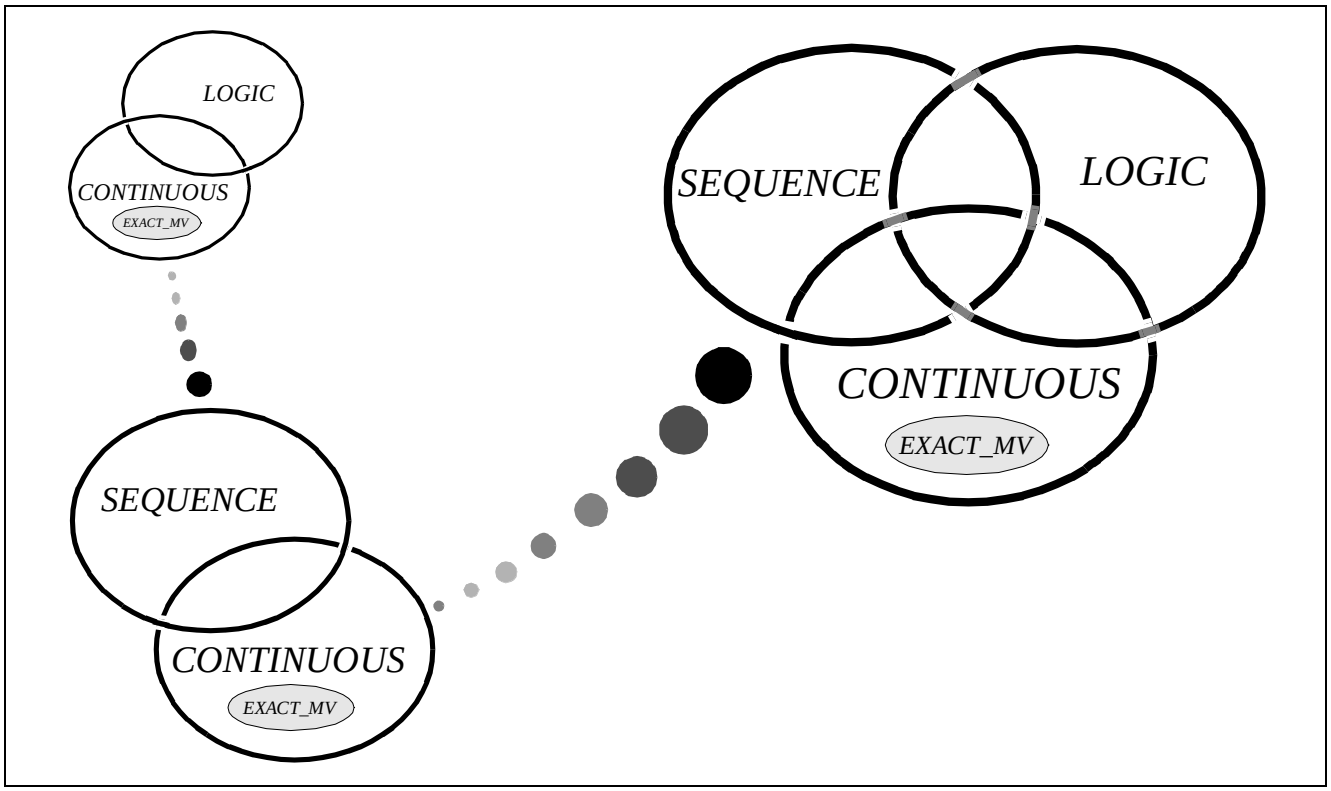
FOXBORO®

PSS 21S-3B1 B3

# I/A Series® Software
# FoxBlock Integrated Control Software




The FoxBlock Integrated Control Software provides a selection of continuous, sequential, and ladder logic control domains. Each domain can be used independently or integrated to meet specific application requirements.

The I/A Series FoxBlock Integrated Control Software provides the user with a comprehensive control strategy and input/output implementation for on/off control, timing, regulatory and feedback applications. Implemented within a block and parameter structure, FoxBlock provides the base for the integration of continuous, ladder logic and sequential control with programmable logic controller functionality.

The software control strategies allow you to mix and match continuous and Programmable Logic Block (PLB) capabilities of compounds and blocks with sequencing in the same control scheme.

The process control domains -- continuous, sequential and batch, and ladder logic -- are primarily intended to execute in any I/A Series control station. FoxBlock Integrated Control Software running in a control processor provides the application interface to the Fieldbus subsystem and other I/O subsystems through I/O blocks. These I/O blocks interface to the specific applications executing in the Fieldbus Modules (FBMs), Fieldbus Processors (FBPs), and Control Integrators.

Access to the FBMs, FBPs, Control Integrators, and associated devices (for example, field sensors, actuators, Intelligent Field Devices, Auto/Manual stations, SPECTRUM I/O Cards, or Cluster I/O Cards) is through the FoxBlock integrated control software in the following stations:

- CP10, CP30, CP40 residing on an I/A Series Nodebus
- AW70 Integrator
- AW51 Integrator
- Micro I/A OMC.

Other types of control stations also handle the integration of data into the control scheme from various devices, such as Allen-Bradley Programmable Logic Controllers, Modicon Programmable Controllers, INTERSPEC serial devices, remote computing devices (such as gas analyzers, sequence of event monitors, and paper machine gauges).

The range of control stations includes:

- Control Processor 40 (CP40)
- Control Processor 30 (CP30)
- Control Processor 10 (CP10)
- Micro I/A
- Allen-Bradley Station (AB STA)
- Gateway stations
  - Allen-Bradley Data Highway (ABG)
  - Modicon Gateway (MODG)
  - Foreign Device Gateway (FDG)
  - Instrument Gateway (IG)
- Integrator 30 stations
  - Device Integrator 30 (FD30)
  - Integrator 30 for Allen-Bradley PLCs (AB30)
  - Integrator 30 for Modicon Modbus-based Programmable Controllers (MG30)
  - Modbus Plus Integrator
  - INTERSPEC Integrator 30 (IS30)
- Application Workstation Integrators
  - AW70 Integrator
  - AW51 Integrator

An optional fault-tolerant control station configuration, where two identical control stations (Primary and Shadow) are running in parallel, provides enhanced reliability to the control process.

Fault-tolerancy of specific control stations (CP30, CP10, AB30, MG30) also provides on-line upgrade capability, where upgrading to subsequent system software releases is handled safely and with minimum hold-control time. The Fast Takeover process (FTCP) provides the capability:

- to install new CP software in the Shadow module of a fault-tolerant module while the Primary module continues to control the process with the previous version of software.
- to place a Shadow module in control of the process while the Primary module goes into a standby state.

A Control Processor (CP) with a Fieldbus and its connections to the FBM(s) is shown in Figure 1. Figure 2 shows an Integrator 30 for Allen-Bradley PLCs and its connections to the AB PLCs.

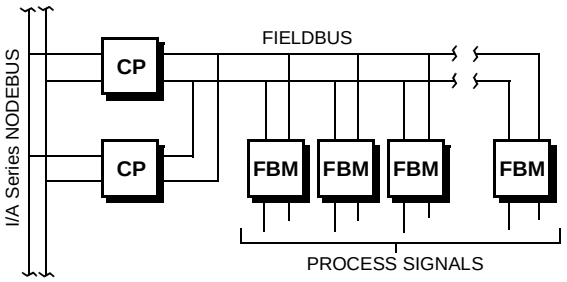



Figure 1. CP and FBM Connections

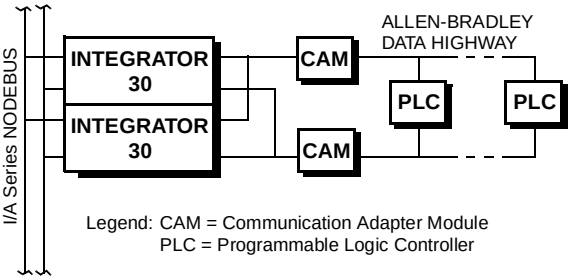



Figure 2. The Integrator 30 and PLC Connections

For large control applications, multiple nodes of stations can be networked via Local Area Networks (LANs) as well as connected to other networks.

Smaller systems are easily scaled to larger I/A Series control systems. See the following for additional information:

| | |
|---|---|
| PSS 21A-1A1 A1 | System Overview |
| PSS 21H-4S1 B3 | Application Workstation Scalable Solutions Overview |
| PSS 21H-4Q1 B3 | Personal Workstation Scalable Solutions Overview |
| PSS 21H-4P5 B4 | Personal Workstation for Fieldbus Interface |

## FOXBLOCK CONTROL CONCEPTS

Process control for I/A Series Systems is based on the concept of compounds and blocks. The compound is a logical collection of blocks, as shown in Figure 3, that perform a control strategy. The compound provides the basis for the integration of continuous control, ladder logic, and sequential control. A block is a member of a set of algorithms that perform a certain control task within the compound structure.

A compound structure:

- Has a 12-character, unique name (tag prefix)
- Can be turned on or off
- Supports continuous/sequential/ladder logic combinations
- Supports alarm destination assignments
- Can be phased.

Within this structure, the Integrated Control Software provides:

- Regulatory control
- Ladder logic
- Sequential control
- User-defined arithmetic and Boolean logic functions
- Alarm detection and reporting

The FoxBlock Integrated Control Software also provides communication interface blocks called Equipment Control Blocks (ECBs). These communication blocks, located in the control station, provide I/O and control information related to the FBMs, FBPs, Control Integrators and their related devices (such as actuators, sensors, Intelligent Field Devices, and I/O cards).

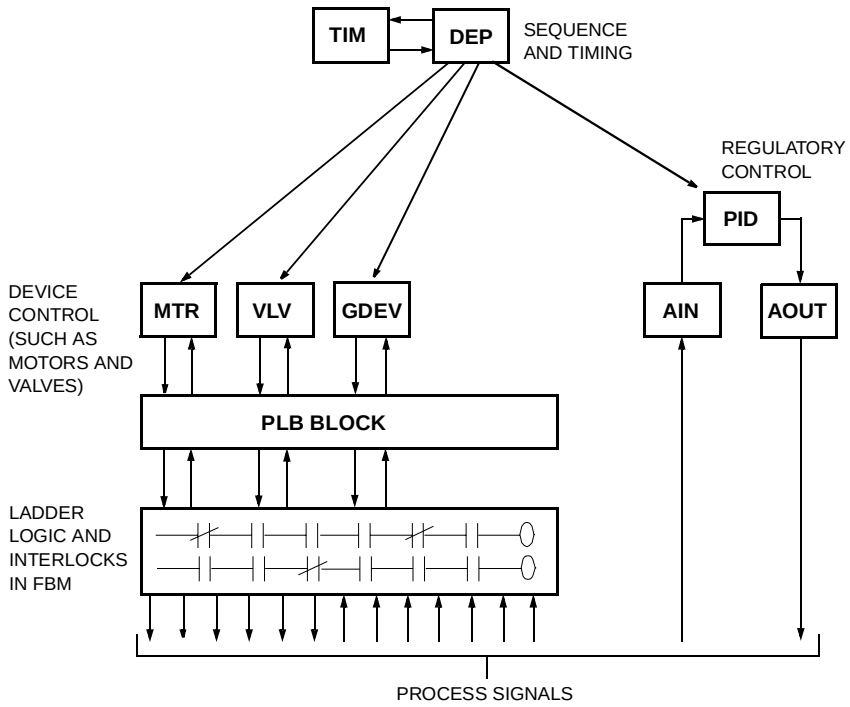



Figure 3. Total Integration in a Single Control Station

### Block Processing

The compound processor is a high-priority task scheduled by the operating system in the control processor to run every basic processing cycle (BPC), shown in Figure 4.

Blocks are configured with one of the fixed processing periods as follows:

| | |
|---|---|
| 0.1 second | 10 seconds |
| 0.2 second | 30 seconds |
| 0.5 second | 1 minute |
| 1 second | 10 minutes |
| 2 seconds | 60 minutes |
| 5 seconds | |

Most gateway/integrator blocks are configured with one of the following fixed processing periods:

| | |
|---|---|
| 0.5 second | 32 seconds |
| 1 second | 64 seconds |
| 2 seconds | 128 seconds |
| 4 seconds | 256 seconds |
| 8 seconds | 512 seconds |
| 16 seconds | 1024 seconds |

At the beginning of compound processing, I/O data is read for each Fieldbus Module, for both input and output channels. Data is read at the ECB scan period specified for that module. However, data cannot be read faster than the Block Processing Cycle (BPC). At the end of compound processing, process outputs are written to the module.

A function known as Phasing may be used to load-level a processor. A phase number is assigned to each block, which allows block execution to be time-shifted with respect to other active blocks. The starting time of one block can lead or lag the starting time of another block by an amount equal to the difference of the phase time. This allows many compounds/blocks to be processed in a single control processor without overload.

### Station Block

Each control processor contains a station block automatically created to provide global data on internal control processor system functions. The station block contains information such as:

- continuous block processing load
- total CP load
- basic processing cycles and CP overruns
- number of bytes of dynamic free memory
- number of peer-to-peer connections
- sequence processing load
- I/O scan load
- OM scan load
- idle time
- station alarm groups
- configuration security option.

## CONTROL CONFIGURATION CONCEPTS

The I/A Series control blocks listed in Table 1 are configured from the Process Engineer's Environment (the default), which is user configurable via the Integrated Control Configurator or FoxCAE Engineering Package. Figure 5 shows the path from the Process Engineer's Environment to the Integrated Control Configurator.

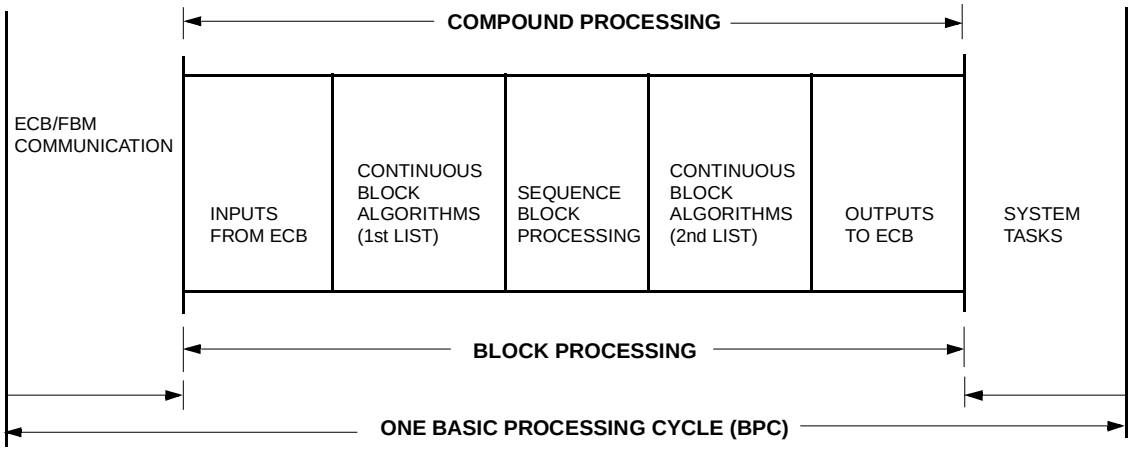



Figure 4. Basic Block Processing (BPC)

The Integrated Control Configurator provides for the configuration of Equipment Control Blocks associated with Fieldbus Devices (FBMs, FBPs, Control Integrators). These blocks provide the means of communicating the control information among the control stations, Fieldbus Devices, and I/O Cards, actuators, sensors, Intelligent Field Devices.

Configuration of continuous block types, ladder logic and sequential blocks is accomplished at the same level and through the same Integrated Control Configurator. During the configuration process the configurator recognizes the various block types chosen by the user and is able to identify their unique control domain association. Thus, when you type in a block type, the configurator knows which of the three domains should be accessed.

The general procedure is to name (create) a compound name under which blocks can be created and run. Any block type from the three domains (continuous, ladder, and sequential) can be created under a compound name. Thus, ladder logic, continuous, and sequential blocks can be combined in a compound to achieve a desired control function. See PSS 21S-8A1 B3 Integrated Control Configurator and PSS 21S-7A1 B3 FoxCAE Engineering Package for Windows for additional information.

Configuration of Intelligent field devices (IFDs) requires the use of the configurator in the Intelligent Transmitter Maintenance Environment as shown in Figure 5. This configurator works in conjunction with the Equipment Control Blocks (ECBs and Window ECBs) information configured using the Integrated Control Configurator. For additional IFD configuration information see PSS 21S-8A2 B3 Intelligent Transmitter Maintenance Environment.

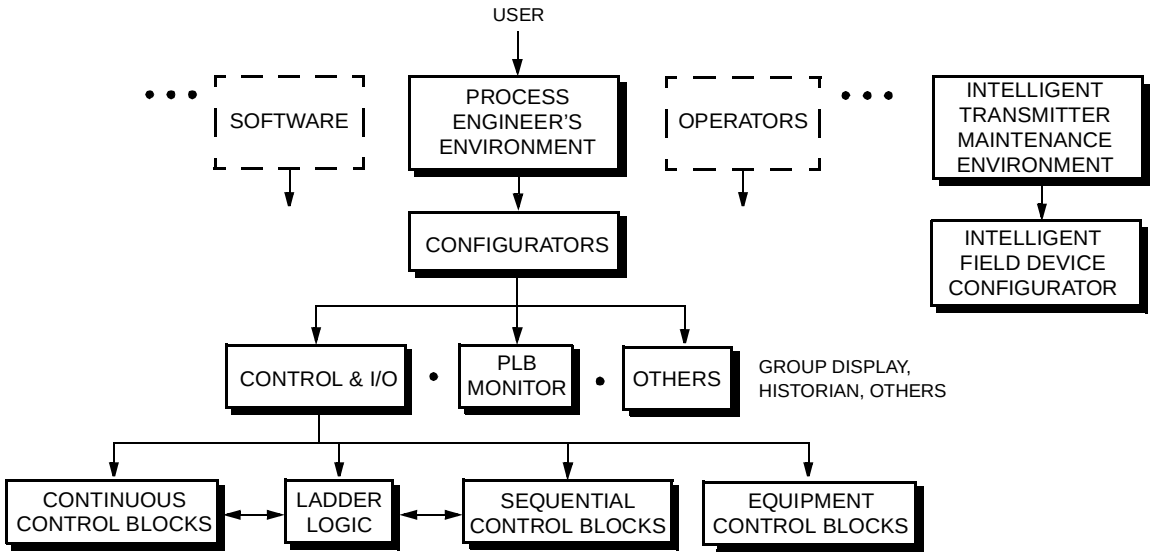



Figure 5. Process Engineer's Environment

Table 1. The FoxBlock Suite

| | |
|---|---|
| **Input/Output** | |
| MAIN | Multiple Analog Input. Supports up to 8 inputs from a Fieldbus Module and an internal channel for a temperature reference sensor. |
| AIN | Analog Input. Supports a single input channel from a Fieldbus Module. |
| AINR | Redundant Analog Input. Supports a redundant input from redundant Fieldbus Modules supporting either a single transmitter or redundant transmitters. |
| MCIN | Multiple Contact Input. Supports up to 32 contact inputs from digital input type Fieldbus Modules. |
| CIN | Contact Input. Supports a single input point for a digital input type Fieldbus Module. |
| AOUT | Analog Output. Provides auto/manual with bias function and supports a single output point for an analog type Fieldbus Module. |
| AOUTR | Redundant Analog Output. Supports a selected redundant output to redundant Fieldbus Modules. |
| MCOUT | Multiple Contact Output. Supports up to 16 digital outputs for a digital type Fieldbus Module. |
| COUT | Contact Output. Supports a single output for a digital type Fieldbus Module. |
| EVENT | Event. Provides messages reporting the sequencing of state-change events detected in a contact input Fieldbus Module. |
| **Device Control** | |
| GDEV | General Device. Provides Open/Close control of motor, or air, operated valves, and Run/Stop control of 2-wire or 3-wire motor circuits. |
| MDACT | Motor Driven Actuator. Used to control single or bi-directional motor-driven valves, solenoid valves, electric heaters, and similar devices. |
| MOVLV | Motor-Operated Valve. Operates two related output contacts, which open/close a motor-operated valve on an incremental basis. |
| MTR | Motor Controller. Performs both 2-wire and 3-wire motor control functions. |
| VLV | Valve on/off controller. Operates two related output contacts, which open or close a solenoid valve. |
| **Regulatory Control** | |
| BIAS | Bias. Produces an output that is the sum of the two input values, MEAS and BIAS, each of which can be scaled independently. |
| RATIO | Ratio. Computes an output that is the scaled multiplication of a measurement input with a ratio set-point input. |
| DGAP | Differential Gap. Provides optional bi-state or tri-state on/off control of two Boolean outputs. |
| LIM | Limiter. Provides a position and velocity limiter. |
| OUTSEL | Output Select. Allows the selection of one of two inputs from upstream blocks to be used as output to the process. |
| PID | Proportional, Integral, Derivative. Provides functions of a traditional, interacting, 3-term controller. |
| PIDA | Advanced Proportional, Integral, Derivative. Provides functions of an EXACT MV multi-variable controller and is used in conjunction with the feedforward and feedback tuning blocks. |
| DPIDA | Distributed PIDA. Interfaces the control processor to a PIDA-type algorithm executing in an analog input/output Fieldbus Module. |
| FFTUNE | Feedforward Tuning. Connects to the PIDA block as an extension block for feedforward control loops. |

Table 1. The FoxBlock Suite (Continued)

| | |
|---|---|
| FBTUNE | Feedback Tuning. Connects to the PIDA block as an extension block for performing enhanced EXACT MV tuning for feedback control loops. |
| PIDE | PID with EXACT Tuning. Provides PID with the EXACT self-tuning algorithm. |
| PIDX | PID Extended. Adds to PID: a sampled-data control option to use with sampling type instruments; TRACK capability that forces the output to track an independent track input; an optional non-linear gain element for Ph control; a batch option that provides preloadable integral bias for batch control. |
| PIDXE | PID Extended with EXACT Tuning. Combines the functionality of PIDX and PIDE. |
| PTC | Proportional Time Controller. Performs the functions of a proportional-time on/off controller. |
| **Selection, Ramping, Dynamic Compensation** | |
| DTIME | Dead Time. Delays the input a variable time interval before making it available at the output. |
| LLAG | Lead/Lag. Compensates signal value by making output dynamically lead or lag the input. |
| RAMP | Ramp. Performs a multi-segment ramp sequence; up to 5 segments may be used. |
| SIGSEL | Signal Selector. Examines up to 8 inputs and produces an output dependent upon a relational selection option. |
| SWCH | Switch Position Selector. Selects either of two independent inputs. |
| **Computation, Logic, and Conversion** | |
| ACCUM | Accumulator. Accumulates a real input signal and scales it to produce a real output quantity. |
| CALC | Calculator. Provides up to 50 sequentially-executed arithmetic and logical operations. Has the capability of a programmable scientific pocket calculator. |
| CALCA | Advanced Calculation. Adds dual-operand efficiency to many mathematical and logical calculation operations. |
| CHARC | Characterizer. Converts a real input to a real output using a table lookup of 20 piecewise linear conversion segments. |
| LOGIC | Logic. Provides logic and timer functions. |
| MATH | Mathematics. Provides a set of mathematics functions for specialized control needs. |
| PATT | Pattern Matching. Provides matching capability for 16-bit patterns. |
| STATE | State. Outputs selected 16-bit patterns. |
| **Alarm** | |
| ALMPRI | Alarm Priority Change. Dynamically reassigns the specified priority of an alarm point. |
| BLNALM | Boolean Alarm. Provides independent state-change alarm messages for 8 Boolean-type inputs. |
| MEALM | Measurement Alarm. Provides an alarm message for one measurement input with limit indicators for Intelligent Field Devices: high-low absolute alarming, rate-of-change alarming, and high-high/low-low alarming. |
| MSG | Message Generator. Provides a state change message for each of eight inputs from Intelligent Field Devices. |
| PATALM | Pattern Alarm. Compares the relationship of up to 8 Boolean inputs to up to 8 unique user-specified patterns. |
| REALM | Real Alarm. Supports 3 types of alarming: high/low absolute alarming on the measurement, rate-of-change alarming on the measurement, high/low deviation alarm on the measurement/set point difference. |
| STALM | State Alarm. Provides state alarming for event changes received from an Intelligent Field Device. |

Table 1. The FoxBlock Suite (Continued)

| | |
|---|---|
| **User-Defined Sequence Blocks** All Sequence Blocks (IND, DEP, and EXC) provide sequential control for regulatory feedback applications at the equipment control level. | |
| DEP | Dependent Sequence. Is used for normal sequence logic to define sequence of events, activate/ deactivate other sequence blocks, activate/deactivate Monitor (MON) blocks or individually monitor cases of a MON block, control timers in Timer (TIM) blocks, access any shared variable/parameter of any block in the system. Pauses when EXC blocks in the same compound are active. |
| EXC | Exception Sequence. Is used to handle abnormal events. Exception blocks are normally activated by monitor blocks. |
| IND | Independent. Is used to provide the same functions as the Dependent block; however, the IND block does not pause as DEP does when any EXC blocks in the same compound are active. |
| MON | Monitor. Provides the capability of monitoring process conditions. |
| TIM | Timer. Contains four individual timers that can be run by a Sequence block (IND, DEP, or EXC) to time sequence activities. |
| **Data Storage** | |
| REAL | Real Data Variable. Provides the capability of storing a real data value for use by other control blocks. |
| BOOL | Boolean Data Variable. Provides the capability of storing a Boolean data value for use by other control blocks. |
| LONG | Long Integer Data Variable. Provides the capability of storing a long integer data variable for use by other control blocks. |
| STRING | String Data Variable. Provides the capability of storing a configurable and settable string data variable for use by other control blocks. |
| PACK | Packed Long Data Variable. Provides the capability of storing a packed long data variable for use by other control blocks. |
| **Gateway Blocks** | |
| ABSCAN | Allen-Bradley Scan. Defines the PLC data table access specifications for a compound. |
| FDSCAN | Foreign Device Scan. Scans the foreign device for data and collects the input values and statuses to be sent to applications. |
| MDSCAN | Modicon Scan. Defines the PC's I/O point specifications for a compound. |
| **Window Equipment Control Blocks** | |
| AMSPRI | Analyzer Management Software Primary. Provides data and configuration information from the 931D Process Gas Chromatograph for output to control blocks. |
| ECB13 | Hydrostatic Tank Gauge. Provides data and configuration information from the Hydrostatic Tank Gauge for output to control blocks. |
| ECB18 | Intelligent Transmitter. Provides data and configuration information directly from Intelligent Transmitters as output to control blocks. |
| ECB22 | Mass Flow Transmitter. Provides data and configuration information directly from a Mass Flow Transmitter for output to control blocks. |
| **Equipment Control Blocks** | |
| ECB01 | Analog Input |
| ECB02 | Analog Input & Analog Output |
| ECB04 | Pulse In Analog Output |
| ECB05 | Digital In, Sustained/Momentary, Digital Out |
| ECB06 | Sequence of Events Input |
| ECB07 | Digital In & Pulse Count Input |
| ECB08 | Ladder Logic - OR - dc Out/Validated Input |

Table 1. The FoxBlock Suite (Continued)

| | |
|---|---|
| ECB09 | Remote/Manual Station (Analog I/O, Digital I/O) |
| ECB11 | Reserved for Primary FBM |
| ECB12 | Parent ECB for Window ECB18 |
| ECB14 | Panel-Mounted Display |
| ECB15 | Allen-Bradley Programmable Logic Controller |
| ECB16 | Modicon Programmable Logic Controller |
| ECB19 | 760 Micro Controller |
| ECB20 | Foreign Device |
| ECB21 | 761 Micro Controller |
| ECB23 | Intelligent Transmitter In, Analog Output. Dual Baud Rate, Redundant Output |
| ECB29 | INTERSPEC Gateway (CCM) |
| ECB30 | INTERSPEC Gateway (AIM) |
| ECB31 | INTERSPEC Gateway (UIO) |
| ECB32 | INTERSPEC Gateway (UFM) |
| ECB34 | MDACT Feedback Tri-State |
| ECB36 | MDACT PWM Tri-State |
| ECB38R | Intelligent Transmitter In, Analog Output, Dual Baud Rate, Redundant I/O |
| ECB39 | Gas Chromatograph Primary |
| ECB40 | Allen-Bradley PLC 5 Series |
| ECB41 to ECB46 | Cluster and SPECTRUM I/O ECBs |
| ECB47 to ECB51 | Cluster and SPECTRUM FBP ECBs |
| ECB48R | Redundant SPECTRUM UCM |
| ECB52 | Distributed PIDA Controller |

## CONTINUOUS CONTROL CONCEPTS

The basic element for implementing any of the various control domains is the block. In continuous control, the block is a member of a set of predefined algorithms. You select, organize, and configure the blocks into a group called a compound. The compound is a logical collection of blocks that perform a specific control task.

To use a cascade and feedforward control strategy as an example, you might build the compound by selecting the following blocks:

- Three Analog Input (AIN) blocks
- One advanced controller (PIDA) block
- One Proportional/Integral/Derivative (PID) control block
- One Analog Output (AOUT) block
- One Feedback Tuner Extender (FBTUNE) for PIDA
- One Feedforward Tuner Extender (FFTUNE) for PIDA

These blocks execute in the order in which they appear in the compound. However, when phasing is used, the phasing of the blocks determines the order of execution.

The name that you assign a compound must be unique throughout the system, for example, CASCADE1 shown in Figure 6. Each block that you choose also requires a name. This can be any combination of up to 12 characters, for example, FLWTRNS100.

FBTUNE and FFTUNE extender blocks may be linked to the PIDA blocks to provide feedback and feedforward adaptive tuning.

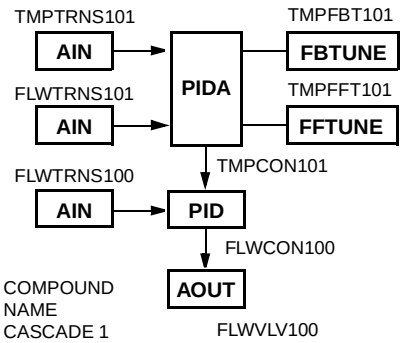



Figure 6. Example of a Compound Control Scheme

The feedforward input may be used to anticipate the effect of a measured load or an interacting controller. Refer to PSS 21S-3A2 B3 EXACT Multivariable Control for additional information.

The Control Station must be sized so that it does not exceed any of the following limits:

- Maximum number of FBMs allowed
- Maximum throughput of the Fieldbus
- Available memory to store the control blocks
- Available memory to store the OM (Object Manager) scanner databases
- Available time for processing control blocks and ECBs

**NOTE**

The Control Station should be sized following the Control Processor Sizing Spreadsheet (PSS 21S-4Q1 B3).

Any block in any compound can be connected to any other block in any other compound, anywhere in the system.

A control processor can have many compounds, but a compound cannot cross station boundaries. However, compounds can be interconnected across station boundaries via block connections.

Again referring to Figure 6, if another compound in the system needed the flow signal FLWTRNS100 as an input, the other compound could access FLWTRNS100 by specifying CASCADE1:FLWTRNS100.PNT

Therefore, the compound name CASCADE1 used in the example can only be used once in that system. But many blocks called FLWTRNS100 can reside in the system, as long as only one block by that name resides in CASCADE1.

When you are configuring connections within a compound, you do not need to specify the compound name. For instance, the output of TMPTRNS101 is the measurement signal to TMPCON101. You specify the measurement of TMPCON101 as:

":TMPTRNS101.OUT"

### Input/Output Blocks

I/O blocks provide the application interface for the physical process inputs and outputs. The I/O blocks relate logical names, such as FLWTRNS100, to physical hardware point addresses, identified by the device ID and point number.

All references to I/O points from control, displays, data logging, and so on use a logical name instead of the physical address.

### Equipment Control Blocks (ECBs)

The software interface centered about the Equipment Control Block (ECB) is the communications network between the I/A Series Control Processor and the process instrumentation. ECBs vary depending on the Fieldbus Module (FBM) and its application. The list of ECBs is located in Table 1.

FBMs read the I/O data from the process instrumentation on a per FBM basis when the compound process begins. The FBM then conditions (for example, digitizes and normalizes) the data where necessary, and stores the data and the status into its ECB. When the control compound is processed, the Compound Processor uses newly retrieved data and generates new outputs for the appropriate ECB.

Window ECB(s) provide an extended communications interface for a direct interchange between the I/A Series system and Intelligent Field Devices (IFDs). IFDs incorporate intelligence at the hardware device level (signal conditioning, process control) and use the software interface (Window ECBs) in the control station to provide the information to the control blocks.

## LADDER LOGIC CONCEPTS

Ladder logic allows you to design modular solutions to logic control problems in familiar, easy-to-use relay ladder symbols.

By itself, ladder logic performs simple relay-type operations. Ladder logic used in conjunction with a control processor's continuous and sequential control blocks can implement sophisticated control strategies. Through the communication capabilities of the process management and control network, ladder logic executing in one Fieldbus Module can be coordinated with other ladder logic and with continuous and batch processes in any network control processor.

A Programmable Logic Block (PLB) provides connections between a ladder diagram and user tasks, other blocks, and other ladder diagrams. Connection is made through PLB input and output parameters. These parameters map to user-configured external flag references within a ladder diagram division called a segment.

Each PLB can represent a segment of a ladder diagram. The name of the segment is the block name. Up to three PLBs can be connected to a single Fieldbus Module to support a diagram consisting of multiple segments.

Creating a PLB establishes a ladder diagram source file. Using these source files, PLB Editor software in the Integrated Control Configurator allows you to construct a ladder diagram in segments, check for syntax errors, and produce a printed copy for documentation. You can compile the ladder diagram and install the code in a digital Fieldbus Module or save the source files for later use. You can develop a library of ladder diagrams and retrieve (copy) segments for inclusion in other ladder diagrams. You can also save ladder diagram source files (as part of a compound) on diskette.

Using the PLB View or PLB Monitor, accessible from the Process Engineer's Environment, you can either view the state of the ladder logic associated with the PLB or monitor the ladder logic from the process and force contacts and coils on or off to verify correct operation of the logic under simulated process conditions.

The PLB Detail Display, automatically created after ladder logic configuration, allows you to monitor the status of ladder logic contacts, timers, counters, and coils through a ladder diagram display, which uses industry standard symbols, or through graphic displays that you create. User-generated displays access the status of ladder logic elements through external flag parameters.

### Ladder Logic Feature Summary

The ladder logic provides:

- Simplified programming of control logic as a relay ladder diagram, using industry standard symbols.
- A user-definable label, up to 14 characters, for each Fieldbus Module coil table address. (Multiple ladder elements referring to the same coil table address have the same user label.)
- Menu-oriented displays for configuration and operation.
- On-line help screens and prompt messages for assistance during interactive program development and maintenance.
- Dynamic process monitoring through graphic ladder display.
- Hard copy documentation of ladder logic diagrams.
- A user-developed library of ladder logic diagrams.
- Ladder logic source files that are protected from unauthorized or accidental change (editing allowed only from Process Engineer's Environment).

- A logic test mode that allows you to force contacts and coils and assign timer/counter preset and reset values for dynamic verification of program operation.

### Ladder Logic I/O Capability

Ladder diagrams sense digital process input status and control digital process outputs.

*Inputs* 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, pushbuttons, and pressure switches. External input flags allow continuous and sequential blocks and user tasks to initiate or control ladder logic action.

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

### Programmable Logic Block (PLB)

The PLB block, shown in Figure 7, supports external flag references to/from ladder logic executing in a discrete (digital) Fieldbus Module. The PLB block translates between ladder logic external flag assignments and I/O parameter assignments. The block reads each ladder logic output flag reference from the Fieldbus Module and updates the appropriate output parameter value. The block writes each input parameter value to the appropriate ladder logic input flag reference in the Fieldbus Module. Using this technique, the integration of continuous and ladder logic domains is accomplished. For example, an output flag parameter might be connected to the auto-manual parameter of a PID block to initiate or stop the action of the controller or it might be connected to the high output limit of a PID controller in order to allow the controller to generate an output to move a final operator.

### Programmable Logic Block Feature Summary

The PLB block provides:

- A mapping parameter and an output parameter for each of 32 ladder logic external output flags for monitoring ladder logic.
- A mapping parameter and an input parameter for each of 32 ladder logic external input flags for activating or controlling ladder logic.
- Manual/Auto mode for manually updating block outputs to allow simulation of flag outputs normally provided by the ladder logic.
- A connectable status parameter to alert other blocks to abnormal Fieldbus Module or ladder logic status.

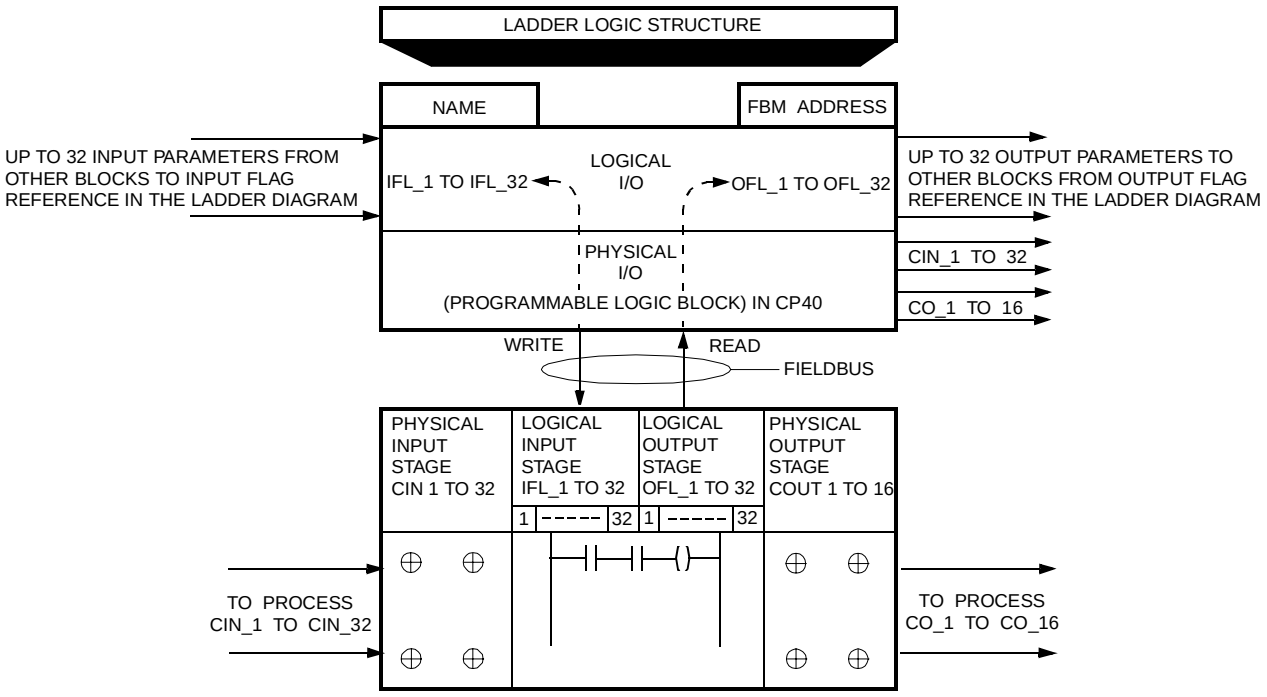



Figure 7. Programmable Logic Block

## LADDER DIAGRAM CONSTRUCTION

The ladder logic software in a Fieldbus Module performs logical operations based on the placement of the contacts and coils in the ladder diagram. These configurations consist of series, parallel, or series/parallel paths. Series paths provide ANDing of the conditions; parallel paths provide ORing of conditions.

The ladder diagram, shown in Figure 8, is constructed within a work area bounded by the menu bar at the top of the screen, the configurator message line at the bottom, and a column at the right for preset, reset, and accumulated values. The work area is reserved for placing the ladder rungs and logic symbols.

The PLB Editor within the Integrated Control Configurator allows you to create ladder diagrams from a set of predefined symbols, 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.

You can select the SYMBOLS function from the main menu bar to display the symbols and function key assignments and HELP to display the cursor control keys and technical identifiers that the Create/Modify function offers for constructing a ladder.

Table 4 indicates the total number of each type of technical identifier available in the FBM, rather than the total for any one segment. You may use the same technical identifier, such as OFL_1, in more than one segment of the final ladder, but this does not create a new OFL_1. When the complete ladder is executed in the FBM, the final value of OFL_1 is transmitted to all PLBs connected to that FBM.

### Symbols

Table 2 shows the ladder instruction set used to implement ladder logic.

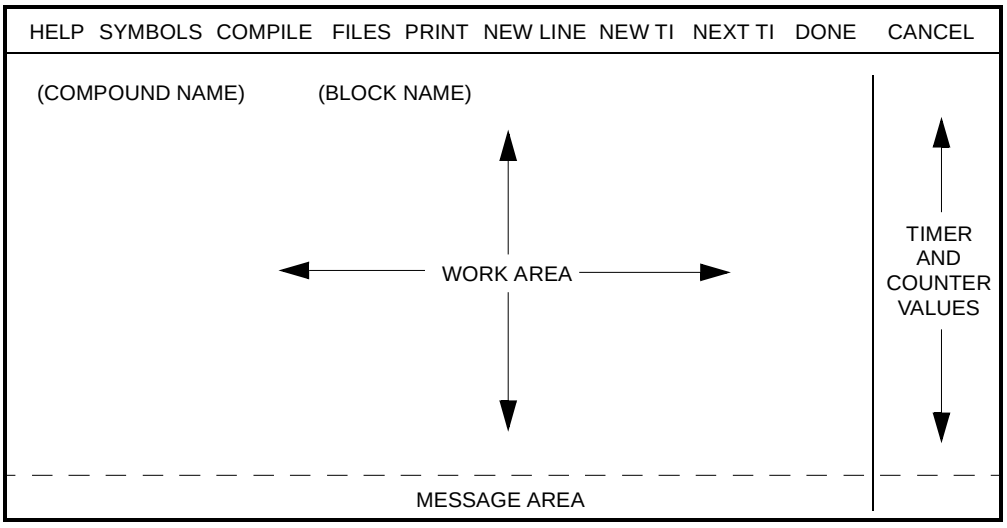



Figure 8. Ladder Diagram Work Area within PLB Editor

Table 2. Ladder Logic Instruction Summary

| Symbol | Name | Description |
|---|---|---|
| —\| \|— | Normally Open Contact | Provides logic value continuity (power flow) from left to right when the named signal is present (true state). |
| —\|/\|— | Normally Closed Contact | Provides logic value continuity (power flow) from left to right when the named signal is absent (false state). |
| ——— | Connector | Provides logic value continuity through a symbol position. |
| —( )— | Energize Coil | Sets Boolean value representing coil status true if any rung path has continuity. If logic continuity is lost, Boolean value is set false. |
| —(/)— | Write Not Coil | Sets Boolean value representing coil status false if any rung path has continuity. If logic continuity is lost, Boolean value is set true. |
| \| ——— | Vertical Connector (up) | Joins two rows when used with a down connector. |
| ——— \| | Vertical Connector (down) | Joins two rows when used with an up connector. Used in pairs, vertical connectors provide logic value continuity (power flow) vertically within a ladder diagram. |
| " " | Blank | Inserts blanks in a symbol position, interrupting logic value continuity. |
| —(L)— | Latch Coil | Sets Boolean value representing coil status true if any rung path has continuity. If logic continuity is lost after the coil is set, Boolean value remains true until associated unlatch coil is set. |
| —(U)— | Unlatch Coil | Unlatches an output that was previously set by a latch coil instruction. |
| —(RTO)— | Retentive Timer-On Delay | Provides a delayed action on a rung transition from false to true. |
| —(RTF)— | Retentive Timer-Off Delay | Provides a delayed action on a rung transition from true to false. |
| —(RST)— | Counter/Timer Reset | Resets a counter or timer having the same technical identifier as this symbol. |
| —(CTU)— | Up Counter | Increments a counter accumulated value on off-to-on transitions. |
| —(CTD)— | Down Counter | Decrements a counter accumulated value on off-to-on transitions. |
| —(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 | Marks the end of the MCR conditional group of rungs. |
| —(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. |
| —(TON)— | Non-Retentive Timer-On Delay | Same as RTO except that when PRESET is reached and the status flag becomes true, the timer is immediately RESET. |
| —(TOF)— | Non-Retentive Timer-Off Delay | Same as RTF except that when PRESET is reached and the status flag becomes true, the timer is immediately RESET. |

Table 3. Text and Numeric Elements

| Text and Numeric Elements | Description |
|---|---|
| Label | Two rows of up to seven characters each. |
| Technical Identifier | A single row of up to six characters that identifies a symbol by type and number (see Table 4). |
| Preset Value | For timer or counter symbol, maximum value, up to 65 535. Timer values represent tenths of seconds; counter values represent counts. |
| Reset Value | For timer or counter symbol, minimum value, up to 65 535. Timer values represent tenths of seconds; counter values represent counts. |
| Accumulated Value | Current value accumulated in a timer or a counter. |
| Rung Descriptor | Up to 3 lines of text with up to 60 characters in each. Used for documentation (see Figure 9). |

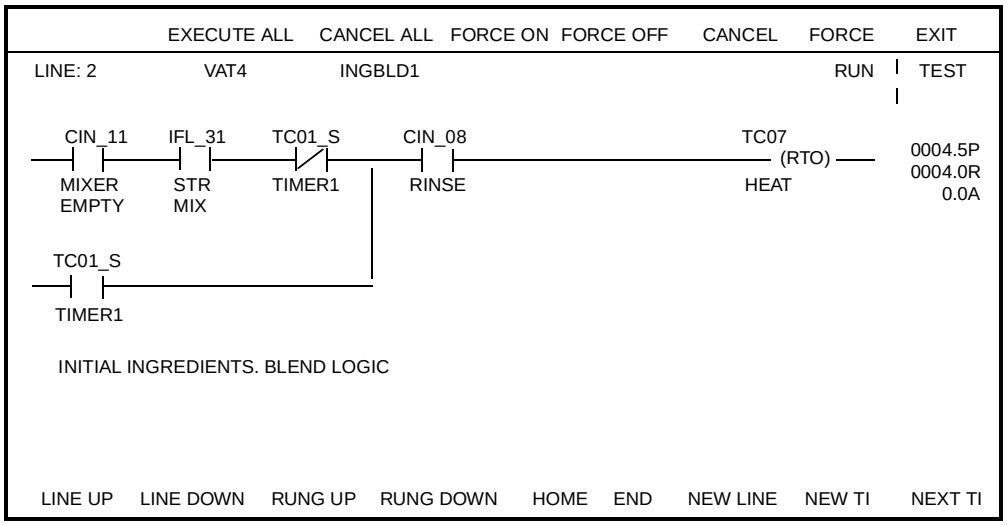



Figure 9. Sample Ladder Diagram in PLB Monitor for Testing

Table 4. 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_O through TC16_O | Timer/Counters Overflow |
| TC01_S through TC16_S | Timer/Counters Status |
| INT_01 through INT_32 | Internal Flags |
| INIT | Initialize Flag |
| POWERF | Power Fail Flag |
| COMMF | Communications Failure Flag |
| FAILSF | Fail_Safe Flag |

**Text and Numeric Entries**

Table 3 describes the text and numeric entries associated with the ladder diagram. A technical identifier is placed above each symbol and a label for it is placed below; an optional rung descriptor can be placed beneath each rung (see Figure 10). Table 4 shows the format for technical identifiers.

A ladder line consists of five screen lines: two lines allocated for the label, one line for the symbol, one line for the technical identifier, and one blank line for separation. You can scroll the ladder diagram up or down.

## SPECIFICATIONS FOR LADDER LOGIC COMPONENTS

**Fieldbus Module**

MEMORY SPACE AVAILABLE FOR A LADDER DIAGRAM
Approx. 1k bytes
NUMBER OF PHYSICAL INPUTS
Up to 32 per Fieldbus Module and expander
NUMBER OF PHYSICAL OUTPUTS
Up to 16 per Fieldbus Module and expander
SCAN RATE
Average of 390 ladder logic symbols scanned in 2 to 5 ms

**Programmable Logic Block**

UPDATE INTERVAL
Updated at 1 of 9 user-specified intervals from 100 ms to 1 hour; default period is 500 ms
NUMBER OF INPUT PARAMETERS
32 external input flags
NUMBER OF OUTPUT PARAMETERS
32 external output flags

**Ladder Diagram**

NUMBER OF FLAGS
*External Input*
32 flags
*External Output*
32 flags
*Internal Input/Output (Combined)*
32 flags
NUMBER OF FLAGS (CONT.)
*Timer/Counter*
32 flags
*Special Purpose (Initialization, Power Failure, Communication Failure, Failsafe)*
4 flags
COUNTERS AND TIMERS
*Combined Total of Counter and Timers (In Any Combination)*
Up to 16 per Fieldbus Module
*Maximum Accumulation*
65 535 counts or tenths of seconds

**Ladder Diagram Display**

LADDER ROW
Up to eight symbols
USER LABEL FOR A SYMBOL
Up to 14 characters (2 lines)
TECHNICAL IDENTIFIER FOR A SYMBOL
Up to 6 characters
RUNG COMMENT LINE LENGTH
Up to 3 rows of up to 60 characters each
LADDER LINE
4 rows
DISPLAY SCROLL SEGMENT SIZE
One ladder line or rung
MAXIMUM NUMBER OF RUNGS IN FBM
Either 98 lines of logic or approximately 390 ladder logic symbols (elements)

## SEQUENTIAL CONTROL CONCEPTS

Sequential control is one of the control domains of the Integrated Control Software. Sequence, continuous, and ladder logic blocks can be combined within a single compound structure. The sequence control software runs in any I/A Series System station that contains control processor software (excluding the tank processor and some gateways).

Sequential control fulfills the needs of sequential, feedback-oriented applications at the equipment control level.

In the continuous control domain, control blocks have fixed algorithms, a fixed number of parameters, and fixed properties. Additionally, continuous block algorithms can refer only to their own parameters.

In the sequential control domain, on the other hand, you can build your own algorithms using a logical set of parameters within the control compound/block structure. And Sequence block algorithms can read and write other compound/block parameters directly.

A high-level Sequence language and a parameter set provide the necessary tools to build Sequence blocks. When built, these blocks can be combined with continuous and/or ladder logic blocks in a compound (refer to Figure 10).

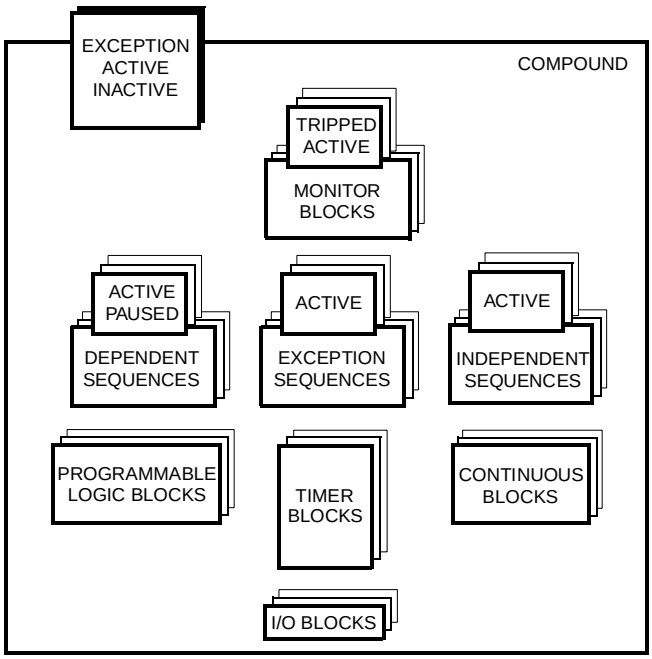



Figure 10. Example of Compound Contents

### Block Classes

There are three classes of blocks: Sequence, Monitor, and Timer.

| | |
|---|---|
| Sequence blocks | Manipulate any compound, block parameter, or shared variable. (A shared variable acts as a linkage between an application and the control database.)<br>Activate other Sequence blocks and Monitor blocks.<br>Send messages to historians. |
| Monitor blocks | Monitor up to 16 process conditions (parameter values and Boolean expressions). |
| Timer blocks | Keep track of time while control strategies are executing. |

### Block Types

Structurally, all sequential control blocks are the same. They differ only in their interaction within a compound. Their type defines this difference. The types are:

| | |
|---|---|
| Dependent Sequence block | (DEP) |
| Independent Sequence block | (IND) |
| Exception Sequence block | (EXC) |
| Monitor block | (MON) |
| Timer block | (TIM) |

Execution of a Dependent Sequence block is automatically delayed while any Exception Sequence block that is nested in the same compound is running.

Execution of an Exception Sequence block, on the other hand, is never delayed.

Execution of the Independent Sequence block does not affect the execution of other sequences nor does the execution of other blocks affect the operation of Independent Sequence blocks.

### Block States

The block states are Inactive, Active, Paused, or Tripped.

The Inactive state means that a Sequence block is not executing statements or that the Monitor block is not evaluating conditions.

The Active state means that a Sequence block is executing statements or that the Monitor block is evaluating conditions.

The Paused state means that a Dependent block is in a suspended condition. Dependent blocks pause whenever an Exception block in the same compound becomes active. The Dependent block becomes active again when the Exception blocks complete their execution.

The Monitor block has a Tripped state (one of its conditions is true). Therefore, a sequence is activated by the Monitor block. All 16 conditions act independently.

The compound parameter SSTATE shows the operational behavior of the Sequence block states within that compound in one of three values:

| | |
|---|---|
| Inactive | Neither the Sequence blocks nor the Monitor blocks nested in the same compound are active |
| Active | One or more Monitor blocks; and/or one or more Dependent Sequence blocks; and/or one or more Independent Sequence blocks that are nested in the compound are active. |
| Exception | One or more Exception Sequence blocks nested in the compound are active. |

**Processing Sequence Blocks**

Sequence blocks can run in parallel with continuous blocks, ladder logic blocks, and each other in that:

- Sequences may be Active concurrently.
- Monitor blocks may be Active in parallel with Sequence blocks.

Timing is an independent feature and can run in parallel with other blocks. The timer block provides four (4) independent timer functions.

Block processing order is:

1. Equipment Control Blocks
2. Pre-Sequence Continuous and PLB blocks
3. Sequence Monitors and timers
4. Sequence Exception Blocks
5. Sequence Dependent and Independent Blocks
6. Post-Sequence Continuous and PLB blocks

**Sequential Control Domain**

Sequence blocks contain logic that supervises the control loops. The logic regulates such things as:

- pressure control
- temperature control
- agitator control
- ingredient fills
- gas control.

Figures 11 and 12 illustrate one example of how you can use Sequence blocks to supervise reactor control flow loops. The intent of this example is to show just a few control loops rather than a complete control strategy.

Figure 11 shows a reactor having two ingredient inputs, an agitator, a condenser, and a heat jacket.

The Figure 12 flowchart shows Sequence blocks within a compound structure coordinating the continuous control loops in the Figure 11 example.

In this example:

1. SEQ_COORD, an Independent block, is coordinating activities in the Dependent blocks. The first action it takes is to activate SEQ_FILL_A and SEQ_FILL_B.
2. SEQ_FIII_A and SEQ_FIL_B start to fill the reactor with two ingredients, concurrently.
3. The SEQ_FILL_blocks then send a set point to the PID block, start the jacket temperature control loop, and start the Monitor block(s) to watch the jacket temperature.
4. SEQ_FILL blocks continue adding ingredients to the reactor. (They no longer need to worry about the jacket temperature alarms since a Monitor block is doing this.)
5. If the jacket temperature exceeds the alarm limits, a Monitor block activates an Exception block to correct/handle the situation.

   When an Exception block is active within a compound, the Dependent blocks within the same compound pause (such as SEQ_FILL_A, etc.). However, Independent blocks continue executing.
6. When the SEQ_FILL blocks reach completion, they turn off the monitors associated only with SEQ_FILL.
7. A given processing phase can be configured to have multiple Monitor-Exception Sequence block combinations, thereby invoking different actions depending upon the particular operation. For example, during the heat-up phase, a process alarm may only need to stop heat and apply full jacket cooling. In another situation, the same alarm might cause full jacket cooling to be applied as well as activating the emergency cooling system.

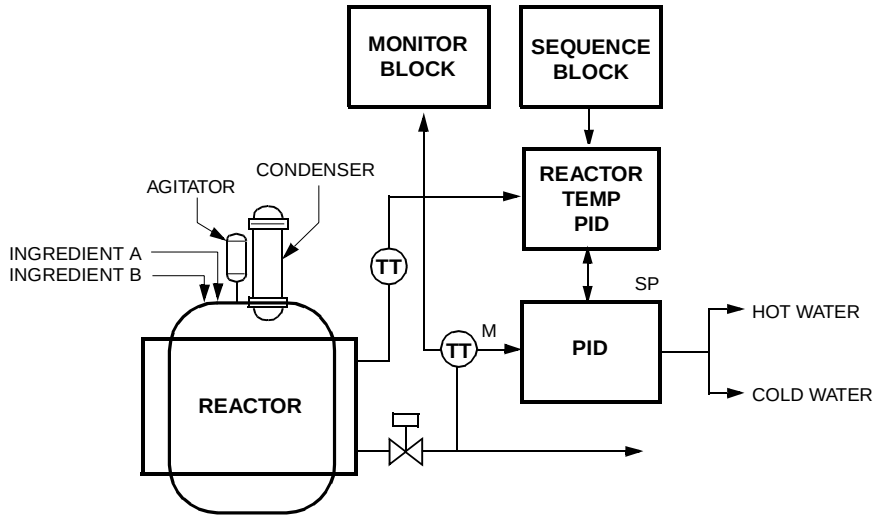



Figure 11. Sample Control Loops

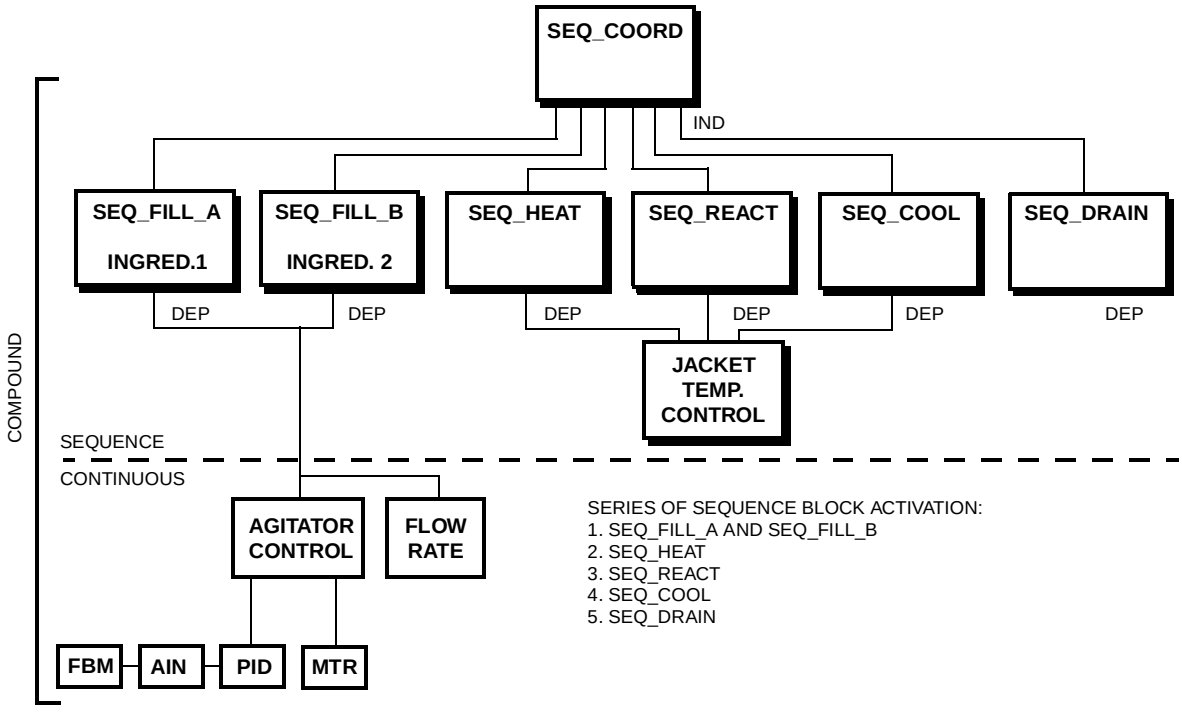



Figure 12. Mixed Compound Samples

### Sequence Language

The Sequence block language is a subset of the I/A Series systems high-level sequential language. It is a structured language somewhat like the programming language, PASCAL. However, its focus is on control applications. The language includes logic flow control statements as well as Boolean and arithmetic functions. Refer to Figure 13 for a sample block built with the Sequence language.

The language statements do not operate the I/O directly. Rather, they make connections between their own parameters and I/O block parameters. They can write the I/O block parameters within continuous, ladder logic, or other Sequence blocks which operate the input/output.

*Logic Flow Control Statements*

These statements determine the flow of control. They may select groups of statements to be executed, skip them, execute them repetitively, or delay their execution. They are:

if. . .then. . .elseif. . .else. . .endif
for. . .to. . .do. . .endfor
repeat. . .until
while. . .do. . .endwhile
exitloop
goto
wait. . .time
wait. . .until condition
exit
retry

```
EXCEPTION SEQUENCE
                                    (*****************************************)
                                    (          EXCEPTION SEQUENCE    )
                                    (            CONTROL BLOCK           )
                                    (             FOR REACTOR            )
                                    (*****************************************)
                                    (*****************************************)
                                    ( Specify the user-labeled parameters in     )        NOTE: THE DELIMITERS,
                                    ( one of the following formats:              )        {...} AND (*...*)
                                    (   user_name : RInnnn;  (real input)        )        ENCLOSE COMMENTS
                                    (   user_name : IInnnn:  (integer input)     )        AND OPERATOR REMARKS,
                                    (   user_name : BInnnn;  (Boolean input)     )        RESPECTIVELY. COMMENTS
                                    (   user_name : ROnnnn;  (real output)       )        ARE USED TO DOCUMENT
                                    (   user_name : IOnnnn;  (integer output)    )        THE PROGRAM. REMARKS
                                    (   user_name : BOnnnn;  (Boolean output     )        EXPLAIN PROGRAM ACTIONS
                                    (   user_name : SNnnnn:  (string name)       )        TO AN OPERATOR.
                                    (*****************************************)
EXP2SS : SN1;
                                    (*****************************************)
                                    ( The declaration part finishes here.  )
                                    ( Now enter the block's statements.  )
                                    (*****************************************)
                                    (* EXCEPTION LOGIC FOR JACKET WATER PUMP FAILURE *)
                                    (* SHUT STEAM VALVES AOV_X15A AND AOV_X15B *)
                :REACT_CONT:AOV_X15A.CLOSE :=TRUE;
                :REACT_CONT:AOV_X15B.CLOSE :=TRUE;
                                    (* PUT CONTROLLER TIC_X03 IN MANUAL *)
                :REACT_ANLG:TIC_X03.MA :=FALSE;
                                    WAIT .1;
                                    (* SET OUTPUT OF TIC_X03 TO 0.0 *)
                :REACT_ANLG:TIC_XO3.OUT :=0.0;
                                    (* SEND ALARM TO ANNUNCIATOR VIA BOOLEAN BLK *0
                :REACH_SEQ:EXCP2_MSG.IN_1 :=TRUE;
                                    (* SEND MESSAGE TO THE OPERATOR *)
                                    SENDMSG ("WATER PUMP NOT RUNNING UNIT BEING SHUT DOWN") TO EXP2SS;
                                    (* PROCEED TO SHUT DOWN THE UNIT *)
                                                .
                                                .
                                                .
END SEQUENCE
```

Figure 13. Sample Block Using the Sequence Language

### Data Operation Statements

There are two kinds of statements that can manipulate data: the Assignment statement and the Procedural statement.

The Assignment statement replaces the current value of some object with a new value that results from evaluation of an expression.

The Procedural statements are:

| | |
|---|---|
| ACTIVATE | Activates a Sequence block or a Monitor. |
| ABORT | Aborts an active Sequence block or Monitor. |
| START_TIMER | Starts timers at current value or select value. |
| STOP_TIMER | Stops timers. |
| ACTCASES | Manipulates activity of the 16 Monitor block cases. |
| SENDMSG | Initiates a message from executing sequence logic. It can address any object that acts like a logical device, such as historians or annunciator keys. It can also assign a message to a string parameter. |
| SEND CONF | Sends messages interactively to logical devices or objects that act like logical devices. A message received confirmation is expected. |

### OPERATOR INTERFACE INTEGRATION

For each and every block (including ECBs) created by the user through the control and I/O configurator, a Block Detail Display with a standard block faceplate is created allowing the user immediate access to block parameters and functions without having to separately configure or specify a faceplate-type. Figure 14 shows the various types of default displays created.

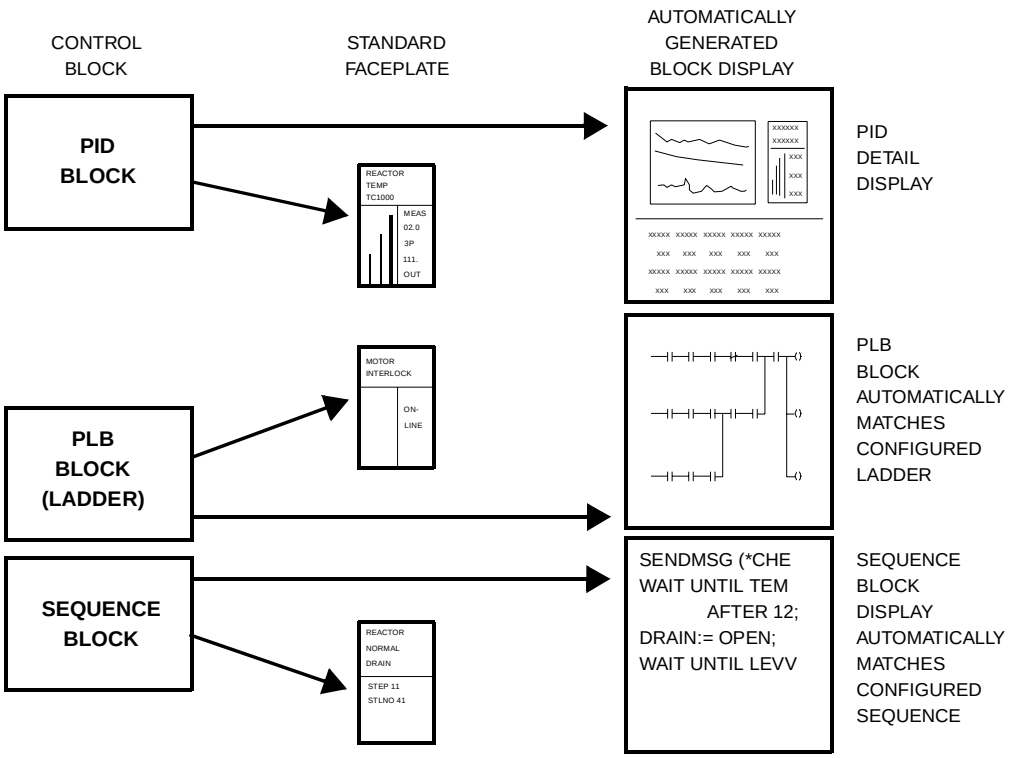



Figure 14. Automatically Generated (Default) Block Detail Displays

Figure 15 illustrates a typical Block Detail Display for a PIDA block. The “SRC” (source) button provides tracing, debugging, and checking of control connections.

Although the system creates certain displays automatically, you can use the Display Builder and Display Configurator (accessible from the Process Engineer's Environment, see Figure 5), to create custom process displays. For instance, you can import individual faceplates and trends as shown in Figure 16.

The 50 Series and NT-based displays allow you simultaneous access to I/A Series windows as well as to non-I/A Series programming and third party application windows. For example, you can view and work within process control displays while working in the Integrated Control Configurator or non-I/A Series software packages, thereby improving productivity.

Optionally available for some workstations are multiple I/A Series Display Managers. Multiple Display Managers provide a more comprehensive view of the process by supporting display windows for operator control as well as for plant and management personnel. Each Display Manager Window functions independently. Access to multiple I/A Series functions and application packages is greatly increased.

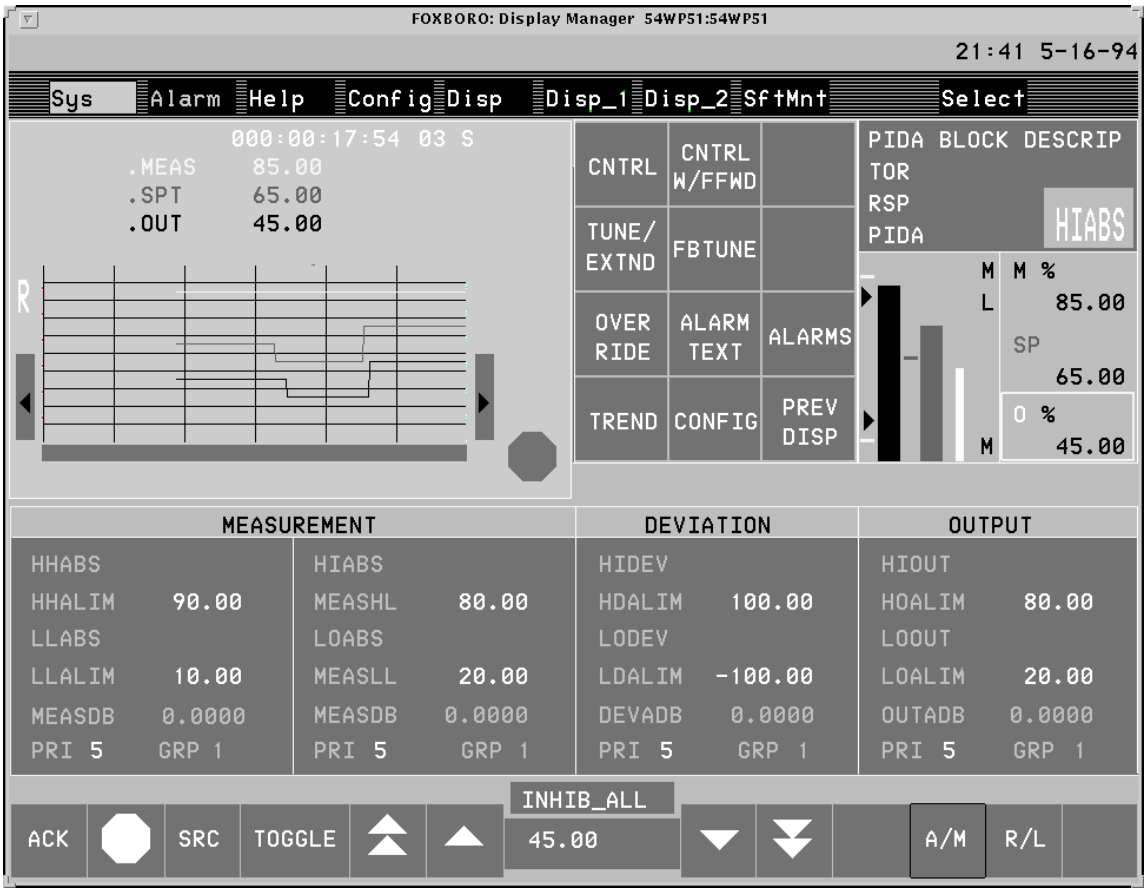



Figure 15. Example of Block Detail Display for PIDA Block

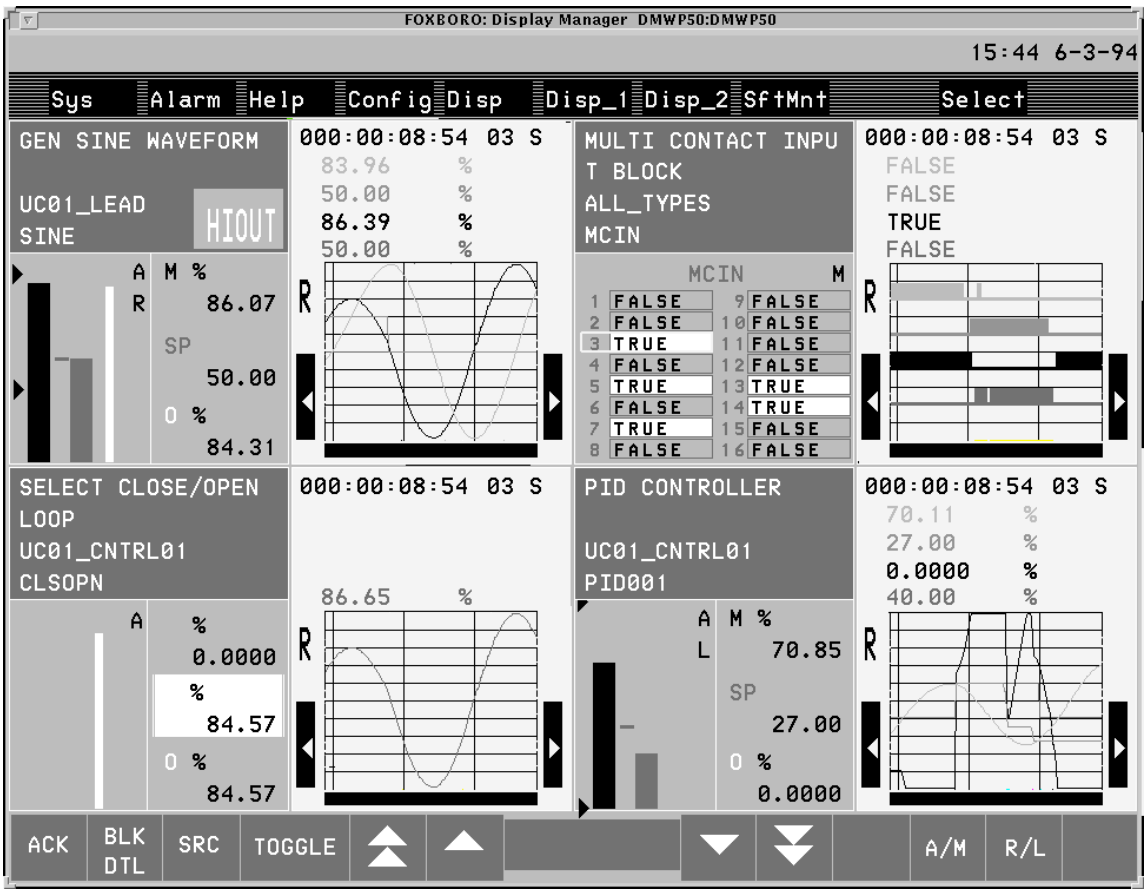



Figure 16. Typical Group of Faceplates and Trends

**The Foxboro Company**
33 Commercial Street
Foxboro, Massachusetts 02035-2099
United States of America
http://www.foxboro.com
Inside U.S.: 1-508-543-8750 or 1-888-FOXBORO (1-888-369-2676)
Outside U.S.: Contact your local Foxboro Representative.



MB 021 Printed in U.S.A. 1296