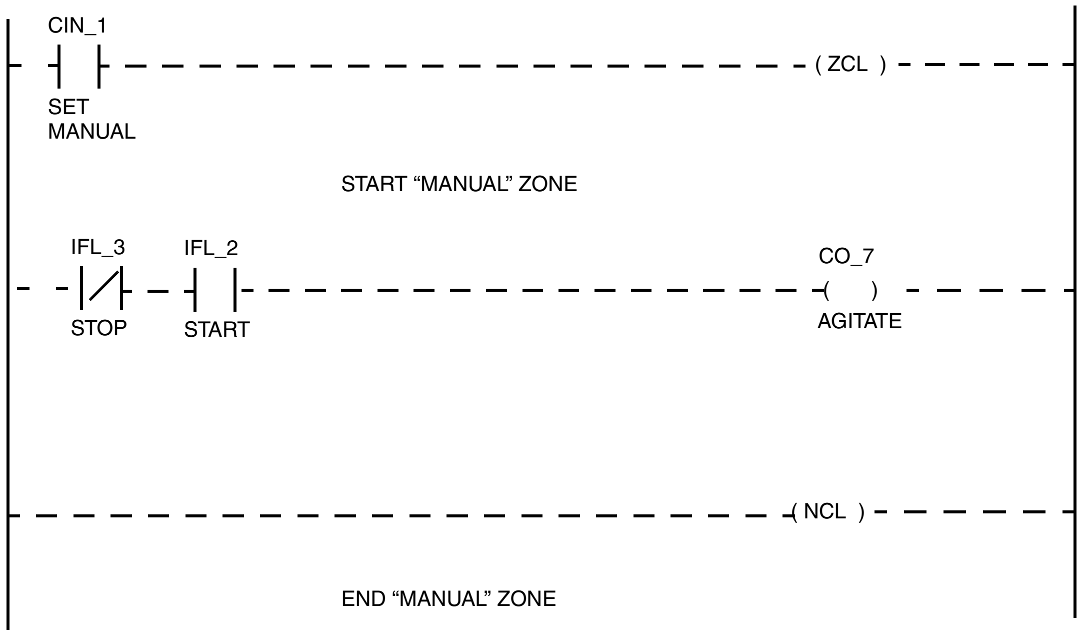


Figure 8-15. Zone Control Logic (Continued)

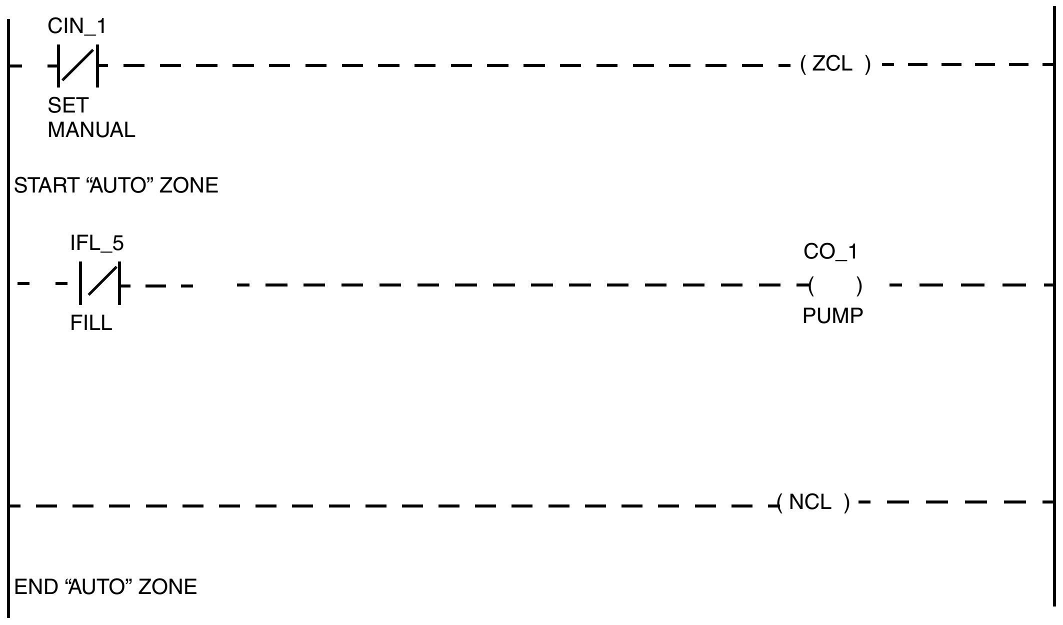


Figure 8-16. Zone Control Logic (Continued)

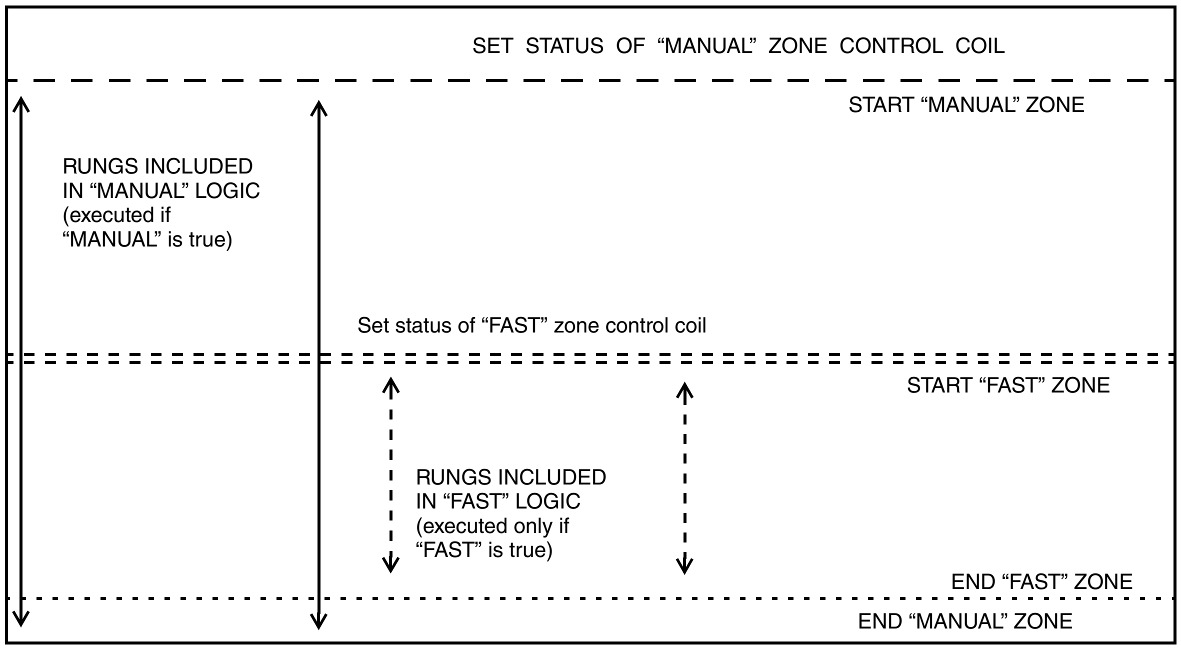

SET STATUS OF "MANUAL" ZONE CONTROL COIL
START "MANUAL" ZONE
RUNGS INCLUDED IN "MANUAL" LOGIC (executed if "MANUAL" is true)
Set status of "FAST" zone control coil
START "FAST" ZONE
RUNGS INCLUDED IN "FAST" LOGIC (executed only if "FAST" is true)
END "FAST" ZONE
END "MANUAL" ZONE

Figure 8-17. Nested Zone Control Logic

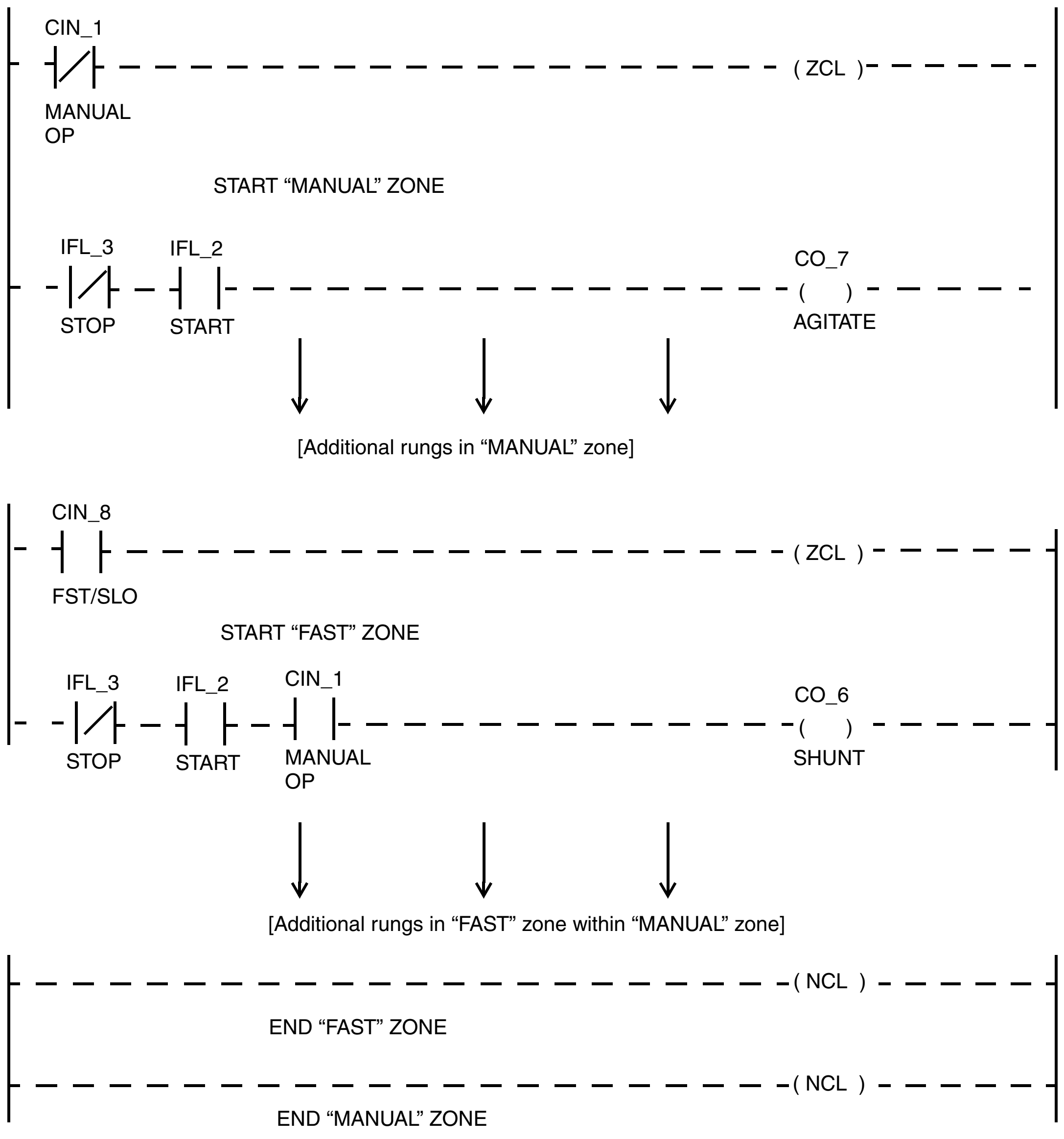


Figure 8-18. Nested Zone Logic (Continued)

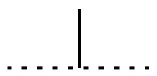
Ladder Diagram Constraints

The object code produced by ladder logic compilation is executable only in a Fieldbus Module and cannot exceed 1K of FBM memory. The ladder logic editing process supports a maximum of 32 contact inputs, 16 contact outputs, and 16 timers/counters for a single Fieldbus Module.

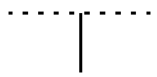
The ladder logic compiler imposes the following restrictions in the composition of a ladder diagram:

- All elements having the same technical identifier (one coil and multiple contacts) have the same user label.
- A maximum of eight logic symbols (not including branches) can be inserted within a ladder row.
- Output symbols may appear only as the final element within a row.
- Fanned outputs must be arranged in a vertical column.

- MCR and ZCL symbols may not be included in an output fan.
- A rung that contains an NCL symbol (end of ZCL) may not include any other symbols. Similarly for NCR symbols.
- There must be an equal number of ZCL and NCL symbols within a ladder diagram.
- The following geometry is not allowed:



- The following geometry is allowed only in the first line of a rung. This constraint disallows nested branches, which are not supported in ladder diagrams:



- Use no more than two vertical connectors to link a row of adjoining symbols to a preceding ladder line.

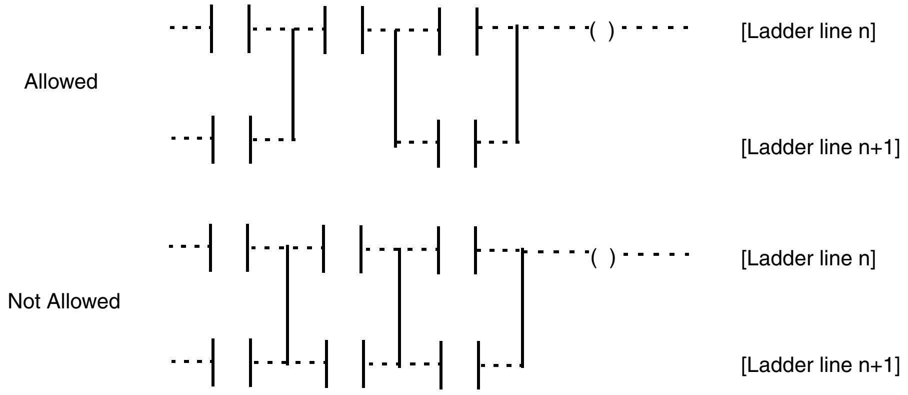


Figure 8-19. Allowed/Not Allowed Connector Geometry

The following considerations relate to the use of overlapping technical identifiers in different segments of the same composite ladder.

A given technical identifier (for example, OFL_1) may be entered in more than one ladder segment, but doing so does not create additional OFLs. There is only one OFL_1, with a single data value, in the FBM. It may be entered as an output coil in multiple segments of the ladder, in which case the final concatenated ladder will have more than one rung ending with symbol OFL_1. When the OFL_1 value is transmitted to the control processor, its value will depend on the evaluation of the highest-numbered rung ending in OFL_1, that is, the last calculation of OFL_1 during the ladder scan. All PLBs connected to that FBM will receive this value of OFL_1 as an input. This implies that if the usage of technical identifiers is overlapped, a PLB block could receive values of parameters from the FBM different from the ones expected on the basis of their own ladder segments. The same considerations apply to overlapped usage of COs, TCs, and IFLs.

If any IFL is referenced in multiple segments of a composite ladder, the value sent down to the FBM for use in solving the ladder will be the value of that IFL parameter in the PLB which is processed last. This will depend on the zones, and positions within the zones, of the various PLBs involved, as shown on the Block/ECB Functions screen of the Integrated Control Configurator.

Programmable Logic Block Operation

The programmable logic block (PLB), shown in the PLB Block Diagram, supports ladder logic executing in a digital Fieldbus Module. The block executes at the interval you specify (one of 13 user-specified scan intervals from 50 ms to 1 hr; default of 500 ms). At the configured interval, the PLB reads each physical I/O channel and each ladder logic external output flag reference from the Fieldbus Module and updates the appropriate block parameter values. The block processor writes each external input parameter value to the appropriate ladder logic external input flag reference in the Fieldbus Module. Refer to *Integrated Control Block Descriptions* (B0193AX) for detailed information on block parameters.

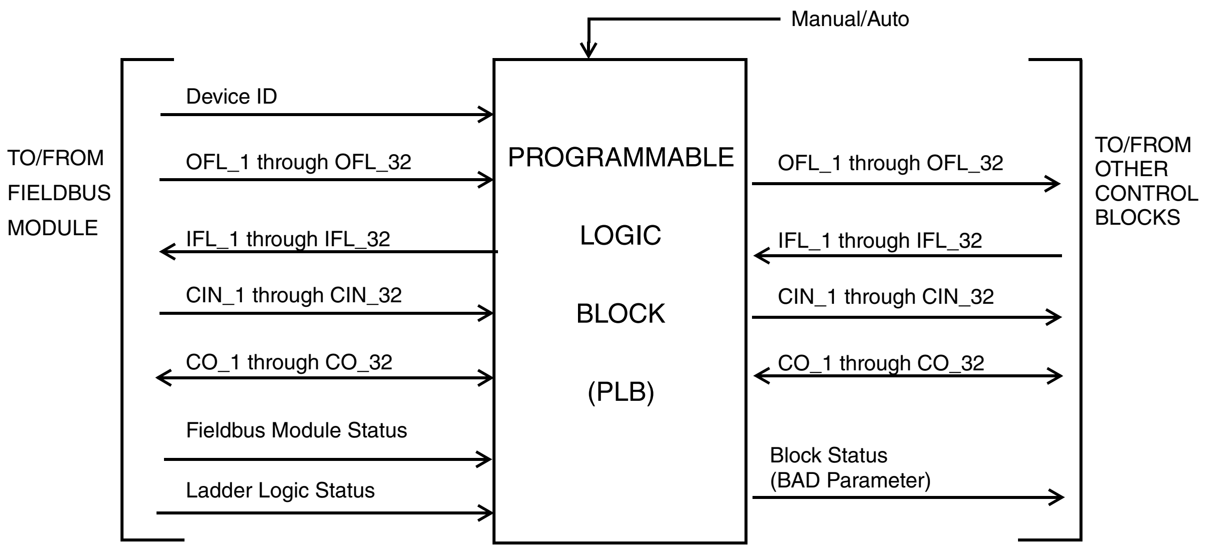


Figure 8-20. Programmable Logic Block (PLB) Block Diagram

As part of normal Fieldbus data input processing, all inputs and outputs are written to and read from the Fieldbus Module in one transaction:

- The Interface block writes the current External Input Flags to the Fieldbus Module. These are signals from other blocks to initiate or modify ladder logic actions.
- The Fieldbus Module returns its latest scanned data for:
  - Fieldbus Module status
  - External Output Flags. These are signals to other blocks of the results of ladder logic actions.

The Fieldbus Module ladder logic now executes the new External Input Flags.

The PLB has an Auto/Manual mode that determines control of block output parameters. Operationally, MA lets you switch control of block flag outputs and physical outputs from the ladder program (Auto) to the user (Manual).

In Auto, the block secures its own output parameters and updates them according to ladder logic and Fieldbus Module inputs.

The block reads the ladder logic external output flag references from the Fieldbus Module and updates the appropriate output parameters with these values. The block writes input parameter values to the appropriate ladder logic external input flag references of the Fieldbus Module.

The input parameter values are written to the specified ladder logic input flag registers. If either the Fieldbus Module is not operational or the ladder program is off-line, the inputs are not written and the outputs are not updated. The BAD parameter is updated and the BAD status of all output flag records are set.

When the block is placed into Manual, the external output flag parameters from the ladder program are updated by the block. The input flag parameters continue to be written to the external input flags in the ladder program.

The Manual mode allows you to disconnect outputs from the Fieldbus Module ladder logic for checkout or simulation from an external source. In Manual, the block unsecures its Boolean type outputs. Any task or process is then allowed to use Set calls to write to the outputs.

Ladder segments in different Fieldbus Modules can communicate with each other through the PLB input and output flag parameters. An external output flag from a ladder program in one Fieldbus Module can be connected to an external input flag of a ladder program in another Fieldbus Module through the respective block's output and input flag parameters. Switching the MA state of the upstream block to manual would allow for the independent and isolated operation of each flag under user control.

The physical inputs at the Fieldbus Module can be read by a CIN block. The physical outputs at the Fieldbus Module can be read by a COUT block. Both physical inputs and physical outputs are read by the PLB.

In Manual, the block writes ladder logic external output flags. The block supports Manual/Auto mode for manually updating block outputs. Manual does not affect the operation of the input parameters or external input flags. In the manual mode, the updating of the output parameters is halted and control of the output parameters by workstations, other blocks, displays, and processes (programs) is possible.

Various software subsystems access internal information to edit a ladder diagram, display rung power flow, read and write timer/counter values, and control the modes of ladder logic operation. Displays that you configure can access ladder logic through external flag parameters in PLBs.

Ladder Logic Operation

The ladder logic functions in the Fieldbus Module can be divided into two modes: foreground and background. Communications and other interrupts are handled in the foreground. The cyclic control function defined by the ladder program is handled in the background. The foreground and background communicate through shared buffers. The background, when it needs to access the shared buffers, locks out the foreground by shutting off interrupts for periods of up to 1 ms (refer to Figure 8-21).

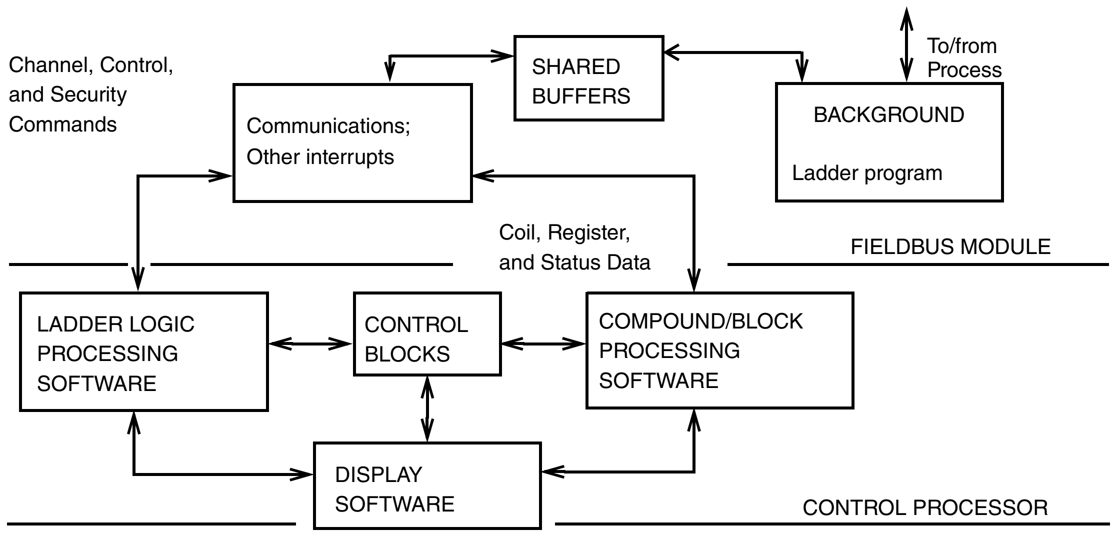

Channel, Control, and Security Commands
Communications; Other interrupts
SHARED BUFFERS
To/from Process
BACKGROUND
Ladder program
Coil, Register, and Status Data
FIELDBUS MODULE
LADDER LOGIC PROCESSING SOFTWARE
CONTROL BLOCKS
COMPOUND/BLOCK PROCESSING SOFTWARE
DISPLAY SOFTWARE
CONTROL PROCESSOR

Figure 8-21. Ladder Logic Functions Diagram

Ladder Logic Scan Cycle

The Fieldbus Module ladder logic processor operates in a cycle referred to as a scan. Each scan requires the following four phases:

1. Processing commands from the control processor.
2. Reading inputs from physical and logical channels.
3. Solving the ladder.
4. Writing outputs to physical and logical channels.

These phases form an infinite loop. Once placed in run, the program repeats until halted. One pass through the loop (one scan) typically takes from 10 to 30 ms. The actual scan time depends on the complexity of the ladder program and the state of the logic.

Scan Phase 1 – Command Processing

Table 8-12 lists the commands that the control processor uses to direct ladder logic in the Fieldbus Module.

The control processor exchanges blocks of data with the Fieldbus Module. In Scan Phase 1, the Fieldbus Module checks for the receipt of any commands that require mode changes. Executing mode changes first avoids the inconsistency between blocks of data that would result from executing only a portion of the user program.

Table 8-12. Ladder Logic Commands

Command	Description
1. Read Channel Data	Read both the state of the physical I/O and the logical outputs of the ladder.
2. Write Channel Data	Write the external input flag values used by the ladder logic processor.

Table 8-12. Ladder Logic Commands (Continued)

Command	Description
3. Object Code Download	Download object code to the program code space that is not in use.
4. Switch Program	Switch the logic processor to the program code space that is not in use.
5. Write Ladder Logic Status	Determine the ladder logic operating mode.
6. Read Coil Table Data	Read the Boolean state of all 255 coils to support displays. This command also updates the register values read by the read register data command.
7. Read Register Values	Read the preset, reset, and accumulator register values to support displays. The values supplied are those saved at the time of the most recent read coil table request.
8. Write Register Values	Change the values of the timer/counter preset and reset registers to support displays.
9. Write Force On List	Write the Fieldbus Module force-on list when the logic processor is in test mode. All force lists are optionally erased on exit from test mode.
10. Write Force Off List	Write the Fieldbus Module force-off list when the ladder processor is in test mode. All force lists are optionally erased on exit from test mode.

Scan Phase 2 – Inputs

The inputs to the ladder logic come from two sources:

1. Physical inputs come from the process through channels provided by the Fieldbus Module hardware.
2. Logical inputs come from control blocks and tasks through Ladder Logic Interface blocks (PLB Blocks) in the control processor.

During the input phase of the scan cycle, the values associated with each input are transcribed into fixed locations in the ladder logic coil memory.

Scan Phase 3 – Ladder Solution

The end of the input phase transfers control to rung evaluation software to execute the ladder program. Execution starts at the beginning of the object code array and proceeds until the end of the program is reached. The rung evaluation alters only local variables and has no effect on any physical I/O processes.

Scan Phase 4 – Outputs

Scan Phase 4, outputs, is the mirror image of the input phase. The outputs from the ladder logic are handled two ways:

1. Physical outputs go to channels provided by Fieldbus Module hardware.
2. Logical outputs are provided to Ladder Logic Interface blocks (PLB Blocks) in the control processor through the external flag buffer.

The values in coil memory are always transcribed to the corresponding Fieldbus Module external flag buffer. External flag outputs await the control processor's next communications request.

The ladder logic sets physical outputs as specified under one of the following situations:

1. I/O connected and fail-safe false: The values in coil memory are transcribed to the corresponding physical Fieldbus Module outputs.
2. Fail-safe true: The physical outputs are set to the preconfigured values specified by the configuration table.
3. I/O not connected: The physical outputs are set as specified by the fail-safe and hold flags.

During the output phase, the buffers that hold the data blocks for display support are also updated. This buffering allows the foreground to handle all the normal communications messages asynchronously with ladder evaluation.

Ladder Logic Modes of Operation

In conjunction with the Test and Install modes of the ladder logic configuration/operation software, the control processor sends commands to the Fieldbus Module to establish ladder modes of operation. In Test mode, you can operate the ladder program with I/O connected or disconnected in continuous (Run) or Single Scan cycling. When you disconnect I/O, you can choose either the hold or the fail-safe option for outputs. These choices are summarized as follows:

Table 8-13. Ladder Modes of Operation

I/O Connected	I/O Disconnected
Run	Run/Hold Run/Fail-Safe
Single Scan	Single Scan/Hold Single Scan/Fail-Safe

Ladder Logic Status Byte

When the Fieldbus Module is on-line, the logic status byte controls and reports the ladder modes of operation. The effect and meaning of the flags in the status byte are defined individually in the pages that follow. With the exception of the Hold flag, the functions controlled by these flags are independent.

Logic Status Byte Flags
1. Run
2. Test
3. Connect
4. FailSafe
5. Hold
6. Single Scan
7. INIT
8. Logic Machine Error

Run Flag

The Run flag defines whether the logic solving portion of the scan cycle is running.

Run Flag Set To One	Run Flag Set To Zero
Ladder program executes.	Ladder program does not execute.

The I/O exchange portions of the scan cycle always occur. The Connect, Hold, and Fail-safe flags determine behavior of the outputs as defined in the following sections.

Test Flag

The ladder logic configuration/operation software in the control processor controls the Test flag.

Test Flag Set To One – The ladder logic performs the following functions:

Accepts and processes force lists.

Writes to timer/counter accumulator registers.

The added processing required to support force lists causes the cycle time performance of the ladder logic to be degraded when the Test flag is set to one.

Connect Flag

The state of the Connect flag determines whether the physical I/O is controlled by the ladder logic.

Connect Flag Set To One	Connect Flag Set To Zero
Outputs are connected to the ladder logic.	Cycling the ladder logic has no effect on the physical outputs.

In the Test mode of the ladder logic configuration/operation software in the control processor, you select an operating mode that includes setting the state of the Connect flag to one.

Fail-safe Flag

When the Fail-safe flag is set to one, the physical outputs are driven to the state determined by the Fieldbus Module configuration table (established through integrated control configuration).

Control of the Fail-safe flag depends upon the status of the Connect flag as follows:

Connect Flag Set To One	Connect Flag Set To Zero
The fail-safe coil in the ladder controls the Fail-safe flag.	The ladder logic configuration/operation software in the control processor controls the Fail-safe flag.

The Fail-safe flag has priority over the Hold flag. If both are set to one, the result is fail-safe status. Fail-safe status is also the default, should the control processor ever set the Connect flag to zero without setting either Fail-safe or Hold to one.

Hold Flag

The Hold flag is recognized only if the Connect flag is set to zero. When the Hold flag is set to one, the physical outputs are held at the state that was in effect at the time the flag was set to one.

Hold Flag Set To One	Hold Flag Set To Zero
The physical outputs are held at the state in effect at the time the flag is set to one.	The physical outputs are either in fail-safe or are connected.

Single Scan Flag

The Single Scan flag is recognized only if the Run flag is not set to one. Other combinations are as follows:

Single Scan Flag Set To One and Run Flag Set To One	Single Scan Flag Set To Zero and Run Flag Set To One
The Single Scan flag initiates a single scan. The scan sets the flag to zero.	The ladder processor cycles continuously.

INIT Flag

If you choose to install the ladder in run mode and initialize on start up, the control processor sets the INIT flag in the ladder status byte. The ladder program performs a cold start initialization of the ladder prior to executing the first scan. All coils are reset, the INIT coil is set, and all timer/counters are reset.

The ladder logic also sets the INIT flag any time a cold program start is requested by other software in the control processor or the INIT coil is set by the internal logic. INIT is set to zero automatically at the end of the first scan following initialization.

Logic Machine Error Flag

The Logic Machine Error flag is set if the ladder logic is ordered to switch to a code buffer that fails the checksum test. The ladder logic also sets fail-safe and halt status.

The only recovery from a logic machine error condition is to load a valid object program and switch to it. This will set the error flag to zero. You can then set the status as desired.

An additional FBM ladder operational mode exists, based on the configurable simulation option SIMOPT, which is an ECB8 parameter. When downloaded to the FBM, SIMOPT causes the ladder logic to disregard the physical contact inputs to the FBM in favor of the values specified by the configurable and settable packed long parameter SIMCIN. SIMCIN is also an ECB8 parameter.

PLB Editor Overview

The ladder logic domain of the PLB Editor in the ICC allows you to:

- Create or modify one ladder diagram segment
- Check for syntax errors in existing ladder diagram source code
- Print a copy of the ladder diagram segment being edited.

You can configure ladder logic from any system display station equipped with a keyboard and pointer device. In the Edit mode of operation, you can insert, delete, and move symbols in the diagram.

As you construct a ladder diagram, the PLB editor builds a source file representing the desired logic. The editing screen displays the ladder corresponding to the PLB name. An “empty” ladder file causes a display of the ladder name only.

Editing consists of placing logic symbols that represent contacts, relays, timers, and counters within the rungs of the ladder. If you elect to save the edited version, the original ladder is replaced by the new. You can check your ladder for syntax errors at any point in your editing.

NOTE

For information on the Ladder Editor in IACC, refer to *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

Configuration Example

When configuring PLBs for different segments of a single ladder diagram, avoid assigning an external input flag to more than one block. If two PLBs were to update the same input flag, the flag value received from the first block would be overwritten when the other block was processed.

The five-part diagram accompanying the configuration example shows part of a control strategy to pump a fluid from either of two tanks. The first part shows the process, the next two show the control blocks, the fourth shows the I/O table and flag table connections to the PLBs, and the fifth shows a portion of the ladder diagram.

The example is not a practical application. It is intended only to show concepts associated with segments, while limiting each segment to one rung for simplicity. Arbitrarily, some of the control is implemented through external I/O flags and some through physical I/O channels to demonstrate the two methods of communicating with ladder logic.

In the example, an operator presses one of two switches to select a tank to supply fluid. If the tank’s low level switch is not closed, the ladder logic output to an On/Off Valve controller (VLV) block opens the tank’s drain valve. Ladder logic causes the pump to run if either tank valve is open.

Physical contact inputs are available to the ladder from input channels of the Fieldbus Module. Contact inputs from the operator panel and from valve limit switches activate one of the module’s physical output channels to operate a motor control relay in the process. To demonstrate another input path, one input is routed through an MCIN block to the PLB as an external input flag to the ladder. To demonstrate other output paths, the PLBs transfer external output flags from the ladder to VLV blocks and a COUT block.

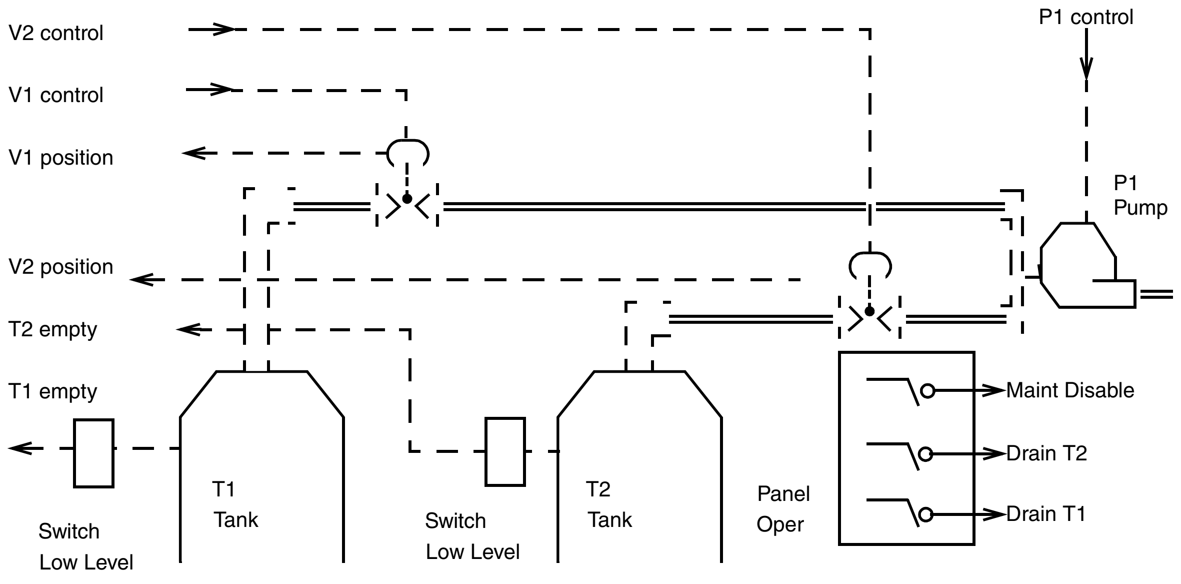


Figure 8-22. Typical Ladder Logic Application

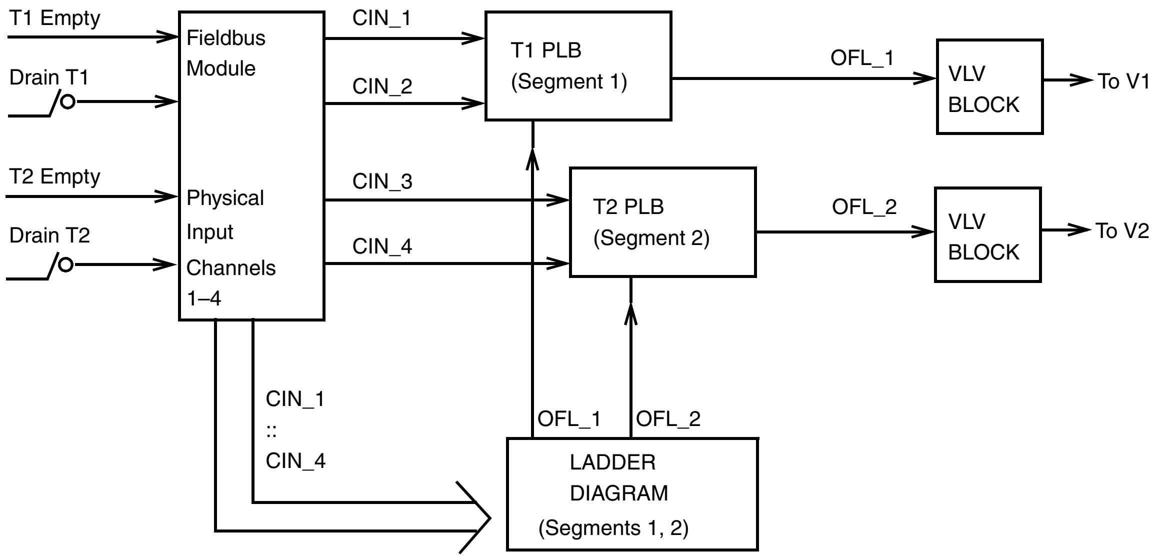


Figure 8-23. Typical Ladder Logic Application (Continued)

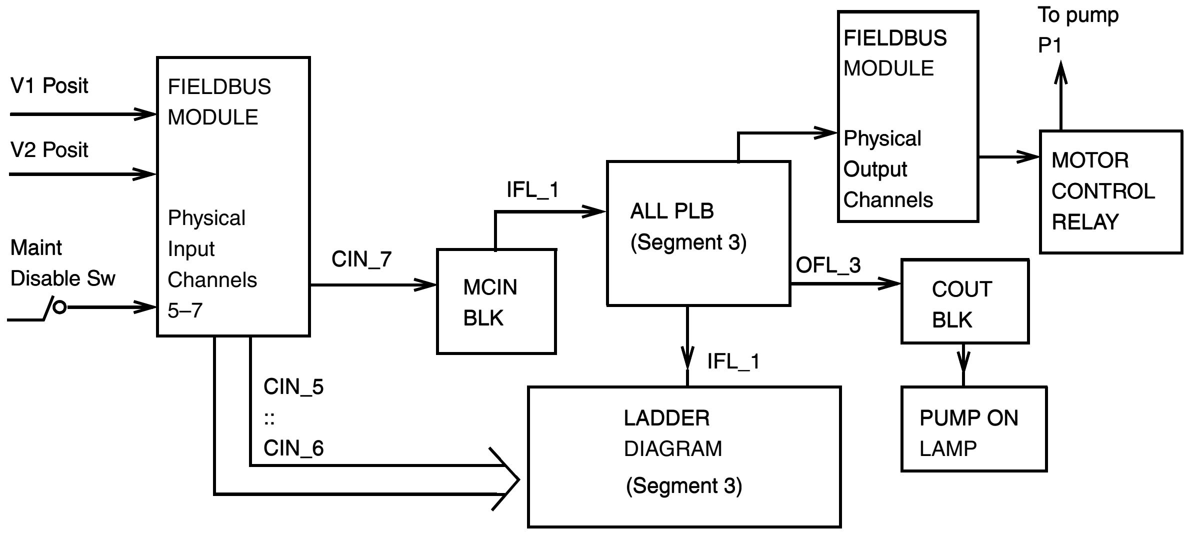


Figure 8-24. Typical Ladder Logic Application (Continued)

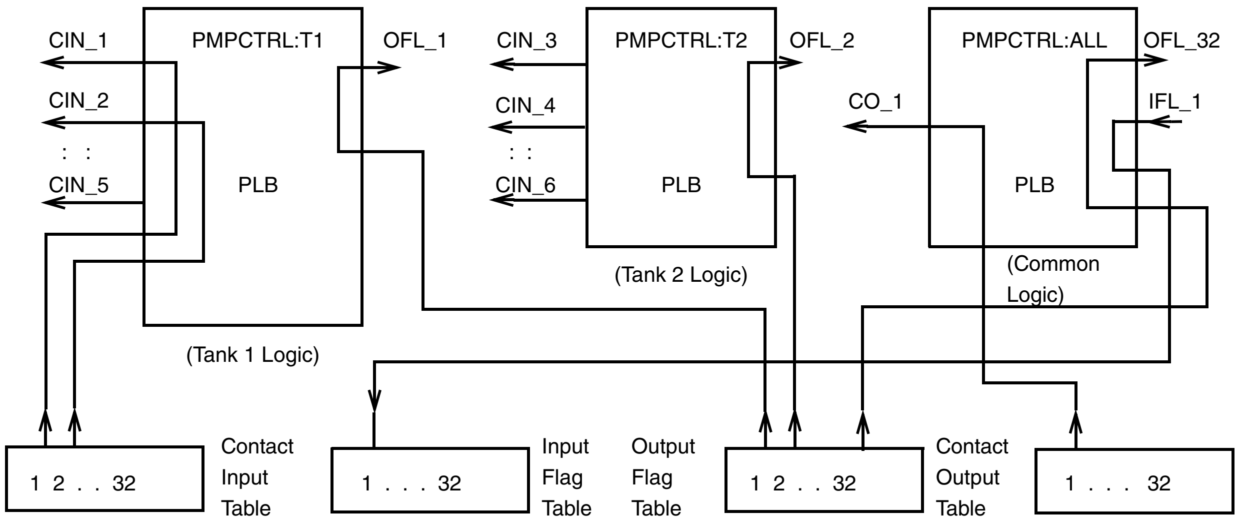


Figure 8-25. Typical Ladder Logic Application (Continued)

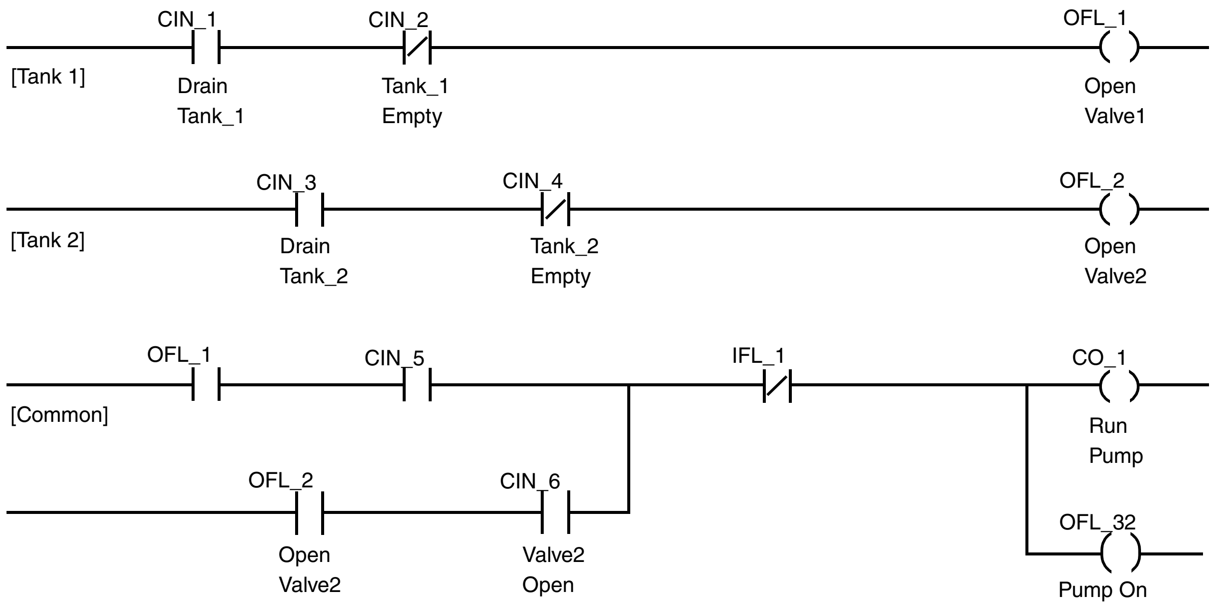


Figure 8-26. Typical Ladder Logic Application (Continued)

9. Sequence Logic

This chapter covers sequence logic control: sequential control block types and their various states, sequence processing, and SBX programming.

Sequence logic control complements continuous and ladder logic control with regulatory feedback applications at the equipment control level. For example, the sequential control software can be used to supervise a sequence of activities such as filling a tank, blending its contents, and draining the tank.

Whereas most continuous control blocks have fixed algorithms, sequence control blocks have user-defined algorithms. Sequential control software enables you to:

- ♦ Define a sequence of events
- ♦ Monitor process conditions, taking corrective action when required
- ♦ Time events
- ♦ Manipulate any compound or block parameter or any shared variable
- ♦ Output messages to any logical device or to the historian.

To introduce sequential control to a control strategy, you must define sequential control blocks and add them to compounds.

Sequential control blocks are configured through the ICC or IACC just as continuous blocks are, except that in addition to configuring parameters, you must define sequence logic for the block. You can choose either the ICC or IACC to configure sequence blocks, but blocks in a controller configured with one configurator cannot be edited with the other configurator.

The ICC offers a series of menu functions for creating sequence logic. For more information refer to *Integrated Control Configurator* (B0193AV).

The IACC offers the ST Code Editor, ST Template Editor, and ST Templates for creating sequence logic. For more information refer to *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

Sequence logic is created with the Sequence Language, a subset of the High Level Batch Language (HLBL). The Sequence Language is a high level programming language resembling Pascal, but specifically geared toward creating process control strategies. High Level Batch Language (HLBL) is described in *High Level Batch Language (HLBL) User's Guide* (B0400DF).

Sequential Control Blocks

Sequential control is performed at the compound level through sequential control blocks of the following types:

- ♦ Sequence (IND, DEP, and EXC)
- ♦ Monitor (MON)
- ♦ Timer (TIM).

Sequence Block

There are three Sequence block types:

- Independent (IND)
- Dependent (DEP)
- Exception (EXC).

IND and EXC blocks run independently of other Sequence blocks in the same compound. DEP blocks pause when an EXC block in the same compound is active.

This relationship between DEP and EXC blocks allows you to separate a sequence algorithm for handling normal conditions from a sequence algorithm for handling alarm conditions. For example, if a Monitor (MON) block detects an alarm condition and activates an EXC block to take corrective action, the DEP block pauses until the corrective action is complete. When the EXC block is done, the DEP block can finish executing its sequence algorithm.

A Sequence block contains a user-defined sequence algorithm. You can use a Sequence block to:

- Manipulate parameters and shared variables.
  Refer to the table "Sequence Compiler Limits" in *High Level Batch Language (HLBL) User's Guide* (B0400DF) for the maximum number of external references in each subroutine or in the main code and all SBXs.
- Change the flow of execution based on the state of parameters and shared variables.
- Activate other Sequence and Monitor blocks.
- Measure time.
- Report to the Historian.
- Send information to logical devices, such as printers.
  Refer to the table "Sequence Compiler Limits" in *High Level Batch Language (HLBL) User's Guide* (B0400DF) for message related limits.
- Call a subroutine and pass arguments, if any.
- Make calculations.
- Simulate a process for testing purposes.

A Sequence block is composed of:

- Standard Parameters
- Block Type Identification
- Symbolic Constants
- Local Block Variables.
  Refer to the table "Sequence Compiler Limits" in *High Level Batch Language (HLBL) User's Guide* (B0400DF) for local variable related limits.
- User Labels
- Include Files
- Subroutines (variables and statements)
- Standard Block Exception Handlers
- Block Statements, grouped into Steps.
  Each statement, whether in the block's main section, in its subroutines, or in its standard block exception handlers, may optionally have a label.

Refer to the table “Sequence Compiler Limits” in *High Level Batch Language (HLBL) User’s Guide* (B0400DF) for labels related limits.

Standard parameters show block operation details and allow you to control block operation and connect the block in a control strategy that includes continuous blocks, ladder logic blocks, and other sequence blocks.

Block Type identification is a small block of information at the start of the sequence language file where you provide data such as block name, type, creator, revision level and date.

Symbolic constants are identifiers which represent constant values. They are used to indicate or illustrate the meaning of such values. The constants are an aid in compiling sequence blocks.

Changing the value of a constant in an include file does not affect currently running or already compiled sequence blocks automatically. After such a change, the blocks containing source code in which the constants are used have to be recompiled to effect the change.

Block variables are local and are not accessible from outside the block. You define their number and their size. There are no user labels for local block variables. Refer to them by their declared names. You can use them in any HLBL expression and you can assign them to each other, to user (array) parameters, and to external references.

Local block variables can be any of the following types:

- Boolean and boolean array
- Long integer and long integer array
- Real and real array
- String and string array.

  For local variables there are three string lengths: short (6 characters), medium (12 characters), or long (80 characters).

All arrays in the local block variables (main section) and the local subroutine variables may be multi-dimensional, with a maximum of 256 dimensions.

When specifying the types of main section or subroutine local variables, you may use a comma-separated list of variables before the type specification.

Refer to the table “Sequence Compiler Limits” in *High Level Batch Language (HLBL) User’s Guide* (B0400DF) for the maximum number of comma-separated arguments before a data type specification.

```
Example:    THIS_VAR, THAT_VAR, OTHER_VAR: R;
            STR_VAR1, STR_VAR2, OTHER_STR: S12;
            MY_BOOLS, YOUR_BOOLS: B [5];
```

The type specifications for local block variables are:

B	=	Boolean
I	=	Long Integer
R	=	Real
S	=	String of 80 Characters

S6	=	String of 6 Characters
S12	=	String of 12 Characters

User-labeled parameters can be referenced by the Sequence block's user-defined algorithm. There are a fixed number of each of the following types: real, long integer, Boolean, and string. All types except string and the data store arrays can be linked with parameters in other blocks and compounds and shared variables. The strings and data store arrays are settable but not connectable.

The standard parameters, all of which may be user-labelled, are:

BI0001 – BI0024:	Boolean Inputs
II0001 – II0008:	Long Integer Inputs
RI0001 – RI0015:	Real Inputs
SN0001 – SN0010:	Strings (always up to 80 characters) *
BO0001 – BO0016:	Boolean Outputs
IO0001 – IO0005:	Long Integer Outputs *
RO0001 – RO0015:	Real Outputs *
BA0001[16]:	Boolean Array Data Store *
BA0002[16]:	Boolean Array Data Store *
BA0003[16]:	Boolean Array Data Store *
BA0004[16]:	Boolean Array Data Store *
IA0001[16]:	Long Integer Array Data Store *
RA0001[16]:	Real Array Data Store *
RA0002[16]:	Real Array Data Store *

* Not available in the Monitor (MON) block (see page 143).

Standard and user-labeled parameters are described in *Integrated Control Block Descriptions* (B0193AX). Configuring parameters is described in *Integrated Control Configurator* (B0193AV) or *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

An include file can be any set of HLBL statements. Use include files to define specific constructs such as sophisticated WAIT loops or complicated expressions of a set of parameters, or to define objects with a global scope such as symbolic constants, subroutines, or standard block exception handlers. You cannot compile include files separately.

The subroutine allows you to specify a general piece of control logic just once and apply it as often as required in the block algorithm. A subroutine is a sequence of HLBL user-defined statements that can be called from the sequence block's main code or from another subroutine. Subroutines can use any HLBL statement except for standard block exception handlers.

You can use a user-defined number of arguments to parameterize a subroutine. The data types of these arguments must be one of the data types supported in HLBL for block parameters and local variables.

You cannot install a sequence subroutine in a station as an independent entity and you cannot access it from outside the sequence block in which it is installed.

An Independent, Dependent, or Exception block may have subroutines. Monitor and Timer blocks do not support subroutines.

Refer to the table "Sequence Compiler Limits" in *High Level Batch Language (HLBL) User's Guide* (B0400DF) for subroutine related limits.

A Standard Block Exception Handler (SBX) is a user-specified section of HLBL statements that allows the sequence block to react to an operational error during automatic execution or to an outside interruption during normal block operation.

There are five events for which SBXs can be specified. Two are error handling SBXs:

- ♦ User errors (OP_ERR between 2000 and 3000)
- ♦ System errors (all other errors).

The other three are state change SBXs:

- ♦ Switch to Inactive
- ♦ Switch to Manual
- ♦ Switch to Paused.

Sequence language statements define the sequential control algorithm, as specified by the user. The Sequence language is described in *High Level Batch Language (HLBL) User's Guide* (B0400DF).

Monitor Block

A Monitor (MON) block contains up to 16 user-defined Boolean expressions called cases. The result of the evaluation of a monitor case is stored in the associated boolean output parameter. When one of the cases evaluates to true, the MON activates a sequence block (EXC, DEP, IND, or MON). In this way, up to 16 blocks can be activated from the MON block.

A Monitor block is composed of:

- ♦ Standard Parameters
- ♦ Blocktype Identification
- ♦ Symbolic Constants
- ♦ User Labels
- ♦ Monitor Cases (up to 16).

User-labeled parameters can be referenced by the Monitor block's user-defined algorithm. There are a fixed number of each of the following types: real, long integer, and boolean. All types can be linked with parameters in other blocks and compounds and shared variables.

Standard and user-labeled parameters are listed and described in *Integrated Control Block Descriptions* (B0193AX). Configuring parameters is described in *Integrated Control Configurator* (B0193AV) or *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

A monitor case consists of a monitor condition and an optional activation request that is performed when the condition is true.

Example:

```
0001 WHEN level_hi DO :TANK_1:HI_LEVEL_EXC
```

- ♦ The case number is 0001.
- ♦ The condition is "level_hi", where level_hi is a user labeled parameter.

- The request is “DO :TANK_1:HI_LEVEL_EXC”. “HI_LEVEL_EXC” is the block name for a sequence block in the compound “TANK_1”. This block will be activated when the condition “WHEN level_hi” is evaluated true.
- When level_hi is true:
  - The TRIPPD parameter of the block goes true and the faceplate indicates that the block has a trip.
  - The boolean output associated with the case is set, in this example, BO01.
  - If the case is still true when HI_LEVEL_EXEC deactivates, the case trips again and activates the block again.
  - BOn and TRIPPD remain set as long as the activated block remains active, even if it goes to manual.
  - MON block parameter ACTPAT determines active patterns.

Timer Block

A Timer (TIM) block keeps track of time while control strategies are executed. It is composed of standard parameters and four timers. TIM blocks do not contain any Sequence language statements.

Standard parameters show block operation details and allow you to control block operation.

A timer is composed of a real and a Boolean parameter. The Boolean parameter value determines whether the real parameter is updated or not when the block is processed. When the Boolean value is true, the real parameter is updated. When the Boolean value is false, the real parameter is not updated.

The TIM block is processed when the compound in which it resides is On and the block is in Auto. When a TIM block is processed, timers that have been started are updated every scheduled Block Processing Cycle (BPC). Timers are started by an external source, such as a statement in a Sequence block.

Standard parameters are listed and described in *Integrated Control Block Descriptions* (B0193AX). Configuring parameters is described in *Integrated Control Configurator* (B0193AV) or *I/A Series Configuration Component (IACC) User's Guide* (B0700FE).

Sequential Control Block States

Block states describe the operational behavior of a block. All blocks have the Application states Manual and Auto.

In addition to the Application states, the sequential control blocks have the following Sequence states: Active, Inactive, Paused, and Tripped. Tripped applies only to the Monitor block. The set of operational modes for sequential blocks include:

Table 9-1. Operational Modes

Application States	Sequence States	Transition
Auto	Active	To_Manual
Semi-Auto (Step Mode)	Inactive	To_Inactive
Manual	Paused	To_Paused

Table 9-1. Operational Modes (Continued)

Application States	Sequence States	Transition
Subr-Trace	Tripped	
SBX-Trace	Suspended on SENDCONF	

The Sequence states and the Application states control sequential control block algorithm execution and the operational state of block outputs.

Application States

The Application states, Auto, Semi-Auto, and Manual, control the operational state of a block's outputs. In conjunction with the Sequence states, they also control sequential control block algorithm execution.

The Application state is determined by the value of the block's MA parameter. When MA is true, the block is in the Auto state. When MA is false, the block is in the Manual state.

Another block parameter, RSTMA, controls the value of the MA parameter when the compound changes from Off to On. When RSTMA is 0, MA becomes false; when RSTMA is 1, MA becomes true; when RSTMA is 2, MA does not change upon the compound switch. You set the value of RSTMA during block configuration using a control configurator.

— NOTE —

For the EXC, IND, DEP, MON, and TIM blocks, the state of RSTMA is ignored when the Control Processor restarts because of an On-Line Upgrade (OLUG).

Auto State

In the Auto state, a block's output parameters are secured. This means that the block algorithm is responsible for updating the output parameters. External sources (other blocks and applications) cannot write values to block output parameters.

Sequential control block algorithms are processed as follows in the Auto state:

- ♦ TIM block timers that have been started are updated once every scheduled BPC. A timer is started with a START_TIMER statement in an IND, DEP, or EXC block.
- ♦ MON block cases are evaluated each scheduled BPC. If a case trips, it may lead to activation of an EXC block. If the EXC block activated is remote, tripping and untripping may require several BPCs to complete.
- ♦ IND, DEP, and EXC blocks process the number of statements specified by the block's BPCSTM parameter each scheduled BPC. When a statement requiring suspension such as WAIT or WAIT UNTIL executes, fewer statements may be processed than the number specified by BPCSTM.

Since Sequence block algorithms vary in length, a block may execute completely in one BPC or it may require several BPCs to execute completely.

Once all statements have been executed, the Sequence block is no longer processed unless a statement in the user algorithm causes it to repeat.

If the sequence block contains state change logic, that logic will be executed if the block switches from the Active/Auto mode to the Inactive, Manual or Paused state. The logic for the state changes are user-defined in SBXs 3, 4, and 5.

The order of statement execution can be altered while in the Auto state. An operator, at a user-defined or default display, can redirect statement execution to a new start location by writing the desired statement number to the STMRQ parameter.

Semi-Auto State

In Semi-Auto (or Step mode), the sequence block executes only the HLBL statements that belong to a particular step. Statement execution stops when a step boundary is passed. Steps can be requested in any order, at a user-defined or default display, by writing the desired step number to the STEPRQ parameter. The block is divided into steps by means of the step labels in HLBL.

If the sequence block contains state change logic, the corresponding logic will be executed if the block switches from the Active/Step mode to the Inactive, Manual, or Paused state.

The logic for the state changes are user-defined in SBXs 3, 4, and 5.

Manual State

In the Manual state, output parameters are not secured. This means that external sources (other blocks and applications) can write values to the block's output parameters. Unlike continuous control blocks, sequential control blocks may have their own statements write to their own output parameters while the block is in Manual.

Sequential control block algorithms are processed as follows in the Manual state:

- ♦ TIM blocks are not processed.
- ♦ MON block cases are executed one at a time by user request. The user selects a case for execution, from a user-defined or default display, by writing the desired case number to the CASERQ parameter.

If a requested case trips (that is, the evaluated condition is true), a block activation request is executed. After the case has been processed completely (tripped and untripped), the standard parameter CASENO is set to indicate the number of the next case. The next case is not evaluated unless requested. The TRPCHG parameter is incremented each time a case changes to or from the tripped state.

The processing of EXC blocks already activated by tripped cases in the MON block are not affected by other case evaluation requests to the MON block.

- ♦ IND, DEP, and EXC block statements are executed one at a time by request. You can select a statement for execution from a user-defined or a default display by:
  - ♦ Writing to the parameter STEPRQ the step number which begins with the requested statement.
  - ♦ Writing to the parameter STMRQ the number of the requested statement.
  - ♦ Setting NXTSTM to true.

A statement requiring several BPCs to execute, such as a WAIT statement, need only be requested once to initiate execution.

Statement execution can be cancelled by requesting that another statement be executed. The standard parameter STMNO indicates the number of the statement currently executing. When the statement finishes execution, STMNO is set to the number of the next statement dictated by execution flow. That statement is not executed unless requested by:

- ♦ Writing its number to the parameter STMRQ
- ♦ Setting NXTSTM to true.

When the requested statement calls a subroutine, all the HLBL statements of that subroutine (and any nested subroutine) are executed. The parameter SUBRNO indicates in which subroutine, if any, the currently executed statements reside. The parameter STMNO indicates the statement number within that subroutine.

The Subr-Trace and SBX-Trace modes enable you to single step through statements of subroutines and SBXs. You can switch the block into one of the Trace modes only when the block is in the Active/Manual state.

Subr-Trace is a substate of the Manual state that enables you to single-step through a subroutine. You enter this substate by selecting the “SUBR TRACE” button in the ALL CODE display. This enters the integer value “1” into the TRACRQ parameter which, in turn, sets the block into the Subr-Trace mode when the block is in Manual. After granting the request, the block resets TRACRQ to 0.

Once in the Subr-Trace mode, you “select” a subroutine by requesting a call-subroutine statement in the block’s main section. The block is then idle before the first statement in the requested subroutine. You can then single-step through the subroutine statements by toggling the NXTSTM parameter. STEPRQ and STMRQ cause the execution of a single statement in the block’s main section.

When you switch into the SBX-Trace mode, the block environment (that is, step, subroutine, statement number) is saved. The block returns to this environment when you exit the SBX-Trace mode.

Once in the SBX-Trace mode, you “select” an SBX by setting the SBXRQ parameter to a value of 1 to 4. SBX5 (a switch to Paused) applies only to the DEP block (the block ignores out of range values). When you select an SBX, the block idles at the first statement within that SBX. You can then single-step through the SBX statements by toggling the NXTSTM parameter. The block ignores step- and statement-requests while it is in the SBX-Trace mode.

In the Manual, Subr-Trace, and SBX-Trace modes, the block does not secure its output parameters. External sources (other blocks and applications) can write values to the block’s outputs. While the block is in one of these modes, the EXC, DEP, or IND block algorithm can update its output parameters after a step-, statement-, or next-statement request.

To exit from the Trace mode, select the “TRACE” field in the faceplate.

Auto/Manual Transitions

You can change the block Auto/Manual state from external sources such as user-defined and default displays, other blocks, and applications.

If a statement is in execution when you request a state change, the statement’s execution is completed as if it had begun in the requested state. Then any following statements are executed as appropriate for the requested state.

NOTE

If one or more cases are making a transition from Active to Tripped (for example, the blocks to be activated are remote blocks) when you change a MON block from Automatic to Manual, then the block activation is completed but the cases do not trip.

Sequence States

The Sequence states, Active, Inactive, Paused, and Tripped, in conjunction with the Application states, control sequential control block algorithm execution. The Sequence states are determined by the values of the block's ACTIVE, PAUSED, and TRIPPD parameters. When ACTIVE is true, the block is in the Active state. When ACTIVE is false, block is in the Inactive state.

Another block parameter, RSTACT, controls the value of the ACTIVE parameter when the compound in which it resides changes from Off to On, or when the control processor in which it resides undergoes a restart operation, as follows:

- RSTACT = 0: ACTIVE is false.
- RSTACT = 1: ACTIVE is true.
- RSTACT = 2: ACTIVE retains the value from the checkpoint file when the Control Processor is restarted, or remains 0 when the compound makes a transition from Off to On.

NOTE

For the EXC, IND, DEP, and MON blocks, the state of RSTACT is ignored when the Control Processor restarts because of an On-Line Upgrade (OLUG).

When a DEP block is in the ACTIVE state, it may also be in the PAUSED state. A DEP block is Paused when the PAUSED parameter is true.

When a MON block is in the Active state, it may also be in the Tripped state. A MON block is tripped when the TRIPPD parameter is true.

Table 9-2. Sequence States

State	Description
Inactive	An IND, DEP, or EXC block is not executing any statements or a MON block is not evaluating conditions.
Active	An IND, DEP, or EXC block is executing statements or a MON block is evaluating conditions.
Paused	A DEP block's execution is suspended because one or more EXC blocks in the same compound are Active. The DEP block remains suspended until all such EXC blocks are done executing.
Tripped	A condition evaluated by a MON block causes it to activate other blocks. The MON block remains tripped until all activated blocks are done executing.

Active State

In the Active state, an IND, DEP, EXC, or MON block is processed. (The TIM block does not have an Active state. It is processed when the compound is On and the Application state is Auto.) How statements are executed depends upon the Application states Auto, Step, and Manual.

Inactive State

In the Inactive state, an IND, DEP, EXC, or MON block is not processed. (TIM blocks do not have an Inactive state. TIM blocks are not processed when the Application state is Manual.)

Active/Inactive Transitions

You can change the block Active/Inactive state from external sources such as user-defined and default displays, other blocks, and applications.

When a linkage to the ACTIVE parameter exists, it is secured. This means that you cannot access the parameter directly. To activate or deactivate the block, you can:

- ♦ Access the ACTIVE parameter through the source of the linkage
- ♦ Write the number of a non-existing statement to STMRQ.

Writing the number of a non-existing statement to STMRQ directs statement execution to the end of the algorithm. Although the block is in effect deactivated, the ACTIVE parameter remains true until it has been released. When released, it is automatically set to false.

Paused State

In the Paused state, DEP block statement execution is suspended due to active EXC blocks in the same compound. The PAUSED parameter indicates whether a DEP block is in the Paused state. When PAUSED is true, the block is in the Paused state.

Tripped State

In the Tripped state, a MON block has one or more cases tripped. A case trips when it is evaluated as true and activates another block. The TRIPPD parameter indicates whether a MON block is in the Tripped state. As long as at least one case is tripped, the TRIPPD parameter is true; otherwise, TRIPPD is false.

Transition States

The Transition states, To_Inactive, To_Manual, and To_Paused, are intermediate states that the Sequence Control block assumes while the block is executing one of the three standard block exception handlers (SBXs 3, 4, and 5) that are provided for state change handling.

To_Inactive State

The To_Inactive state is an intermediate state that an IND, DEP, or EXC block assumes while SBX 3 is executing. SBX 3 is the user-defined, user-enabled response to an externally initiated change of block state from the Active/Auto (or Active/Step) state to the Inactive state.

The MON and TIM blocks do not have a To_Inactive state since they do not contain SBXs.

To_Manual State

The To_Manual state is an intermediate state that an IND, DEP, or EXC block assumes while SBX 4 is executing. SBX 4 is the user-defined, user-enabled response to an externally initiated change of block state from the Active/Auto (or Active/Step) state to the Manual state.

The MON and TIM blocks do not have a To_Manual state since they do not contain SBXs.

To_Paused State

The To_Paused state is an intermediate state that a DEP block assumes while SBX 5 is executing. SBX 5 is the user-defined, user-enabled response to an externally initiated change of block state from the Active/Auto (or Active/Step) state to the Paused state.

The IND and EXC blocks do not have a To_Paused state since they do not have a Paused state. The MON and TIM blocks do not have a To_Paused state since they do not contain SBXs.

Compound Sequence State

The collective operational state of all sequential control blocks in a compound is represented by the compound parameter SSTATE. SSTATE can be one of three values:

- SSTATE = 0 (Inactive)
- SSTATE = 1 (Active)
- SSTATE = 2 (Exception).

See Table 9-3 for further definition.

Table 9-3. Compound Sequence State

SSTATE Value	Description
Inactive (SSTATE = 0)	All MON blocks and all Sequence blocks (IND, DEP, or EXC) in the compound are Inactive.
Active (SSTATE = 1)	One or more MON blocks and/or one or more Sequence blocks (IND and DEP) in the compound are Active. No EXC blocks are active.
Exception (SSTATE = 2)	One or more EXC blocks in the compound are Active. IND blocks may be active; TIM blocks may be running.

When a compound is switched OFF, all the sequence blocks in that compound go to the Manual state, thereby releasing their output parameters.

Sequence Processing

Sequential control blocks are processed every scheduled Block Processing Cycle (BPC) as defined for the Control Processor in which they operate. The following figure shows the processing order within a scheduled BPC.

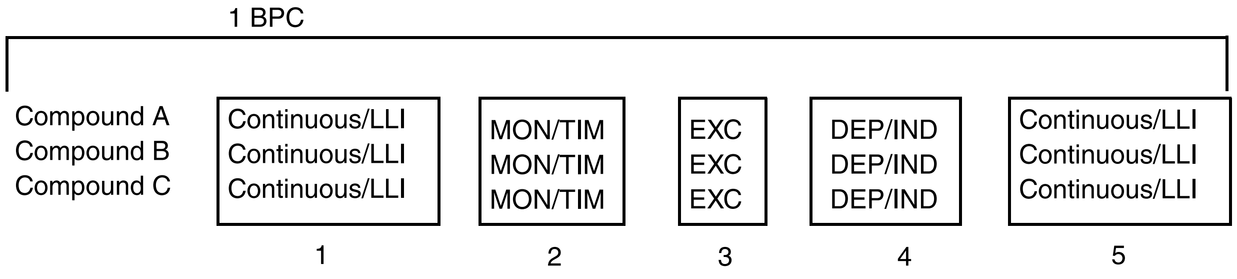


Figure 9-1. Processing Order Within a Scheduled BPC

The scheduled BPC is determined by the block parameters PHASE and PERIOD. The PHASE parameter specifies when a block should be executed relative to the other blocks in that PERIOD. The PERIOD parameter specifies how frequently a block should be executed. For more information, refer to "Block Phasing" on page 64, and "Relationship Between Block Period and Phase" on page 65.

When a sequential control block is activated, it begins executing its block algorithm in the next scheduled BPC as defined by the block parameters PERIOD and PHASE.

When a MON or TIM block is processed, its entire algorithm is executed each scheduled BPC until deactivated.

When an IND, DEP, or EXC block is processed, a specified number of statements in the algorithm are executed each BPC. The number of statements processed is determined by the value of the block's BPCSTM parameter. When the last statement is executed, the block automatically deactivates itself.

Sequence language statements are executed in the order programmed by the user. Statement execution continues in a given BPC until:

- The number of statements specified by the BPCSTM parameter are executed in Auto, or in Step mode.
- The last statement of a step is executed in Step mode.
- One statement is executed by request in Manual for an IND, DEP, or EXC block.

The number of statements executed in automatic may be less than the number specified by the BPCSTM parameter if an executed statement requires more than one BPC to complete. For example:

- A statement makes a request to access a block parameter in a remote Control Processor.
- A WAIT or WAIT UNTIL statement has a wait condition exceeding the BPC.

Sequence Processing Overrun

If the block processor cannot process all scheduled blocks in a BPC, there is BPC overrun. In a serious system overload, sequence (and continuous) blocks are always processed, even if it delays the next BPC for the remaining blocks not yet processed.

Useful Hints on SBX Programming

NOTE

Only one exception handler should be active at any one time. Nested exception handling can cause a system failure.

Operational Error SBXs

The sequence Interpreter has comprehensive error protection. In case of any operational error, the parameter OP_ERR assumes an integer number that indicates the kind of failure.

If OP_ERR has a value larger than 3000, a serious error has occurred. When a serious error has occurred, you must be careful when defining SBX (exception logic) TO_SYS_ERROR. An attempt to continue to run or retry the same statement may cause memory violation, memory corruption, or unpredictable control algorithm behavior. The nature of this kind of error is usually a software or hardware problem. Therefore, you should use basic logic (no subroutine calls and external references) to write OP_ERR, SBRNO, SBXNO, BLOCK_STMNO, and STMNO, then stop the block, collect any related information, and report the information to Invensys Global Customer Support at 1-866-746-6477.

All other error conditions, including user errors (SBX TO_USR_ERROR: 2000 < OP_ER < 3000), allow retrying the same statement. (If you do this, you should be careful to avoid infinite loops.) For most of the cases the "retry" does not clear the error.

Example: OP_ERR 2314 "string type expected in string expression" does not go away after retry. The important thing to remember is that if you decide to continue you may end up with unexpected results.

```
   LOC_VAR, STR1 : S;
   STATUS        : B;
   LOC_VAR    := “:CONTROLLER_1.BI0001”;
   STATUS     := :’LOC_VAR’;
   STR1       := “:MAIN_LOOP.BO0001”;
   LOC_VAR    :=  STRING  :’STR1’;
   :’LOC_VAR’ := TRUE;
```

STR1 is not a FPN (full path name) for a string type parameter. In this case LOC_VAR keeps its value unchanged and CONTROLLER_1 BI0001 parameter is set to "TRUE".

The following example may cause problems in algorithm behavior:

Example:

```
INDEPENDENT_SEQUENCE
{* A TEST CASE *}
CONSTANTS
VARIABLES
USER_LABELS
SUBROUTINE SUBR2(IN  II : I)
VARIABLES
STATEMENTS
   II0002 := II;
ENDSUBROUTINE

SUBROUTINE SUBR3()
VARIABLES
   RR : R;
STATEMENTS
```

```
   RR := :WRONG:PATH.NAME; {* OP_ERR ‘-1’ *}
IO0003 := ROUND(RR);
 {* All subroutines in the block share the same memory for local variables and
arguments. In the example, RR has a value of II0001 which is left after SUBR2
call.
When II0001 is equal to ‘1’, ‘01 00 00 00’ is not a valid binary representation
of a float number.
 *}
ENDSUBROUTINE
SUBROUTINE SUBR1()
VARIABLES
STATEMENTS
   CALL SUBR2(II0001);
   CALL SUBR3();
ENDSUBROUTINE
BLOCK_EXCEPTION TO_SYS_ERROR
STATEMENTS
   SN0001 := “ERROR= “,OP_ERR,” B_STMNO= “, BLOCK_STMNO;
ENDEXCEPTION
  {*Wrong practice: continue to run no matter what! *}
STATEMENTS
<<START>>
   CALL SUBR1();
   WAIT 0.1;
   GOTO START;
ENDSEQUENCE
```

The FCP280, FCP270, and ZCP270 zero out denormalized values for cases like the above to avoid system failure but this may cause unexpected algorithm behavior by using a value of zero instead of what would be appropriate.

A common practice might be:

* TO_SYS_ERROR SBX – stop running if OP_ERROR > 3000

* TO_USR_ERROR SBX – stop running except OP_ERR 2401 if found necessary for the algorithm strategy.

You should be careful when continuing under error conditions triggered TO_SYS_ or TO_USR_ERROR SBXs.

SBX RETRY Logic

If you want the same number of retries on any HLBL statement which caused an operational error, you must build the logic to properly reset the retry counter. It is important to remember that each HLBL statement can produce more then one error. Therefore, you should choose a proper limit for retries since the RETRY instruction repeats the entire statement not just the faulty element. An example of the proper implementation of the “retry” is as follows:

```
RETRY_CNT   : I;  {* to count retries on individual statements *}
SAVED_STMNO : I;
SAVED_STMNO := -1;  {* initialization to guarantee miscompare with
   BLOCK_STMNO *}

  BLOCK_EXCEPTION TO_SYS_ERROR (or TO_USR_ERROR)
IF OP_ERR <> ERR1 AND OP_ERR <> ERR2 AND OP_ERR <> ERR3 THEN
        GOTO FINAL;
{* ERRx are errors to retry. For TO_SYS_ERROR SBX the statement
   could be replaced with IF OP_ERR > 3000 THEN GOTO FINAL;
*}
ENDIF
IF BLOCK_STMNO <> SAVED_STMNO   THEN
   RETRY_CNT    := 0;
   SAVED_STMNO  := BLOCK_STMNO;
ENDIF;
IF RETRY_CNT < LIMIT THEN
```

```
   RETRY_CNT := RETRY_CNT + 1;
 {* In some cases, you may want to add the “WAIT” statement at this
point.   For example, OP_ERR ‘-1’ or ‘-45’
 *}
RETRY;
ENDIF;
  <<FINAL>>
SENDMSG (“ERROR! op_err=”,OP_ERR,” stm=”, BLOCK_STMNO,
    “ SBXNO=”,SBXNO) to SN0001;    {* or to MSGGRx *}
EXIT;
  ENDEXCEPTION
```

The above SBX design assumes that SAVED_STMNO has been initialized to -1:

1. In the first statement of the main code section, the first statement of each subroutine, and the first statement of the state change SBX (TO_INACTIVE, TO_MANUAL, TO_PAUSED).
2. In the last statement of the state change SBX unless the logic terminates the block.
3. In the first statement after any label which is used as the “GOTO” statement argument to change execution flow backwards or to jump from any type of SBX to the block main section.

Operational Error Conditions Inside Subroutines

Presently in an SBX there is no information about the subroutine number and statement number of a subroutine which has triggered an SBX execution. BLOCK_STMNO always refers to the main section. In the SBX the STMNO refers to the current SBX statement, SUBRNO is 0 unless the SBX itself calls a subroutine. To make the retry logic work properly, SAVED_STMNO must be reset before each subroutine statement which may trigger the SBX retry logic. In most cases, this results in a significant overhead.

An alternative is to use a more elaborate approach which is primarily based on an assumption that the retries make sense only on statements with external references. Related OP_ERRs which may be fixed with retries are:

- All negative values. They are related only to external references outside the controller or to targets which do not exist.
- “3”: Attempt to write a secured parameter (for example, an output parameter while the block is in AUTO).
- “18”: Locked access (for example, the OWNER parameter of a block).
- “2401”: Attempt to set the ACTIVE parameter to a value that is already active.

The idea of the approach is to reset SAVED_STMNO inside subroutines before each statement with:

- External references (including ACTCASES, ABORT and ACTIVATE statements) to a remote station database.
- Any of external sets including ABORT, and ACTIVATE statements which may produce OP_ERR 3, 18, or 2401 if retry is expected.

If speed and memory are not major issues, instead of resetting the SAVED_STMNO multiple times for the kind of statement listed above, you may program HLBL logic to avoid usage of this statement more than once inside of each subroutine. To remove multiple external references from the subroutines, you can create a set of “set” and “get” subroutines to be called when needed.

Example:

For SET_VALx() (where 'x' stands for R,I,B,S data type)

```
SUBROUTINE SET_VALR(IN FPN : S; INOUT VAR : R)
STATEMENTS
 SAVED_STMNO := -1;
 VAR := :'FPN';
ENDSUBROUTINE

SUBROUTINE SET_VALI(IN FPN : S; INOUT VAR : I)
STATEMENTS
 SAVED_STMNO := -1;
 VAR := :'FPN';
ENDSUBROUTINE
SUBROUTINE SET_VALB(IN FPN : S; INOUT VAR : B)
STATEMENTS
 SAVED_STMNO := -1;
 VAR := :'FPN';
ENDSUBROUTINE
SUBROUTINE SET_VALS(IN FPN : S; INOUT VAR : S)
STATEMENTS
 SAVED_STMNO := -1;
 VAR := STRING :'FPN';
ENDSUBROUTINE
```

NOTE

To continue after unsuccessful retries the you can secure the results by writing the default value into VAR. Example:

```
SUBROUTINE SET_VALB(IN FPN  : S);
 INOUT VAR : B;
 IN DFLT : B )
STATEMENTS
 SAVED_STMNO := -1;
 VAR := DFLT;
 VAR := :'FPN';
ENDSUBROUTINE
```

Operational Error Condition Inside State Change SBX

Presently in an operational error SBX there is no information about the state change SBX which has triggered an operational error SBX execution. The situation is exactly the same as described for subroutines. All recommendations for the retry logic which were given before are suitable for the state change SBXs.

Complex Statements

You should avoid complex HLBL statements such as the following:

```
SAVED_STMNO := -1;   {* the initialization is not needed for the main
code section.  *}
IF :COMP1:BLOCK1.BO0001 <> :COMP2:BLOCK2.BO0002 THEN
  ...........
ENDIF
```

This statement can produce OP_ERR "-1" twice and RETRY_CNT is not reset for the second one. It is preferable to avoid usage of two or more external references in the single statement. You should use extra assignment statement instead:

```
SAVED_STMNO := -1;
TMP_BOOL := :COMP2:BLOCK2.BO0002;
```

```
SAVED_STMNO := -1;
IF :COMP1:BLOCK1.BO0001 <> TMP_BOOL THEN
  ............
ENDIF
```

You may also double the number of retries by setting a proper value for LIMIT (the variable LIMIT was used in the above example for SBX programming).

Usage of the SUBR_LEVEL Standard Parameter

Some applications use SBX logic to decide to “retry” or to “not retry” a particular HLBL statement on a particular OP_ERR. Those applications may need more precise identification for subroutines which triggered the SBX. While the subroutine statement identification is the user’s responsibility, there are ways to identify the subroutine.

The subroutine can be identified in the SBXs by the BLOCK_STMNO and SUBR_LEVEL standard parameters in cases when:

- HLBL logic does not have subroutine calls from other subroutines (the maximum nesting level is one).
- All subroutines have not more than one call of another subroutine (no limits on the nesting level).

Remember that if the SBX does the identification inside a subroutine (the SBX calls another subroutine), the SUBR_LEVEL is incremented by one.

If subroutines are called from the state change SBXs and the block main section, you user must set the special flag at the beginning of the each state change SBX to be used in the operational error SBX to ensure the correct subroutine identification. The flag must be cleared at the end of the state change SBX.

10. System/Control Configuration Concepts

This chapter discusses various integrated control configurator concepts which are common to all the system/control configurators; for example, hardware and software interfaces, and control station, library volume, and compound configuration.

Configurators

Invensys provides a range of software packages to perform the system/control configuration for your system, as listed below. Certain concepts are common to each of these configurators, and these are described in the remainder of this chapter.

- System configuration is the configuration of the overall Foxboro Evo Process Automation System – the availability and relationships between the hardware – Foxboro Evo Control Editors, IACC, and SysDef.
- Control configuration is the definition of the Process itself, or specifically, the Process Control Database – control loops, blocks, etc. – Foxboro Evo Control Editors, IACC, and ICC.

Foxboro Evo Control Editors

The Foxboro Evo Control Editors are a suite of engineering and configuration tools built on the ArchestrA Integrated Development Environment (IDE). They provide a complete set of functionality for configuring, operating and maintaining a Control Core Services system, including developing control strategies from compounds, blocks and ladder logic, and building user displays.

For a full overview of the available Control Editors packages (components), refer to *Foxboro Evo Process Automation System Deployment Guide* (B0750BA).

The software packages which comprise the Control Editors functionality are described in the following documents:

- *Appearance Object Editor User's Guide* (B0750AE)
- *Bulk Data Editor User's Guide* (B0750AF)
- *Common Graphical Editor Features User's Guide* (B0750AG)
- *Block Configurator User's Guide* (B0750AH)
- *Control Database Deployment User's Guide* (B0750AJ)
- *PLB Ladder Logic Editor User's Guide* (B0750AK)
- *Sequence Block HLBL Editor User's Guide* (B0750AL)
- *Sequence Block SFC Editor User's Guide*(B0750AM)
- *Strategy Editor User's Guide* (B0750AN)
- *Configuration Utilities User's Guide* (B0750AZ)
- *Logic Block Editor and Troubleshooting Tool* (B0750BL)

- *Scripting with Direct Access User's Guide* (B0750BM)

I/A Series Configuration Component (IACC)

The I/A Series Configuration Component (IACC) provides a graphical user interface for developing process control strategy diagrams (CSD), implementing CSD templates, configuring the required compounds, blocks and equipment control blocks, and downloading the configuration database to the control processors. For more information on IACC, refer to the following:

- *I/A Series Configuration Component (IACC) User's Guide* (B0700FE) and the latest release notes
- *Learning to Use IACC* (B0400BT)
- *Intelligent Design Studio (IDS) Library for IACC* (B0400BQ)

Alternately, you can use the Integrated Control Configurator (ICC) to configure your process control system. This chapter provides an overview of the features and functions of the ICC, which is both a configurator and a compound/block editor. As a configurator, it allows you to add or modify Fieldbus Module (FBM) software data as necessary.

Hardware configuration takes place in the System Configurator as part of system startup, expansion, or maintenance. A list of the configured modules is passed to the control configurator, where it is accessible from the menu-bar function combination SHOW/Configured ECBs.

When you add an FBM to your system, you create an Equipment Control Block (ECB) for each FBM selected. The ECB is the "holding place" for that FBM's software data. ECBs are created using the Insert Block/ECB function of the ICC main menu or the FBM/Fix All menu-bar function.

Additionally, database files are provided for each library volume as it is created in the system. Once installed, these files can be identified by the lowercase characters "vol."

All modifications to this physical data are made through the Integrated Control Configurator. These modifications can be made to on-line stations or off-line library volumes. As a compound/block editor, the Integrated Control Configurator provides compound/block-building templates along with a full range of editing functions.

With it you can:

- Create and integrate continuous, sequence, and ladder logic type blocks in a single compound structure
- Group and connect compounds
- Modify, copy, and delete compounds and blocks
- Configure and modify Fieldbus modules
- Assign control schemes to stations in a distributed processing environment
- Build and maintain compound libraries
- Add device ECB blocks on line.

Configuration Aids

Configurator selections are either from a menu bar along the top of the workstation screen, or from menus that appear, making additional choices possible.

From initial start-up, your directions on "what to do next" are provided by the logical structure of the configurator.

As all-inclusive as the configurator program is, however, there are times when more information and further assists are necessary. The following aids provide that assistance.

Help Screens

Help screens provide necessary backup information. They provide help on issues such as ranges, choices, and error messages. They also provide help on the help function itself.

Parameter List

When you edit a compound/block, you can select either the standard parameters, or the complete list of parameters for that compound/block.

Sequential Control Block Reference Information

When a block is defined as one of the Sequence types, the configurator program transfers control to the Sequence Environment. Selecting EDIT from this environment provides a character editor for building and compiling source code for user-defined sequence blocks.

After the source code is compiled, control is again resumed by the configurator program.

Hardware Interface

The configurator operates with any Control Processor or any Application Workstation. It also operates with any Workstation Processor supported by the Human Interface (HI) library.

The configurator must have a display screen and a keyboard.

User Interface

An alphanumeric keyboard is the primary input device.

A secondary input device is a mouse or trackball.

Software Interface

The configurator uses software processing routines to find:

- Control Processor letterbug names
- Fieldbus Module letterbug names for system-configured FBMs
- Control Processor database file (checkpointed) name
- Control Processor file status
- Control type availability for a Control Processor.

Compound Summary Access

The Compound Summary Access (CSA) utility displays the list of active compounds and returns the associated letterbug (if any) to the configurator. This provides a means of determining if a specific active compound name exists in the system.

CSA is a search and find utility and the "doorway" to the configurator.

Refer to *Integrated Control Configurator* (B0193AV), Chapter 2, for more on CSA.

Software Processing

Software processing provides the functions to:

- ♦ Initialize the software processing facilities.
- ♦ Reset access to the local host.
- ♦ Determine the host associated with a specified Control Processor.
- ♦ Return a file containing the letterbug and note the installed (active) Control Processors.
- ♦ Return a file containing a list of the active printers.
- ♦ Return a file containing a list of the active Workstation Processors.
- ♦ Return a file containing a list of the active historians.
- ♦ Lock or permit access to the Control Station database file associated with a specified Control Station letterbug.
- ♦ Lock or permit access to a library volume database.
- ♦ Unlock a Control Processor database.
- ♦ Return a list of FBMs, which were configured with the System Configurator, for a specified Control Station letterbug.
- ♦ Return a list of the subset of the above System Configurator-configured FBMs which have actually been created.
- ♦ Return a glossary of equipment configuration parameters for each FBM type, to aid you in configuring an FBM which has not been pre-defined using the System Configurator.

System Monitor Interface

The configurator requests a Control Processor database checkpoint of an active station in the following manner:

- ♦ The configurator finds the name of the system monitor responsible for the station.
- ♦ The configurator requests a checkpoint of the station from that system monitor.
- ♦ The system monitor responds when the checkpoint is in progress.

When the checkpoint is complete or the checkpoint fails (times-out), the system monitor sends a message to the configurator indicating the result of the checkpoint. If so configured, the system monitor creates a history of the event and prints a message to a communications server.

Control Processor Database Installer

This Control Processor-resident routine:

- ♦ Receives and acknowledges INSERT commands sent by the configurator, communicated through an Interprocess Communication (IPC) interface.
- ♦ Updates the controller checkpoint file authorization code at the successful completion of an INSERT command.

- Initializes a compound or block after a modification is made.
- Provides single-channel, single-instance registration with the IPC subsystem, guaranteeing that only one control configurator can connect to one controller at a time.

Configuration Target

You can select either a Control Station or a Library Volume as your configuration target.

Control Station Configuration

Before beginning an editing session with a new configuration target, the configurator:

- Retrieves all the appropriate station data
- Provides a list of the Control Stations that are configured for the system
- Compares the authorization codes of the resident controller database and the checkpointed controller database
- Provides the means to make necessary adjustment(s) to mismatched, invalid data, or disconnected hardware.

When you select **DONE** on the menu bar to exit the edit session, all data pertaining to the selected station is validated and checkpointed.

Library Volume Configuration

Before beginning an editing session with a new Library Volume, the configurator provides a list of volume letterbugs which have been configured for the user's system. Once a volume letterbug has been selected and the edit session initiated, the session status field displays INACTIVE, indicating that there is no Control Station involved. The rest of the user interface is similar to that used for Control Stations. When the user selects **DONE** from the menu bar, the volume data is validated and the edit session is exited.

Compound Configuration

When you choose a Compound name, the configurator determines if the compound name exists, is valid (up to 12 characters), and is unique in the system.

If the compound does not exist, the configurator assumes that you are attempting to add a compound to the "working" library volume. It returns the letterbug of a default library volume, and enters the INACTIVE editing mode.

If the compound does exist, the configurator displays the station or volume letterbug and the compound name, along with a list of all the compounds in that Control Processor (in the leftmost column on the screen). It then enters the ACTIVE mode.

You can also search for a selected compound name. This feature allows you to quickly find a specific compound from all the compounds in the defined domain. Once you make the selection, the Compound Summary File informs the configurator in which checkpoint file the compound resides.

Copying Compounds Between Configuration Targets

The configurator provides an editable area called a paste buffer. The paste buffer resembles a conventional configurator volume, but its contents are not associated with a particular station or library volume (targets).

The paste buffer allows you to copy compounds between stations or volumes. To do this, copy a selected compound into the paste buffer.

Then choose a compound in the list and select `PASTE` to move the contents of the paste buffer into the list just before the selected compound.

Since all compound names are unique within the system, you must rename the compound that you copied into the paste buffer. You can copy more than one compound into the paste buffer. However, if you are copying more than one compound, select `PASTE AND APPEND` rather than PASTE.

PASTE overwrites the contents of the buffer, while PASTE AND APPEND queues the compounds in the buffer for later retrieval.

Checkpoint Files

You can request a Control Station database checkpoint during an edit session by selecting `CHECKPOINT` from the menu bar. This checkpoints the entire database.

Compound and Block Name Conventions

Compound Names

Compound names are up to 12 characters in length and must be globally unique in the active system. When you request that a new compound be added, the configurator station interface checks to see if another compound by the same name exists. If the proposed compound name already exists, the station interface returns a message that the compound cannot be added using that name.

However, if the proposed compound name proves to be unique, the compound is added to the Compound Summary File.

When you delete a compound, the compound is removed from the Compound Summary File.

Block Names

Block names, also up to 12 characters in length, must be unique within a compound but not necessarily throughout the system. The configurator displays an error message if you try to add a block to a compound when another block by the same name already exists within that compound.

Configurator Editing Modes

The configuration target determines the editing mode of the configurator. If the target is a library volume, the editing mode is always INACTIVE. If the target is a station, and an IPC connection can be established with the appropriate station database installer, the editing mode is ACTIVE.

Active Editing

In the Active editing mode, the configurator is connected to a Control Station for interactive display and editing. In this mode you can:

- `ADD` and `DELETE` compounds, blocks, and ECBs
- `MODIFY` compound, block, and ECB parameters and connections
- `COPY` compounds to and from the paste buffer
- `SAVE` and `LOAD` compounds from floppy disk
- `CHANGE` Fieldbus Module per-unit and per-point parameters.

When you finish modifying a block or another modifiable object definition, the configurator automatically installs it when you select `DONE`.

The station returns acknowledgment messages on receipt of these commands informing the configurator either of acceptance or rejection. The configurator maintains its station workfile to reflect these results.

When you change a Fieldbus Module (FBM) configuration with the System Configurator, you can optionally update the Control Station with the change. If the FBM is not known to the station (no ECB present in the station checkpoint file), or the FBM data has either been deleted or modified, you can use the configurator to produce appropriate insert commands so that the station will install the appropriate ECB for the module.

The configurator makes a request to the system monitor to initiate a checkpoint operation and the Control Station responds. When the operation is complete, the configurator is enabled for further edit operations.

In addition, an ECB for a non-system configured FBM can be built from scratch, as in the case of any other new block. The FBM corresponding to the new ECB is specified by entering the DEV_ID parameter in the ECB configuration.

Inactive Editing

In this editing mode, the configurator is not connected to a station but modifies the database of a Library Volume. Editing a Library Volume is essentially the same as editing a station. The internal differences are that:

- There is no station in the communications loop
- There is no validity checking for installation commands
- There is no IPC linkage.

A workfile (which also exists for the active mode) substitutes as the station and adjusts the inactive data base in response to changes.

Configuration Validation

Whenever you install a new block or modify the parameters of an existing block, certain parameters of the affected block are validated according to block-specific rules. If a violation is detected, a warning message is sent to the control configurator and displayed at the console. The same message is shown on the primary page of the block's detailed display. The string parameter ERCODE is also set to contain the violation message. Certain violations are considered fatal errors, and prevent the block from being installed. Others are considered only warnings. These warning messages are handled the same way as error messages, except that the incorrect configuration is preserved in the configuration work file. This allows you to see the error that must be corrected. Validation of block parameters does not proceed past the first error encountered by the block logic. Table 10-1 shows the warning messages. At present, all warnings except “W59 – DUPLICATE OUTPUT CHANNEL” cause the block to be marked Undefined.

Table 10-1. ERCODE Parameter Messages

Message	Meaning
“W43 – INVALID PERIOD/PHASE COMBINATION”	PHASE does not exist for given block PERIOD, or block PERIOD not compatible with compound PERIOD.
“W44 – INVALID ENGINEERING RANGE”	High range value is less than or equal to low range value.
“W45 – CONFIGURATION ERROR IN STEP nn”	A parsing error has been detected in a CALC, CALCA, LOGIC, or MATH block; nn identifies the step in error.
“W46 – INVALID INPUT CONNECTION”	The source parameter specified in the input connection cannot be found in the source block, or the source parameter is not connectable, or an invalid boolean extension connection has been configured.
“W47 – INVALID PARAMETER CONNECTION”	A tuning block is connected to a PIDA block containing a connected tuning constant.
“W48 – INVALID BLOCK OPTION”	The configured value of a block option is illegal.
“W49 – INVALID BLOCK EXTENSION”	An illegal block extension has been configured for EXTBLK (AIN, AINR, MAIN blocks), NLNBLK (PIDA block), or PIDBLK (FBTUNE, FFTUNE blocks).
“W50 – INVALID SIGNAL CONDITIONING INDEX”	An SCI or SCO parameter setting is invalid.
“W51 – INVALID HARDWARE/SOFTWARE TYPE”	An I/O block is connected to an ECB or the wrong type.
“W52 – INVALID I/O CHANNEL/GROUP NUMBER”	An I/O block is connected to an ECB when the specified point number is invalid or when the specified group or octet number is invalid.
“W53 – INVALID PARAMETER VALUE”	A parameter value is not in the acceptable range.
“W54 – ECB DOES NOT EXIST”	An I/O block has a connection to an ECB that does not exist or has not yet been installed. When the ECB is installed, previously installed I/O blocks waiting for that ECB will initialize automatically.
“W55 – CONTROLLER DOES NOT EXIST”	An FBTUNE or FFTUNE block has an unspecified or unresolved extension connection to a PIDA controller block. When the PIDA is installed, previously installed tuning blocks waiting for that PIDA will initialize automatically.

Table 10-1. ERCODE Parameter Messages (Continued)

Message	Meaning
"W56 – INVALID CONTROLLER MODE"	An FBTUNE or FFTUNE block has an extension connection to a PIDA block whose mode (MODOPT) is not tunable.
"W57 – TUNING_CONSTANT LINKED"	An FBTUNE or FFTUNE block has an extension connection to a PIDA block that has a linked tuning constant.
"W58 – INSTALL ERROR; DELETE/UNDELETE BLOCK"	A Database Installer error has occurred.
"W59 – DUPLICATE OUTPUT CHANNEL"	This block and another output block are connected to the same output point. Since this may be intentional, this message is only a warning.

Programmable Logic and Sequential Control Block Source Files

The Ladder Logic and Sequential control block types contain some fixed sets of parameters, but, in addition, supply the Ladder Logic source code (for Ladder Logic), and/or Sequence Language source code (for Sequence blocks). Either allows you to define the operation of the block.

When you define a block type as being either a Programmable Logic block (PLB) or one of the Sequence types, additional functions highlight in the Block Functions menu, as appropriate. They are:

- Edit Ladder Logic
- Edit Sequence Logic.

Ladder Logic and Sequence Control Logic are described in "Ladder Logic Overview" on page 105 and "Sequential Control Blocks" on page 139, respectively.

11. Control Processor/Fieldbus Application Interface

The communications network between the Foxboro Control Processor and the process instrumentation consists of a hardware interface built around the Fieldbus Module (FBM) and Fieldbus Communications Module (FCM) and a software interface centered about the Equipment Control Block (ECB). This chapter discusses the interfaces – FBMs, FCMs and ECBs, Fieldbus scanning, Fieldbus integration time, FBM types, and ECB types.

The Interfaces – FBMs, FCMs, and ECBs

The Fieldbus and the Fieldbus Modules or Fieldbus Communications Modules (FCMs) provide the hardware interface between the process and the Control Processor. Process instrumentation connects to the FBMs via field wiring. The ZCP270 requires a Fieldbus Communications Module (FCM100Et or FCM100E - non-redundant or redundant) to interface with the FBMs. The FCM100Et should be used when time synchronization of 1 ms is required, Sequence of Events (SOE) or Transient Data Recorder and Analyzer (TDR/TDA) will be used, or redundancy requirements specify dual communication between each FCM and each controller. The FCM100E can be used in all other cases.

— NOTE

The term “FBM” is used throughout this chapter to refer to all 200 Series FBMs and DCS FBMs for migration to APACS+ systems and Westinghouse Process Control WPDF systems.
The term “FCM” is used throughout this chapter to refer to all different FCM types (for example, DCMs, FBIs, FCM100 family, and so forth) unless otherwise noted. For a complete list of FBMs, see Appendix C “FBM – ECB Cross Reference”.

For diagrams and an explanation on how the FCP280 connects to the process via a 2 Mbps High-level Data Link Control (HDLC) protocol Fieldbus that connects to 200 Series FBMs or DCS FBMs, refer to *Field Control Processor 280 (FCP280) User's Guide* (B0700FW).

Figure 11-2 shows that the FCP270 connects to the process via a 2 Mbps High-level Data Link Control (HDLC) protocol Fieldbus that connects to 200 Series FBMs or DCS FBMs. Refer to *Field Control Processor 270 User’s Guide* (B0700AR) for additional information.

Figure 11-3 shows the FCP270 in a dual baud application. When used with properly installed FBI200 modules, an FCP270 can communicate with 100 Series FBMs and 200 Series FBMs simultaneously. The FBI200 modules receive signals from the 2 Mbps module Fieldbus and the 268 Kbps module Fieldbus, but filter out the 2 Mbps communications to ensure the 100 Series FBMs and 100 Series-based migration modules only receive the 268 Kbps signals. FBI200 modules also allow the 268 Kbps module Fieldbus to be extended over a twinaxial Fieldbus cable up to 1830 m (6000 ft). *Field Control Processor 270 User’s Guide* (B0700AR) and *Standard and Compact 200 Series Subsystem User’s Guide* (B0400FA) provide additional information on dual baud applications and Fieldbus extension of the FBI200.

Figure 11-4 shows that the ZCP270 Control Processor connects to FCM100Ets, and Figure 11-5 shows that the ZCP270 Control Processor connects to FCM100Es, via an Ethernet switch and 100 Mbps Ethernet Fieldbus. Each FCM100Et and FCM100E module connects to the process via a 2 Mbps HDLC Fieldbus that connects to 200 Series FBMs or DCS FBMs. For more information, refer to *Z-Module Control Processor 270 User's Guide* (B0700AN).

System Definition limits the number of FBMs to a maximum of:

- 128 of the 200 Series FBMs per FCP280 when the FCP280 is used with 200 Series FBMs exclusively - up to 32 200 Series FBMs on each of the FCP280's four Expanded fieldbuses
- 64 of the 100 Series FBMs per FCP280. (The FCP280 supports up to 128 100 Series and 200 Series modules total. For an FCP280 supporting 100 Series FBMs, the remainder of the 128 module limit may be 200 Series FBMs.)
- 32 of the 200 Series FBMs per FCP270
- 32 FCM100Ets or FCM100Es per ZCP270
- 32 of the 200 Series FBMs per FCM100Et (128 of the 200 Series FBMs per ZCP270 when using an FCM100Et)
- 32 of the 200 Series FBMs, 100 Series FBMs, or a combination of 100 and 200 Series FBMs per FCM100E (128 total FBMs per ZCP270 when using an FCM100E)

The appropriate number of ECBs can be determined using the CP Sizing Spreadsheet and/or Station Block. FBMs differ depending on the electrical nature and distribution of inputs and outputs (FBM varieties are listed in "FBM Types" on page 182).

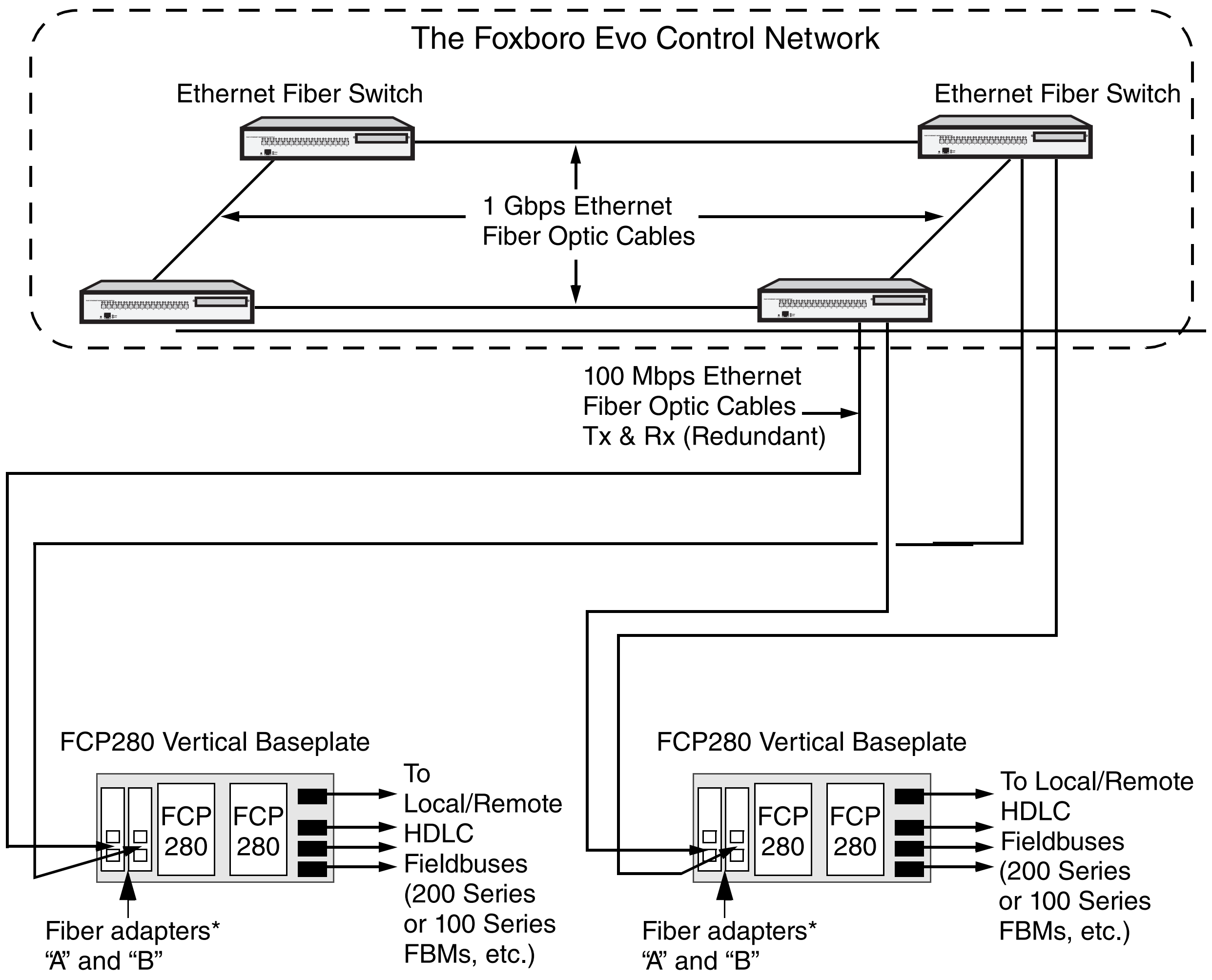
For sizing constraints that may affect the limits listed above, refer to:

- *Field Control Processor 280 (FCP280) Sizing Guidelines and Excel Workbook* (B0700FY)
- *Field Control Processor 270 (FCP270) Sizing Guidelines and Excel Workbook* (B0700AV)
- *Z-Module Control Processor 270 (ZCP270) Sizing Guidelines and Excel Workbook* (B0700AW).

The Equipment Control Block (ECB) provides the software interface between the FBM or FCM and the I/O blocks.

ECBs differ depending on the FBM or FCM and its application [the varieties of ECBs are listed in "Equipment Control Blocks (ECBs)" on page 182].

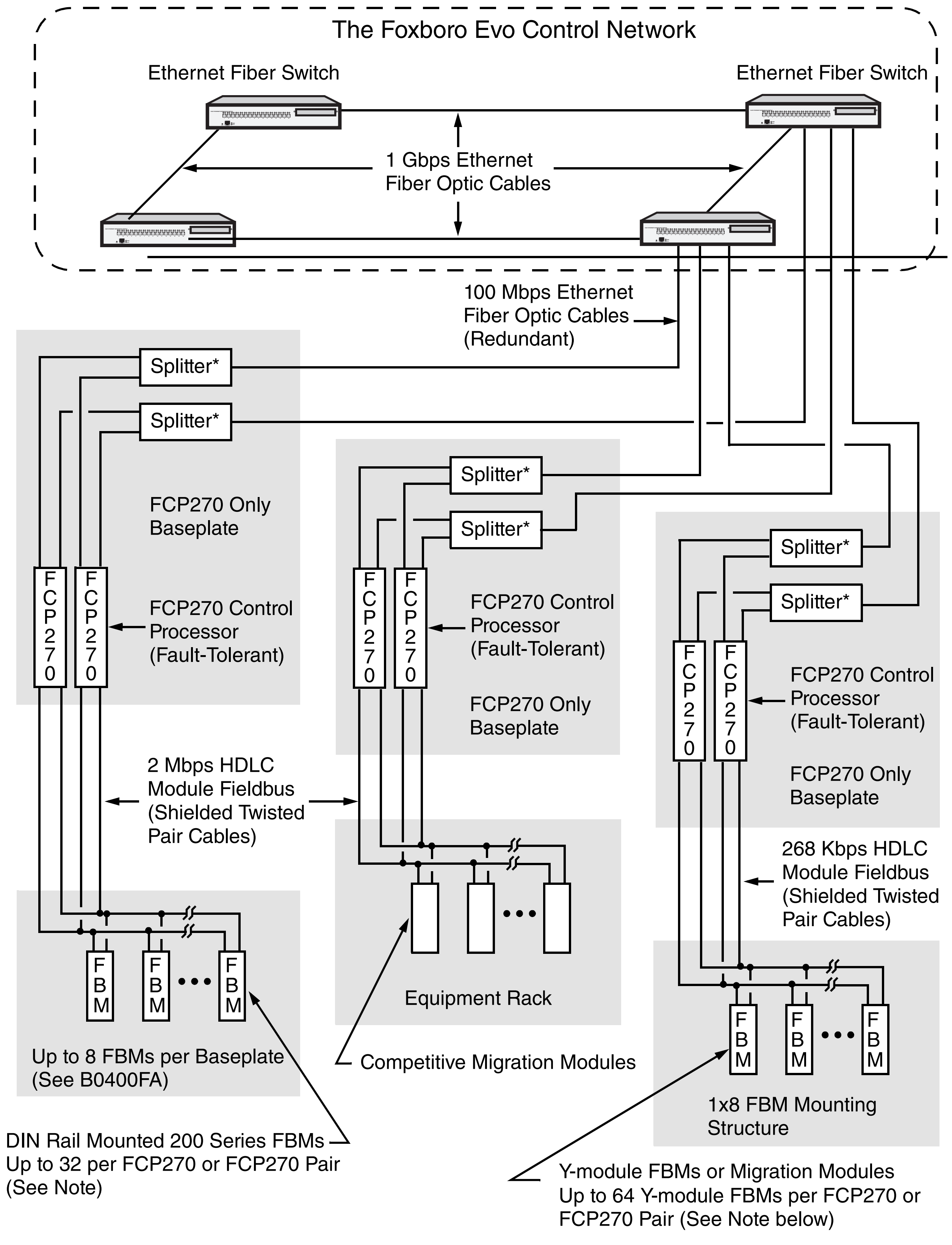
When you configure an FBM or FCM and its software application type, the System Configurator/Definition automatically adds the appropriate ECB type.


The Foxboro Evo Control Network
Ethernet Fiber Switch
Ethernet Fiber Switch
1 Gbps Ethernet Fiber Optic Cables
100 Mbps Ethernet Fiber Optic Cables Tx & Rx (Redundant)
FCP280 Vertical Baseplate
FCP 280
FCP 280
To Local/Remote HDLC Fieldbuses (200 Series or 100 Series FBMs, etc.)
Fiber adapters* "A" and "B"
FCP280 Vertical Baseplate
FCP 280
FCP 280
To Local/Remote HDLC Fieldbuses (200 Series or 100 Series FBMs, etc.)
Fiber adapters* "A" and "B"

Note: For additional sizing information, refer to *Field Control Processor 280 (FCP280) Sizing Guidelines and Excel Workbook* (B0700FY).
For cabling connections to FBMs, refer to *Field Control Processor 280 (FCP280) User's Guide* (B0700FW).

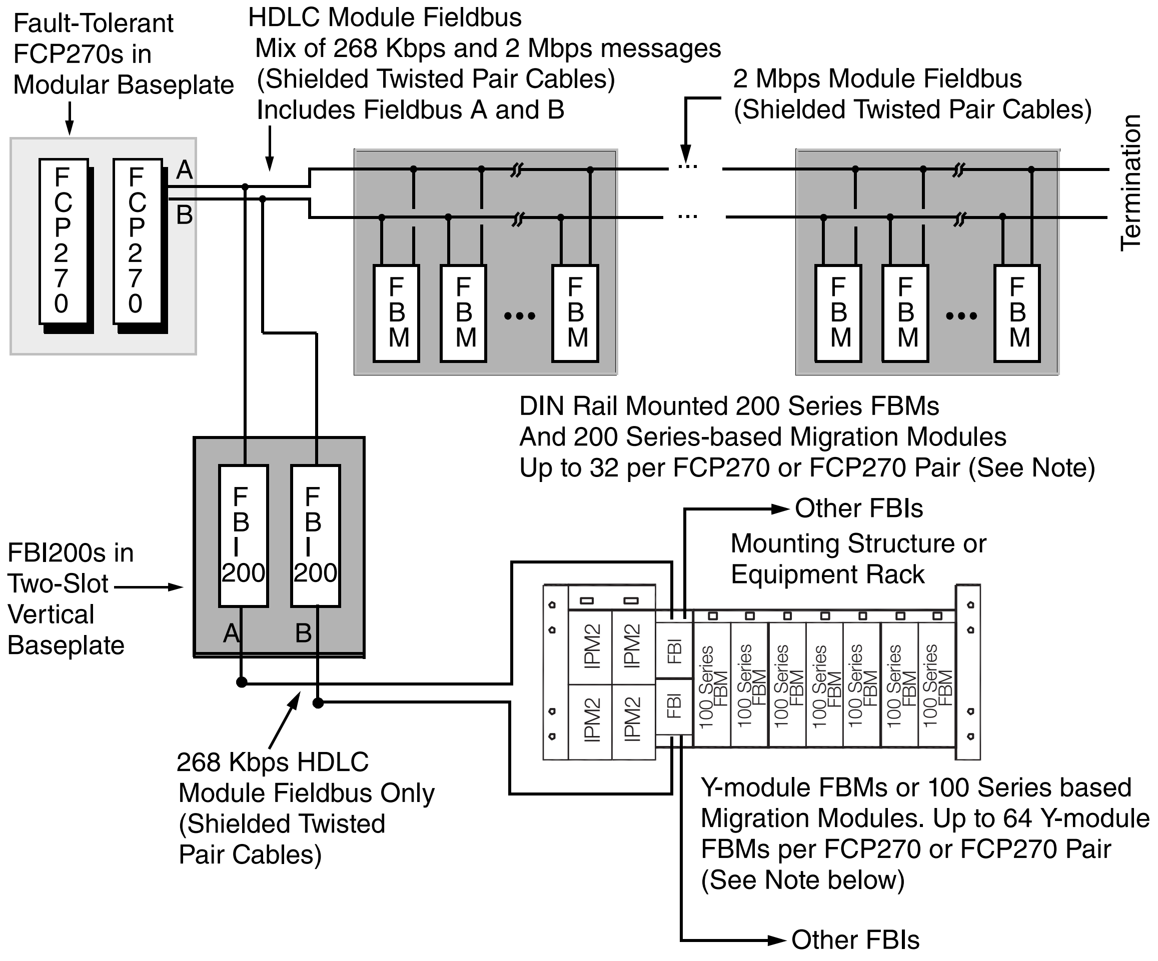
* Copper adapters are also available for copper twinaxial cabling to The Foxboro Evo Control Network, up to 100 m (328 ft).

Figure 11-1. Multiple FCP280s in The Foxboro Evo Control Network (Simplified)


The Foxboro Evo Control Network
Ethernet Fiber Switch
Ethernet Fiber Switch
1 Gbps Ethernet Fiber Optic Cables
100 Mbps Ethernet Fiber Optic Cables (Redundant)
Splitter*
Splitter*
Splitter*
Splitter*
Splitter*
Splitter*
FCP270 Only Baseplate
FCP270
FCP270
FCP270 Control Processor (Fault-Tolerant)
FCP270 Control Processor (Fault-Tolerant)
FCP270 Only Baseplate
FCP270 Control Processor (Fault-Tolerant)
FCP270 Only Baseplate
2 Mbps HDLC Module Fieldbus (Shielded Twisted Pair Cables)
268 Kbps HDLC Module Fieldbus (Shielded Twisted Pair Cables)
FBM
FBM
FBM
Up to 8 FBMs per Baseplate (See B0400FA)
Equipment Rack
Competitive Migration Modules
1x8 FBM Mounting Structure
DIN Rail Mounted 200 Series FBMs Up to 32 per FCP270 or FCP270 Pair (See Note)
Y-module FBMs or Migration Modules Up to 64 Y-module FBMs per FCP270 or FCP270 Pair (See Note below)

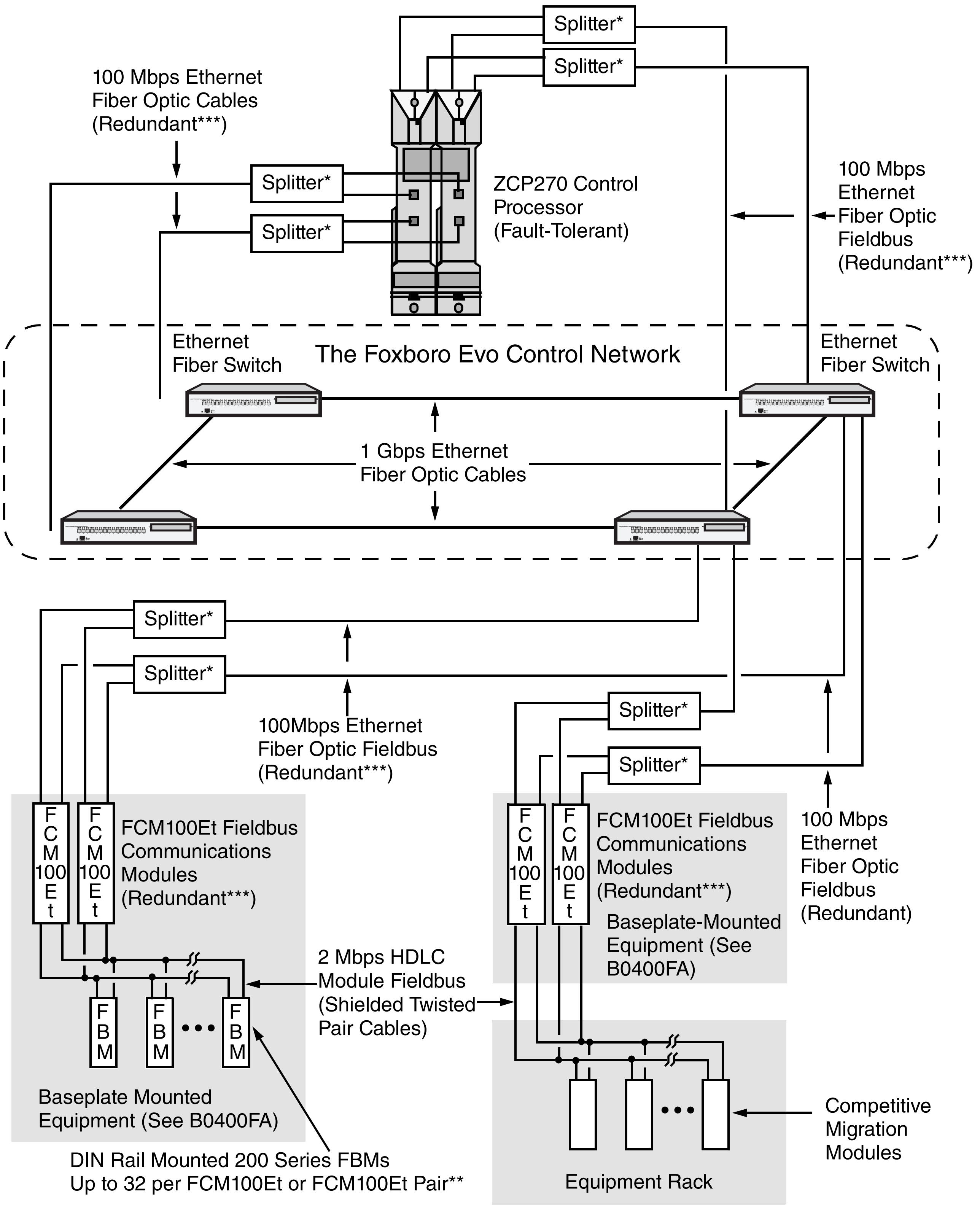
* Splitter/Combiner - not required for FCP280, which uses network adapters instead.
Note: For additional sizing information, refer to *Field Control Processor 270 (FCP270) Sizing Guidelines* (B0700AV).

Figure 11-2. Typical FCP270 Network Configuration (Conceptual, Without Dual Baud)


Fault-Tolerant FCP270s in Modular Baseplate
HDLC Module Fieldbus Mix of 268 Kbps and 2 Mbps messages (Shielded Twisted Pair Cables) Includes Fieldbus A and B
2 Mbps Module Fieldbus (Shielded Twisted Pair Cables)
Termination
FCP270
FCP270
A
B
FBM
FBM
FBM
DIN Rail Mounted 200 Series FBMs And 200 Series-based Migration Modules Up to 32 per FCP270 or FCP270 Pair (See Note)
Other FBIs
Mounting Structure or Equipment Rack
FBI200s in Two-Slot Vertical Baseplate
FBI 200
FBI 200
IPM2
FBI
100 Series FBM
268 Kbps HDLC Module Fieldbus Only (Shielded Twisted Pair Cables)
Y-module FBMs or 100 Series based Migration Modules. Up to 64 Y-module FBMs per FCP270 or FCP270 Pair (See Note below)
Other FBIs

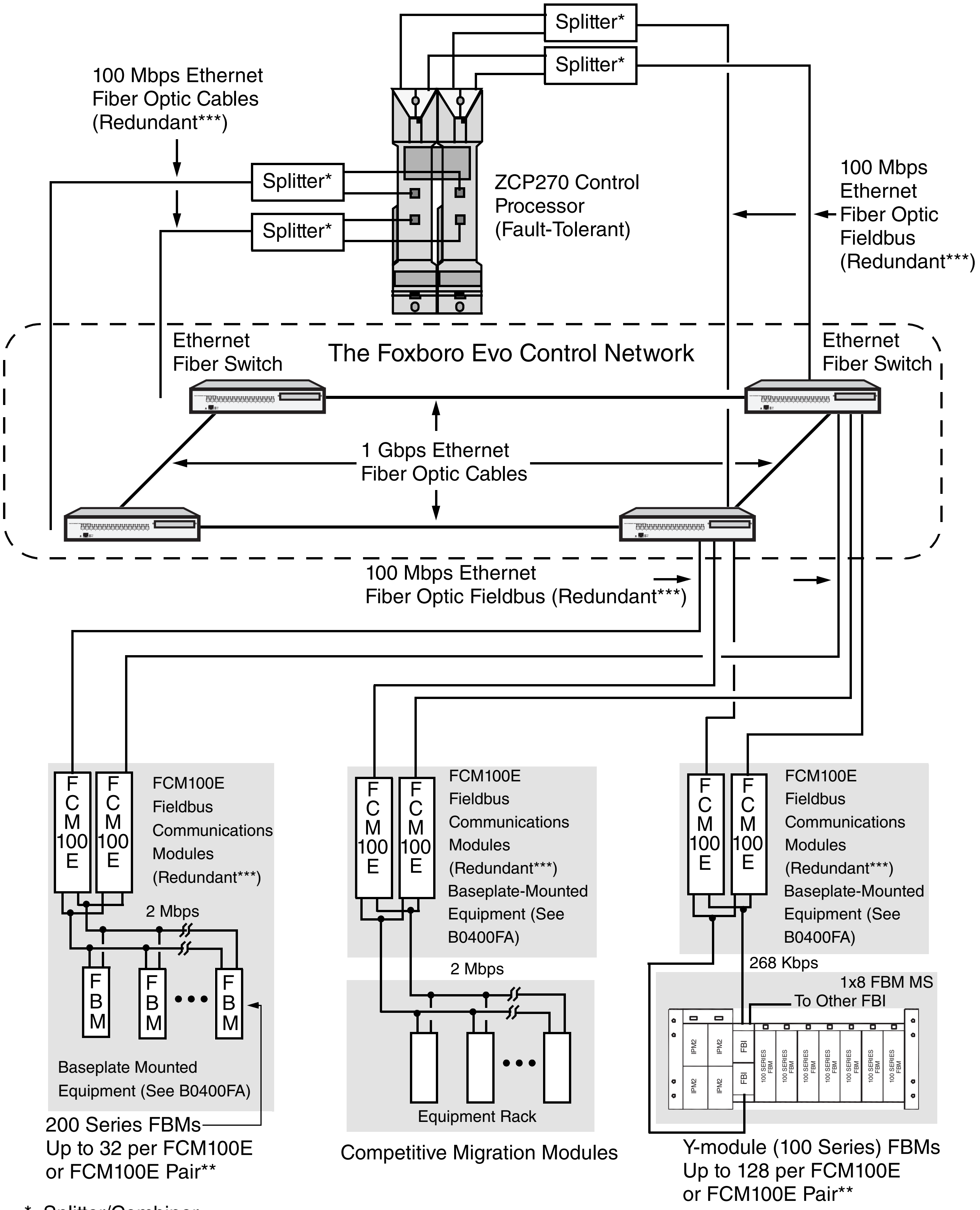
Notes:
1. For additional sizing information, refer to *Field Control Processor 270 (FCP270) Sizing Guidelines* (B0700AV).
2. For FBI200 installation instructions and other FBI200 configurations, refer to *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA) and *Field Control Processor 270 (FCP270) User's Guide* (B0700AR)

Figure 11-3. Example Network with FCP270 with Dual Baud Functionality (Conceptual, With Dual Baud)


Splitter*
Splitter*
100 Mbps Ethernet Fiber Optic Cables (Redundant***)
Splitter*
Splitter*
ZCP270 Control Processor (Fault-Tolerant)
100 Mbps Ethernet Fiber Optic Fieldbus (Redundant***)
Ethernet Fiber Switch
The Foxboro Evo Control Network
Ethernet Fiber Switch
1 Gbps Ethernet Fiber Optic Cables
Splitter*
Splitter*
100Mbps Ethernet Fiber Optic Fieldbus (Redundant***)
Splitter*
Splitter*
FCM100Et
FCM100Et
FCM100Et Fieldbus Communications Modules (Redundant***)
FCM100Et
FCM100Et
FCM100Et Fieldbus Communications Modules (Redundant***)
Baseplate-Mounted Equipment (See B0400FA)
100 Mbps Ethernet Fiber Optic Fieldbus (Redundant)
2 Mbps HDLC Module Fieldbus (Shielded Twisted Pair Cables)
FBM
FBM
FBM
Baseplate Mounted Equipment (See B0400FA)
DIN Rail Mounted 200 Series FBMs Up to 32 per FCM100Et or FCM100Et Pair**
Equipment Rack
Competitive Migration Modules

* Splitter/Combiner.
** For additional sizing information, refer to *Z-Module Control Processor 270 (ZCP270) Sizing Guidelines* (B0700AW).
*** ZCP270 supports either a non-redundant or redundant Fieldbus. Only one FCM100Et required in non-redundant configuration.

Figure 11-4. Typical ZCP270 with FCM100Ets Network Configuration (Simplified)


Splitter*
Splitter*
100 Mbps Ethernet Fiber Optic Cables (Redundant***)
Splitter*
Splitter*
ZCP270 Control Processor (Fault-Tolerant)
100 Mbps Ethernet Fiber Optic Fieldbus (Redundant***)
Ethernet Fiber Switch
The Foxboro Evo Control Network
Ethernet Fiber Switch
1 Gbps Ethernet Fiber Optic Cables
100 Mbps Ethernet Fiber Optic Fieldbus (Redundant***)
FCM100E
FCM100E Fieldbus Communications Modules (Redundant***)
2 Mbps
FBM
Baseplate Mounted Equipment (See B0400FA)
200 Series FBMs Up to 32 per FCM100E or FCM100E Pair**
FCM100E Fieldbus Communications Modules (Redundant***) Baseplate-Mounted Equipment (See B0400FA)
2 Mbps
Equipment Rack
Competitive Migration Modules
FCM100E Fieldbus Communications Modules (Redundant***) Baseplate-Mounted Equipment (See B0400FA)
268 Kbps
1x8 FBM MS
To Other FBI
IPM2
FBI
100 SERIES FBM
Y-module (100 Series) FBMs Up to 128 per FCM100E or FCM100E Pair**

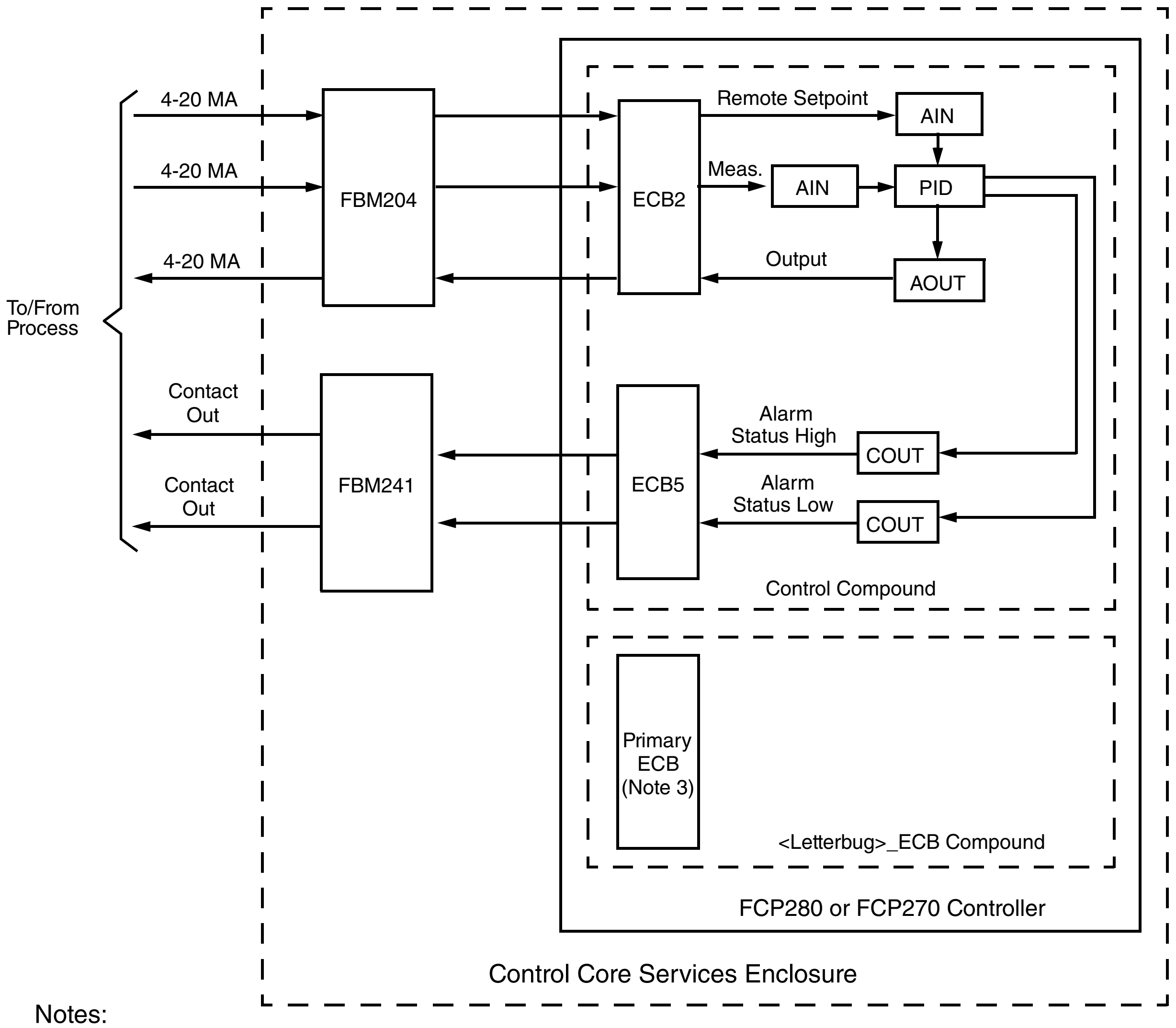
* Splitter/Combiner.

** For additional sizing information, refer to *Z-Module Control Processor 270 (ZCP270) Sizing Guidelines* (B0700AW).

*** ZCP270 supports either a non-redundant or redundant Fieldbus. Only one FCM100E required in non-redundant configuration.

Figure 11-5. Typical ZCP270 with FCM100E Network Configuration (Simplified)

Figure 11-6 shows a typical control scheme using an FCP280 or FCP270 and 200 Series FBMs. The ECB is the interface between the I/O blocks of a compound and the FBM data.



Notes:
1. For ECB parameters definitions and ECB to FBM assignments, refer to *Integrated Control Block Descriptions* (B0193AX).
2. The Primary ECB(s) are automatically assigned to a compound named <Letterbug>_ECB, for example, if an FCP letterbug is H51FCP, the compound name is H51FCP_ECB.
3. The FCP280 has four Primary ECBs - one for each of the HDLC fieldbuses (PIO channels) that it supports.

Figure 11-6. Typical Control Scheme Using an FCP280 or FCP270 and 200 Series FBMs

When compound processing starts, each FBM or equivalent reads the I/O data (both input and output channels) from the process instrumentation on a per-FBM basis. Any ECBs should be configured for a scan period equal to the fastest scan period specified for any Input-type I/O block configured for that FBM. The FBM then conditions, digitizes, and normalizes the data, as necessary, and stores the data and the status into its ECB.

As a control compound is processed, the input blocks retrieve the data from the ECB. Using this new data, the Compound Processor generates new outputs which are forwarded to the appropriate ECB. The ECB transmits the data to its FBM which converts the output value to a signal (such as 4 to 20 mA or pulses) compatible with the process instrumentation.

FCP280 and Equipment Control Blocks

FCP280 control processors support four HDLC fieldbuses (PIO channels); this is also known as the Expanded fieldbus. The FCP280's baseplate has four Fieldbus ports, one for each HDLC fieldbus. This is discussed in detail in *Field Control Processor 280 (FCP280) User's Guide* (B0700FW).

Unlike previous control processors, the FCP280 has four Primary ECBs - one for each HDLC fieldbus (PIO channel). Each one is assigned a number, equivalent to the number of the Fieldbus port for which they are responsible. These are named PRIMARY_ECB, PRIMARY_ECB2, PRIMARY_ECB3, and PRIMARY_ECB4; PRIMARY_ECB2 is associated with the HDLC fieldbus off of Fieldbus port 2, etc.

For an FCP280, each of its FBM's ECBs must be assigned to the Primary ECB for the HDLC fieldbus on which their FBM resides.

For Control Core Services v9.0 or later, the Primary ECB contains the BAUD2M parameter, which is only used with the FCP280. This parameter defines the baud rate at which the HDLC fieldbus associated with the Primary ECB will operate:

- **1** = 2 Mbps (default) - for 200 Series FBMs and similar modules
- **0** = 268 Kbps - for 100 Series FBMs and competitive migration modules

Be sure that each FBM type is assigned to the appropriate HDLC fieldbus - 100 Series FBMs should not be assigned to a 2 Mbps HDLC fieldbus, etc.

As well, for Control Core Services v9.0 or later, FBM ECBs (not device ECBs) contain the CHAN parameter, a one-time only configurable parameter which has a value of 1-4 (1 is default), representing the PIO channel number (i.e. the number of the Primary ECB) to which the FBM ECB is assigned. CHAN must be set for its ECB when the ECB is created, and it cannot be edited once set.

You must set these parameters with your appropriate control configurator (Control Editors, IACC, or ICC) in order to maintain proper communications between the FCP280 and their FBMs. For control configuration details, refer to:

- Foxboro Evo Control Editors - see *Block Configurator User's Guide* (B0750AH) and *Hardware Configuration User's Guide* (B0700BB).
- IACC - *I/A Series Configuration Component (IACC) User's Guide* (B0700FE)
- ICC - *Integrated Control Configurator* (B0193AV).

The BAUD2M and CHAN parameters are not used with the FCP270/ZCP270 or earlier control processors, or I/A Series software v8.8 or earlier.

I/O Fail-safe Strategy

Non-Ladder Logic FBMs use one of two strategies for obtaining output values:

- Normal strategy, in which the FBM uses the outputs received from the CP, as determined by the application software.
- Fail-safe strategy, in which the FBM uses outputs from its gate array registers. The gate array registers contain:
  - The current value for each of the FBM's outputs
  - A fallback value for each of the FBM's outputs

- ♦ A hold/fallback bit for each output, to specify which of the two values each output will take.

Fail-safe is an FBM hardware logic state, designed to provide I/O security. This section describes the events that place the FBM in the Fail-safe state, and the actions of the FBM while in the Fail-safe state.

FBM fail-safe strategies are implemented by both software and hardware. The state of fail-safe is indicated by the LEDs (Red/Green) on each FBM.

Whenever the Red LED is on, the I/O hardware has asserted Fail-safe. If the software is asserting fail-safe, the Green LED is blinking on DIN FBMs.

Cold Start

(Red LED)

When the FBM is first powered up, the FBM firmware writes zeroes to the gate array registers and places the FBM in its Fail-safe state. Since all the hold/fallback bits equal 0 (Fallback) and each fallback value equals 0, each output is at its de-energized value of 0, waiting for the FBM to go on-line.

Placing the FBM On-Line

(Red/Green LED)

When the application software takes control it:

- ♦ Reads all the gate array registers
- ♦ Computes the current value of each I/O point
- ♦ Writes the current value into the Current Value register.

FBM Exits the Fail-safe State

(Solid Green LED)

When the FBM receives the write message, it exits the Fail-safe state and:

- ♦ turns the Red LED to solid Green if it is on or
- ♦ turns the blinking Green LED to solid green.

In this state, the control strategy is sending its outputs to the FBM.

At this point, the application software maintains control until some event disrupts that control and forces the FBM to Fail-safe.

The Fail-safe Configuration

The controller, as part of I/O hardware initialization, writes the ECB fail-safe configuration from the ECB to the FBM gate array registers. For example, the controller implements the Fail-safe configuration by writing the following type 02 ECB parameters to a type 204 FBM:

FSMM02	Fail-safe Main Mask, ECB type 02 – provides the four bits to the hold/fallback register that determine how the four output channels (05, 06, 07, and 08) respond during Fail-safe. If the bit = 1, the output goes to hold and takes its value from the current value register. If the bit = 0, the output goes to fallback and assumes the value from the fallback register. FSMM02 is digitally coded as follows:

Table 11-1. FSMM02 Coding

Bit number	7	6	5	4	3	2	1	0 (LSB)
Channel number	08	07	06	05	xx	xx	xx	xx

If, for example, channel 05 is to assume the fallback value in FS5D02 while channels 06, 07, and 08 hold their current value, then FSMM02 must = decimal 224, or hexadecimal E0 (bits 7, 6, and 5 = true).

FS5D02

Fail-safe, channel 5 Data, ECB type 2 – the fallback value for the channel 5 output is written to the fallback value register.

The value represents the D/A raw count and must be related to the intended signal conditioning. For example, if this output is going to a 4-20 mA device (raw count range: 12,800 to 64,000) then FS5D02 must be within that range.

FS6D02, FS7D02, and FS8D02, the fallback values for channels 6, 7, and 8 are likewise written to the fallback value register.

You should be especially careful when configuring the fail-safe mask and fail-safe values in ECBs associated with redundant analog output FBMs. In addition to installing a special Termination Cable Assembly (TCA) or Redundant Adapter, you must configure the ECB of each side so that its FBM “fails low”. This is a constraint imposed by the hardware arrangement of the redundant-type TCA or Redundant Adapter. Unless a failed FBM outputs 0 mA, the good side does not take over and drive the output correctly. This implies that the fail-safe mask of each ECB must specify Fallback for each channel, rather than Hold. The Fallback value must be 0 raw counts for each channel, even if the normal low end of the output span is 12,800 raw counts, corresponding to 4 mA. Table 11-2 summarizes the required ECB parameter values:

Table 11-2. Redundant Output Fallback Values

FBM	ECB	Mask	Fallback Value
205 and 05	2	FSMM02 = 0	FS5D02 = 0, ... , FS8D02 = 0
208	2	FSMM02 = 0	FS5D02 = 0, ... , FS8D02 = 0
215 and 237	53	FSMM53 = 0	FS1D53 = 0, ... , FS8D53 = 0

Events Leading to Fail-safe

Some of the events that cause the FBM to transition from control to the fail-safe state include:

- Software failures (SOFT):
  - FBM download from the System Manager or legacy System Management Display Handler (SMDH)
  - EEPROM update from System Manager or SMDH
  - FBM taken off-line from System Manager or SMDH
  - FBM turned on-line from System Manager or SMDH.
- Hardware failures (HARD):
  - Fieldbus Communications failure

- ♦ Power failure

The FBM response to each of these events is described below.

FBM Download from System Manager or SMDH

(Previously Green LED)

The FBM goes to the hardware Fail-safe state (Red LED) and the FBM outputs assume the Fail-safe configuration until the FBM is back on-line and the application software resumes control (Green LED).

Fieldbus Communications Failure

Fieldbus Communications Failure or controller Failure:

Two ECB parameters determine the if and when of a communications failure between the controller and the FBM configured in that ECB.

FSENAB	Fail-safe Enable is a boolean parameter that, when true, enables the FSDLAY timer. If communications fail with FSENAB false, there is no fail-safe activity: the FBM continues to drive the outputs using the current values, which is equivalent to holding the outputs. The Green LED remains on.

NOTE

When FSENAB transitions to “false”, be aware that there is no immediate fail-safe behavior initiated to recover from any hardware faults which may have occurred.

FSDLAY	Fail-safe Delay is a timer that, when enabled by FSENAB, specifies (in units of 0.01 seconds) the length of time the FBM waits for a communication from the CP before entering a Commfail or software Fail-safe state (in which outputs are under hardware control). This state asserts the output values specified by the fail-safe (solid Red LED) configuration. Example: The default value of 1000 causes the FBM to wait 10 seconds between controller communications before going to the Fail-safe state, if FSENAB is set true.

NOTE

When setting the Failsafe Delay value (FSDLAY) on a ZCP270 with associated FCMs connected to the control network, it is important to note that a CP hot-remarry or a network re-span can each cause a loss of communication for up to one second.

The ECB parameter FSOPTN is the Fail-Safe Option and is a configurable option that specifies the fail-safe conditions and action to be taken in an FBM for an output point in a ROUT block.

EEPROM Update from System Manager or SMDH

(Alternately flashing Red and Green LEDs)

The FBM goes to the Fail-safe state and the FBM outputs assume the Fail-safe configuration defined by the gate array registers prior to the EEPROM update. The FBM maintains this definition until the EEPROM update is completed and the application software resumes control (Green LED).

FBM Turned Off-Line From System Manager or SMDH

If the FBM is taken off-line from SMDH, the controller sends a message to the FBM as long as the CP-to-FBM communications are good, shutting down the FBM. The FBM goes to the software Fail-safe state (Green LED) and the outputs assume the fail-safe configuration values.

If the controller-to-FBM communications are bad, the FBM responds in one of two ways:

1. FSENAB is configured true. The FBM goes to software fail-safe state (solid Green LED for FBMs, blinking Green LED for DIN FBMs) after the delay timer times out, and the outputs assume the fail-safe configuration values.
2. FSENAB is configured false. The FBM, unaware of any communications breakdown, continues to drive the outputs using the current values – essentially a Hold state (solid Green LED for FBMs, blinking Green LED for DIN FBMs).

Power Failure

FBM01 to FBM46 and Equivalents

When the main power fails, 100 Series FBM outputs are de-energized (zero state) since it is the main power that drives them. If the FBM has battery-backup, where the battery maintains the hardware fail-safe configuration in the gate array registers, the LEDs are dark.

If main power returns while the FBM is still battery-backed, the FBM continues the fail-safe configuration, now via the software. But if the main power returns after the backup battery discharges, the fail-safe configuration is gone, and the FBM reverts to a Cold Start state.

FBM201 to FBM246 (DIN Rail Mounted FBMs and Equivalents)

When the main and secondary (if used) power fails, 200 Series FBM outputs are de-energized (zero state) since it is the main and secondary (if used) power that drives them. If main or secondary (if used) power returns, the FBM reverts to a Cold Start state.

FBM Turned On-line from System Manager or SMDH

(Red/Green to Green LED)

The manipulations associated with this state change do not bump the outputs. The outputs have the values written to the hardware registers, by the application software, prior to transition to restart. If the software was never run, the registers have the zero values received on power-up initialization, and the outputs are de-energized.

Fieldbus Scanning

Controller scan of a Fieldbus Module is performed on a per-FBM basis. For FBM01 to FBM46, the Control Processor treats the Main digital type FBM and its Expansion Module as one FBM when scanning the Fieldbus.

The ECB parameter, PERIOD, determines the period at which an FBM is scanned.

Fieldbus scanning of FBM data completes before block processing begins.

The Analog-type FBMs read in their Analog inputs every 25 or 50 ms, for purposes of integration, independent of CP/Fieldbus scan intervals.

Fieldbus Integration Time

The FBM converts each analog input to a digital value within a time interval called the update time, presently set to 25 ms.

The converted value at each update interval is summed and averaged over a specified time period called the integration time or conversion time. This continuous integration technique provides a filter time equal to the integration period.

A new value is available to the controller each update period. The value represents the average of all the updated values over the last integration period.

The integration time should be twice the loop scan period for best resolution and most effective anti-aliasing and filtering.

FBM201 to FBM205, FBM208, and FBM211 to FBM213

Analog input FBMs use a fixed resolution of 15 bits for each analog input. The Resolution (RES) parameter for each ECB allows you to select a RES value of 1 to 5 which selects the conversion time of the input. Conversion times of 50 ms, 100 ms, 200 ms, 500 ms and 1000 ms can be specified for analog inputs. Specifying a longer conversion time gives more filtering by providing a longer filter time constant.

Table 11-3 illustrates these time factors:

Table 11-3. FBM201 to FBM205 and FBM208 Integration Time Factors

Resolution	Update Time	Conversion Time	RES Value
15 bits	25 ms	50 ms	5
15 bits	25 ms	100 ms	1
15 bits	25 ms	200 ms	2
15 bits	25 ms	500 ms	3
15 bits	25 ms	1000 ms	4

For example, for a 100 ms interval, a new analog value is converted every 25 ms. Each 25 ms sample is continuously averaged with the previous 25 ms samples over a 100 ms interval.

The higher 500 ms and 1000 ms conversion times are normally required for the fixed span thermocouple and RTD applications to meet the wider temperature spans (FBM202 and FBM212 - Thermocouple Input, and FBM203 and FBM213 - RTD Input). Lower conversion times (50 ms, 100 ms, or 200 ms) may result in unreliable measurement values.

FBM01 to FBM05

Resolutions of 11, 12, 13, 14, and 15 bits can be specified for analog inputs. The greater the resolution, the longer the conversion time. The Resolution (RES) parameter for each ECB allows you to select a RES value of 1 to 5 which selects the resolution and integration time of the input. Specifying 15-bit resolution gives more filtering by providing a longer filter time constant.

Table 11-4 illustrates these time factors:

Table 11-4. FBM01 to FBM05 Integration Time Factors

Resolution	Update Time	Integration Time	RES Value
11 bits	25 ms	50 ms	5
12 bits	25 ms	100 ms	1
13[1]	25 ms	200 ms	2
14	25 ms	500 ms	3
15	25 ms	1000 ms	4

1. Default value for typical applications. 14-bit resolution is default for thermocouple and RTD applications (FBM02 and FBM03).

For example, for a 12-bit specified resolution, a new analog value is converted to 12 bits every 25 ms. Each 25 ms sample is continuously averaged with previous 25 ms samples over a 100 ms interval.

The higher 14-bit and 15-bit resolutions are normally required for the fixed span thermocouple and RTD applications to meet the wider temperature spans (14-bit resolution is the default value for FBM02 – Thermocouple Input and FBM03 – RTD Input). Lower resolutions (12 or 13 bits) may result in unreliable measurement values.

Cluster I/O

The three Cluster I/O analog input modules (FBC01, FBC17, and FBC21) always provide 12-bit resolution. No user options exist for Cluster I/O.

FBM Rate of Change Monitoring

FBMs containing analog input points perform rate-of-change checking on each such point by using the configured ECB parameters ROCx, where x = 1, 2, ... The value of ROCx is the maximum rate of change for that channel, expressed in units of normalized raw counts per 100 ms. If the input changes by more than that amount in 100 ms, the channel status is marked Bad.

The default value of zero for any ROC parameter inactivates rate-of-change checking for that channel. (For ECB34 and 36, the parameters are labelled ROC1MD through ROC4MD. Cluster I/O analog inputs do not have rate-of-change checking.)

FBM Types

Digital-type FBMs can execute four different applications or software types selected by configuring the software type of ECB during System Configuration/Definition or Integrated Control Configuration. These are:

- ♦ Digital Input/Digital Output
- ♦ Ladder Logic (LL)
- ♦ Sequence of Events (SOE)
- ♦ Pulse Counters (PC).

You must select how a digital-type FBM is to function. It can operate in the Ladder Logic mode, or in the Sequence of Events mode, or in the Pulse Counter mode, or in the Digital Input/Digital Output mode, but it cannot combine modes. Cluster I/O FBCs of the digital type (FBC07, FBC09, and FBC10) operate only in the Digital Input/Digital Output mode.

Digital inputs can always be read by the controller as independent inputs, even though LL, SOE, or PCs are executing in the FBM.

More than one type of I/O block can be connected to an FBM.

An FBM is identified by a user-defined, physical device ID label. This device ID label, referred to as the "letterbug", is used to relate the ECB to a specific FBM.

For a complete list of FBMs, see Appendix C "FBM – ECB Cross Reference".

Equipment Control Blocks (ECBs)

When you configure an FBM, the System Configurator/Definition creates a file with the appropriate ECB (software) types, which is used by the Integrated Control Configurator (ICC) and I/A Series Configuration Component (IACC).

For the ICC, use the FBM Fix All function or the Insert Block/ECB function to add the ECB to the control database and to specify or modify the parameter values for this added ECB.

Table 11-5. Equipment Control Blocks

ECB Number	Description
ECB01	Analog Input
ECB02	Analog Input and Analog Output
ECB04	Pulse In and Analog Output
ECB05	Digital In, Sustained/Momentary, and Digital Out
ECB06	Sequence of Events Input
ECB07	Digital In and Pulse Count Input
ECB08	Ladder Logic
ECB09	Analog I/O, Digital I/O
ECB11	Reserved for Primary FBM
ECB18	Intelligent Transmitter 2 Interface (Child ECB)
ECB200	Parent ECB for Non-redundant DCI FBMs
ECB201	Child ECB for Non-redundant DCI FBMs

Table 11-5. Equipment Control Blocks (Continued)

ECB Number	Description
ECB202	Parent ECB for Redundant DCI FBMs
ECB210	Fieldbus Communications Module FCM100Et and FCM100E
ECB23	Intelligent Field Device and Analog Output
ECB34	MDACT Feedback Lag, Tri-State
ECB36	MDACT Pulse Width, Tri-State
ECB38R	Intelligent Field Device In and Analog Out, Dual Baud Rate, Redundant
ECB41	Analog Input, Cluster I/O
ECB42	Digital Input, Cluster I/O
ECB43	Analog Output, Cluster I/O
ECB44	Digital Output, Cluster I/O
ECB46	Digital I/O, Cluster I/O
ECB47	FBP10 Cluster I/O Interface
ECB47R	FBP10R ECKARDT Migration
ECB48R	Redundant SPECTRUM™ UCM
ECB52	DPIDA Interface
ECB53	Analog Output
ECB73	Parent ECB for FoxCom™ FBMs
ECB74	Child ECB for Intelligent Positioners
ECB110	FCM10
ECB210	FCM100

The ECB types listed in Table 11-5 include the ECB Primary (ECBP). An ECBP is associated with each Control Station and specifies Fieldbus communication parameters (for example, bus switching and watch dog timer).

You can establish the ECB type during System Configuration/Definition when you select the FBM Hardware/software type, or specify it when executing the ICC Insert Block/ECB function to add the ECB. When adding an ECB in the ICC, you must specify the ECB type first, then define its hardware and software type.

For a list the ECB Assignments by System Configurator/Definition, see Appendix C “FBM – ECB Cross Reference”.

Diagrams – ECB Interfaces to I/O Blocks and FBMs

ECB Interfaces to I/O Blocks and 200 Series FBMs

This section uses a series of interface diagrams to show the relationship of the ECBs to the I/O blocks at the Control interface, and to the 200 Series FBMs at the Process interface.

Each diagram shows all the I/O blocks and all the FBMs that connect to the ECB in the center of the diagram.

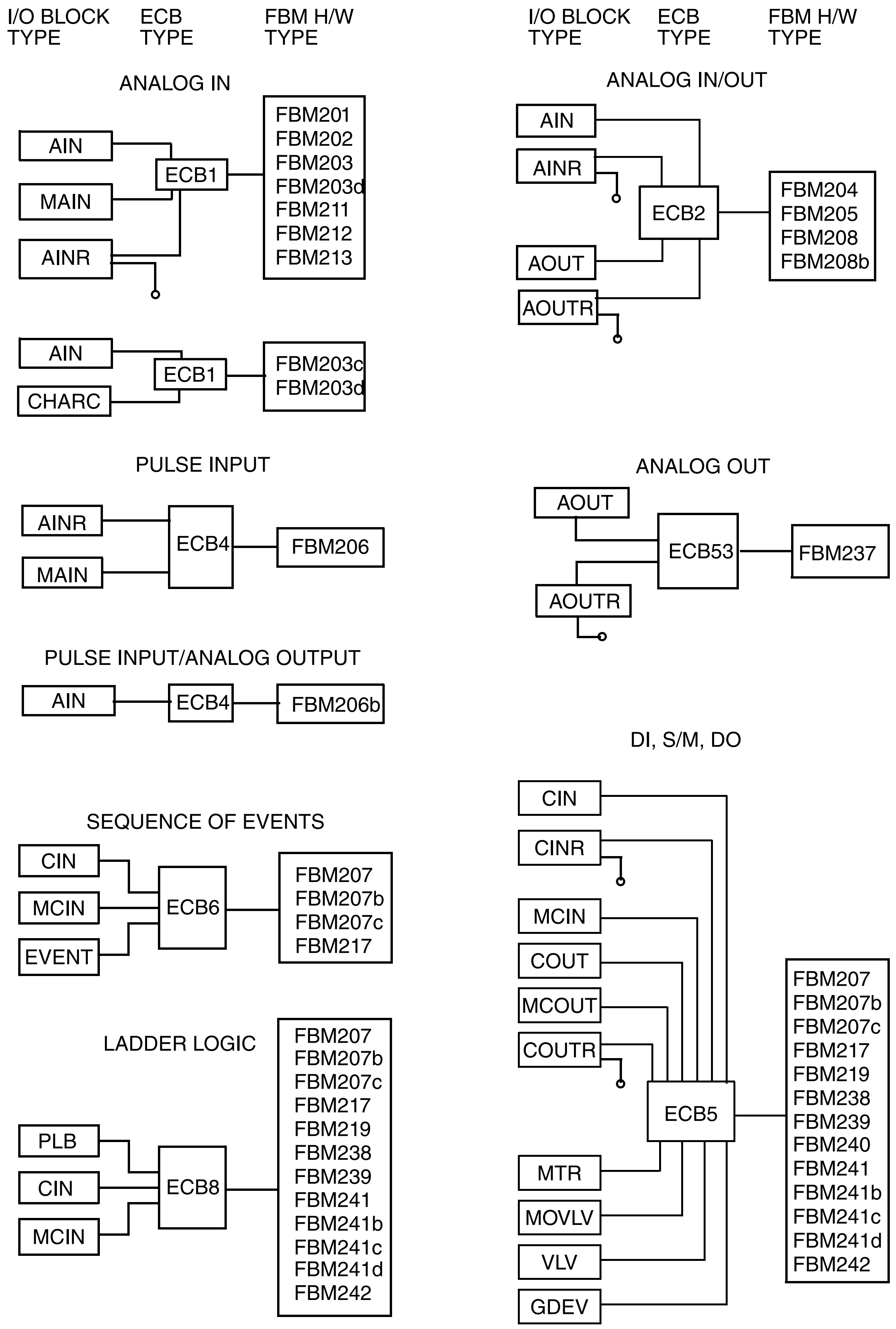


Figure 11-7. DIN Rail Mounted FBM Connections to ECBs 1, 2, 4, 5, 6, 8, and 53

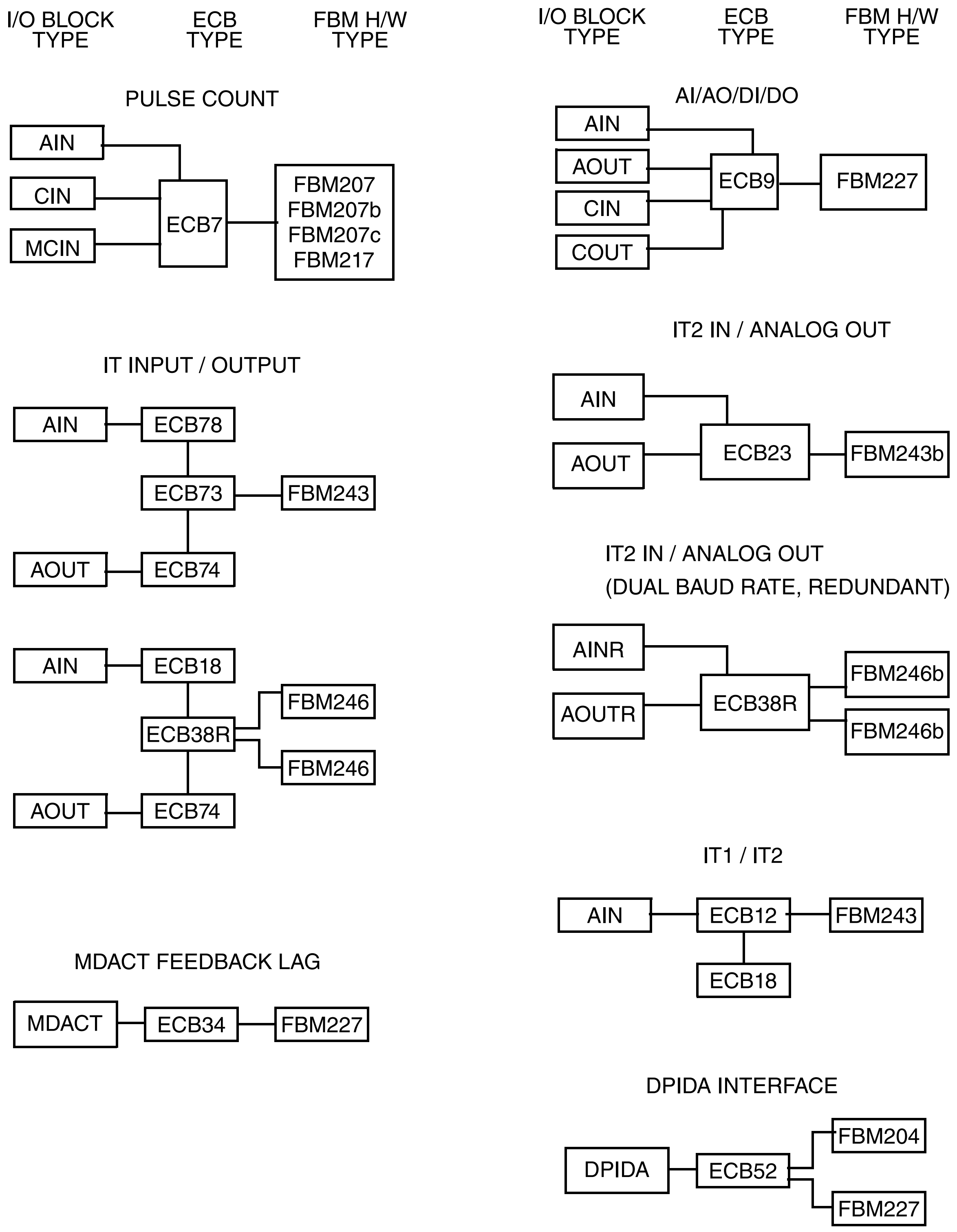


Figure 11-8. Connections to ECB5, 6, 7, 8, 9, 23, 34 and 38R

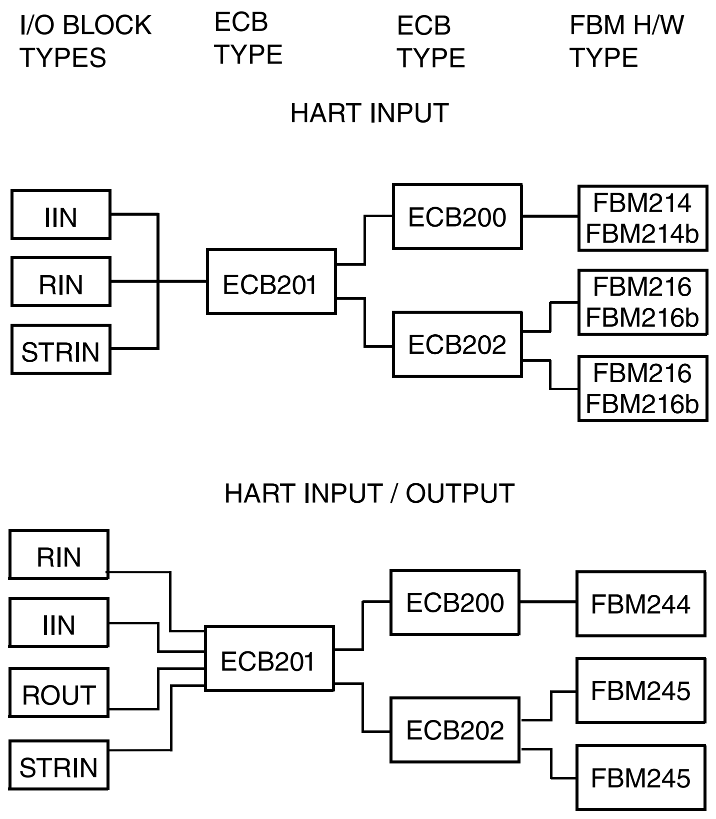


Figure 11-9. FBM214, 214b, 216, 216b, 244 and 245 Connections to ECB200, 201, and 202

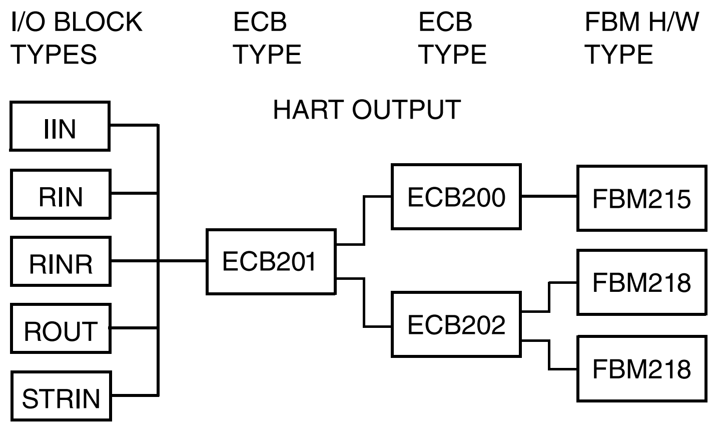


Figure 11-10. FBM215 and 218 Connections to ECB200, 201, and 202

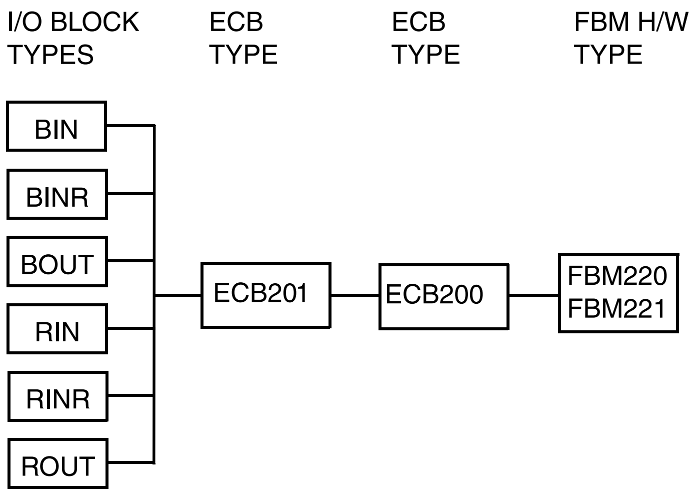


Figure 11-11. FBM220 and 221 Connections to ECB200 and 201

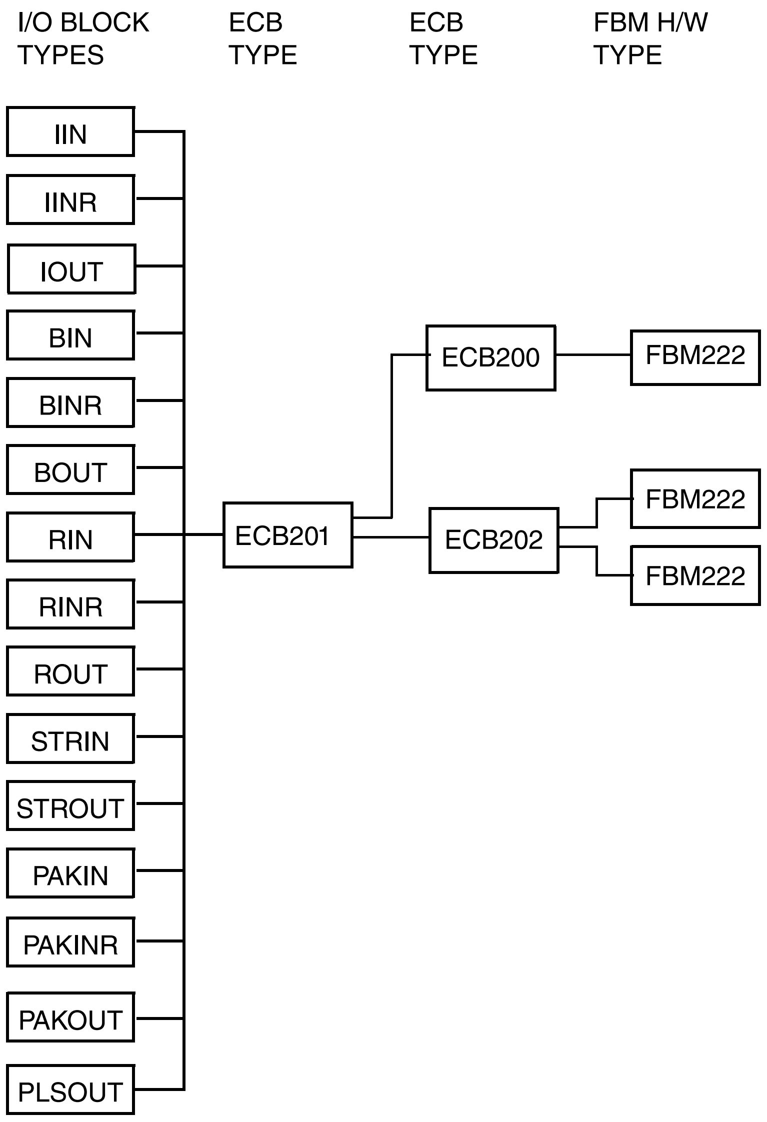


Figure 11-12. FBM222 Connections to ECB200, 201, and 202

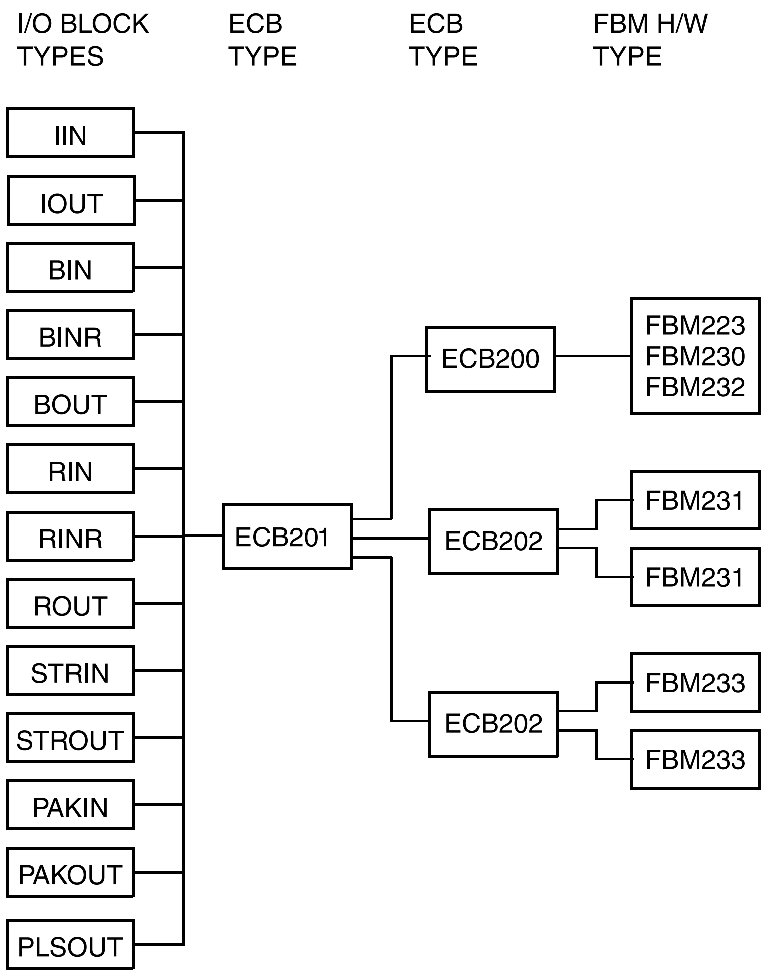


Figure 11-13. FBM223, 230, 231, 232 and 233 Connections to ECB200, 201, and 202

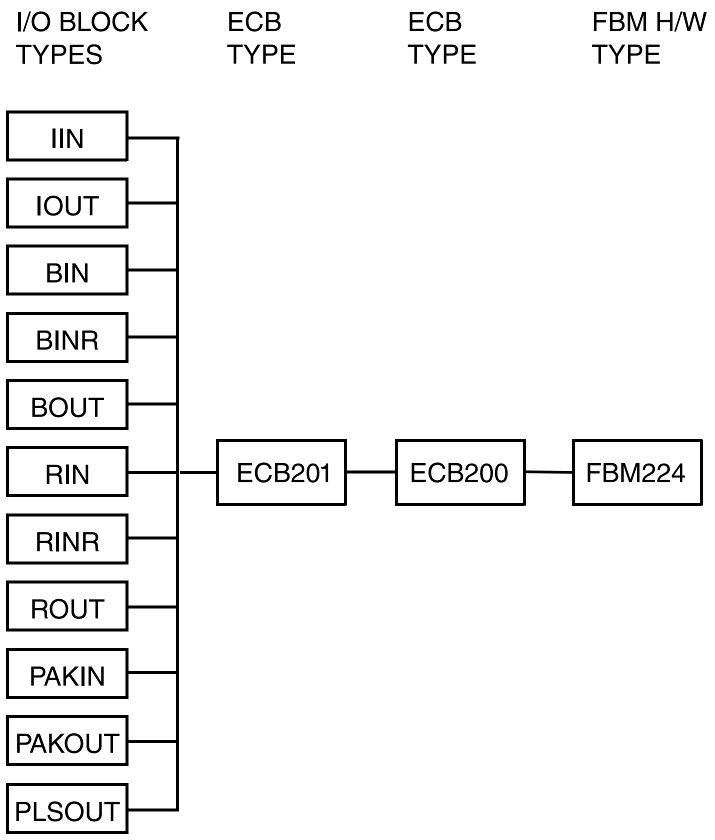


Figure 11-14. FBM224 Connections to ECB200 and 201

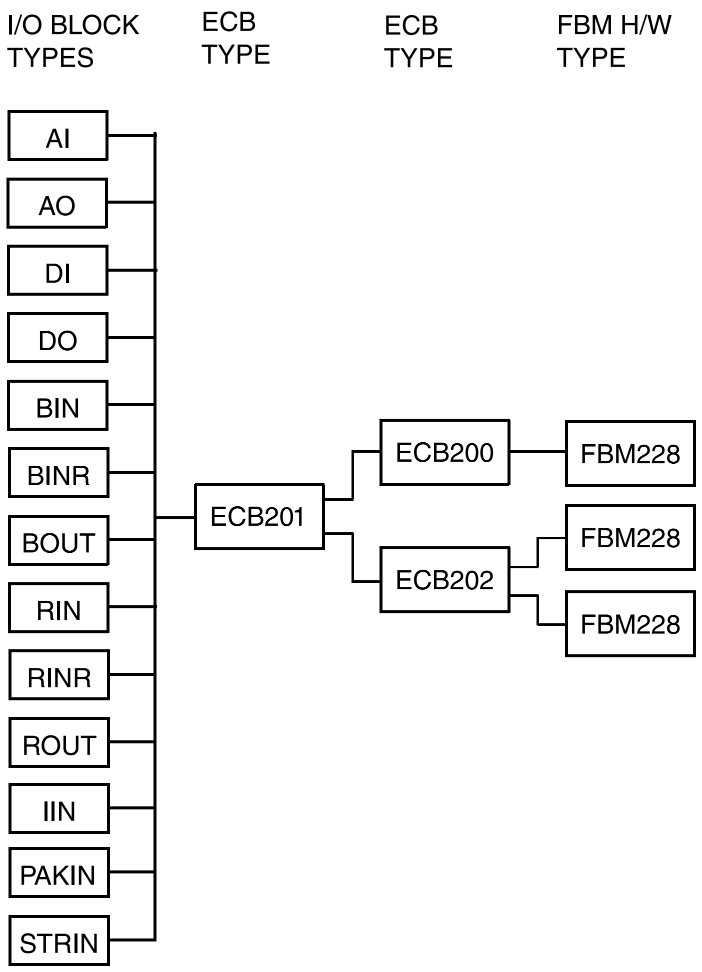


Figure 11-15. FBM228 Connections to ECB200, 201, and 202

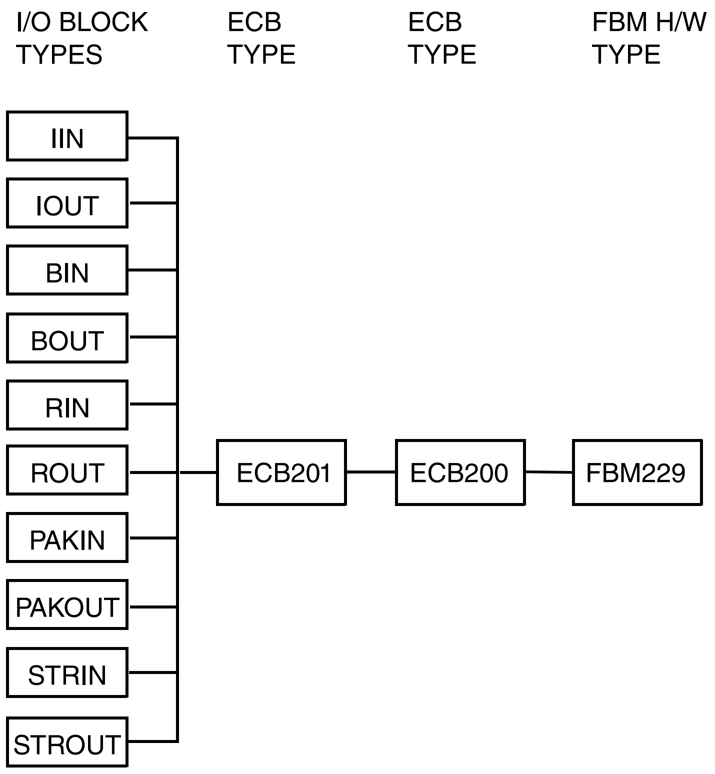


Figure 11-16. FBM229 Connections to ECB200 and 201

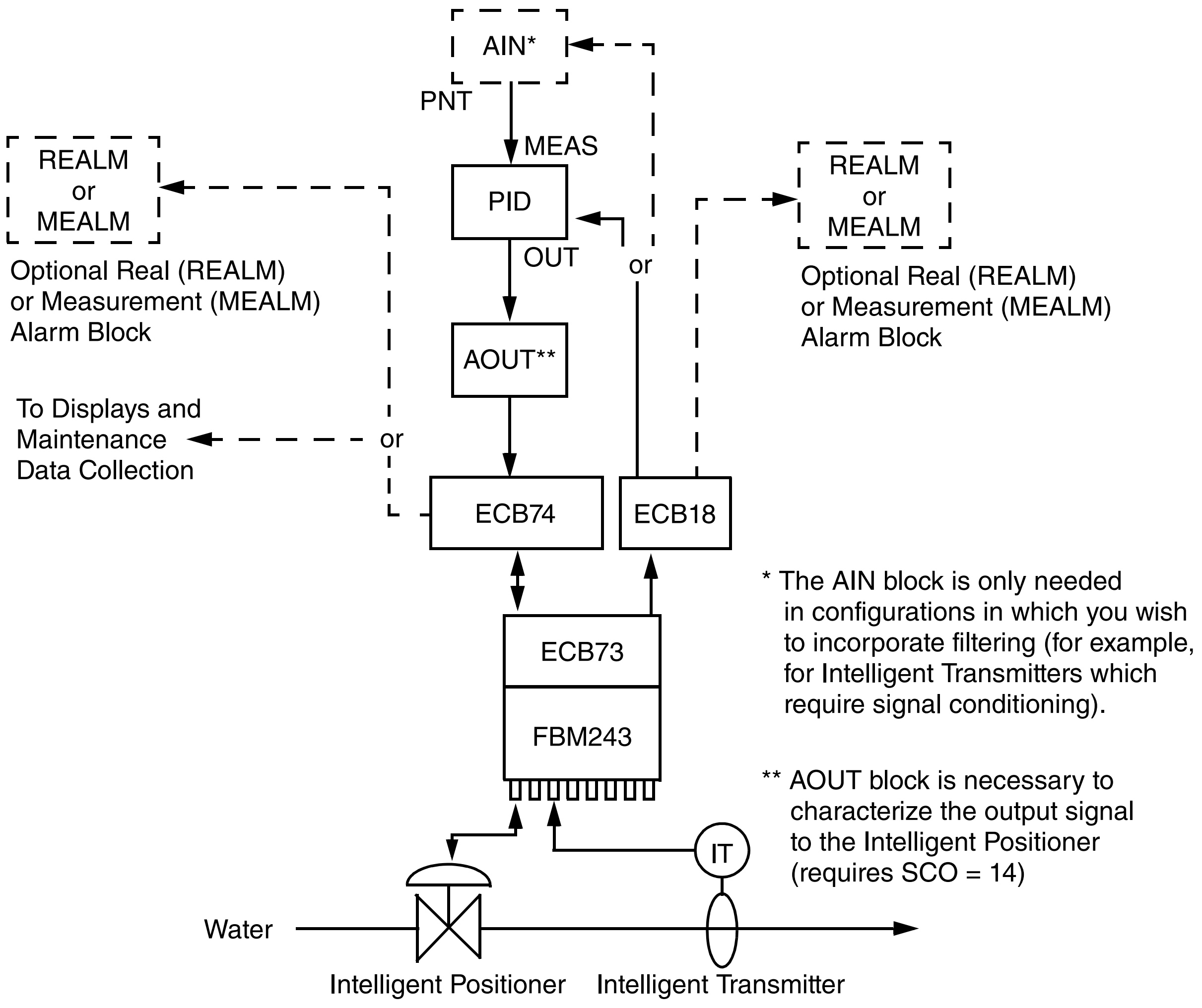

AIN*
PNT
MEAS
PID
OUT
AOUT**
ECB74
ECB18
ECB73
FBM243
REALM or MEALM
Optional Real (REALM) or Measurement (MEALM) Alarm Block
REALM or MEALM
Optional Real (REALM) or Measurement (MEALM) Alarm Block
To Displays and Maintenance Data Collection
or
or
IT
Water
Intelligent Positioner
Intelligent Transmitter
* The AIN block is only needed in configurations in which you wish to incorporate filtering (for example, for Intelligent Transmitters which require signal conditioning).
** AOUT block is necessary to characterize the output signal to the Intelligent Positioner (requires SCO = 14)

Figure 11-17. Typical FBM243 and ECB73 Configuration

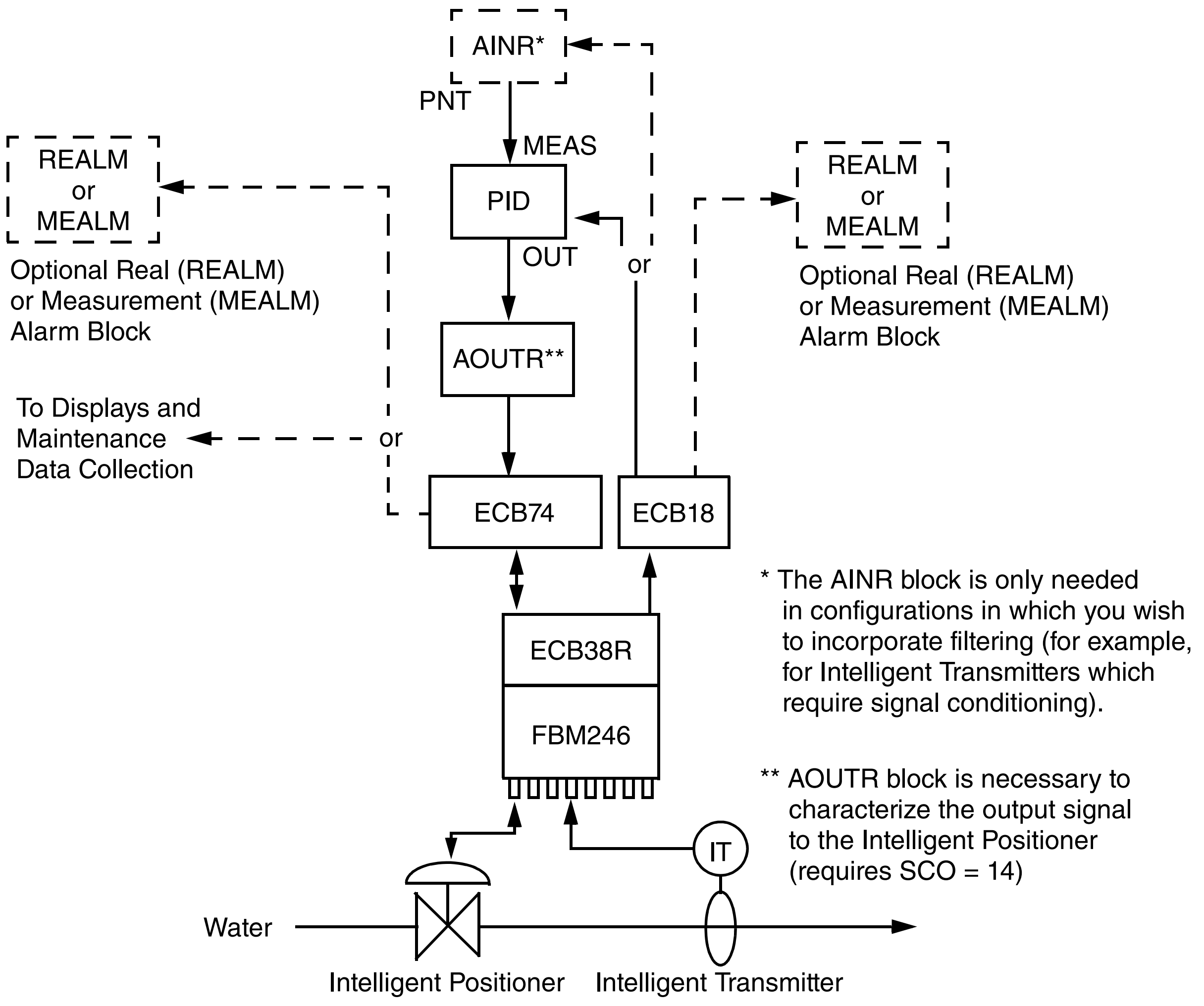

AINR*
PNT
MEAS
PID
OUT
or
REALM or MEALM
Optional Real (REALM) or Measurement (MEALM) Alarm Block
REALM or MEALM
Optional Real (REALM) or Measurement (MEALM) Alarm Block
AOUTR**
To Displays and Maintenance Data Collection
or
ECB74
ECB18
ECB38R
FBM246
IT
Water
Intelligent Positioner
Intelligent Transmitter
* The AINR block is only needed in configurations in which you wish to incorporate filtering (for example, for Intelligent Transmitters which require signal conditioning).
** AOUTR block is necessary to characterize the output signal to the Intelligent Positioner (requires SCO = 14)

Figure 11-18. Typical FBM246 and ECB38R Configuration

ECB Interfaces to I/O Blocks and 100 Series FBMs

This section uses a series of interface diagrams to show the relationship of the ECBs to the I/O blocks at the Control interface, and to the 100 series FBMs at the Process interface.

Each diagram shows all the I/O blocks and all the FBMs that connect to the ECB in the center of the diagram.

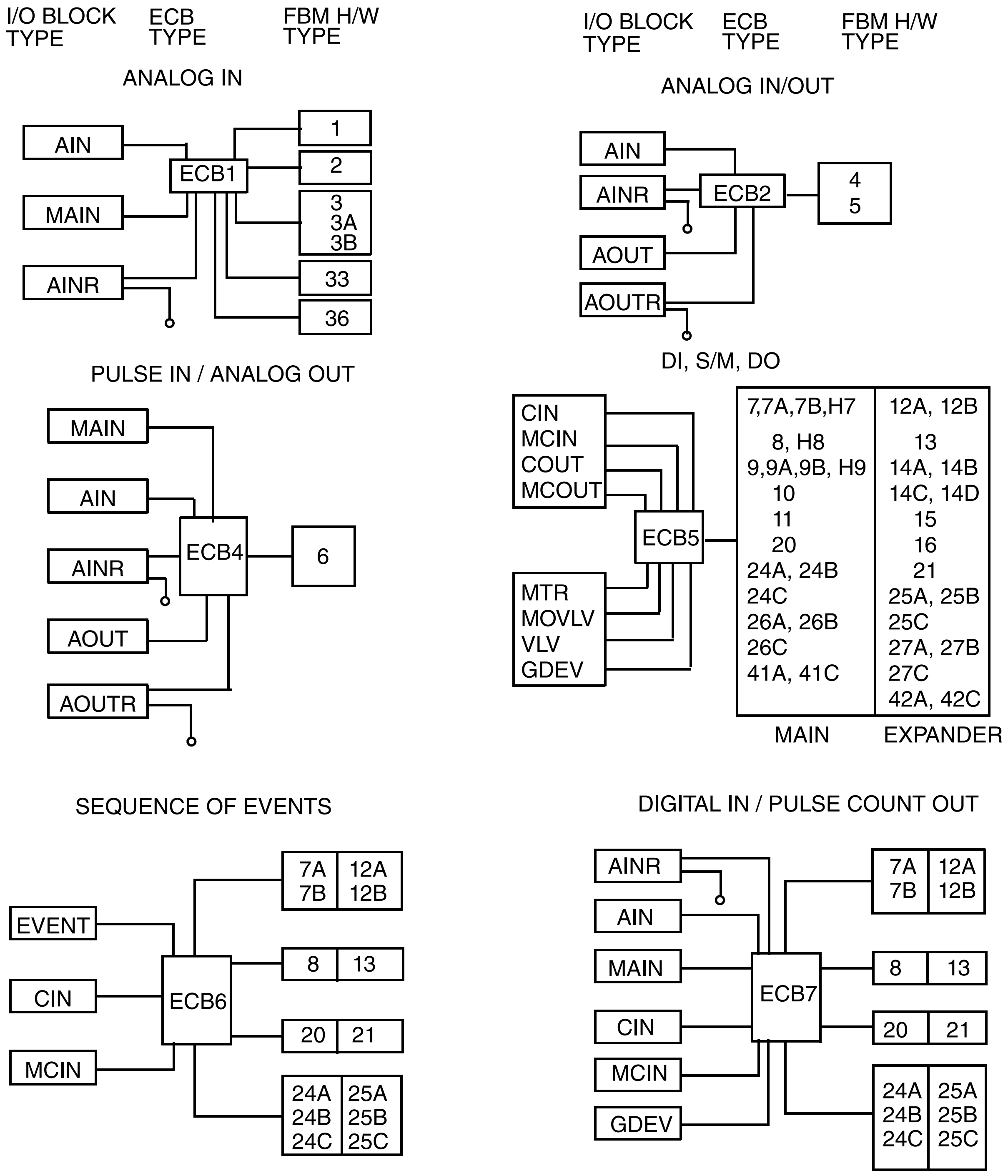


Figure 11-19. Connections to ECB1/2/4/5/6/7

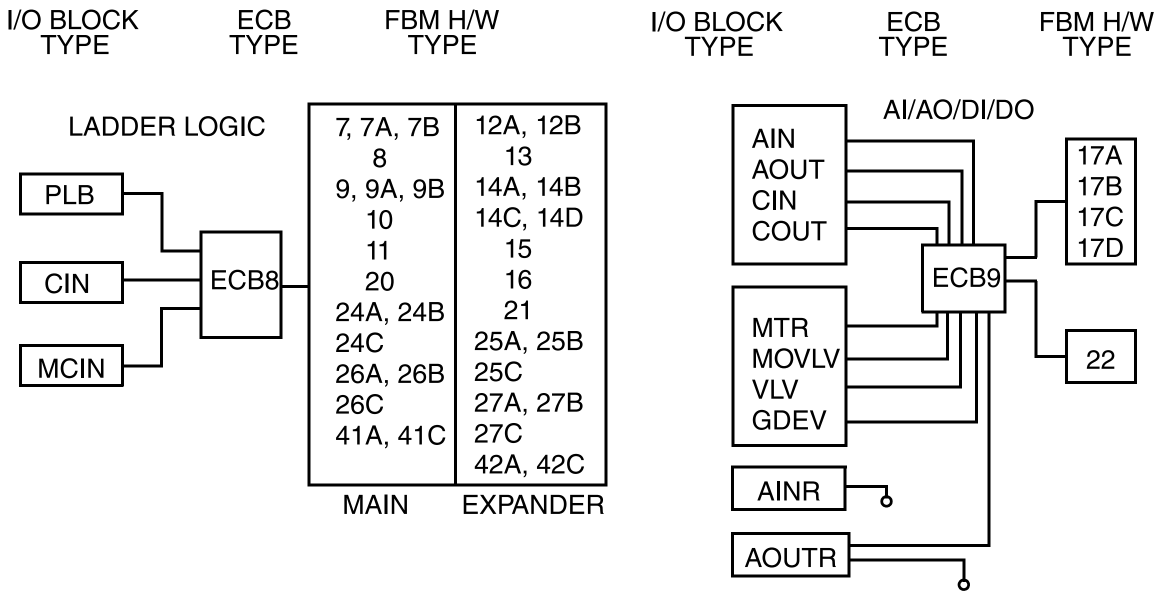


Figure 11-20. Connections to ECB8/9

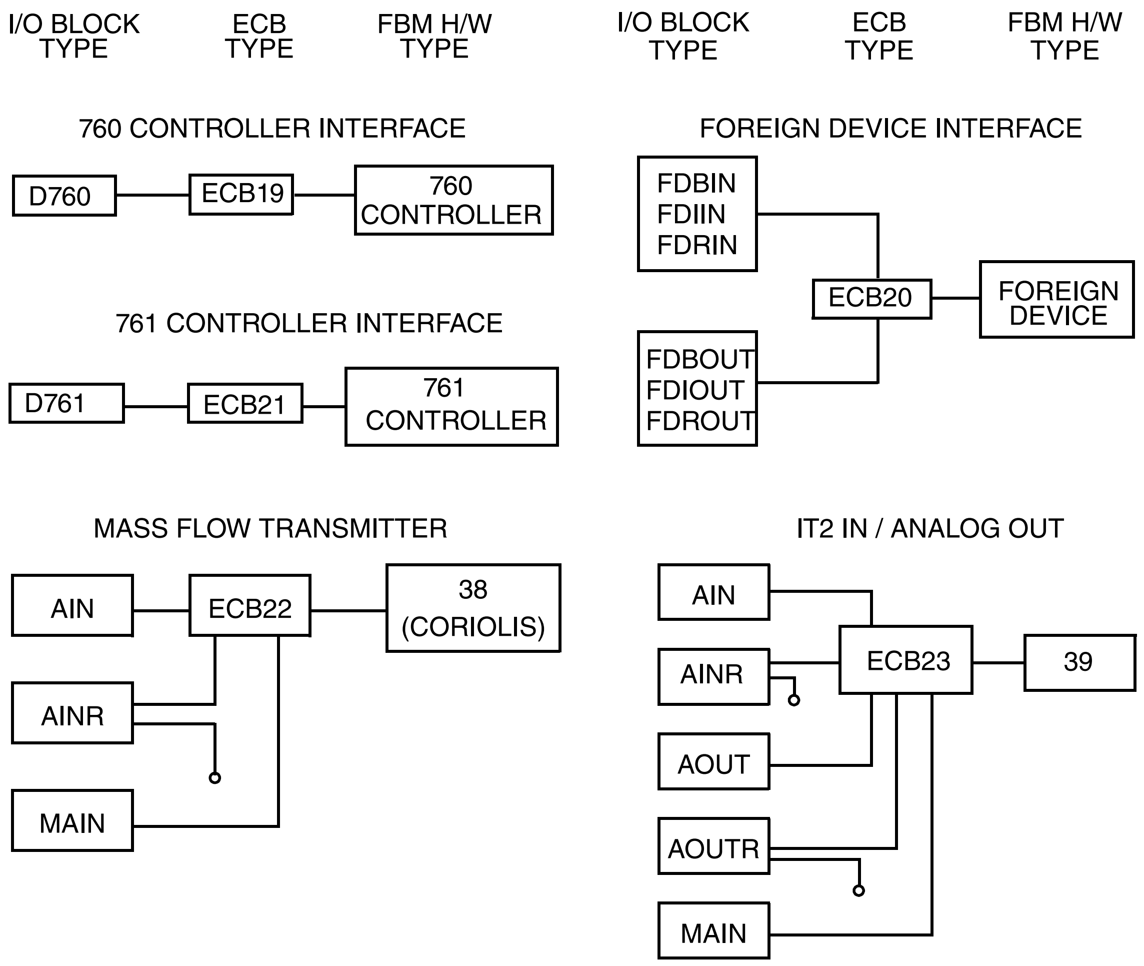


Figure 11-21. Connections to ECB19/20/21/22/23

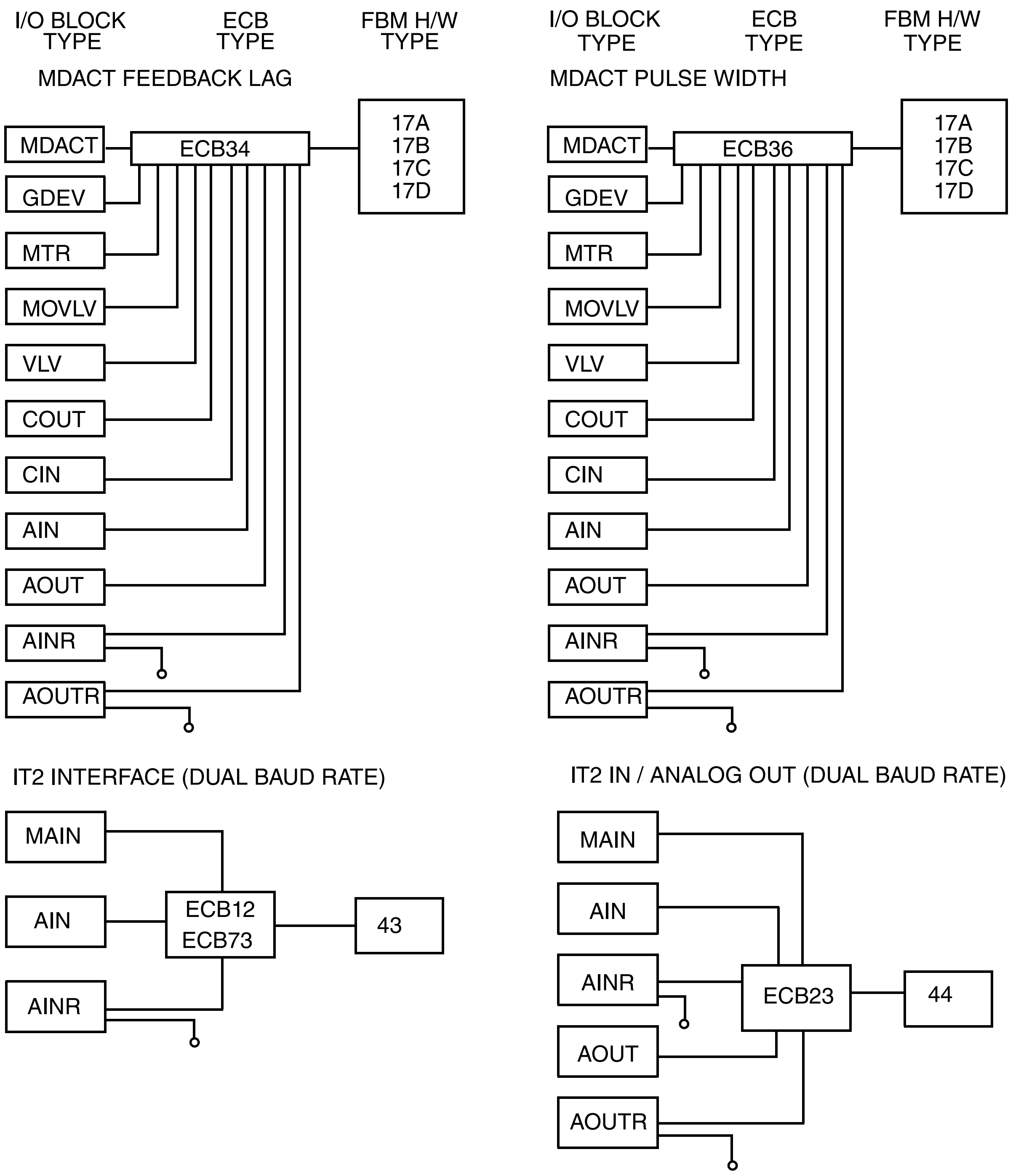


Figure 11-22. Connections to ECB12/23/34/36/73

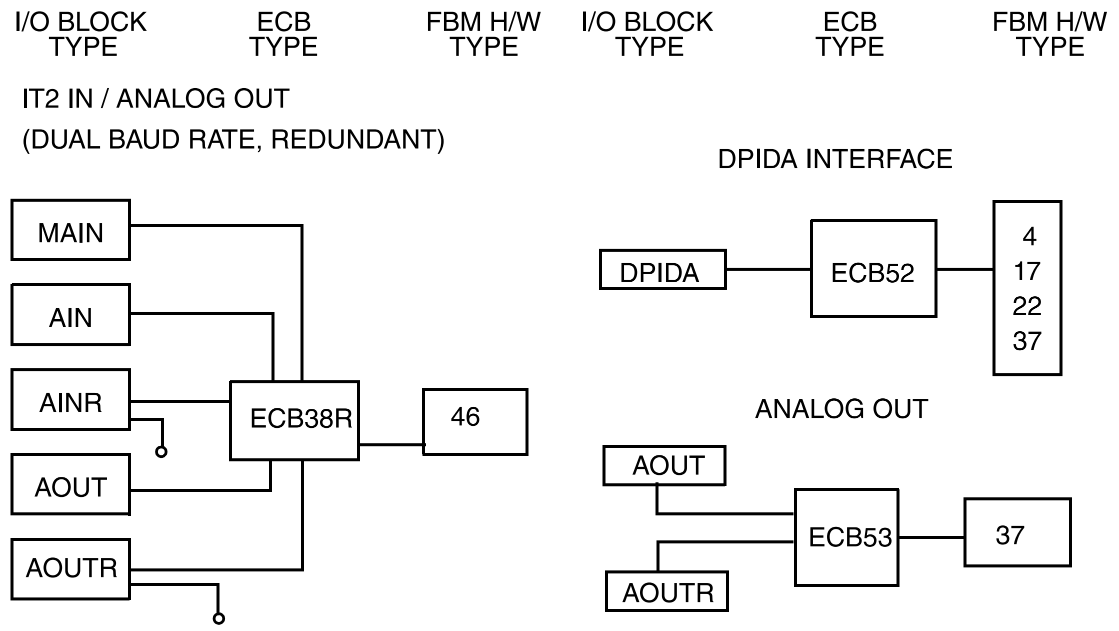


Figure 11-23. Connections to ECB38/52/53

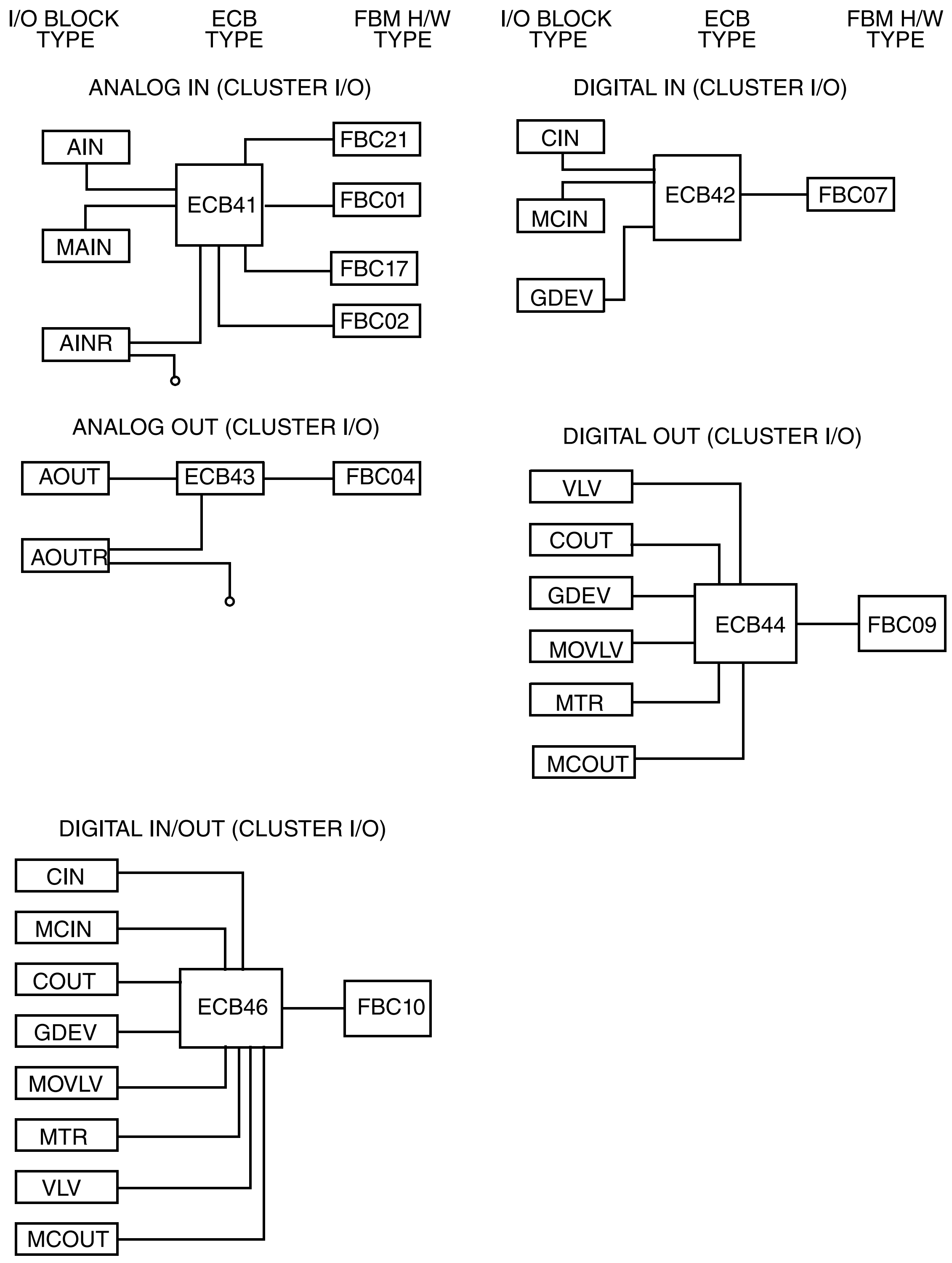


Figure 11-24. Connections to ECB41/42/43/44/46

Window Equipment Control Blocks

Window Equipment Control Blocks (window ECBs) provide an extended communications interface between the Foxboro Evo Process Automation System and Intelligent Field Devices (IFDs), the next generation of field devices. IFDs, as their name implies, incorporate intelligence at the hardware device level, such as signal conditioning and even process control, and so require a new software interface. This interface is the Window ECB, which integrates IFDs into the Foxboro Evo Process Automation System while providing access to the IFDs extended capabilities.

Window ECBs are device-specific and will grow in number as the number of IFDs increases.

This section discusses:

- ♦ General Window ECB Concepts
- ♦ Individual Window ECBs

The following table lists the current Window ECBs and the specific devices they support.

Table 11-6. Window Equipment Control Blocks

Window ECB	Supported Device
ECB18	The 820 Series and 860 Series Intelligent Transmitters and Vortex Meters

General Window ECB Concepts

Overview

Because IFDs are intelligent devices, the model of the FBM (hardware) and ECB (software) providing the device interface to the CP no longer holds in exactly the same way. Many of the functions performed by CP blocks, such as signal conditioning and translation to engineering units, may now be performed by the IFD. Accordingly, the software interface to the IFD, which the window ECB provides, requires both less and more support from the CP. Some IFDs connect directly to an FBM, others, to an intermediary device.

On the one hand, many of the functions previously performed by ECB and I/O blocks are now performed directly by the device. Thus, many signals require only appropriate pass through to the Foxboro Evo Process Automation System to be integrated into the control context. On the other hand, because the devices themselves offer more configuration and reporting capabilities, the software interface between the Foxboro Evo Process Automation System and the device must now make these capabilities accessible from the CP. Thus, window ECBs perform two basic functions:

- ♦ The Window ECB offers access to IFD inputs and outputs as Control Core Services parameters, allowing inputs to be set for simulation purposes and connected directly to Control Core Services blocks, including control (for example, PIDE) and alarm (MSG, STALM, and MEALM) blocks.
- ♦ The Window ECBs, through an extended, graphically-based interface—namely, the window—provide access to the IFD's sophisticated, on-board capabilities, ranging from minimal amounts of device configuration to device-specific functions that go far beyond the management of control-related contact points.

The following subsections detail general Window ECB capabilities and requirements.

Configuration

Window ECBs operate directly on the device for configuration. Configuration parameters are of two types. The first group representing essentially non-dynamic data specifies the operation of the window ECB, for example, period, phase, and source connections for outbound process data. These parameters are configurable through the IACC or ICC in the same way as parameters of standard blocks.

The second group of parameters represents dynamic, real time data, usually measurements or status information (the parameter names and data types vary with each window ECB). These parameters are normally not settable, since they reflect current device values. However, a special BYPASS switch allows for simulation setting.

To support device-specific alarm functions, window ECBs can be connected to other blocks, such as STALM, MEALM, and MSG.

Installation

The window ECB presents the device as a group of Control Core Services parameters, providing access to device-specific parameters, but also thereby integrating device outputs into the generic parameter interface demanded by the Foxboro Evo Process Automation System. Accordingly, all window ECBs, regardless of device, follow the same overall design. The following description of the generic window ECB also serves as a narrative of how the window ECB is installed, made active, operates, and is shut down.

Validation

Window ECBs validate themselves only regarding normal connection. Device parameter validation is, for IFDs, device-resident and thus will take place prior to connection.

Initialization

Initialization is invoked as part of the PIO Maintenance task based upon the generic flag request mechanism. Initialization performs two basic functions:

1. The initialization required by the window ECB's processes (the connected dynamic data). This initialization is executed before any other ECB code, allowing resetting the ECB history of the previous cycle states.
2. The initialization of non-dynamic data by directly reading the device.

During initialization, all parameters are marked OOS (out of service), and remain OOS until the first input/output scan.

The trigger event for start up initialization is the ECB off-line to on-line transition. Upon completion, the static data upload function triggers the update of displays and other processes dependent on the data. Device-specific window ECBs provide for multiple trigger events for complete and partial re-initializations at times other than start up; for example, device reconfiguration and communication failures (see the individual window ECBs for details).

Real Time Input Points

To provide real time input points, window ECBs present device values as Control Core Services parameters of the appropriate type. Window ECBs thus include a portion of the functionality usually provided by CIN and AIN blocks. In addition, window ECBs support simulation and hardware testing.

The window ECB for Cluster I/O (ECB47) does not contain any real time input or output point data. This dynamic data is contained in the ECBs for the separate FBCs attached to the cluster, and these ECBs are not treated as window ECBs. ECB47, which serves the cluster's communication interface, the FBP10, is considered a window ECB since it is treated as such by System Management and the CP I/O scan logic. The information in this section and the section on Real Time Output Points does not apply to the ECB47 window.

Input Point Parameters

Each window ECB input point is associated with three parameters, DEVICE_VALUE, POINT_VALUE, and BYPASS, where:

- DEVICE_VALUE is the generic name for parameters IVAL1 to IVAL_3
- POINT_VALUE is the generic name for parameters MEAS1 to MEAS3
- BYPAS is the generic name for parameters BYP16

Up to three points can come from the device.

DEVICE_VALUE is always secured, since the value can be updated only from the device and thus cannot be overwritten if data flow is interrupted. If the device can supply "bad" values, the input point must have a "last good value" parameter (see below).

POINT_VALUE is the value to which the control strategy connects. Its behavior is controlled by the BYPASS parameter.

The BYPASS parameter is a single bit of a packed boolean or packed long parameter. By default, it is set to Off, which enables direct copying of DEVICE_VALUE to POINT_VALUE, to be updated every input cycle. When simulation is required, however, you can set BYPASS to On, and enable manual setting of the POINT_VALUE. The point's DEVICE_VALUE parameter continues to reflect the real time value from the device, and is immediately copied to POINT_VALUE upon turning BYPASS off.

The "last good value" parameter is required if (and only if) the device can supply "bad" values. It is a packed boolean or packed long configurable parameter.

Input Point Scanning

Input points are updated through real time input scanning based on the window ECB's period and phase parameters. The scanning first sends data requests to the device, then processes the communication responses from the device to update the input point parameters.

Updating a window ECB input point requires the simultaneous update of related status parameters, to be consistent with Control Core Services conventions. The following parameters ensure compliance:

- OM on scan, which is true unless the value is a sink for a broken or invalid connection.
- ON control, which is always true unless the window is Off.
- OOS (Out-Of-Service), set in all cases when fresh data is not available (for example, communication interruption, during initialization, window Off).
- BAD, which implies a fault or error condition.
- ERROR, which indicates a problem with the upstream data required to compute the value.

Real Time Output Points

The control system acts by manipulating output points for data that are sent to the device. The output points are, however, more than the result of a conversion to the device's signal conventions. Rather, they are one of the means whereby the Foxboro Evo Process Automation System implements its control strategy, for events ranging from state transitions such as initialization to communication failures and field device state changes.

Accordingly, the functionality associated with a point can range from a switch toggled from a process operator display to a set point fed from a complex cascade. The former, a simple settable output, is implemented with a single shadow parameter. The latter requires all the elements of the Foxboro Evo Process Automation System cascade interface, including connection points for the cascade's INITI, BCALCI inputs and support for the PRIBLK handshake.

Output Point Parameters

The following parameters support this wide range of output point functions:

- Device output point, a shadow parameter that associates two values with the same name: a current field value and a hidden, "in-transit" write value.
- Control output point, the device output point's point value parameter, to enable test, simulation, and reconfiguration.
- R/L switch, to enable remote and local connections for output manipulation.
- Upstream BCALCI support, provided by the shadow parameter point in simple cases and a device-supported parameter for complex cases, to be called BCALCO only in this case.
- Upstream INITI support, expected to be connected to INITO, if any controller is to be connected upstream.
- The PRIBLK handshake, to ensure that the output value has been computed before being output. A per-point enable is required in multiple point contexts.

Output Point Scanning

As with input point scanning, a first phase constructs the outbound messages to the device, the second phase interprets and acts on the replies. The write operation is the inverse of the data transformation performed by input scanning. The model for state definition and transition logic requirements is the AOUT block, which is the point of reference for the window ECB output point parameters.

Initialization allows for outputs to be read back.

In the updating of shadow parameters the read half of the parameter is to be updated at the proper control point in cases of communication failure. In such cases, the value waiting to be written back cannot necessarily simply be kept.

Device status requirements ensures the output value is "good" before being sent to the device, in cases where the device does not itself perform validation.

Commfail Issues

For IFDs, communication failures must be handled by the device, since all the Control Core Services can do is create alarms once communication has failed.

Shutdown

The trigger event for shutdown is the SMDH off-line action. It is functionally similar to turning off a compound: the window ECB goes OOS and real time scanning of the device is shut down.

Alarming

The Foxboro Evo Process Automation System supports two families of alarms: process alarms and system alarms. Process alarms highlight process conditions; system alarms report exception conditions relating to the control system itself. IFDs, which include both diagnostic logic to report hardware faults and application-level intelligence recognizing process exception conditions require both process and system alarms.

Alarm blocks—STALM, MEALM, and MSG—provide Foxboro Evo Process Automation System support for device-resident alarms, by converting the device's detection of alarm conditions into a parameter interface to the Control Core Services

Device States and Status

The ECBSTA parameter is designed to coordinate control actions with device states and status for IFDs, whose device states are often complex.

The ECBSTA parameter has two portions, a generic control status section and a device-specific section. The device-specific portion is defined by each particular window ECB.

The generic portion of ECBSTA provides information in three basic categories: window, device, and data flow. Each is mutually exclusive and inherently hierarchical.

Window States

A window has two states, On and Off. Transitions are triggered by SMDH on-line and off-line actions.

Off shuts down all mechanisms associated with the window. All parameters are marked OOS. Existing values are left unchanged. Where appropriate, output parameters are released.

On turns on all mechanisms associated with the window. With the window state on, the device's functions are part of the user's real time control strategy. The device state must be ON_SCAN.

Device States

Devices have five explicit, mutually exclusive states. A sixth state, OFF, follows from the OFF window state.

1. ON_SCAN, which is the device state that supports real time control. In the ON_SCAN state, device parameters can be used by the Control Core Services.
2. COMM_FAIL, which is entered when communication fails and left when communication is resumed. In this state, window points are BAD and OOS.
3. DEVICE_FAILED, which is entered when the device reports a fatal hardware fault or other fatal error condition, except a communication failure.
4. DEVICE_NOT_READY, which is the transition state, as in normal start-up.
5. DEVICE_OOS, a transition state set by the operator or other external control. The device is healthy, but not to be used for its normal functions.

Data Flow Indicators

In the case of data flowing up from a device, the status section of the ECBSTA parameter provides all necessary information for control and display purposes.

For data flowing down to the device, status alone does not suffice. To account for per-point status, there is a packed boolean OUTSTA parameter. Each output point is assigned two bits. The first bit, when set, indicates data is being held by the window Control Core Services output interface logic. The second, when set, indicates that the output is not writable at the device level. OUTSTA is marked OOS if the device is not ON_SCAN.

Additional Status Indicators

These are window-level device interface properties:

1. Window Configuration Error, which indicates an invalid parameter connection, indicated in the ECBSTA parameter.
2. Initialization Complete, which indicates that the window's support data from the device is complete.
3. Simulation Active, which indicates that the window ECB is in simulation mode.
4. Bypass Active, which is set if any of the window BYPASS bits is set (it is the OR of all per-point window BYPASS switches).
5. Output Point Open, which is set if any of the point OUTSTA parameters is set (it is the OR of all window point OUTSTA parameters).
6. Fail-safe, which indicates that the device considers that Control Core Services' Control has failed and is controlling Control Core Services outputs to itself.

Individual Window ECBs

Individual window ECBs conform to the general window ECB design presented earlier. This section discusses conceptual features specific to the each window ECB.

Intelligent Transmitter Window ECB (ECB18)

The Intelligent Transmitter window ECB (IT ECB) receives up to three measurement values and status conditions from the device for integration into the control scheme. ECB18 is internally a window ECB interface to the ECB12 parent IT ECB directly accepting communication variables, information parameters, and bypass switches.

From the IT ECB, operators can set upper and lower range levels for each input. Status conditions are also available, for display and for optional connections to the control scheme. The IT ECB supports both hardware fault reporting and bad measurement detection.

Optional features include marking primary measurements as bad when the temperature measurement is out of range, and "last good value" retention.

Appendix A. Signal Conditioning Indexes

This Appendix defines the conversion algorithms for signal conditioning indexes. These are optionally offered for both the AIN and MAIN block types. The inverses of some of these algorithms are available for use in the AOUT and AOUTR blocks.

Input Signal Conditioning

Each Signal Conditioning Input (SCI) index specifies a particular conversion algorithm that converts the raw count data value to a real floating point value in the specified engineering units span. The following subsection describes the signal conditioning indexes for the 100 and 200 Series Fieldbus Modules (FBMs).

FBM Input Signal Conditioning

Characterizer tables are used to convert raw data from either an FBM202, FBM212, or FBM02 and FBM36 Millivolt I/O Module (for thermocouple temperature measurements), or an FBM203, FBM213, or FBM03 RTD I/O Module (for resistance temperature measurements).

Characterizer tables support various standard thermocouple and RTD sensor types for signal conversion indexes in the range 20 through 39.

The thermocouple FBM provides a fixed millivolt range of -10.5 to 69.5 mV for 0 to 64,000 counts. The overall conversion resolution is 2.5 microvolts/count.

The RTD FBM provides a fixed resistance range of 0 to 320 Ohms for 0 to 64,000 counts. The overall conversion resolution is 10 milliohms/count.

The engineering span is specified by the high scale and low scale parameters for the PNT output. The high scale and low scale values correspond to the raw signal values of 64,000 counts and 0 counts, respectively. Note that the user may use the CHAR block extender to build custom piecewise linear SCIs.

Linear and Nonlinear Signal Conversion (FBM)

SCI = 0: No Conditioning

```
converted_value = raw_value
```

SCI = 1: Linear 0 to 64000 raw counts

```
converted_value = (raw_value * (HSCO1 - LSCO1) / 64000) + LSCO1
```

SCI = 2: Linear 1600 to 64000 raw counts

```
converted_value = ((raw_value - 1600) * (HSCO1 - LSCO1) / 62400) +
LSCO1
```

SCI = 3: Linear 12800 to 64000 raw counts

```
converted_value = ((raw_value - 12800) * (HSCO1 - LSCO1) / 51200)
+ LSCO1
```

SCI = 4: Square Root 0 to 64000 raw counts

```
converted_value = (SQRT(64000 * raw_value) * (HSCO1 - LSCO1) /
64000) + LSCO1
```

SCI =5: Square Root 12800 to 64000 raw counts

```
IF raw_value <= 12800
THEN
   converted_value = LSCO1;
ELSE
   converted_value = (SQRT(51200 * (raw_value - 12800)) * (HSCO1 -
   LSCO1) / 51200) + LSCO1;
```

SCI = 6: Square Root 0 to 64000 raw counts with low cutoff

```
IF raw_value <= 480
THEN
   converted_value = LSCO1;
ELSE
   converted_value = (SQRT(64000 * raw_value) * (HSCO1 - LSCO1) /
   64000) + LSCO1;
```

SCI = 7: Square Root 12800 to 64000 raw counts with low cutoff

```
IF raw_value <= 13184
THEN
   converted_value = LSCO1;
ELSE
   converted_value = (SQRT(51200 * (raw_value - 12800)) * (HSCO1-
   LSCO1) / 51200) + LSCO1;
```

SCI = 9: Linear 1600 to 64000 raw counts with low cutoff

```
IF raw_value <= 1600
THEN
   converted_value = LSCO1;
ELSE
   converted_value = ((raw_value - 1600) * (HSCO1 - LSCO1) / 62400)
   + LSCO1;
```

SCI = 10: Linear 12800 to 64000 raw counts with low cutoff

```
IF raw_value <= 12800
THEN
   converted_value = LSCO1;
ELSE
   converted_value = ((raw_value - 12800) * (HSCO1 - LSCO1) /
   51200) + LSCO1;
```

SCI = 12: Linear 14080 to 64000 raw counts

```
IF raw_value <= 14080
THEN
  converted_value = ((raw_value - 14080) * (HSCO1 - LSCO1) /
  49920) + LSCO1
```

SCI = 13: Square Root 14080 to 64000 raw counts with low cutoff

```
IF raw_value <= 14080
THEN
  converted_value = LSCO1;
ELSE
  converted_value = (SQRT(49920 * (raw_value - 14080)) * (HSCO1
    - LSCO1) / 49920) + LSCO1;
```

Pulse Rate (FBM)

SCI = 8: Pulse Rate

SCI = 8, converts the incoming pulse rate raw value into the proper engineering units. The FBM can measure a range of pulse rate frequencies from 1/(integration time) to the maximum hardware limit 25 kHz for FBM206 or 12.5 kHz for FBM06.

A meter factor scaling constant (MTRF) is provided for the SCI 8 option. It can be used to translate the measured pulse rate signal (Hz) to the desired rate measurement in the chosen engineering units; for example, flow rate in gallons per minute.

Therefore, the possible span of the converted value can range between the low cutoff frequency and the maximum FBM pulse rate frequency of 25 kHz (or 12.5 kHz).

MTRF has the units of the configured engineering units (EO) per Hz.

```
converted_value = MTRF * pulses_per_second (EO)
```

The assigned values for the output engineering range scale parameters, HSCO1 and LSCO1, are not used during the conversion process, because the output span is not normalized to the FBM's conversion span. They are provided for display indication purposes only, and are usually configured to represent the range of the measuring device; for example, a flow meter. If the range of the measuring device is unknown, then they may be set to the engineering units that correspond to the FBM's conversion range, which ranges between fco and 25 kHz (for FBM206, for example); that is,

```
HSCO1 = MTRF * 25,000
LSCO1 = MTRF * fco
```

Thermocouples (FBM)

SCI = 20: Type B Thermocouple

Material	:	Platinum-Platinum
Range	:	0 to 1820 deg C
Curve	:	P331-0/68 (TI 15-18a)

SCI = 21: Type E Thermocouple

Material	:	Chromel™-Constantan
Range	:	-270 to 910 deg C
Curve	:	S303-0/68 (TI 5-17c)

SCI 23: Type J Thermocouple

Material	:	Iron-Constantan
Range	:	-210 to 1200 deg C
Curve	:	S99J-0/68 (TI 5-12f)

SCI 24: Type K Thermocouple

Material	:	Chromel-Alumel™
Range	:	-270 to 1372 deg C
Curve	:	K223-0/68 (TI 5-13c)

SCI 25: Type N Thermocouple

Material	:	Nicrosil-Nisil
Range	:	-270 to 1300 deg C
Curve	:	IPTS-68 (TI 5-19)

SCI = 26: Type R Thermocouple

Material	:	Platinum-Platinum
Range	:	-50 to 1768 deg C
Curve	:	P329-0/68 (TI 5-14d)

SCI = 27: Type S Thermocouple

Material	:	Platinum-Platinum
Range	:	-50 to 1768 deg C
Curve	:	P307-0/68 (TI 5-15e)

SCI = 28: Type T Thermocouple

Material	:	Copper-Constantan
Range	:	-270 to 400 deg C
Curve	:	S233-0/68 (TI 5-11c)

(FBM) Resistance Temperature Detectors (RTDs)

SCI = 40: Copper RTD (SAMA)

Material	:	Copper
Range	:	-70 to 150 deg C
Curve	:	CR-229 (TI 5-25a)

SCI = 41: Nickel RTD (SAMA)

Material	:	Nickel
Range	:	-100 to 160 deg C
Curve	:	NR-227 (SAMA) (TI 5-24a)

SCI = 42: Platinum RTD (100 Ohm DIN 43760-1968)

Material	:	Platinum
Range	:	0 to 620 deg C
Curve	:	PR-238 (TI 5-26a)

SCI = 43: Platinum RTD (100 Ohm IEC) (DIN 43760-1980)

Material	:	Platinum
Range	:	-200 to 620 deg C
Curve	:	Foxboro drawing 10104MV (TI 5-28)

SCI = 44: Platinum RTD (100 Ohm SAMA)

Material	:	Platinum
Range	:	-200 to 600 deg C
Curve	:	PR-279 (SAMA) (TI 5-27a)

Other Linear/Square Root SCIXs

SCI = 50: Linear 0 to 65535 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 65535 ) + LSCO1
```

SCI = 51: Linear -32768 to 32767 raw counts

```
Conditioned = (( Raw + 32768 ) * ( HSCO1 - LSCO1 ) / 65535 ) +
LSCO1
```

SCI = 52: Linear 0 to 32767 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 32767 ) + LSCO1
```

SCI = 53: Linear 0 to 1000 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 1000 ) + LSCO1
```

SCI = 54: Linear 0 to 9999 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 9999 ) + LSCO1
```

SCI = 55: Linear 0 to 2048 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 2048 ) + LSCO1
```

SCI = 56: Linear 409 to 2048 raw counts

```
Conditioned = (( Raw - 409 ) * ( HSCO1 - LSCO1 ) / 1639 ) +
LSCO1
```

SCI = 57: Square root 0 to 2048 raw counts

```
Conditioned = ( SQRT( 2048 * Raw ) * ( HSCO1 - LSCO1 ) / 2048 ) +
LSCO1
```

SCI = 58: Square root 409 to 2048 raw counts with low cutoff

```
IF raw_value <= 409
THEN
  Conditioned = LSCO1;
ELSE
  Conditioned = ( SQRT( 1639 * ( Raw -  409 ) ) * ( HSCO1 - LSCO1 )
/ 1639 ) + LSCO1;
```

SCI = 59: Linear 0 to 4095 raw counts

```
Conditioned = ( Raw * ( HSCO1 - LSCO1 ) / 4095 ) + LSCO1
```

Standard Conversion Tables (for FBMs)

The following Foxboro Technical Information sheets (TIs) contain the standard conversion tables.

TI #	Title
5-18a	Thermocouple Temperature/mV Data For Platinum-6% Rhodium vs. Platinum-30% Rhodium Thermocouples, Type B
5-17c	Thermocouple Temperature/mV Data For Nickel-Chromium vs. Copper-Nickel (Chromel-Constantan) Thermocouples, Type E
5-34	Electromotive Force vs. Temperature For PRC Thermocouple Curve EA-2
5-12f	Thermocouple Temperature/mV Data For Iron vs. Copper-Nickel (Iron-Constantan) Thermocouples, Type J
5-13c	Thermocouple Temperature/mV Data For Nickel-Chromium vs. Nickel-Aluminum (Chromel-Alumel) Thermocouples, Type K
5-19	Temperature-Electromotive Force (EMF) Tables For Type N (Nicrosil vs. Nisil) Thermocouples
5-14d	Thermocouple Temperature/mV Data For Platinum vs. Platinum-13% Rhodium Thermocouples, Type R
5-15e	Thermocouple Temperature/mV Data For Platinum vs. Platinum-10% Rhodium Thermocouples, Type S
5-11c	Thermocouple Temperature/mV Data For Copper vs. Copper-Nickel (Copper-Constantan) Thermocouples, Type T
5-25a	SAMA Copper RTD Temperature-Resistance Tables Curve CR-229 (Degrees C vs. Absolute Ohm) and Curve CR-228 (Degrees F vs. Absolute Ohm)

TI #	Title
5-24a	SAMA Type II Nickel RTD Temperature-Resistance Tables Curve NR-227 (Degrees C vs. Absolute Ohm) and Curve NR-226 (Degrees F vs. Absolute Ohm)
5-26a	DIN Platinum RTD Temperature-Resistance Tables Curve PR-238 (Degrees C vs. Absolute Ohm) and Curve PR-239 (Degrees F vs. Absolute Ohm)
5-28	ASTM* and IEC* Platinum RTD Temperature-Resistance Tables (Degrees C vs. Absolute Ohm and Degrees F vs. Absolute Ohm)
5-27a	SAMA 100 Ohm (Nominal) Platinum RTD Temperature-Resistance Tables Curve PR-279 (Degrees C vs. Absolute Ohm) and Curve PR-278 (Degrees F vs. Absolute Ohm)

Output Signal Conditioning

The following subsection describes the output Signal Conditioning indexes (SCOs) for the 100 and 200 Series Fieldbus Modules.

SCOs for FBMs

Before an output block writes a value to an appropriate type FBM, it is converted to raw counts.

The analog value, which is written to an FBM point, is converted from engineering units to counts using the high and low output range. The processing is the inverse operation of signal conditioning the input (SCI). This value is conditioned according to the applied signal conditioning output (SCO) index. The result is clamped between the high and low actuator limit value.

If the output (OUT) of the block is sent to an FBM, then it must be converted, by the application of signal conditioning, from engineering units within Range Output 1 (RO1) into raw count, which is an integer value in the range used by the particular FBM. The value IOMOPT = 1 specifies that FBMs are connected.

You may also apply signal conditioning, for test purposes, when no FBMs are connected. You should configure IOMOPT = 0 in this case. The values of IOM_ID, IOMIDR, and PNT_NO are ignored.

The value IOMOPT = 2 indicates that there are no connected FBMs and signal conditioning is not used. The values of IOM_ID, IOMIDR, PNT_NO, and SCO are all ignored.

The values of SCO, which must be in the ranges 0 to 5 or 12 to 13, cause the following inverse linear or inverse square root signal conditioning to be applied to the scaled, balanced, and clamped input. In each case, the input to the signal conditioning algorithm is the value of OUT, and the output of the algorithm is stored in RAWC:

The following output Signal Conditioning indexes are applicable for analog output blocks used in conjunction with FBMs:

SCO = 0: No Conditioning

```
raw count = output
```

SCO = 1: 0 to 64000 inverse linear
(Analog Output 0 to 20 mA)

```
raw count = (output - LSCO1) * 64000 / (HSCO1 - LSCO1)
```

SCO = 2: 1600 to 64000 inverse linear
(Analog Output 0 to 10 V dc)

```
raw count = ((output - LSCO1) * 62400 / (HSCO1 - LSCO1) + 1600
```

SCO = **3:** 12800 to 64000 inverse linear
(Analog Output 4 to 20 mA)

```
raw count = ((output - LSCO1) * 51200 / (HSCO1 - LSCO1) + 12800
```

SCO = **4:** 0 to 64000 inverse square root
(Analog Output 0 to 20 mA)

```
raw count = ((output - LSCO1) * 62400 / (HSCO1 - LSCO1))2 / 64000
```

SCO = **5:** 12800 to 64000 inverse square root, clamped
(Analog Output 4 to 20 mA)

```
raw count = ((output - LSCO1) * 51200 / (HSCO1 - LSCO1))2 / 51200 +
12800
```

SCO = **12:** 14080 to 64000 inverse linear
(Analog Output 2 to 10 V dc)

```
raw count = ((output - LSCO1) * 49920 / (HSCO1 - LSCO1)) + 14080
```

SCO = **13:** 14080 to 64000 inverse square root with low cutoff
(Analog Output 2 to 10 V dc)

```
raw count = ((output - LSCO1) * 49920 / (HSCO1 - LSCO1))2 / 49920 +
14080
```

SCO = **14:** Inverse linear (0 to 16383)

```
RAWC = ( OUT - LSCO1 ) * 16383 / ( HSCO1 - LSCO1 )
```

SCO = **15:** Inverse square root (1600 to 64000)

$$\text{RAWC} = ((\text{OUT} - \text{LSCO1}) * 62400 / (\text{HSCO1} - \text{LSCO1}))^2 / 62400 + 1600$$

SCO = **50:** Inverse linear (0 to 65535)

```
RAWC = ( OUT - LSCO1 ) * 65535 / ( HSCO1 - LSCO1 )
```

SCO = **51:** Inverse linear (-32768 to 32767)

```
RAWC = ( OUT - LSCO1 ) * 65535 / ( HSCO1 - LSCO1 ) - 32768
```

SCO = **52:** Inverse linear (0 to 32767)

```
RAWC = ( OUT - LSCO1 ) * 32767 / ( HSCO1 - LSCO1 )
```

SCO = **53:** Inverse linear (0 to 1000)

```
RAWC = ( OUT - LSCO1 ) * 1000 / ( HSCO1 - LSCO1 )
```

SCO = **54:** Inverse linear (0 to 9999)

```
RAWC = ( OUT - LSCO1 ) * 9999 / ( HSCO1 - LSCO1 )
```

SCO = **55:** Inverse linear (0 to 2048)

```
RAWC = ( OUT - LSCO1 ) * 2048 / ( HSCO1 - LSCO1 )
```

SCO = **56:** Inverse linear (409 to 2048)

```
RAWC = ( OUT - LSCO1 ) * 1639 / ( HSCO1 - LSCO1 ) + 409
```

SCO = **59:** Inverse linear (0 to 4095)

```
RAWC = ( OUT - LSCO1 ) * 4095 / ( HSCO1 - LSCO1 )
```

Each SCO value has an associated raw span, consisting of the difference between the upper and lower range values of the raw value. For example, SCO = 2 has a raw span of 64000 minus 1600, or 62400. The upper range value is always 64000 for any nonzero SCO.

The effect of inverse linear signal conditioning is to place RAWC at the same relative position within the raw span as OUT occupies within the engineering units span.

Example:

If,

LSCO1	= 20^0C
HSCO1	= 90^0C
OUT	= 50^0C
SCO	= 2

Then,

RAWC = ((50 - 20) * 62400 / (90 - 20)) + 1200
= (30 * 62400/70) + 1200 = 27943

Inverse square root signal conditioning is similar, but the relationship between RAWC and OUT is non-linear.

Appendix B. Fieldbus Modules

This appendix lists and describes Fieldbus Module types and their software configurable specifications.

200 Series Fieldbus Module Types (FBMs)

FBM201, FBM201b, FBM201c, FBM201d (0 to 20 mA Inputs)

The Analog Input Interface Modules (FBM201, FBM201b, FBM201c, FBM201d) contain eight analog input channels, each channel accepting a 2-wire, dc input from an analog sensor such as a 4 to 20 mA or 0 to 5V transmitter, or a self-powered 20 mA source. Input ranges for each channel are:

- FBM201 - 0 to 20.4 mA dc - available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):
  - Compact 200 Series FBM version
  - Standard 200 Series FBM version
- FBM201b - 0 to 100 mV dc
- FBM201c - 0 to 5 V dc
- FBM201d - 0 to 10 V dc

The module performs the signal conversion required to interface the electrical input signals from the field sensors to the redundant Fieldbus.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options for this program and their ranges follow.

Table B-1. FBM201, FBM201b, FBM201c, FBM201d Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

For accuracy specifications, refer to *FBM201/b/c/d Analog Input (0 to 20 mA, 0 to 100 mV, 0 to 5 V, 0 to 10 V dc) Interface Modules* (PSS 31H-2Z1).

Conversion Time (software configurable): see Table B-2.

Table B-2. FBM201, FBM201b, FBM201c, FBM201d Conversion Time Factors

Conversion Time	Resolution	Settling Time[1] (Milliseconds)	Update Time	RES Value
50 ms	15 bits	100	25 ms	5
100 ms	15 bits	125	25 ms	1
200 ms	15 bits	200	25 ms	2
500 ms	15 bits	500	25 ms	3
1000 ms	15 bits	1000	25 ms	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM202 (Thermocouple/mV Inputs)

The Thermocouple/mV Input Interface Module (FBM202) contains eight isolated thermocouple input channels, and one isolated reference junction compensation channel (for terminal temperature sensing).

The FBM202 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

Each channel has a full scale indication on burnout feature (up-scale) and accepts standard thermocouples for various temperature ranges.

The module performs the signal conversion required to interface the electrical input signals from the thermocouples to the redundant Fieldbus.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-3. FBM202 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-9)	Normalized Raw Counts/100 ms

Input Range: -10.5 to 78.125 mV dc equals 0 to 65535 raw counts.

Reference Junction:

For discrete and direct terminal connections, a 100 ohm platinum RTD is internally provided at the termination cable assembly.

For a plug termination connection, the reference junction connection is provided by the user with a 4-wire 100 ohm platinum RTD (IEC 751, Class B).

For accuracy specifications, refer to *FBM202, Thermocouple/mV Input Interface Modules* (PSS 31H-2Z2).

Conversion Time (software configurable): see Table B-4.

Table B-4. FBM202 Conversion Time Factors

Conversion Time	Resolution	Settling Time[1] (Milliseconds)	Update Time	RES Value
50 ms	15 bits	100	25 ms	5
100 ms	15 bits	125	25 ms	1
200 ms	15 bits	200	25 ms	2
500 ms	15 bits	500	25 ms	3
1000 ms	15 bits	1000	25 ms	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM203, FBM203b, FBM203c and FBM203d (RTD Inputs)

The Platinum and Nickel Resistance Temperature Detector (RTD) Input Interface Module contains eight RTD input channels. Each channel accepts a 2-, 3-, or 4-wire RTD sensor input:

- FBM203 - within a 0 to 320 ohm resistance range - available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):
    - Compact 200 Series FBM version
    - Standard 200 Series FBM version
- FBM203b - within a 0 to 640 ohm resistance range
- FBM203c - within a 0 to 30 ohm resistance range
- FBM203d - within a 0 to 320 ohm resistance range; when used with termination assembly RH924EX (supersedes P0924EX), it supports the functionality of a 100 Series FBM03 used with a 2-wire or 4-wire RTD sensor input.

The module performs the signal conversion required to interface the electrical input signals from the RTD to the redundant Fieldbus.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-5. FBM203/b/c/d Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

Input Range (each channel):

FBM203 and FBM203d: 0 to 320 ohms
FBM203b: 0 to 640 ohms
FBM203c: 0 to 30 ohms

Sensor Current:

FBM203 and FBM203d: 0.19 mA dc
FBM203b: 0.10 mA dc
FBM203c: 0.54 mA dc

For accuracy specifications, refer to *FBM203/b/c/d Platinum/Nickel/Copper RTD Input Interface Modules* (PSS 31H-2Z3).

Conversion Time (software configurable): see Table B-6.

Table B-6. FBM203/b/c/d Conversion Time Factors

Conversion Time	Resolution	Settling Time[1] (Milliseconds)	Update Time	RES Value
50 ms	15 bits	100	25 ms	5
100 ms	15 bits	125	25 ms	1
200 ms	15 bits	200	25 ms	2
500 ms	15 bits	500	25 ms	3
1000 ms	15 bits	1000	25 ms	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM204 (0 to 20 mA Inputs/Outputs)

The Channel Isolated 0 to 20 mA Input/Output Interface Module contains four 20 mA dc analog input channels and four 20 mA dc analog output channels.

The FBM204 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version

- Standard 200 Series FBM version

Each input channel accepts an analog sensor input such as a 4 to 20 mA transmitter or a self-powered 20 mA source.

Each output channel drives an external load and produces a 0 to 20 mA output.

The module performs the signal conversion required to interface the electrical input/output signals from/to the field sensors to/from the redundant Fieldbus.

The module independently connects to the redundant Fieldbus. This module executes the Analog I/O application program. The configurable options and their ranges follow.

Table B-7. FBM204 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms
Analog Output: Fail-safe Configuration (Hold/ Fallback on a per-channel basis)	0 = fallback; 1 = hold
Fallback Values (Channels 5-8)	0 to 64000 Raw Counts

For accuracy specifications, refer to *FBM204, 0 to 20 mA I/O Interface Module* (PSS 31H-2Z4).

Conversion Time (software configurable): see Table B-8.

Table B-8. FBM204 Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	25 ms	100	5
100 ms	15 bits	25 ms	125	1
200 ms	15 bits	25 ms	200	2
500 ms	15 bits	25 ms	500	3
1000 ms	15 bits	25 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM205 (Channel Isolated, Redundant 0 to 20 mA Inputs/Outputs)

The Channel Isolated, Redundant 0 to 20 mA I/O Interface Module (FBM205) contains four isolated 0 to 20 mA dc analog input channels and four isolated 20 mA dc analog output channels. Each input and output channel is galvanically isolated from the other channels and ground.

A redundant pair of the modules combine to provide redundancy at the Fieldbus Module (FBM) level, with field I/O wired to one common termination assembly. Each module independently attempts to hold the output(s) at its specified output value(s), and each independently reports its observed value of the inputs. A redundant analog input and redundant analog output block in the control software validates each input and output in conjunction with information to/from the module.

In the control processor, a redundant analog output function block, AOUTR, is used for each redundant pair of outputs. The AOUTR block handles output writes and initialization logic for the redundant channels. On each execution cycle of the AOUTR block, identical output writes are sent to both FBMs in the redundant pair, fully exercising the communication path to the FBMs and the logic circuitry of each FBM.

A redundant analog input function block, AINR, is used for each redundant pair of inputs. The AINR block handles input reads and initialization logic for the redundant channels. On each execution cycle of the AINR block, identical read commands are sent to both FBMs in the redundant pair, fully exercising the communication path to the FBMs and the logic circuitry of each FBM.

Each input channel accepts an analog sensor input, such as a 4 to 20 mA transmitter or a self-powered 20 mA source. Each output channel drives an external load and produces a 0 to 20 mA output. Transmitter power from each module is diode OR'd together in the redundant adapter to assure redundant power. The microprocessor of each module executes the analog I/O application program, plus security routines that validate the health of the FBM.

Input channel options include a configurable choice of analog input conversion time on a per module basis. Input channel security is enhanced by redundantly powering the input current loop from per-channel power supplies in each FBM of the pair.

The configurable options for output security and their ranges are listed in Table B-9. The Analog Output Fail-safe Fallback Data option must be set for 0 mA output. This removes one of the pair of redundant output channels from service for detectable problems such as an FBM not properly receiving output writes, not passing security tests on FBM microprocessor writes to output registers, failure of internal FBM diagnostics, or FBM module watchdog timer time-out. Setting of the Output Fail-safe Configuration (Hold/Fallback) option for 0 mA output also minimizes the possibility of a “fail high” result.

Table B-9. FBM205 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms
Analog Output: Fail-safe Configuration (Hold/ Fallback on a per-channel basis)	0 = fallback; 1 = hold
Fallback Values (Channels 5-8)	0 to 64000 Raw Counts

For accuracy specifications, refer to *FBM205, 0 to 20 mA I/O Interface Module (Redundant)* (PSS 21H-2Z5 B4).

Conversion Times (software configurable) are listed in Table B-10.

Table B-10. FBM205 Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	25 ms	100	5
100 ms	15 bits	25 ms	125	1
200 ms	15 bits	25 ms	200	2
500 ms	15 bits	25 ms	500	3
1000 ms	15 bits	25 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM206 and FBM206b (Pulse Inputs)

The Pulse Inputs Interface module (FBM206) contains multiple configurable pulse input channels.

FBM206 contains eight isolated, independent, configurable pulse input channels, each accepting a pulse input with a maximum rate of 25 kHz.

FBM206b contains four isolated, independent, configurable pulse input channels, each accepting a pulse input with a maximum rate of 25 kHz, as well as four isolated independent 0 to 20 mA analog output channels, which are used when FBM206b replaces the 100 Series FBM06.

The input devices include vortex and turbine meters, solid state or electro-mechanical contacts and other sensors with similar pulse outputs.

The module performs the signal conversion required to interface the electrical input signals from the field sensors from the redundant Fieldbus.

The module independently connects to the redundant Fieldbus, and executes the Pulse Input application program.

Table B-11. FBM206b Options

Option	Range
Analog Output Failsafe Configuration (Hold/Fallback on a per channel basis)	0 = fallback; 1 = hold
Fallback Values (Channels 5-8)	0 to 64000 Raw Counts

Conversion Times (software configurable) are listed in Table B-12.

Table B-12. FBM206 and FBM206b Conversion Time Factors

Conversion Time (Seconds)	Update Time (Milliseconds)	Settling Time (Seconds)	Minimum Frequency	RES Value
0.1	10	0.25	8 Hz	1
0.2	10	0.5	4 Hz	2
0.5	20	1.0	2 Hz	3
1.0	20	2.0	1 Hz	4

FBM207, FBM207b and FBM207c (Contact/dc Inputs)

The Contact/dc Input Interface Module functions as a 16-channel dc voltage monitor (FBM207) or 16-channel contact sensor (FBM207b or 207c).

The FBM207b is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

This set of modules perform the signal conversion required to interface these digital (that is, on/off state) electrical input signals from the field sensors to the redundant Fieldbus.

The modules independently connect to the Fieldbus.

The modules execute the Digital I/O or Ladder Logic application program. The configurable options for the program are:

Table B-13. FBM207 Options

Option	Range

Input Filter Time	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

Input Range (each channel): Contact open (off) or closed (on).

For accuracy specifications, refer to *FBM207/b/c Voltage Monitor/Contact Sense Input Interface Modules* (PSS 31H-2Z7).

FBM208 and FBM208b (Channel Isolated, Redundant, Readback, 0 to 20 mA Inputs/Outputs)

The Channel Isolated, Redundant with Readback, 0 to 20 mA I/O Interface Module (FBM208/FBM208b) contains four isolated 0 to 20 mA dc analog input channels and four isolated 20 mA dc analog output channels. Each input and output channel is galvanically isolated from the other channels and ground.

NOTE

The FBM208b may only be used to migrate field I/O wiring formerly used with the 100 Series FBM05 to the DIN rail mounted subsystem.

A pair of the modules combine to provide redundancy at the Fieldbus Module (FBM) level, with field I/O wired to one common termination assembly. Each module independently attempts to hold the output(s) at its specified output value(s), and each independently reports its observed value of the inputs. A redundant analog input and redundant analog output block in the control software validates each input and output in conjunction with information to/from the module.

In addition, each FBM208/FBM208b monitors its own input loop power supply and reports the input as BAD if the supply voltage drops below minimum compliance levels, thereby preventing a possible undetected fault.

A redundant analog input function block, AINR, is used for each redundant pair of inputs. The AINR block handles input reads and initialization logic for the redundant channels. On each execution cycle of the AINR block, identical reads are sent to both modules, fully exercising the Fieldbus and the logic circuitry of each module.

Each input channel accepts an analog sensor input such as a 4 to 20 mA transmitter or a self-powered 4 to 20 mA source. Each output channel drives an external load and produces a 0 to 20 mA output. Transmitter power from each module is diode OR'd together in the redundant adapter to assure redundant power. The microprocessor of each module executes the analog I/O application program, plus security routines that validate the health of the FBM.

Input channel options include a configurable choice of analog input conversion time on a per module basis. Input channel security is enhanced by redundantly powering the input current loop from per-channel power supplies in each FBM of the pair.

In the control processor, a redundant analog output block, AOUTR, is used for each redundant pair of outputs. The AOUTR block handles output writes and initialization logic for the redundant channels. On each execution cycle of the AOUTR block, identical output writes are sent to both modules, fully exercising the Fieldbus and the logic circuitry of each module.

The FBM compares the output value with the current readback value. If the readback value differs from the desired output by more than ± 2%, the output channel is marked BAD. In addition, if the output value is greater than the desired output by more than +2%, the power to that channel is shut off preventing the bad channel from interfering with the control of that channel by the redundant partner FBM. The power to a failed channel remains off until the FBM is replaced or rebooted by the user.

When the output channel is marked BAD, the CP presents that information to the system for display as a System Management warning alarm and a control block alarm.

Configurable options for output security and their ranges are listed in Table B-14. The Analog Output Fail-Safe Fallback Data option must be set for 0 mA output. This removes one of the pair of redundant output channels from service for detectable problems such as a module not properly receiving output writes or not passing security tests on FBM microprocessor writes to output registers. Setting of the Analog Output Fail-Safe Fallback Data option for 0 mA output also minimizes the possibility of a “fail high” result.

Table B-14. FBM208 and FBM208b Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms
Analog Output: Fail-safe Configuration (Hold/ Fallback on a per-channel basis)	0 = fallback; 1 = hold
Fallback Values (Channels 5-8)	0 to 64000 Raw Counts

For accuracy specifications, refer to *FBM208/b, Redundant with Readback, 0 to 20 mA I/O Interface Module* (PSS 31H-2Z8).

Conversion Times (software configurable) are listed in Table B-15.

Table B-15. FBM208 and FBM208b Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	25 ms	100	5
100 ms	15 bits	25 ms	125	1
200 ms	15 bits	25 ms	200	2

Table B-15. FBM208 and FBM208b Conversion Time Factors (Continued)

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
500 ms	15 bits	25 ms	500	3
1000 ms	15 bits	25 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM211 (16-Channel Differential-Isolated 0 to 20 mA Inputs)

The 16-Channel Differential-Isolated Analog Input Module (0 to 20 mA) contains sixteen 0 to 20.4 mA dc analog input channels.

Each input channel accepts a 2-wire, field powered 0 to 20 mA source.

The FBM211 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- ♦ Compact 200 Series FBM version
- ♦ Standard 200 Series FBM version

The module performs the signal conversion required to interface the electrical input signals from the field sensors to the redundant Fieldbus. Differential-Isolated means that the inputs are electrically separate module-to-module but not channel-to-channel on the same card.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-16. FBM211 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 50 ms update period 2 = 200 ms conversion time, 50 ms update period 3 = 500 ms conversion time, 50 ms update period 4 = 1000 ms conversion time, 50 ms update period 5 = 50 ms conversion time, 50 ms update period
Rate of Change Limits (Channels 1-16)	Normalized Raw Counts/100 ms

For accuracy specifications, refer to *FBM211, 0 to 20 mA Input Interface Module* (PSS 31H-2Z11).

Conversion Time (software configurable): see Table B-17.

Table B-17. FBM211 Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	50 ms	100	5
100 ms	15 bits	50 ms	125	1

Table B-17. FBM211 Conversion Time Factors (Continued)

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
200 ms	15 bits	50 ms	200	2
500 ms	15 bits	50 ms	500	3
1000 ms	15 bits	50 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.

FBM212 (14-Channel Differential-Isolated Thermocouple Inputs)

The 14-Channel Differential-Isolated Analog Input Fieldbus Module for Thermocouples contains 14 analog single-ended input channels, and one isolated reference junction compensation channel (for terminal temperature sensing).

Each channel has a full scale indication on burnout feature (up-scale) and accepts standard thermocouples for various temperature ranges.

The module performs the signal conversion required to interface the electrical input signals from the thermocouples to the redundant Fieldbus. Each channel has a differential input to allow voltage differences between channels without introducing errors. The channels are not galvanically isolated from each other, but are galvanically isolated from ground and module logic. Differential group isolated inputs and outputs use the FBM subsystem power supply for field power.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-18. FBM212 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 50 ms update period 2 = 200 ms conversion time, 50 ms update period 3 = 500 ms conversion time, 50 ms update period 4 = 1000 ms conversion time, 50 ms update period 5 = 50 ms conversion time, 50 ms update period
Rate of Change Limits (Channels 1-14)	Normalized Raw Counts/100 ms

Input Range: -10.5 to 78.125 mV dc.

Reference Junction:

For discrete and direct terminal connections, a 100 ohm platinum RTD is internally provided at the termination cable assembly.

For a wire termination connection, the reference junction connection is provided by the user with a 4-wire 100 ohm platinum RTD (IEC 751, Class B).

For accuracy specifications, refer to *FBM212 Thermocouple/mV Differential Input Interface Module* (PSS 31H-2Z12).

Conversion Time: see Table B-19.

Table B-19. FBM212 Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	50 ms	100	5
100 ms	15 bits	50 ms	125	1
200 ms	15 bits	50 ms	200	2
500 ms[2]	15 bits	50 ms	500	3
1000 ms[2]	15 bits	50 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.
2. Use 500 ms or 1000 ms conversion time for FBM212.

FBM213 (8-Channel Differential-Isolated RTD Inputs)

The 8-Channel Differential-Isolated Resistance Temperature Detector (RTD) Input Interface Module contains eight RTD input channels. Each channel accepts a 2- or 3-wire RTD sensor input within a 0 to 320 ohm resistance range.

The module performs the signal conversion required to interface the electrical input signals from the RTD to the redundant Fieldbus. Each channel has a differential input to allow voltage differences between channels without introducing errors. The channels are not galvanically isolated from each other, but are galvanically isolated from ground and module logic. Differential group isolated inputs and outputs use the FBM subsystem power supply for field power.

The module independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-20. FBM213 Options

Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 50 ms update period 2 = 200 ms conversion time, 50 ms update period 3 = 500 ms conversion time, 50 ms update period 4 = 1000 ms conversion time, 50 ms update period 5 = 50 ms conversion time, 50 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

Input Range (each channel): 0 to 320 ohms.

Sensor Current: 0.25 mA dc.

For accuracy specifications, refer to *Differential Platinum/Nickel RTD Input Module (FBM213)* (PSS 21H-2Z13B4).

Conversion Time (software configurable): see Table B-21.

Table B-21. FBM213 Conversion Time Factors

Conversion Time	Resolution	Update Time	Settling Time[1] (Milliseconds)	RES Value
50 ms	15 bits	50 ms	100	5
100 ms	15 bits	50 ms	125	1
200 ms	15 bits	50 ms	200	2
500 ms[2]	15 bits	50 ms	500	3
1000 ms[2]	15 bits	50 ms	1000	4

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.
2. Use 500 ms or 1000 ms conversion time for FBM213.

FBM214 and FBM214b (HART Communication Input Interface Module)

The FBM214 and FBM214b provide analog and digital communications to and from HART compliant field devices. They also support standard 4 to 20mA signals from analog devices.

The FBM214b is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

The FBM214 contains eight group isolated analog input channels and the FBM214b contains eight individually isolated analog input channels. Each of their eight channels accepts a digital HART Frequency-Shift Keying (FSK) signal superimposed on a 4 to 20 mA analog input signal. Each channel provides bi-directional digital communications with a HART compliant field device, and performs analog to digital conversion on the 4 to 20 mA input control signal from that device. Refer to *HART™ Communication Interface Modules User's Guide* (B0400FF).

FBM215 (HART Communication Output Interface Module)

The FBM215 provides analog and digital communications to and from HART compliant field devices. It also supports standard 4 to 20mA signals from analog devices.

The FBM215 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

The FBM215 contains eight channel-isolated output channels. Each of the eight channels accepts a digital HART Frequency-Shift Keying (FSK) signal superimposed on a 4 to 20 mA analog output signal. Each channel provides bi-directional digital communications with a HART compliant actuator, and provides an output signal of 4 to 20 mA to the actuator. Refer to *HART™ Communication Interface Modules User's Guide* (B0400FF).

FBM216 and FBM216b (HART Communication Redundant Input Interface Module)

The FBM216 and FBM216b provide analog and digital communications to and from HART compliant field devices. They also support standard 4 to 20mA signals from analog devices.

The FBM216b is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- ♦ Compact 200 Series FBM version
- ♦ Standard 200 Series FBM version

The FBM216 contains eight group isolated analog input channels and the FBM216b contains eight individually isolated analog input channels. Each of the eight channels accepts a digital HART Frequency-Shift Keying (FSK) signal superimposed on a 4 to 20 mA analog input signal. Each channel provides bi-directional digital communications with a HART compliant field device, and performs analog to digital conversion on the 4 to 20 mA input control signal from that device.

A redundant pair of the modules combine to provide redundancy at the Fieldbus Module (FBM) level. Refer to *HART™ Communication Interface Modules User's Guide* (B0400FF).

FBM217 (32-Channel Group-Isolated Voltage Monitor Inputs)

The 32-Channel Group-Isolated dc Input Module provides 32-channel digital voltage input channels. It performs voltage or contact sensing (that is, on/off state).

The FBM217 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- ♦ Compact 200 Series FBM version
- ♦ Standard 200 Series FBM version

It performs the signal conversion required to interface these digital (that is, on/off state) electrical input signals from the field sensors to the redundant Fieldbus. Group-Isolated means that the inputs are electrically separate module-to-module but not channel-to-channel on the same card.

The module independently connects to the Fieldbus.

The module executes the Digital I/O or Ladder Logic application program. The configurable options for the program are:

Table B-22. FBM217 Options

Option	Range

Input Filter Time	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

Input Range (each channel): Contact open (off) or closed (on).

FBM217 can replace the following 100 Series Main/Expansion FBM pairs as well.

Table B-23. 100 Series Main/Expansion FBMs Replaced By FBM217

100 Series Main FBM	100 Series Expansion FBM	Input Channels	Output Channels
FBM07, FBM08, FBM20 or FBM24 (16CI)	FBM12, FBM13, FBM21 or FBM25 (16CI)	1-32	None

Refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ) for more information on the FBM217 when used to replace 100 Series FBMs.

FBM218 (HART Communication Redundant Output Interface Module)

The FBM218 provides analog and digital communications to and from HART compliant field devices. It also supports standard 4 to 20mA signals from analog devices.

The FBM218 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

The FBM218 contains eight channel-isolated output channels. Each of the eight channels accepts a digital HART Frequency-Shift Keying (FSK) signal superimposed on a 4 to 20 mA analog output signal. Each channel provides bi-directional digital communications with a HART compliant actuator, and provides an output signal of 4 to 20 mA to the actuator.

A redundant pair of the modules combine to provide redundancy at the Fieldbus Module (FBM) level. Refer to *HART™ Communication Interface Modules User's Guide* (B0400FF).

FBM219 (32-Channel Group-Isolated Discrete I/O Module)

The 32-Channel Group-Isolated Discrete Input/output Module provides 24-channel digital input channels and eight digital output channels. It performs voltage or contact sensing (that is, on/off state).

For input functions, all twenty-four input channels are used exclusively for either contact sensing or voltage monitoring. For output functions, all eight output channels are used for dc output switching, either with an internal or external power source.

The module performs the signal conversion required to interface the digital electrical input/output signals from/to the field sensors/actuators to/from the redundant Fieldbus. Group-Isolated means that the inputs are electrically separate module-to-module but not channel-to-channel on the same card.

The module executes the Digital I/O or Ladder Logic application program. The configurable options and their ranges follow.

Table B-24. FBM219 Digital Options

Option	Range

All Applications
Input Filter Time
	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

DI/DO; Ladder Logic Applications Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
DI/DO Applications Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

Input Range (each channel): Contact open (off) or closed (on).

FBM219 can replace the following 100 Series Main/Expansion FBM pairs as well.

Table B-25. 100 Series Main/Expansion FBMs Replaced By FBM219

100 Series Main FBM	100 Series Expansion FBM	Input Channels	Output Channels
FBM07, FBM08, FBM20 or FBM24 (16CI)	FBM14, FBM15, FBM16, FBM27 or FBM42 (8I/8O)	1-24	25-32

Refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ) for more information on the FBM219 when used to replace 100 Series FBMs.

FBM220/221 FOUNDATION fieldbus H1 Communications Interface Module

FBM220/221 provide an interface between FOUNDATION fieldbus field devices and the Foxboro Evo Process Automation System. FBM220 supports one H1 segment and FBM221 provides integration of four H1 channel isolated fieldbus segments. For detailed information, refer to *FOUNDATION fieldbus H1 Communication Interface Modules (FBM220/FBM221) User's Guide* (B0400FD).

FBM222, Redundant PROFIBUS-DP Communication Interface Module

The Redundant PROFIBUS-DP Communication Interface Module (FBM222) provides an interface between the Foxboro Evo Process Automation System and PROFIBUSDP/PA slave devices, including motor drives, I/O modules, and field I/O devices. The FBM222, which can be used in a single or redundant configuration, supports two PROFIBUS links with a maximum of

125 slave devices per link when repeaters are utilized. The FBM222 connects the slave devices to the versatile and robust Control Core Services system via Distributed Control Interface (DCI) blocks. Physical PROFIBUS-DP wiring is in accordance with Electronic Industrial Association (EIA) standard RS-485.

FBM223 (PROFIBUS-DP Communication Interface Module)

The FBM223 provides an interface between Profibus-DP slave devices and the Foxboro Evo Process Automation System. FBM223 supports two Profibus-DP communication buses, which are galvanically isolated from the Control Core Services Ethernet Fieldbus and from each other. For detailed information, refer to *Profibus-DP Communication Interface Module (FBM223) User's Guide* (B0400FE).

FBM224 (Modbus Communication Interface Module)

The Modbus Communication Interface Module integrates third-party devices that have a Modbus interface, into a Foxboro Evo Process Automation System. The FBM224 provides digital communication to and from Modbus slave devices (input/output devices) on Modbus networks.

Devices with which the FBM224 can successfully communicate are those that support the appropriate Modbus function codes from the list in Table B-26 and operate in the Modbus RTU mode. These devices must respond as slave devices to some or all of the subset of commands of the Modbus function codes as listed in Table B-26. Refer to *Modbus Communication Interface Module (FBM224) User's Guide* (B0400FK).

Table B-26. Modbus Function Codes for Modicon Programmable Controllers

Code	Function
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Force Single Coil
6	Preset Single Register
8	Loopback Diagnostic Test
15	Force Multiple Coils
16	Preset Multiple Registers

FBM227 (Contact/dc I/O Interface Module with DPIDA and MDACT Support)

The FBM227 provides:

- four 0 to 10 V dc analog inputs,
- two 0 to 10 V dc analog outputs,
- four digital inputs, and
- two digital outputs.

The analog inputs and outputs are individually isolated. The digital inputs and outputs are isolated in pairs.

It executes either the Analog I/O or Digital I/O application program, and has support for MDACT or DPIDA control.

Table B-27. FBM227 Options

ECB9 Option	Range
Analog Input Conversion Time (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms conversion time, 25 ms update period 2 = 200 ms conversion time, 25 ms update period 3 = 500 ms conversion time, 25 ms update period 4 = 1000 ms conversion time, 25 ms update period 5 = 50 ms conversion time, 25 ms update period
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms
Digital and Analog Output Failsafe Configuration (Hold/Fallback on a per channel basis)	0 = fallback; 1 = hold
Analog Fallback Values (Channels 5-6)	0 to 64000 Raw Counts
Digital Fallback Values (Channels 11-14)	0 or 1

FBM228 (Redundant 4-Channel Interface to FOUNDATION fieldbus)

The FBM228 Redundant 4-Channel Interface to FOUNDATION™ fieldbus is an Intelligent Device Interface that input/outputs engineering units, device tag information and automatic Link Active Scheduling (LAS). It is a single module or optionally redundant module. For more information, refer to *FOUNDATION™ fieldbus User's Guide for the Redundant FBM228 Interface* (B0700BA).

FBM229 (DeviceNet™ Communication Interface Module)

The FBM229 DeviceNet Communication Interface Module provides a reliable, high-capacity interface between DeviceNet devices and the Foxboro Evo Process Automation System. The FBM229 supports a network of up to 64 devices (including the FBM itself, the slave I/O modules and a third-party configuration workstation). For more information, refer to *Implementing a DeviceNet Network on the Foxboro Evo Core Services Applications* (B0750CH).

FBM230/231/232/233 (Field Device System Interface Modules)

The FBM230/231/232/233 Field Device System Interface (FDSI) modules integrate third-party field device protocols into a Foxboro Evo Process Automation System. The FBM230/231 provides digital communications to/from field devices (input/output devices) using RS-232, RS422, or RS-485 communication standard and the FBM232/233 provides digital communications to/from field devices (input/output devices) using Ethernet10/100 Mbps networks.

- The FBM230 has four serial ports that can be independently software configured for RS-232, RS422, or RS-485 communication standard
- The FBM231, used in pairs, provides a redundant version of the FBM230

- The FBM232 has a single 10/100Mbps copper Ethernet connection to field devices
- FBM233, used in pairs, provides a redundant version of the FBM232.

The FBM230/231/232/233 services all field devices and communicates with the devices on a master/slave basis. As a master, the FBM230/231/232/233 initiates each data communication exchange; the field devices can only send messages to the master when requested to do so. For more information, refer to *Field Device System Integrators (FBM230/231/232/233) User's Guide* (B0700AH).

FBM237 (Channel Isolated, Redundant-Ready 0 to 20 mA Outputs)

The Channel Isolated, Redundant-Ready 0 to 20 mA Output Interface Module (FBM237) contains eight 0 to 20 mA dc analog output channels. The module can be used as a non-redundant (single) module or as a redundant pair (two FBM237s). Each channel is galvanically isolated from the other channels and ground.

The FBM237 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

When used as a redundant pair, the modules combine to provide redundancy at the FBM level, with associated field I/O signals wired to one common termination assembly. Each module independently attempts to hold the output(s) at its specified output value(s), and each independently reports its observed value of the inputs. A redundant analog output (AOUTR) block in the control software validates each output in conjunction with information to/from the module.

In the control processor, a redundant analog output function block, AOUTR, is used for each redundant pair of outputs. The AOUTR block handles output writes and initialization logic for the redundant channels. On each execution cycle of the AOUTR block, identical output writes are sent to both FBMs in the pair, fully exercising communication path to the FBMs and the logic circuitry of each FBM.

Each output channel drives an external load and produces a 0 to 20 mA output. Transmitter power from each module is diode OR'd together in the Redundant Adapter to assure redundant power from either module. The microprocessor of each module executes the Analog I/O application program, plus security routines that validate the health of the module.

Configurable options in the modules for output security include Fail Safe Action (Hold/Fallback), Analog Output Fail-safe Fallback Data (on a per channel basis), Fieldbus Fail-safe Enable, and Fieldbus Fail-safe Delay Time. The Analog Output Fail-safe Fallback Data option must be set for 0 mA output. This removes one of the pair of redundant output channels from service for detectable problems such as an FBM not properly receiving output writes, not passing security tests on FBM microprocessor writes to output registers, failure of internal FBM diagnostics, or FBM module watchdog timer time-out. Setting of the Output Fail-safe Configuration (Hold/Fallback) option for 0 mA output also minimizes the possibility of a "fail high" result.

This module executes the Analog I/O application program. The configurable options and their ranges follow.

Table B-28. FBM237 Options

Option	Range
Analog Output: Fail-safe Configuration (Hold/Fallback on a per-channel basis)	0 = fallback; 1 = hold
Fallback Values (Channels 1-8)	0 to 64000 Raw Counts

For accuracy specifications, refer to *FBM237, 0 to 20 mA Output Interface Module (Redundant Ready)* (PSS 31H-2Z37).

FBM238 – Group Isolated Digital I/O Mix Expander Module

The FBM238 contains twenty-four digital inputs (voltage monitor or contact sense) with eight digital output (external sourcing) channels, used for either output switching with an external source only (e.g. to control powering of various external loads), or dc output switching with an internal source only (e.g. to power external solid state relays or other similar devices). All its channels are group isolated from the earth (ground). Its associated termination assemblies (TAs) provide for discrete inputs of under 60 V ac, 120 V ac/125 V dc or 240 V ac. The module performs signal conversion required to interface the electrical input signals from the field sensors to the Module Fieldbus.

Depending on the type of I/O signal required, its TAs contain current limiting devices, high voltage attenuation circuits, optical isolation and external power source connections.

FBM238 can replace the following 100 Series Main/Expansion FBM pairs as well.

Table B-29. 100 Series Main/Expansion FBMs Replaced By FBM238

100 Series Main FBM	100 Series Expansion FBM	Input Channels	Output Channels
FBM09, FBM10, FBM11, FBM26, FBM41 or (8I/8O)	FBM12, FBM13, FBM21 or FBM25 (16CI)	1-8, 17-32	9-16

Refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ) for more information on the FBM238 when used to replace 100 Series FBMs.

Table B-30. FBM238 Digital Options

Option	Range

All Applications
Input Filter Time
	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

DI/DO; Ladder Logic Applications Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold

Table B-30. FBM238 Digital Options (Continued)

Option	Range
DI/DO Applications Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

FBM239 – Group Isolated Digital I/O Mix Expander Module

The FBM239 contains sixteen digital inputs (voltage monitor or contact sense) with sixteen digital output (external sourcing) channels, used for either output switching with an external source only (e.g. to control powering of various external loads), or dc output switching with an internal source only (e.g. to power external solid state relays or other similar devices). All its channels are group isolated from the earth (ground). Its associated termination assemblies (TAs) provide for discrete inputs of under 60 V ac, 120 V ac/125 V dc or 240 V ac. The module performs signal conversion required to interface the electrical input signals from the field sensors to the Module Fieldbus.

Depending on the type of I/O signal required, its TAs contain current limiting devices, high voltage attenuation circuits, optical isolation and external power source connections.

FBM239 can replace the following 100 Series Main/Expansion FBM pairs as well.

Table B-31. 100 Series Main/Expansion FBMs Replaced By FBM219

100 Series Main FBM	100 Series Expansion FBM	Input Channels	Output Channels
FBM09, FBM10, FBM11, FBM26, FBM41 or (8I/8O)	FBM14, FBM15, FBM16, FBM27 or FBM42 (8I/8O)	1-8, 17-24	9-16, 25-32

Refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ) for more information on the FBM239 when used to replace 100 Series FBMs.

Table B-32. FBM239 Digital Options

Option	Range

All Applications
Input Filter Time
	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

DI/DO; Ladder Logic Applications Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
DI/DO Applications Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

FBM240 (Channel Isolated, Redundant with Readback, Discrete Outputs)

The Channel Isolated, Redundant with Readback, Discrete Output Interface Module (FBM240) has eight discrete output channels. Associated termination assemblies (TAs) support discrete outputs capable of switching 10 A at voltages up to 120 V ac, or 5 A at voltages up to 125 V dc, or 5 A at voltages up to 120 V ac.

The FBM240 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

A pair of the modules combine to provide redundancy at the Fieldbus Module (FBM) level, with field I/O wired to one common termination assembly. Each module independently attempts to hold the output(s) at its specified output value(s), and each independently reports its observed value of the readback inputs. A redundant contact output block in the control software validates each output in conjunction with information to/from the module.

In the control processor, a redundant contact output function block, COUTR, is used for each redundant pair of outputs. The COUTR block handles output writes and initialization logic for the redundant channels. On each write of the COUTR block, identical output writes are sent to both modules, fully exercising the Fieldbus and the logic circuitry of each module. You can select a sustained output that follows the block input or a pulsed output with a selectable pulse width. When a failure is detected in one of the modules, its output is driven to the failsafe state and the corresponding channel in the good module automatically continues selecting the proper discrete outputs.

Each output channel drives an externally powered load. Power for each FBM240 module is diode OR'd together in the redundant adapter to assure redundant power. The microprocessor of each module executes the digital output application program, plus security routines that validate the health of the FBM.

The FBM240 has eight internal readback channels to verify that the outputs have changed to the requested state. If a readback differs from the desired output, that output channel is marked BAD.

When the output channel is marked BAD, the CP presents that information to the system for display as a System Management warning alarm and as a control block alarm.

Configurable options in the modules for output security include Fail-Safe Action. This removes one of the pair of redundant output channels from service for detectable problems such as a module not properly receiving output writes or not passing security tests on FBM microprocessor writes to output registers.

The module executes the Digital I/O application program. The configurable options for the program are:

Table B-33. FBM240 Options

Option	Range

Input Filter Time	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.
Input Range (each channel): Contact open (off) or closed (on).

FBM241 (Channel Isolated, Discrete Input/Output)

The Channel Isolated Voltage Monitor/Contact Sense I/O Interface Module (FBM241/FBM241b/FBM241c/FBM241d) functions as an eight-channel dc voltage monitor with eight-channel output switching, or as an eight-channel contact sensor with eight-channel output switching. The Module is available in four distinct types, to provide the following discrete input and output functions:

- ♦ FBM241 – Accepts eight inputs from a dc voltage source, and provides eight dc switching outputs with an external source.
- ♦ FBM241b – Accepts eight inputs from a dc voltage source, and provides eight dc switching outputs with an internal source.
- ♦ FBM241c – Accepts eight contact sensing inputs, and provides eight dc switching outputs with an external source.
- ♦ FBM241d – Accepts eight contact sensing inputs, and provides eight dc switching outputs with an internal source.

The FBM241 and FBM241c are available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- ♦ Compact 200 Series FBM version
- ♦ Standard 200 Series FBM version

For input functions, all eight input channels are used exclusively for either contact sensing or voltage monitoring. For output functions, all eight output channels are used for dc output switching, either with an internal or external power source.

The module performs the signal conversion required to interface the digital electrical input/output signals from/to the field sensors/actuators to/from the redundant Fieldbus.

The module executes the Digital I/O or Ladder Logic application program. The configurable options and their ranges follow.

Table B-34. FBM241 Digital Options

Option	Range

All Applications
Input Filter Time
	
	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

DI/DO; Ladder Logic Applications Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
DI/DO Applications Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM242 (Channel Isolated, Discrete Outputs)

The Channel Isolated Digital Output Interface Module (FBM242) is a discrete output module providing 16 outputs for dc voltage switching with an external power source. Each output is galvanically isolated from other channels and ground. Field Outputs are wired to one common termination assembly.

The FBM242 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

The module performs the signal conversion required to process the discrete outputs from a control processor to electrical output signals compatible with the field actuators.

The module executes the Digital I/O and Sustained/Momentary (S/M) application program with Ladder Logic support, and a Fail-Safe Configuration configurable option for its outputs.

Table B-35. FBM242 Digital Options

Option	Range

All Applications
Input Filter Time
	
	0 = none;
	1 = 4 ms;
	2 = 8 ms;
	3 = 16 ms;
	4 = 32 ms

Table B-35. FBM242 Digital Options (Continued)

Option	Range
DI/DO; Ladder Logic Applications Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
DI/DO Applications Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary configuration is selected, then the Output Interval is configurable in increments of 10 ms with a 50% duty cycle.

FBM243 and FBM243b (Channel Isolated FoxCom Dual Baud Rate Intelligent Device Interface Module)

The FBM243 contains eight individual channels that provide isolated power and FoxCom communication capabilities to an Intelligent Transmitter/Positioner over a single twisted pair of wires. The FBM243b contains four individual dual baud, FoxCom communication channels and four 0 to 20 mA analog outputs for similar field devices. Both modules allows the use of an external power supply to power the IT. The use of an external power supply common to two or more loops requires a Cable Balun Module to maintain communication signal line balance.

The modules provide bidirectional digital communications at 4800 baud rate between the Intelligent Field Device and the system redundant Fieldbus, or provides bidirectional digital communications at 600 baud rate between the field device and the module while allowing a simultaneous 4 to 20 mA analog signal to an emergency shutdown system. The baud rate is determined by the configuration of the field device connected to each channel, independently of the other channels.

The modules are IT hosts, enabling the system to receive digital messages from the transmitter in engineering units. Each FoxCom message is received 10 times per second at 4800 baud, and two times per second at 600 baud and contains:

- Up to three measured variables in IEEE 32-bit floating-point format
- Security information
- Diagnostics
- Message checking.

This information is available to all elements of the system.

Since FoxCom communication is bidirectional, the system can display the output, transmitter temperature (°C and °F), and continuous self-diagnostics. In addition, information that can be displayed or reconfigured from the console or a Hand-Held Terminal (HHT) is as follows:

- Output in engineering units
- Fail-safe status
- Tag number, name, and location
- Device name (letterbug)
- Last calibration date
- Two levels of upload/download capabilities.

FBM244 – 0 to 20 mA I/O Interface Module with HART® Support

The FBM244 contains four 0 to 20 mA galvanically isolated analog input and output channels (eight total). FBM244 supports any mix of standard 4 to 20 mA devices and HART devices.

Each input channel accepts an analog sensor input such as a 4 to 20 mA transmitter or a self-powered 20 mA source. Each output channel drives an external load and produces a 0 to 20 mA output. The module performs the signal conversion required to interface the electrical input/output signals from the field sensors and actuators to the redundant Fieldbus.

For detailed information, refer to *HART Communication Interface Module User's Guide* (B0400FF).

FBM245 – Redundant 0 to 20 mA I/O Interface Module with HART® Support

Each FBM245 in a redundant pair contains four 0 to 20 mA galvanically isolated analog input and output channels (eight total). They support any mix of standard 4 to 20 mA devices and HART devices.

Each input channel accepts an analog sensor input such as a 4 to 20 mA transmitter or a self-powered 20 mA source. Each output channel drives an external load and produces a 0 to 20 mA output. They perform the signal conversion required to interface the electrical input/output signals from the field sensors and actuators to the redundant Fieldbus.

They connect to their TA via a redundant adapter (RH924DU (supersedes P0924DU)).

For detailed information, refer to *HART Communication Interface Module User's Guide* (B0400FF).

FBM246 and FBM246b (Channel Isolated FoxCom Redundant Dual Baud Rate Intelligent Device Interface Module)

FBM246 contains eight individual FoxCom channels. Each FBM246b contains four dual baud, FoxCom communication channels and four 0 to 20 mA analog output channels.

A pair of FBM246s provides a redundant version of the FBM243, and a pair of FBM246bs provides a redundant version of the FBM243b. Both support transmitters and valve positioners. Their FoxCom channels can support any combination of these devices. Individual transmitters or positioners may be in either analog or digital mode.

NOTE

FBM246 is primarily a digital interface and operation of all transmitters in digital mode provides substantially better performance. Analog mode should be used only for special applications (For example, when the measurement is required for an Emergency Shutdown System that requires 4 to 20 mA signals).

The pair of modules combine to provide redundancy at the FBM level. To achieve redundancy, a Redundant Adapter module is placed on the two adjacent baseplate termination cable connectors to provide a single termination cable connection. A single termination cable connects from the Redundant Adapter to the associated termination assembly (TA).

The pair of FBM246s/FBM246bs supports transmitters and positioners, using ECB18 for transmitters and ECB74 for positioners. Control blocks connect to the ECB18 and ECB74 child ECBs in the same way the equivalent non-redundant strategy would be configured using an FBM243, providing a redundant digital communications path to single FoxCom devices. The connected device ECBs normally get their input data from the **active** member of the pair. They use input data from the module in **track** mode, if this was the only good source of data. Output writes to positioners are sent to both modules.

The ECB used with the pair of FBM246s/FBM246bs is ECB38R. The letterbug for the ECB38R is the letterbug of the first member of the pair. This is the letterbug that is displayed in the icon on the System Management Display Handler display.

The modules provide bidirectional communication at 4800 baud between the IT and the system redundant Fieldbus, or provides bidirectional communications at 600 baud while allowing a simultaneous 4 to 20 mA analog signal to an emergency shutdown system. The baud rate is determined by the configuration of the field device connected to each channel independently of the other channels.

The modules are IT hosts, enabling the system to receive digital messages from the transmitter in engineering units. Each FoxCom message is received 10 times per second at 4800 baud, and two times per second at 600 baud and contains:

- Up to three measured variables in IEEE 32-bit floating-point format
- Security information
- Diagnostics
- Message checking.

This information is available to all elements of the system.

Since communication is bidirectional, the system can display the output, transmitter temperature (°C and °F), and continuous self-diagnostics. In addition, information that can be displayed or reconfigured from the console or a Hand-Held Terminal (HHT) is as follows:

- Output in engineering units
- Fail-safe status
- Tag number, name, and location
- Device name (letterbug)
- Last calibration date
- Two levels of upload/download capabilities.

FBM247 – Current/Voltage Analog/Digital/Pulse I/O Configurable Channel Interface Module with HART® Support

The FBM247 contains eight individual channels which can be individually configured for a range of analog, digital and pulse field I/O signals, including the following:

- HART Analog Input (AI)/Analog Output (AO) 4-20 mA
- 0-20 mA AI/AO, non-HART
- 0-10 V and 0-5 V AI, non-HART
- Digital dry contact sense 24 V dc
- Discrete voltage monitor, configurable 0 and 1 thresholds 0-10 V

- NAMUR sensor discrete input
- Signal level according to DIN EN 50227 (NAMUR)[1]:
  - "On" switching threshold of 2.1 mA dc with short circuit detection at > 6 mA
  - "Off" switching threshold 1.2 mA dc with open detection at <0.25 mA
- Pulse count, frequency, acceleration and jerk, contact sense or voltage input
- Discrete Output 24 V, 20 mA current or solid state switch output

Each I/O channel performs the signal conversion required to interface the electrical input/output signals from the field sensors and actuators to the redundant Fieldbus, and is galvanically isolated from other channels and ground.

The FBM247 is available in two versions, as explained in *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA):

- Compact 200 Series FBM version
- Standard 200 Series FBM version

For detailed information, refer to *HART Communication Interface Module User's Guide* (B0400FF) - particularly the "Configuring Blocks for FBM247" section - and *Standard and Compact 200 Series Subsystem User's Guide* (B0400FA).

100 Series Fieldbus Module Types (FBMs)

— NOTE —

For the list of 200 Series FBMs which can replace each of the 100 Series FBMs, refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ).

FBM01 (0 to 20 mA Inputs)

The 0 to 20 mA Input Interface contains eight 20 mA dc analog input channels. Each channel accepts a 2-wire, analog sensor input such as a 4 to 20 mA transmitter or a self-powered 20 mA source.

The module performs the signal conversion required to interface the electrical input signals from the field sensors to the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus.

This module executes the Analog Input application program. The configurable options for this program and their ranges follow.

1. A shield terminal connection (SH) is provided for each I/O point. The shield terminals are connected to the earth at the system power supply.

Table B-36. FBM01 Options

Option	Range
Analog Input Resolution (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period 5 = 50 ms integration time, 25 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

Input Range (each channel): 0 to 20.4 mA dc.

Rated Mean Accuracy: ±0.05% of span.

Conversion Time (software configurable): see Table B-37.

Table B-37. FBM01 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.05	0.125	0.0250	11
0.1	0.25	0.0125	12
0.2	0.50	0.0075	13
0.5	1.00	0.005	14
1.0	2.00	0.005	15

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.
2. Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM02 (Thermocouple/mV Inputs)

The Thermocouple/mV Input Interface contains eight isolated thermocouple input channels, and one isolated reference junction compensation channel (for terminal temperature sensing).

Each channel has a full scale indication on burnout feature and accepts standard thermocouples for various temperature ranges.

The module performs the signal conversion required to interface the electrical input signals from the thermocouples to the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-38. FBM02 Options

Option	Range
Analog Input Resolution (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period 5 = 50 ms integration time, 25 ms update period
Rate of Change Limits (Channels 1-9)	Normalized Raw Counts/100 ms

Input Range: -10 to 70 mV dc.

Reference Junction:

For discrete and direct terminal connections, a 100 ohm platinum RTD is internally provided at the termination cable assembly.

For a plug termination connection, the reference junction connection is provided by the user with a 4-wire 100 ohm platinum RTD (DIN 43760, Class B).

Rated Mean Accuracy: ± 0.025% of span.

Conversion Time (software configurable): see Table B-39.

Table B-39. FBM02 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.05	0.125	0.0250	11
0.1	0.4	0.0125	12
0.2	0.6	0.0075	13
0.5	1.2	0.005	14
1.0	2.1	0.005	15

1. Value settles within a 1% band of steady state for an input step change of 0 to 60 mV.
2. Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM03A ([3-wire] and 03B [4-wire] RTD Inputs)

The RTD Input Interface contains eight Resistance Temperature Detector (RTD) input channels. Each channel accepts either a 2- and 4-wire RTD, or 3-wire RTD sensor input within a 0 to 320 ohm resistance range.

Within the same module, 3-wire RTDs may not be mixed with 2- or 4-wire RTDs.

The module performs the signal conversion required to interface the electrical input signals from the RTD to the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus.

This module executes the Analog Input application program.

The configurable options and their ranges follow.

Table B-40. FBM03 Options

Option	Range
Analog Input Resolution (on a per module basis)	1, 2, 3, 4, 5 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period 5 = 50 ms integration time, 25 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

Input Range (each channel): 0 to 320 ohms.

Sensor Current: 0.25 mA dc.

Rated Mean Accuracy: ±0.025% of span.

Conversion Time (software configurable): see Table B-41.

Table B-41. FBM03 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.05	0.125	0.0250	11
0.1	0.4	0.0125	12
0.2	0.6	0.0075	13
0.5	1.2	0.005	14
1.0	2.4	0.005	15

1. Value settles within a 1% band of steady state for an input step change of 30 to 320 ohms.
2. Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM04 (0 to 20 mA Inputs/Outputs)

The 0 to 20 mA Input/Output Interface contains four 20 mA dc analog input channels and four 20 mA dc analog output channels.

Each input channel accepts an analog sensor input such as a 4 to 20 mA transmitter or a self-powered 20 mA source.

Each output channel drives an external load and produces a 0 to 20 mA output.

The module performs the signal conversion required to interface the electrical input/output signals from/to the field sensors to/from the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus.

This module executes the Analog I/O application program. The configurable options and their ranges follow.

Table B-42. FBM04 Options

Option	Range
Analog Input Resolution (on a per-module basis)	1, 2, 3, 4, 5 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period 5 = 50 ms integration time, 25 ms update period
Analog Output: Fail-safe Configuration (Hold/Fall-back on a per-channel basis)	0 = fallback; 1 = hold
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms

Rated Mean Accuracy: ±0.05% of span.

Conversion Time (software configurable): see Table B-43.

Table B-43. FBM04 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% Of Range)	Resolution Bits
0.05	0.125	0.0250	11
0.1	0.25	0.0125	12
0.2	0.50	0.0075	13
0.5	1.00	0.005	14
1.0	2.00	0.005	15

1. Value settles within a 1% band of steady state for a 10 to 90% input step change.
2. Monotonic; ensures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM05 (0 to 20 mA Inputs/Outputs) for Redundant Applications

The FBM05 provides redundant input capability using a single transmitter. The FBM05 is similar to the FBM04 module, except for the internal current sense resistor, which is absent from the FBM05 inputs.

A special termination block is provided that includes a current sense resistor. This allows two FBM05 modules to monitor the same transmitter.

The input accuracy for the FBM05 is 0.3% of span. This is based on 0.05% for the input module and 0.25% for the external sense resistor. Otherwise, the FBM05 meets the same physical and performance specifications as the FBM04.

The special termination block (TCA) also provides for each of the output points and its corresponding point in the other member of the redundant pair to drive a single process point.

FBM06 (Pulse Inputs, 0 to 20 mA Outputs)

The Pulse Input, 0 to 20 mA Output Interface, contains four configurable pulse input channels and four 20 mA dc analog output channels.

Each input channel accepts a pulse input, with a maximum rate of 12.5 kHz.

The input devices include vortex and turbine meters, solid state or electro-mechanical contacts and other sensors with similar pulse outputs.

Each output channel drives an external load and produces a 0 to 20 mA output.

The module performs the signal conversion required to interface the electrical input/output signals from/to the field sensors to/from the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus.

This module executes the Pulse Input/Analog Output application program.

The configurable options and their ranges follow.

Table B-44. FBM06 Options

Option	Range
Analog Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Analog Output Fail-safe Fallback Data (on a per-channel basis)	0 to 64000 counts

FBM17 (0 to 10 V dc, Contact/dc Inputs/Outputs)

The 0 to 10 V dc, Contact/dc Input/Output Interface Module is a non-expandable main module that provides the following input and output functions for analog and digital field signals.

Analog Signals

INPUTS – Four 0 to 10 V dc channels used collectively for either:

- dc voltage measuring only
- Slidewire (position) sensing only.

OUTPUTS – Two 0 to 10 V ac channels used for driving positioners, controllers or remote indicators.

Digital Signals

INPUTS – 4 channels used collectively for either:

- Contact sensing only

- dc voltage monitoring only.

OUTPUTS – 4 channels used collectively for either:

- dc output switching with an external source only (for example, to control powering of various external loads).
- dc output switching with an internal source only (for example, to power external solid state relays or similar devices).

The module performs the signal conversion required to interface these analog and digital electrical input/output signals from/to the field sensors/actuators to/from the redundant Fieldbus.

The configurable analog options and their ranges follow.

Table B-45. FBM17 Analog Options

Option	Range
Analog Input Resolution (on a per-module basis)	1, 2, 3, 4 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period
Analog Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Rate of Change Limits (Channels 1-4)	Normalized Raw Counts/100 ms

The configurable digital options and their ranges follow.

Table B-46. FBM17 Digital Options

Option	Range
Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Digital Output Fail-safe Fallback Data (on a per-channel basis)	Specifiable per channel

Analog Input Configurable Specifications: See Table B-47.

Table B-47. FBM17 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.1	0.3	0.013	12
0.2	0.5	0.008	13

Table B-47. FBM17 Conversion Time (Continued)

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.5	1.1	0.005	14
1.0	2.1	0.005	15

1. Output value settles within a 1% band of steady state for a 10 to 90% input step change.
2. Monotonic (that is, the signal used for Fieldbus communications increases or remains the same for increasing analog input signals).

Analog Signal Specifications

INPUTS

Voltage Measuring:

- Range (each channel): -0.2 to 10.2 V dc
- Rated Mean Accuracy (each channel): ±0.025% of span.

OUTPUTS (Output Drivers)

- Capacity: 2 independent channels
- Range (each channel): -0.2 to 10.2 V dc
- Current (each channel): 10 mA (maximum)
- Rated Mean Accuracy: ±0.05% of span
- Settling Time: 10 ms (to 1% of final value for 10 to 90% step change)
- Linearity Error: ±0.025% of span
- Resolution: 12 bits.

Digital Signal Specifications

INPUTS

- Capacity: 4 independent channels
- Filter Time: Configurable (4, 8, 16, or 32 ms)
- Contact Sensor
- Range (each channel): Contact open (off) or closed (on).

OUTPUTS

- Output Switch (with external source)
- Output Switch (with internal source).

FBM18 (Intelligent Transmitter Interface)

The Intelligent Transmitter Interface contains eight individual channels, each providing isolated power and communication capabilities to an Intelligent Transmitter over a single pair of wires.

The module provides bidirectional digital communications between the Intelligent Transmitter and the system redundant Fieldbus.

The module is a transmitter host, enabling the system to receive digital messages from the transmitter in engineering units. Each measurement message is received ten times per second and contains:

- Up to three measured variables, that is, primary pressure, static pressure (d/p cell only), and sensor temperature in IEEE 32-Bit Floating Point
- Security information
- Diagnostics
- Message checking.

This information is available to all elements of the system.

Since the communications are bidirectional, the generator can display the output, transmitter serial number, sensor temperature (degrees C and F), and continuous self-diagnostics.

In addition, information that can be displayed or reconfigured from the console and/or a Hand-Held Terminal is:

- Output in percent or engineering units
- Zero and Span
- Elevation or Suppression
- Linear or Square Root Output
- Damping
- Fail-safe
- Tag Number, Name, and Location
- Last calibration date
- Two levels of upload/download capabilities.

This module executes the interface application program.

FBM07/12 (Contact/dc Inputs and Expansion Inputs)

The Contact/dc Input Interface Module (an expandable main module) and its counterpart Contact/dc Expansion Input Interface Module individually function as a 16-channel contact sensor or 16-channel dc voltage monitor.

Each module performs the signal conversion required to interface these digital (that is, on/off state) electrical input signals from the field sensors to the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module.

The expansion module connects to the Fieldbus via any appropriate main module and is functionally dependent on the supporting main module.

The main module is capable of executing any one of the application programs identified in the following schedule:

- When used alone or in conjunction with an expansion module that interfaces field input signals only, the main module executes the Digital I/O, or Sequence of Events Monitor, or Pulse Count Inputs, or Ladder Logic program.
- When used in conjunction with an expansion module that interfaces field input and output signals, the main module executes either the Digital I/O or Ladder Logic program.

The configurable options for each program are:

Table B-48. FBM07/12 Options

Option	Range
(All applications)	
Inputs: Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	*
(DI/DO; Ladder Logic applications)	
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
(DI/DO Applications)	
Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.
Input Range (each channel): Contact open (off) or closed (on).

NOTE

Subject to prevalent wiring rules on mixing field wiring, FBM Main modules 7, 8, 9, 10, 11, 20, 24, 26, and 41 can mix with expansion modules 12, 13, 14, 15, 16, 21, and 42.

FBM08/13 (120 V ac Inputs and Expansion Inputs)

The 120 V ac Input Interface Module (an expandable main module) and its counterpart 120 V ac Expansion Input Interface Module each function as a 16-channel 120 volt ac monitor.

Each module performs the signal conversion required to interface these digital (that is, on/off state) electrical input signals from the field sensors to the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module. The expansion module connects to the Fieldbus via any expandable main module and is functionally dependent on the supporting main module.

The main module is capable of executing any one of the application programs identified in the following schedule:

- When used alone or in conjunction with an expansion module that interfaces field input signals only, the main module executes either the Digital I/O, Sequence of Events Monitor, Pulse Count Inputs or Ladder Logic program.
- When used in conjunction with an expansion module that interfaces field input and output signals, the main module executes either the Digital I/O or Ladder Logic program.

The configurable options for each program are:

Table B-49. FBM08/13 Options

Option	Range
(All applications)	
Main Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Digital Output: Fail-safe Fallback Data (on a per-channel basis)	Specifiable per channel
Sustained or Momentary Output configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM09/14 (Contact/dc I/O and Expansion I/O)

The Contact/dc Input/Output Interface Module (an expandable main module) and its counterpart Contact/dc Expansion Input/Output Interface Module each provide the following I/O functions:

INPUTS – 8 channels used collectively for either:

- Contact sensing only
- dc voltage monitoring only.

OUTPUTS – 8 channels used collectively for either:

- dc output switching with an external source only (for example, to control powering of various external loads)
- dc output switching with an internal source only (for example, to power external solid state relays or other similar devices).

Each module performs the signal conversion required to interface these digital electrical input/output signals from/to the field sensors or actuators to/from the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module.

The expansion module connects to the Fieldbus via any expandable main module and is functionally dependent on the main module.

When used alone or in conjunction with any expansion module, the main module executes either the Digital I/O or Ladder Logic application program.

The configurable options for each program are:

Table B-50. FBM09/14 Options

Option	Range
(All applications)	
Main Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM10/15 (120 V ac I/O and Expansion I/O)

The 120 V ac Input/Output Interface Module (an expandable main module) and its counterpart 120 V ac Expansion Input/Output Module each provide the following functions:

- 8 input channels for 120 Volt ac voltage monitoring
- 8 output channels for 120 Volt ac output switching with current overload protection.

Each module performs the signal conversion required to interface these digital (that is, on/off state) electrical input/output signals from/to the field sensors or actuators to/from the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module.

The expansion module connects to the Fieldbus via any expandable main module and is functionally dependent on the supporting module.

When used alone or in conjunction with any expansion module, the main module executes either the Digital I/O or Ladder Logic application program.

The configurable options for each program are specified in Table B-51.

Table B-51. FBM10/15 Options

Option	Range
(All applications)	
Main Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Digital Output: Fail-safe Fallback Data (on a per-channel basis)	Specifiable per channel
(DI/DO applications)	
Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM11/16 (240 V ac I/O and Expansion I/O)

The 240 V ac Input/Output Interface Module (an expandable main module) and its counterpart 240 V ac Expansion Input/Output Module each provide the following functions:

- ♦ 8 input channels for 240 Volt ac voltage monitoring
- ♦ 8 output channels for 240 Volt ac output switching with current overload protection.

Each module performs the signal conversion required to interface these digital (that is, on/off state) electrical input/output signals from/to the field sensors or actuators to/from the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module.

The expansion module connects to the Fieldbus via any expandable main module and is functionally dependent on the supporting main module.

When used alone or in conjunction with any expansion module, the main module executes either the Digital I/O or Ladder Logic application program.

The configurable options for each program are:

Table B-52. FBM11/16 Options

Option	Range
(All applications)	
Main Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Digital Output: Fail-safe Fallback Data (on a per-channel basis)	Specifiable per channel
(DI/DO application)	
Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM20/21 (240 V ac Inputs and Expansion Inputs)

The 240 V ac Input Interface Module (an expandable main module) and its counterpart 240 V ac Expansion Input Interface Module each function as a 16-channel 240 Volt ac monitor.

Each module performs the signal conversion required to interface these digital (that is, on/off state) electrical input signals from the field sensors to the redundant Fieldbus.

The expandable main module independently connects to the Fieldbus and is capable of supporting a single expansion module.

The expansion module connects to the Fieldbus via any expandable main module and is functionally dependent on the supporting main module.

The main module is capable of executing any one of the application programs identified in the following schedule:

- When used alone or in conjunction with an expansion module that interfaces field input signals only, the main module executes either the Digital I/O, Sequence of Events Monitor, Pulse Count Inputs, or Ladder Logic program.
- When used in conjunction with an expansion module that interfaces field input and output signals, the main module executes either the Digital I/O or Ladder Logic program.

The configurable options for each program are:

Table B-53. FBM20/21 Options

Option	Range
(All applications)	
Main Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Expansion Input Filter Time	0 = 4 ms; 1 = 8 ms; 2 = 16 ms; 3 = 32 ms
Digital Output: Fail-safe Configuration Hold/Fallback (on a per-channel basis)	0 = fallback; 1 = hold
Digital Output: Fail-safe Fallback Data (on a per-channel basis)	Specifiable per channel
(DI/DO application)	
Sustained or Momentary Output Configuration (on a per-channel basis)	0 = sustained; 1 = momentary

If the Momentary Output configuration is selected, then the Output Interval is also configurable in increments of 10 ms with a 50% duty cycle.

FBM33A (3-wire RTD Inputs) and 33B (2- and 4-wire RTD Inputs)

The Copper RTD Input Interface contains eight Resistance Temperature Detector (RTD) input channels. Each channel accepts either a 2- and 4-wire copper RTD or a 3-wire copper RTD sensor input within a 0 to 30 ohm resistance range.

Within the same module, 3-wire RTDs may not be mixed with 2- or 4-wire RTDs.

The module performs the signal conversion required to interface the electrical input signals from the RTDs to the redundant Fieldbus.

The module is a non-expandable main type, and independently connects to the redundant Fieldbus.

This module executes the Analog Input application program.

The configurable options and their ranges follow.

Table B-54. FBM33 Options

Option	Range
Analog Input Resolution (on a per module basis)	1, 2, 3, 4, 5 1 = 100 ms integration time, 25 ms update period 2 = 200 ms integration time, 25 ms update period 3 = 500 ms integration time, 25 ms update period 4 = 1000 ms integration time, 25 ms update period 5 = 50 ms integration time, 25 ms update period
Rate of Change Limits (Channels 1-8)	Normalized Raw Counts/100 ms

Input Range (each channel): 0 to 30 ohms.

Sensor Current: 0.25 mA dc.

Rated Mean Accuracy: ±0.125% of span (±0.04 ohms).

Conversion Time (software configurable): see Table B-55.

Table B-55. FBM33 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.05	0.2	0.125	11
0.1	0.4	0.0625	12
0.2	0.6	0.0375	13
0.5	1.2	0.025	14
1.0	2.4	0.025	15

1. Value settles within a 1% band of steady state for an input step change of 3 to 30 ohms.
2. Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM36 (Thermocouple/mV Inputs)

The Thermocouple/mV Input Interface contains eight isolated thermocouple input channels, and one isolated RTD reference junction compensation channel (for terminal temperature sensing).

Each thermocouple/mV channel has a full scale indication on burnout feature and accepts Type R thermocouples.

The module performs the signal conversion required to interface the electrical input signals from the thermocouples to the redundant Fieldbus.

The module is a main type, and independently connects to the redundant Fieldbus. This module executes the Analog Input application program. The configurable options and their ranges follow.

Table B-56. FBM36 Options

Option	Range
Analog Input Resolution (on a per-module basis)	0 to 21.1 mV
Rate of Change Limits (Channels 1-9)	Normalized Raw Counts/100 ms

Input Range: 0.1 to 21.1 mV.

Reference Junction:

For discrete and direct terminal connections, a 100 ohm platinum RTD is internally provided at the termination cable assembly.

For a plug termination connection, the reference junction connection is provided by the user with a 4-wire 100 ohm platinum RTD (DIN 43760, Class B).

Rated Mean Accuracy: ± 0.025% of span.

Conversion Time (software configurable): see Table B-57.

Table B-57. FBM36 Conversion Time

Conversion Time (Seconds)	Settling Time[1] (Seconds)	Linearity Error[2] (% of Range)	Resolution Bits
0.05	0.2	0.1	11
0.1	0.4	0.05	12
0.2	0.6	0.03	13
0.5	1.2	0.02	14
1.0	2.1	0.02	15
* Value settles within a 1% band of steady state for an input step change of 10 to 90%. ** Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.			

1. Value settles within a 1% band of steady state for an input step change of 10 to 90%.
2. Monotonic; assures that the signal for Fieldbus communications either increases or remains the same for increasing analog input signals.

FBM37 (0 to 20 mA Outputs)

The Redundant 0 to 20 mA Output Interface Module (FBM37) consists of a pair of Fieldbus Modules (two FBM237s), each of which contains eight 0 to 20 mA dc analog output channels. The module can be used as a nonredundant (single) module or as a redundant pair (two) modules. The pair of Fieldbus Modules combine to provide redundancy at the Fieldbus Module level, with associated field output signals wired to two termination assemblies. Each redundant Fieldbus Module independently attempts to hold the output(s) at its specified output value(s). A redundant analog output (AOUTR) block in the control software validates each output in conjunction with information from the Fieldbus Module.

The module performs the signal conversion required to interface the electrical output signals to the field sensors from the redundant Fieldbus.

The module is a main type, and independently connects to the redundant Fieldbus.

This module executes the Analog Output application program. The configurable options and their ranges follow.

Table B-58. FBM37 Options

Option	Range
Analog Output: Fail-safe Configuration (Hold/Fallback on a per-channel basis)	0 = fallback; 1 = hold

Rated Mean Accuracy: ±0.05% of span.

Cluster I/O FBCs

Refer to *Fieldbus Cluster I/O User's Guide* (B0193RB) for hardware specifications of the Fieldbus Cluster modules (FBCs) currently available to Control Core Services users.

Appendix C. FBM – ECB Cross Reference

The following tables present the complete list of all types of 200 Series FBMs, DCS Fieldbus Modules for Westinghouse Process Control WPDF Systems, and DCS Fieldbus Modules for APACS+ Systems. All of these modules and processors interface to control processors.

Legacy FBMs and Equivalents

The following is a list of legacy FBMs and equivalents. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type
- EXP – Expander type
- Notes – Additional ECBs in use on this FBM.

NOTE

For the list of 200 Series FBMs which can replace each of the I/A Series 100 Series FBMs, refer to *100 Series Fieldbus Module Upgrade User's Guide* (B0700BQ).

Table C-1. Legacy FBMs and Equivalents

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM01	0 to 20 mA Input	8A		1	IOM1	1		
BAMM01	Bailey RTD 0-320 Ohms	4A		1	IOM1	1		
BASM01	Bailey 4 to 20 mA, 1 to 5 V dc, 0 to 10 V dc, -10 to +10 Vdc	16A		47,1,1	IOM56	52,1,1		
BASI01	Bailey 4 to 20 mA, 0 to 5 V dc, 0 to 10 V dc, -10 to 10 V dc, 0 to 1 V dc	15A		47,1,1	IOM56	52,1,1		
F1M01A	Fisher™ Series 10 1-5 Vdc, 4-20 mA	8A		1	IOM1	1		
F1M01C	Fisher Series 10 1-5 Vdc, 4-20 mA, ISO	4A		1	IOM1	1		

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
F1M01E	Fisher Series 10 0-10 V dc	8A		1	IOM1	1		
F1M01F	Fisher Series 10 0-10 V dc ISO	4A		1	IOM1	1		
FBM02	Thermocouple/mV Input	8A		1	IOM1	2		
BASM02	Bailey thermocouple, +/- 100 mV	8A		1	IOM1	2		
F1M02	Fisher Series10 -10.2 to 70 mV, 5 kinds of ISO thermocouple	4A		1	IOM1	2		
FBM03	RTD Input 0-320 Ohms	8A		1	IOM1	3		
BASI03	Bailey mV, TC, RTD, +/-10V	16A		47,1,1	IOM55	52,2,2		
BASM03	Bailey RTD platinum, nickel 0-320 Ohms	8A		1	IOM1	3		
BASM33	Bailey RTD copper 0-30 ohms	8A		1	IOM1	3		
F1M03A	Fisher Series 10 RTD -50 to 200 F	4A		1	IOM1	3		
F1M03C	Fisher Series 10 RTD 100 to 500 F	4A		1	IOM1	3		
F1M03D	Fisher Series 10 RTD	4A		1	IOM1	3		
FBM03A	RTD 3 wire (uses FBM03)	8A		1	IOM1	3		
FBM03B	RTD 4 wire (uses FBM03)	8A		1	IOM1	3		
FBM04	0 to 20 mA Input/Output	4A	4A	2,52	IOM2,52	4		PID - ECB52
F1M04A	Fisher Series 10 1-5 V dc		4A	1	IOM2	4		
F1M04B	Fisher Series 10 4-20 mA		4A	1	IOM2	4		
FBM05	Redundant 0 to 20 mA Input/Output	4A	4A	2	IOM2	5		
FBM06	Pulse Input, 0 to 20 mA Output	4P	4A	4	IOM4	6		
BDSM06	Bailey pulse	8P		47,4,4	IOM4	52,6,6		
F1M06	Fisher Series 10 4-30 V dc, dry contacts, or current pulse	4P		4	IOM4	6		3 types
FBM07	Contact/dc input	16D		5	IOM5	7		
BSEM01	Bailey 24 V dc, 48 Vdc, 120 V ac/dc SOE	16D		6	IOM6	7	14	
BDSI07	Bailey 24 V dc, 125 V ac/dc	16D		5	IOM5	7		

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM07A	15 to 130 V dc (uses FBM07)	16D		5-8	IOM5-8	7		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM07B	24 V dc Contact Sense Input (uses FBM07)	16D		5-8	IOM5-8	7		SOE-ECB6, pulse count-ECB7, ladder-ECB8
F1M07	Fisher Series 104-30 V dc, dry contact, 120 V ac	8D		5-8	IOM5-8	7		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM08	120 V ac Input	16D		5-8	IOM5-8			SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM09	Contact/dc Input/Output	8D	8D	5	IOM5	9		
BDSM09	Bailey 24 V dc solid state		16D	5	IOM5	9	14	
BDSM9A	Bailey 24 V dc, 125 V ac/dc/24 V dc	8D	8D	5	IOM5	9		
BDSM9B	Bailey 24 Vdc	16D	16D	5	IOM5	9	14	
F1M09	Fisher Series 10 FET 4-30 V dc, relay or relay external		8D	5	IOM5	9		3 types
FBM09A	Voltage Monitor - 15 to 130 V dc Input, 60 V dc @ 0.5 mV Output	8D	8D	5,8	IOM5,8	9		ladder-ECB8
FBM09B	Voltage Monitor - 15 to 130 V dc Input, 0 to 10 V dc Output	8D	8D	5,8	IOM5,8	9		ladder-ECB8
FBM09C	Contact Sense - 24 V dc Input, 60 V dc @ 0.5 mV Output	8D	8D	5,8	IOM5,8	9		ladder-ECB8
FBM09D	Contact Sense - 24 V dc Input, 0 to 10 V dc Output	8D	8D	5,8	IOM5,8	9		ladder-ECB8
FBM10	120 V ac Input/Output	8D	8D	5,8	IOM5,8			ladder-ECB8
BDSO10	Bailey 24-240 V dc solid state		8D	5,8	IOM5,8	10	14	ladder-ECB8

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM11	240 V ac Input/Output	8D	8D	5,8	IOM5,8	11		ladder-ECB8
FBM12	FBM07 Expander	16D		see above			12	
FBM12A	FBM07A Expander	16D		see above			12	
FBM12B	FBM07B Expander	16D		see above			12	
FBM13	FBM08 Expander	16D		see above			13	
FBM14	FBM09 Expander	8D	8D	see above			14	
FBM14A	FBM09A Expander	8D	8D	see above			14	
FBM14B	FBM09B Expander	8D	8D	see above			14	
FBM14C	FBM09C Expander	8D	8D	see above			14	
FBM14D	FBM09D Expander	8D	8D	see above			14	
FBM15	FBM10 Expander	8D	8D	see above			15	
FBM16	FBM11 Expander	8D	8D	see above			16	
FBM17	0 to 10 V dc, Contact/dc Input/Output	4A, 4D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT-ECB34, MDPulse-ECB36, PID - ECB52
FBM17A	15 to 130 V dc Input, 60 V dc @ 0.5 mV Output	4A, 4D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT-ECB34, MDPulse-ECB36, PID-ECB52
FBM17B	15 to 130 V dc Input, 0 to 10 V dc Output	4A, 4D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT - ECB34, MDPulse - ECB36, PID - ECB52
FBM17C	24 V dc Input, 60 V dc @ 0.5 mV Output	4A, 4D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT-ECB34, MDPulse - ECB36, PID - ECB52

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM17D	24 V dc Input, 0 to 10 V dc Output	4A, 4D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT-ECB34, MDPulse-ECB36, PID - ECB52
BCOM17	0 to 10 V dc, Contact/dc Input/Output	4A, 3D	2A, 4D	9,34, 36,52	IOM9,34, 36,52	17		MDACT - ECB34, MDPulse - ECB36, PID - ECB52
FBM18	Intelligent Transmitter Input/Output	8 In		12	IOM12	18		
FBM19	Analog Input	?A		1	IOM1	19		
FBM20	240 V ac Input	16D		5-8	IOM5-8	20		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM21	FBM20 Expander	16D		see above			21	
FBM22	Single 0 to 20 mA Input/Output Auto/Manual	1A	1A	9,52	IOM9,52	22		PID-ECB52
FBM23 (HIU)	HTG Interface Unit	32D		13	IOM13	23		
FBM24	Contact/125 V dc Input - External Power	16D		5-8	IOM5-8	24		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM24A	24 to 125 V dc Input/Output	16D		5-8	IOM5-8	24		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM24B	Contact Sense Input/Output	16D		5-8	IOM5-8	24		SOE-ECB6, pulse count-ECB7, ladder-ECB8

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM24C	Contact Sense Input/Output - External Power (16th=power connection)	15D		5-8	IOM5-8	24		SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM25	FBM24 Expander	16D		see above			25	
FBM25A	FBM24A Expander	16D		see above			25	
FBM25B	FBM24B Expander	16D		see above			25	
FBM25C	FBM24C Expander	16D		see above			25	
FBM26	Contact/125 V dc Input/ Output - External	8D	8D	5,8	IOM5,8	26		ladder-ECB8
BDSO26	Bailey 4 -50 V dc solid state		8D	5,8	IOM5,8	26	14	ladder-ECB8
FBM26A	24 to 125 V dc Input (uses FBM26)	8D	8D	5,8	IOM5,8	26		ladder-ECB8
FBM26B	Contact Sense Input (uses FBM26)	8D	8D	5,8	IOM5,8	26		ladder-ECB8
FBM26C	Contact Sense Input - External Power (8th=external power)	7D	8D	5,8	IOM5,8	26		ladder-ECB8
FBM27	FBM26 Expander	8D	8D	see above			27	
FBM27A	FBM26A Expander	8D	8D	see above			27	
FBM27B	FBM26B Expander	8D	8D	see above			27	
FBM27C	FBM26C Expander	8D	8D	see above			27	
FBM33	Copper RTD Input, Expanded Range	8A		1	IOM1	33		
FBM33A	Expanded Range Copper RTD 3-wire	8A		1	IOM1	33		
FBM33B	Expanded Range Copper RTD 2- and 4-wire	8A		1	IOM1	33		
FBM36	Type R Thermocouple/mV Input	8A		1	IOM1	36		
FBM37	0-20 mA Output		8A	53	IOM53	37		
BAS037	Bailey 1 to 5 V dc, 4 to 20 mA		14A	47,53, 53	IOM56	52, 37,37		
BAOM37	Bailey 0 to 10 V dc, 1 to 5 V dc, 4 to 20 mA		8A	53	IOM53	37		
FBM38	Coriolis Mass Flow	Coriolis		22	IOM22			

Table C-1. Legacy FBMs and Equivalents (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	EXP	Notes
FBM39	Intelligent Transmitter Inputs/ 0 to 20 mA Out	4 IT	4A	23	IOM23	39		
FBM41	Isolated Contact 24 V dc In/0 to 60 V dc Out	8D	8D	5,8	IOM5,8	41		ladder-ECB8
BDSO41	Bailey 5-160 dc solid state		8D	5,8	IOM5,8	41	14	ladder-ECB8
FBM41A	15 to 60 V dc Input, 0 to 60 V dc @ 2.25 A Output	8D	8D	5,8	IOM5,8	41		ladder-ECB8
FBM41C	24 V dc Contact Sense Input, 0 to 60 V dc @ 2.25 A Output	8D	8D	5,8	IOM5,8	41		ladder-ECB8
FBM42	FBM41 Expander	8D	8D	see above			42	
FBM42A	FBM41A Expander	8D	8D	see above			42	
FBM42C	FBM41C Expander	8D	8D	see above			42	
FBM43	Dual Baud Rate Intelligent Device Interface	8 IT		12,73	IOM37, 73	43		FoxCom - ECB73 (parent of 18, 43), ECB12 (parent of 18)
FBM44	Dual Baud Rate IT/0 to 20 mA Out Redundant Output	4 IT	4A	23,38	IOM23,38	44		multi-baud-ECB38
FBM45	Gas Chromatograph	Gas Chro-mato-graph		39	IOM39	45		
FBM46	Dual Baud Rate IT/0 to 20 mA In/Out Redundant	4 IT	4A	38R	IOM49	98		FoxCom-(parent of 18)
PDISP	Panel Display	-	-	14	IOM14	28		This is not a real FBM, but the I/A Series system treats it as one.

Fieldbus Processors

The following is a list of fieldbus processors. The table heading definitions are as follows:

- Type – FBP Type
- Signal Description – Purpose of FBP
- In – Number of input channels (dependent on I/O cards)
- Out – Number of output channels (dependent on I/O cards)
- SW ECB# – Equipment control block software type used for this FBM
- IOM Number – The software that is downloaded to this FBM
- HWT – Hardware type.

Table C-2. Fieldbus Cluster Modules

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT
FBP10	Eckardt Migration			47	IOM42	52
FBP10R	Eckardt Migration (Redundant)			47R	IOM47	87
FBP11	Spectrum UCM			48	IOM43	53
FBP11R	Spectrum UCM (Redundant)			48R	IOM48	88
FBP12	Spectrum FIO™	N/A	N/A	44	IOM44	54
FBP13	Spectrum UFM™	N/A	N/A	45	IOM45	55
FBP14	Spectrum UIO			46	IOM46	56

Eckardt I/O Cards

The following is a list of I/O cards that are found in Eckardt racks.The table heading definitions are as follows:

- Type – FBP Type
- Signal Description – Purpose of FBP
- In – Number of input channels (dependent on I/O cards)
- Out – Number of output channels (dependent on I/O cards)
- SW ECB# – Equipment control block software type used for this FBM
- HWT – Hardware type.

Table C-3. Eckardt Rack I/O Cards

Type	Signal Description	In	Out	SW ECB#	HWT
FBC01	0 to 20 mA Inputs	32A		41	58
FBC01R	0 to 20 mA (Redundant)	32A		41	90
FBC02	Thermocouple Inputs	32A		41	63
FBC02R	Thermocouple (Redundant)	32A		41	95
FBC04	0 to 20 mA Outputs		16A	43	61
FBC04R	0 to 20 mA (Redundant)		16A	43	93
FBC07A/B	Contact/dc Inputs	32D		42	60
FBC07R	Contact/dc (Redundant)	32D		42	92
FBC09	Contact/dc Outputs		32D	44	62
FBC09R	Contact/dc (Redundant)		32D	44	94
FBC10	Contact/dc Input/Output	32D	32D	46	86
FBC10R	Contact/dc (Redundant)	32D	32D	46	97
FBC17	0 to 10 V dc Inputs	32A		41	59
FBC17R	0 to 10 V dc (Redundant)	32A		41	91
FBC21	0 to 20 mA Inputs	16A		41	57
FBC21R	0 to 20 mA (Redundant)	16A		41	89

UCM/UIO

The following is a list of I/O cards that can be found in either UCM or UIO racks. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- HWT – Hardware type.

Table C-4. UCM or UIO Rack I/O Cards

Type	Signal Description	In	Out	SW ECB#	HWT
3A2-V2D	High Level ISO	2A		41	65
3A2-V3D	High Level	2A		41	65
3A2-E2D	ac Voltage input	2A		41	65
3A2-I2D	ISO Current Input	2A		41	65
3A2-I2DA	ISO High Level	2A		41	65
3A2-I3D	Current Input	2A		41	65
3A2-I3DA	High Level	2A		41	65
3A2-H3D	High Level	2A		41	65
3A4-I2D	Quad Current In	4A		41	66
3A2-M2D	MV/TC Adj Range	2A		41	65
3A2-T2DJ1	Thermocouple Input	2A		41	68
3A2-T2DJ2	Thermocouple Input	2A		41	68
3A2-T2DK1	Thermocouple Input	2A		41	68
3A2-T2DK2	Thermocouple Input	2A		41	68
3A4-M2DA1	Millivolt/TC Input	2A		41	69
3A4-M2DA2	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A4-M2DA3	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A4-M2DA4	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A4-M2DA5	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A4-M2DA6	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A4-M2DA7	Millivolt/TC Input (4 channel UCM ONLY)	4A		41	69
3A2-R2DC	Copper RTD	2A		41	71
3A2-R2DN	Nickel RTD	2A		41	71
3A2-R2DP	Platinum 1 RTD	2A		41	71
3A2-R2DP	Platinum 2 RTD	2A		41	71
3A2-R2DP	Platinum 3 RTD	2A		41	71
3A2-R2DP2	Platinum RTD	2A		41	71
3C8-C3D	Octal Contact In	8D		42	77
3C8-E2D	Octal Line V Monitor	8D		42	77
3A2-F2D	Pulse Rate Counter	2P		41	75
3A2-Q2D	Pulse Input Count	2P		41	73
3A2-D3V	Dual Voltage Out		2A	43	78
3A2-D2I	Dual Current Out		2A	43	78
3A2-D3I	Dual Current Out		2A	43	78
3C4-D2CS	dc Switch Out		4D	44	80
3C4-D2KS	ac Switch Out		4D	44	80
3C8-D2CS	Octal Contact Out		8D	44	81
3C4-D2VS	dc Volt Out		4D	44	80
3C4-D2CP	dc Pulse Out		4D	44	82
3C4-D2KP	ac Pulse Out		4P	44	82
3C4-D2VP	dc Pulse Out		4P	44	82
3AS-I2I	Single Loop I/O	1A	1A	45	84
3AS-I3I	Single Loop I/O	1A	2A	45	84
3AD-I3I	Dual Loop I/O (4 channel UCM ONLY)	2A	2A	45	85

FIO

The following is a list of I/O cards that can be found in FIO racks. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- HWT – Hardware type.

Table C-5. FIO Rack I/O Cards

Type	Signal Description	In	Out	SW ECB#	HWT
3F8-V2DA1	Octal High Level	8A		41	67
3F8-V2DA2	Octal High Level	8A		41	67
3F8-V2DA	Octal High Level	8A		41	67
3F4-I2D1A	Quad 0-20 mA dc, 16V	4A		41	66
3F4-I2D2A	Quad 0-20 mA dc, 22V	4A		41	66
3F8-I2DA	Octal High Level	8A		41	67
3F8-H2DA	Octal High Level	8A		41	67
3F8-T2DA1	Octal Thermocouple	8A		41	70
3F8-T2DA2	Octal Thermocouple	8A		41	70
3F8-T2DA3	Octal Thermocouple	8A		41	70
3F8-T2DA4	Octal Thermocouple	8A		41	70
3F8-T2DA5	Octal Thermocouple	8A		41	70
3F8-T2DA6	Octal Thermocouple	8A		41	70
3F8-T2DA7	Octal Thermocouple	8A		41	70
3F8-M2DA1	Octal Millivolt	8A		41	67
3F8-M2DA2	Octal Millivolt	8A		41	67
3F8-M2DA3	Octal Millivolt	8A		41	67
3F8-M2DA4	Octal Millivolt	8A		41	67
3F8-M2DA5	Octal Millivolt	8A		41	67
3F8-M2DA6	Octal Millivolt	8A		41	67
3F8-M2DA7	Octal Millivolt	8A		41	67
3F8-R2DCA	Octal RTD (Copper)	8A		41	72
3F8-R2DNA	Octal RTD (Nickel)	8A		41	72
3F8-R2DPA1	Octal RTD (Platinum 1)	8A		41	72
3F8-R2DPA2	Octal RTD (Platinum 2)	8A		41	72
3F8-R2DPA3	Octal RTD (Platinum 3)	8A		41	72
3F8-C2DCA	Octal Contact	8D		42	77
3F8-C2DNA	Octal Prox Sensor	8D		42	77
3F8-E2DA	Octal HL Digital	8D		42	77
3F4-F2DA	Quad Pulse Rate	4P		41	76
3F4-Q2DA	Quad Pulse Counter	4P		41	74
3F4-D2VA	Quad High Level Out		4A	43	79

Table C-5. FIO Rack I/O Cards (Continued)

Type	Signal Description	In	Out	SW ECB#	HWT
3F4-D2IA	Quad High Level Out		4A	43	79
3F8-D2CSA	Octal dc Switch		8D	44	81
3F8-D2ZA	Octal Solenoid Dvr		8D	44	81
3F8-D2KSA	Octal ac Switch		8D	44	81
3F4-D2WA	Quad Solenoid Dvr		4D	44	80
3F8-D2CPA	Octal dc Pulse Sw		8P	44	83
3F8-D2KPA	Octal ac Pulse Sw		8P	44	83

UFM

The following is a list of I/O cards that can be found in UFM racks. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type.

Table C-6. UFM Rack I/O Cards

Type	Signal Description	In	Out	SW ECB#	HWT
3A8-V2D1	Octal High Level			41	67
3A8-I2D1	Octal High Level			41	67
3A8-M2D1	Octal Millivolt			41	67
3A8-M2D2	Octal Millivolt			41	67
3A8-M2D3	Octal Millivolt			41	67
3A8-T2D1	Octal Thermocouple			41	70
3A8-T2D2	Octal Thermocouple			41	70
3A8-T2D3	Octal Thermocouple			41	70
3A8-R2DC1	Octal RTD (Copper)			41	72
3A8-R2DN1	Octal RTD (Nickel)			41	72
3A8-R2DP1	Octal RTD (Platinum 1)			41	72
3A8-R2DP2	Octal RTD (Platinum 2)			41	72
3A8-R2DP3	Octal RTD (Platinum 3)			41	72
3D8-C2D1	Octal Contact			42	77

Fisher Series 20 Migration Cards

The following is a list of Fisher Series 20 migration cards. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type
- Notes – Additional ECBs in use on this FBM.

Table C-7. Fisher Series 20 Migration Cards

Type	Signal Description	In	Out	SW ECB# [1]	IOM#	HWT	Notes
F2M214	Two FBM214 modules	8A x 2 (16A Total)	-	200, 201	IOM214 [1]	214	Configured as two FBM214 modules, each with 8 analog HART inputs
F2M215	One FBM215 module	-	8A[2]	200, 201	IOM215 [1]	215	
F2M239	One FBM239 module	Up to 16D	Up to 16D	5	IOM79 [1]	239	16 I/O channels - Jumper for either input or output only
F2M67A (Legacy)	Fisher Series 20 FBP10, FBM09 (Replaced by F2M239)	Up to 16D	Up to 16D	47,5	IOM59	52,9	16 I/O channels - Jumper for either input or output only
F2M68A (Legacy)	Fisher Series 20 FBP10, 2 FBM01s, FBM37 (Analog inputs replaced by F2M214, analog outputs replaced by F2M215)	16A	8A	47,1, 53	IOM58	52,1,37	Inputs and Outputs are independent and both may be used
FRM701	Fisher Controller Configurable FBP10, 1 or 2 FBM17s	4A, 2D	1A, 7D	47,9	IOM57	52,17	
FRM711	Fisher Controller Computing FBP10, 1 or 2 FBM17s	5A, 4D	2A, 2D	47,9	IOM57	52,17	
FRMMPU	Fisher Controller FBP10, 2 FBM17, FBM01	10A, 4D	3A, 4D	47,9,1	IOM57	52,17,1	

1. Refer to "DCS Fieldbus Module Control Schemes" in *DCS Fieldbus Modules for Fisher PROVOX® Series 20 Systems User's Guide* (B0193YV) for details on how to configure the blocks for these devices.
2. F2M215 has HART output channels.

DIN Rail Mounted 200 Series FBMs and Equivalents

DIN Rail Mounted 200 Series FBMs

Table C-8 lists the 200 Series (DIN rail mounted) FBMs. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type
- Notes – Additional ECBs in use on this FBM.

Table C-8. 200 Series (DIN Rail Mounted) FBMs

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
FBM201/ FBM201e	8 Input, 0-20 mA	8 AI		1	IOM83	201	
FBM201b	8 Input, 0-100mV	8 AI		1	IOM83	201	
FBM201c	8 Input, 0-5 V	8 AI		1	IOM83	201	
FBM201d	8 Input 0-10 V	8 AI		1	IOM83	201	
FBM202	8 Input, Thermocouple/mV	8 AI		1	IOM84	202	
FBM203	8 Input, RTD (Pt, Ni)	8 AI		1	IOM85	203	
FBM203b	8 Input, RTD (Pt,Ni) Extended Range	8 AI		1	IOM85	203	
FBM203c	8 Input, RTD (Cu)	8 AI		1	IOM85	203	
FBM203d	8 Input 4 wire RTD (Pt, Ni, Cu)	8 AI		1	IOM85	203	
FBM204	4 Input, 0-20 mA / 4 Output, 0-20 mA	4 AI	4 AO	2	IOM86	204	
FBM204b	4 Input, 0-20 mA / 4 Output, 0-20 mA (Readback)	4 AI	4 AO	2	IOM86	204	
FBM205	4 Input, 0-20 mA / 4 Output, 0-20 mA (Redundant capable)	4 AI	4 AO	2	IOM87	205	
FBM206	8 Input, Pulse	8 PI		4	IOM88	206	
FBM206b	4 Input, Pulse / 4 Output, 0-20 mA	4 PI	4 AO	4	IOM88	206	
FBM207	16 Input, Voltage Monitor	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8

Table C-8. 200 Series (DIN Rail Mounted) FBMs (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
FBM207b	16 Input, 24 V dc Contact Sense	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM207c	16 Input, 48 V dc Contact Sense	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM208	0 to 20 mA Redundant with readback	4 AI	4 AO	2	IOM80	208	
FBM208b	0 to 20 mA Redundant with readback	4 AI	4 AO	2	IOM80	208	Replaces FBM05 to non-HART devices - no TA
FBM211	16 Input, 0-20 mA Differential	16 AI		1	IOM90	211	
FBM212	16 Input, Thermocouple/mV Differential	14 AI		1	IOM91	212	
FBM213	8 Input, RTD (Pt, Ni) Differential	8 AI		1	IOM92	213	
FBM214	8 Communication, HART Input	8 AI or HART		200/ 201	IOM214	214	
FBM214b	8 Communication, HART Input	8 AI or HART		200/ 201	IOM214	214	
FBM215	8 Communication, HART Output		8 AO or HART	200/ 201	IOM215	215	
FBM216	8 Communication, HART Input (Redundant)	8 AI or HART		201/ 202	IOM216	216	
FBM216b	8 Communication, HART Input (Redundant)	8 AI or HART		201/ 202	IOM216	216	
FBM217	32 Input, Voltage Monitor (Group Isolated)	32 DI		5-8	IOM96	217	SOE-ECB6, pulse count-ECB7, ladder-ECB8
FBM218	8 Communication, HART Output (Redundant)		8 AO or HART	201/ 202	IOM218	218	
FBM219	24 Input, Voltage Monitor/8 Output, Switch (External Source)	24 DI	8 DO	5,8	IOM98	219	ladder-ECB8
FBM220	1 Communication Port, Fieldbus Foundation	1 FF	1 FF	200/ 201	IOM220	220	
FBM221	4 Communication Ports, Fieldbus Foundation	4 FF	4 FF	200/ 201	IOM221	221	
FBM222	2 Communication Ports, Profibus DP	2 Profibus	2 Profibus	200/ 202	IOM222	222	Redundant modules use ECB 202
FBM223	2 Communication Ports, Profibus DP	2 Profibus	2 Profibus	200/ 201	IOM223	223	
FBM224	4 Communication Ports, Modbus (Redundant capable)	4 Modbus	4 Modbus	200/ 201	IOM224	224	2000 pts/ 64 devices

Table C-8. 200 Series (DIN Rail Mounted) FBMs (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
FBM227	4 0 to 10 V dc and 4 Discrete Input 2 0 to 10 V dc and 4 Discrete Output	4 0-10V dc and 4 Discrete	2 0-10V dc and 4 Discrete	9	IOM77	227	
FBM228	4 Communication Ports, Fieldbus Foundation (Single or Redundant)	4 FF	4 FF	200/ 201/ 202	IOM228	228	Redundant modules use ECBs 202/201
FBM229	1-Channel Interface to DeviceNet Devices	Supports I/O for a network of up to 64 DeviceNet devices (including the FBM itself, the slave I/O modules and a third-party configuration workstation)		200/ 201	IOM229	229	
FBM230	4 Serial Ports, RS-232, RS-422, or RS-485	4 Serial	4 Serial	200/ 201	IOM230	230	
FBM231	4 Serial Ports, RS-232, RS-422, or RS-485 (Redundant)	4 Serial	4 Serial	202/ 201	IOM231	231	
FBM232	1 Ethernet Port	1 Ethernet	1 Ethernet	200/ 201	IOM232	232	
FBM233	1 Ethernet Port (Redundant)	1 Ethernet	1 Ethernet	202/ 201	IOM233	233	
FBM237	8 Output, 0-20 mA		8 AO	53	IOM93	237	
FBM238	24 Input, Voltage Monitor (Group Isolated) and 8 Discrete Output	24 DI	8 Discrete	5-8	IOM78	238	
FBM239	16 Input, Voltage Monitor (Group Isolated) and 16 Discrete Output	16 DI	16 Discrete	5-8	IOM79	239	
FBM240	8 Output, Switched 120v ac or 125 V dc, with Readback		8 DO	5	IOM81	240	
FBM241	8 Input, Voltage Monitor/8 Output, Switch (External Source)	8 DI	8 DO	5,8	IOM94	241	ladder-ECB8
FBM241b	8 Input, Voltage Monitor/8 Output, Switch (Internal Source)	8 DI	8 DO	5,8	IOM94	241	ladder-ECB8
FBM241c	8 Input, Contact Sense/8 Output, Switch (External Source)	8 DI	8 DO	5,8	IOM94	241	ladder-ECB8
FBM241d	8 Input, Contact Sense/8 Output, Switch (Internal Source)	8 DI	8 DO	5,8	IOM94	241	ladder-ECB8
FBM242	16 Output, DC Switch (External Source)		16 DO	5,8	IOM95	242	ladder-ECB8
FBM243	8 Channel Isolated, dual baud rate FoxCom (Inteligent Device)	8 FoxCom	8 FoxCom	73/18 or 74	IOM97	243	

Table C-8. 200 Series (DIN Rail Mounted) FBMs (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
FBM243b	4 Channel Isolated, dual baud rate FoxCom (Intelligent Device) and 4 Output, 0-20 mA	4 FoxCom	4 AO	23	IOM97	243	
FBM244	4 Input, 0-20 mA, 4 Output, 0-20 mA (HART Support)	4 AI	4 AO	200/ 201	IOM244	244	
FBM245	4 Input, 0-20 mA, 4 Output, 0-20 mA Redundant (HART Support)	4 AI	4 AO	202/ 201	IOM245	245	
FBM246	4 Redundant Channel Isolated, dual baud rate FoxCom (Intelligent Device)	4 FoxCom	4 Fox-Com	38R/18 or 74	IOM97	246	
FBM246b	4 Redundant Channel Isolated, dual baud rate FoxCom (Intelligent Device) and 4 Output, 0-20 mA	4 FoxCom	4 AO	38R/18	IOM97	246	
FBM247	8-Channel Current/Voltage Analog/Digital/Pulse I/O Configurable Channel Interface Module (with HART® Support on All Channels) - Includes support for additional communication types	8 Configurable I/O Channels		200/ 201	IOM247	247	Eight channels may be configured as input or output for a variety of channel types, with or without HART support - see page 244

Intrinsically Safe I/O Subsystem (ISCM) Cards

Table C-9 lists the appropriate Equipment Control Blocks (ECBs) for use in conjunction with the ISCM cards.

Table C-9. Intrinsically Safe I/O Subsystem (ISCM) Cards

P+F Model No.	Description	SW ECB	HWT	Software Type	I/O Block
ISCM8100	Intrinsically Safe Communication Module for Zone 2 applications (LB-style), nonredundant	200	250	250	none
ISCM8200	Intrinsically Safe Communication Module for Zone 1 applications (FB-style), nonredundant	200	250	250	none
ISCM8100	Intrinsically Safe Communication Module for Zone 2 applications (LB-style), redundant	202	250	250	none
ISCM8200	Intrinsically Safe Communication Module for Zone 1 applications (FB-style), redundant	202	250	250	none

For the ISCM and available P+F I/O modules and base/extension units which can be used with the Foxboro Evo Process Automation System, refer to *Intrinsically Safe I/O Subsystem User's Guide* (B0700DP). The Foxboro Evo Process Automation System views the P+F I/O modules as 200 Series FBMs.

DCS FBMs for ABB/Taylor MOD300 Systems

Table C-10 lists the DCS FBMs for migration of ABB/Taylor MOD300 Systems. Each DCS FBM is electrically equivalent to a 200 Series FBM, as indicated by the software type (IOM#) and hardware type (HWT).

Table C-10. ABB/Taylor MOD300 Systems Migration Cards

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT
DIO214	8 Communication, HART Input	8 AI or HART		200/ 201	IOM214	214
DIO215	8 Communication, HART Output		8 AO or HART	200/ 201	IOM215	215
DIO216	8 Communication, HART Input (Redundant)	8 AI or HART		201/ 202	IOM216	216
DIO218	8 Communication, HART Output (Redundant)		8 AO or HART	201/ 202	IOM218	218
DIOSDM	MOD300 digital migration	Up to 48D	Up to 48D,P	200/ 201	IOM215	236
DIOSDM	MOD300 digital migration (redundant)	Up to 48D	Up to 48D,P	200/ 202	IOM215	236 (2 sets)

Refer to *DCS Fieldbus Modules for ABB MOD300 Direct I/O Systems with HART I/O Capability User's Guide* (B0700AE) for additional information on configuring these DCS FBMs.

DCS FBMs for Westinghouse Process Control WPDF Systems

Table C-11 lists the DCS FBMs for migration of Westinghouse Process Control WPDF Systems. Each DCS FBM is electrically equivalent to a 200 Series FBM, as indicated by the software type (IOM#) and hardware type (HWT).

Table C-11. DCS FBMs for Westinghouse Process Control WPDF Systems

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
WAI01A	Westinghouse -512 to +512 mV dc	4 AI		1	IOM83	201	
WAI01B	Westinghouse -1.02 to +1.02 V dc	4 AI		1	IOM83	201	
WAI01C	Westinghouse -10.24 to +10.24 V dc	4 AI		1	IOM83	201	
WAI01D	Westinghouse 0 to 20.48 mA	4 AI		1	IOM83	201	

Table C-11. DCS FBMs for Westinghouse Process Control WPDF Systems (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
WAW01A	Westinghouse 0 to 1.02 V dc	6 AI		1	IOM83	201	
WAW01B	Westinghouse 0 to 5.12 V dc	6 AI		1	IOM83	201	
WAW01C	Westinghouse 0 to 10.24 V dc	6 AI		1	IOM83	201	
WAW01D	Westinghouse 0 to 20.48 mA	6 AI		1	IOM83	201	
WAW01E	Westinghouse 0 to 20.48 mA	6 AI		1	IOM83	201	
WAW01F	Westinghouse 0 to 51.2 mA	6 AI		1	IOM83	201	
WAX01A	Westinghouse 0 to 1.02 V dc	12 AI		1	IOM83	(2) 201	
WAX01B	Westinghouse 0 to 5.12 V dc	12 AI		1	IOM83	(2) 201	
WAX01C	Westinghouse 0 to 10.24 V dc	12 AI		1	IOM83	(2) 201	
WAI02A	Westinghouse -100 to +100 mV thermo	4 AI		1	IOM84	202	
WAV02A	Westinghouse -100 to +100 mV thermo	6 AI		1	IOM84	202	
WAX02A	Westinghouse -100 to +100 mV thermo	12 AI		1	IOM84	(2) 202	
WRF03A	Westinghouse 0 to 640 ohm RTD	6 AI		1	IOM85	203	
WRF03B	Westinghouse 0 to 320 ohm RTD	6 AI		1	IOM85	203	
WRT03A	Westinghouse 0 to 30 ohm RTD	4 AI		1	IOM85	203	
WRT03B	Westinghouse 0 to 320 ohm RTD	4 AI		1	IOM85	203	
WLJ04A	Westinghouse 0 to 10.24 V dc	3 AI	1 AO	2	IOM86	204	
WLJ04B	Westinghouse 0 to 5.12 V dc/10.24 V dc	3 AI	1 AO	2	IOM86	204	
WLJ04C	Westinghouse 0 to 20.48 mA	3 AI	1 AO	2	IOM86	204	
WPA06A	Westinghouse Pulse Counter	4 DI		4	IOM88	206	
WCI07A	Westinghouse Contact 48 V dc self powered	16 DI		5-8	IOM89	207	SOE - ECB6, pulse count - ECB7, ladder - ECB8
WDI07A	Westinghouse 5 V dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WDI07B	Westinghouse 24 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8

Table C-11. DCS FBMs for Westinghouse Process Control WPDF Systems (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
WDI07C	Westinghouse 48 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WDI07D	Westinghouse 120 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WDI07E	Westinghouse 12 V dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07A	Westinghouse 5 V dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07B	Westinghouse 24 V ac/dc	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07C	Westinghouse 24 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07D	Westinghouse 48 V ac/dc	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07E	Westinghouse 48 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07F	Westinghouse 120 V ac/dc	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07G	Westinghouse 120 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07H	Westinghouse 12 V dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07I	Westinghouse 12 V ac/dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07J	Westinghouse 48 V dc pulse	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07K	Westinghouse 120 V ac, high threshold	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07L	Westinghouse 120 V ac	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07M	Westinghouse 220 V ac	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07N	Westinghouse 220 V ac	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8

Table C-11. DCS FBMs for Westinghouse Process Control WPDF Systems (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
WID07O	Westinghouse 220 V dc	8 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WID07P	Westinghouse 220 V dc	16 DI		5-8	IOM89	207	SOE-ECB6, pulse count-ECB7, ladder-ECB8
WAH01A	Westinghouse -10.24 to +10.24 V dc	8 AI		1	IOM90	211	
WAH01B	Westinghouse -5.12 to +5.12 V dc	8 AI		1	IOM90	211	
WAH01C	Westinghouse 0 to +10.24 V dc	8 AI		1	IOM90	211	
WAH01D	Westinghouse 0 to +5.12 V dc	8 AI		1	IOM90	211	
WAO37A	Westinghouse 0 to 20.48 mA		4 AO	53	IOM93	237	
WAO37B	Westinghouse 0 to 10.24 V dc		4 AO	53	IOM93	237	
WAO37C	Westinghouse -10.24 to 10.24 V dc		4 AO	53	IOM93	237	
WAO37D	Westinghouse 0 to 5.12 V dc		4 AO	53	IOM93	237	
WAO37E	Westinghouse -5.12 to 5.12 V dc		4 AO	53	IOM93	237	
WAO37F	Westinghouse -10.24 to 10.24 V dc		4 AO	53	IOM93	237	
WAO37G	Westinghouse 0 to 20.48 mA		4 AO	53	IOM93	237	
WBO09A	Westinghouse 60 V dc		16 DO	5,8	IOM95	242	ladder - ECB8
WBO09B	Westinghouse 20 V dc		16 DO	5,8	IOM95	242	ladder - ECB8
WRO09A	Westinghouse inductive, mercury		8 DO	5,8	IOM95	242	ladder-ECB8
WRO09B	Westinghouse non-inductive, mercury		8 DO	5,8	IOM95	242	ladder-ECB8
WRO09C	Westinghouse inductive, solid state		8 DO	5,8	IOM95	242	ladder-ECB8
WRO09D	Westinghouse non-inductive, solid state		8 DO	5,8	IOM95	242	ladder - ECB8
WTO09A	Westinghouse triac		8 DO	5,8	IOM95	242	ladder - ECB8

DCS FBMs for APACS+ Systems

Table C-12 lists the DCS FBMs for migration of APACS+ Process Automation Systems. Each DCS FBM is electrically equivalent to a 200 Series FBM, as indicated by the software type (IOM#) and hardware type (HWT).

Table C-12. DCS FBMs for APACS+ Process Control Systems

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
AVI202	APACS+, Voltage Input, Thermocouple	16 AI		1	IOM84	(2) 202	
ART203	APACS+, RTD	16 AI		1	IOM85	(2) 203	
ASA211	APACS+, 0 to 5 V dc, 0 to 20 mA	32 AI		1	IOM90	(2) 211	
AHF214	APACS+ 4 to 20 mA, HART	16 AI or HART		200/ 201	IOM214	(2) 214	
AHF216	APACS+, 1 to 5 V dc, Redundant HART	16 AI or HART		201/ 202	IOM216	(2) 216	
AID115	APACS+, 115 V ac Digital Input	32 DI		5-8	IOM96	217	SOE-ECB6, pulse count-ECB7, ladder-ECB8
AID230	APACS+, 230 V ac	32 DI		5-8	IOM96	217	SOE-ECB6, pulse count-ECB7, ladder-ECB8
AEAM01	APACS+, Analog/Digital Input/Output, Pulse Input	16 AI, DI, PI*	16 AO, DO*	200/ 201	IOM234	234	
ASAM01	APACS+, Analog/Digital Input/Output	32 AI, DI*	32 AO, DO*	200/ 201	IOM235	235	
ASDM24	APACS+, 24 V dc, Digital Input/Output	32 DI**	32 DO**	200/ 201	IOM236	236	
ASDM48	APACS+, 48 V dc, Digital Input/Output	32 DI**	32 DO**	200/ 201	IOM236	236	
ADO125	APACS+, 125 V dc Digital Output		16 DO	5,8	IOM95	242	ladder-ECB8
AOD115	APACS+, 115 V ac Digital Output		32 DO	5,8	IOM95	(2) 242	ladder-ECB8
AOD230	APACS+, 230 V ac Digital Output		32 DO	5,8	IOM95	(2) 242	ladder-ECB8

* Each point can be configured as analog or digital, input or output.
** Each point can be configured as input or output.

Honeywell TDC2000 Migration Cards

The following is a list of Honeywell™ TDC2000 migration cards. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels
- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type
- Notes – Additional ECBs in use on this FBM.

Table C-13. Honeywell TDC2000 Migration I/O Cards

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
H2214A	Honeywell TDC2000 (8) 4-20 mA or HART	8 AI		200/ 201	IOM214	214	BC, MFC
H2214B	Honeywell TDC2000 (8) 4-20 mA or HART	8 AI		200/ 201	IOM214	214	HLPIU
H2215A	Honeywell TDC2000 (8) 4-20 mA or HART		8 AO	200/ 201	IOM215	214	BC, MFC
H2215B	Honeywell TDC2000 (4) 4-20 mA or HART		4 AO	200/ 201	IOM215	214	HLPIU
H2242	Honeywell TDC2000 (16) 60 V dc contact		16D	5 or 8	IOM95	242	
H2C02A	Honeywell TDC2000 (4) 4-20.4 mA, (4) 5 V dc or +/- 5 V dc	8A		1,3	IOM1	2	
H2C02B	Honeywell TDC2000 (4) 4-20.4 mA, (4) 40 V dc or +/- 40 V dc	8A		1,3	IOM1	2	
H2C02D	Honeywell TDC2000 (4) 4-20.4 mA, (4) 0-1 mA	8A		1	IOM1	2	
H2C02E	Honeywell TDC2000 (4) 4-20.4 mA, (4) 0-10 mA	8A		1	IOM1	2	
H2C02F	Honeywell TDC2000 (4) 4-20.4 mA, (4) -10.5-71.419 or 0-100 or +/-100 mV dc	8A		1,3	IOM1	2	
H2C02G	Honeywell TDC2000 (4) 4-20.4 mA, (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt	8A		1	IOM1	2	
H2C02H	Honeywell TDC2000 (4) 4-20.4 mA, (4) 1 V dc or +/- 1 V dc	8A		1,3	IOM1	2	
H2C02J	Honeywell TDC2000 (4) 0-1 mA, (4) 5 V dc or +/- 5 V dc	8A		1,3	IOM1	2	
H2C02K	Honeywell TDC2000 (4) 0-1 mA, (4) 40 V dc or +/- 40 V dc	8A		1,3	IOM1	2	
H2C02L	Honeywell TDC2000 (4) 0-1 mA, (4) 4-20.4 mA	8A		1	IOM1	2	
H2C02M	Honeywell TDC2000 (8) 0-1 mA	8A		1	IOM1	2	
H2C02N	Honeywell TDC2000 (4) 0-1 mA, (4) 0-10 mA	8A		1	IOM1	2	
H2C02P	Honeywell TDC2000 (4) 0-1 mA, (4) -10.5-71.419 or 0-100 or +/-100 mV dc	8A		1,3	IOM1	2	
H2C02Q	Honeywell TDC2000 (4) 0-1 mA, (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt	8A		1	IOM1	2	

Table C-13. Honeywell TDC2000 Migration I/O Cards (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
H2C02R	Honeywell TDC2000 (4) 0-1 mA, (4) 1 Vdc or +/- 1 V dc	8A		1,3	IOM1	2	
H2C02S	Honeywell TDC2000 (4) 0-10 mA, (4) 5 V dc or +/- 5 V dc	8A		1,3	IOM1	2	
H2C02T	Honeywell TDC2000 (4) 0-10 mA, (4) 40 V dc or +/- 40 V dc	8A		1,3	IOM1	2	
H2C02U	Honeywell TDC2000 (4) 0-10 mA, (4) 4-20.4 mA	8A		1	IOM1	2	
H2C02V	Honeywell TDC2000 (4) 0-10 mA, (4) 0-1 mA	8A		1	IOM1	2	
H2C02W	Honeywell TDC2000 (8) 0-10 mA	8A		1	IOM1	2	
H2C02X	Honeywell TDC2000 (4) 0-10 mA, (4) -10.5-71.419 or 0-100 or +/-100 mV dc	8A		1,3	IOM1	2	
H2C02Y	Honeywell TDC2000 (4) 0-10 mA, (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt	8A		1	IOM1	2	
H2C02Z	Honeywell TDC2000 (4) 0-10 mA, (4) 1 Vdc or +/- 1 V dc	8A		1,3	IOM1	2	
H2D02A	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 5 V dc	8A		1,3	IOM1	2	
H2D02B	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 40 V dc	8A		1,3	IOM1	2	
H2D02C	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 4-20.4 mA	8A		1,3	IOM1	2	
H2D02D	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 0-1 mA	8A		1,3	IOM1	2	
H2D02E	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 0-10 mA	8A		1,3	IOM1	2	
H2D02G	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 30ohm Cu, 120ohm Ni or 320ohm Pt	8A		1,3	IOM1	2	
H2D02H	Honeywell TDC2000 (4) -10.5-71.419 or 0-100 or +/-100 mV dc, (4) 1 V dc or +/- 1 V dc	8A		1,3	IOM1	2	
H2J02A	Honeywell TDC2000 (4) Ref RTD for TC, (4) 5 V dc or +/- 5 V dc	8A		1,3	IOM1	2	

Table C-13. Honeywell TDC2000 Migration I/O Cards (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
H2J02B	Honeywell TDC2000 (4) Ref RTD for TC, (4) 40 V dc or +/- 40 V dc	8A		1,3	IOM1	2	
H2J02C	Honeywell TDC2000 (4) Ref RTD for TC, (4) 4-20.4 mA	8A		1	IOM1	2	
H2J02D	Honeywell TDC2000 (4) Ref RTD for TC, (4) 0-1 mA	8A		1	IOM1	2	
H2J02E	Honeywell TDC2000 (4) Ref RTD for TC, (4) 0-10 mA	8A		1	IOM1	2	
H2J02F	Honeywell TDC2000 (4) Ref RTD for TC, (4) -10.5-71.419 or 0-100 or +/-100 mV dc	8A		1,3	IOM1	2	
H2J02G	Honeywell TDC2000 (4) Ref RTD for TC, (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt	8A		1	IOM1	2	
H2J02H	Honeywell TDC2000 (4) Ref RTD for TC, (4) 1 V dc or +/- 1 V dc	8A		1,3	IOM1	2	
H2M01A	Honeywell TDC2000 5 V dc or 1-5 V dc or +/- 5 V dc	8A		1,3	IOM1,3	2	
H2M01B	Honeywell TDC2000 4-20 mA	8A		1	IOM1	2	
H2M01C	Honeywell TDC2000 4-20 mA (powered)	8A		1	IOM1	2	
H2M01D	Honeywell TDC2000 4-20 mA	8A		1	IOM1	2	
H2M02	Honeywell TDC2000 -10.5-71.419 or 0-100 or +/- 100 mV dc	8A		1,3	IOM1	2	
H2M02A	Honeywell TDC2000 5 V dc or +/- 5 V dc	8A		1,3	IOM1	2	
H2M02B	Honeywell TDC2000 40 V dc or +/- 40 V dc	8A		1,3	IOM1	2	
H2M02E	Honeywell TDC2000 -10.5-71.419 or 0-100 or +/-100 mV dc (on card cold junc sense)	8A		1,3	IOM1	2	
H2M03	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) 5 V dc or +/- 5 V dc	8A		1,3	IOM1	3	
H2M03A	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) 40 V dc or +/- 40 V dc	8A		1,3	IOM1	3	
H2M03B	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) 4-20.4 mA	8A		1	IOM1	3	

Table C-13. Honeywell TDC2000 Migration I/O Cards (Continued)

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
H2M03C	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) 0-1 mA	8A		1	IOM1	3	
H2M03D	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) 0-10 mA	8A		1	IOM1	3	
H2M03E	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt, (4) -10.5-71.419 or 0-100 or +/-100 mV dc	8A		1,3	IOM1	3	
H2M03F	Honeywell TDC2000 (8) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt	8A		1	IOM1	3	
H2M03G	Honeywell TDC2000 (4) 0-30 ohm Cu, 120 ohm Ni or 320 ohm Pt Ref RTD for TC, (4) 1 V dc or +/- 1 V dc	8A		1,3	IOM1	3	
H2M04	Honeywell TDC2000 0-20.4 mA		4A	1	IOM2	4	
H2M06	Honeywell TDC2000 24 V dc or 48 V dc external pulse		4P	4	IOM4	6	
H2M06A	Honeywell TDC2000 125 V dc external pulse		4P	4	IOM4	6	
H2M07	Honeywell TDC2000 5 V dc (jumper input source or power bus)	16D		5	IOM5	7	
H2M07E	Honeywell TDC2000 24 V dc (supplied at term.)	16D		5	IOM5	7	
H2M09	Honeywell TDC2000 60 V dc contact		8D	5	IOM5	9	
H2M17	Honeywell TDC2000 5 V dc in, (2) 0-20.4 mA (4) 60 Vdc out	4A	2A, 4D	9,34, 36, 52	IOM9,34, 36, 52	17	MDACT-ECB34, MDPulse-ECB36, PID ECB52
H2M24	Honeywell TDC2000 125 V dc contact (external power)	16D		5-8	IOM 5-8	24	SOE-ECB6, pulse count-ECB7, ladder-ECB8
H2M26	Honeywell TDC2000 125 V dc		8D	5,8	IOM5,8	26	ladder-ECB8

Honeywell TDC3000 Migration Cards

The following is a list of Honeywell TDC3000 migration cards. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- In – Number of input channels

- Out – Number of output channels
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type
- Notes – Additional ECBs in use on this FBM.

Table C-14. Honeywell TDC3000 Migration I/O Cards

Type	Signal Description	In	Out	SW ECB#	IOM#	HWT	Notes
H3M01	Honeywell TDC3000 0 to 5 V dc, 1 to 5 V dc, 4 to 20 mA	16A		1,47	IOM01, IOM54	1,52	sw type is 42 before 6.1.1
H3M03	Honeywell TDC3000 -10.5-+71.4 mV, 0-5 V, 0-100 mV, thermoc, RTD	8A		3	IOM01	2	
H3M06	Honeywell TDC3000 pulse input FTAs	8P		4	IOM04	6	dual FBM06s
H3M07	Honeywell TDC3000 digital input FTAs	32D		6	IOM05	7	
H3M09	Honeywell TDC3000 digital output FTAs		16D	5	IOM05	9	
H3M37	Honeywell TDC3000 analog output FTAs 0 to 20.4 mA		8A	53	IOM53	37	

FCMs and Equivalents

The following is a list of FCMs and equivalents. The table heading definitions are as follows:

- Type – FBM Type
- Signal Description – Purpose of FBM
- SW ECB# – Equipment control block software type used for this FBM
- IOM – The software that is downloaded to this FBM
- HWT – Hardware type.

Table C-15. FCMs and Equivalents

Type	Signal Description	SW ECB#	IOM#	HWT
WFCM10E	Westinghouse DIN/copper version of FCM	110	IOM82	200
WFCM10Ef	Westinghouse fiber version of FCM	110	IOM82	200
DCM10E	DIN rail mount version of FCM	110	IOM82	200
FBI10E	268 Kb HDLC, 10 Mbaud copper Ethernet, copper, half-Y-module form factor	110	IOM82	200
FCM100Et	Fieldbus Communications Module 100Et, 2 Mb HDLC, 100 Mbaud dual fiber Ethernet	210	IOM210	210
FCM100E	Fieldbus Communications Module 100E, 2 Mb/268 Kb HDLC, 100 Mbaud single fiber Ethernet	210	IOM210E	210

Table C-15. FCMs and Equivalents (Continued)

Type	Signal Description	SW ECB#	IOM#	HWT
FEM100	4-to-1 fieldbus multiplexer for Expanded fieldbus support for the FCP270	None	None	-
FBI	268 Kb half-Y-module bus extender (100 Series FBMs only)	None	None	-
FBI100	2 Mb Fieldbus Extender/Isolator	None	None	-
FBI200	2 Mb and 268 Kb Fieldbus Extender	None	None	-
FCM2F2	2 Mb HDLC Fiber Fieldbus Extender, 2 km (1.24 mi)	None	None	-
FCM2F4	2 Mb HDLC Fiber Fieldbus Extender, 4 km (2.5 mi)	None	None	-
FCM2F10	2 Mb HDLC Fiber Fieldbus Extender, 10 km (6.2 mi)	None	None	-

Index