Honeywell

UMC800 Controller RS232 Communications Reference Manual

51-52-25-76B

8/99

ATTENTION

THE SOFTWARE WITH THIS MANUAL IS PROVIDED FREE OF CHARGE. IT IS TO BE USED AS-IS, WITH NO LIABILITY OR SUPPORT FROM HONEYWELL.

TotalPlant

Copyright, Notices, and Trademarks

Revision B – 8/99

Honeywell
Industrial Automation and Control
Automation College
1100 Virginia Drive
Fort Washington, PA. 19034

About This Document

Abstract

This Reference Manual contains detailed information regarding RS232 Communications for the UMC800 Controller. It provides a descriptive breakdown of the message formats in the data exchange between a computer and the UMC800 controller. There is a Table of Contents listing the accessible function blocks in alphabetical order. Each Function Block includes tables for the dynamic and static parameters that can be accessed through RS232 Communications.

References

Honeywell Documents

The following list identifies all Honeywell documents that may be sources of reference for the material discussed in this publication.

Document Title	ID #
UMC800 Controller Technical Overview	51-52-03-24
UMC800 Controller Installation and User Guide	51-52-25-61
UMC800 Operator Interface User Guide	51-52-25-62
UMC800 Control Builder User Guide	51-52-25-63
UMC800 Control Builder Function Block Reference Guide	51-52-25-64

Contents

Tables and Figures

1. UMC800-RS232 Communications

1.1 Overview

Purpose of this manual

This manual provides information about request and response messages used to exchange information between a personal computer and a UMC800 Universal Multiloop Controller using the Leaderline Control Builder RS232 communications port.

How the manual is structured

In the first section, a full RS232 Protocol description for message exchange is provided for those wishing to write a driver in a specific programming language.

For those specifically using Microsoft Visual C++ on Windows 95 or NT, a driver code library is provided on a separate diskette (supplied with the manual) allowing direct linkage to C++ programs without further driver development. The PC Communications Driver section describes use of this library.

A “peek & poke” EXE utility file (with source code) is also provided on the diskette, which can be used to experiment with UMC 800, RS232 communications to a Controller and can also serve as a guide in general development using another language

Reader assumptions

We assume that you have had prior experience with programming and interpreting communications protocols. Also, that you are familiar with UMC800 Controller Function Blocks and the LeaderLine Control Builder Control Strategy Configuration concepts.

How to use this manual

Use this manual as a reference dictionary of UMC800 controller communications message formats for its application programs. It is not intended as a comprehensible “how to” documents for initiating computer communications with a UMC800 Controller.

In this manual

These major topics are covered in this manual.

2. RS232 Port Protocol Description

2.1 Overview

This section of the manual describes the RS-232 “Configuration” port protocol for read/write access to parameters in the UMC 800 Controller. It is intended for those not desiring to use the Microsoft Visual C++ sample driver library provided and wish to use another programming language. The methods described in the PC Communications Driver section along with the “peek & poke” utility file provided on diskette may be used as a guide in driver development.

The protocol is loosely based on ANSI X3.28, utilizing asynchronous data transmission, DLE insertion for data transparency prior to control characters, a two-byte CRC, and 32-bit 4-byte IEEE data formats. Data Parameter access is via the function block number (1 through 249) and the index number of the parameter within the block. Variables are addressed as block 250 with index numbers 0 through 149 (corresponding to Variables 1 to 150).

The protocol allows for multiple parameter access in a read request message packet to increase data throughput. This can be done by accessing a function block number and a contiguous range of parameter indexes within that block or you may use a “scattered” message format to address multiple blocks and indexes in any order. For the scattered message block method, up to 60 parameters may be read at once. Writes are usually on a single block number, index basis but may also be on a multiple parameter basis using the “scattered” message format (Maximum 40 or maximum packet size of 250 bytes). For instance, this may apply to a range of Variables.

The link level data transmission format is to be fixed at 8 data bits, no parity, one stop bit and 9600 baud.

2.2 Definition of Error Types

Error Type	Definition
Application	The instrument rejected the message with an error code because the request was invalid, bad sequence number, bad address.
Link	Errors due to noise on the link, i.e. bad CRC.
Physical	Errors which are trapped by the system or the hardware, i.e. timeout, bad parity, bad framing.

2.3 DLE Insertion and Deletion

A DLE character will precede control characters. Any DLE that exists in the fields between the packet header and footer will have an extra DLE inserted in front of it, which will be stripped out by the receiver, similar to the zero insertion and deletion in HDLC protocol. On transmit, the CRC will be calculated before DLE insertion is performed. On receive, DLE deletion will be performed before the CRC is checked.

2.4 CRC

The 16 bit cyclic redundancy is computed using a CRC-16 algorithm (as opposed to the CCITT CRC algorithm). It will be placed in the packet LSB first. The check will include everything between the packet header and footer. It will not include any inserted DLE characters. This is because the DLE insertion must protect the CRC in the event that CRC == DLE|STX or CRC == DLE|ETX.

For the CRC-16 Calculation, refer to Appendix A.

2.5 Sequence Numbering

The sequence number is a serial number that uniquely identifies a message. The host generates an arbitrary sequence number when it sends a request to an instrument. The instrument echoes this number in its reply. This is so that the host can verify that the instrument is replying to the correct message.

Imagine the following scenario: the host sends a message with sequence number 5 to the instrument. However, due to a firmware update, it now takes the instrument 10 seconds to reply to this particular message. After 3 seconds, the host times out. A few seconds later, the host sends a different message with sequence number 6 to the instrument. By this time, the instrument has just finished processing message number 5 and sends the reply. The host is able to detect the out-of-synch error because the request had sequence number 6 but the reply had sequence number 5.

Note that it is not necessary for the sequence numbers to be in any sort of order. Therefore, for the sake of efficiency, the host may optionally skip sequence number 16 to unburden the instrument from having to DLE-strip the sequence number. ***This is not a feature of the protocol***. The instrument should not assume that it would never get a sequence number 16.

2.6 Response to Link and Physical Errors

The instrument will not respond to bad incoming messages in any way. It will be the host's responsibility to detect the timeout and retransmit the message. The host will respond to bad replies from the instrument by retransmitting the message without incrementing the sequence number, as mentioned before.

2.7 Reserved Addresses

Unit address 1 is reserved for broadcast messages, which all of the instruments on the link will obey, but none of the instruments will reply to. Unit address 255 is reserved for the host.

2.8 Static and Dynamic Parameters

There are two types of parameters that can be accessed:

STATIC – a static parameter is a configuration parameter

DYNAMIC – A dynamic parameter is an Input/Output parameter.

Each of these parameter types has a specific index number set.

Configuration parameters (Static) and I/O parameters (Dynamic) cannot be accessed in the same message since both the Static parameters and the Dynamic parameters each start at Index Number 0.

2.9 Anatomy of a Message Packet

Symbol key:

Symbol	Value	Meaning
DLE	0x01	Precedes control characters.
STX	0x02	Start of transmission.
ETX	0x03	End of transmission.
DADR	 1 0xff	Destination address. Controller Host
SADR	 1 0xff	Source address. Controller Host
SEQ	n/a	Sequence number.
CRC	n/a	Cyclic redundancy check.
ANAK	0x09	Application level negative acknowledge
AACK	0x0a	Application level acknowledge.
ERRCODE	n/a	Error code explaining why instrument rejected message. (See Subsection 2.11- Table 2-1)

Host to Instrument

DLE|STX|DADR|SADR|SEQ|...message*...|CRC|CRC|DLE|ETX

***** *Refer to Section 4 - Message Exchange Formats*

Instrument to Host

(1) If message was good and the instrument understood the message

DLE|STX|DADR|SADR|SEQ|...message*...|CRC|CRC|DLE|ETX

***** *Refer to Section 4 - Message Exchange Formats*

(2) If message was good but the instrument rejected the message (i.e. application level failure)

DLE|STX|DADR|SADR|SEQ|ANAK|ERRCODE (see table 2-1)|CRC|CRC|DLE|ETX

(3) If message was bad

Silence.

2.10 Message Exchange Examples

2.10.1 Read Contiguous 32-bit Values

Below is an example of a Read Contiguous 32-bit Value message exchange.

Example

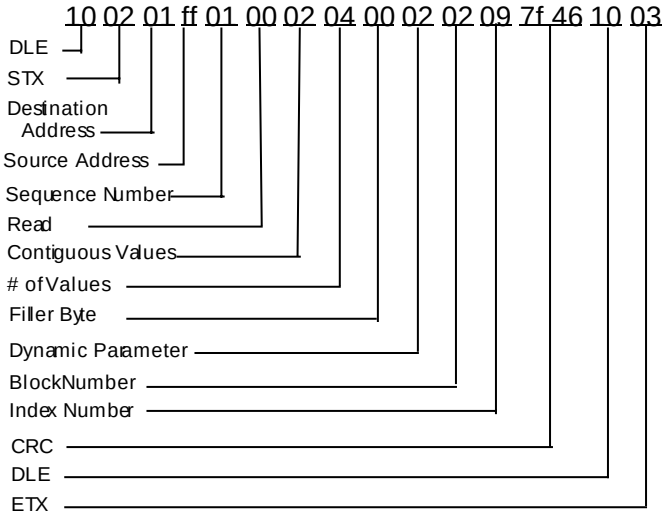
Read of PV, WSP (Working (or active) Set Point), Output, and Mode from the controller assuming the function block number for the PID block is 2 (PID2) starting at Index #9.

References

See Subsection 2.9 and Subsection 4.21(Figure 4-1) for formats.
See Subsection 5.47 for PID block dynamic parameters and indices.

Request Message

Host to UMC 800 Controller

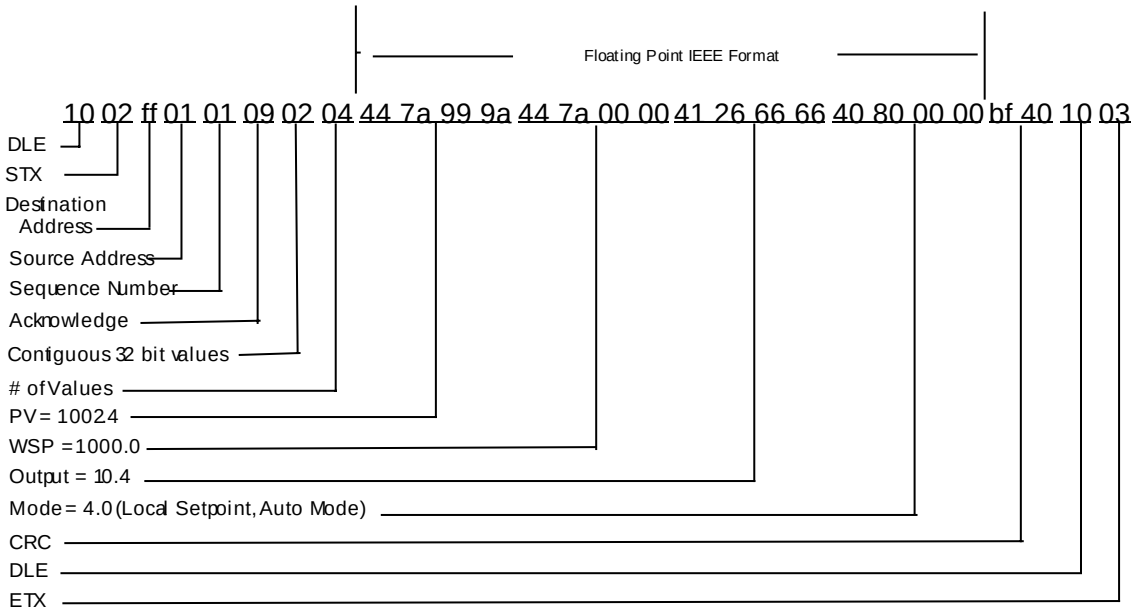


2.10.1 Read Contiguous 32-bit Values, *continued*

Response Message

Controller to Host: ***(good message)***

Assume values are PV=1002.4, WSP = 1000.0, output = 10.4, and mode = 4.0 (Local SP, Automatic)



Mode is decoded as follows (also see Sect 2,48 of Manual 51-52-25-64):

0.0= RSP AUTO
1.0= RSP MAN
2.0= RSP Initialization Manual
3.0= RSP Local Override
4.0= LSP AUTO
5.0= LSP MAN
6.0= LSP Initialization Manual
7.0= LSP Local Override

Controller to Host ***(bad message, with a typical error code)***

10|02|FF|01|0a|01|7e|80|10|03

2.10.2 Read Scattered 32-bit Values

Below is an example of a Read Scattered 32-bit Value message exchange.

Example

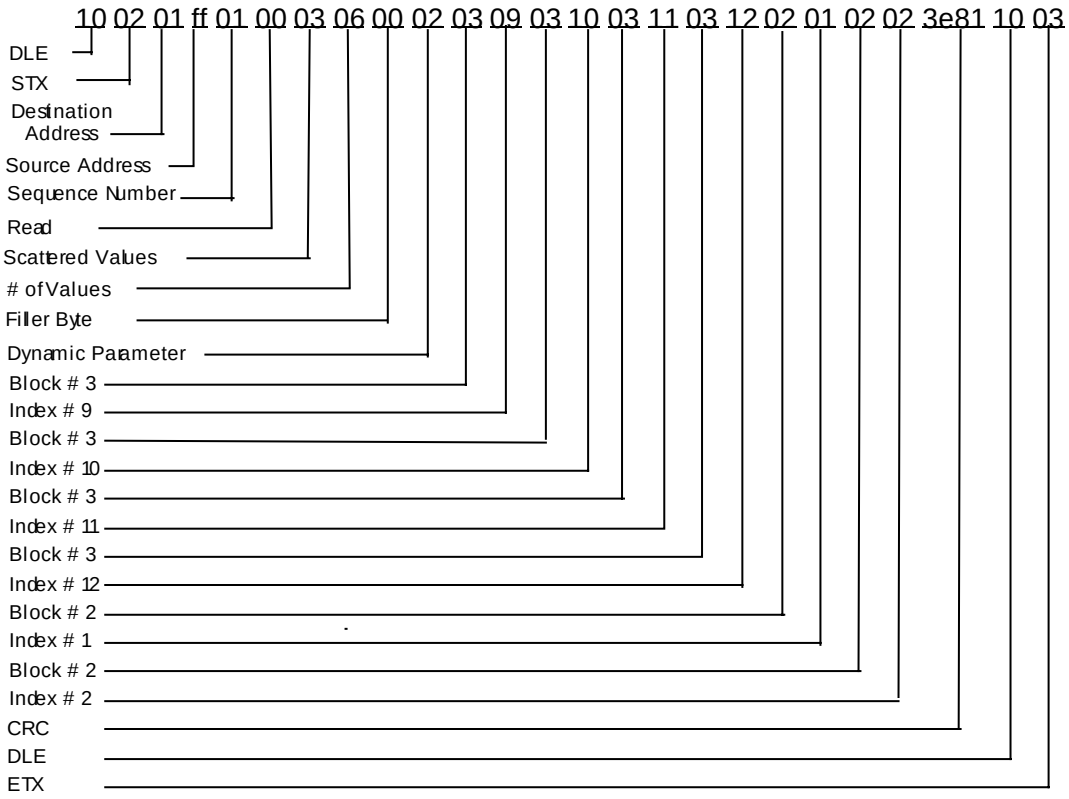
Read of PV, WSP (Working Set Point), Output, and Mode from the controller assuming the function block number for the PID block is 3 (PID3) plus Read the value of an Analog Input (assume block number is 2) and its failed input status, six parameters total.

References

See Subsection 2.9 and Subsection 4.3 (Figure 1) for formats.
See Subsection 5.5 for Analog Input block and Subsection 5.45 for PID block dynamic parameter references.

Request Message

Read message request from Host to UMC 800 (in hex):



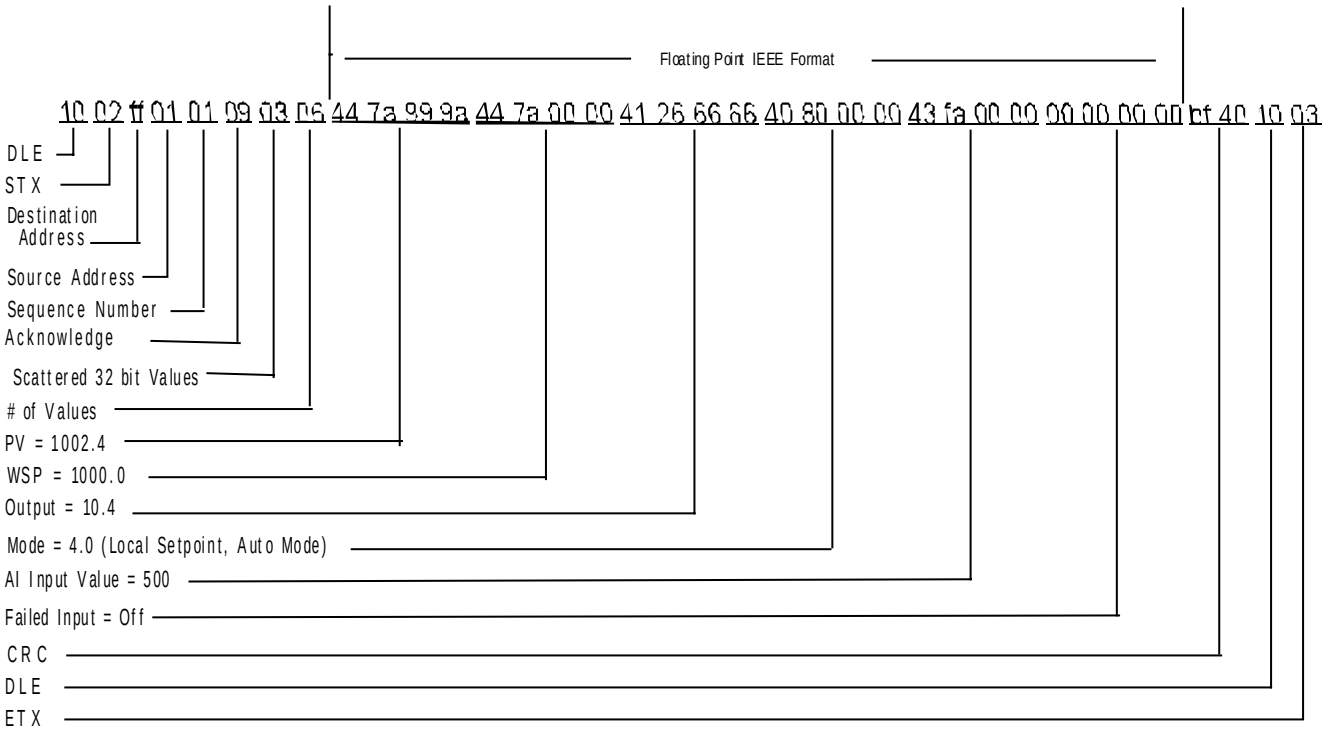
continued next page

2.10.2 Read Scattered 32-bit Values, continued

Response Message

Controller to Host: ***(good message)***

Assume values are PV=1002.4, WSP = 1000.0, Output = 10.4, Mode = 4.0 (Local SP, Automatic), AI Input Value = 500, Failed Input = off.



Controller to Host ***(bad message, with a typical error code)***

10|02|FF|01|0a|01|7e|80|10|03

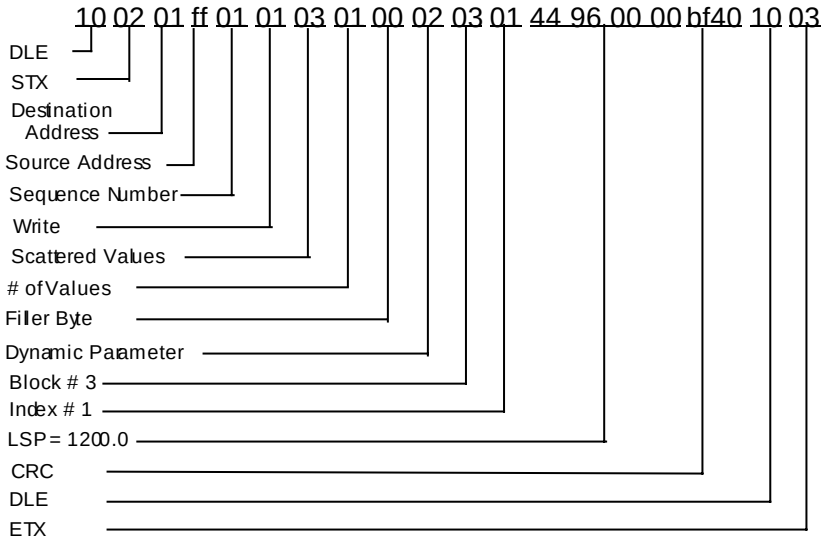
2.10.3 Write Scattered 32-bit Values

Below are examples of a Read Scattered 32-bit Value message exchange.

Example 1

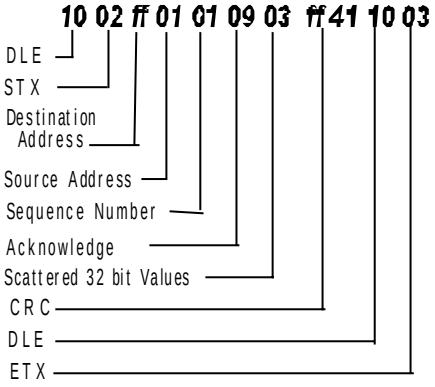
Write a local Set Point of 1200.0 to a controller PID block, assuming block number is 3.

Request Message



Response Message

Controller to Host: ***(good message)***



Controller to Host (***bad message, with a typical error code)***

10|02|FF|01|0a|01|7e|80|10|03

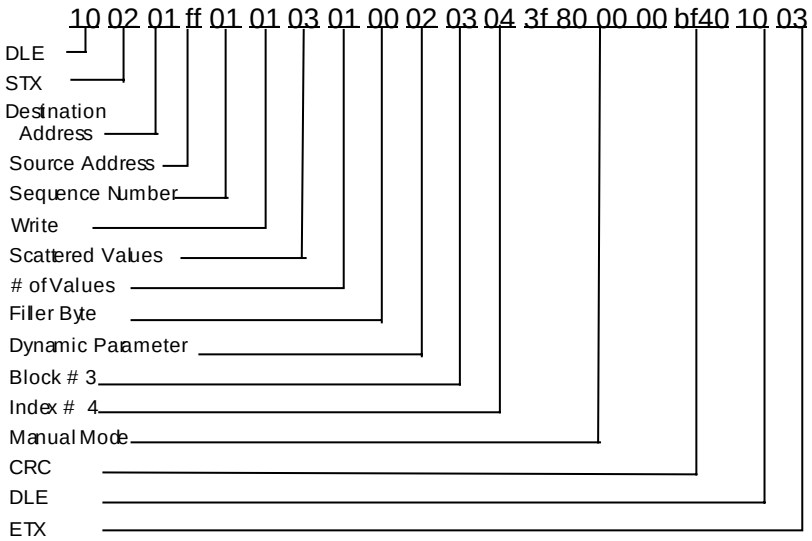
2.10.3 Write Scattered 32-bit Values, continued

Example 2:

Write the mode of a controller PID block to Man mode, assuming block number is 3.

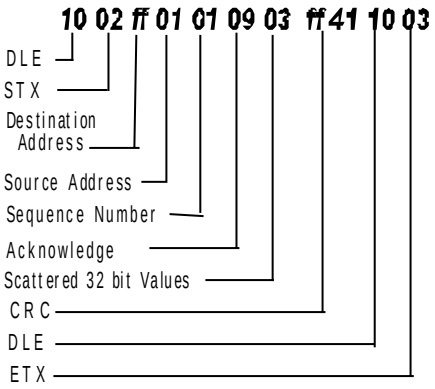
Request Message

Write message request from Host to UMC 800:



Response Message

Controller to Host: ***(good message)***



Controller to Host ***(bad message, with a typical error code)***

10|02|FF|01|0a|01|7e|80|10|03

2.11 Application Error Codes

List of Codes

These codes are returned when ANAK is received in the response for a request that is rejected.

Table 2-1 Application Error Codes

Error Code	Definition
0	No Errors
0x1	Bad Command
0x2	Bad Operation
0x4	Bad data for the Command
0x8	Not Used
0xA	Busy Doing Command – Try Again Later
0x10	Invalid Header Byte
0x11	Bad Message Length
0x20	Bad Database Download Sequence
0x21	Invalid Database Table Number
0x22	Bad Record Number
0x23	Table Number Changed in the Middle of Download
0x24	Database Revision Level Mismatch
0x25	Bad Packet Size
0x26	Incomplete Record
0x27	SPP Database Table too big
0x30	Setpoint Program Segment Overrun
0x31	SPP – Invalid Mode for Editing (Not Reset or Hold)

2.12 Sequence of Operations

On Send:

1. Insert address and sequence fields.
2. Calculate and append CRC.
3. Do DLE-insertion.
4. Add packet header and footer.

On Receive:

1. Strip packet header and footer.
2. Do DLE-deletion.
3. Verify and strip CRC.
4. Verify and strip address and sequence fields.

3. PC Communication Driver for RS232 Port

3.1 Overview

General Information

The purpose of this communication driver is to support RS232 communication protocol. The communication driver will be implemented as a .LIB file. The physical reads and writes will be done via the ReadFile and WriteFile AFX messages, which are supported in Windows 95 and Windows NT. The driver will not be backward compatible with Windows 3.1. Communication is in non-overlapped mode, which means that the API functions will not return until a reply has been received or the port times out.

Utilization

This section explains how to utilize a Honeywell-supplied communication driver to access controller data on a PC. The user must link the Honeywell supplied code library with a program of their own design using Microsoft Visual C++ Version 5.0. The driver exposes a C++ interface and is useful only with C++ programs.

Disk Contents

The disk, which accompanies this document, contains an EXE (with source code) for a simple "peek & poke" utility. It forms a sample of how to use the driver library and provides a useful tool for experimenting with UMC800 communication.

Reads and Writes

The utility allows a user to read and write values of function block parameters. For simplicity, the utility assumes that you will only write to parameters which are floating-point numbers. UMC800 Boolean (on-off) parameters are floating point numbers with FALSE represented as 0.0 and TRUE represented as 1.0. The utility represents a value read from the instrument as a floating-point number in the Value field and as a raw byte stream in the Hex field.

PeekNPoke

Since this is a dialog-based application, most of the code which interests the programmer is in the module PeekNPokeDlg.cpp. Of particular interest are the Read and Write button handlers, OnRead and OnWrite, and the utility functions FormatCommErrString and XlateErrCode.

Hierarchy of Layers

The driver will consist of the following layers:

The API layer

- Interfaces with the application via API functions.
- Keeps track of unit address and message sequence number.
- Traps errors that result from the instrument rejecting the message, replying to the wrong message, or the wrong instrument replying.
- Performs retries.

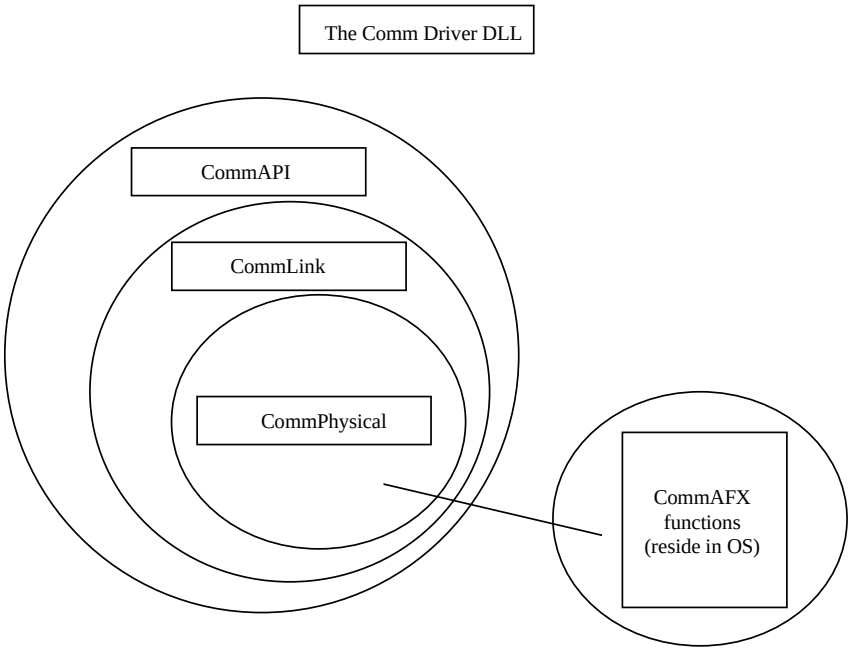
The Link Layer

- Computes/verifies the CRC.
- Performs DLE insertion/deletion.
- Appends/strips headers and footers.
- Traps link level errors (i.e. bad CRC).
- Runs the receive polling loop that contains the logic for recognizing packet delimiters.

The Physical Layer

- Hides the operating system AFX functions from the link layer.
- Traps physical errors (i.e., parity/framing errors, timeout errors).

In general, commands are handed down from layer to layer until they reach the physical layer, while errors are bubbled back up to the API layer.



3.2 For the Programmer - Getting Started Quickly

3.2.1 How Do I Use This Driver?

This section contains instructions for getting started if you want to use this driver in your project.

- Copy the following files into your project directory: **Elnapi.h, Elnlink.h, Elnphys.h, Commbase.h, Elndrv.lib.**
- Add **Elnapi.lib** as a dependency for your project.

3.2.2 Code sample 1: instantiation of a driver object on the stack

```
    #include “Commbase.h”
    #include “Elnapi.h”

    void SomeClass::SomeFunction()
    {
      unsigned char pbyTxMsg[64];
      unsigned char pbyRxMsg[64];
      int nTxLength;
      int nRxLength;
      long lRet;
   CElnApi CommDriver(        “COM2”,  // Connect driver to com2.
                              5000,    // 5 sec. timeout.
                              5,       // 5 retries.
                              2000);   // 2 sec. delay between retries.

   lRet = CommDriver.SetUpPort(      9600,   // 9600 baud.
                                     FALSE,  // No parity.
                                     8,      // 8 data bits.
                                     1);     // 1 stop bit.
   if(lRet)
   {
      // Error handling goes here.
   }

 sprintf(pbyTxMsg, “My dog has fleas.”)

 lRet = CommDriver.OpenPort();
 if(lRet)
 {
       /*
       Error handling goes here.  Chances are another application has locked
       out access to the port.
       */
 }

lRet = CommDriver.TransCeive(      pbyTxMsg,    // Outgoing message ptr.
                                   pbyRxMsg,    // Rx buffer.
                                   17,          // Outgoing msg length.
                                   64,          // Rx buffer size.
                                   &nRxLength,  // Actual length of Rx.
                                   1);          // Unit addr. 1.
 if(lRet)
```

continued next page

Code sample 1: instantiation of a driver object on the stack, *continued*

```
{
    // Error handling goes here.
}

lRet = CommDriver.ClosePort();
if(lRet)
{
      // Error handling goes here.
   }
}
```

3.2.3 Code sample 2: dynamic instantiation of one of two driver objects on the heap

```
#include “Commbase.h”
#include “Elnapi.h”

void SomeClass::SomeFunction()
{
   unsigned char pbyTxMsg[64];
   unsigned char pbyRxMsg[64];
   int nTxLength;
   int nRxLength;
   long lRet;
   CCommBase* pDriver;

   // Dynamically instantiate an EL+N driver object.
   pDriver = new CElnApi(            “COM2”, // Connect driver to com2.
                                     5000,   // 5 sec. timeout.
                                     5,      // 5 retries.

   if(!pDriver)
   {
      // Error handling goes here.
   }

   /*
   From this point on we don’t care what kind of driver it is because
   all driver objects that are derived from the CCommBase base class
   have the same public API functions.
   */

   lRet = pDriver->SetUpPort(       9600,     // 9600 baud.
                                    FALSE,    // No parity.
                                    8,        // 8 data bits.
                                    1);       // 1 stop bit.
   if(lRet)
```

continued next page

Code sample 2: dynamic instantiation of one of two driver objects on the heap, *continued*

```
{
      // Error handling goes here.
   }

   sprintf(pbyTxMsg, “My dog has fleas.”)

   lRet = pDriver->OpenPort();
   if(lRet)
   {
      /*
      Error handling goes here.  Chances are another application has locked
      out access to the port.
      */
   }

   lRet = pDriver->TransCeive(       pbyTxMsg,    // Outgoing message ptr.
                                     pbyRxMsg,    // Rx buffer.
                                     17,          // Outgoing msg length.
                                     64,          // Rx buffer size.
                                     &nRxLength,  // Actual length of Rx.
                                     1);          // Unit addr. 1.
   if(lRet)
   {
      // Error handling goes here.
   }

   lRet = pDriver->ClosePort();
   if(lRet)
   {
      // Error handling goes here.
   }

   if(pDriver)
      delete pDriver;   // Destroy the comm driver object.
}
```

3.2.4 Code sample 3: error handling

```
if(lRet)
{
   if(lRet & ERR_RX_OVERRUN)
      // Receive overrun error.

   else if(lRet & ERR_LINK)
      // Link error: bad CRC indicates garbled data.

   else if(lRet & ERR_PHYSICAL)
      // Physical error: timeout, baud rate, parity or framing.

   else if(lRet & ERR_APPLICATION)
```

```
   {
      // Application error.
      if(lRet & ERR_CODE)
         // Instrument rejected message and returned error code.
         ErrorCode = (int)(lRet & ERR_CODE);
      else
         // Bad address or sequence number in return packet.
   }
}
```

3.2.5 Multiple Instances of the Driver in the Same App

This driver is completely re-entrant. Suppose there are two separate communication interfaces on COM1 and COM2 respectively for two UMC800 controllers. The app can handle this by creating two instances of the driver class and attaching them to separate ports:

```
#include "commapi.h"

CElnApi Drv1("COM1",   // COM1.
             2000,     // 2000 mS timeout.
             5,        // 5 retries.
             1000);    // 1000 mS Delay between retries.

CElnApi Drv2("COM2",   // COM2.
              2000,    // 2000 mS timeout.
              5,       // 5 retries.
              1000);   // 1000 mS Delay between retries.
```

3.2.6 Port Lockout

The communication driver relies on the operating system to perform port lockouts in the event that two concurrently running applications attempt to access the same port simultaneously. For example, a configuration utility and a live data viewing utility may each instantiate a CElnApi object, and both CElnApi's may be attached to the same port. However, they may not both open the port simultaneously, because the operating system will not issue a valid port handle to app (a)'s driver before app (b)'s driver has closed the port and returned its handle to the system. This means that if app (a) is still talking when app (b) tries to start talking, app (b) will get a physical error from CElnApi::OpenPort(). App (b) can then go into a polling loop waiting for the port to become available, or it can issue an error message to the user.

Always check the return value of this CElnApi::OpenPort(). If someone else has already opened the port, you cannot get access to it until they close the port. When you are done communicating, close the port. All other threads, including other apps, will be locked out as long as you have the port open.

3.2.7 Managing Multiple Threads

Do not pass a pointer to the same CElnApi object to two concurrently running threads. This will defeat the system's port lockout capability and may cause fatal errors.

If you want to do an upload or download as a background routine, spawn a thread, then allow the thread to instantiate a CElnApi object that is attached to the same port as the app's CElnApi object. The two objects will coexist harmoniously despite being attached to the same port. However, TheApp.CElnApi.OpenPort() will fail as long as the thread has access. Note also that if the app doesn't close the port before spawning the thread, the thread will not be able to open the port.

3.3 Taxonomy of Classes

CCommBase is the base class from which all driver classes (CXxxApi) are derived. CCommBase contains virtual implementations of all of the API functions that are common to all drivers. CCommBase only exists so that a CCommBase pointer can be cast as a CElnApi pointer, a CLnApi pointer, a CModbusApi pointer, etc. The implementations of the API functions in CCommBase don't do anything, though, so don't instantiate a CCommBase object. It won't be useful for anything.

3.4 API Functions Common To All Drivers

CElnApi:: CElnApi(char* pszCommPort, int nTimeOut, int nRetry, int nRetryDelay);

pszCommPort:The name of the communication port you wish to attach the driver to (e.g. "COM1", "COM2", etc.).

nTimeOut: Length of time (in mS) driver will wait for an instrument to reply.

nRetry: Number of repeats in the event of a communication failure.

nRetryDelay: Length of time (in mS) to wait between retries.

long CElnApi::SetUpPort(int baud, BOOL parity, int data, int stop);

baud: Baud rate.

parity: TRUE for parity, FALSE for no parity.

data: Number of data bits.

stop: Number of stop bits.

You cannot call this function while the port is open.

long CElnApi::OpenPort();

Call this function to get access to the port. If an error is returned, another app has probably locked out the port.

long CElnApi::ClosePort();

Call this function when you are done communicating to relinquish ownership of the port.

continued next page

3.4 API Functions Common To All Drivers, *continued*

long CElnApi::TransCeive(unsigned char* pbyTxMsg, unsigned char* pbyRxMsg, int nMaxTx int nMaxRx, int* pnActualLength, int nUnitAddr);

pbyTxMsg:	Ptr to the outgoing message.
pbyRxMsg:	Buffer ptr for incoming message.
nMaxTx:	Outgoing msg length.
nMaxRx:	In buffer size.
pnActualLength:	Return value in disguise; length of incoming message.
nUnitAddr:	Address of unit.

This function sends a message to an instrument and receives the response. Make sure that the buffer that pRxMsg points to is big enough to hold the instrument's reply. If the reply is longer than MaxRx, it will be truncated and a receive buffer overflow error will be returned.

Although the protocol makes it theoretically possible to transmit new messages while outstanding messages are pending, this mode will be unsupported for the sake of simplicity

void CElnApi::SoftReset()

This function performs a reset of the port hardware. It shouldn't be necessary to use it, but it has been necessary in the past when dealing with some particularly ornery SCC chips.

3.5 Anatomy of a Standard Return Value

0 indicates success.

Bit 0-7: Error code returned from instrument if application error occurred.

Bit 8: Receive buffer overrun.

Bit 9: Application error; instrument returned ANAK.

Bit 10: Link error; bad CRC.

Bit 11: Physical error; bad framing or parity, timeout.

3.6 Frequently Asked Questions

"Why do I get a stream of zeroes out of the port when I try to write to it?"

If the DCB (device control block) of the port has not been written since the computer was booted, you will get garbage out of the port. CElnApi::SetUpPort writes the DCB; call it at least once and the problem should disappear. Note: you cannot set up an open port.

"Why do I get physical errors every time I call CElnApi::TransCeive?"

Did you remember to open the port before attempting to transmit? Did you remember to check the return value from CElnApi::OpenPort? Don't assume that OpenPort will succeed. If you try to open a driver that is attached to COM1 and your mouse is on COM1, the system will not give you a handle to the port. Ditto if another concurrently running app or thread is using the port.

"Why does my message from the computer to the instrument get cut off in the middle?"

The timeout (the second argument in the constructor) is used both as a receive timeout and a transmit timeout. In other words, when you call transceive, the message from the computer to the instrument must be sent in less than timeout milliseconds. Otherwise, NT will assume that something has gone awry with the UART. When the instrument replies, the reply must be received in less than timeout milliseconds. Otherwise, we assume that the instrument that we are talking to is not there.

When calling the constructor, consider the longest message you expect to send and the longest reply you expect to receive. Set timeout to a number slightly greater than the larger of the two.

"Can two CElnApi objects exist simultaneously, each on a different port?"

Yes.

"Can two CElnApi objects exist simultaneously, both on the same port?"

Yes, but only one CElnApi object will be able to open the port at a time. This is true whether or not the same thread owns both objects.

"Can I spawn a thread, then pass it a pointer to the app's comm driver?"

No. Two concurrently running threads cannot access the same CElnApi object simultaneously. Instead, pass the constructor and initialization arguments to the thread and let it instantiate its own CElnApi object. The thread's driver can be attached to the same port as the app's driver, but only one of them will be able to open the port at a time. In fact, as a general rule, Windows NT disallows the practice of two concurrently running threads having pointers to the same object. If you do this, NT will generate an exception.

"If I dynamically allocate a CElnApi object, will it use a lot of heap space?"

No. The driver does not maintain an internal receive buffer; you must supply a pointer to an external receive buffer when you call CElnApi::TransCeive.

"What happens if a CElnApi object is destroyed before it closes the port?"

The CElnApi destructor handles this automatically.

"Will the driver work with Windows 3.1?"

No, it's 32-bit code.

Continued next page

3.6 Frequently Asked Questions, *continued*

"Can I recompile the source code with a 16-bit compiler?"

No. The AFX functions that deal with communications are completely different in the 32-bit world. Some Win32 functions, such as Sleep, don't exist at all in Win16. You would have to rewrite significant sections of the physical layer.

"Why is some initialization in the constructor, while other initialization is in SetUpPort?"

The initialization that you are obligated to perform is in the constructor. The initialization, which you are not necessarily obligated to perform, is in CElnApi::SetUpPort.

"What happens if the reply from the instrument is longer than the receive buffer whose pointer I have passed to TransCeive?

The driver will truncate the incoming message to the length of the buffer and return an Rx overflow error. The app should size the Rx buffer to the approximate length that you expect the reply to be. If the packet footer is obliterated by a burst of noise, the driver will keep stuffing incoming bytes until a timeout occurs or the end of the Rx buffer is reached, whichever happens first. If someone in the plant is running an arc welder or a shop vacuum, the driver may very well continue happily stuffing bytes of noise until it runs out of buffer space. Therefore, do not allocate an Rx buffer that is 65535 bytes long unless you really expect that much data to come back.

"Why is this driver a LIB and not a DLL?"

There are two main reasons. One is because, if the interface to the driver is ever changed, it becomes a maintenance problem for client apps to determine which version of the DLL the user has on his hard drive. The second problem, however, has to do with memory management.

Suppose the driver is a DLL, and the app declares a new CElnApi object in the usual way (arguments to constructor omitted for clarity):

```
CCommBase* pDriver = new CElnApi(…);
```

Even though the object was created by the app, the heap memory is nonetheless tagged as belonging to the DLL because the implementation of CElnApi is in the DLL. This means that if the app tries to destroy the driver in the usual way:

```
delete pDriver;
```

the system will throw an exception, because the app is trying to free memory which doesn't belong to it. The usual way around this problem is to disambiguate ownership of the memory by giving CElnApi a static Factory function and a Destroy function. The app would then have to do this:

```
CcommBase* pDriver = CElnApi::Factory(…);
…
pDriver->Destroy();
```

continued next page

3.6 Frequently Asked Questions, *continued*

The implementation of these functions in the DLL would look like this:

```
CCommBase* CElnApi::Factory(…) // Static function
{
       CCommBase* pDriver = new CElnApi(…);
       Return pDriver;
}

void CElnApi::Destroy()
{
       delete this;
}
```

Note: The Destroy function looks dangerous due to its self-destructive nature, but this is actually standard practice. Just don't try to access any member data after calling delete this.

Anyway, all of this complication can be neatly avoided by making the driver a statically linked library. Since the drivers are small, there's really no good reason to fight these battles.

4. Application Layer Message Formats

4.1 Overview

Introduction

This section describes the overall format of the request and response messages for the application layer. Refer to *subsection 2.9* for an anatomy of a message format.

Available Messages

Below are listed the available message requests that are available in this application:

- Read Contiguous 32-bit Values
- Read Scattered 32-bit Values
- Write Scattered 32-bit Values
- Loopback

Block Parameters

The location or address for data in the UMC800 controller consists of a Table Type, a Block Number, and an Index Number.

Table Type identifies whether it's a Dynamic (I/O) parameters or Static (configuration) parameters.

Block Number identifies a given function block entered in the Function Block Diagram (FBD) configuration with a unique assigned number between 1 and 250. The block number assignment can be printed out from the LeaderLine Control Builder.

Index Number identifies a particular parameter in a given block type that is accessible for communication purposes. This index information is available from the tables in *Section 5 - Function Parameter Index Reference*.

Variables

Note that a block number (251) has indices that have been assigned automatically for variables entered in the Function Block Diagram configuration. *Refer to subsection 5.77.* The index number assignments for variables can be printed out from the LeaderLine Control Builder.

In this section

These major topics are covered in this section.

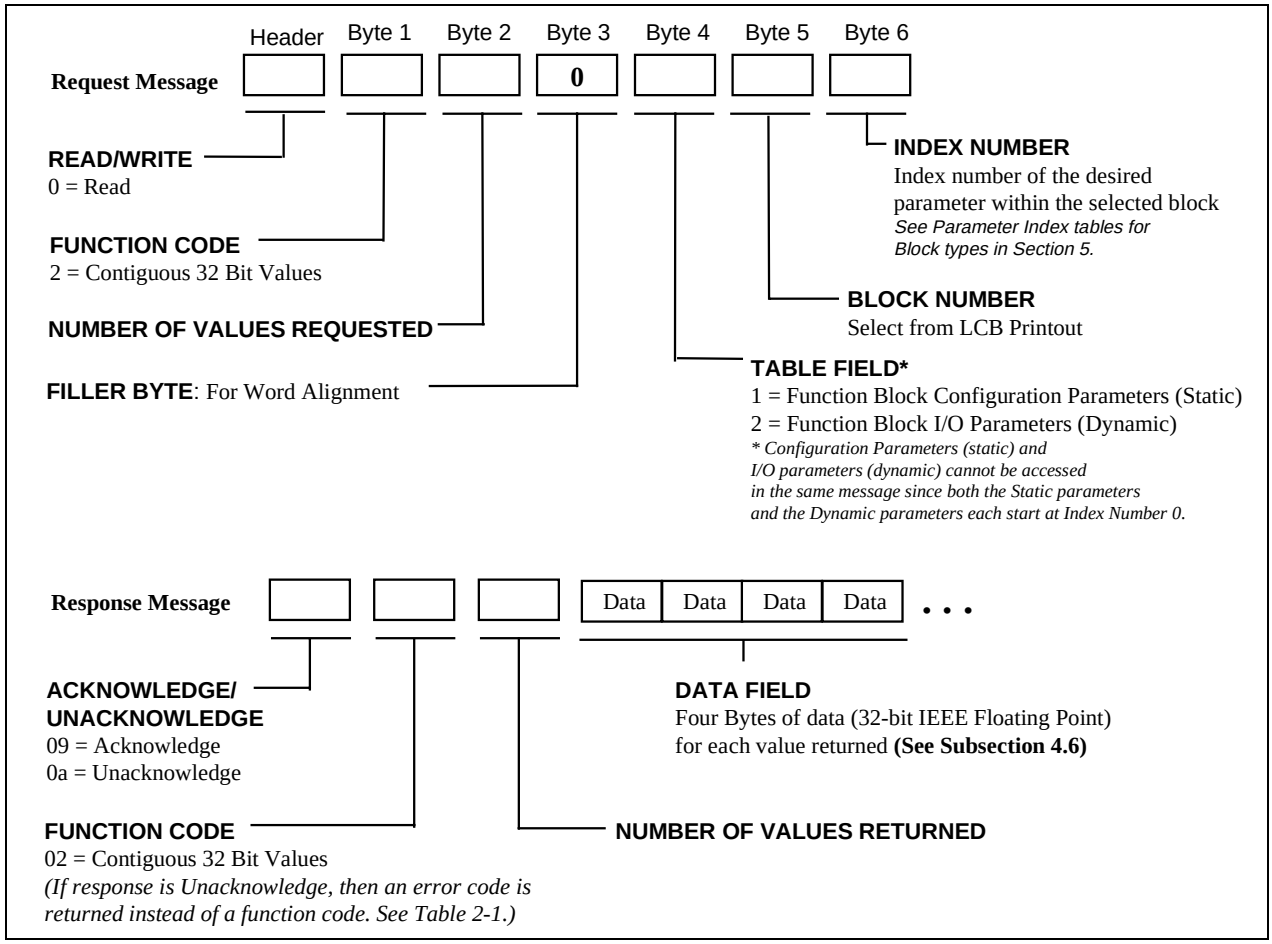
4.2 Read Contiguous 32-Bit Values

Introduction

Figure 4-1 shows the Read request and response format for Function Code 2 - Contiguous 32-Bit values. This operation lets you read a list of 32-Bit values from a specific block given the starting index number and the number of values involved.

Message Formats

Figure 4-1 Read Contiguous 32-Bit Request and Response Message Formats



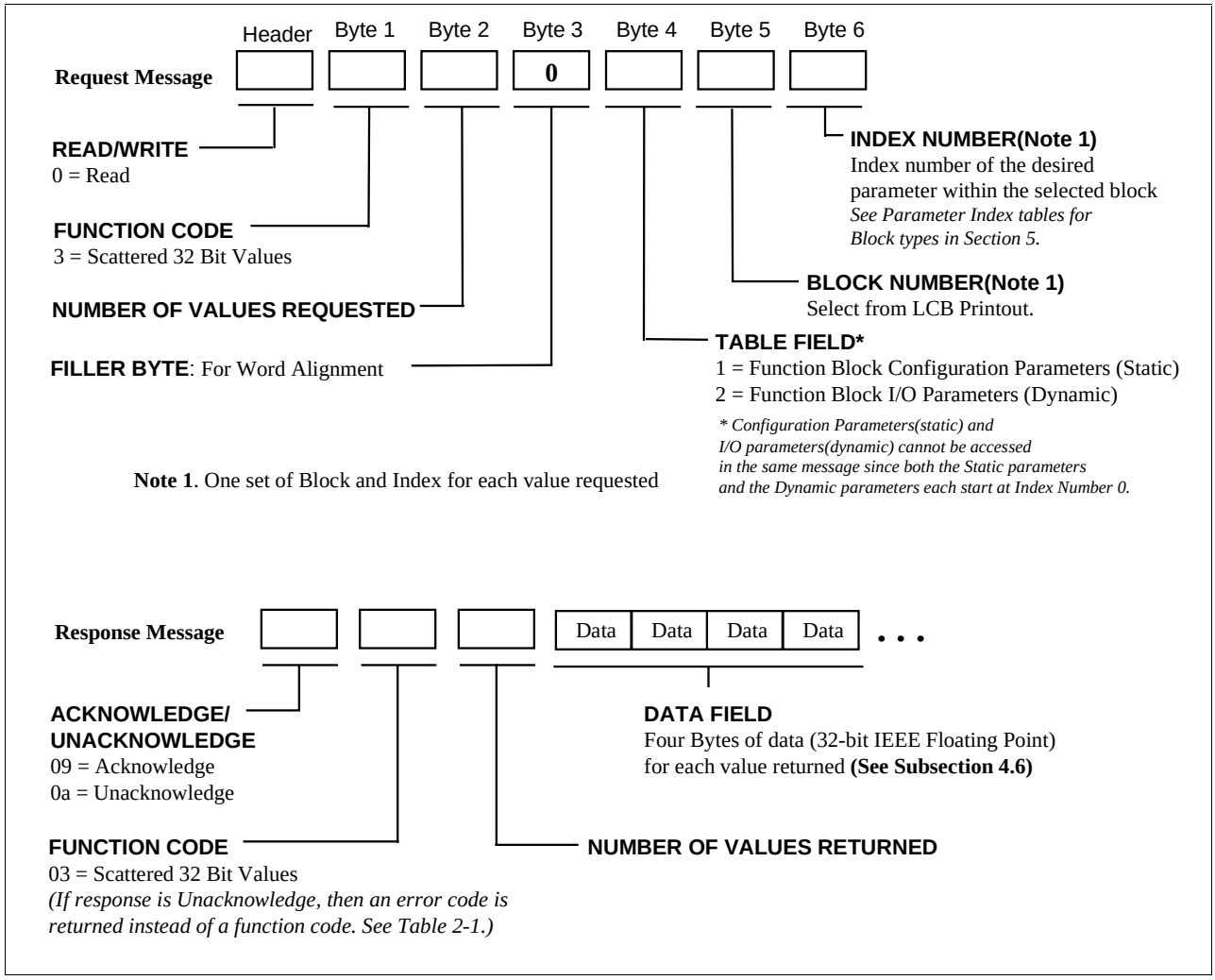
4.3 Read Scattered 32-Bit Values

Introduction

Figure 4-2 shows the Read request and response format for Function Code 3 - Scattered 32-Bit values. This operation lets you read 32-Bit values from a specific block given the index numbers and the number of values involved.

Message Formats

Figure 4-2 Read Scattered 32-Bit Request and Response Message Formats



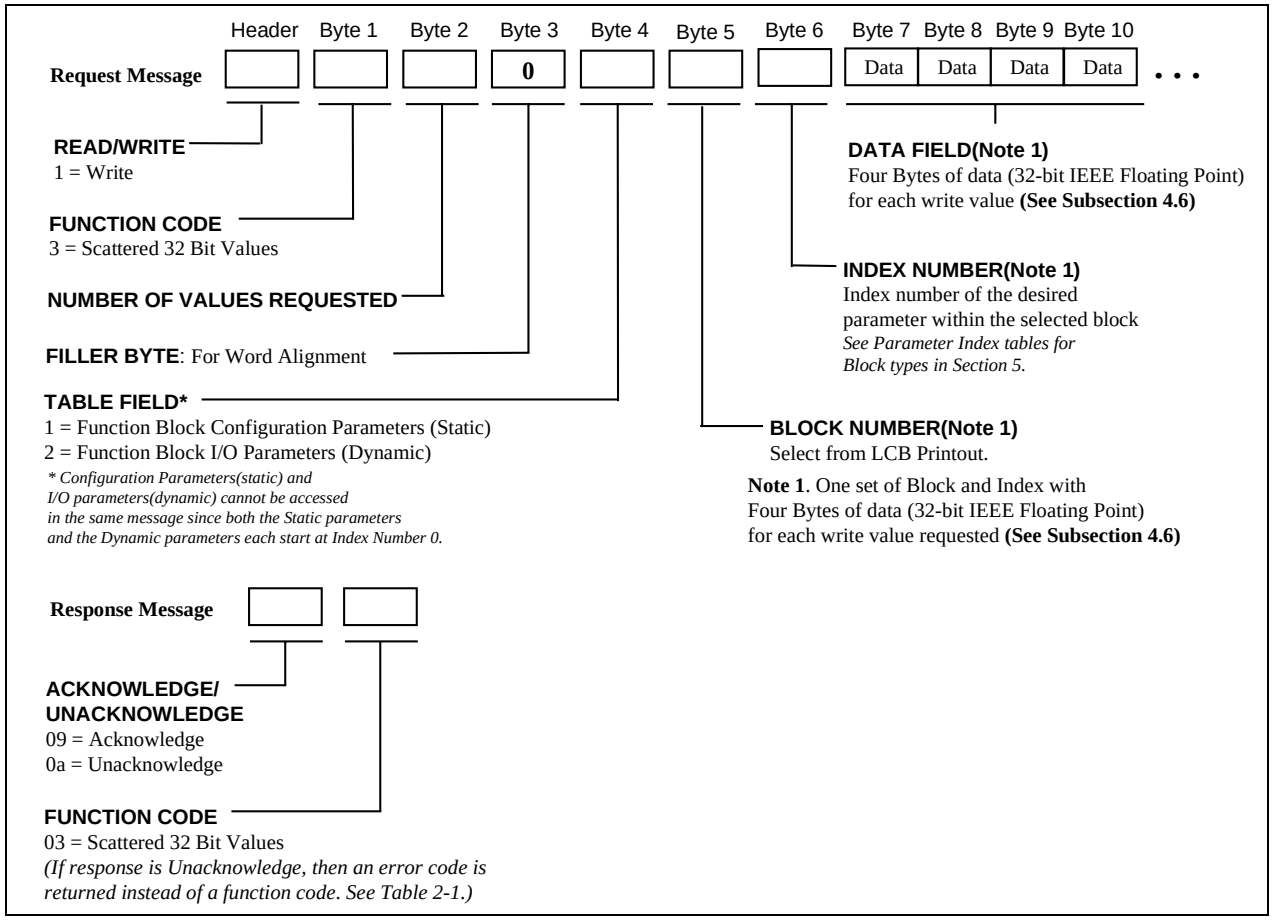
4.4 Write Scattered 32-Bit Values

Introduction

Figure 4-3 shows the Write request and response format for Function Code 3 - Scattered 32-Bit values. This operation lets you write 32-Bit values to a specific block given the index numbers and the number of values involved.

Message Formats

Figure 4-3 Write Scattered 32-Bit Request and Response Message Formats



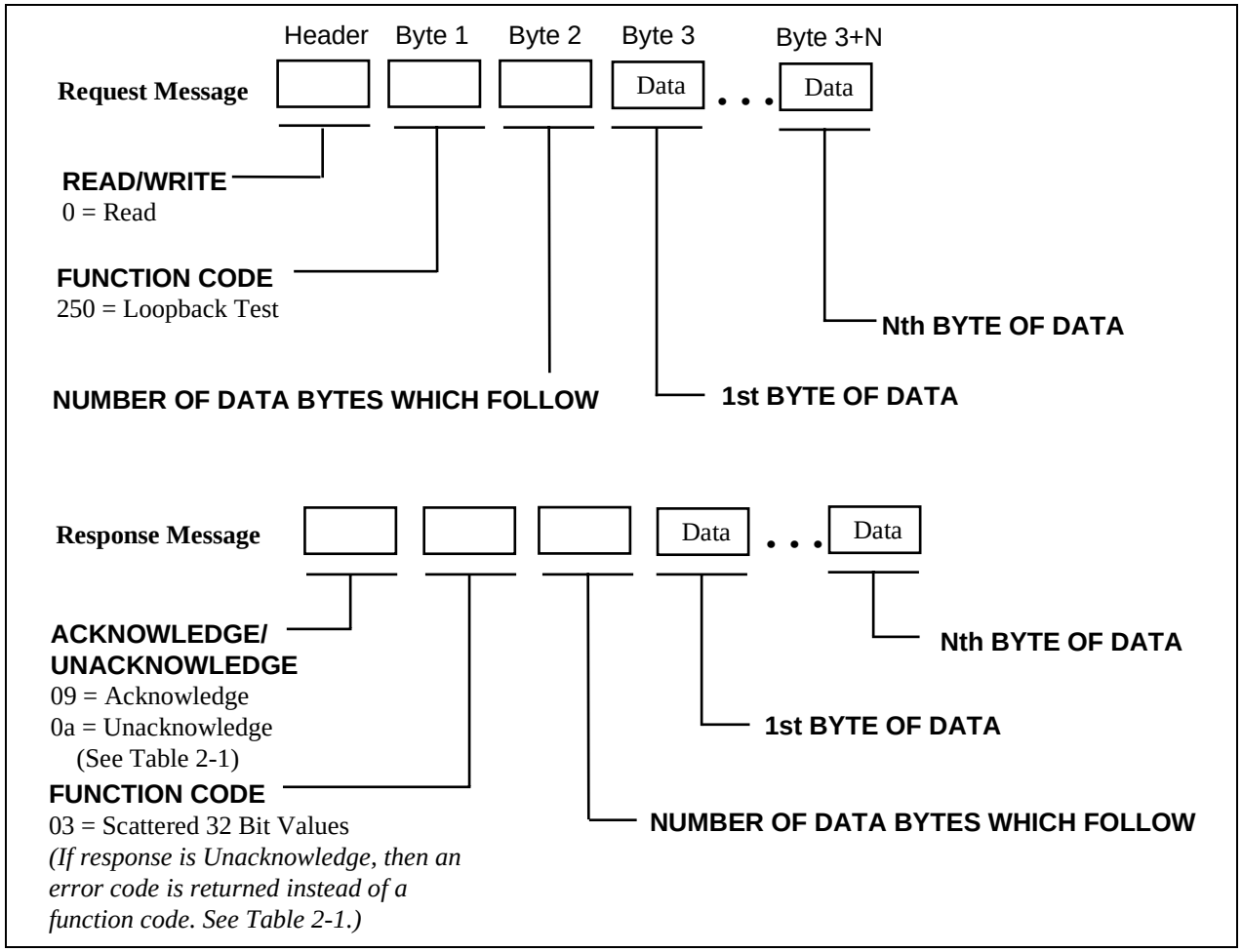
4.5 Loopback

Introduction

Figure 4-4 shows the Request and Response format for Function Code 250 - Loopback Test. This operation lets you echo back a string of bytes from the controller to the Host.

Message Formats

Figure 4-4 Loopback Request and Response Message Formats



4.6 Floating Point Format

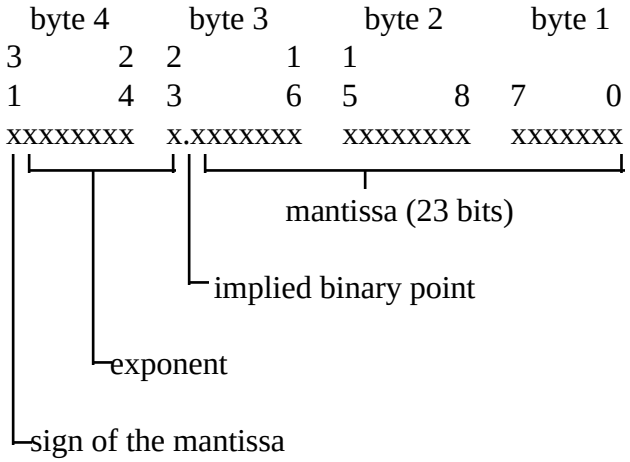
Floating point format is per the IEEE Standard for Binary Floating Point Arithmetic (TASK P754). This format conforms to that of the Unit's Math Firmware and/or Co-Processor.

4.6.1 IEEE 754 Format

The Floating Point data format consists of four contiguous bytes containing a 23-bit signed binary mantissa with an 8-bit biased binary exponent. The bits are numbered from 31 to 0, left to right. Byte 4 contains bits 31-24, byte 3 contains bits 23 to 16, and so on. A description of the layout and accompanying figure follow:

BITS	DESCRIPTION
0-22	**Mantissa**
23-30	**Exponent: 8-bit unsigned value**
31	**Sign of the mantissa:** **0 = positive, 1 = negative**

An implied binary point exists between bits 22 and 23. Note that the transmission order is byte 1 to byte 4.



The formula for calculating the floating point number is: $+/-.\text{mantissa} \times 2^{\text{exponent}}$

Continued next page

4.6.2 Mantissa and Sign

The mantissa is defined by a sign bit (31) and a 23-bit binary fraction. This binary fraction is combined with an "implied" value of 1 to create a mantissa value which is greater than or equal to 1.0 and less than 2.0.

The mantissa is positive if the sign bit is zero (reset), and negative if the sign bit is one (set). For example:

DECIMAL	HEXADECIMAL	BINARY
100	**42C80000**	**01000010 11001000 00000000 00000000**

The sign bit is zero, indicating a positive mantissa. Removing the sign bits and exponent bits, the mantissa becomes:

HEXADECIMAL	BINARY
480000	**xxxxxxxx x1001000 00000000 00000000**

Add an "implied" value of one to the left of the binary point:

BINARY
1.1001000 00000000 00000000

Using positioned notation, this binary number is equal to:

$$1.0 + (1x2^{-1}) + (0x2^{-2}) + (0x2^{-3}) + (1x2^{-4}) = 1.0 + 0.5 + 0.0 + 0.0 + 0.0625 = 1.5625$$

4.6.3 Exponent

The exponent is defined by an unsigned 8-bit binary value (bits 23 through 30). The value of the exponent is derived by performing a signed subtraction of 127 (decimal) from the 8-bit exponent value.

DECIMAL	HEXADECIMAL	BINARY
100	**42C80000**	**01000010 11001000 00000000 00000000**

removing the sign and mantissa bits, the exponent becomes:

DECIMAL	HEXADECIMAL	BINARY
133	**85**	**x1000010 1xxxxxxx xxxxxxxx xxxxxxxx**

or:

$$1x2^{7} + 0x2^{6} + 0x2^{5} + 0x2^{4} + 0x2^{3} + 1x2^{2} + 0x2^{1} + 1x2^{0}$$

Subtract a bias of 127 from the exponent to determine its value: 133 – 127 = 6.

Continued next page

4.6.4 Mantissa and Exponent Combination

Combining the mantissa and exponent from the two previous examples:

$$\text{float number} = \text{mantissa} \times 2^{\text{exponent}}$$

$$\text{float number} = 1.5625 \times 2^{6} = 1.5625 \times 64 = 100.0$$

Below is a list of sample float values in IEEE 754 format:

DECIMAL	HEXADECIMAL
100.0	**42C80000**
-100.0	**C2C80000**
0.5	**3F000000**
-1.75	**BFE00000**
0.0625	**3D800000**

4.6.5 Reserved Operands

Per the Standard certain exceptional forms of floating point operands are excluded from the numbering system. These are as follows:

EXCEPTION	EXPONENT	MANTISSA
+/- Infinity	**All 1's**	**All 0's**
Not-a-Number (NAN)	**All 1's**	**Other than 0's**
Denormalized Number	**All 0's**	**Other than 0's**

5. Function Parameter Index Reference

5.1 Parameter Index Numbers

Function Block parameter tables

Refer to the tables listed below to find the correct Dynamic (I/O) or Static (Configuration) parameter index number for a given function block that is to be accessed through a communications message.

Abbreviations

The abbreviations used in the tables are:

REAL Floating Point Analog Numbers

BOOL Digital ON and OFF states

C Contained Parameter

I Input Value

O Output Value

R Read Only

R/W Read/Write

W Write

In this section

Refer to Table 4-1 for a listing of each function block type and respective reference page.

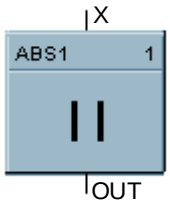
Table 5-1 Function Block Look-up Table

Function Block Type Identification Label	See Subsection
ABS (Absolute Value)	5.2
ADD (Addition 2 Inputs**)**	5.3
4ADD (Addition 4 Inputs)	5.4
AI (Analog Input)	5.5
ALM (Analog Alarm)	5.6
2AND (AND - 2 Inputs)	5.7
4AND (AND - 4 Inputs)	5.8
8AND (AND - 8 Inputs)	5.9
AMB (Auto/Manual Bias)	5.10
AO (Analog Output)	5.11
ASYS (Alarm System Monitor)	5.12
BCD (Binary Coded Decimal Translator)	5.13
BOOL (Free Form Logic)	5.14
CARB (Carbon Potential)	5.15
CMPR (Comparison)	5.16
DCMP (Deviation Compare)	5.17

5.2 ABS Function Block

Description

The **ABS** label stands for **Absolute Value**. This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-2 ABS Dynamic Parameters

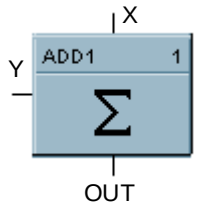
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	result
X	2	REAL	I	R	input

Static Configuration Parameters: None

5.3 ADD Function Block

Description

The **ADD** label stands for **Addition Mathematical Operation (2 Inputs).** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-3 ADD Dynamic Parameters

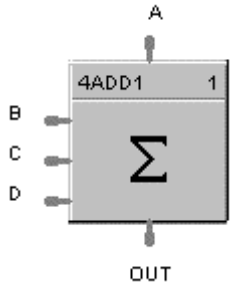
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input 1
Y	3	REAL	I	R	input 2

Static Configuration Parameters: None

5.4 4ADD Function Block

Description

The **4ADD** label stands for **Addition Mathematical Operation (4 Inputs).** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-4 4ADD Dynamic Parameters

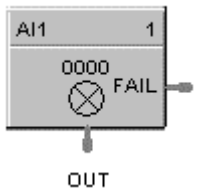
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
A	2	REAL	I	R	input 1
B	3	REAL	I	R	input 2
C	4	REAL	I	R	input 3
D	5	REAL	I	R	input 4

Static Configuration Parameters: None

5.5 AI Function Block

Description

The **AI** label stands for **Analog Input**. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-5 AI Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	Block Status (see section 5.2 for code list)
OUT	1	REAL	O	R	analog input value (eu)
FAIL	2	BOOL	O	R	Failed input indication

Static Configuration Parameters:

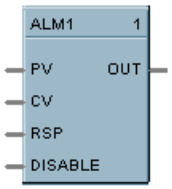
Table 5-6 AI Static Configuration Parameters

Parameter	Index	Type	Description
filt_time	2	REAL	filter time constant (seconds) {0-120} [default 0]
bias	3	REAL	bias (eu).{-99999 to 99999} [default 0]
failsafe	4	REAL	failsafe value (eu) [default 0]
range_hi	6	REAL	high range value [default 0]
range_lo	7	REAL	low range value [default 100]

5.6 ALM Function Block

Description

The **ALM** label stands for the **Analog Alarm function.** This block is part of the *Alarms/Monitor* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-7 ALM Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
ack	1	BOOL	C	R/W	ON acknowledges the alarm
OUT	2	BOOL	O	R	output
PV	3	REAL	I	R	Process variable
CV	4	REAL	I	R	Compare value
RSP	5	REAL	I	R	Remote setpoint
DISABLE	6	BOOL	I	R	ON disables alarm action

Static Configuration Parameters:

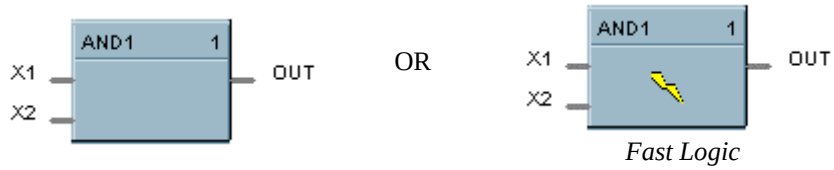
Table 5-8 ALM Static Configuration Parameters

Parameter	Index	Type	Description
lsp	0	REAL	local setpoint in eu {-99999.9 to 99999.9}
rem_mode	1	BOOL	ON selects RSP

5.7 2AND Function Block

Description

The **2AND** label stands for the **AND Boolean function (2 Inputs).** This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-9 2AND Dynamic Parameters

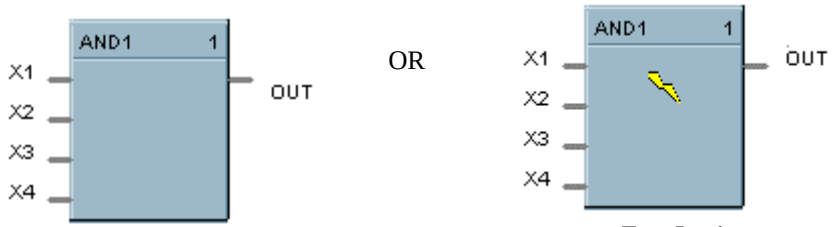
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1(X1)	2	BOOL	I	R	input
DIG_2(X2)	3	BOOL	I	R	input

Static Configuration Parameters: None

5.8 4AND Function Block

Description

The **4AND** label stands for the **AND Boolean function (4 Inputs).** This block is part of the *Logic or Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-10 4AND Dynamic Parameters

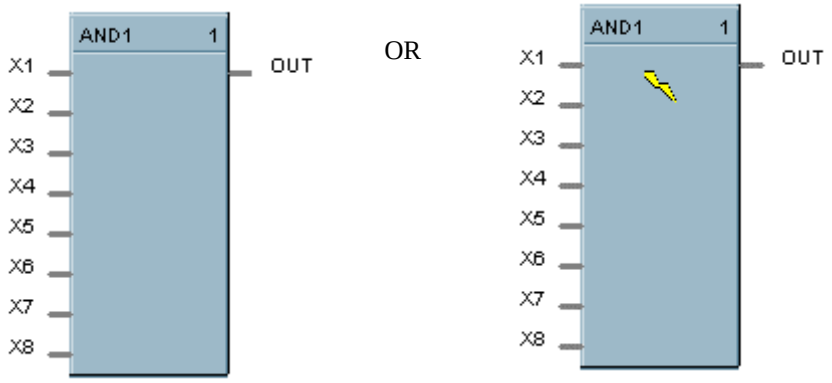
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1(X1)	2	BOOL	I	R	input
DIG_2(X2)	3	BOOL	I	R	input
DIG_3(X3)	4	BOOL	I	R	input
DIG_4(X4)	5	BOOL	I	R	input

Static Configuration Parameters: None

5.9 8AND Function Block

Description

The **8AND** label stands for the **AND Boolean function (8 Inputs)**. This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-11 8AND Dynamic Parameters

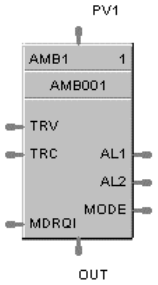
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1(X1)	2	BOOL	I	R	input
DIG_2(X2)	3	BOOL	I	R	input
DIG_3(X3)	4	BOOL	I	R	input
DIG_4(X4)	5	BOOL	I	R	input
DIG_5(X5)	6	BOOL	I	R	input
DIG_6(X6)	7	BOOL	I	R	input
DIG_7(X7)	8	BOOL	I	R	input
DIG_8(X8)	9	BOOL	I	R	input

Static Configuration Parameters: None

5.10 AMB Function Block

Description

The AMB label stands for Auto/Manual Bias Function. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Values:

Table 5-12 AMB Dynamic Values

Parameter	Index	Type	Use	R/W	Description
Status	0	REAL	C	R	block status
Bias	1	REAL	C	R	calculated bias (%)
Man_mode	2	BOOL	C	R/W	manual output mode request {OFF, ON}
Man_out	3	REAL	C	R/W	manual output value –5 to 105 (%)
Pv	4	REAL	C	R	Process Variable in % for monitoring
OUT	5	REAL	O	R	control output –5 to 105 (%)
MODE	6	REAL	O	R	actual mode encoded per note 3
ALl	7	BOOL	O	R	Alarm 1
AL2	8	BOOL	O	R	Alarm 2
PVI	9	REAL	I	R	Process Variable Input (%) (pv_lo <= PV <= pv_hi)
TRV	10	REAL	I	R	Output Track Value (%)
TRC	11	BOOL	I	R	Output Track Command {OFF, ON}
MDRQI	12	REAL	I	R	External Mode Request encoded per note 4

Static Configuration Values:

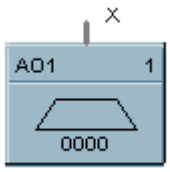
Table 5-13 AMB Static Configuration Values

Parameter	Index	Type	Description
pv_hi	0	REAL	pv High Range value -5 to 105%
pv_lo	1	REAL	pv Low Range value –5 to 105%
outhilim	5	REAL	output high limit, -5 to **105**
outlolim	6	REAL	output low limit, **-5** to 105
failsafe	7	REAL	failsafe output value, -5 to 105, (default **0**)
al_sp[4]	8-11	REAL	alarm set points al1spl, al1sp2, al2spl, al2sp2, -99999 to 99999 (default **0**)
al_hyst	16	REAL	alarm hysterysis **0** to 5 (%)

5.11 AO Function Block

Description

The **AO** label stands for **Analog Output**. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-14 AO Dynamic Parameters

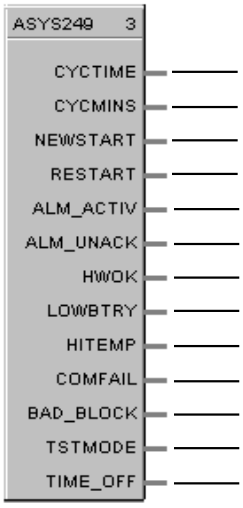
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
IN (X)	1	REAL	I	R	analog input value (eu)

Static Configuration Parameters: None

5.12 ASYS Function Block

Description

The **Analog System Status Block (ASYS)** is a function block and is part of the *Alarm/Monitor* category. It provides read access to controller status values including those related to the Analog execution cycle. The output may be connected to function block inputs. The outputs may also be connected to signal tags for operator interface monitoring. The ASYS System Monitoring block is assigned block number 249. It looks like this graphically on the Control Builder:



CYCTIME — Analog function block scan cycle time in sec.
CYCMINS — Analog function block scan cycle time in min.
NEWSTART — Logic 1 (Hi) for one analog scan cycle after a new start of program (Program to Run)
RESTART — Logic 1 (Hi) for one analog scan cycle after power up
ALM_ACTIV — Logic 1 (Hi) if there is any operator panel alarm is present
ALM_UNACK — Logic 1 (Hi) if there is any operator panel unacknowledged alarm present
HWOK — Logic 1 (Hi) if a hardware fault is detected
LOWBTRY — Logic 1 (Hi) if battery is low
HITEMP — Logic 1 (Hi) if internal temperature is too high (>70C)
COMFAIL — Logic 1 (Hi) if there is a comm. failure to the operator interface
BAD_BLOCK — Logic 1 (Hi) if a function block fails to operate properly
TSTMODE — Logic 1 (Hi) when the controller run/program/test switch is in the test mode
TIME_OFF — Time controller was off (in sec.) after power loss (available for one analog scan cycle)

Dynamic Parameters:

Table 5-15 ASYS Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
CYC_CNT	1	REAL	C	R	number of control block cycles
EXCTIME	3	REAL	C	R	control block execution time in seconds
PK_EXCTIME	4	REAL	C	R	peak control block execution time in seconds
EXECFAULT	7	BOOL	C	R	ON = executive fault
CYCTIME	8	REAL	O	R	control block cycle time in seconds
CYCMINS	9	REAL	O	R	control block cycle time in minutes
NEWSTART	10	BOOL	O	R	ON = new start
RESTART	11	BOOL	O	R	ON = warm start
ALM_ACTIV	12	BOOL	O	R	ON = active alarm
ALM_UNACK	13	BOOL	O	R	ON = unacknowledged alarm
HWOK	14	BOOL	O	R	ON = no hardware faults (See note 6)
LOWBTRY	15	BOOL	O	R	ON = battery is low
HITEMP	16	BOOL	O	R	ON = high RJ temperature

BAD_BLOCK	18	BOOL	O	R	ON = one or more blocks have bad status
RUNMODE	20	BOOL	C	R	ON = run mode is active
PRGMODE	21	BOOL	C	R	ON = program mode is active
MIN_PER_TICK	26	REAL	C	R	minutes per OS tick
CODE_REV	28	REAL	C	R	code revision number
REALTIME_OFF	30	REAL	O	R	Number of seconds the controller was powered down.*

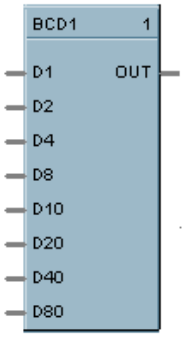
*Valid for only control block execution cycle after power-up.

Static Configuration Parameters: None.

5.13 BCD Function Block

Description

The **BCD** label stands for **Binary Coded Decimal Translator.** This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-16 BCD Dynamic Parameters

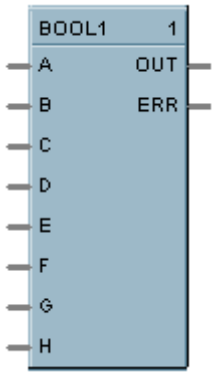
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	W	Result (O to 99)
DIG_A(D1)	2	BOOL	I	R	Bit 0 of the BCD lower digit
DIG_B(D2)	3	BOOL	I	R	Bit 1 of the BCD lower digit
DIG_C(D4)	4	BOOL	I	R	Bit 2 of the BCD lower digit
DIG_D(D8)	5	BOOL	I	R	Bit 3 of the BCD lower digit
DIG_E(D10)	6	BOOL	I	R	Bit 0 of the BCD upper digit
DIG_F(D20)	7	BOOL	I	R	Bit 1 of the BCD upper digit
DIG_G(D40)	8	BOOL	I	R	Bit 2 of the BCD upper digit
DIG_H(D80)	9	BOOL	I	R	Bit 3 of the BCD upper digit

Static Configuration Parameters: None

5.14 BOOL Function Block

Description

The **BOOL** label stands for **Free Form Logic.** This block is part of the *Logic* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-17 BOOL Dynamic Parameters

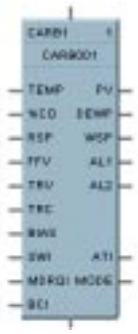
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	Result
ERR	2	BOOL	O	R	Error indication
A	3	BOOL	I	R	Input 1
B	4	BOOL	I	R	Input 2
C	5	BOOL	I	R	Input 3
D	6	BOOL	I	R	Input 4
E	7	BOOL	I	R	Input 5
F	8	BOOL	I	R	Input 6
G	9	BOOL	I	R	Input 7
H	10	BOOL	I	R	Input 8

Static Configuration Parameters: None

5.15 CARB Function Block

Description

The **CARB** label stands for **Carbon Potential.** This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-18 CARB Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
lsp	1	REAL	C	R/W	local set point (eu)
lsp2	2	REAL	C	R/W	local set point 2 (eu)
rem_mode	3	BOOL	C	R/W	remote set point mode request {OFF, ON}
man_mode	4	BOOL	C	R/W	manual output mode request {OFF, ON}
man_out	5	REAL	C	R/W	manual output value –5 to 105 (%)
tune_req	6	BOOL	C	R/W	limit cycle auto-tuning request {OFF, ON}
rsp_eu	7	REAL	C	R	remote set point in eu for monitoring
deviation	8	REAL	C	R	Deviation in eu for monitoring
PV	9	REAL	O	R	Calculated Process Variable (Percent Carbon) for monitoring
DSP	10	REAL	O	R	Display Set Point in eu for monitoring
OUT	11	REAL	O	R	control output –5 to 105 (%)
MODE	12	REAL	O	R	actual mode encoded
All	13	BOOL	O	R	Alarm 1
AL2	14	BOOL	O	R	Alarm 2
DEWPT	15	REAL	O	R	Calculated dewpoint [replaces BCO]
ATI	16	BOOL	O	R	Auto Tune Indicator. ON = Auto Tune in progress
O2	17	REAL	I	R	Oxygen sensor input (0 to 100%)
RSP	18	REAL	I	R	Remote Set Point (% or eu per sp_units)

FFV	19	REAL	I	R	Feed Forward Value (%)
TRV	20	REAL	I	R	Output Track Value (%)
TRC	21	BOOL	I	R	Output Track Command {OFF, ON}
BCI	23	REAL	I	R	Back Calculation Input (%)
BIAS	24	REAL	I	R	Remote bias value for ratio PID
TEMP	26	REAL	I	R	Temperature input (°F or °C per USE_METRIC)
%CO	27	REAL	I	R	Percent carbon monoxide

Static Configuration Parameters:

Table 5-19 CARB Static Configuration Parameters

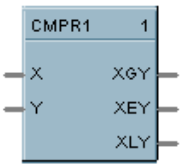
Parameter	Index	Type	Description
GAIN	0	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 1]
RATE	1	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 1]
RESET	2	REAL	integration time, **0** or 0.02 to 50 (minutes) or repeats per minute, 0 or 0.02 to 50 (repeats) [Tune Set 1]
pv_hi	4	REAL	pv High Range value –99999 to 99999 (default **100**)
pv_lo	5	REAL	pv Low Range value –99999 to 99999 (default **0**)
sp_hi_lim	11	REAL	set point high limit, -99999 to 99999 (default **100**)
sp_lo_lim	12	REAL	set point low limit, -99999 to 99999 (default **0**)
outhilim	14	REAL	output high limit, -5 to **105**
outlolim	15	REAL	output low limit, **-5** to 105
failsafe	16	REAL	failsafe output value, -5 to 105, (default **0**)
al_sp[4]	17-20	REAL	alarm set points al1spl, al1sp2, al2spl, al2sp2, -99999 to 99999 (default **0**)
al_hyst	25	REAL	alarm hysterysis **0** to 5 (%)
man_reset	26	REAL	Manual Reset, -100 to 100 (in % output) (default **0**) [used for both tune sets]
FUZZY	28	BOOL	ON enables fuzzy logic overshoot suppression (default **OFF**)
TUNESET2	29	BOOL	Use tune set 2 (default **OFF**)
GAIN2	30	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 2]
RATE2	31	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 2]
RESET2	32	REAL	integration time, **0** or 0.02 to 50 (minutes) or repeats per minute, 0 or 50 to 0.02 (repeats) [Tune Set 2]
use_propband	33	BOOL	Use Gain (**0**) or Proportional Band (1)

use_rpm	34	BOOL	Use minutes (**0**) or repeats per minute (1) for integral constant
sp_rate_dn	35	REAL	Set point low rate of change limit, **0** (off) to 99999 (eu/min)
sp_rate_up	36	REAL	Set point high rate of change limit, **0** (off) to 99999 (eu/min)
FF_GAIN	37	REAL	Feed forward gain, **0.0** to 10.0
RATIO	39	REAL	Gain value for Ratio PID (-20 to 20) (default **1**) [used when RA_BIAS > 0]
LBIAS	40	REAL	Bias value for Ratio PID when RA_BIAS = LOC_BIAS](-99999 to 99999) (**0**)
devbar_hi	41	REAL	High scale value for deviation bar graph (0 to 99999) (default **100**)
devbar_low	42	REAL	Low scale value for deviation bar graph [always = -devbar_hi]
L%CO	43	REAL	Local percent carbon monoxide (2.0 to 35.0, default 20.0)
REM_CO	44	BOOL	Use %CO input instead of local L%CO (default = OFF)
FURNACE_FACTOR	45	REAL	Furnace Factor in %C (-0.5 to 0.5)
ANTI_SOOT	46	BOOL	Anti-soot SP limit enable (default = OFF)
TEMP_LO_LIM	48	REAL	Trigger value for LOTEMP Boolean output (0 to 2500°F)
PERCENT_H	50	REAL	Percent hydrogen (1 to 100, default = 40)

5.16 CMPR Function Block

Description

The **CMPR** label stands for **Comparison Calculation.** This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-20 CMPR Dynamic Parameters

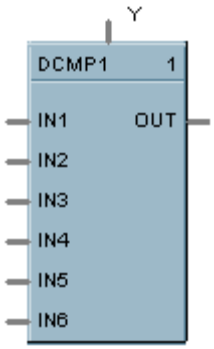
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
XGY	1	BOOL	O	R	X Greater than Y
XEY	2	BOOL	O	R	X Equals Y
XLY	3	BOOL	O	R	X Less than Y
X	4	REAL	I	R	input 1
Y	5	REAL	I	R	input 2

Static Configuration Parameters: None

5.17 DCMP Function Block

Description

The **DCMP** label stands for **Deviation Compare.** This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-21 DCMP Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
IN1	2	REAL	I	R	input 1
IN2	3	REAL	I	R	input 2
IN3	4	REAL	I	R	input 3
IN4	5	REAL	I	R	input 4
IN5	6	REAL	I	R	input 5
IN6	7	REAL	I	R	input 6
REF(Y)	8	REAL	I	R	reference input

Static Configuration Parameters:

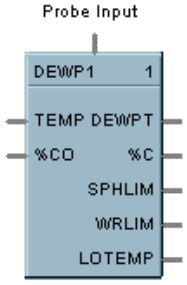
Table 5-22 DCMP Static Configuration Parameters

Parameter	Index	Type	Description
+ DEV	0	REAL	plus deviation
- DEV	1	REAL	minus deviation

5.18 DEWP Function Block

Description

The **DEWP** label stands for **Dewpoint Calculation**. This block is part of the *Calculations* category. It looks like this graphically on the Control builder.



Dynamic Parameters:

Table 5-23 DEWP Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
DEWPT	1	REAL	O	R	Calculated dew point output.
%C	2	REAL	O	R	Calculated percent carbon output.
SPHLIM	3	REAL	O	R	Control set point high limit for anti-soot
WRLIM	4	BOOL	O	R	Command to write the set point high limit.
LOTEMP	5	BOOL	O	R	ON when TEMP is <= calculated low temperature dropoff.
O2	6	REAL	I	R	Oxygen sensor input (0 to 100%)
TEMP	7	REAL	I	R	Temperature input (°F or °C per USE_METRIC)
%CO	8	REAL	I	R	Percent carbon monoxide input

Static Configuration Parameters:

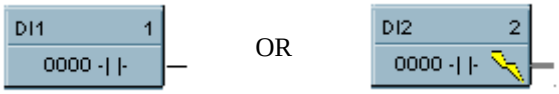
Table 5-24 DEWP Static Configuration Parameters

Parameter	Index	Type	Description
L%CO	0	REAL	Local percent carbon monoxide (2.0 to 35.0, default 20.0)
REM_CO	1	BOOL	Use %CO input instead of local L%CO (default = OFF)
FURNACE_FACTOR	2	REAL	Furnace Factor in %C (-0.5 to 0.5)
ANTI_SOOT	3	BOOL	Anti-soot SP limit enable
TEMP_LO_LIM	5	REAL	Trigger value for LOTEMP Boolean output (0 to 2500°F)
PERCENT_H	7	REAL	Percent hydrogen (1 to 100, default = 40)

5.19 DI Function Block

Description

The **DI** label stands for **Discrete Input.** This block is part of the *Logic* or *Fast Logic* categories. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-25 DI Dynamic Parameters

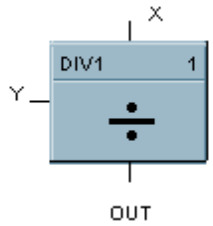
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT_D	1	BOOL	O	R	

Static Configuration Parameters: None

5.20 DIV Function Block

Description

The **DIV** label stands for **Division Mathematical operation**. This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-26 DIV Dynamic Parameters

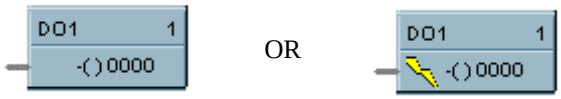
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input 1
Y	3	REAL	I	R	input 2

Static Configuration Parameters: None

5.21 DO Function Block

Description

The **DO** label stands for **Digital Output.** This block is part of the *Logic or Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-27 DO Dynamic Parameters

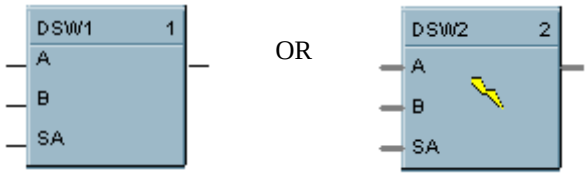
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT_D	1	REAL	C	R	Physical output value
IN_D	2	BOOL	I	R	

Static Configuration Parameters: None

5.22 DSW Function Block

Description

The **DSW** label stands for **Digital Switch.** This block is part of the *Logic* or *Fast Logic* categories. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-28 DSW Dynamic Parameters

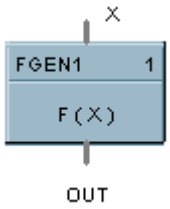
Index	Parameter	Type	Use	R/W	Description
0	status	REAL	C	R	block status (see section 5.2 for code list)
1	OUT	BOOL	O	R	output
2	A	BOOL	I	R	input A
3	B	BOOL	I	R	input B
4	SA	BOOL	I	R	Select A

Static Configuration Parameters: None

5.23 FGEN Function Block

Description

The **FGEN** label stands for **Function Generator - 10 Segment**. This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder:



Dynamic Parameters:

Table 5-29 FGEN Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input

Static Configuration Parameters:

Table 5-30 FGEN Static Configuration Parameters

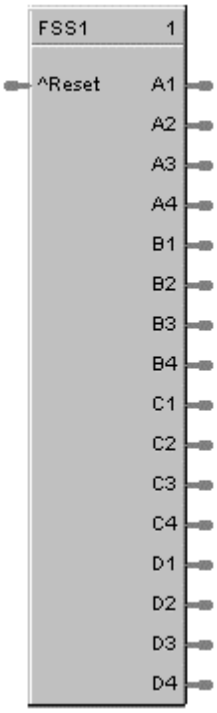
Parameter	Index	Type	Description
xb[0]	0	REAL	x breakpoint 1 (-99999 to 999999)
xb[1]	1	REAL	x breakpoint 2 (-99999 to 999999)
xb[2]	2	REAL	x breakpoint 3 (-99999 to 999999)
xb[3]	3	REAL	x breakpoint 4 (-99999 to 999999)
xb[4]	4	REAL	x breakpoint 5 (-99999 to 999999)
xb[5]	5	REAL	x breakpoint 6 (-99999 to 999999)
xb[6]	6	REAL	x breakpoint 7 (-99999 to 999999)
xb[7]	7	REAL	x breakpoint 8 (-99999 to 999999)
xb[8]	8	REAL	x breakpoint 9 (-99999 to 999999)
xb[9]	9	REAL	x breakpoint 10 (-99999 to 999999)
xb[10]	10	REAL	x breakpoint 11 (-99999 to 999999)
yb[0]	11	REAL	output value at x breakpoint 1 (-99999 to 999999)
yb[1]	12	REAL	output value at x breakpoint 2 (-99999 to 999999)
yb[2]	13	REAL	output value at x breakpoint 3 (-99999 to 999999)

yb[3]	14	REAL	output value at x breakpoint 4 (-99999 to 999999)
yb[4]	15	REAL	output value at x breakpoint 5 (-99999 to 999999)
yb[5]	16	REAL	output value at x breakpoint 6 (-99999 to 999999)
yb[6]	17	REAL	output value at x breakpoint 7 (-99999 to 999999)
yb[7]	18	REAL	output value at x breakpoint 8 (-99999 to 999999)
yb[8]	19	REAL	output value at x breakpoint 9 (-99999 to 999999)
yb[9]	20	REAL	output value at x breakpoint 10 (-99999 to 999999)
yb[10]	21	REAL	output value at x breakpoint 11 (-99999 to 999999)

5.24 FSS Function Block

Description

The **FSS**label stands for **Four-Selector Switch.** This block is part of the *Logic* category. It looks like this graphically on the Control Builder:



Dynamic Parameters:

Table 5-31 FSS Dynamic Parameters

Index	Parameter	Type	Use	R/W	Description
0	status	REAL	C	R	block status
9	A1	BOOL	O	R	Bank A output #1
10	A2	BOOL	O	R	Bank A output #2
11	A3	BOOL	O	R	Bank A output #3
12	A4	BOOL	O	R	Bank A output #4
13	B1	BOOL	O	R	Bank B output #1
14	B2	BOOL	O	R	Bank B output #2
15	B3	BOOL	O	R	Bank B output #3
16	B4	BOOL	O	R	Bank B output #4
17	C1	BOOL	O	R	Bank C output #1
18	C2	BOOL	O	R	Bank C output #2
19	C3	BOOL	O	R	Bank C output #3

Index	Parameter	Type	Use	R/W	Description
20	C4	BOOL	O	R	Bank C output #4
21	D1	BOOL	O	R	Bank D output #1
22	D2	BOOL	O	R	Bank D output #2
23	D3	BOOL	O	R	Bank D output #3
24	D4	BOOL	O	R	Bank D output #4
25	Reset	BOOL	I	R	Off to On requests a reset state

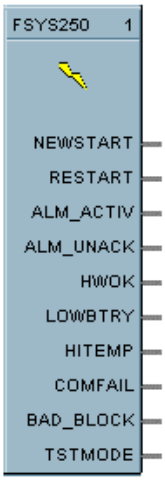
Static Configuration Parameters:

None

5.25 FSYS Function Block

Description

The **Fast Logic Status Block (FSYS)** is a function block and is part of the *Fast Logic* category. It provides read access to controller status values including those related to the Fast Logic execution cycle. The output may be connected to function block inputs. The outputs may also be connected to signal tags for operator interface monitoring. The FSYS System Monitoring block is assigned block number 250. It looks like this graphically on the Control Builder:



Dynamic Parameters:

Table 5-32 FSYS Dynamic Parameters

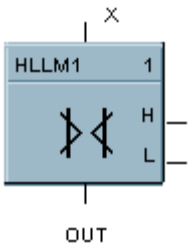
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
CYC_CNT	1	REAL	C	R	number of control block cycles
EXCTIME	3	REAL	C	R	control block execution time in seconds
PK_EXCTIME	4	REAL	C	R	peak control block execution time in seconds
EXECFAULT	7	BOOL	C	R	ON = executive fault
CYCTIME	8	REAL	C	R	control block cycle time in seconds
CYCMINS	9	REAL	C	R	control block cycle time in minutes
NEWSTART	10	BOOL	O	R	ON = new start
RESTART	11	BOOL	O	R	ON = warm start
ALM_ACTIV	12	BOOL	O	R	ON = active alarm
ALM_UNACK	13	BOOL	O	R	ON = unacknowledged alarm
HWOK	14	BOOL	O	R	ON = no hardware faults (See note 3)
LOWBTRY	15	BOOL	O	R	ON = battery is low
HITEMP	16	BOOL	O	R	ON = high RJ temperature
BAD_BLOCK	18	BOOL	O	R	ON = one or more blocks have bad status

Static Configuration Parameters: None

5.26 HLLM Function Block

Description

The **HLLM** label stands for **High Low Limiter**. This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder:



Dynamic Parameters:

Table 5-33 HLLM Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	primary output
H	2	BOOL	O	R	high limit indication
L	3	BOOL	O	R	low limit indication
X	4	REAL	I	R	input

Static Configuration Parameters:

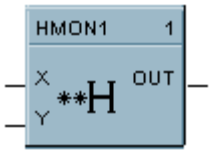
Table 5-34 HLLM Static Configuration Parameters

Parameter	Index	Type	Description
hilim	0	REAL	high limit {-99999 to 999999} for Analog X value
lolim	1	REAL	low limit {-99999 to 999999} for Analog X value

5.27 HMON Function Block

Description

The **HMON** label stands for **High Monitor.** This block is part of the *Alarm/Monitor* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-35 HMON Dynamic Parameters

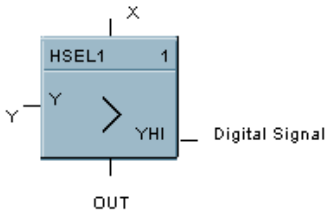
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	Output
X	2	REAL	I	R	Monitored input
Y	3	REAL	I	R	Trip point

Static Configuration Parameters: None

5.28 HSEL Function Block

Description

The **HSEL** label stands for **High Selector.** This block is part of the *Signal Selectors* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-36 HSEL Dynamic Parameters

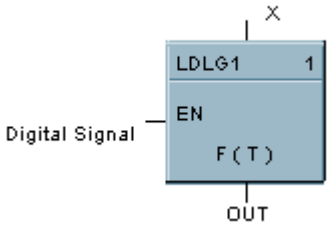
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	primary output
YHI(YLO)	2	BOOL	O	R	override indication
X	3	REAL	I	R	input
Y	4	REAL	I	R	input

Static Configuration Parameters: None

5.29 LDLG Function Block

Description

The **LDLG** label stands for **Lead/Lag.** This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-37 LDLG Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
IN	2	REAL	I	R	primary input
EN	3	BOOL	I	R	enable

Static Configuration Parameters: None

5.30 LMON Function Block

Description

The **LMON** label stands for **Low Monitor.** This block is part of the *Alarm/Monitor* category. It looks like this graphically on the Control Builder.

Dynamic Parameters:

Table 5-38 LMON Dynamic Parameters

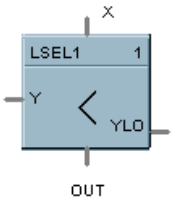
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	Output
X	2	REAL	I	R	Monitored input
Y	3	REAL	I	R	Trip point

Static Configuration Parameters: None

5.31 LSEL Function Block

Description

The **LSEL** label stands for **Low Selector.** This block is part of the *Signal Selectors* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-39 LSEL Dynamic Parameters

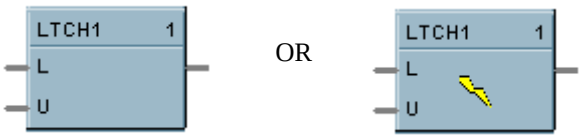
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	primary output
YHI(YLO)	2	BOOL	O	R	override indication
X	3	REAL	I	R	input
Y	4	REAL	I	R	input

Static Configuration Parameters: None

5.32 LTCH Function Block

Description

The **LTCH** label stands for **Latch.** This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-40 LTCH Dynamic Parameters

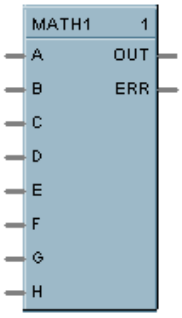
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
L	2	BOOL	I	R	latch command
U	3	BOOL	I	R	unlatch command

Static Configuration Parameters: None

5.33 MATH Function Block

Description

The **MATH** label stands for **Free Form Math.** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-41 MATH Dynamic Parameters

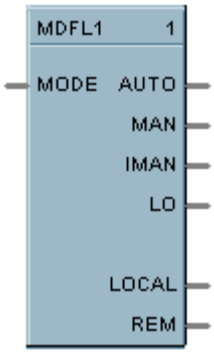
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	Result
ERR	2	BOOL	O	R	Error indication
A	3	REAL	I	R	Input 1
B	4	REAL	I	R	Input 2
C	5	REAL	I	R	Input 3
D	6	REAL	I	R	Input 4
E	7	REAL	I	R	Input 5
F	8	REAL	I	R	Input 6
G	9	REAL	I	R	Input 7
H	10	REAL	I	R	Input 8

Static Configuration Parameters: None

5.34 MDFL Function Block

Description

The **MDFL** label stands for **Mode Flag.** This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-42 MDFL Dynamic Parameters

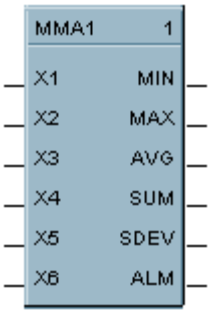
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
AUTO	1	BOOL	O	R	ON if MODE = 0.0 or 4.0, else OFF
MAN	2	BOOL	O	R	ON if MODE = 1.0 or 5.0, else OFF
IMAN	3	BOOL	O	R	ON if MODE = 2.0 or 6.0, else OFF
LO	4	BOOL	O	R	ON if MODE = 3.0 0r 7.0, else OFF
LOCAL	5	BOOL	O	R	ON if MODE > 3.0, else OFF
REM	6	BOOL	O	R	ON if MODE < 4.0, else OFF
MODE	7	REAL	I	R	Encoded mode input

Static Configuration Parameters: None

5.35 MMA Function Block

Description

The **MMA** label stands for **Min-Max-Average-Sum.** This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-43 MMA Dynamic Parameters

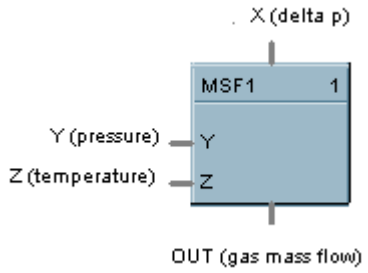
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
MIN	1	REAL	O	R	minimum input value
MAX	2	REAL	O	R	maximum input value
AVG	3	REAL	O	R	average of input values
SUM	4	REAL	O	R	sum of input values
SDEV	5	REAL	O	R	standard deviation of inputs
ALM	6	BOOL	O	R	deviation alarm
X1-X6	7-12	REAL	I	R	inputs

Static Configuration Parameters: None

5.36 MSF Function Block

Description

The **MSF** label stands for **Mass Flow Calculation.** This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-44 MSF Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	
Y	3	REAL	I	R	
Z	4	REAL	I	R	

Static Configuration Parameters:

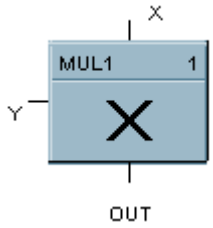
Table 5-45 MSF Static Configuration Parameters

Parameter	Index	Type	Description
Kg	0	REAL	Orifice Constant {-99999 to 999999}
Kx	1	REAL	Delta Pressure Scaler {-99999 to 999999}
Ky	2	REAL	Pressure Scaler {-99999 to 999999}
Kz	3	REAL	Temperature Scaler {-99999 to 999999}
Bx	4	REAL	Pressure Bias {-99999 to 999999}
By	5	REAL	Delta Pressure Bias {-99999 to 999999}
Bz	6	REAL	Temperature Bias {-99999 to 999999}
DROPOFF	7	REAL	low dropoff value (EU) {0 to 99999} [default 0]

5.37 MUL Function Block

Description

The **MUL** label stands for **Multiplication Mathematical Operation (2 Inputs).** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-46 MUL Dynamic Parameters

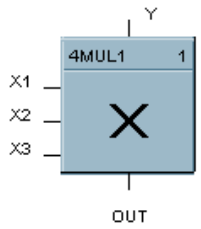
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input 1
Y	3	REAL	I	R	input 2

Static Configuration Parameters: None

5.38 4MUL Function Block

Description

The **4MUL** label stands for **Multiplication Mathematical Operation (4Inputs)**. This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-47 4MUL Dynamic Parameters

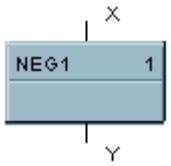
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
A	2	REAL	I	R	input 1
B	3	REAL	I	R	input 2
C	4	REAL	I	R	input 3
D	5	REAL	I	R	input 4

Static Configuration Parameters: None

5.39 NEG Function Block

Description

The **NEG** label stands for **Negate**. This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-48 NEG Dynamic Parameters

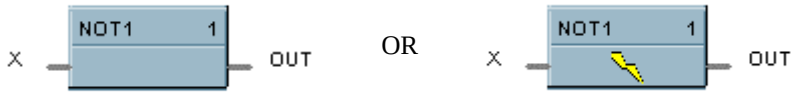
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	result
X	2	REAL	I	R	input

Static Configuration Parameters: None

5.40 NOT Function Block

Description

The **NOT** label stands for the **NOT Boolean logic function or Logic Inverter**. This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-49 NOT Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
X	2	BOOL	I	R	input

Static Configuration Parameters: None

5.41 ONDT Function Block

Description

The **ONDT** label stands for the **On Delay Timer**. This block is part of the *Fast Logic* category. It looks like this graphically on the Control Builder.

Dynamic Parameters:

Table 5-50 ONDT Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	W	output
RUN_RST	2	BOOL	I	R	ON = run, OFF = reset

Static Configuration Parameters:

Table 5-51 ONDT Static Parameters

Parameter	Index	Type	Description
Delay	0	REAL	Delay Time (0 seconds, 0 to 9999.9)

5.42 OFDT Function Block

Description

The **OFDT** label stands for the **Off Delay Timer**. This block is part of the *Fast Logic* category. It looks like this graphically on the Control Builder.

Dynamic Parameters:

Table 5-52 OFDT Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	W	output
RST_RUN	2	BOOL	I	R	ON = reset, OFF = run

Static Configuration Parameters:

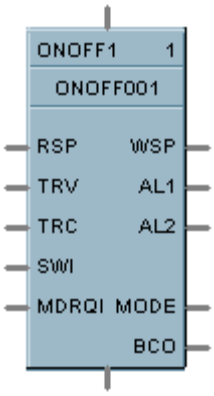
Table 5-53 ONDT Static Parameters

Parameter	Index	Type	Description
Delay	0	REAL	Delay Time (0 seconds, 0 to 9999.9)

5.43 ON/OFF Function Block

Description

The **ON/OFF** label stands for the **On/Off Control function**. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-54 ON/OFF Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
lsp	1	REAL	C	R/W	local set point (eu)
lsp2	2	REAL	C	R/W	local set point 2 (eu)
rem_mode	3	BOOL	C	R/W	remote set point mode request {OFF, ON}
man_mode	4	BOOL	C	R/W	manual output mode request {OFF, ON}
man_out	5	BOOL	C	R/W	On/Off
rsp_eu	6	REAL	C	R	remote set point in eu for monitoring
deviation	7	REAL	C	R	Deviation in eu for monitoring
pv	8	REAL	C	R	Process Variable in eu for monitoring
WSP	9	REAL	O	R	Working Set Point in eu for monitoring
OUT	10	BOOL	O	R	On/Off
MODE	11	REAL	O	R	actual mode encoded
All	12	BOOL	O	R	Alarm 1
AL2	13	BOOL	O	R	Alarm 2
BCO	14	REAL	O	R	Back Calculation Out (%)
PVI	15	REAL	I	R	Process Variable Input (eu) (pv_lo <= PV <= pv_hi)
RSP	16	REAL	I	R	Remote Set Point (% or eu per sp_units)
TRV	17	BOOL	I	R	On/Off
TRC	18	BOOL	I	R	Manual On/Off

Static Configuration Parameters:

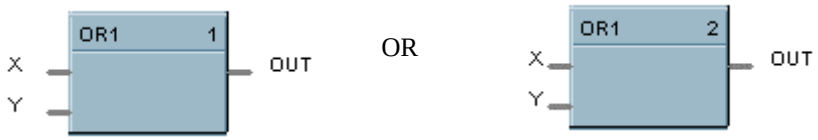
Table 5-55 ON/OFF Static Configuration Parameters

Parameter	Index	Type	Description
pv_hi	0	REAL	pv High Range value –99999 to 99999 (default **100**)
pv_lo	1	REAL	pv Low Range value –99999 to 99999 (default **0**)
sp_hi_lim	6	REAL	set point high limit, -99999 to 99999 (default **100**)
sp_lo_lim	7	REAL	set point low limit, -99999 to 99999 (default **0**)
sp_rate_dn	9	REAL	Set point low rate of change limit, **0** (off) to 99999 (eu/min)
sp_rate_up	10	REAL	Set point high rate of change limit, **0** (off) to 99999 (eu/min)
devbar_hi	11	REAL	High scale value for deviation bar graph (0 to 99999) (default **100**)
devbar_low	12	REAL	Low scale value for deviation bar graph [always = -devbar_hi]
Output Hysteresis	13	REAL	Off – 0 to 10% of Input Span
al_sp[4]	14-17	REAL	alarm set points al1sp1, al1sp2, al2sp1, al2sp2, -99999 to 99999 (default 0)
al_hyst	22	REAL	alarm hysterysis 0 to 5 (%)
Fail Safe Out	23	BOOL	Fail Safe Out = On/Off

5.44 2OR Function Block

Description

The **2OR** label stands for the inclusive **OR (2 Inputs) Boolean logic function**. This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-56 2OR Dynamic Parameters

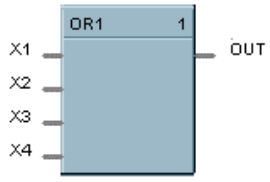
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1	2	BOOL	I	R	input
DIG_2	3	BOOL	I	R	input

Static Configuration Parameters: None

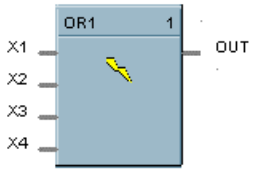
5.45 4OR Function Block

Description

The **4OR** label stands for the inclusive **OR (4 Inputs) Boolean logic function**. This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



OR



Fast Logic

Dynamic Parameters:

Table 5-57 4OR Dynamic Parameters

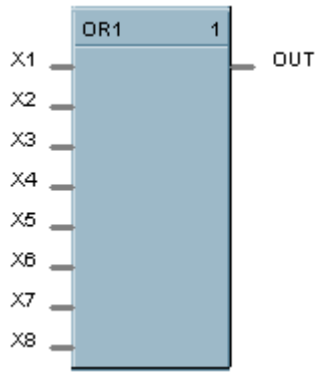
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1	2	BOOL	I	R	input
DIG_2	3	BOOL	I	R	input
DIG_3	4	BOOL	I	R	input
DIG_4	5	BOOL	I	R	input

Static Configuration Parameters: None

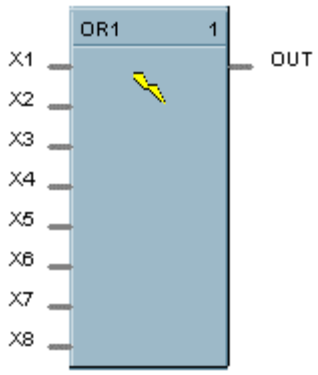
5.46 8OR Function Block

Description

The **8OR** label stands for the inclusive **OR (8 Inputs) Boolean logic function**. This block is part of the *Logic or Fast Logic* category. It looks like this graphically on the Control Builder.



OR



Fast Logic

Dynamic Parameters:

Table 5-58 8OR Dynamic Parameters

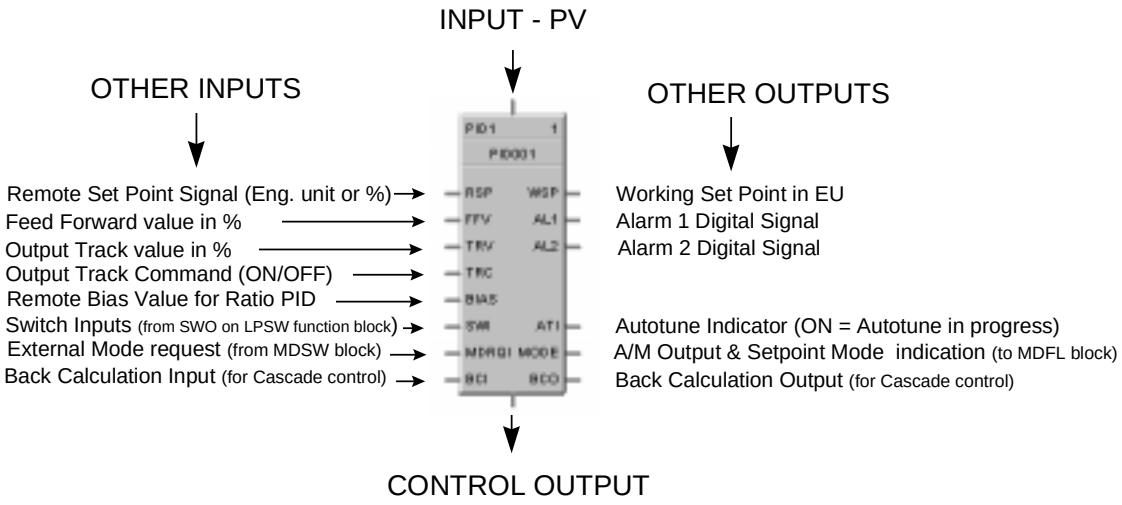
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1	2	BOOL	I	R	input
DIG_2	3	BOOL	I	R	input
DIG_3	4	BOOL	I	R	input
DIG_4	5	BOOL	I	R	input
DIG_5	6	BOOL	I	R	input
DIG_6	7	BOOL	I	R	input
DIG_7	8	BOOL	I	R	input
DIG_8	9	BOOL	I	R	input

Static Configuration Parameters: None

5.47 PID Function Block

Description

The **PID** label stands for **Proportional, Integral, Derivative (3-mode)** control action. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-59 PID Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
lsp	1	REAL	C	R/W	local set point (eu)
lsp2	2	REAL	C	R/W	local set point 2 (eu)
rem_mode	3	BOOL	C	R/W	remote set point mode request {OFF, ON}
man_mode	4	BOOL	C	R/W	manual output mode request {OFF, ON}
man_out	5	REAL	C	R/W	manual output value –5 to 105 (%)
tune_req	6	BOOL	C	R/W	limit cycle auto-tuning request {OFF, ON}
rsp_eu	7	REAL	C	R	remote set point in eu for monitoring
deviation	8	REAL	C	R	Deviation in eu for monitoring
pv	9	REAL	C	R	Process Variable in eu for monitoring
WSP	10	REAL	O	R	Working setpoint display in eu for monitoring
OUT	11	REAL	O	R	control output –5 to 105 (%)
MODE	12	REAL	O	R	actual mode encoded
ALl	13	BOOL	O	R	Alarm 1
AL2	14	BOOL	O	R	Alarm 2

BCO	15	REAL	O	R	Back Calculation Out (%)
ATI	16	BOOL	O	R	Auto Tune Indicator. ON = Auto Tune in progress
PVI	17	REAL	I	R	Process Variable Input (eu) (pv_lo <= PV <= pv_hi)
RSP	18	REAL	I	R	Remote Set Point (% or eu per sp_units)
FFV	19	REAL	I	R	Feed Forward Value (%)
TRV	20	REAL	I	R	Output Track Value (%)
TRC	21	BOOL	I	R	Output Track Command {OFF, ON}
BCI	23	REAL	I	R	Back Calculation Input (%)
BIAS	24	REAL	I	R	Remote bias value for ratio PID

Static Configuration Parameters:

Table 5-60 PID Static Configuration Parameters

Parameter	Index	Type	Description
GAIN	0	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 1]
RATE	1	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 1]
RESET	2	REAL	integration time, **0** or 0.02 to 50 (minutes) or repeats per minute, 0 or 0.02 to 50 (repeats) [Tune Set 1]
pv_hi	4	REAL	pv High Range value –99999 to 99999 (default **100**)
pv_lo	5	REAL	pv Low Range value –99999 to 99999 (default **0**)
sp_hi_lim	11	REAL	set point high limit, -99999 to 99999 (default **100**)
sp_lo_lim	12	REAL	set point low limit, -99999 to 99999 (default **0**)
outhilim	14	REAL	output high limit, -5 to **105**
outlolim	15	REAL	output low limit, **-5** to 105
failsafe	16	REAL	failsafe output value, -5 to 105, (default **0**)
al_sp[4]	17-20	REAL	alarm set points al1spl, al1sp2, al2spl, al2sp2, -99999 to 99999 (default **0**)
al_hyst	25	REAL	alarm hysterysis **0** to 5 (%)
man_reset	26	REAL	Manual Reset, -100 to 100 (in % output) (default **0**) [used for both tune sets]
FUZZY	28	BOOL	ON enables fuzzy logic overshoot suppression (default **OFF**)
TUNESET2	29	BOOL	Use tune set 2 (default **OFF**)
GAIN2	30	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 2]
RATE2	31	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 2]

RESET2	32	REAL	integration time, **0** or 0.02 to 50 (minutes) or repeats per minute, 0 or 50 to 0.02 (repeats) [Tune Set 2]
use_propband	33	BOOL	Use Gain (**0**) or Proportional Band (1)
use_rpm	34	BOOL	Use minutes (**0**) or repeats per minute (1) for integral constant
sp_rate_dn	35	REAL	Set point low rate of change limit, **0** (off) to 99999 (eu/min)
sp_rate_up	36	REAL	Set point high rate of change limit, **0** (off) to 99999 (eu/min)
FF_GAIN	37	REAL	Feed forward gain, **0.0** to 10.0
RATIO	39	REAL	Gain value for Ratio PID (-20 to 20) (default **1**) [used when RA_BIAS > 0]
LBIAS	40	REAL	Bias value for Ratio PID when RA_BIAS = LOC_BIAS](-99999 to 99999) (**0**)
devbar_hi	41	REAL	High scale value for deviation bar graph (0 to 99999) (default **100**)
devbar_low	42	REAL	Low scale value for deviation bar graph [always = -devbar_hi]

5.48 PTMR Function Block

Description

The **PTMR** label stands for **Periodic Timer**. This block is part of the *Logic* category. It looks like this graphically on the Control Builder.

Dynamic Parameters:

Table 5-61 PTMR Dynamic Parameters

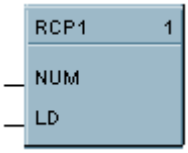
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
EVENT	1	BOOL	O	R	output
RST	2	BOOL	I	R	reset input

Static Configuration Parameters: None

5.49 RCP Function Block

Description

The **RCP** label stands for **Recipe Selector**. This block is part of the *Setpoint Program* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-62 RCP Dynamic Parameters

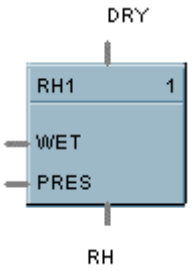
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
NUM	1	REAL	I	R	recipe number
LOAD	2	BOOL	I	R	load command

Static Configuration Parameters: None

5.50 RH Function Block

Description

The **RH** label stands for **Relative Humidity**. This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-63 RH Dynamic Parameters

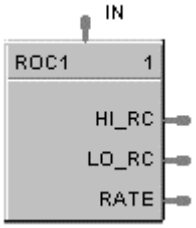
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
RH	1	REAL	O	R	Relative Humidity
DRY	2	REAL	I	R	Dry bulb temperature
WET	3	REAL	I	R	Wet bulb temperature
PRES	4	REAL	I	R	Atmospheric Pressure

Static Configuration Parameters: None

5.51 ROC Function Block

Description

The **ROC** label stands for **Rate of Change**. This block is part of the Auxiliary category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-64 ROC Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
hi_rc	1	BOOL	O	R	ON if rate > setpoint, else OFF
lo_rc	2	BOOL	O	R	ON if rate < setpoint, else OFF
rate	3	REAL	O	R	Rate of Change in EU/min
IN	4	REAL	I	R	Analog Input

Static Configuration Parameters:

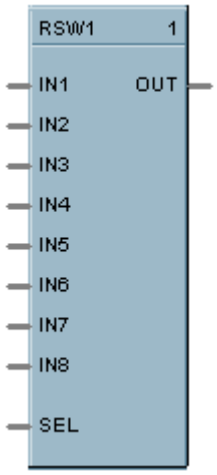
Table 5-65 ROC Static Configuration Parameters

Parameter	Index	Type	Description
filt_time	0	REAL	filter time constant
range_hi	1	REAL	high rate of change setpoint Range = 0 (off) to 99999.9 eu/min
range_lo	2	REAL	low rate of change setpoint Range = 0 (off) to 99999.9 eu/min
direction_hi	3	REAL	0 = both, 1 = increment only, 2 = decrement only
direction_lo	4	REAL	0 = both, 1 = increment only, 2 = decrement only
hysteresis	5	REAL	Range (0 – 999)

5.52 RSW Function Block

Description

The **RSW** label stands for **Rotary Switch**. This block is part of the *Signal Selectors* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-66 RSW Dynamic Parameters

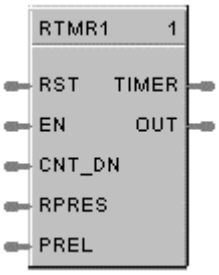
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
IN1	2	REAL	I	R	input 1
IN2	3	REAL	I	R	input 2
IN3	4	REAL	I	R	input 3
IN4	5	REAL	I	R	input 4
IN5	6	REAL	I	R	input 5
IN6	7	REAL	I	R	input 6
IN7	8	REAL	I	R	input 7
IN8	9	REAL	I	R	input 8
SEL	10	REAL	I	R	select input to output

Static Configuration Parameters: None

5.53 RTMR Function Block

Description

The **RTMR** label stands for **Resettable Timer**. This block is part of the *Counters/Timers* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-67 RTMR Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
OUT	1	BOOL	O	R	output
TIMER	2	REAL	O	R	remaining time in seconds
RPRES	3	REAL	I	R/W	Remote preset (0.0 – 99999.9) if 'count-up' then represents Stop value in seconds if 'count-down' then represents Start value in seconds
RST	4	BOOL	I	R	OFF to ON transition, Reset
EN	5	BOOL	I	R	ENABLE ON = run; timer is counting, OFF = timer is stopped; output (TIMER) held at last value
PREL	6	REAL	I	R	Preload: if 'count-up' then represents Start value in seconds if 'count-down' then represents Stop value in seconds (Range = 0.0 – 99999.9)
CNT_DN	7	BOOL	I	R	ON = count-down, OFF = count-up

Static Configuration Parameters:

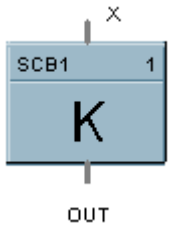
Table 5-68 RTMR Static Configuration Parameters

Parameter	Index	Type	Description
lpres	0	REAL	Local preset (0.0 – 99999.9) if count-up then Stop value in seconds if count-down then Start value in seconds
remote	1	BOOL	ON = use Remote Preset, OFF = use Local Preset
use_preload	2	BOOL	Use external preload rather than zero for starting or stopping.

5.54 SCB Function Block

Description

The **SCB** label stands for **Scale and Bias**. This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-69 SCB Dynamic Parameters

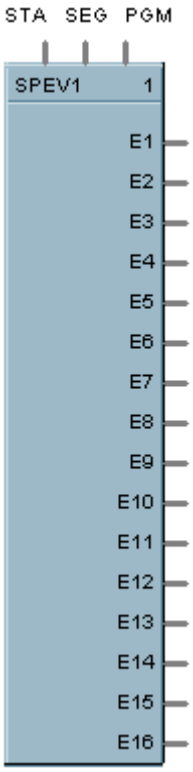
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input

Static Configuration Parameters: None

5.55 SPEV Function Block

Description

The **SPEV** label stands for **Setpoint Programming Events**. This block is part of the *Setpoint Program* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-70 SPEV Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
E1	1	BOOL	O	R	event 1
E2	2	BOOL	O	R	event 2
E3	3	BOOL	O	R	event 3
E4	4	BOOL	O	R	event 4
E5	5	BOOL	O	R	event 5
E6	6	BOOL	O	R	event 6
E7	7	BOOL	O	R	event 7
E8	8	BOOL	O	R	event 8
E9	9	BOOL	O	R	event 9

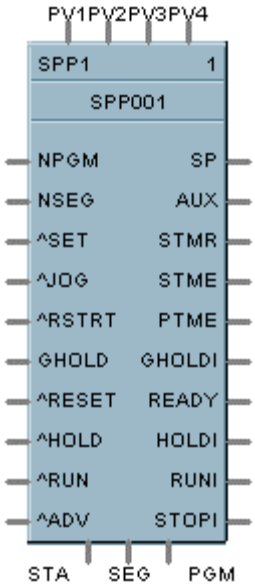
E10	10	BOOL	O	R	event 10
E11	11	BOOL	O	R	event 11
E12	12	BOOL	O	R	event 12
E13	13	BOOL	O	R	event 13
E14	14	BOOL	O	R	event 14
E15	15	BOOL	O	R	event 15
E16	16	BOOL	O	R	event 16
STA	17	REAL	I	R	program state
SEG	18	REAL	I	R	current segment number
PGM	19	REAL	I	R	current program number

Static Configuration Parameters: None

5.56 SPP Function Block

Description

The **SPP** label stands for **Setpoint Programmer**. This block is part of the *Setpoint Program* category. It looks like this graphically on the Control Builder.



Dynamic Contained Parameters:

Table 5-71 SPP Dynamic Contained Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
save_req	1	REAL	C	R/W	save current program request mailbox
sta_req	2	REAL	C	R/W	state change request
pgm_req	3	REAL	C	R/W	program change request
seg_req	4	REAL	C	R/W	segment change request
seg_ramp	5	BOOL	C	R	TRUE if current segment is a ramp
soaksp	6	REAL	C	R	soak setpoint (eu)
duration	7	REAL	C	R	segment duration (minutes)
PV	8	REAL	C	R	value of PV being controlled by SP program
adv_req	9	BOOL	C	R/W	segment advance request (leading edge)
lTimeStamp	16	LONG	C	R	Time of last modification
fGuarSoako	17	REAL	C	R/W	guaranteed soak low limit
fGuarSoakHi	18	REAL	C	R/W	guaranteed soak high limit

fRestartRamp	19	REAL	C	R/W	ramp rate to use when power down restart is in effect
fJogSeg	20	REAL	C	R/W	segment jumped to on a pulse to JOG input
fLoopStart	21	REAL	C	R/W	first segment in a loop – 0->no loop
fLoopEnd	22	REAL	C	R/W	last segment in a loop – 0->no loop
fDispHiLim	28	REAL	C	R/W	output high limit for display purposes only
fDispLoLim	29	REAL	C	R/W	output low limit for display purposes only
bRampTime	31	BOOL	C	R/W	TRUE = ramp time, FALSE = ramp rate

Dynamic Output Parameters:

Table 5-72 SPP Dynamic Output Parameters

Parameter	Index	Type	Use	R/W	Description
STA	34	REAL	O	R	program state {N/A, RESET, RUN, HOLD, GHOLD, STOP}
SEG	35	REAL	O	R	current segment number
PGM	36	REAL	O	R	current program number
SP	37	REAL	O	R	setpoint output (EU)
AUX	38	REAL	O	R	auxilary output (EU)
STMR	39	REAL	O	R	time remaining in current segment (minutes)
STME	40	REAL	O	R	time elapsed in current segment (minutes)
PTME	41	REAL	O	R	time elapsed in program (minutes).
GHOLDI	42	BOOL	O	R	set when program is in the GHOLD state
READY	43	BOOL	O	R	set when program is in RESET state
RUNI	44	BOOL	O	R	set when program is in RUN state
HOLDI	45	BOOL	O	R	set when program is in HOLD state
STOPI	46	BOOL	O	R	set when program is in STOP state

Dynamic Input Parameters:

Table 5-73 SPP Dynamic Input Parameters

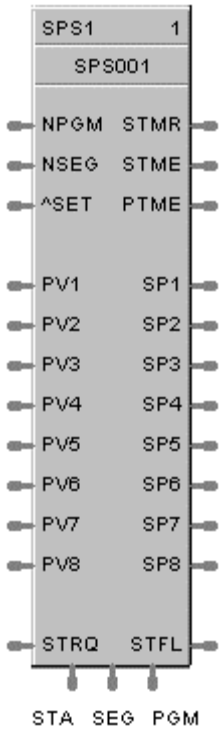
Parameter	Index	Type	Use	R/W	Description
PV1	47	REAL	I	R	process variable (EU), for deviation check
PV2	48	REAL	I	R	2nd process variable (EU), for deviation check
PV3	49	REAL	I	R	3rd process variable (EU), for deviation check
PV4	50	REAL	I	R	4th process variable (EU), for deviation check
NPGM	51	REAL	I	R	program number (when SET is ON)
NSEG	52	REAL	I	R	starting segment number (when SET is ON)
SET	53	BOOL	I	R	pulse input to load PGM and SSEG numbers
JOG	54	BOOL	I	R	pulse input for jog
RSTRT	55	BOOL	I	R	pulse input for restart action
GHOLD	56	BOOL	I	R	guaranteed soak hold, for sync
RESET	57	BOOL	I	R	pulse input for reset
HOLD	58	BOOL	I	R	pulse input for hold
RUN	59	BOOL	I	R	pulse input for run
ADV	60	BOOL	I	R	pulse input for advance

Static Configuration Parameters: None

5.57 SPS Function Block

Description

The **SPS** label stands for **Master Setpoint Scheduler**. This block is part of the *Setpoint Scheduler* category. It looks like this graphically on the Control Builder.



Dynamic Contained Parameters:

Table 5-74 SPS Dynamic Contained Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
save_req	1	REAL	C	R/W	save current program request mailbox
sta_req	2	REAL	C	R/W	state change request
pgm_req	3	REAL	C	R/W	program change request
seg_req	4	REAL	C	R/W	segment change request
soaksp [8]	5-12	REAL	C	R	soak setpoint (eu)
duration	13	REAL	C	R	segment duration (minutes/hours)
PV [8]	14-21	REAL	C	R	value of PV being controlled by SP program
adv_req	22	BOOL	C	R/W	segment advance request (leading edge)

fGuarLimit[8]	30-37	REAL	C	R/W	guaranteed soak low/high limit
fJogSeg	38	REAL	C	R/W	segment jumped to on a pulse to JOG input

Dynamic Output Parameters:

Table 5-75 SPS Dynamic Output Parameters

Parameter	Index	Type	Use	R/W	Description
STA	40	REAL	O	R	program state {N/A, RESET, RUN, HOLD, GHOLD, STOP}
SEG	41	REAL	O	R	current segment number
PGM	42	REAL	O	R	current program number
SP1	43	REAL	O	R	Setpoint #1 output (EU)
SP2	44	REAL	O	R	Setpoint #2 output (EU)
SP3	45	REAL	O	R	Setpoint #3 output (EU)
SP4	46	REAL	O	R	Setpoint #4 output (EU)
SP5	47	REAL	O	R	Setpoint #5 output (EU)
SP6	48	REAL	O	R	Setpoint #6 output (EU)
SP7	49	REAL	O	R	Setpoint #7 output (EU)
SP8	50	REAL	O	R	Setpoint #8 output (EU)
STMR	51	REAL	O	R	time remaining in current segment (minutes)
STME	52	REAL	O	R	time elapsed in current segment (minutes)
PTME	53	REAL	O	R	time elapsed in program (minutes).
STFL	54	REAL	O	R	current state flag

Dynamic Input Parameters:

Table 5-76 SPS Dynamic Input Parameters

Parameter	Index	Type	Use	R/W	Description
PV1	55	REAL	I	R	1st process variable (EU)
PV2	56	REAL	I	R	2nd process variable (EU)
PV3	57	REAL	I	R	3rd process variable (EU)
PV4	58	REAL	I	R	4th process variable (EU)
PV5	59	REAL	I	R	5th process variable (EU)
PV6	60	REAL	I	R	6th process variable (EU)
PV7	61	REAL	I	R	7th process variable (EU)
PV8	62	REAL	I	R	8th process variable (EU)
STRQ	63	REAL	I	R	Encoded state request from STSW block
NPGM	64	REAL	I	R	Program number (when SET is ON)
NSEG	65	REAL	I	R	Starting segment number (when SET is ON)
SET	66	BOOL	I	R	Pulse input to load PGM and SSEG numbers

Static Configuration Parameters:

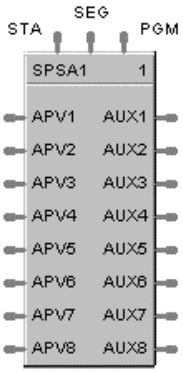
Table 5-77 SPS Static Configuration Parameters

Index	Parameter	Type	Description
0 - 7	failsafe[8]	REAL	failsafe setpoint, value (eu)

5.58 SPSA Function Block

Description

The **SPSA** label stands for **Setpoint Scheduler Auxiliary Setpoint Block**. This block is part of the *Setpoint Scheduler*category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-78 SPSA Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
AUX1	1	REAL	O	R	Auxiliary output #1
AUX2	2	REAL	O	R	Auxiliary output #2
AUX3	3	REAL	O	R	Auxiliary output #3
AUX4	4	REAL	O	R	Auxiliary output #4
AUX5	5	REAL	O	R	Auxiliary output #5
AUX6	6	REAL	O	R	Auxiliary output #6
AUX7	7	REAL	O	R	Auxiliary output #7
AUX8	8	REAL	O	R	Auxiliary output #8
STA	9	REAL	I	R	Program state (for configuration - cosmetic only)
SEG	10	REAL	I	R	Current segment number
PGM	11	REAL	I	R	Current program number
APV1	12	REAL	I	R	1st Aux. process variable (EU)
APV2	13	REAL	I	R	2nd Aux. process variable (EU)
APV3	14	REAL	I	R	3rd Aux. process variable (EU)
APV4	15	REAL	I	R	4th Aux. process variable (EU)
APV5	16	REAL	I	R	5th Aux. process variable (EU)
APV6	17	REAL	I	R	6th Aux. process variable (EU)
APV7	18	REAL	I	R	7th Aux. process variable (EU)
APV8	19	REAL	I	R	8th Aux. process variable (EU)

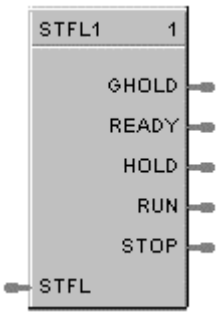
Static Configuration Parameters:

None

5.59 STFL Function Block

Description

The **STFL** label stands for the **Setpoint Scheduler State Flags.** This block is part of the *Setpoint Scheduler* category. It looks like this graphically on the Control Builder.



Dynamic Values:

Table 5-79 STFL Dynamic Values

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
GHOLD	1	BOOL	O	R	ON if state = 1.0, else OFF
READY	2	BOOL	O	R	ON if state = 2.0, else OFF
HOLD	3	BOOL	O	R	ON if state = 4.0, else OFF
RUN	4	BOOL	O	R	ON if state = 8.0, else OFF
STOP	5	BOOL	O	R	ON if state = 16.0, else OFF
STFL	6	REAL	I	R	Encoded state input (See Note 1)

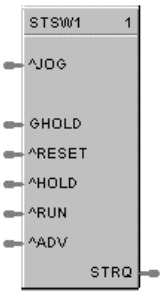
Static Configuration Values

None

5.60 STSW Function Block

Description

The **STSW** label stands for the **Setpoint Scheduler State Switch.** This block is part of the *Setpoint Scheduler* category. It looks like this graphically on the Control Builder.



Dynamic Values:

Table 5-80 STSW Dynamic Values

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
STRQ	1	REAL	O	R	Encoded state request output
JOG	2	BOOL	I	R	OFF to ON requests jog state
GHOLD	3	BOOL	I	R	ON = guaranteed hold state ON to OFF and previous state was Run, then return to RUN mode
RESET	4	BOOL	I	R	OFF to ON requests reset state
HOLD	5	BOOL	I	R	OFF to ON requests hold state
RUN	6	BOOL	I	R	OFF to ON requests run state
ADV	7	BOOL	I	R	OFF to ON requests advance state

Static Configuration Values

None

5.61 SQRT Function Block

Description

The **SQRT** label stands for **Square Root**. This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.

Dynamic Parameters:

Table 5-81 SQRT Dynamic Parameters

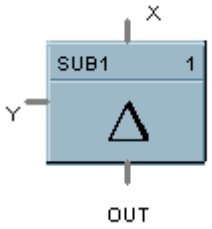
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	result
X	2	REAL	I	R	input

Static Configuration Parameters: None

5.62 SUB Function Block

Description

The **SUB** label stands for the **Subtraction mathematical operation (2 Inputs).** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-82 SUB Dynamic Parameters

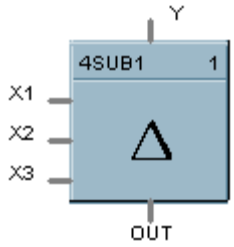
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	input 1
Y	3	REAL	I	R	input 2

Static Configuration Parameters: None

5.63 4SUB Function Block

Description

The **4SUB** label stands for the **Subtraction mathematical operation (4 Inputs).** This block is part of the *Math* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-83 4SUB Dynamic Parameters

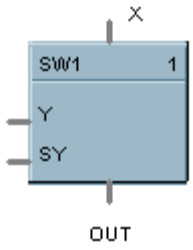
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
A	2	REAL	I	R	input 1
B	3	REAL	I	R	input 2
C	4	REAL	I	R	input 3
D	5	REAL	I	R	input 4

Static Configuration Parameters: None

5.64 SW Function Block

Description

The **SW** label stands for **Analog Switch**. This block is part of the *Signal Selectors* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-84 SW Dynamic Parameters

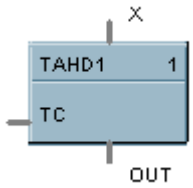
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	
Y	3	REAL	I	R	
SY	4	BOOL	I	R	select Y when ON

Static Configuration Parameters: None

5.65 TAHD Function Block

Description

The **TAHD** label stands for **Track and Hold.** This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-85 TAHD Dynamic Parameters

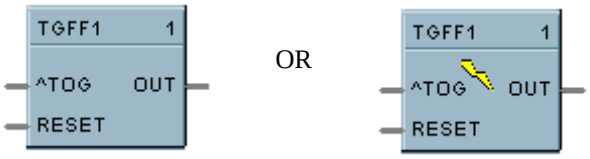
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	primary input
TC	3	BOOL	I	R	track command

Static Configuration Parameters: None

5.66 TGFF Function Block

Description

The **TGFF** label stands for **Toggle Flip-Flop.** This block is part of the *Logic or Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-86 TGFF Dynamic Parameters

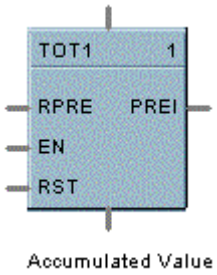
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
TOG	2	BOOL	I	R	toggle input
RESET	3	BOOL	I	R	reset input

Static Configuration Parameters: None

5.67 TOT Function Block

Description

The **TOT** label stands for **Totalizer.** This block is part of the *Calculations* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-87 TOT Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
TOT	1	REAL	O	R	total (eu)
PREI	2	BOOL	O	R	preset indicator
IN	3	REAL	I	R	analog input value (eu)
RPRE	4	REAL	I	R	remote preset in eu (1 to 999999)
EN	5	BOOL	I	R	ON enables the totalizer
RST	6	BOOL	I	R	ON resets the totalizer

Static Configuration Parameters:

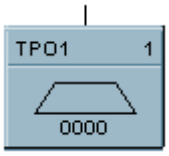
Table 5-88 TOT Static Configuration Parameters

Parameter	Index	Type	Description
lpre	1	REAL	local preset (1 to 999999)
rem	2	BOOL	ON selects remote preset
decr	3	BOOL	ON selects decreasing from preset

5.68 TPO Function Block

Description

The **TPO** label stands for **Time Proportional Output.** This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-89 TPO Dynamic Parameters

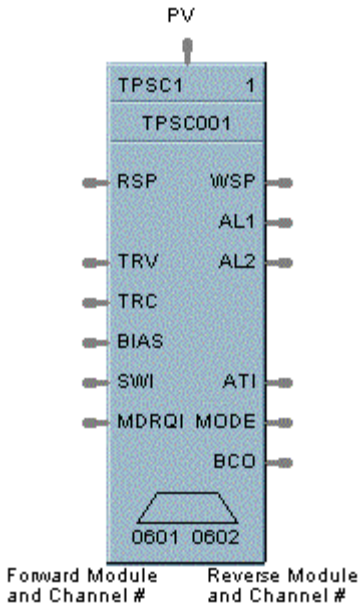
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
IN	1	REAL	I	R	analog input value (usually %)

Static Configuration Parameters: None

5.69 TPSC (3POS) Function Block

Description

The **TPSC (3POS)** label stands for **Three Position Step Control operation**. This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-90 TPSC Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
lsp	1	REAL	C	R/W	local set point (eu)
lsp2	2	REAL	C	R/W	local set point 2 (eu)
rem_mode	3	BOOL	C	R/W	remote set point mode request {OFF, ON}
man_mode	4	BOOL	C	R/W	manual output mode request {OFF, ON}
man_out	5	REAL	C	R/W	manual output value 0 to 100 (%)
tune_req	6	BOOL	C	R/W	limit cycle auto-tuning request {OFF, ON}
rsp_eu	7	REAL	C	R	remote set point in eu for monitoring
deviation	8	REAL	C	R	Deviation in eu for monitoring
pv	9	REAL	C	R	Process Variable in eu for monitoring
fbpct	10	REAL	C	R	Percent feedback estimation for monitoring
DSP	11	REAL	O	R	Display Set Point in eu for monitoring
MODE	12	REAL	O	R	actual mode encoded per note 8

All	13	BOOL	O	R	Alarm 1
AL2	14	BOOL	O	R	Alarm 2
BCO	15	REAL	O	R	Back Calculation Out (%)
ATI	16	BOOL	O	R	Auto Tune Indicator. ON = Auto Tune in progress
PVI	17	REAL	I	R	Process Variable Input (eu) (pv_lo <= PV <= pv_hi)
RSP	18	REAL	I	R	Remote Set Point (% or eu per sp_units)
TRV	19	REAL	I	R	Output Track Value (%)
TRC	20	BOOL	I	R	Output Track Command {OFF, ON}
BIAS	22	REAL	I	R	Remote bias value for ratio PID

Static Configuration Parameters:

Table 5-91 TPSC Static Configuration Parameters

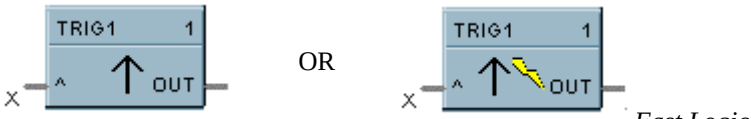
Parameter	Index	Type	Description
GAIN	0	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 1]
RATE	1	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 1]
RESET	2	REAL	integration time, 0.02 to 50 (minutes) or repeats per minute, 0.02 to 50 (repeats) [Tune Set 1]
pv_hi	3	REAL	pv High Range value –99999 to 99999 (default **100**)
pv_lo	4	REAL	pv Low Range value –99999 to 99999 (default **0**)
sp_hi_lim	10	REAL	set point high limit, -99999 to 99999 (default **100**)
sp_lo_lim	11	REAL	set point low limit, -99999 to 99999 (default **0**)
failsafe_hi	13	BOOL	ON sets motor to 100% when in failsafe. OFF sets motor to 0% (default **OFF**)
al_sp[4]	14-17	REAL	alarm set points al1spl, al1sp2, al2spl, al2sp2, -99999 to 99999 (default **0**)
al_hyst	22	REAL	alarm hysterysis **0** to 5 (%)
ATOUTHILIM	25	REAL	auto-tuning output high limit, 0% to **100%**
ATOUTLOLIM	26	REAL	auto-tuning output low limit, **0%** to 100%
FUZZY	27	BOOL	ON enables fuzzy logic overshoot suppression (default **OFF**)
TUNESET2	28	BOOL	Use tune set 2 (default **OFF**)
GAIN2	29	REAL	proportional gain, 0.1 to 1000 or proportional band, 0.1% to 1000% [Tune Set 2]
RATE2	30	REAL	derivative time, **0** or 0.1 to 10 (minutes) [Tune Set 2]
RESET2	31	REAL	integration time, 0.02 to 50 (minutes) or repeats per minute, 50 to 0.02 (repeats) [Tune Set 2]
use_propband	32	BOOL	Use Gain (**0**) or Proportional Band (1)

use_rpm	33	BOOL	Use minutes (**0**) or repeats per minute (1) for integral constant
sp_rate_dn	34	REAL	Set point low rate of change limit, **0** (off) to 99999 (eu/min)
sp_rate_up	35	REAL	Set point high rate of change limit, **0** (off) to 99999 (eu/min)
RATIO	37	REAL	Gain value for Ratio PID (-20 to 20) (default **1**) [used when RA_BIAS > 0]
LBIAS	38	REAL	Bias value for Ratio PID when RA_BIAS = LOC_BIAS](-99999 to 99999) (**0**)
devbar_hi	39	REAL	High scale value for deviation bar graph (0 to 99999) (default **100**)
devbar_low	40	REAL	Low scale value for deviation bar graph [always = -devbar_hi]
deadband	43	REAL	adjustable gap between forward and reverse motor operation (.5 to 5%)

5.70 TRIG Function Block

Description

The **TRIG** label stands for **Trigger or "One Shot" operation**. This block is part of the *Logic* or *Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-92 TRIG Dynamic Parameters

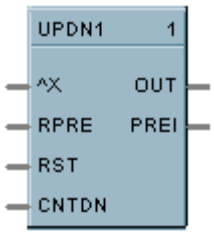
Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
X	2	BOOL	I	R	input

Static Configuration Parameters: None

5.71 UPDN Function Block

Description

The **UPDN** label stands for **UP/DOWN Counter.** This block is part of the *Logic* category. It looks like this graphically on the Control Builder:



Dynamic Parameters:

Table 5-93 UPDN Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	W	output
PREI	2	BOOL	O	W	preset indicator
X	3	BOOL	I	R	positive edge detect count input
RPRE	4	REAL	I	R	remote preset (1 to 999999)
RST	5	BOOL	I	R	ON resets the count
CNTDN	6	BOOL	I	R	ON counts down

Static Configuration Parameters:

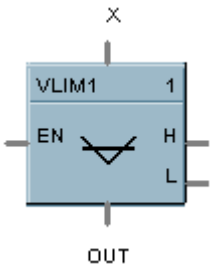
Table 5-94 UPDN Static Configuration Parameters

Parameter	Index	Type	Description
lpre	0	REAL	local preset (1 to 99999)
rem	1	BOOL	ON selects remote preset

5.72 VLIM Function Block

Description

The **VLIM** label stands for **Velocity (Rate) Limiter.** This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-95 VLIM Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	primary output
H	2	BOOL	O	R	high rate limit indication
L	3	BOOL	O	R	low rate limit indication
X	4	REAL	I	R	primary input
EN	5	BOOL	I	R	enable input

Static Configuration Parameters:

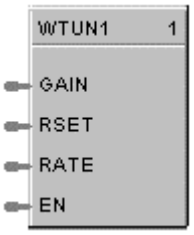
Table 5-96 VLIM Static Configuration Parameters

Parameter	Index	Type	Description
irate	0	REAL	increase rate limit (eu/min, >=0) {0 to 99999}
drate	1	REAL	decrease rate limit (eu/min, >=0) {0 to 99999}

5.73 WTUN Function Block

Description

The **WTUN** label stands for **Write Tuning Constants.** This block is part of the *Loops* category. It looks like this graphically on the Control Builder.



Dynamic Values

Table 5-97 WTUN Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
GAIN	1	REAL	I	R	proportional gain, 0.1 to 1000
RSET	2	REAL	I	R	integration time, 0.02 to 50 (minutes)
RATE	3	REAL	I	R	derivative time, 0.1 to 10 (minutes)
EN	4	BOOL	I	R	enable

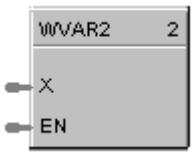
Static Configuration Values

None

5.74 WVAR Function Block

Description

The **WVAR** label stands for **Write Variable.** This block is part of the *Auxiliary* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-98 WVAR Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status
X	1	REAL or BOOL	I	R	value to be written
EN	2	BOOL	I	R	enable change

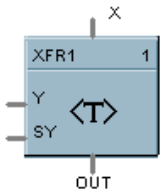
Static Configuration Parameters:

None

5.75 XFR Function Block

Description

The **XFR** label stands for **Bumpless Analog Transfer Switch**. This block is part of the *Signal Selectors* category. It looks like this graphically on the Control Builder.



Dynamic Parameters:

Table 5-99 XFR Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	REAL	O	R	output
X	2	REAL	I	R	
Y	3	REAL	I	R	
SY	4	BOOL	I	R	select Y when ON

Static Configuration Parameters:

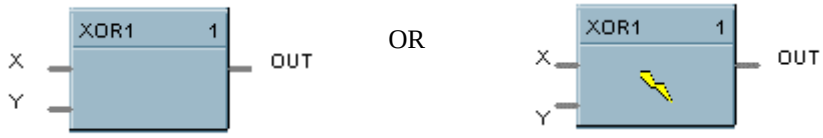
Table 5-100 XFR Static Configuration Parameters

Parameter	Index	Type	Description
xrate	0	REAL	transfer to X rate (eu/min, >=0) {0 to 99999}
yrate	1	REAL	transfer to Y rate (eu/min, >=0) {0 to 99999}

5.76 XOR Function Block

Description

The **XOR** label stands for the **Exclusive OR Boolean operation**. This block is part of the *Logic and Fast Logic* category. It looks like this graphically on the Control Builder.



Fast Logic

Dynamic Parameters:

Table 5-101 XOR Dynamic Parameters

Parameter	Index	Type	Use	R/W	Description
status	0	REAL	C	R	block status (see section 5.2 for code list)
OUT	1	BOOL	O	R	output
DIG_1	2	BOOL	I	R	input
DIG_2	3	BOOL	I	R	input

Static Configuration Parameters: None

5.77 Variables

Description

Assigned Block Number (250) provides a means of reading from and writing to various arrays of variables.

List Block Number 250 in your request message and the required index number. The index numbers (0 to 149) for the various variables can be obtained from a LeaderLine Control Builder variable list printout.

ATTENTION

For Communications, subtract 1 from the variable index number on the Print out.

For Example: Variable 1 will be Variable 0 for communications purposes.

Dynamic Parameters:

Table 5-102 Variables

Parameter	Index	Type	Use	R/W	Description
Variable Parameter	0 - 149	REAL	C	R	Array of Variables (0 to 99999.9)

6. Block Status Types

6.1 Overview

Introduction

Table 6-1 lists the Function Block Status Values and definitions for communication reference when Index 0 is requested in a Dynamic (I/O) Table of a function block.

Common Function Block Status Types

These status types are common to all block types:

- UNEXECUTED
- OK
- BAD TYPE
- BAD ICNT (Input Count)
- BAD SSR

6.2 Block Status Values and Definitions

Table 6-1 lists the Status Values that could be returned when you request Index Number 0 in a dynamic table for a block in your request message. The Status Type and Definition for each is also listed.

Table 6-1 Block Status Values

Floating Point Value	Status Type	Definition
0.0	UNEXECUTED	The block was never executed.
100.0	OK	Normal successful execution.
200.0	FORCED	Output is being forced.
300.0	DIV BY 0	Attempted divide by 0.
400.0	BAD ICNT	The input count passed to the block is greater than the number of inputs for the block.
500.0	BAD TYPE	The block has an illegal control block type assigned to it.
600.0	BAD SSR	Invalid signal source record.
700.0	BAD PV	PV is out of range.
800.0	BAD PGM	Setpoint program is invalid or unused.
900.0	BAD SGM	Setpoint program segment number is invalid or out of range.
1000.0	BAD VPID	Illegal variable parameter index.
1100.0	BAD CVID	Illegal configuration value index.

Block Status Types, *continued*

Floating Point Value	Status Type	Definition
1200.0	BAD BLKNUM	Illegal control block number.
1300.0	NEG SQRT	Attempted square root of a negative number.
1400.0	BAD TUNVAL	Bad tuning constant value.
1500.0	COMM FAIL	cannot communicate with device
1600.0	DEV FAIL	Device reports self test failure
1700.0	BAD DEVID	Invalid device ID
1800.0	BAD BPADDR	Illegal backpan address. This could indicate that the backpan board is either not installed or the wrong board is installed at the address.
1900.0	BAD RECIPE	Illegal recipe number.
2000.0	BAD CHANID	Illegal channel number or module number
2100.0	BAD RANGE	PV HI <= PV LO in PID blocks.
2200.0	TOO MANY	Too many blocks of a restricted type
2300.0	BAD LOOPID	Illegal Loop number
2400.0	UNDERFLOW STACK	During the equation evaluation, an attempt was made to pop an item off the evaluation stack when the stack was empty.
2500.0	OVERFLOW STACK	During equation evaluation, an attempt was made to push an item on the stack when the stack was full.
2600.0	NOT EMPTY STACK	At the conclusion of the equation evaluation, the evaluation stack was not empty.
2700.0	LOG OF NEG NUM	Attempted log of a negative number.
2800.0	UNKNOWN TOKEN	An unknown token was encountered during the evaluation of the equation.
2900.0	POWER ERR	For the power operator (x^y), either x is zero and y is less than or equal to zero, or x is less than zero and y is not an integer.
3000.0	EXP ERR	x is large enough to make e^x overflow.
3100.0	LOG ERR	Attempted log of 0.
3200.0	LOG10 ERR	Attempted log10 of 0.

Appendix A - CRC-16 Calculation

CRC-16 Calculation

See following function:

```
extern void calculate_CRC(unsigned char *message, int length, unsigned char *CRC)
{
unsigned char CRCHi, CRCLo, TempHi, TempLo;

static const unsigned char table[512] = {
0x00, 0x00, 0xC0, 0xC1, 0xC1, 0x81, 0x01, 0x40, 0xC3, 0x01, 0x03, 0xC0, 0x02, 0x80, 0xC2, 0x41,
0xC6, 0x01, 0x06, 0xC0, 0x07, 0x80, 0xC7, 0x41, 0x05, 0x00, 0xC5, 0xC1, 0xC4, 0x81, 0x04, 0x40,
0xCC, 0x01, 0x0C, 0xC0, 0x0D, 0x80, 0xCD, 0x41, 0x0F, 0x00, 0xCF, 0xC1, 0xCE, 0x81, 0x0E, 0x40,
0x0A, 0x00, 0xCA, 0xC1, 0xCB, 0x81, 0x0B, 0x40, 0xC9, 0x01, 0x09, 0xC0, 0x08, 0x80, 0xC8, 0x41,
0xD8, 0x01, 0x18, 0xC0, 0x19, 0x80, 0xD9, 0x41, 0x1B, 0x00, 0xDB, 0xC1, 0xDA, 0x81, 0x1A, 0x40,
0x1E, 0x00, 0xDE, 0xC1, 0xDF, 0x81, 0x1F, 0x40, 0xDD, 0x01, 0x1D, 0xC0, 0x1C, 0x80, 0xDC, 0x41,
0x14, 0x00, 0xD4, 0xC1, 0xD5, 0x81, 0x15, 0x40, 0xD7, 0x01, 0x17, 0xC0, 0x16, 0x80, 0xD6, 0x41,
0xD2, 0x01, 0x12, 0xC0, 0x13, 0x80, 0xD3, 0x41, 0x11, 0x00, 0xD1, 0xC1, 0xD0, 0x81, 0x10, 0x40,
0xF0, 0x01, 0x30, 0xC0, 0x31, 0x80, 0xF1, 0x41, 0x33, 0x00, 0xF3, 0xC1, 0xF2, 0x81, 0x32, 0x40,
0x36, 0x00, 0xF6, 0xC1, 0xF7, 0x81, 0x37, 0x40, 0xF5, 0x01, 0x35, 0xC0, 0x34, 0x80, 0xF4, 0x41,
0x3C, 0x00, 0xFC, 0xC1, 0xFD, 0x81, 0x3D, 0x40, 0xFF, 0x01, 0x3F, 0xC0, 0x3E, 0x80, 0xFE, 0x41,
0xFA, 0x01, 0x3A, 0xC0, 0x3B, 0x80, 0xFB, 0x41, 0x39, 0x00, 0xF9, 0xC1, 0xF8, 0x81, 0x38, 0x40,
0x28, 0x00, 0xE8, 0xC1, 0xE9, 0x81, 0x29, 0x40, 0xEB, 0x01, 0x2B, 0xC0, 0x2A, 0x80, 0xEA, 0x41,
0xEE, 0x01, 0x2E, 0xC0, 0x2F, 0x80, 0xEF, 0x41, 0x2D, 0x00, 0xED, 0xC1, 0xEC, 0x81, 0x2C, 0x40,
0xE4, 0x01, 0x24, 0xC0, 0x25, 0x80, 0xE5, 0x41, 0x27, 0x00, 0xE7, 0xC1, 0xE6, 0x81, 0x26, 0x40,
0x22, 0x00, 0xE2, 0xC1, 0xE3, 0x81, 0x23, 0x40, 0xE1, 0x01, 0x21, 0xC0, 0x20, 0x80, 0xE0, 0x41,
0xA0, 0x01, 0x60, 0xC0, 0x61, 0x80, 0xA1, 0x41, 0x63, 0x00, 0xA3, 0xC1, 0xA2, 0x81, 0x62, 0x40,
0x66, 0x00, 0xA6, 0xC1, 0xA7, 0x81, 0x67, 0x40, 0xA5, 0x01, 0x65, 0xC0, 0x64, 0x80, 0xA4, 0x41,
0x6C, 0x00, 0xAC, 0xC1, 0xAD, 0x81, 0x6D, 0x40, 0xAF, 0x01, 0x6F, 0xC0, 0x6E, 0x80, 0xAE, 0x41,
0xAA, 0x01, 0x6A, 0xC0, 0x6B, 0x80, 0xAB, 0x41, 0x69, 0x00, 0xA9, 0xC1, 0xA8, 0x81, 0x68, 0x40,
0x78, 0x00, 0xB8, 0xC1, 0xB9, 0x81, 0x79, 0x40, 0xBB, 0x01, 0x7B, 0xC0, 0x7A, 0x80, 0xBA, 0x41,
0xBE, 0x01, 0x7E, 0xC0, 0x7F, 0x80, 0xBF, 0x41, 0x7D, 0x00, 0xBD, 0xC1, 0xBC, 0x81, 0x7C, 0x40,
0xB4, 0x01, 0x74, 0xC0, 0x75, 0x80, 0xB5, 0x41, 0x77, 0x00, 0xB7, 0xC1, 0xB6, 0x81, 0x76, 0x40,
0x72, 0x00, 0xB2, 0xC1, 0xB3, 0x81, 0x73, 0x40, 0xB1, 0x01, 0x71, 0xC0, 0x70, 0x80, 0xB0, 0x41,
0x50, 0x00, 0x90, 0xC1, 0x91, 0x81, 0x51, 0x40, 0x93, 0x01, 0x53, 0xC0, 0x52, 0x80, 0x92, 0x41,
0x96, 0x01, 0x56, 0xC0, 0x57, 0x80, 0x97, 0x41, 0x55, 0x00, 0x95, 0xC1, 0x94, 0x81, 0x54, 0x40,
0x9C, 0x01, 0x5C, 0xC0, 0x5D, 0x80, 0x9D, 0x41, 0x5F, 0x00, 0x9F, 0xC1, 0x9E, 0x81, 0x5E, 0x40,
0x5A, 0x00, 0x9A, 0xC1, 0x9B, 0x81, 0x5B, 0x40, 0x99, 0x01, 0x59, 0xC0, 0x58, 0x80, 0x98, 0x41,
0x88, 0x01, 0x48, 0xC0, 0x49, 0x80, 0x89, 0x41, 0x4B, 0x00, 0x8B, 0xC1, 0x8A, 0x81, 0x4A, 0x40,
0x4E, 0x00, 0x8E, 0xC1, 0x8F, 0x81, 0x4F, 0x40, 0x8D, 0x01, 0x4D, 0xC0, 0x4C, 0x80, 0x8C, 0x41,
0x44, 0x00, 0x84, 0xC1, 0x85, 0x81, 0x45, 0x40, 0x87, 0x01, 0x47, 0xC0, 0x46, 0x80, 0x86, 0x41,
0x82, 0x01, 0x42, 0xC0, 0x43, 0x80, 0x83, 0x41, 0x41, 0x00, 0x81, 0xC1, 0x80, 0x81, 0x40, 0x40,
  };
        CRCHi = 0xff;
        CRCLo = 0xff;

        while(length)
        {
                TempHi = CRCHi;
                TempLo = CRCLo;
                CRCHi = table[2 * (*message ^ TempLo)];
                CRCLo = TempHi ^ table[(2 * (*message ^ TempLo)) + 1];
                message++;
                length--;
        };
        CRC [0] = CRCLo;
        CRC [1] = CRCHi;
        return;

}
```

Index

G

H

I,J,K

L

M

N

O

P,Q

R

S

T

U

V

W

X,Y,Z

.

Honeywell

Industrial Automation and Control
Honeywell Inc.
1100 Virginia Drive
Fort Washington, Pennsylvania 19034