GE Fanuc Automation

Programmable Control Products

Series Six™ Bus Controller User's Manual

GFK-0171B ***July, 1991***

WARNINGS, CAUTIONS, AND NOTES AS USED IN THIS PUBLICATION

WARNING

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

CAUTION

Caution notices are used where equipment might be damaged if care is not taken.

NOTE

Notes merely call attention to information that is especially significant to understanding and operating the equipment.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware and software, nor to provide for every possible contingency in connection with installation, operation, and maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

Preface

This manual explains how to set up and install a Series Six™ PLC Bus Controller. It also provides programming information needed to complete the interface between a Series Six PLC and one or more Genius™ I/O communications busses.

Use the *Genius I/O System User's Manual* (GEK-90486) as your primary reference for information about Genius I/O products. It describes types of systems, system planning, installation, and system components.

Contents of this Manual

This book contains 10 chapters and 2 appendixes. For basic information about the functions of a Bus Controller in a Series Six PLC system, begin by reading chapter 1. Lists of topics at the end of chapter 1 will help you find additional information.

Chapter 1. Introduction: Chapter 1 describes Bus Controller types, the serial bus, Bus Controller operation, and system operation.

Chapter 2. Setup and Installation: Chapter 2 explains how to set the on-board and backplane DIP switches when installing a Bus Controller.

Chapter 3. Using Genius I/O with the Series Six PLC: Chapter 3 includes information about assigning Reference Numbers, and describes special programming for analog blocks.

Chapter 4. Automatic Diagnostics and Fault Clearing: Chapter 4 explains how the Expanded Functions of the Series Six Plus PLC can automatically capture, display, and clear faults.

Chapter 5. Programming Window Commands: Chapter 5 explains optional programming to read or write configuration data, read diagnostics, read analog inputs, read a status table address, or switch a BSM.

Chapter 6. Datagrams: Chapter 6 explains optional programming for Datagram and Global Data communications between devices on the bus.

Chapter 7. Global Data: Chapter 7 explains optional programming for Global Data communications.

Chapter 8. Programming for Diagnostics using the Bus Controller Input References: Chapter 8 describes optional programming to obtain diagnostics information from the Bus Controller.

Chapter 9. Programming Commands to I/O Blocks using the Bus Controller Output References: Chapter 9 describes optional programming to clear faults, Pulse Test discrete outputs, or selectively disable outputs.

Chapter 10. Troubleshooting: Chapter 10 lists errors that might occur and suggests corrective actions.

Appendix A. Expanded I/O Addressing: Appendix A shows how Expanded I/O is mapped for CPUs with different amounts of memory.

Appendix B. Bus Controller Compatibility: Compares the features of phase A and phase B Bus Controllers.

Related Publications

For additional information, refer to the following publications:

Logicmaster 6 Software User's Manual (GEK-25379). This book is the primary reference for the Logicmaster 6 programming software. It serves both as a software user's guide and a programmer's guide. The first part of the book describes the operating features of the Logicmaster 6 software, such as

file-handling, display tables, and program printout. The second part of the book defines ladder logic instructions for the Series Six and Series Six Plus PLCs.

Series 90-70 Bus Controller User's Manual (GFK-0398). This manual describes the Series 90-70 Bus Controller, and explains how to interface a Series 90-70 PLC to a Genius bus.

Series Six Bus Controller data sheet (GFK-0025). This data sheet describes operation, installation, and specifications of the Bus Controller.

Genius I/O PCIM User's Manual (GFK-0074). This manual describes the Genius I/O IBM PC Interface Module (PCIM). The PCIM module is used to interface a Workmaster® or CIMSTAR™ I industrial computer or an IBM PC/XT/AT industrial computer to the Genius I/O serial bus.

Series Five™ Bus Controller User's Manual (GFK-0248). This manual explains how to set up a Series Five PLC Bus Controller and how to interface a Series Five PLC to one or more Genius™ I/O busses.

In addition, each Genius I/O product has its own data sheet, which includes basic reference information and installation instructions.

Jeanne Grimsby
Technical Writer

GFK-0171

Chapter 1 Introduction

GFK-0171

The Series Six™ PLC Bus Controller is used to interface a Genius™ I/O serial bus to the Series Six PLC. The bus can serve up to 31 other devices including Genius I/O blocks, Hand-held Monitors, and other interface modules.

The bus can provide I/O service to all types of Genius I/O blocks. It can also be used for programmed communications between the Series Six PLC and other devices. The same PLC system may include several busses, interacting with a wide range of I/O devices, as well as other PLCs and host computers.

Types of Bus Controller

Two types of Bus Controller are available:

- a Bus Controller with Diagnostics (IC660CBB902), which automatically sends diagnostic reports from Genius I/O blocks to the CPU.
- a Bus Controller without Diagnostics (IC660CBB903), which does not. Diagnostics from the blocks are still available to the Hand-held Monitor when this Bus Controller is used.

Both types of Bus Controller send the CPU diagnostic information about their own operation, and the condition of the bus.

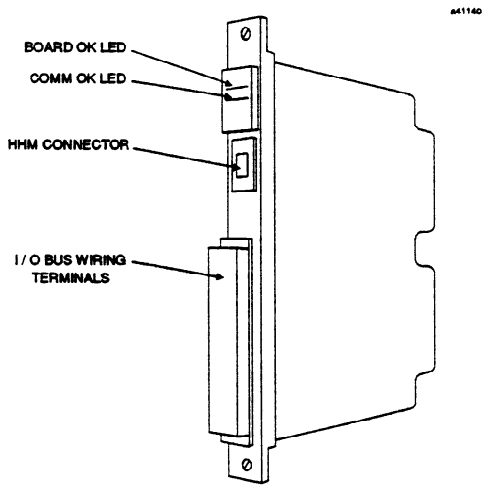
This book describes "phase B" Bus Controllers (IC660CBB902 and CBB903). These enhanced modules are required for:

- A bus which uses any baud rate other than 153.6 Kbaud standard, or which is than 2000 feet.
- A bus that includes RTD blocks or any Genius I/O block that has more than 64 input bits or 64 output bits.
- Programmed communications from one CPU to another.
- Redundancy (more than one bus to the same group of I/O blocks, more than one CPU that is able to communicate with the same group of blocks, or more than one Bus Controller controlling I/O on the same bus).

Using Phase B Bus Controller with Phase A Products

A complete programmable controller system may include both phase A and phase B Bus Controllers (IC660CBB900 and CBB901). Phase B Bus Controllers can be used to replace phase A Bus Controllers, or be added to a PLC system that already includes phase A Bus Controllers. However, only one phase B Bus Controller can be located on a bus with any Phase A products of any kind. For more information about differences between Phase A and Phase B Bus Controllers, see appendix A.

The Bus Controller is a standard, rack-mounted Series Six PLC module.



A Bus Controller may be located in the CPU rack or in a regular or High-capacity local I/O rack (that is, a rack communicating with the CPU via a parallel chain). Each Bus Controller draws 20 units of load (one unit = 300mW) at +5 volts. A phase B Bus Controller does not use +12 volts.

The only limit on the number of Bus Controllers used in a PLC system is the I/O addressing capacity of the CPU.

LEDs

The Bus Controller has two LEDs:

- **BOARD OK** shows the status of the Bus Controller.
- **COMM OK** shows the status of the bus.

Hand-held Monitor Connector

The Hand-held Monitor connector on the Bus Controller faceplate provides access to all devices on the bus. All Hand-held Monitor functions except I/O block configuration can be performed with the HHM connected to the Bus Controller. Bus and block operation can be monitored, circuits forced or unforced, outputs Pulse Tested, diagnostic messages displayed, and faults cleared, from a convenient central location.

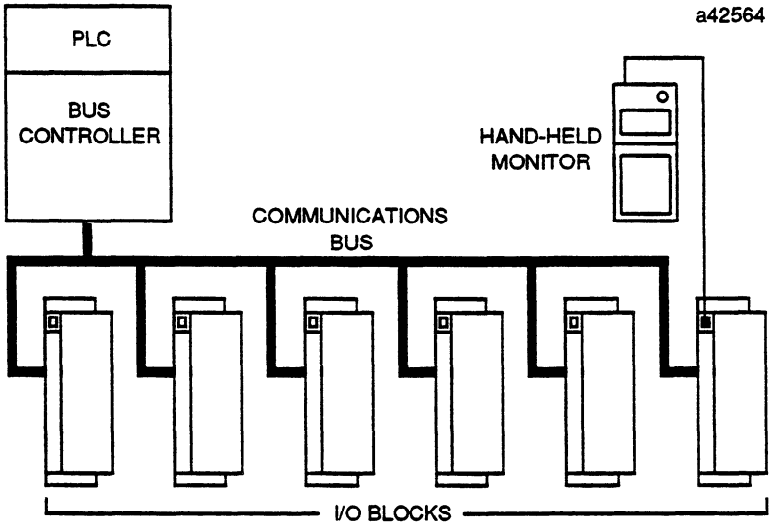
Bus Wiring Terminals

The upper four terminals on the terminal strip are used for the serial bus and shield wiring connections. The lowest connector jumpers Shield Out to chassis ground. The remaining connectors are not used.

GFK-0171

The Bus

A shielded, twisted pair cable connects the Bus Controller to up to 31 other devices. These may be Genius I/O blocks, Hand-held Monitors, or other interface modules (Bus Controllers or PCIMs).



A bus can serve all types of Genius I/O blocks. Phase A Genius I/O blocks can be located on a bus that meets the restrictions for use with phase A blocks. In a basic I/O control system, one bus may serve up to 30 Genius I/O blocks and one Hand-held Monitor. The total number of I/O circuits that can be included on the bus depends on the types of I/O blocks that are used. Genius discrete I/O blocks are available in several different types, with 8 to 32 circuits each. The maximum number of I/O circuits possible on one bus would result from using 30 discrete blocks with 32 I/O circuits each. The total number of discrete circuits would be 960. For applications requiring very fast response times, fewer blocks might be located on a bus, with additional blocks distributed on other busses.

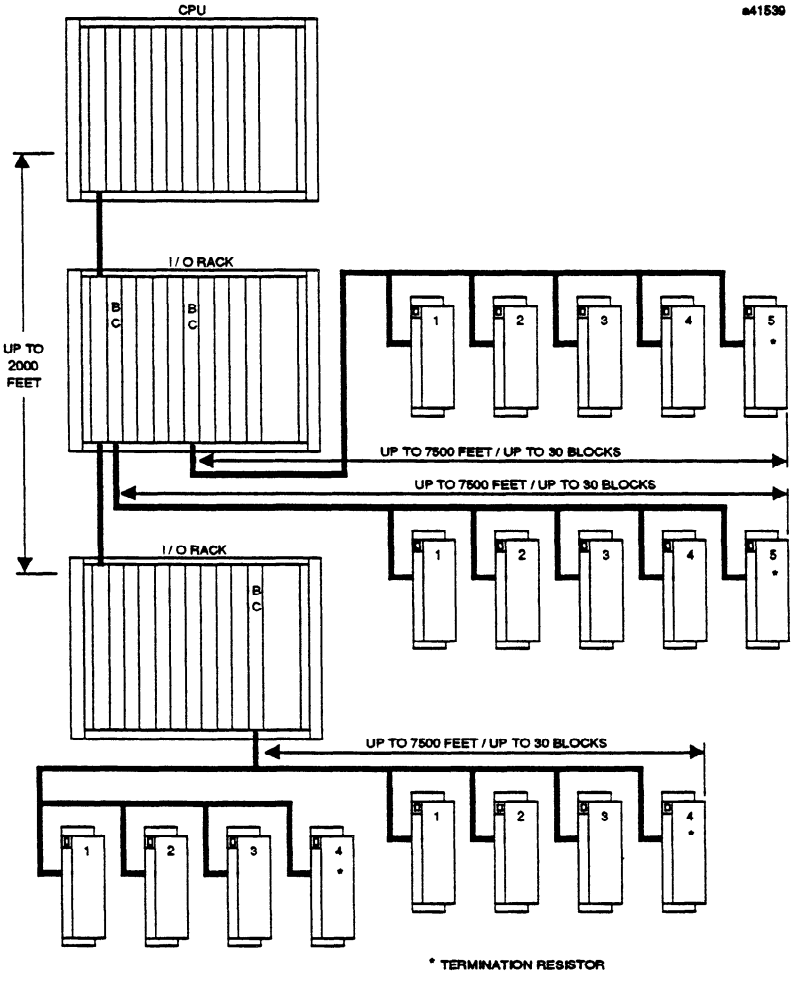
Analog blocks (including the low-level analog RTD blocks) have 6 circuits each. Therefore, the maximum number of analog circuits on one bus is 6 circuits times 30 blocks, or 180. Because of the length of time required to transmit analog data, special programming (see chapter 3) will be required if more than 5 analog blocks are used on the same bus.

Number of Busses in the System

The only limit to the number of busses in the system is the I/O addressing capacity of the Series Six CPU. Different busses in the system can be different lengths, use different cable types, have different I/O response times, and operate at different baud rates. Each bus in the system is essentially independent from the others. Care must be taken to ensure that individual busses do not interfere with I/O references assigned to other Genius Bus Controllers, I/O blocks, or conventional I/O modules in the system.

Bus Controller Location

A Bus Controller can be placed in the CPU rack or in a local I/O rack up to 2000 feet from the CPU. Multiple Bus Controllers can be placed in one rack. Since each Bus Controller consumes 20 units of I/O power, as many as six Bus Controllers can be placed in a Series Six Plus CPU rack, or ten in a High Capacity I/O rack.



Using More than One Bus Controller on an I/O Chain or Channel

Within each I/O chain or channel, there can be more than one bus and Bus Controller, as long as references assigned to the blocks on different busses do not overlap.

Using I/O Transmitter Modules and Auxiliary I/O Modules

Depending on the I/O addressing used for the system, the Bus Controller may need to be located downstream of an Auxiliary I/O Module, and/or an I/O Transmitter Module, as explained below.

Locating the Bus Controller Downstream of an Auxiliary I/O Module

If the I/O points on a bus will be given program references in the auxiliary I/O table or in "Expanded" I/O channels 9-F, the Bus Controller must be installed downstream of an Auxiliary I/O module.

Locating the Bus Controller Downstream of an I/O Transmitter Module

If the PLC system uses Expanded ("channelized") I/O addressing, channel selection can be made by either the Bus Controller (phase B only) or an I/O Transmitter module. If an I/O channel has Genius I/O only, the Bus Controller can select channelization and no upstream I/O Transmitter module will be permitted.

A Bus Controller set up for channelization must not be located downstream of an I/O Transmitter Module. It is possible to have Genius I/O blocks controlled by a Bus Controller set up for channelization, while conventional I/O modules are also assigned to the same channel. It is important to be sure that the reference numbers assigned to the Genius blocks and I/O modules do not overlap. It is also possible to have several channelized Bus Controllers assigned to the same channel, as long as overlapping references are avoided.

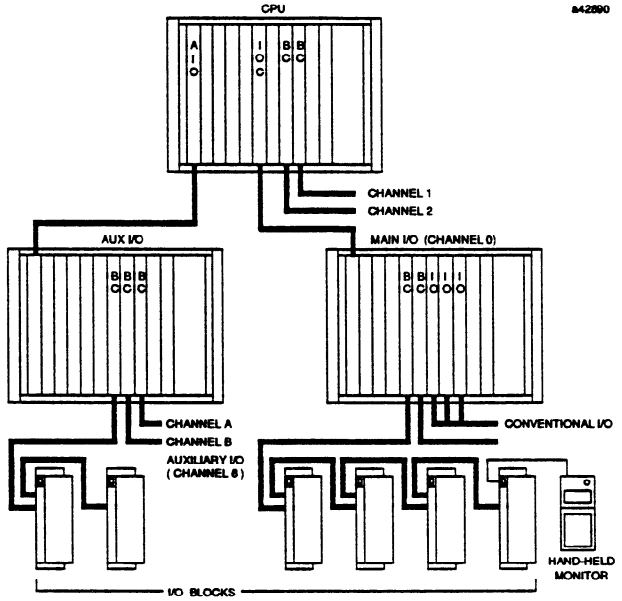
Ideally, if a system includes both conventional I/O and Genius I/O blocks, they should not be mixed on the same channel. However, if the two I/O types must share a channel, restricting each to a predetermined part of that channel (for example, one half), will help prevent reference conflicts.

Channel	Bus Stream Controller Channelized	Upstream Main I/O Transmitter	Aux I/O Transmitter
9-F	no	no	yes **
0-8	no	yes*	no **
or			
0-8	yes	no	no ***

* accommodates conventional I/O in the channel, as well as Genius I/O modules.

** phase A or phase B Bus Controller

*** phase B Bus Controller only



Locating Bus Controllers in Remote I/O Racks

A Bus Controller may also be placed in a Remote I/O rack, but this is not recommended. If the application requires great distance between the CPU and I/O blocks, it is better to use a longer bus between a Bus Controller and the I/O blocks than to place the Bus Controller in a Remote I/O rack.

The cable from the Bus Controller to its I/O blocks can be up to 7500 feet long. At maximum length, the Bus Controller supports up to 15 I/O blocks, providing up to 480 discrete references on the bus. In addition, the Bus Controller can be placed in a high-capacity Local I/O rack up to 2000 feet from the CPU, giving a total maximum distance from the CPU to the end of the bus of 9500 feet. Such a system provides all of the diagnostics and communications features of Genius I/O.

Placing a Bus Controller in a Remote I/O rack severely limits its capabilities. The Remote I/O rack must be hardwired from the Remote I/O Transmitter module to the Remote I/O Receiver module (a modem cannot be used). Communications between the Remote I/O Transmitter and Remote I/O Receiver modules are via a serial twisted pair link of two unidirectional data lines (Transmit Data and Receive Data). This method of I/O update is slower than normal Genius I/O service. Up to 496 total I/O references can be included in one Remote I/O station. Because of the reduced capacity of the remote I/O communications link, this type of system does not support the use of any communications commands, and is not recommended for fast-acting I/O. In addition, diagnostics may not always reach the CPU. Using Remote I/O also requires more complex logic in the application program.

Operation of Genius I/O in a Remote I/O System

With a Bus Controller in a Remote I/O rack, communications between the Remote I/O Transmitter and Remote I/O Receiver modules must be set at 57.6 Kbaud. The I/O will be serviced each time the upstream Remote I/O Receiver module performs a partial Series Six I/O sweep. The update rate is affected by asynchronism between the Series Six I/O sweep and the transmission time required for data on the serial link between the Remote I/O Transmitter and Remote I/O Receiver modules. The Remote I/O Receiver module scans the Bus Controller for current inputs from the Genius I/O blocks. It then transmits input data and the Bus Controller status to the Remote I/O Transmitter module via the serial link until the next programmer window occurs. On the next CPU I/O sweep, these same inputs are read from the Remote I/O Transmitter module into the Series Six Input Status Table.

If an output changes in response to an input or status bit, the earliest the new output will get from the CPU to the Bus Controller is 11mS after the next programmer window. See GEK-83537, *Remote I/O Modules*, for a discussion of system delay times.

New output states on a given I/O sweep are not guaranteed to be picked up by the Remote I/O Transmitter for transmission, so care must be taken to ensure that output states of short duration actually get to the output. The Remote I/O Transmitter provides a handshake bit to assist in this. Since faults that are reported for one scan may be missed by the CPU, the application program should periodically issue a Clear All Faults command to the Bus Controller.

Bus Controller Operation

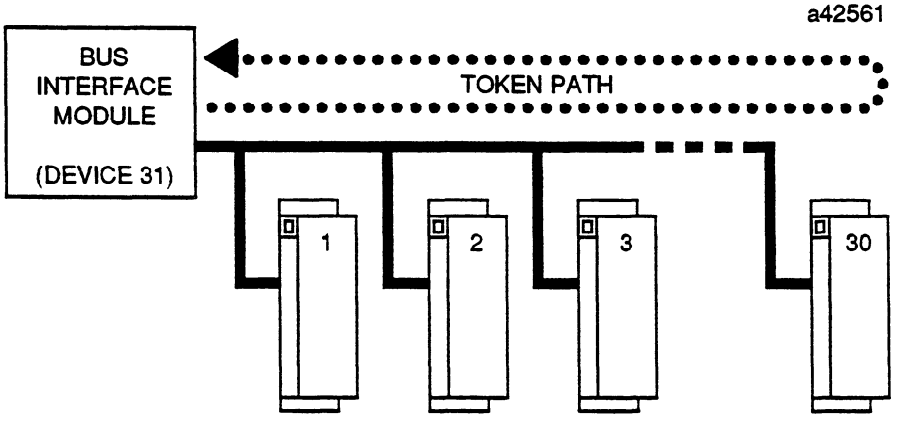
All data transfer between the PLC and the devices on a bus is handled by the Bus Controller. The Bus Controller interfaces two completely separate and asynchronous activities:

- the Genius bus scan, a cycle of communications between the devices on a bus (this includes the Bus Controller itself).
- the CPU sweep, the cycle of actions that includes communications between the CPU and the Bus Controller.

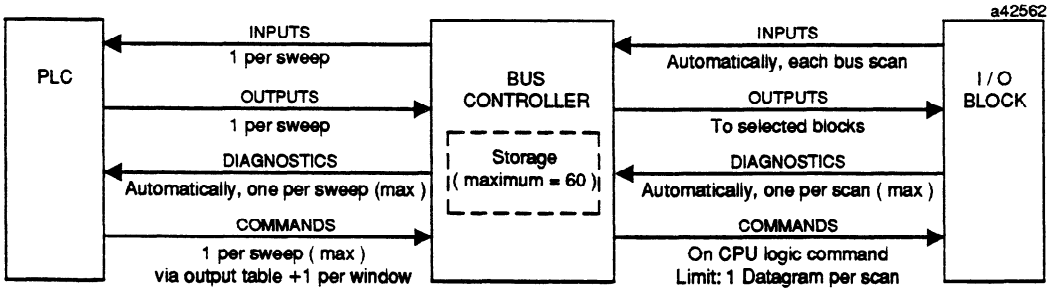
The Bus Controller manages data transfer between these two asynchronous cycles by maintaining two separate on-board RAM memories. One interfaces with the bus and the other (referred to as "shared RAM") interfaces with the CPU. The Bus Controller automatically transfers data between these two memories, making data available to the bus or to the CPU when it is needed.

The Genius Bus Scan

A Genius bus scan consists of one complete rotation of a "token" among the devices on the bus.



During a bus scan, the Bus Controller automatically:



- Receives and stores all inputs from the I/O blocks on the bus.
- Updates outputs, as permitted, on the I/O blocks. With a phase B Bus Controller, transmission of outputs from the CPU can be disabled for one or more blocks on the bus. For all systems, this feature can be used to initialize outputs at startup. For advanced systems with distributed control of outputs, or computer monitoring of I/O block data, outputs can be disabled to individual blocks. Both phase A and phase B Bus Controllers can disable output updates as described in chapter 5.
- A Bus Controller with Diagnostics receives any fault messages issued by devices on the bus, and stores up to 60 in an on-board fault table. The Series Six Plus PLC can be set up to provide its own internal fault table for storage of diagnostic messages, and can access these diagnostic messages automatically. In addition, ladder logic can be added to the program of any Series Six CPU to access diagnostic messages.
- Sends any command received from the CPU (for example, Clear Circuit Fault) to the appropriate device.

Bus Scan Time

The amount of time required for one complete rotation of the token to all devices will depend on the baud rate, the number and types of blocks, the number of bus interface modules (Bus Controllers and PCIMs), and the occurrence of optional Global Data and of programmed messages on that bus.

Bus scan time directly affects:

- the response time for servicing I/O on the bus.
- the time required for programmed communications to be transmitted on the bus.
- the relationship between the bus scan and the CPU sweep time. Because CPU sweep time is probably less flexible, it may be necessary to change I/O block distribution or programmed communications to create a balance.

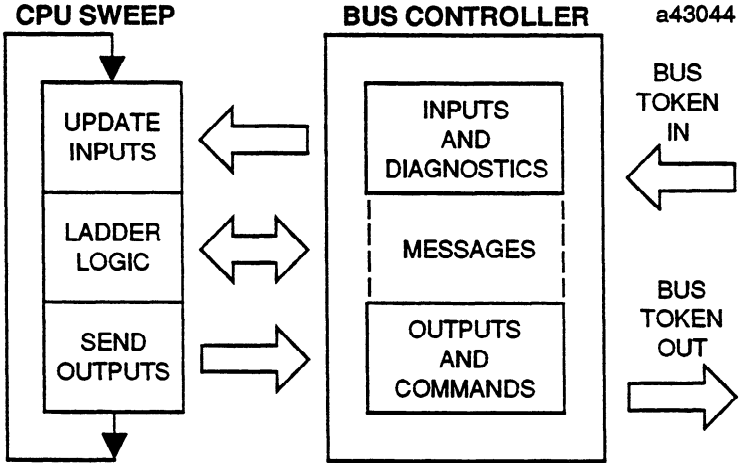
The Genius I/O System User's Manual explains bus scan time in detail.

The CPU Sweep

The CPU sweep is the PLC's regular cycle of operations. During each sweep, the CPU updates inputs and outputs, executes the application program, and communicates with other devices. The CPU sweep is executed independently of the Genius bus scan.

During one CPU sweep, the Bus Controller:

- Transfers all discrete inputs and one channel input value (16 bits) per analog (or RTD) block from shared RAM memory to the CPU Input Status Table. *To update all the inputs on an analog block during a single CPU sweep, the command Read Analog Inputs can be added to the ladder logic program.*
- Receives current outputs and new commands from the CPU Output Status Table and places them in shared RAM.
- Reports its status and that of the serial bus. A Bus Controller with Diagnostics also reports the status of the I/O blocks and provides one new diagnostic (if any) to the CPU. Diagnostics data is moved to the Input Table, starting at the address set with the backplane DIP switches.
- May communicate with the CPU in response to window instructions in the application program. If the program directs DPREQ or WINDOW instructions to a Bus Controller, a communications window to that Bus Controller opens during the ladder logic portion of the sweep. The Bus Controller may also receive messages from other devices on the bus. These messages will be returned to the CPU when the program opens a communications window to the Bus Controller.



Completing the Interface

The rest of this book explains how to install a Bus Controller and describes programmable features. The tables on the following pages will help you locate the information you need for:

- Setting up a Bus Controller.
- Setting up the PLC system.
- Programming for analog I/O.
- Programming for diagnostics and fault clearing.
- Programming for communications between CPUs.

Each topic is explained in more detail later in the book.

Setting up a Bus Controller

Each Bus Controller in the system, and the I/O blocks on the bus, must be configured.

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Prevent program from sending unwanted outputs to blocks following powerup of the Bus Controller.	Set DIP switch 4 at position U16.	2
Indicate whether failure of the Bus Controller's self test should stop the CPU.	Use switch 1 at position U3.	2
Match the Bus Controller's baud rate to the other devices on the bus.	Use switches 2 and 3 at position U3.	2
Select a Device Number for the Bus Controller.	Use switches 4 through 8 at position U3.	2
Set up Bus Controller for Expanded I/O scanning, and select channel.	Use switches 1-3 at position U59.	2
Select terminating impedance for the Bus Controller	For single bus cable, use jumper on Bus Controller. For redundant cable, use resistor.	2
Select the Bus Controller reference number.	Set backplane DIP switches	2
Automatic I/O update.	Assign I/O (not register) reference numbers to I/O blocks.	3

Setting up the PLC System

The Logicmaster 6 programming software (LM6) is used to set up Expanded I/O scanning, and specify the use of automatic Genius I/O diagnostics for the entire PLC system.

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Enable/disable Expanded functions.	Enter CPU Configuration instruction as first rung in ladder.	4
Enable/disable automatic diagnostics.	Use LM6 Expanded Functions menu.	4
Enable/disable Expanded I/O scan.	Use LM6 Expanded Functions menu.	4
Specify channel pairs for I/O scanning.	Use LM6 Expanded Functions menu.	4
Specify channel pairs for diagnostics.	Use LM6 Expanded Functions menu.	4
Select length of the fault table for automatic diagnostics. Default is 8 uncleared faults for system.	Use LM6 Expanded Functions menu.	4
Specify locations of Bus Controllers that will report diagnostics.	Make selection on LM6 Expanded Functions menu.	4
Create internal status table for all I/O points with diagnostics.	Select Y for "Diagnostic Tables" on LM6 Expanded Functions menu.	4
	Assign Bus Controller on 256 bit reference boundaries.	2

Programming for Analog I/O

If there will be analog I/O blocks on a bus, logic must be added to the program to store input values for each analog block. If there are more than 5 analog blocks on the bus, additional logic must be used to assure that all inputs are received by the CPU.

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Copy multiplexed analog inputs from block's input references.	Use program logic to read one input per sweep into registers.	3
Read all inputs from an analog block during the same CPU sweep.	Program Read Analog Inputs command using DPREQ or WINDOW instruction.	5
Store inputs from more than 5 analog blocks on the same bus.	Program Read Analog Inputs command using DPREQ or WINDOW instruction.	5

Additional Programming Capabilities

The program may also include logic to do the following:

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Pulse Test discrete outputs from program.	Set Bus Controller output reference bit 4.	9
Read or write configuration data for the Bus Controller or a block.	Use DPREQ or WINDOW instruction with a Read or Write Configuration command.	5
Read the I/O type assignment of any point on the bus.	Use DPREQ or WINDOW instruction with a Read Configuration command to the Bus Controller.	5
Determine whether outputs are disabled to a device on the bus.	Use DPREQ or WINDOW instruction with a Read Configuration command to the Bus Controller.	5
Determine error rate on bus, or bus scan time.	Use DPREQ or WINDOW instruction with a Read Diagnostics command to the Bus Controller.	5
Determine the current fault status of the Bus Controller or a block.	Use DPREQ or WINDOW instruction with a Read Diagnostics command.	5
Read the Status Table reference of the Bus Controller or a block.	Use DPREQ or WINDOW instruction with Read Status Table command.	5

Programming for Diagnostics and Fault Clearing

Program logic can be used in addition to, or instead of, automatic diagnostics to access specific types of information and to send commands to I/O blocks on the bus.

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Detect failure of Bus Controller self-test.	Monitor Bus Controller input reference bit 1.	8
Detect bus error condition.	Monitor Bus Controller input reference bit 2.	8
Detect I/O circuit faults.	Monitor Bus Controller input reference bit 3.	8
Detect addition or loss of block.	Monitor Bus Controller input reference bits 4 and 5.	8
Detect duplicate or overlapping reference numbers.	Monitor Bus Controller input 5 reference bit 6.	8
Detect execution of a Pulse Test.	Monitor Bus Controller input reference bit 7.	8
Detect force condition on the bus.	Monitor Bus Controller input reference bit 8.	8
Create register table to store faults.	See example logic in chapter 5.	8
Clear all faults on a bus from program.	Set Bus Controller output reference bit 2.	9
Clear fault(s) on a specific circuit from the program.	Set Bus Controller output reference bit 3, use additional references to specify circuit.	9

Programming for a Bus with Redundant Interface Modules

If the bus will interface additional Bus Controller or PCIM modules, program logic may be needed for CPU redundancy, distributed control, or an assigned monitor:

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Prevent the Bus Controller from sending outputs to one or more devices.	Program Write Configuration command to Bus Controller using DPREQ or WINDOW instruction.	5
Disable all outputs from a Bus Controller used as a monitor.	Set Bus Controller output reference bit 1.	9
Set up blocks to issue extra fault reports and configuration change messages to an assigned monitor.	Use DPREQ or WINDOW instruction Assign Monitor Datagram.	5
Switch a Bus Switching Module to a specified bus.	Use DPREQ or WINDOW instruction with Switch BSM command.	5

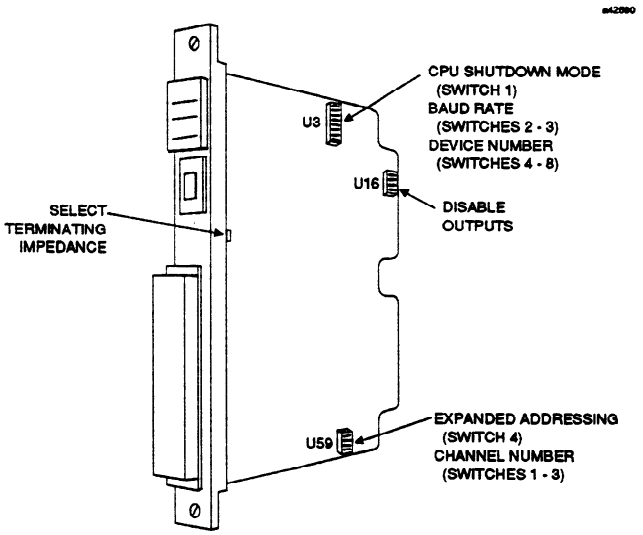
Programming for Communication between CPUs:

If the Series Six PLC will send or receive Datagram or Global Data messages on a bus with other Bus Controllers or PCIMs, additional program logic will be needed.

FEATURE	HOW TO DO IT	DESCRIBED IN CHAPTER
Read or write Global Data address, Global Data length.	Use DPREQ or WINDOW instruction with a Read or Write Configuration command to the Bus Controller.	5. 7
Read the Global Data address of another interface module on the bus.	Use DPREQ or WINDOW instruction with Read Status Table command.	5
Handling Global Data: transfer up to 128 bytes of data between CPUs using Global Data.	Use DPREQ or WINDOW instruction to send Write Configuration command to Bus Controller.	5, 7
	Use Idle DPREQ or WINDOW instruction to transfer Global Data to/from the Series Six CPU.	7
Send up to 128 bytes of data to a Bus Controller or PCIM as a Datagram.	Use DPREQ or WINDOW instruction to send Write Device Datagram.	6
Read up to 128 bytes of data from another CPU on the bus as a Datagram.	Use DPREQ or WINDOW instruction to send Read Device Datagram. In the other CPU, use logic to open communication between the CPU and its bus interface module.	6
Set or clear up to 16 individual data bits in another CPU.	Use DPREQ or WINDOW instruction to send Write Point Datagram.	6

Chapter 2
Setup and Installation

Before a Bus Controller is installed, its on-board and backplane DIP switches can be set to select the features described below. The board is shipped from the factory with default settings. A discussion of each setting follows.



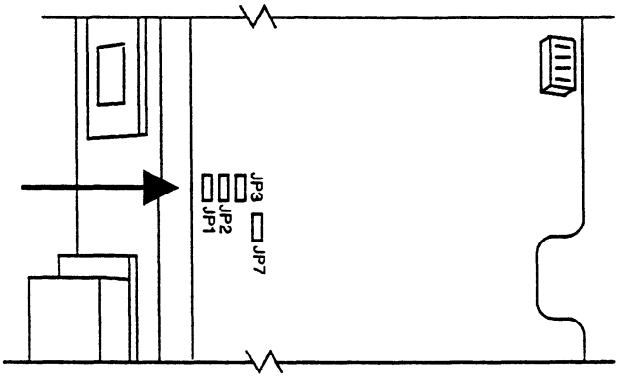
- **Terminating impedance for the communications bus.** If the Bus Controller is not at the end of the bus, do not change the default switch setting.
- **CPU shutdown mode.** If program execution should continue in the event of Bus Controller failure, do not change this.
- **Baud rate.** The default setting is 153.6 Kbaud. The Bus Controller's baud rate must match that of the other devices on the bus. Selection of a baud rate is based on cable length and type, and the use of phase A devices on the bus.
- **Device Number.** The default Device Number is 31. Unless there is another bus interface module on the same bus, this number need not be changed.
- **Expanded I/O addressing.** If the PLC system uses Normal I/O addressing, or if the Bus Controller will be installed downstream of an I/O Transmitter module that will select Expanded I/O addressing, do not change the default setting.
- **Outputs disabled or enabled when Bus Controller powers up.** By default, outputs are automatically transmitted to the blocks when the Bus Controller powers up with the PLC in the Run/Enable mode. This can be changed to disable outputs at powerup. The program can subsequently enable these outputs.

Refer to the descriptions that follow to determine whether the defaults should be changed. A sample configuration worksheet form is provided at the end of this chapter. A copy of this worksheet can be used to record the configuration of the Bus Controller.

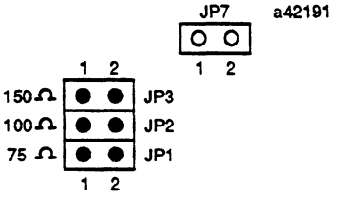
In addition to the on-board switch settings, a Reference Number for the Bus Controller is selected using DIP switches on the rack backplane.

Selecting Terminating Impedance

Each Genius communications bus must be terminated at both ends by its characteristic impedance. A jumper on the Bus Controller is used to select impedance.



The jumper across JP7 selects no impedance.



This position is correct if the Bus Controller is not physically installed at the end of the bus. If the Bus Controller is at either end of the bus, move the jumper to select 75Ω, 100Ω, or 150Ω terminating impedance. The *Genius I/O System User's Manual* explains correct terminating impedance and baud rate for different cable types and lengths.

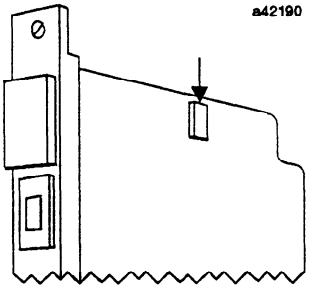
Terminating Impedance on a Redundant Bus

A bus that uses two Bus Controllers for redundancy on the same bus requires special planning for termination. If either Bus Controller will be located at the end of such a redundant bus, do not select terminating impedance using the on-board jumper. Instead, install a resistor of correct impedance across the Serial 1 and Serial 2 terminals on the Bus Controller's faceplate. This will make it possible to keep the bus properly terminated and in operation should removal of the Bus Controller ever be necessary.

GFK-0171

SWITCH	SETTING
CPU Shutdown 1	Continue CPU Sweep: x Stop CPU Sweep: -
Baud Rate	153.6 standard: 76.8 / 38.4 / 153.6 extended
2	x / x / - / -
3	x / - / x / -
Device Number	11111111112222222233 01234567890123456789012345678901
4	----------xxxxxxxxxxxxxxxx
5	--------xxxxxxxx--------xxxxxxxx
6	----xxxx----xxxx----xxxx----xxxx
7	--xx--xx--xx--xx--xx--xx--xx--xx
8	-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x

x = open/off. Default is all open.



Enabling CPU Shutdown Mode

Switch 1 of the DIP switch pack at position U3 determines whether Bus Controller failure will stop the CPU sweep. The default selection is for the CPU sweep to continue. If the CPU sweep should stop because one of the Bus Controllers has failed its self-test, then the program must be able to detect the failure. If the Expanded Functions are enabled (this requires Logicmaster 6 software release 3.0 or later), the CPU will continually check for Bus Controller failure. If an earlier software version is used, the ladder logic program must check for a Bus Controller failure by resetting the Bus Controller OK Status Bit every CPU sweep. For more information about this status bit, see chapter 5.

Setting the Baud Rate

All devices on a bus (including the Bus Controller and the Hand-held Monitor) must be set up to use the same baud rate. (Other busses connected to the PLC may operate at different baud rates, however). The default baud rate selected for a new Bus Controller is 153.6 Kbaud (standard). This default is provided so that the Bus Controller is compatible with phase A devices. However, 153.6 Kbaud extended will provide better performance. Depending on the length of the bus and other factors, it may be desirable or necessary to change this to 153.6 Kbaud extended, 76.8 Kbaud. or 38.4 Kbaud using switches 2 and 3 at position U3. The *Genius I/O System User's Manual* gives guidelines for baud rate selection.

Changing the Bus Controller Device Number

Each device on the bus must have a bus address, which is referred to as its Device Number. For a new Bus Controller, the Device Number 31 is selected by default. *If there is just one Bus Controller on the bus, no change is needed.* If there will be more than one bus interface module (Bus Controller or PCIM) on the *same*bus, it is necessary to assign each a different Device Number. Switches 4 through 8 at position U3 select a Device Number. A Bus Controller may use any available Device Number except in CPU Redundancy mode. In Redundancy mode, the blocks require Bus Controllers at Device Numbers 30 and 31.

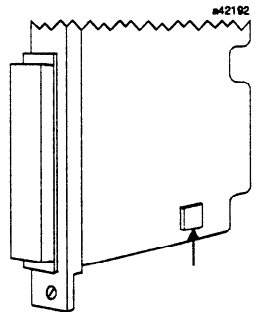
Selecting Expanded I/O Addressing

The DIP switches at position U59 select the type of I/O addressing used by the Bus Controller. Do not change the default switch settings if the system uses Normal I/O addressing, or if the Bus Controller will be installed downstream of an I/O Transmitter module which will be selecting an Expanded I/O channel.

If the system uses Expanded I/O addressing with the channel selected by the Bus Controller, use the switches at position U59 to select Expanded I/O addressing and specify the channel number.

SWITCH	SETTING	
	0 1 2 3 4 5 6 7	Aux 9 A B C D E F
1	- x - x - x - x	- x - x - x - x
2	- - x x - - x x	- - x x - - x x
3	- - - - x x x x	- - - - x x x x
4	set 4 to open for Expanded I/O	

x = open/off. Default is all closed.



The Bus Controller must be installed downstream of an Auxiliary I/O Module to address the Auxiliary I/O chain or Expanded I/O channels 9 through F.

A phase A Bus Controller cannot be used for channel selection, and must be installed downstream of an I/O Transmitter module in an Expanded I/O system.

Logicmaster 6 Version Number

It is important to know which version of Logicmaster 6 software is being used, because that determines which Expanded I/O channels the CPU scans by default.

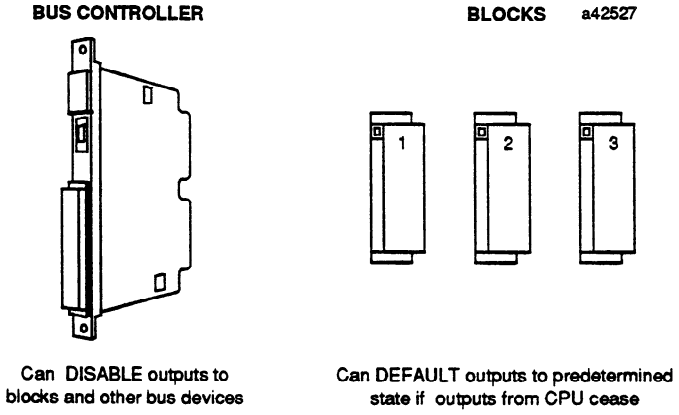
Logicmaster 6 software version 4 and later defaults to scanning channels 0 and 8, which correspond to the Main and Auxiliary I/O chains. For Logicmaster 6 version 3.0, the software defaults to the first 4 channels enabled (channels 0-3).

The setup of the Bus Controllers must match the channels scanned by the CPU. As an example, suppose the system has one Bus Controller and that it has been installed without changing its DIP switch settings. That means it is not set up for Expanded addressing. Subsequently, the system is powered up using Logicmaster 6 software version 3. This software defaults to Expanded I/O Enabled, and scans channels 0-3. Because the Bus Controller has not selected a particular channel, it receives outputs on successive CPU sweeps as though they were intended for Bus Controllers set up for channels 0, 1, 2, and 3. That means it receives constantly-changing output data from the CPU. To avoid this problem, the application program and the Bus Controller setup must match.

Disabling Outputs

The Bus Controller can be used to disable outputs from the CPU to the blocks.

Disabling outputs means preventing the transmission of output information to the block(s) from the CPU (this is determined at the Bus Controller). Disabling outputs is not the same as defaulting outputs. *Defaulting* outputs means automatically setting outputs to a predetermined state if the block ceases to receive CPU communications for 3 bus scans. When to default outputs is determined by the I/O blocks. If program action causes the CPU to disable outputs while the system is in operation, any blocks that stop receiving outputs as a result will default their outputs to the appropriate predetermined state. The blocks will also default outputs if CPU communications are lost for any other reason, such as a broken cable.



How Outputs can be Enabled or Disabled

There are three different ways to disable (or enable) outputs:

First: All outputs on a bus can be disabled (or enabled) at powerup by setting a DIP switch at position U16. This is explained on the next page.

Second: All outputs from the Bus Controller can be disabled and enabled together during system operation. This might be done in a system with two or more CPUs on the same bus, where one was used only for data monitoring.

Third: Outputs can be enabled or disabled during system operation on a block-by-block basis by setting and clearing individual Disable Outputs bits. This might be done in a system where two or more CPUs were used for selected control of blocks on the same bus.

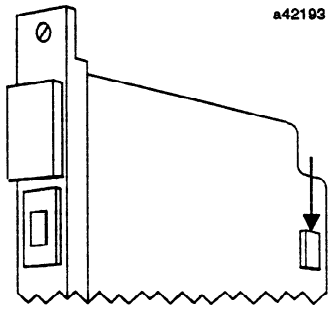
Disabling Outputs at Powerup

Under certain conditions, it is possible that the Bus Controller may begin transmitting output data before the CPU program has set up the desired output values. Setting a Bus Controller DIP switch (see below) prevents possibly incorrect operation of outputs at powerup. This requires the use of program logic to enable the outputs after correct system operation is assured. To do this:

1. Set Bus Controller DIP switch 4 at position U16 to disable outputs when the Bus Controller is powered up.

SWITCH	SETTING	
	Disable Outputs	Enable_Outputs
1 - 3	not used	
4	x	-

x = open/off



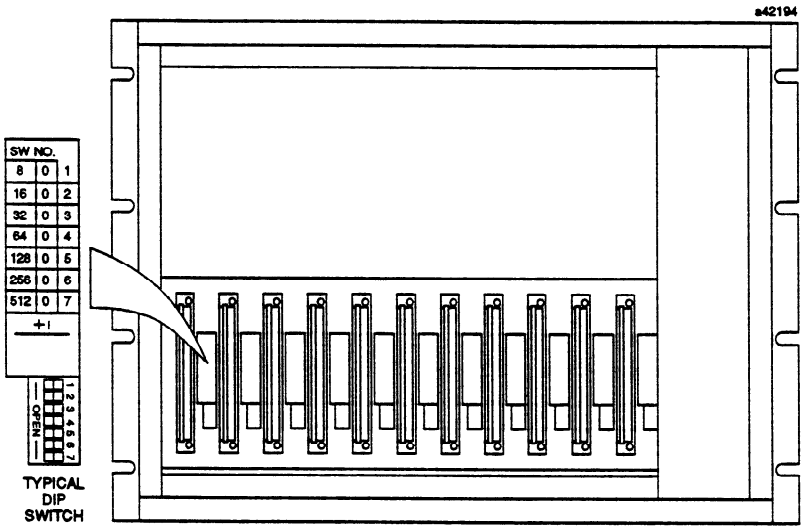
2. At a subsequent time, determined as suitable by the application, enable outputs to some or all of the blocks under control of the program.

If switch 4 is open, the Bus Controller will hold off output transmissions until it completes its powerup sequence. If the PLC is stopped because the Bus Controller has powered down (which is typical), it must execute two sweeps in Run Disabled mode. While the PLC is in Run Disabled mode, the Bus Controller will continue to withhold output transmissions. These two sweeps typically permit the Bus Controller to be updated with correct outputs for the blocks. However, this does depend on the relative lengths of the PLC sweeps and the Genius bus scans. Switch 4 only determines what happens after the Bus Controller is powered up. It has no effect if the Bus Controller is in a rack that is powered up and the PLC is powered up later.

Thus, if switch 4 is closed, the Bus Controller will update outputs as soon as its powerup sequence is completed, or as soon as the CPU goes into Run/Enabled mode, whichever is later. With switch 4 open, the onset of output updating is determined by program logic in the PLC.

Bus Controller Reference Number

Each Bus Controller in the Series Six PLC system must be assigned a Reference Number by setting the DIP switches on the rack backplane.



The Reference Number assigned to each Bus Controller is a beginning address in the I/O Table where the Bus Controller's input and output data will be located. This data includes diagnostics and command information.

Be sure each Bus Controller on an I/O chain has a unique Reference Number. Also be sure no Bus Controller on the Main I/O Chain has the same Reference Number as a Bus Controller on the Auxiliary I/O Chain. References used by Bus Controllers must not overlap with those of any Genius I/O blocks or of any conventional I/O modules.

References Required (8): Bus Controller without Diagnostics

Assign Reference Numbers for a Bus Controller without Diagnostics on a byte boundary (9, 17, 25 ...). This type uses 8 references (both inputs and outputs).

References Required (48): Bus Controller with Diagnostics

Reference Numbers for Bus Controllers with Diagnostics must be assigned on 64-point boundaries (unless Diagnostic Tables are enabled as explained below). Because the Bus Controller uses only the first 48 references, the final 16 references in each 64-reference block can be assigned for general I/O use.

For some Series Six Plus PLC applications, register memory can be used to store the current diagnostic status of all I/O points on the bus. If DIAGNOSTIC TABLES is selected on the Logicmaster 6 software CPU Configuration Setup Menu, Bus Controller Reference Numbers must be assigned on 256-reference (not 64-reference) boundaries. The Bus Controller will use the first 48 references for diagnostics, and will place the current diagnostic status of all I/O points on the bus in the remaining 208 references.

Bus Controller
Configuration Worksheet

Bus Controller Type:

Bus Controller with diagnostics (IC660CBB902) ____.
Bus Controller without diagnostics (IC660CBB903) ____.

Description/General Information:

Location to install this board: _____.

Before installing the board, configure it using the DIP switches and jumpers.

Position U3:

Switch 1 = CPU Shutdown Mode: continue (default)/ stop ____.

Switches 2 and 3 = Baud Rate: 153.6 Kb standard (default), 153.6 Kb extended, 76.8 Kb, 38.4 Kb ____.

Switches 4 through 8 = Device Number: 0-31 (default is 31) ____.

Position 59:

Switches 1 through 3 = I/O channel:

Main I/O Chain: channel 0 (default) through 7 ____.

Auxiliary I/O Chain (must be downstream of Auxiliary I/O Module):
Aux I/0 table (default) through channel F ____.

Switch 4 = Expanded I/O Addressing: no (default)/yes ____.
If yes, enter channel number ____.

Position U16:

Switch 4 = Disable Outputs at powerup: yes (default)/no ____.
Switches 1 through 3 must be OPEN

Select the terminating impedance using the on-board jumper:

JP7 = none (default)
JP1 = 75Ω ____.
JP2 = 100Ω ____.
JP3 = 150Ω ____.

Select the Reference Number using the backplane DIP switches:

Reference Number (1 - 993) ____

Chapter 3

3-1 Interfacing Genius I/O Blocks to the Series Six PLC

GFK-0171

Each Genius I/O block in a Series Six PLC system must be assigned a Reference Number when the block is configured, using a Hand-held Monitor. The Reference Number is the address in PLC memory reserved for use by the block's inputs and outputs. This chapter explains:

- Assigning Reference Numbers in I/O or Register Memory.
- Analog block input and output data formats.
- Required programming for analog inputs.
- Required programming for analog blocks, High-speed Counter Blocks, and PowerTRAC Modules in I/O memory.

Assigning Reference Numbers in I/O or Register Memory

When a block is configured, either an I/O reference or register reference can be assigned as its Reference Number. For most applications, I/O blocks should be assigned Reference Numbers in I/O memory.

However, a block might be assigned a register reference to conserve I/O references or to permit more devices to be used on the same bus. When register references are used, allow enough to accommodate both the input data and the output data. Inputs are stored before outputs, and the input data and output data must each begin on a register boundary. For example, if a block with 8 inputs and 8 outputs were assigned to register memory, two registers would be required (one for the inputs and one for the outputs). If the beginning address assigned to the block was R0129, the input data would be occupy the first 8 bits of R0129 and the output data would occupy the first 8 bits of R0130.

When the following blocks are assigned to I/O memory, they may require some special ladder logic to avoid premature access to data in the input table while the Bus Controller is still updating that input data: 4 Input/2 Output Analog Blocks, Current-source Analog Blocks, RTD Input Blocks, Thermocouple Blocks, PowerTRAC Blocks. See page 3-9 for more information.

Programming Required for Blocks Assigned to Register Memory

Automatic diagnostics and automatic I/O update are not performed on blocks assigned to register references. References in register memory are NOT updated during the I/O scan portion of the sweep. A window must be opened (using a DPREQ or WINDOW instruction at the beginning of the sweep), to update I/O points assigned to register memory. An "idle" DPREQ or WINDOW instruction can be used. To obtain fault reports from a block assigned to register memory, a Read Diagnostics Datagram must be sent to the block using the Send Datagram command.

The Read Configuration, Write Configuration, Read Diagnostics, and Read Analog Inputs commands cannot be send to a block that is assigned a register Reference Number. These functions must be done using Datagrams.

Assigning Reference Numbers in I/O Memory

Reference Numbers can be selected from any available references in a chain (or channel in an Expanded I/O system). Each Reference Number must begin on an even byte boundary. A byte boundary is a number that always leaves a remainder of 1 when divided by 8 (1, 9, 17 ...). The references used by each device go up in sequence (0001, 0002, 0003 ...) to the maximum required by the device (always multiples of 8 references).

In an Expanded I/O system, it will probably be most convenient to assign the first references in each channel to the Bus Controller, then assign the I/O blocks to references beginning at 49.

Avoiding Reference Number Conflicts

References assigned to any Genius block must not conflict with or overlap references assigned to other devices anywhere else in the PLC system. This includes other blocks on the same or any other bus, other Bus Controllers, or conventional I/O modules.

In an Expanded I/O system, if multiple Bus Controllers on the same bus are installed directly in the CPU rack and any references assigned to blocks on that bus overlap, a bus fault message will be generated. However, if the Bus Controllers are on different busses there will be a bus conflict resulting in erroneous I/O data for the overlapping references. This problem may be difficult to detect when the system is operating.

If multiple Bus Controllers are installed downstream of an I/O Transmitter module and any references assigned to the I/O blocks on those Bus Controllers overlap, a system parity error may result, and the system may shut down.

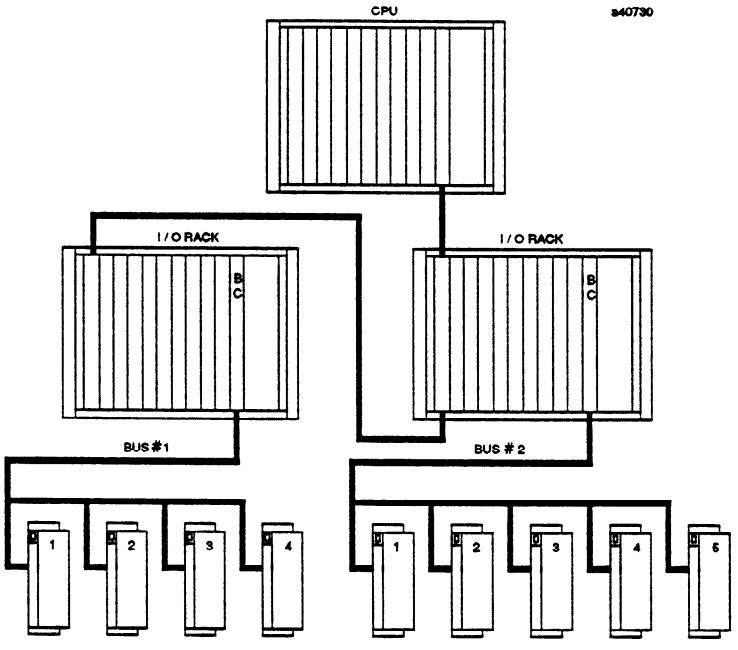
I/O References Used by a Block

4 Input/2 Output analog blocks and RTD Input blocks use 24 references in the input table. 4 Input/2 Output analog blocks also use 32 references in the output table. Analog block data format is shown later in this chapter.

The references needed for a discrete block depend on the number of circuits it has, and whether it has all inputs, all outputs, or both. Point 1 on a discrete block occupies the lowest-numbered reference assigned (which is also the block's beginning address).

- All-inputs discrete blocks, or I/O blocks configured as inputs-only blocks, need one reference in the input table for each circuit on the block.
- All-outputs blocks, or blocks configured as outputs-only blocks use references in the output table only.
- Discrete blocks with both inputs and outputs will use references in both the Input and Output Tables, beginning at the configured Reference Number. This includes blocks configured as combination blocks, *even if the I/O circuits on the block are set up as all inputs or all outputs.*

The following example shows a non-Expanded I/O system with two Bus Controllers on the same I/O chain.



	BUS #1			
	1	2	3	4
BLOCK TYPE	I	O	OI	I
INPUT REFERENCES	49-56	-	65-72	73-80
OUTPUT REFERENCES	-	49-64	65-72	-

The first Bus Controller is assigned to references 1 through 48. Block 1 on bus 1 is a discrete 8-circuit block with inputs only. It is assigned to Reference Number 0049, reserving 8 input references (I0049-I0056). *Because the block is configured for inputs only,*outputs O0049-O0056 are not reserved; they can be used by other blocks.

On the same bus, block 2 is a discrete 16-circuit block configured to use outputs only. Reference Number 0049 can also be assigned to that block. The block will use 16 references (O0049-O0064) in the output table.

Depending on the actual I/O mix, some input or output references will not be used for physical devices. References not used by I/O on a block (such as inputs I0057-I0064 in this example) are not available for use by other physical I/O. However, they can be used for logical coils in the program.

Block 3 on the same bus is a discrete 8-circuit block that uses 5 inputs and 3 outputs. Reference Number 0065 is assigned to the block. Because it is a combination block, it needs equivalent references in both the input table (I0065-I0072) and the output table (O0065-O0072). However, only 5 input references and 3 output references will actually correspond to field devices. The remaining three inputs will contain feedback data from the output circuits and cannot be used for internal program logic.

Block 4 on bus 1 is an input block assigned to I0073-I0080. To allow for the addition of blocks to both busses in the future, the second Bus Controller is assigned to I/O references 513-560. The blocks on bus 2 are assigned I/O references from 561 to 993.

First and Last I/O Table References

Reference ranges for blocks with 8, 16, 24, or 32 circuits are listed below. References Number assignments can be recorded on copies of the Configuration Worksheet on the next page.

First Reference	Last Reference 8	16	24	32
001	008	016	024	032
009	016	024	032	040
017	024	032	040	048
025	032	040	048	056
033	040	048	056	064
041	048	056	064	072
049	056	064	072	080
057	064	072	080	088
065	072	080	088	096
073	080	088	096	104
081	088	096	104	112
089	096	104	112	120
097	104	112	120	128
105	112	120	128	136
113	120	128	136	144
121	128	136	144	152
129	136	144	152	160
137	144	152	160	168
145	152	160	168	176
153	160	168	176	184
161	168	176	184	192
169	176	184	192	200
177	184	192	200	208
185	192	200	208	216
193	200	208	216	224
201	208	216	224	232
209	216	224	232	240
217	224	232	240	248
225	232	240	248	256
233	240	248	256	264
241	248	256	264	272
249	256	264	272	280
257	264	272	280	288
265	272	280	288	296
273	280	288	296	304
281	288	296	304	312
289	296	304	312	320
297	304	312	320	328
305	312	320	328	336
313	320	328	336	344
321	328	336	344	352
329	336	344	352	360
337	344	352	360	368
345	352	360	368	376
353	360	368	376	384
361	368	376	384	392
369	376	384	392	400
377	384	392	400	408
385	392	400	408	416
393	400	408	416	424
401	408	416	424	432
409	416	424	432	440
417	424	432	440	448
425	432	440	448	456
433	440	448	456	464
441	448	456	464	472
449	456	464	472	480
457	464	472	480	488
465	472	480	488	496
473	480	488	496	504
481	488	496	504	512
489	496	504	512	520
497	504	512	520	528
505	512	520	528	536
513	520	528	536	544
521	528	536	544	552
529	536	544	552	560
537	544	552	560	568
545	552	560	568	576
553	560	568	576	584
561	568	576	584	592
569	576	584	592	600
577	584	592	600	608
585	592	600	608	616
593	600	608	616	624
601	608	616	624	632
609	616	624	632	640
617	624	632	640	648
625	632	640	648	656
633	640	648	656	664
641	648	656	664	672
649	656	664	672	680
657	664	672	680	688
665	672	680	688	696
673	680	688	696	704
681	688	696	704	712
689	696	704	712	720
697	704	712	720	728
705	712	720	728	736
713	720	728	736	744
721	728	736	744	752
729	736	744	752	760
737	744	752	760	768
745	752	760	768	776
753	760	768	776	784
761	768	776	784	792
769	776	784	792	800
777	784	792	800	808
785	792	800	808	816
793	800	808	816	824
801	808	816	824	832
809	816	824	832	840
817	824	832	840	848
825	832	840	848	856
833	840	848	856	864
841	848	856	864	872
849	856	864	872	880
857	864	872	880	888
865	872	880	888	896
873	880	888	896	904
881	888	896	904	912
889	896	904	912	920
897	904	912	920	928
905	912	920	928	936
913	920	928	936	944
921	928	936	944	952
929	936	944	952	960
937	944	952	960	968
945	952	960	968	976
953	960	968	976	984
961	968	976	984	992
969	976	984	992	1000
977	984	992	1000	xxx
985	992	1000	xxx	xxx
993	1000	xxx	xxx	xxx

I/O Reference Number Assignment
Configuration Worksheet

Main I/O Chain (corresponds to channel 0) ____.
Auxiliary I/O Chain (corresponds to channel 8) ____.
Expanded I/O Channel Number (1-7, 9-F) ____.
Channelization provided by Bus Controller (Y/N) ____.

Ref. Start:	001	009	017	025	033	041	049	057	065	073	081	089	097	105	113	121	129	137	145	153	161
End:	008	016	024	032	040	048	056	064	072	080	099	096	104	112	120	128	136	144	152	160	168
INPUTS USED																					
OUTPUTS USED																					

Ref. Start:	169	177	185	193	201	209	217	225	233	241	249	257	265	273	281	289	297	305	313	321	329
End:	176	184	192	200	208	216	224	232	240	248	256	264	272	280	288	296	304	312	320	328	336
INPUTS USED																					
OUTPUTS USED																					

Ref. Start:	337	345	353	361	369	377	385	393	401	409	417	425	433	441	449	457	465	473	481	489	497
End:	344	352	360	368	376	384	392	400	408	416	424	432	440	448	456	464	472	480	488	496	504
INPUTS USED																					
OUTPUTS USED																					

Ref. Start:	505	513	521	529	537	545	553	561	569	577	585	593	601	609	617	625	633	641	649	657	665
End:	512	520	528	536	544	552	560	568	576	584	592	600	608	616	624	632	640	648	656	664	672
INPUTS USED																					
OUTPUTS USED																					

Ref. Start:	673	681	689	697	705	713	721	729	737	745	753	761	769	777	785	793	801	809	817	825	833
End:	680	688	696	704	712	720	728	736	744	752	760	768	776	784	792	800	808	816	824	832	840
INPUTS USED																					
OUTPUTS USED																					

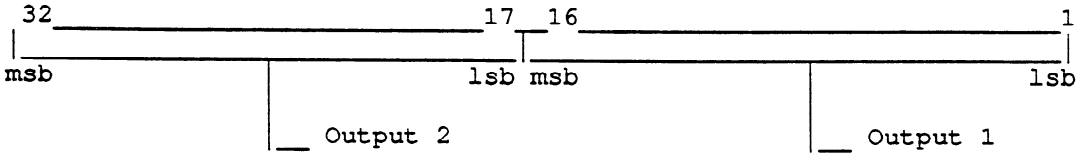
Ref. Start:	841	849	857	865	973	881	889	897	905	913	921	929	937	945	953	961	969	977	985	993	
End:	848	856	864	872	880	888	896	904	912	920	928	936	944	952	960	968	976	984	992	1000	
INPUTS USED																					
OUTPUTS USED																					

Analog Input and Output Data

An analog block (of any type, such as a 4 Input/2 Output Analog block, or an RTD block) uses 24 references in the input table. A 4 Input/2 Output Analog block also uses 32 references in the output table. Analog input data is multiplexed into the input table references, so that data from only one of the circuits on the block is presented each sweep. The next input circuit is presented during the following sweep. The number of sweeps required is the same as the number of input circuits on the block.

Analog Output Data

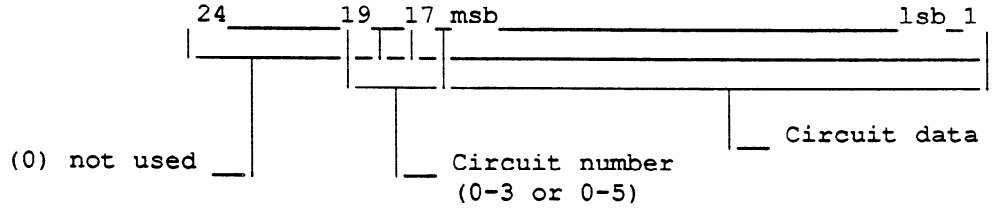
Analog blocks with 4 inputs and 2 outputs use 32 output references for the two circuits.



Unlike input data (described below), output data is not multiplexed. Data for output 1 is stored in (relative) references 1-16. Data for output 2 is stored in (relative) references 17-32. Their data formats are identical and are in two's complement format. They represent a scaled engineering units value. Refer to *the Genius I/O System Uses's Manual* for information about scaling analog data.

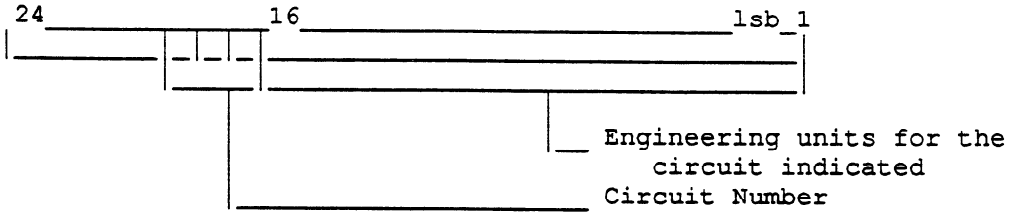
Analog Input Data

During each Genius I/O bus scan, analog blocks send data from all of their input circuits to the Bus Controller and the Hand-held Monitor. During the CPU sweep, the CPU reads from the Bus Controller the value of one analog circuit per block. This analog value is placed in the block's input references as shown below.

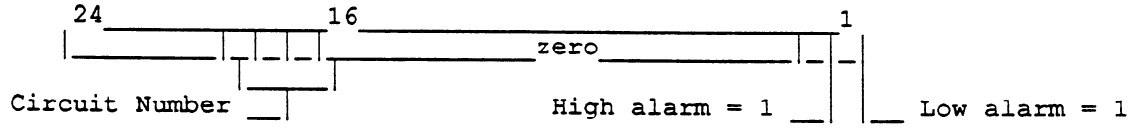


References 1-16 store the circuit data. The content of these bits depends on whether the input is configured for Normal Input Mode or Alarm Input Mode. (Alarm Input Mode, which is a feature of the 4 Input/2 Output Analog block only, replaces actual circuit data with alarm data).

In Normal Input Mode, bits 1-16 store the two's complement engineering units value of a circuit. Bit 1 is the LSB and bit 16 is the MSB.



In Alarm Input Mode, bits 1-2 store the Alarm Input mode data for the circuit. Bit 1 = 1 indicates that the current input value exceeds the programmed Low Alarm limit. Bit 2 = 1 indicates that the current input value exceeds the programmed High Alarm limit. If both bits = 0, no alarm limits have been exceeded.



Relative references (bits) 17-19 store a number that identifies the circuit on the block. The circuit number is a binary value ranging from 0 (which represents input circuit #1) to 5, depending on the number of inputs on the analog block.

The value remains in the CPU's input table for only one CPU sweep. On the next sweep, it is replaced by the value of the next analog input on the block, and the circuit number is changed accordingly.

Programming for Analog Inputs

The input circuit number and circuit value from one analog input per block are automatically placed in the block's assigned input references each CPU sweep. No ladder logic is required to move the value into the references, although ladder logic is needed to capture this value and place it into a register. This is explained on the next page. DO I/O instructions do not obtain data that is any "fresher" than the data already available, and therefore should not be used for analog inputs.

If faster update of analog inputs is desired, ladder logic can be used to read all of a block's analog inputs directly into registers in a single CPU sweep. For information, see the description of Read Analog Inputs in chapter 5.

Transferring Analog Inputs to CPU Registers

Because each analog input value stays in the CPU's input table for just one CPU sweep, logic should be used to capture analog inputs from these references and move them into registers. Example logic is shown below.

```
<< RUNG 1 >>

 ************************************************************************
 * The rung below moves the input circuit number to a register (R0001).  It  *
 * will be used as a pointer in the next rung to demultiplex the four        *
 * analog input circuits and to store the data into a table of four          *
 * registers.                                                                *
 ************************************************************************

  BUS      INPUT          TABLPTR
CONTRLR  CIRCUIT         ANALOG
  OK      NUMBER         INPUTS
 I0257    I0353           R0001
+--] [---[  MOVE RIGHT 8 BITS  ]-                                    (  )

 << RUNG 2 >>

 ************************************************************************
 * The Source-to-Table Move instruction below takes the analog input data    *
 * for each of the four input circuits from I0337 through I0352 and then     *
 * stores it into a table starting at register R0002.  Register R0001        *
 * contains the analog circuit number and is the pointer to one of the       *
 * four registers in the table.  Six should be used for an RTD block (LEN=6).*
 ************************************************************************

  BUS     ANALOG   TABLPTR
CONTRLR    DATA    ANALOG
  OK        IN     INPUTS
 I0257    I0337     R0001   Const
+--] [---[ SRC-TO-TABLE       LEN]-                                  (  )
                              004

 << RUNG 3 >>

 ************************************************************************
 * This rung turns off the Bus Controller OK bit at the end of each CPU      *
 * sweep so that the Bus Controller can turn it back on.  It is then used in *
 * the logic above to enable the analog data to be processed.  If the Bus    *
 * Controller OK bit does not come back on the analog data is not processed. *
 ************************************************************************

                            BUS
                          CONTRLR
                            OK
 Const            I0257    Const
+[ BIT CLEAR    MATRIX       LEN]-                                  (  )
 00001                       001
```

Logic is needed to transfer analog inputs only. Analog outputs and discrete inputs and outputs are not multiplexed.

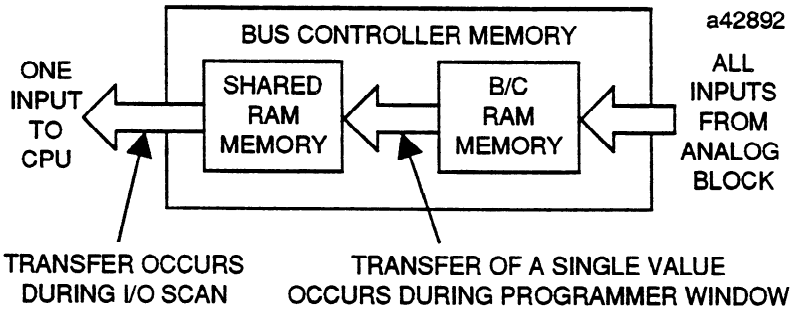
Special Programming Required for:

Analog Blocks (all types)
High-Speed Counter Blocks
PowerTRAC Blocks

The application program may need to include logic to extend the programmer window time if the following blocks are assigned to I/O references:

- 5 or more analog, RTD, or Thermocouple blocks
- 5 or more PowerTRAC blocks (IC660BPM100 version C or later)
- 1 or more High-speed Counter blocks
- 1 or more Power Monitor Modules (IC660BPM100 version B or earlier)

This logic should be located before any logic that tries to access the input information.



The Bus Controller begins organizing input data from the blocks listed above at the start of the programmer window portion of the CPU sweep, and continues until all the data has been organized. If the program includes DO I/O instructions to obtain the latest data from the block or if the program is very short, program logic should be used to extend the time of the programmer window. This will allow the Bus Controller time to complete the task before the I/O update begins.

Logic to Extend the Programmer Window

To find the total time needed to update inputs during the programmer window, find the contributions of any analog, RTD, Thermocouple, High-speed Counter, PowerTRAC or Power Monitor Blocks on the bus that are assigned I/O Reference Numbers.

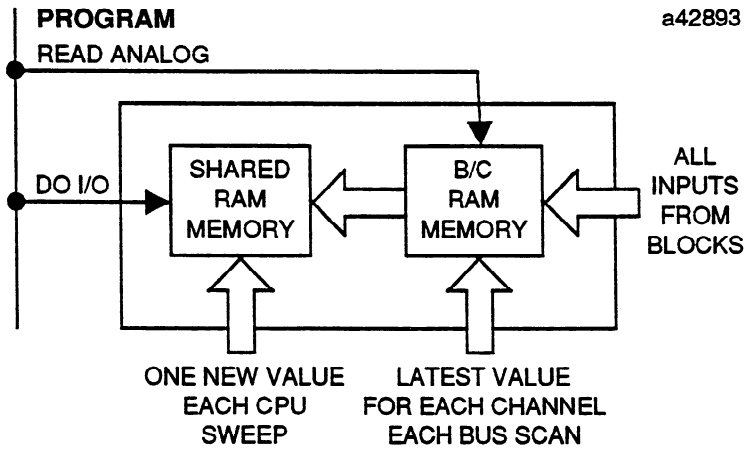
Number of analog, RTD, Thermocouple and PowerTRAC blocks:	_____	x 0.057mS	=	_____mS
Number of High-speed Counters:	_____	x 0.422mS	=	_____mS
Number of Power Monitor Blocks:	_____	x 0.503mS	=	_____mS
		total	=	_____mS
Subtract the programmer window time:			—	0.311mS
				_____mS delay required

To extend the programmer window, add one of the following commands to the program before any I/O update (either normal I/O update or DO I/O instructions):

- to extend the programmer window by 1.2mS, direct an extra idle (status = 0) DPREQ or WINDOW instruction to the Bus Controller.
- to extend the programmer window by 5mS, program a DPREQ or WINDOW instruction with no Bus Controller address.

Using the Read Analog Inputs Command

Another way to update inputs from these devices is with a Read Analog Inputs command, which reads values from the Bus Controller's own RAM memory (not from shared RAM). This area of memory always contains the latest values from all input circuits on each such device. If the program is required to have the most current values of these inputs as the logic executes, a Read Analog Inputs command should be used instead of a DO I/O instruction. DO I/O reads input values from shared RAM. Since the Bus Controller updates shared RAM only once per CPU sweep, multiple DO I/O instructions in the same program sweep will return the same values each time.



For more information about the Read Analog Inputs command, see chapter 7.

Chapter 4 Automatic Diagnostics and Fault Clearing

GFK-0171

This chapter explains how set up an application program to use the built-in Genius I/O diagnostics and fault-clearing features of the Logicmaster 6 software. Use of these features requires the Expanded Functions in both the PLC and in the software. If the system will make use of more than one application program at different times, each program should be set up as described in this chapter. Begin by starting up the Logicmaster 6 software and loading the application program into programmer memory.

The CPU Configuration Instruction

To access the Expanded Functions, the CPU Configuration instruction must be placed in the ladder logic program. When this has been done, the following line should appear as rung 1 of the program:

```
-|SERIES SIX CONFIGURATION DATA|-
```

If there is already a CPU Configuration instruction in the program, do not enter another one. With the CPU Configuration instruction in the program, the screens shown on the following pages can be used to set up and display information about the Bus Controllers and Genius I/O in the PLC system. *If you change any of the setup screens for a program that was already stored to the Series Six CPU, the program must be stored to the CPU again for the changes to be used.*

Expanded Functions Menu

Press F7 (Expanded Functions) from the Supervisor menu to display the Expanded Functions menu.

```
                                                    L/M OFFLINE

                    L O G I C M A S T E R  (TM) 6
                  E X P A N D E D   F U N C T I O N S


      KEY 3                                                  FUNCTION

      F1 - CPU CONFIG . . . . . .Display/Modify CPU Configuration
      F2 - I/O FAULTS . . . . . . Display/Clear Genius I/O Faults
      F4 - MSD FUNC . . . . . . Display/Modify Machine Setup Data
      F5 - 90-70 CONFIG . . . . . . Display/Modify 90-70 operands
      F6 - 90-70 DSPLY. . . . . Display 90-70 operands and status
      F8 - SUPERV MENU. . . . . . . . . . Return to Supervisor Menu


  CPU    I/O            MSD             90-70   90-70           SUPERV
1CONFIG 2FAULTS 3      4FUNC           5CONFIG 6DISPLY 7       8 MENU
```

This menu is used to select several different Expanded CPU functions (not all are related to Genius I/O diagnostics).

CPU Configuration Setup Menu

The first step in customizing the application program is to match it to the intended characteristics of the system. This is done by completing entries on the CPU Configuration Setup Menu. To access this menu, select F1 (CPU Configuration) from the Expanded Functions menu.

```
                                                    L/M OFFLINE

      C P U   C O N F I G U R A T I O N   S E T   U P   M E N U

  EXPANDED I/O       ENABLED                    Y        (Y/N)
  SCAN:
                     BEGIN RANGE      CHANNEL 0  POINT    1
                     END RANGE        CHANNEL 0  POINT 1024

 GENIUS I/O          DIAGNOSTICS ENABLED        Y        (Y/N)
 DIAGNOSTICS:        DIAGNOSTIC TABLES          Y        (Y/N)
                     B/C -> POINT FAULTS        N        (Y/N)
                     DIAGNOSTIC RANGE LIMIT     7        (0-7) CHANNELS
                     CPU REGISTER SIZE       8192              REGISTERS
                     FAULT TABLE LENGTH         8              ENTRIES
                     BUS STATUS/CONTROL
                     BYTE LOCATION            993

   COMPUTER MAILBOX: ENABLED                    N        (Y/N)

  B/C                                                          XPNDED
1 MAP   2       3       4                5       6       7       8 FUNC
```

The CPU Configuration Setup Menu displays entries for selecting:

- Expanded I/O scanning.
- Automatic diagnostics and fault clearing.
- The Computer Mail Box.

When the CPU Configuration function described on the previous page is placed in the program, it uses the entries currently selected on this menu. Check the entries on the screen against the descriptions in this chapter.

Diagnostics Enabled: Setting this entry to Y causes the CPU to create a fault table to automatically store Genius I/O faults. The table will occupy the number of registers specified by the entry FAULT TABLE LENGTH (see the next page). The maximum length and location of the fault table depend on the CPU memory capacity. See appendix A for the maximum fault table length and location for your CPU.

Individual Bus Controllers will always be capable of reporting faults and clearing faults. Selecting Diagnostics Enabled = YES allows all fault reports from all configured Bus Controllers to be automatically gathered together; it also allows all faults on all busses to be cleared with a single action from the programmer or PLC.

A Bus Controller with diagnostics automatically provides fault information for this table. If diagnostic reports are not needed from a Bus Controller, it may be excluded from diagnostics scanning by specifying a smaller range with the entry for DIAGNOSTIC RANGE LIMIT (see below).

If this entry is set to N, the CPU will not store diagnostics information; neither automatic nor programmed diagnostics or fault clearing from the programmer will be possible.

However, fault reports and fault clearing will still be accessible with a Hand-held Monitor on the bus.

Diagnostic Tables: This is an optional selection; *for most applications it should be set to N.* If this entry is set to Y, when an I/O circuit fault occurs the CPU sets the internal bit which corresponds to the circuit reference to 1. For example, if a fault occurs at I2+0003, the CPU sets the internal bit at I2-0003 to 1.

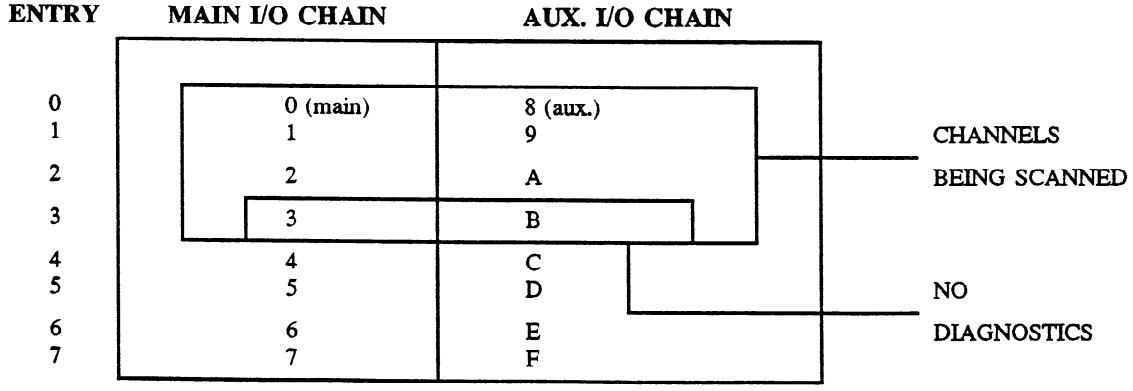
Because this function will operate over the entire range of Genius I/O for which diagnostics is enabled (by the entry DIAGNOSTIC RANGE LIMITS), it requires that an amount of memory *equal to the amount for which diagnostics scanning is enabled* be set aside in the CPU. For example, if Expanded I/O scanning is enabled for channel pairs 1-3 and 9-B, the CPU will use registers R129 - R512 and R1153 - R1472 for inputs and outputs. If Diagnostic Tables were enabled by setting this entry to Y, the CPU would use registers R2177 - R2496 and R3201 - R3520 to store I/O circuit faults.

If this entry is set to Y, it is important to avoid using any registers within the reserved area for any other purpose (the location of internal references in register memory is shown in appendix A). In addition, Bus Controller reference numbers must be assigned (with the backplane DIP switches) on 256-reference boundaries, NOT on 64-reference boundaries. The Bus Controller will use the additional references for I/O diagnostics.

B/C -> Point Faults: This entry is an extension of the one directly above; it defaults to N. *Unless the application requires all I/O point status bits to be set in case of Bus Controller failure, do not select Y for this entry.*

If both entries above were set to Y, setting this entry to Y would cause the CPU to set all 208 input and 208 output point fault bits associated with a Bus Controller to 1 during any CPU sweep if the Bus Controller failed to communicate with the CPU. This would affect only the internal status table; it would not change any real I/O states. To use these internal references, additional program logic would be required.

Diagnostic Range Limits: This entry specifies the upper limit for diagnostics scanning. The CPU scans I/O channels as pairs. For a system with non-Expanded I/O addressing, this entry should be 0 to scan the Main/Auxiliary I/O chain pair. For a system with Expanded I/O addressing, it defaults to the highest channel pair for which I/O scanning is enabled. For example, if Expanded I/O scanning for channels 0-3 and 8-B were enabled, but diagnostic reports were not needed from channels 3 and B, you would enter the number 2 here.



In this example, the program would report diagnostics from channel pairs 0-2 and 8-A.

CPU Register Size: This defaults to the number of registers in the Scratch Pad. The availability of register memory may limit the number of I/O points for which diagnostics can be stored. If the CPU register size were 256, diagnostics could be stored for only the first 256 inputs and outputs in the Main I/O chain. For a CPU register size of 1K, diagnostics would be stored up to the first 1024 inputs and 1024 outputs of the Main I/O chain. If the CPU register size is either 8K or 16K, diagnostics can be stored for 16K inputs and 16K outputs. If the Scratch Pad has already been set up, this entry should be correct for the CPU. If it is not correct, change it by entering 256 or 1 (for 1K), or 8 (for 8K) or 16 (for 16K).

Fault Table Length: If DIAGNOSTICS ENABLED is set to Y, this entry specifies the size (in registers) of the area in CPU memory where faults will be stored. The length of the fault table determines the number of faults that can be stored at the same time. (A fault will remain in the table until it is cleared by the program or from the computer keyboard). If the table became full, there would be no room for additional fault storage, and the overflow faults would be lost. No new faults would be stored until the table was cleared. The default fault table length is 8. Each fault occupies 10 registers of memory. The default Fault Table size, therefore, is 80 registers.

Each entry in the Fault Table has the following format:

```
Register 1:             Bus Controller Address
      bits 0-9:         Reference of Bus Controller (1-1000)
      bits 10-11:       zeros
      bits 12-15:       Bus Controller channel (0-F)
Register 2:             Series Six I/O Address
      bits 0-9:         I/O reference of circuit (1-1000) within a channel
      bit 10:           1 = input reference affected
      bit 11:           1 = output reference affected
      bits 12-15        Series Six I/O channel number (0-F)
Register 3:
      bits 0-3          relative number of the circuit on the block (0-15*)d
                        with zeros if the fault is not a circuit fault.
      bits 4-7          unused, set to zero
      bits 8-15         Bus Controller status byte #3
Register 4:
      bits 0-7          Bus Controller status byte #4
      bits 8-15         Bus Controller status byte #5
Register 5:
      bits 0-7          Bus Controller status byte #6
      bits 8-15         Fault type
                        bit 8: 1  = Bus Controller not responding
                               0  = Bus Controller OK
                        bit 9: 1  = Serial bus error
                        bit 10: 1 = Circuit fault
                        bit 11: 1 = Loss of block
                        bit 12: 1 = Addition of block
                        bit 13: 1 = I/O address conflict
                        bit 14: 1 = Block terminal assembly EEPROM failure
                        bit 15: unused, set to 0
Register 6:             Fault type
      bits 0-3          0000 = Block headend fault
                        0001 = discrete circuit fault
                        0010 = analog circuit fault
                        1001 = discrete circuit fault (circuits 17-32 only) **
                        0100 = RTD circuit fault **
      bits 4-7          unused, set to 0
      bits 8-15:        Fault description
                        bit 8: 1 = Loss of I/O power (Isolated block only)
                        bit 9: 1 = Short circuit
                        bit 10: 1 = Overload
                        bit 11: 1 = No load/open line
                        bit 12: 1 = Over temperature
                        bit 13: 1 = Switch failed
                        bits 14, 15: undefined
Register 7:             Fault description
      bit 0:            1 = Analog input low alarm
      bit 1:            1 = Analog input high alarm
      bit 2:            1 = Analog input under range
      bit 3:            1 = Analog input over range
      bit 4:            1 = Analog input open wire
      bit 5:            1 = Analog output under range, or RTD wiring error
      bit 6:            1 = Analog output over range, or RTD internal fault
      bit 7:            1 = not used, or RTD circuit shorted
      bits 8-15:        undefined, set to 0
Register 8:             Fault time - seconds
      bits 0-7:         tenths of seconds (00-09) two binary-coded decimal
                        digits
      bits 8-15:        seconds (00-59) two BCD digits
Register 9:             Fault time - minutes/hours
      bits 0-7:         minutes (00-59) two BCD digits
      bits 8-15:        hours (00-23) two BCD digits
Register 10:            Fault time - days
      bits 0-7:         days (00-99) two BCD digits
      bits 8-15:        hundreds of days (00-99) two BCD digits
```

* Add 16 to this if fault type is 1001.
** Bits 0-3 will show faults 1001 and 0100, but no further decoding is performed in register 6 or 7. Logicmaster 6 version 4.01 decodes the Bus Controller status bytes for this information.

The size selected for the fault table will depend on the amount of register memory available, and the number of faults that are expected to accumulate before an operator is able to clear them.

The amount of register memory available for the Fault Table depends both on the entry for CPU Register Size (above), and on the Computer Mailbox. Appendix A shows the allocation of registers in CPU memory.

The Computer Mailbox is explained on the next page. If the Computer Mailbox is enabled, it will occupy some of the register memory that would otherwise be used for the fault table. Maximum table lengths (number of faults that can be stored) are listed below.

REGISTER SIZE	MAXIMUM TABLE LENGTH	
	COMPUTER MAILBOX ENABLED?	
	NO	YES
256	8	1
1K	68	75
8K	399	406
16K	1218	1225

Bus Status/Control Byte Location: This is an address shared by ALL Bus Controllers in the system for status and control information. The Logicmaster 6 software defaults this address to 993, which reserves references 993 to 1000 in both the input and output tables in the Main I/O table for the Bus Controllers. Any byte of available memory may be assigned, however *it is important that the contents of this memory location not be used for any other purpose by the program.* If the application program requires the use of I/O addresses 993-1000 in the Main I/O chain, change this entry on the CPU Configuration Setup Menu.

Computer Mailbox: Enabled: The Computer Mailbox is an area of memory that can be reserved for communications between devices. If the Computer Mailbox is enabled, any device that needs to communicate with another device can place a command and address of the other device in the Computer Mail Box. The CPU detects that a command is waiting, and opens a window to that device. The device should look in the mailbox registers for a command, then either read or write data (depending on the command specified). Devices that can communicate using the Computer Mailbox include the Series Six CPU, the ASCII/BASIC Module, the CCM Module, and the Bus Controller.

The Computer Mailbox occupies the last 70 consecutive registers out of the total number available. Therefore, its starting address varies according to the CPU register memory size. For example, for a CPU with 8K register memory, the Computer Mailbox is located from R8123 through R8192. If the Computer Mailbox is not enabled, these registers are available for other use, for example, for the CPU fault table described above.

Genius Bus Controller Locations Screen

After completing the CPU Configuration Setup Menu, the next step is to indicate the locations of the Bus Controllers. Press B/C Map (F1) from the CPU Configuration Setup Menu. The screen will show each possible Bus Controller location in the system:

```
                                                L/M OFFLINE

        G E N I U S   B U S   C O N T R O L L E R   L O C A T I O N S

                CURSOR:        CHANNEL  0   LOCATION  0001


           MAIN CHAIN                              AUX. CHAIN
         CHANNEL      LOCATIONS                 CHANNEL      LOCATIONS

            0     00000000 00000000                8      00000000 00000000
            1     00000000 00000000                9      00000000 00000000
            2     00000000 00000000                A      00000000 00000000
            3     00000000 00000000                B      00000000 00000000
            4     00000000 00000000                C      00000000 00000000
            5     00000000 00000000                D      00000000 00000000
            6     00000000 00000000                E      00000000 00000000
            7     00000000 00000000                F      00000000 00000000

              INSERT                                                    CPU
    1       2  BC   3        4           5        6        7        8CONFIG
```

The number beside the word LOCATION represents a possible Reference Number that might be assigned to a Bus Controller (using the backplane DIP switches). The cursor first appears at the rightmost location in the Main I/O chain (location 0001). Move the cursor to select a location, then press F2 to insert a Bus Controller. For example, this partial screen shows a Bus Controller at address 0129 in the Main I/O chain (channel 0):

```
                                  ___ Bus Controller        ___ location
                                 |                         |
                                 |                         |
                CURSOR:          | CHANNEL 0  LOCATION  0129
                                 |
           MAIN CHAIN            |                  AUX. CHAIN
         CHANNEL      LOCATIONS  |               CHANNEL      LOCATIONS
                                 |
            0     00000000 00000100|                 8      00000000 00000000
            1     00000000 00000000|                 9      00000000 00000000
```

In this example, the CPU would use the 64 I/O references starting at 0129 for diagnostic data for the Bus Controller assigned to Reference Number 0129 in the Main I/O chain. This is explained in chapters 8 and 9. If there were only one Bus Controller in the system, no other entries would be needed.

Entering the Bus Controller locations on this screen completes the setup steps.

Genius I/O Fault Table Screen

When the PLC system is in operation, faults can be displayed and cleared from the Genius I/O Fault Table screen. This screen lists all the faults currently stored by the CPU. Faults appear in the order they are reported to the CPU, with the first fault at the top. *To display and clear faults from 32-Circuit DC blocks and RTD blocks, Logicmaster 6 version 4.01 or later is required.*

```
  TOTAL FAULTS: 0000          GENIUS I/O FAULT TABLE              date
  TOP FAULT DISPLAYED:0000                                        time
  FAULT DISPLAYED:                                  0000:00:00:00:0

B/C   POINT  CIRC      FAULT        FAULT        FAULT
ADDR. ADDR.  NO.      CATEGORY      TYPE      DESCRIPTION    DAY:HR:MN:SC:T
----- ------ ---- ---------------- -------- ---------------- ---------------

    NEXT     PREV   CLEAR                                      XPNDED
  1 PAGE   2 PAGE  3FAULTS 4 TOP       5BOTTOM 6       7       8 FUNC
```

This screen can only be displayed if the current ladder logic program includes the CPU Configuration function, and if Genius I/O diagnostics is enabled in the CPU Configuration Set Up Menu.

The table includes both I/O block faults, and the following faults which may be reported directly by the Bus Controller:

- Bus Controller fault
- Bus Error
- Loss of block
- Addition of block
- Address conflict
- EEPROM failure

Fault definitions are given in chapter 5. The format of each fault listing in the table is explained on the following pages.

Fault Table Display: Definitions

The entries in the fault table show the following information about a fault. If a fault occurs that has more than one category (see below) each description is listed as a separate line in the table.

B/C Addr: *Bus Controller I/O Reference:* Displayed for all faults. This entry has two fields:

Series Six Channel Number: The first field is the number of the channel where the error occurred. A hex value from 0 to F. The Main I/O Status Table is shown as 0 and the Auxiliary I/O Status Table is shown as 8.

MAIN	AUXILIARY
0	8
1	9
2	A
3	B
4	C
5	D
6	E
7	F

Channel Byte Address: The second field is byte address of the error within the indicated channel. Range = 0 to 125.

Point Addr: *Series Six Point Address:* Not displayed for a Bus Controller or Serial Bus fault. This entry has two fields:

Input/Output: The first two characters indicate an input (I) or output (O). Both may appear for some faults.

Address: The second field shows I/O reference address of the error within the channel (base address for analog blocks). Range = 1 to 1000.

Circuit Number: The circuit number on the block. Displayed only for a circuit fault. Range = 1 to 32.

Fault Category: This entry shows the category of error that has occurred.

BC FAULT
BUS ERROR
CIRCUIT FAULT
LOSS OF BLOCK
BLOCK ADDITION
ADDRESS CONFLICT
EEPROM FAILURE
???????????????? (displayed for unknown entries)

Fault Type: If the fault is an I/O block circuit fault, this entry shows the error type: BLOCK, DISCRETE, or ANALOG. For other types of faults, this field is blank.

Fault Description: Displayed only if the Fault Category is CIRCUIT FAULT. Multiple lines are displayed if more than one description is associated with a fault data entry. Some example descriptions are:

EEPROM FAILURE	LOSS OF POWER
SHORT CIRCUIT	OVERLOAD
NO LOAD	OVER TEMP
SWITCH FAILED	LOW ALARM
HIGH ALARM	UNDER RANGE
OVER RANGE	OPEN WIRE
BC NOT OK	SERIAL BUS ERROR
LOSS OF MODULE	ADDITION OF MODULE
I/O ADDRESS CONFLICT	IN WIRING ERROR
INTERNAL FAULT	INPUT CHAN SHORT

Fault Time: The day, hour, minute, second, and tenth of a second. The value of the CPU's real-time clock when the error report was received from the Bus Controller.

Clearing Faults

Press the F3 (Clear Faults) key to clear all faults in the table. This sets the fault count to zero, clears out any faults currently buffered in the Bus Controller, and clears the fault data currently latched at the Genius I/O blocks. Subsequent incoming faults fill the Fault Table from the beginning. If a condition which caused a fault has not been corrected, the fault message will reappear.

For the serial version of Logicmaster 6 software, On-Line changes to the CPU must be enabled to clear faults from the programmer.

Printing a Copy of the Fault Table Screen

When the fault table is displayed on the screen, it can be printed using the ALT/P keys. (Printing must be set up as explained in *the Logicmaster 6 Software User's Manual).*

Chapter 5 Programming Window Commands

GFK-0171

This chapter explains how to use "window" commands if the program needs to communicate with a device on the bus.

Each window instruction recognizes the following commands:

1 **Idle**
2 **Read Configuration***
3 **Write Configuration***
4 **Read Diagnostics***
7 **Read Analog Inputs**
9 **Read Status Table Address**
11 **Switch BSM***
12 **Send Datagram**
13 **Receive Datagram**

* *Can also be programmed as a Datagram. See chapter 6 for more information.*

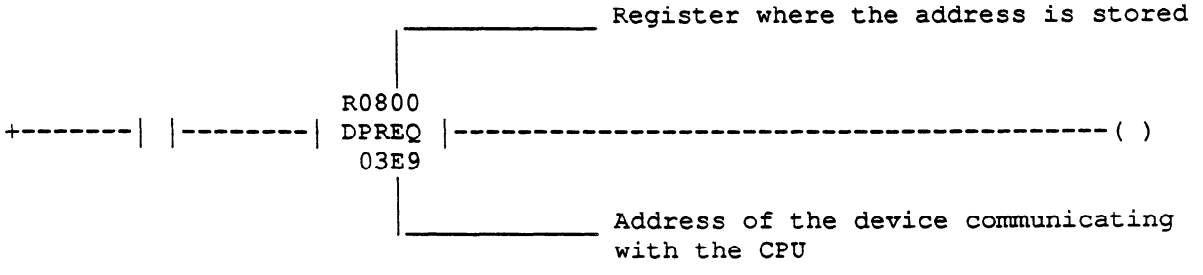
If the system includes multiple CPUs connected by a Genius bus, communications between them can be programmed as described in chapters 6 and 7.

Program Instructions

Communications between the CPU and one of its Bus Controllers can be specified with a DPREQ or WINDOW instruction in the program, or via the "Computer Mailbox". For most applications, a DPREQ will be used if the system is set up for Normal I/O addressing (main and auxiliary I/O chains only). A WINDOW instruction will be used if the system is set up for Expanded (channelized) I/O addressing.

Format of the DPREQ Instruction

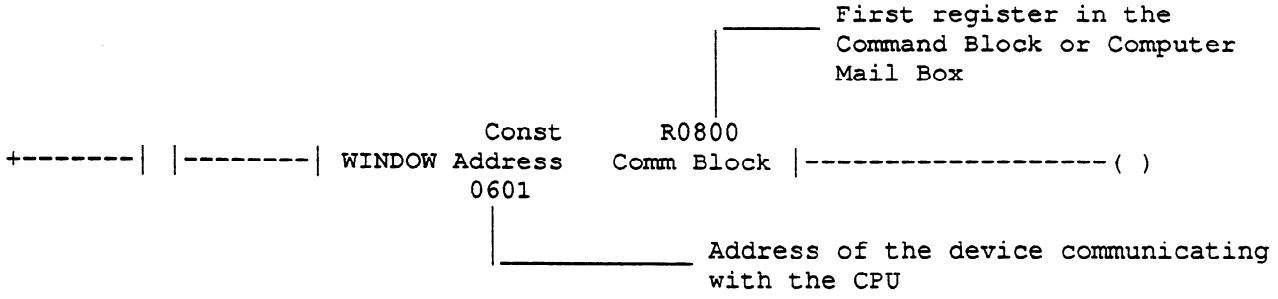
The ladder logic instruction for the DPREQ has this format:



Do not use a DPREQ in a system with Expanded addressing. It would be broadcast to the specified address on all channels. If that address were used for more than one intelligent device (for example, more than one Bus Controller), conflicting replies would be received.

Format of the WINDOW Instruction

The WINDOW instruction is similar to a DPREQ, except that it specifies both an address and a channel. When the CPU encounters a WINDOW instruction in the program, it opens a communications window to the device at the specified address on the specified channel. The ladder logic for the WINDOW instruction has this format:



The WINDOW instruction is available with Logicmaster 6 software release 3.0 and later. It requires a CPU with microcode version 110 or greater.

Using the Computer Mailbox

The Computer Mail Box is an automatic communications window that operates when the CPU is in Run mode. The only ladder logic required to enable the Computer Mail Box is the CPU Configuration instruction as the first rung in the program. Use of the Computer Mail Box must be set up on the CPU Configuration Setup Menu, as described earlier. Any device that needs to communicate with another can place a command and window address of the other device in the Mailbox. At the end of the CPU sweep, the CPU detects that a command is waiting and either reads data from the Mailbox or places data in the Mailbox to be returned to the requesting device. The window that allows the Bus Controller to access the Mailbox is generated automatically by the CPU at the end of the CPU sweep. It is the last Executive window, after the PDT, DPU, and CCM windows. It has a 5mS timeout length.

The CPU continues to open windows to the specified address every sweep there is a valid address in the Mailbox. To stop opening windows, the address must be cleared out of the Mailbox by the device that accepts the window or the device or program that set up the address in the Mailbox. Although one of the registers in the Mailbox contains a status code, the CPU will not examine this register, and will not automatically close the window. The Computer Mail Box always uses the last 70 consecutive registers out of the total number available. The location of the Computer Mail Box registers depends on the CPU register memory size. For example, for a CPU with 8K register memory, the Computer Mail Box is located from R8123 through R8192. For a CPU with 16K register memory, the Computer Mail Box is located from R16315 through R16384.

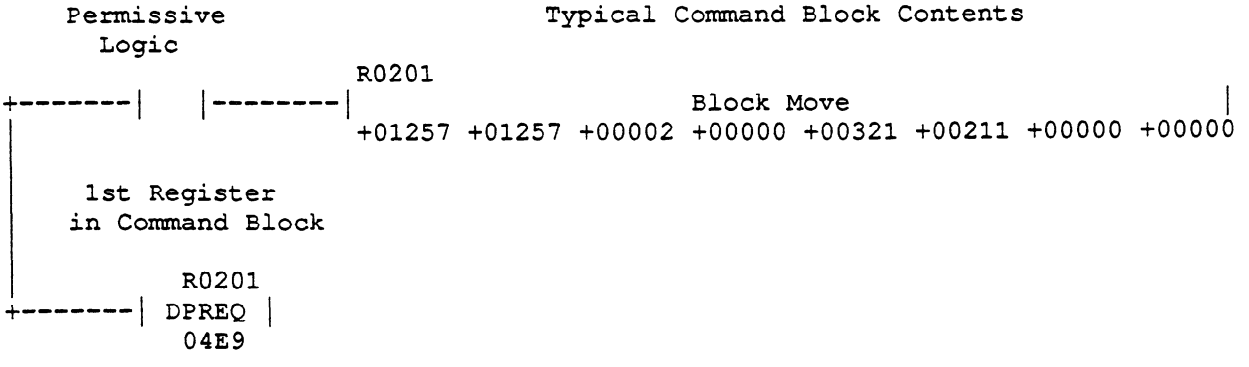
GFK-0171

The Command Block

The DPREQ and WINDOW instructions and the Computer Mailbox are programmed with a group of registers called a Command Block. The Command Block contain the instructions for the data transfer. A typical Command Block has the following content:

Bus Controller Address	Register 1
Command Number	Register 2
Communications Status	Register 3
Target Address	Register 4
Address for data	Register 5
data registers	

Data can be placed in the Command Block registers with one or more Block Moves, or similar program instructions:



The exact content of the Command Block depends upon the type of command it describes, as summarized below. *In this discussion, the term "Register 1" simply means the first register of a group. Register 2 is the second register, and so on.*

Register 1. The first register of the Command Block provides the backplane address of the Bus Controller to which the CPU sends the command.

For a WINDOW instruction, use the Bus Controller's address. For example, 257.

For a DPREQ, use the Bus Controller's address plus 1000. For example, 1257.

For the Computer Mailbox, use (channel number times 1024 plus starting I/O reference). Use the decimal equivalent of the channel number. For example, Bus Controller location 257 on channel E (14 decimal) would be calculated like this:

```
(14x1024)+257 = 14593
```

Register 2. The second register contains the Command Number, which may be one of the following:

1	=	Idle
2	=	Read Configuration
3	=	Write Configuration (also used to start/stop Global Data)
4	=	Read Diagnostics
7	=	Read Analog Inputs
9	=	Read Status Table Reference
11	=	Switch BSM
12	=	Send Datagram: Assign Monitor/Write Device/Write Point
13	=	Receive Datagram: Read Device

Register 3. Status Code. This register should first be cleared to zero by the CPU. It will be loaded by the Bus Controller at the end of the window command (when status is available). Its content may be:

Hex	Decimal		
0000	00000	=	Not accepted, CPU or Bus Controller busy with previous DPREQ.
0001	00001	=	Command in process but not completed.
0002	00002	=	Command completed successfully.
000C	00012	=	Command terminated due to syntax error.
0014	00020	=	Command terminated due to data transfer error.

The Bus Controller verifies the Command Block for valid command syntax and the absence of a command of that type already in execution. If any syntax errors exist, the Bus Controller writes the error code (12) into the Status Code in the third register of the Command Block in the CPU.

Registers 4 - 10. The remaining registers in the Command Block are used for varying purposes by different commands, as indicated later in this chapter. Not all registers are used by all commands.

GFK-0171

Program Structure

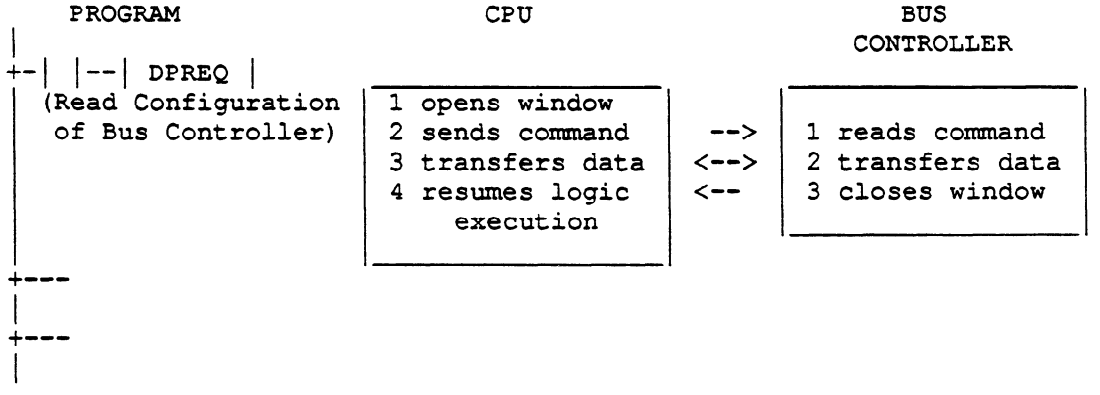
Window instructions can be programmed as separate rungs with conditional logic. It may be preferred to use just one window command and change the content of the Command Block each time. If that method is used, conditional logic should check the current Status Code (in Command Block register 3) before the content of the Command Block is changed.

If the program will also include programmed CPU to CPU communications, there are additional considerations for using window instructions which are explained in chapters 6 and 7.

Timing for Window Commands

The impact of window commands on the CPU sweep time depends on the type of commands used, and the relationship between the CPU sweep and the bus scan.

- *An Immediate* command is started and completed during a single window. The Bus Controller returns data to the CPU in the same instruction in which the command was issued and indicates Done (2) in the Status Code (Command Block register 3).



It is possible for several immediate window commands to execute during the same CPU sweep.

- *A Non-immediate* command is not completed during the same CPU/Bus Controller window. When a Non-immediate command is accepted, the CPU opens a window to the Bus Controller, and sets the Status Code to 1 (In Progress). If the command cannot be executed immediately, the Bus Controller puts the command into a queue and the window closes. While the window is closed, the Bus Controller will not accept any more non-immediate window commands for the Bus Controller.

After the Bus Controller has finished processing the command (for example, sending configuration data to a block or reading configuration data from a block), it changes the status to "complete" *during the next available window* to that Bus Controller. This probably will not occur during the same CPU sweep. In the interim, the status indicates "in progress". If the processing fails to complete, perhaps due to a broken cable, the Bus Controller sets the status to 20, "transfer error".

The table below shows how long the CPU/Bus Controller window may be open for both immediate and non-immediate commands.

Command	Immediate for a Block	Immediate for a Bus Controller	Window Time	
			Best Case	Worst Case
Idle	n/a	yes	1.2mS	1.2mS
Read Configuration	no	yes	2.5mS	4.5mS
Write Configuration	no	yes	2.5mS	4.5mS
Read Diagnostics	no	yes	2.5mS	4.5mS
Read Analog Inputs	yes	n/a	2.5mS	3.5mS
Read Status Table Reference	no*	yes	2.5mS	3.5mS
Switch BSM	no**	n/a	2.5mS	3.5mS
Send Datagram	no**	no**	2.5mS	5.5mS
Receive Datagram	no**	no**	2.5mS	5.5mS

n/a = the command cannot be sent to that type of target device).

* = or to another bus interface module on the same bus.

** = non-immediate if directed to a specific device. If a Datagram command is broadcast to all devices on the bus, it is an immediate command.

When estimating CPU sweep time, add together the times of all windows the CPU might open to Bus Controllers in the system during the same sweep. Remember that a Bus Controller will only accept one non-immediate command at a time. For each Bus Controller, the types of commands used and the relationship of the CPU sweep to the bus scan will determine how many commands are actually executed in one sweep.

In addition to these window times, the sweep time estimate must include the time required by some commands to transfer the requested data to the CPU. Each byte of data will add approximately .031mS to the CPU sweep time. For non-immediate commands this data will be returned during the next available window after receipt of data by the Bus Controller handling the command.

Idle

The Idle command opens a window from the CPU to the Bus Controller, but does not specify a task to be executed. Some program uses for the Idle command are:

1. to delay the CPU sweep.
2. to update I/O points that have been assigned register Reference Numbers.
3. to receive communications from other bus interface modules. See chapter 6 for information.

The Idle command takes approximately 1.2mS to execute *if there is no pending operation* to delay its starting, *or data transfer* to delay the window's closing. The Idle command is immediate; it is always accepted.

Once the Idle command is accepted, if the previous command requested data from a device on the bus, or if an incoming Datagram has been received, the window will remain open while the data is transferred to the CPU. Similarly, the Bus Controller will transfer all Global Data it has received during the previous CPU sweep to the CPU during the first available window in the program. If the Idle command is first in the sweep, Global Data will be transferred at that time, adding .031mS per byte to the CPU sweep.

Command Block for the Idle Command

The Idle command uses only registers 1-3. Command Block format for Idle is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 1
Register 3:	Status Code (supplied by the Bus Controller)

Delaying the CPU Sweep--More than 5 Analog Blocks

The Idle command can be used to delay the CPU sweep at a selected place in the program. For example, a delay might be programmed if there were more than 5 analog blocks on the bus. This delay would allow the Bus Controller enough time to update all analog input values in its shared RAM. When used for this purpose, the Idle command should be located before any I/O update in the program. See chapter 3 for more information.

Updating I/O Points with Register References

For most applications, I/O points are assigned Reference Numbers in I/O memory and are updated during the I/O scan portion of the CPU sweep.

If I/O points have been assigned Reference Numbers which are in register memory instead of I/O memory, the points *will not be updated* during the I/O scan. An Idle command can be used to open a communications window, during which these I/O points will be updated.

Read Configuration

The Read Configuration command is used to request current configuration data from the Bus Controller or a block. It cannot request data from another interface module on the bus, a Hand-held Monitor, or a block that has been assigned a register Reference Number.

If configuration data is requested from the Bus Controller, no message is sent on the bus; the Bus Controller returns its configuration data to the CPU during the same window.

If configuration data is requested from a block, the Bus Controller schedules a Read Configuration message to the block as a "background" task then returns the Status Code 1 (In Process) to the CPU and closes the CPU/Bus Controller window. The Bus Controller accepts no additional non-immediate CPU/Bus Controller window commands from the CPU until the task is completed. With the window closed, program execution resumes. The Bus Controller reads the configuration data from the block in 16-byte increments. When all the data has been received, the Bus Controller transfers it to the CPU at the start of the next available CPU/Bus Controller window and sets the Status Code to 2 (Done).

Command Block for the Read Configuration Command

Command Block format for Read Configuration is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 2
Register 3:	Status Code (supplied by the Bus Controller)
Register 4:	Ten bit beginning address of the device from which configuration data is to be read. If the data is requested from an outputs-only block (not outputs with feedback), bit 15 (MSB) must be set to 1. For an inputs-only block, bit 15 must be 0.
Register 5:	Pointer to the group of registers in the Series Six CPU where the configuration data will be placed after the command executes.

Data Returned by a Read Configuration Command

The content of the reply message depends on the type of device being queried. Read Configuration Reply contents for the Series Six PLC Bus Controller are shown on the following pages. Read Configuration Reply messages for I/O blocks are defined in the *Genius I/O System User's Manual.*

Data Returned by a Read Configuration Command to a Bus Controller

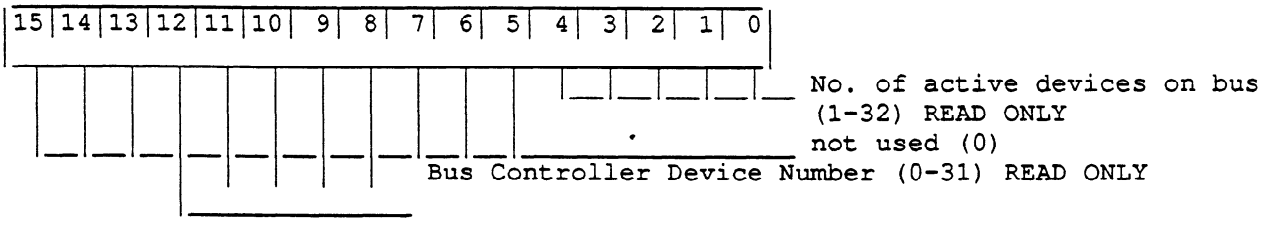
If the command requests the configuration of the Bus Controller, the Bus Controller immediately returns to the CPU its software revision number, the current I/O configuration of the bus, the states of the Outputs Disable bits, and its Global Data (if any) starting address and length. Data format is shown below (register numbers below show relative locations).

REG. #	BYTE	DESCRIPTION	
1	lsb	Bus Controller Type (see Bit Assignments)	READ ONLY
	msb	Software revision number	
2	lsb	bits 0-4: No. of devices on bus (1-32)	
	msb	Bus Controller Device Number (0-31)	
3	lsb	Serial bus baud rate (see Bit Assignments)	
	msb	not used	
4		Bit map of input points 1-128	
5		Bit map of input points 129-256	
6		Bit map of input points 257-384	
7		Bit map of input points 385-512	
8		Bit map of input points 513-640	
9		Bit map of input points 641-768	
10		Bit map of input points 769-896	
11		Bit map of input points 897-1000	
12		Bit map of output points 1-128	
13		Bit map of input points 129-256	
14		Bit map of output points 257-384	
15		Bit map of output points 385-512	
16		Bit map of output points 513-640	
17		Bit map of output points 641-768	
18		Bit map of output points 769-896	
19		Bit map of output points 897-1000	Bus Controller CBB902 or CBB903 only
20		Outputs Disable flags for devices 0-15	
21		Outputs Disable flags for devices 16-31	
22		Global Data starting address	
23		Global Data/Datagram length (in bytes)	

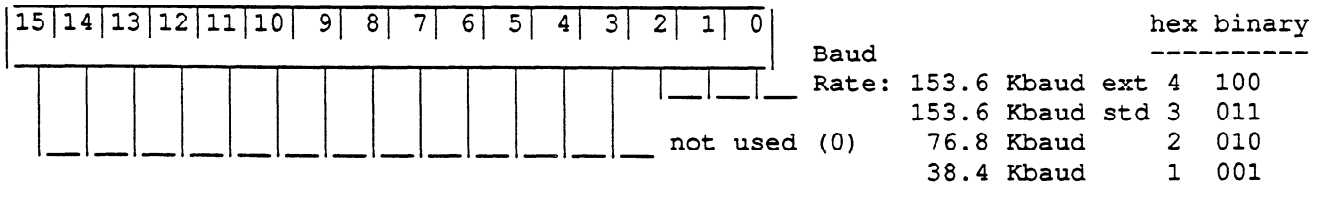
Device Type (lsb of register 1) may be:

		Decimal	Binary
Bus Controller w diagnostics	(IC660CBB900)	1	00000001
Bus Controller w/o diagnostics	(IC660CBB901)	3	00000011
Bus Controller w diagnostics	(IC660CBB902)	6	00000110
Bus Controller w/o diagnostics	(IC660CBB903)	7	00000111

Register 2: Block Configuration Bit Assignments:



Register 3: Baud Rate

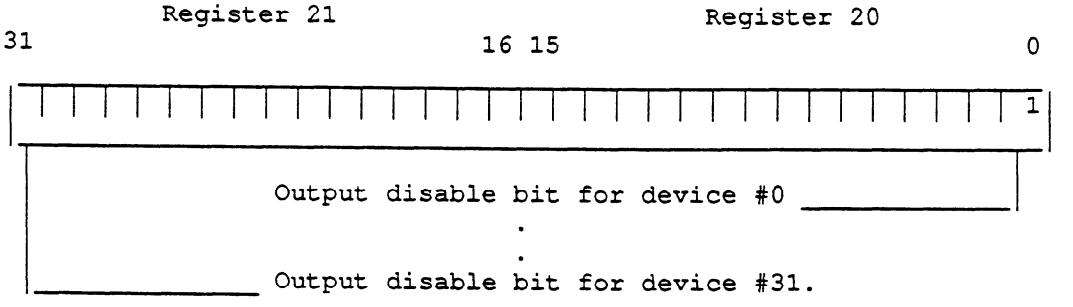


Baud Rate:		hex	binary
	153.6 Kbaud ext	4	100
	153.6 Kbaud std	3	011
	76.8 Kbaud	2	010
	38.4 Kbaud	1	001

Registers 4-19: Bit map of active blocks (read only): A binary "1" in a bit indicates that the corresponding addresses (8 references) are assigned to a block. Some blocks (such as 16 circuit discretes) require more than 1 bit in this map. Registers 4-11 are a bit map of INPUTS assigned to active blocks. Registers 12-19 are a bit map of OUTPUTS assigned to active blocks.

BIT #	Register 4/12	Register 5/13	Register 6/14	Register 7/15	Register 8/16	Register 9/17	Register 10/18	Register 11/19
bit 0	001-008	129-136	257-264	385-392	513-520	641-648	769-776	897-904
bit 1	009-016	137-144	265-272	393-400	521-528	649-656	777-784	905-912
bit 2	017-024	145-152	273-280	401-408	529-536	657-664	785-792	913-920
bit 3	025-032	153-160	281-288	409-416	537-544	665-672	793-800	921-928
bit 4	033-040	161-168	289-296	417-424	545-552	673-680	801-808	929-936
bit 5	041-048	169-176	297-304	425-432	553-560	681-688	809-816	937-944
bit 6	049-056	177-184	305-312	433-440	561-568	689-696	817-824	945-952
bit 7	057-064	185-192	313-320	441-448	569-576	697-704	825-832	953-960
bit 8	065-072	193-200	321-328	449-456	577-584	705-712	833-840	961-968
bit 9	073-080	201-208	329-336	457-464	585-592	713-720	841-848	969-976
bit 10	081-088	209-216	337-344	465-472	593-600	721-728	849-856	977-984
bit 11	089-096	217-224	345-352	473-480	601-608	729-736	857-864	985-992
bit 12	097-104	225-232	353-360	481-488	609-616	737-744	865-872	993-1000
bit 13	105-112	233-240	361-368	489-496	617-624	745-752	873-880	
bit 14	113-120	241-248	369-376	497-504	625-632	753-760	881-888	
bit 15	121-128	249-256	377-384	505-512	633-640	761-768	889-896	

Registers 20 and 21: (read/write) For Bus Controllers IC660CBB902 and 903 only, registers 20 and 21 are used as Output Disable flags, with one bit for each device on the bus. The least significant bit of register 20 represents Device Number 0 and the most significant bit of register 21 represents Device Number 31.



For each bit, a one causes the CPU to read the block's inputs, but not to send outputs. When a bit is 0, outputs are sent to the corresponding Device Number. No more than two CPUs on the same bus should have their outputs enabled to the same blocks. This table can be defaulted to all 0 or all 1 at powerup using the Disable Outputs DIP switch on the Bus Controller (see chapter 2).

Register 22 (Read/write): For Bus Controllers IC660CBB902 and 903 only, register 22 contains the starting register address for the Global Data in the resident CPU. The Bus Controller defaults register 22 to FFFF hexadecimal, indicating no Global Data to be sent. See chapter 7 for information about using Global Data.

Register 23 (Read/write): For Bus Controllers IC660CBB902 and 903 only, register 23 contains the length in bytes (number of registers times 2) to be transmitted from resident CPU to another CPU on the bus using Global Data. Maximum is 128 bytes (64 registers). At powerup, the Bus Controller defaults register 23 to 0.

Write Configuration

The Write Configuration command is used to send configuration data from the CPU to the (phase B) Bus Controller or to a block on the bus. It cannot be sent to another bus interface module or to a block that has been assigned a register Reference Number.

When sent to the Bus Controller, this command is immediate (completed during the Bus Controller window). The Write Configuration command can also be used to:

1. enable or disable outputs from the CPU to devices on the bus.
2. start or suspend Global Data transfer (see chapter 7).

I/O blocks can be configured (or reconfigured) using this command. However, each block must first have its Reference Number and Device Number (serial bus address) entered using the Hand-held Monitor. If configuration data will be sent to a block, the Bus Controller first reads the intended configuration data from the CPU registers during the CPU/Bus Controller window and schedules background Write Configuration messages to the block. The Bus Controller returns the Status Code 1 (In Process) to the CPU then closes the window. The Bus Controller accepts no more non-immediate window commands from the CPU. Once message transmission begins, the Bus Controller sends the configuration data to the block in 16-byte increments. The block does not use any of the new configuration data until it all has been received. No new commands can be sent to the block until the operation has been completed. When all the data has been sent, the Bus Controller changes the Status Code to 2 (Done) at the next available DPREQ or WINDOW instruction, then closes the window.

NOTE

When performing a Write Configuration command to the Bus Controller, pay special attention to the output enable/disable bits to ensure that these are changed only when that is the intent. Outputs should be disabled if there are no blocks on the bus.

Command Block for the Write Configuration Command

Command Block format for the Write Configuration command is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 3
Register 3:	Status Code (supplied by the Bus Controller)
Register 4:	Ten bit beginning reference address of device to which configuration data is to be written. If the data is written to an outputs-only block, you must set the MSB (bit 15) to 1.
Register 5:	Pointer to block of registers where configuration data for the block or Bus Controller starts.

Data Written by the Write Configuration Command

The configuration data to be written must be set up in the registers before the command executes. The data must have the format shown for the Read Configuration command. Also see the *Genius I/O System User's Manual,* "Read Configuration Reply". Items marked Read Only are ignored by the blocks. Changing the register values for one feature (such as disabling outputs) will not change another feature (such as Global Data) if the registers for that feature retain their previously-configured values.

Enabling and Disabling Outputs

When sent to the Bus Controller, the Write Configuration command can be used to enable or disable the sending of output messages from the CPU to some or all of the blocks on the bus. *This applies to input-only blocks too, as they rely on receipt of the null output message to turn their I/O Enabled LEDS on and off.*

Enabling Outputs at Powerup

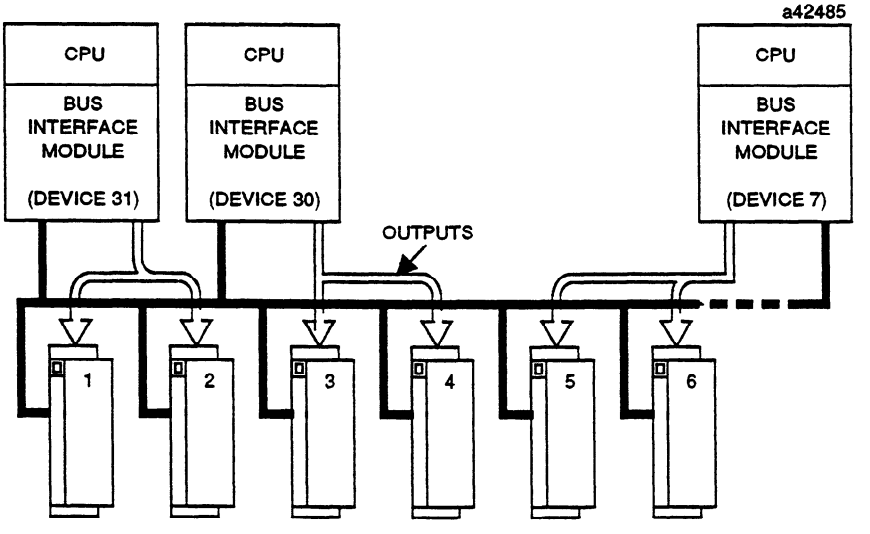
The Bus Controller's on-board DIP switch can be set to disable all outputs at powerup as explained in chapter 2. With program logic, use the Write Configuration command to enable outputs to the blocks which are intended to be under the control of this Bus Controller. This includes input-only blocks.

Disabling Outputs on a Communications Bus

If the bus is used for communications, and has no blocks, outputs should be disabled to conserve scan time.

Selectively Disabling Outputs for Distributed Control of I/O Blocks

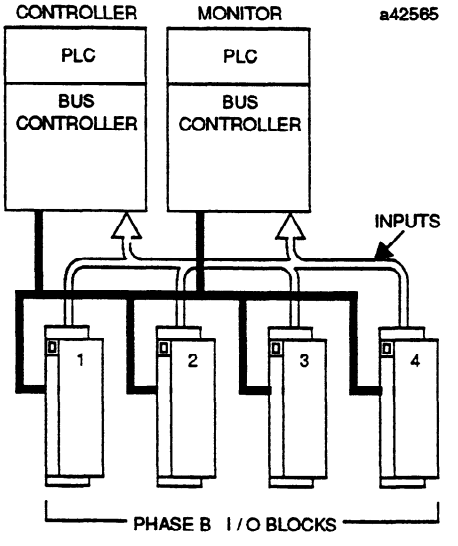
Some systems use two or more CPUs on the same bus for distributed control of I/O blocks. Each CPU sends outputs to (and receives fault reports from) certain blocks on the bus and not others. This is accomplished by selectively enabling or disabling outputs with a Write Configuration command to the Bus Controller.



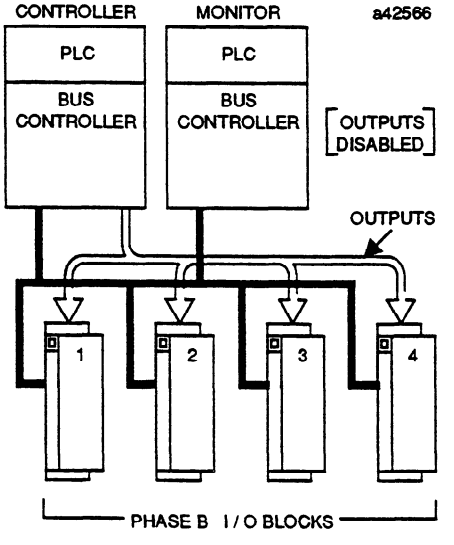
Obtaining Diagnostics from Blocks with Outputs Disabled. Diagnostic messages are automatically sent from a block only to the Bus Controller that is sending or has previously sent it outputs. If the CPU should receive all diagnostics reports from one or more blocks to which outputs are permanently disabled, the Assign Monitor Datagram can be used.

Disabling Outputs for an Assigned Monitor

In some systems, the Series Six PLC will be used as a monitoring device only, to receive I/O data from the blocks on the bus. When being used as a monitor, the PLC can also receive fault reports and configuration change messages from any blocks that have been sent the Assign Monitor command.



Output data for these blocks will be supplied by one or more other CPUs on the same bus.



A device that is assigned as a monitor should have all its outputs disabled. This can be done using a Write Configuration command in the program or by setting the Bus Controller DIP switch. *If a Bus Controller is used as a monitor, it cannot be located in the same CPU as any other Bus Controllers on the same bus. Otherwise, the CPU would receive input data from both Bus Controllers for the same references, and parity errors could result.*

Read Diagnostics

The Read Diagnostics command can be used to request diagnostic information about a block and its I/O circuits, or the status of Bus Controller and the bus. It cannot be sent to a block that has been assigned a register Reference Number.

The diagnostic data returned by blocks in response to this command indicates faults that have occurred since powerup, or since the last Clear Faults message was issued. Current diagnostic state can be found by first issuing a Clear Faults command to the circuit(s) or channel(s) to clear the fault history, then issuing a Read Diagnostics command.

A Read Diagnostics command is completed during the same window only if it is sent to the Bus Controller. If diagnostics data is requested from a block, the Bus Controller schedules a background message to the block and returns the Status Code 1 (In Process) to the CPU. The Bus Controller then closes the CPU/Bus Controller window and accepts no additional non-immediate window commands from the CPU. With the window closed, the Bus Controller reads the diagnostic data from the block in 16-byte increments. When all the data has been received, the Bus Controller transfers it to the CPU at the start of the next available CPU/Bus Controller window and sets the Status Code to 2 (Done) then closes the window.

Command Block for the Read Diagnostics Command

Command Block format for the Read Diagnostics command is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 4
Register 3:	Status Code (supplied by the Bus Controller)
Register 4:	Ten bit beginning reference address of the block or Bus Controller from which diagnostic data is to be read. If the diagnostic data is read from an outputs-only block, set the MSB (bit 15) to 1.
Register 5:	this register contains a pointer to an address where the returned diagnostic data will be stored in the CPU registers.

Data Returned by the Read Diagnostics Command

The exact content of the Read Diagnostics reply message depends on the type of device being queried. Read Diagnostics Reply message contents for the Series Six PLC Bus Controller are shown on the following page. Read Diagnostics Reply messages for I/O blocks are defined in the *Genius I/O System User's Manual.*

The PLC can also obtain diagnostic information from I/O blocks by:

1. enabling the automatic diagnostics features of the Logicmaster 6 software, as described in chapter 4.
2. monitoring the Bus Controller input references, as described in chapter 8.

Data Available using a Read Diagnostics Command to a Bus Controller

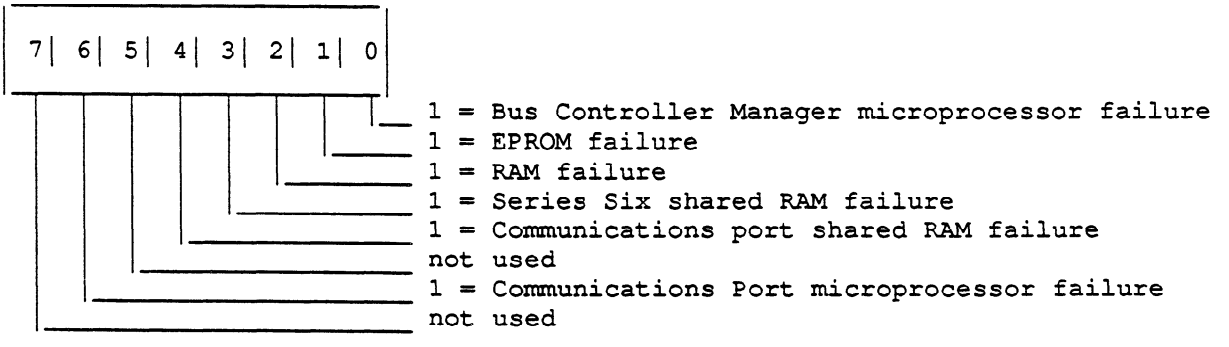
If the Read Diagnostics command is sent to the Bus Controller, the returned data has the following format (register numbers below are relative locations). If the Bus Controller does not have diagnostics capabilities (IC660CBB901 or 903), an error message is returned. This command is immediate when sent to the Bus Controller; the data is returned during the same window.

REG.	BYTE	DESCRIPTION
1	lsb	Device type
	msb	Software revision number
2	lsb	Self-test Diagnostics
	msb	not used, always 0
3		Serial bus error count
4		Serial bus scan time in milliseconds (3-400 decimal)
5		Number of active blocks (0-31)

Device Type (lsb of register #1) may be:

		Decimal	Binary
Bus Controller w diagnostics	(IC660CBB900)	1	00000001
Bus Controller w diagnostics	(IC660CBB902)	6	00000110

Self-test Diagnostics (Register 2) may be:



Error Count (register 3) is a sixteen-bit rollover count of the number of CRC receive errors detected on the serial bus. This count will roll over from 65,535 to 0, and may not be reset.

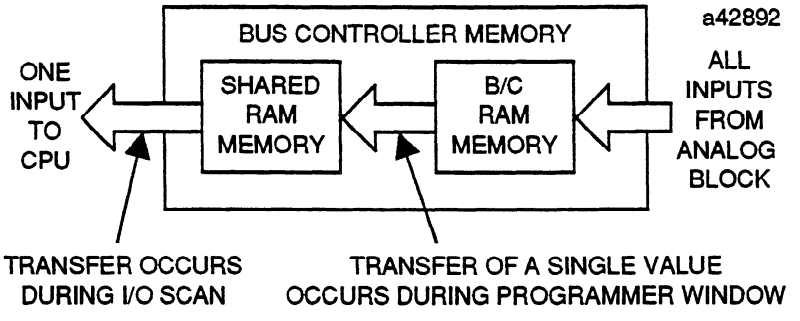
An FFFF value in Register 4 indicates that the scan time has exceeded 400mS which means that the Bus Controller must have missed its turn on the serial bus.

Read Analog Inputs

The Read Analog Inputs command is used to read all of the input values from an analog block, and certain future devices during one CPU sweep. Read Analog Inputs is an immediate command and can therefore always be executed. *Note that this command cannot be sent to a block that has been assigned a register Reference Number.*

How Analog Inputs are Obtained by the CPU

An analog block broadcasts current values for all of its input circuits each bus scan. The Bus Controller receives these values, and stores them in its on-board RAM memory. Once each CPU sweep, (during the *programmer* window), the Bus Controller transfers the latest inputs for a specific analog circuit on the block into the portion of its RAM memory it shares with the CPU. Each sweep, the circuit number changes on a rotating basis.



During the I/O scan portion of the CPU sweep, one input per analog block and its circuit number are picked up by the CPU automatically from the block's assigned input references. On successive CPU sweeps, other inputs from the block overwrite the same input references. As explained in chapter 3, logic must be used to copy the input values into other registers to prevent their being constantly overwritten.

Using Read Analog Inputs

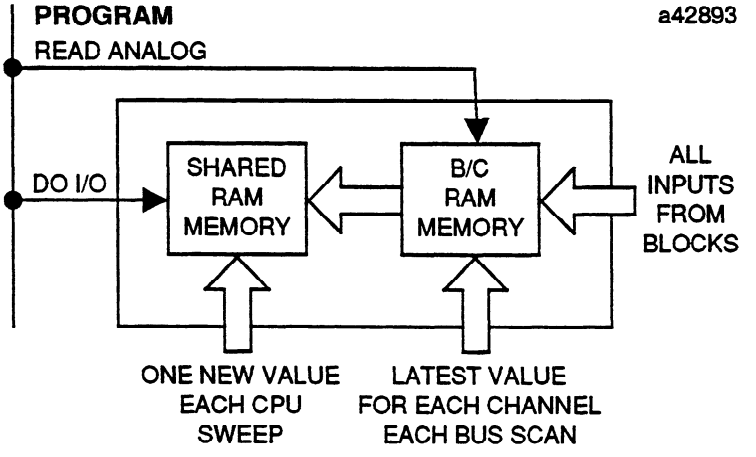
The Read Analog Inputs command reads input values from the "direct access" area of memory in the Bus Controller (not from shared RAM). This area of memory always contains the latest values from all blocks on the bus. Since Read Analog Inputs supplies a register bank into which the analog input values are to be transferred, circuits do not overwrite one another. Only the data values are supplied, one value per register; no circuit number is supplied.

Logic for a Bus with More than 5 Analog Blocks

If there are more than 5 analog blocks, a High-speed Counter, or a Power Monitor Module on the bus, the programmer window may be too brief for the Bus Controller to copy all the input data for those modules from its RAM memory into shared RAM. A Read Analog Inputs command can be used to obtain input values, as described above. See chapter 3 for information about using analog blocks, High-speed Counters and PowerTRAC Modules.

Using DO I/O to Update Analog Inputs

If the program must know the most current values of analog inputs as the logic executes, use a Read Analog Inputs command, not a DO I/O instruction. DO I/O reads input values from shared RAM. Since the Bus Controller updates shared RAM only once per CPU sweep, multiple DO I/O instructions in the same program sweep will return the same values each time.



Example program logic for Read Analog Inputs is shown on the next page.

Command Block for the Read Analog Inputs Command

Format of the Command Block for the Read Analog Inputs command is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 7
Register 3:	Status Code (supplied by the Bus Controller)
Register 4:	Ten-bit Reference Number of the device whose input data will be read.
Register 5:	Pointer to a register address where returned analog data will be placed.

For analog blocks, the Bus Controller provides input values received from the block to the PLC, in the following format (register numbers are relative).

Register 1:	Input channel 1 (16 bit engineering units)
Register 2:	Input channel 2 (16 bit engineering units)
Register 3:	Input channel 3 (16 bit engineering units)
Register 4:	Input channel 4 (16 bit engineering units)
Register 5:	Input channel 5 (16 bit engineering units), RTD only
Register 6:	Input channel 6 (16 bit engineering units), RTD only

Example Program Logic to Read Analog Inputs

The example logic that follows would repeatedly execute a communications command to read analog inputs from a 4 Input/2 Output analog block. The program would store the engineering units values of the analog inputs in four consecutive registers starting at register R0131. Therefore, R0131 would always contain the value of input 1, R0132 would contain the value of input 2, and so on.

```
<< RUNG 1 >>

 ************************************************************************
 * The Block Move instruction below moves seven constants to seven      *
 * consecutive registers starting with R011.  Five of these constants   *
 * represent a READ ANALOG INPUTS command that will read the four analog*
 * inputs from an Analog I/O block with I/O table address 337 and will store *
 * the engineering units values into four consecutive registers starting*
 * at R0131.  O0957 is permissive logic set elsewhere in the program to *
 * control execution of the DPREQ.                                      *
 ************************************************************************

        ANALOG
         DPREQ

 O0957   R0011
+--]/[---[                        BLOCK MOVE                          ]-  ( )
               +01257  +00007  +00000  +00337  +00131  +00000  +00000

 << RUNG 2 >>

 ************************************************************************
 * Other logic in the program tests the state of the Bus Controller OK bit.  *
 * If the Bus Controller is OK, this DPREQ instruction will execute the *
 * READ ANALOG INPUTS command every other CPU sweep.                    *
 ************************************************************************

  BUS
CONTRLR          ANALOG
  OK              DPREQ

 I0257   O0957   R0011
+--] [-----]/[---[DPREQ]-                                               ( )

 << RUNG 3 >>

 ************************************************************************
 * The rung below checks the READ ANALOG INPUTS command block status    *
 * register to determine whether the current status of the command is 2, *
 * indicating successful completion.  If the status is not equal to 2,  *
 * input data may not be current and could be invalid.                  *
 ************************************************************************

                                 ANALOG                                  ANALOG
                ALWAYS  ALWAYS    DPREQ                                IN DATA
                +00002  +00002   STATUS                                 VALID

 Const           R0010   R0010   R0013                                   O0001
+[   A     MOVE      B ]-[   A  :    B ]-----------------------------------( )
 +00002
```

Read Status Table Reference

The Read Status Table Reference command is used to:

1. read the Reference Number of the Bus Controller.
2. read the Reference Number and I/O type of an I/O block.
3. read the Global Data address of another CPU on the same bus.

This command is immediate if sent to the target Bus Controller. It is non-immediate if sent to a block or another bus interface module elsewhere on the bus.

Command Block for the Read Status Table I/O Reference Command

Command Block format for the Read Status Table I/O Reference Command is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 9
Register 3	Status Code (supplied by the Bus Controller)
Register 4:	5-bit Device Number of device to be read (0-31)
Register 5:	Pointer to pair of registers where the I/O reference information from the device is to be written in the CPU.

Data Returned by a Read Status Table I/O Reference Command to an I/O Block

If register 4 contains the Device Number of an I/O block, the following data about the block is returned to the CPU.

Register 1:	10-bit Status Table Reference of block requested above.	
Register 2:		
bits 0-1:	Block I/O Configuration:	1 = Inputs only 2 = Outputs only 3 = I/O combination
bits 2-15:	not used (zero)	

Data Returned by a Read Status Table I/O Reference Command to a Bus Controller

If register 4 contains the Device Number of a Bus Controller or any other Global Data device, the following data about the device is returned to the CPU.

Register 1:	starting address of Global Data for a Global Data device on the same bus. FFFF = no Global Data
Register 2:	3

Any device on the bus which is capable of sending Global Data will return this information; it does not actually have to be using this feature.

GFK-0171

Switch BSM

Use the Switch BSM command to cause a Bus Switching Module to select a bus in a dual bus system. The program must already know which bus is *currently* selected. The CPU may issue the Switch BSM command at intervals to ensure continued proper bus switching capability.

If the command is successful, the CPU will report a Loss of Block diagnostic for the BSM Controller block and for any other block on the same bus stub. If the dual bus system includes a second Bus Controller in the same CPU or another CPU controlling the other bus, that Bus Controller should report an Addition of Block diagnostic for each of those blocks. If the BSM position is currently forced by the Hand-held Monitor, the command will be ignored. It is also ignored if the block does not control a BSM.

Command Block for the Switch BSM Command

Format of the Command Block for the Switch BSM command is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 11
Register 3:	Status Code (supplied by the Bus Controller)
Register 4:	5-bit Device Number of the discrete block that controls the BSM. Be sure the block is the BSM Controller--this command does not check the block's configuration.
Register 5:	Pointer to the register in which the desired BSM position is indicated (see Register n below)
Registers 6-8:	not used

The content of the data sent from the CPU registers to the target device is:

Register n:	BSM position (bit 0): 0 = bus A, 1 = bus B.
	If not 0 or 1, syntax error number 0C hex is returned in the status code.

Chapter 6 Datagrams

This chapter describes:

- Types of Datagrams Supported
- Normal or High Priority Datagrams
- Using Datagrams instead of Global Data
- Programming for Incoming Datagrams
- Effect of Datagrams on the Genius I/O Bus
- Maximum CPU Sweep Time Increase
- Assign Monitor Datagram
- Write Point Datagram
- Write Device Datagram
- Read Device Datagram

Types of Datagrams Supported

The Series Six Bus Controller supports the following Datagrams:

Message Type	Subfunction Code (Hex)
Read Configuration *	02
Read Configuration Reply	03
Write Configuration *	04
Assign Monitor	05
Begin Packet Sequence	06
End Packet Sequence	07
Read Diagnostics *	08
Read Diagnostics Reply	09
Write Point	0B
Report Fault	0F
Pulse Test	10
Pulse Test Complete	11
Clear Circuit Fault	12
Clear All Circuit Faults	13
Switch BSM	1C
Read Device	1E
Read Device Reply	1F
Write Device	20
Configuration Change	22
Read Data	27
Read Data Reply	28
Write Data	29

* Window commands can be used to send Read Configuration, Write Configuration, and Read Diagnostics Datagrams to the Bus Controller or to blocks on the bus which are assigned to I/O memory.
For blocks assigned to register memory, the Send Datagram and Receive Datagram commands (described in chapter 5) can be used to send these Datagrams instead.

Using Datagrams instead of Global Data

Datagrams and Global Data can be used to send messages between CPUs on the same bus. Datagrams are individual messages, while Global Data is transferred automatically and repeatedly.

CPU to CPU Datagrams can be used together with Global Data, or can replace Global Data. Consider using Datagrams instead of Global Data if:

1. Global Data takes up too much serial bus scan time for the application.
2. The data does not need to be sent every serial bus scan.
3. CPU sweep time becomes too long for the application.

In addition, Datagrams can be sent to either I/O or register memory in the receiving CPU, while Global Data must be sent to register memory.

Normal or High Priority Datagrams

The Bus Controller handles Datagram commands in the same way as the other window commands described in chapter 7. When a Datagram command is encountered in the program, the CPU opens a window to the specified Bus Controller. The Bus Controller reads the command, sets the status, then closes the window. The Bus Controller sends the Datagram according to its assigned priority (see below). *Until transmission of the Datagram is complete, the Bus Controller will not accept any other non-immediate CPU/Bus Controller window commands for processing.*

A Bus Controller can send exactly one Datagram per bus scan. That Datagram may be assigned either Normal Priority or High Priority. During one bus scan, there may be one Normal Priority datagram followed by up to 31 High Priority Datagrams, or up to 32 High Priority datagrams sent by the devices on the bus.

If the bus will also be used for I/O block control, Normal Priority datagrams are recommended to allow other messages such as fault reports (which the system handles as Normal-priority Datagrams) to get through. In addition, Normal Priority Datagrams ensure that bus scan time is only modestly delayed for communications. Bus scan time affects the response time of any I/O data on the bus. If there are I/O blocks on the bus, use High Priority only if the Datagram transmission cannot be delayed. Normal Priority will work satisfactorily except when there are many devices attempting to send Datagrams simultaneously.

Programming for Incoming Datagrams

The Bus Controller may receive Write Point, Read Device, or Write Device Datagrams from other interface modules on the bus. The Bus Controller will transfer the received Datagram message to the CPU during the next open CPU/Bus Controller window. To open a window, a DPREQ or WINDOW instruction with the address of that Bus Controller must be present in the program. If the application program does not include any window commands to the Bus Controller, an Idle window command can be used.

It is important to handle incoming Datagrams efficiently, to prevent data loss. Each Bus Controller maintains an area in its RAM memory where it can store 16 incoming Datagrams at the same time. It transfers one to the CPU each time a DPREQ or WINDOW instruction with its address is encountered in the program. If there are not enough window commands directed to a Bus Controller during one CPU sweep, it may accumulate only 16 incoming Datagrams. If that happens, additional Datagrams *will be lost*. This loss will not be detected by the system.

Because only one incoming Datagram can be sent to the CPU during a single window, it may be necessary to place additional window commands in the program if multiple incoming Datagrams are expected. The number of window instructions to a Bus Controller that are needed depends on whether the Datagrams have been sent using Normal or High Priority, and the relative lengths of the CPU sweep time and the scan time of the bus.

If the Bus Scan Time is Greater than the CPU Sweep Time

If all Datagrams on the bus are sent with Normal Priority, there is a limit of one incoming Datagram per CPU sweep. Therefore, only one DPREQ or WINDOW instruction per sweep will be needed to handle the incoming Datagrams.

If all Datagrams on the bus are sent with High Priority, the Bus Controller can receive one Datagram from each transmitting device during each scan. The program should include the same number of DPREQ or WINDOW instructions as incoming Datagrams.

If the Bus Scan Time is Less than the CPU Sweep Time

If the bus scan time is significantly shorter than the CPU sweep time, you can estimate the number of DPREQ or WINDOW instructions that must be sent to the Bus Controller to accommodate incoming Datagrams on that bus.

First, determine how many scans can occur in one CPU sweep. For example, if the bus scan were 20mS and the CPU sweep were 90mS, the ratio between them would be 4.5 to 1. This should be rounded upward to 5.

This is the maximum number of Normal Priority Datagrams that might be received in a single CPU sweep. Plan to have the same number of DPREQ or WINDOW instructions in the program to handle the incoming Datagrams.

For High Priority Datagrams, multiply the number found above by the total number of devices on the bus that might send a High Priority Datagram to the Bus Controller in one bus scan. This is the total number of incoming Datagrams from that bus the program might have to handle in a single CPU sweep. Plan on this number of DPREQs or WINDOW instructions.

Effects of Datagrams on the Genius I/O Bus

Normal Priority Datagrams allow fault reports and Hand-held Monitor communications on a bus to continue undisturbed. Only one Normal Priority Datagram is allowed each bus scan, so the scan time stays relatively constant, and I/O update timing varies only by small increments.

If High Priority Datagrams are being transmitted constantly, the Hand-held Monitor will not function properly; fault reports from blocks will be prevented from being transmitted on the bus, and regular DPREQ or WINDOW instructions (such as Write Configuration commands) to that Bus Controller will fail with a transmission error. For these reasons, use of High Priority Datagrams on a bus with I/O blocks should be avoided if possible.

If High Priority Datagrams are transmitted infrequently, they will cause some delay in the Hand-held Monitor communications and other normal system messages, but the delay should not be noticeable.

High Priority Datagrams will typically put more pressure on the Bus Controller to transfer multiple Datagrams per CPU sweep. However, this can also occur with Normal Priority Datagrams if the bus scan time is much shorter than the CPU sweep time.

Maximum CPU Sweep Time Increase for Datagrams

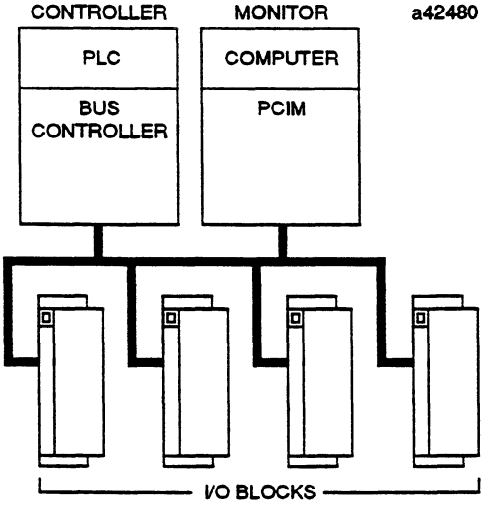
To estimate the impact of Datagrams on CPU sweep time, add together the times required for all Datagrams that might be sent between the Bus Controller and the CPU during one sweep. Repeat this for each Bus Controller in the Series Six PLC that sends or receives Datagrams.

	total Datagram Bytes Sent (may be none)	____ x	.031mS	=	
+	LARGEST incoming Normal Priority Datagram Received, bytes	____ x	.031mS	=	
OR					
+	total incoming High Priority Datagram Bytes Received	____ x	.031mS	=	
+	2.5mS to 5.5mS for each window command used				_____mS
+			1.20 mS	=	_____mS

Assign Monitor Datagram

A Bus Controller or CPU on the bus can be used to monitor block faults and configuration changes. Blocks broadcast their inputs to all devices on the bus; inputs will be received by a monitoring device automatically. In addition, one or more blocks can be instructed to send extra fault and configuration change messages to the assigned monitor. These extra reports will add to the bus scan time, as explained in the *Genius I/O System User's Manual.* There can only be one device assigned as a monitor to any given block.

Send the Assign Monitor Datagram to all blocks that should report faults and configuration changes to the monitor. The message contains the Device Number of the monitor. If necessary, the assigned monitor may be changed by issuing another Assigned Monitor Datagram, with a new Device Number, to the block.



The Assign Monitor Datagram can be sent to phase B Genius I/O blocks only. If sent to phase A blocks or to bus interface modules, the Assign Monitor command has no effect. A complete listing of phase A and phase B Genius devices is located in the *Genius I/O System User's Manual* (GEK-90486).

Command execution is not immediate; the Bus Controller will not set the Status Code to 2 (Done) until it receives an acknowledgement from the block.

Command Block for the Assign Monitor Datagram

Command Block format for the Assign Monitor Datagram is:

Register 1: Bus Controller Reference Number

Register 2: Command Number 12 (Send Datagram)

Register 3: Status code (supplied by the CPU). If the message is sent to one device, Assign Monitor is a non-immediate command; the status register will indicate "done" when that device acknowledges the message. However, if FF is specified in register 4, Assign Monitor is an immediate command; register 3 is set to "done" as soon as the message is sent, with no guarantee that the devices have received the message.

Register 4: the Device Number (serial bus address) of the block which should issue extra fault reports. If all blocks on the bus should issue extra fault reports, enter FF (hex) in this register. If only some blocks should report to the faults to the assigned monitor (for example, to minimize bus scan time), program separate Assign Monitor commands to each one.

Register 5: the location in register memory of the Assign Monitor Datagram (header plus data) to be sent. (See register n, below).

Register 6: Length of message (1 byte).

Register 7: Command code:

Lower byte: 08 (hex) for Assign Monitor Datagram

Upper byte: 20 (hex) for normal priority, or
A0 (hex) for high priority

Content of the Assign Monitor Datagram

Message length for the Assign Monitor message is 1 byte.

Register n: The least significant byte of this reference contains the Device Number of the bus interface module which will receive the fault reports.

GFK-0171

Write Device Datagram

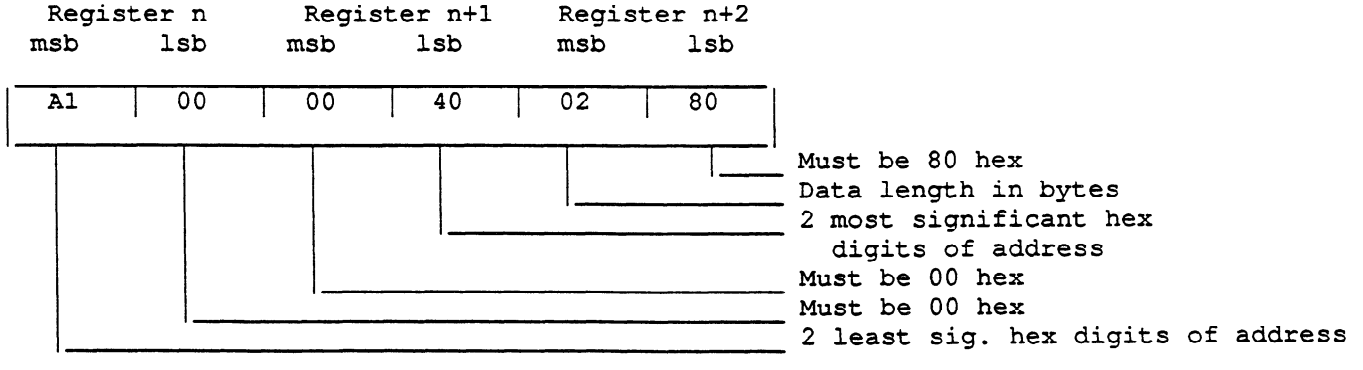
To send up to 128 bytes of register data to register or I/O memory in another CPU on the bus, use the Send Datagram command to send a Write Device Datagram.

Command Block for the Write Device Datagram

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command Number 12 (Send Datagram)
Register 3:	Status code (supplied by the CPU). If the message is sent to one device, it is a non-immediate command; the status register will indicate "done" when that device acknowledges the message. However, if FF is specified in register 4, the command is immediate; register 3 is set to "done" as soon as the message is sent, with no guarantee that the devices have received the message.
Register 4:	Device Number of the bus interface module to which the Datagram will be sent. To broadcast this message to all devices on the bus, enter FF (hex) in this register.
Register 5:	Location in CPU memory of the Write Device Datagram to be sent. (See register n, below).
Register 6:	Length of message in bytes up to 134. This is equal to the number of data bytes (up to 128) plus 6 header bytes.
Register 7:	Command code:
	Lower byte: 20 (hex) = Write Device datagram
	Upper byte: 20 (hex) for normal priority A0 (hex) for high priority

Content of the Write Device Datagram

Register n:	Lower byte must be 00 hex. Upper byte contains 2 least significant hex digits of the absolute memory address (see below) in the destination CPU where the data will be placed.
Register n+1:	Lower byte contains 2 most significant hex digits of the destination CPU absolute memory address. Upper byte must be 00 hex.
Register n+2:	Lower byte must be 80 hex. Upper byte is data length in bytes.
Register n+3:	First register of data (may be up to 64 registers total).



Specifying the Address of the Other CPU

The Command Block for a Write Device, Write Point, or Read Device Datagram must specify the address of the receiving CPU or computer in terms of absolute memory. If the receiving device is a Series Six PLC, a target location in either register memory or I/O Status Table memory can be specified (however, the source of the data in the sending Series Six PLC must always be register memory). For another type of PLC or computer, refer to the documentation supplied with its bus interface module.

CAUTION

Be sure the CPU address specified is for the register table (first hex digit will be 4-7) or the I/O Status Table (first hex digit will be 2). Writing CPU data to any other absolute memory location may cause potentially hazardous control conditions.

SERIES SIX MEMORY TYPE		ABSOLUTE ADDRESS	
		DECIMAL	HEXADECIMAL
I/O Status Table	Outputs	08192 - 08319	2000 - 207F
	Inputs	08320 - 08447	2080 - 20FF
Register Memory	R00001-R16384	16384 - 32767	4000 - 7FFF

The absolute address in decimal for any register is equal to 16383 plus the register number. For example:

Register number (R3000)	3000
Add 16383	+16383
Decimal absolute address	19383

To find the hexadecimal equivalent of this number using the Logicmaster 6 software:

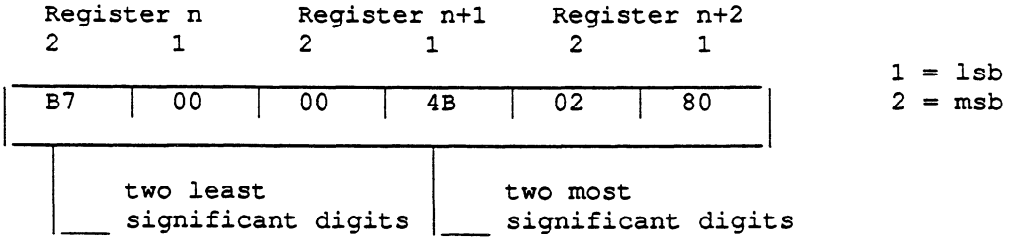
1. When entering the command block, place the work area in decimal format by pressing the Shift and Dec keys. Then, enter the value you want to convert to hex. For example:

   ```
   DEC     19383
   ```

2. Convert the work area to hex format by pressing the Shift and Hex keys. The screen displays the hex equivalent of the number:

   ```
   HEX     4BB7
   ```

Enter this hex value into the registers as explained previously. For Write Device, it would be entered like this:



Example Ladder Logic for a Write Device Datagram

This logic uses a Write Device datagram to send two bytes of data from one CPU to another. The DPREQ contains the address of the resident Bus Controller (513 + 1000 decimal). The other CPU must include logic in its application program to receive the Datagram.

```
 << RUNG 1 >>
*****************************************************************************
* Rung 1 uses a one-shot to initialize the program.  The output O0042 will be used*
* as a permissive in subsequent rungs.  The datagram is only sent once; to send a *
* new datagram, the status byte should be cleared and set to zero.              *
*****************************************************************************
 I1008                                                                  O0042
+--] [-------------------------------------------------------------------(OS)

 << RUNG 2 >>
*****************************************************************************
* Rung 2 uses output O0042 (from rung 1) to initiate loading the Command Block.   *
*        R0290:Bus Controller address +1000 (1513)                            *
*        R0291:Command Number 12 (Send Datagram)                              *
*        R0292:Communications status                                          *
*        R0293:Device Number of second Bus Controller (31)                    *
*        R0294:Pointer to the location of the Write Device datagram to be sent    *
*        R0295:Length of the Write Device datagram (2 data bytes + 6 header bytes)*
*        R0296:Command Code.  Lower byte = 20 hex for Write Device datagram       *
*                             Upper byte = 20 hex for normal priority             *
*****************************************************************************
  O0042    R00290
+--] [---[                          BLOCK MOVE                              ]-  ( )
                  +01513  +00012  +00000  +00031  +00350  +00008  +08224

 << RUNG 3 >>
*****************************************************************************
* Rung 3 contains the Write Device header (R0350-R0352).  First two registers     *
* (R0350 and R0351) contain the absolute memory address of the destination CPU.   *
* Third register is data length in bytes (not counting header), then 80 hex.8     *
* Enter these three registers in hex.  The example hex entries look like this:    *
*                                                                                 *
*                    A100   0040   0280   +00000  +00000  +00000  +00000         *
*                                                                                 *
* Pressing the Accept key converts them to the decimal values shown below.        *
* R0353 contains the actual data to be sent (22222 decimal).                      *
*****************************************************************************
  O0042    R00350
+--] [---[                          BLOCK MOVE                              ]-  ( )
                 -24320  +00064  +00640  +22222  +00000  +00000  +00000

 << RUNG 4 >>
*****************************************************************************
*  DPREQ opens the communications window.  It should be tied to the left rail.    *
*****************************************************************************
 R0290
+[DPREQ]-                                                                    ( )

 << RUNG 5 >>

+[ENDSW]-
```

Write Point Datagram

The Write Point datagram is used to set or reset up to 16 individual bits of data in another CPU. The target address must be specified in terms of absolute memory (see Write Device). Do not send a Write Point datagram to a Series 90-70 PLC. Use a Write Device datagram to bit memory instead.

Setting the Mask Bits

Changes are made to the specified 16 bits by setting the corresponding bits in the AND mask and the OR mask (see above).

A. To set a bit to 0:
   1. set the corresponding AND bit to 0, and
   2. set the corresponding OR bit to 0.

B. To set a bit to 1:
   1. the corresponding AND bit may be either 0 or 1,
   2. the corresponding OR bit MUST be 1.

C. To keep all other bits the same (no change):
   1. set the remaining AND bits to 1, and
   2. set the remaining OR bits to 0.

Example

1010	0000	0101	0000	original data
	1	**0**	**1**	intended bit changes
1111	11**0**1	111**0**	1111	AND mask
0000	00$\underline{\bar{1}}$0	000$\underline{\bar{0}}$	00$\underline{\bar{1}}$0	OR mask

Notice that the AND mask bits for bits 7 and 15 are not the same. When setting a bit to 1, its AND mask bit can be either 0 or 1.

Command Block for the Write Point Datagram

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command Number 12 (Send Datagram)
Register 3:	Status code (supplied by the Bus Controller). If the message is sent to one device, command execution is immediate; the status register will indicate "done" when the device acknowledges the message. However, if FF is specified in register 4, command execution is non-immediate; register 3 is set to "done" as soon as the message is sent, with no guarantee that the devices have received the message.
Register 4:	Device Number (serial bus address) of the bus interface module that will receive the Datagram. To broadcast the message to all devices on the bus, enter FF (hex) in this register.
Register 5:	Location in register memory of the Write Device datagram. (See register n, below).
Register 6:	Length of message (9 bytes).
Register 7:	Command code:

Lower byte:	0B (hex) = Write Point datagram
Upper byte:	20 (hex) for normal priority A0 (hex) for high priority

Content of the Write Point Datagram

Register n:	Lower byte must be 00 hex. Upper byte contains 2 least significant hex digits of the absolute memory address in the destination CPU where the data will be placed. See "Write Device Datagram" for instructions on specifying absolute memory addresses.
Register n+1	Lower byte contains 2 most significant hex digits of the destination CPU absolute memory address. Upper byte must be 00 hex.
Register n+2	Lower byte must be 80 hex. Upper byte: AND mask for bits 0-7. For the AND mask, set to 0 any bits to be changed. Set to 1 all other bits.
Register n+3	Lower byte: OR mask for bits 0-7. For the OR mask, set to the desired state any bits to be changed. Set to 0 all other bits. Lower byte: AND mask for bits 8-15.
Register n+4	Upper byte: OR mask for bits 8-15

Read Device Datagram

To read I/O or register data from another CPU and place it in register memory, use the Receive Datagram command to send a Read Device datagram.

Command Block for the Read Device Datagram

Command Block format for the Read Device Datagram is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command Number 13 (Receive Datagram).
Register 3:	Status code (supplied by the Bus Controller).
Register 4:	Destination Device Number (serial bus address).
Register 5:	Pointer to the first register of the data buffer in the CPU for the Read Device Datagram header. (See register n, below).
Register 6:	Length of the Read Device Datagram header, which is always 6 bytes.
Register 7:	Command code: Lower byte must be 1E hex, upper byte is 20 hex for normal priority or A0 for high priority.
Register 8:	Pointer to the first register of the data buffer in the CPU where the reply message is placed. (See register m, below).
Register 9:	Length of reply message in bytes up to 134. This is equal to the number or data bytes (up to 128) plus header. The Bus Controller supplies this information; it is not necessary to enter a value.
Register 10:	Command code: Lower byte must be 1F hex, upper byte must be 20 hex.

Content of the Receive Datagram Header

When the Read Device Datagram executes, the group of registers beginning at the location specified by register 8 (above) must contain the following information:

Register n	Lower byte must be 00 hex. Upper byte contains 2 least significant hex digits of the source CPU absolute memory address the data will be read from. See "Write Service Datagram" for instructins on specifying absolute memory addresses.
Register n+1	Lower byte contains 2 most significant hex digits of the source CPU absolute memory address. Upper byte must be 00 hex.
Register n+2	Lower byte must be 80 hex. Upper byte is data length in bytes (hex).

Content of the Reply

Data returned by the Read Device Datagram has the following format. The content of the first three registers of the reply is the same as the Receive Datagram header (above).

Register m	Lower byte must be 00 hex. Upper byte contains 2 least significant hex digits of source CPU absolute memory address.
Register m+1	Lower byte contains 2 most significant hex digits of source CPU absolute memory address. Upper byte must be 00 hex.
Register m+2	Lower byte must be 80 hex Upper byte is data length in bytes (hex).
Register m+3	First register of data content, up to 64 registers total.

Example Ladder Logic for the Read Device Datagram

This logic uses a Read Device datagram to read two bytes of data from another CPU. The DPREQ contains the address of the resident Bus Controller (513 + 1000 decimal). The ladder logic for the other CPU must include a communications instruction with the address of its Bus Controller to be able to provide the data requested.

```
+[   Start of Program   ]-
|
|  << RUNG 1 >>
|
|  ************************************************************************
|  * Rung 1 uses a one-shot to initialize the program.  The output O0042 will be used*
|  * as a permissive in subsequent rungs.  The datagram is only sent once; to send a *
|  * new datagram, the status byte should be cleared and set to zero.               *
|***************************************************************************
|
|  I1008                                                              O0042
+--] [-------------------------------------------------------------------(OS)
|
|  << RUNG 2 >>
|
|  ************************************************************************
|  * Rung 2 uses output O0042 (from rung 1) to initiate loading the Command Block.   *
|  *        R0290:Bus Controller address +1000 (1513)                               *
|  *        R0291:Command Number 13 (Receive Datagram)                              *
|  *        R0292:Communications status                                             *
|  *        R0293:Device Number of second Bus Controller (31)                       *
|  *        R0294:Pointer to the location of the Read Device header (R0350)         *
|  *        R0295:Length of the Read Device header in bytes (always 6)              *
|  *        R0296:Command Code.  Lower byte = 1E hex for Read Device datagram        *
|  *                             Upper byte = 20 hex for normal priority            *
|  *                             201E hex = 8222 decimal                            *
|  ************************************************************************
|
|  O0042    R00290
+--] [---[                          BLOCK MOVE                              ]-  ( )
|                  +01513  +00013  +00000  +00031  +00350  +00006  +08222
|
|  << RUNG 3 >>
|
|  ************************************************************************
|  * Rung 3 contains the Read Device header (R0297-R0299).  First register is the    *
|  * address where the received data will be placed.  Enter this in decimal.  The    *
|  * second register is the data length; this value will be supplied by the Bus      *
|  * Controller.  Enter the third Block Move register in hex: 201F.  Pressing the    *
|  * Accept key converts it to the decimal value shown in R0299 below.              *
|  ************************************************************************
|  O0042    R00297
+--] [---[                          BLOCK MOVE                              ]-  ( )
|                  +00350  +00000  +08223  +00000  +00000  +00000  +00000
|
```

```
 << RUNG 4 >>

*****************************************************************************
*  Rung 4 is entered in hex format.  Registers R0350 and R0351 contain the  *
*  destination device absolute address in hex (40 A1 hex = register R0162)  *
*        R0350:  A1 00 hex = -24320 decimal                                 *
*                Lower byte = 00 hex                                        *
*                Upper byte = destination device absolute address byte 1 (LSB *
*                             A1 hex in this example                        *
*        R0351:  00 40 hex = 0064 decimal                                   *
*                Lower byte = destination device absolute address byte 2 (MSB) *
*                             40 hex in this example                        *
*                Upper byte = 00 hex                                        *
*        R0352:  02 80 hex = 0640 decimal                                   *
*                Lower byte = 80 hex                                        *
*                Upper byte = 02 hex, length of data (2 in this example)    *
*  The example hex entries look like this:                                  *
*                                                                           *
*                    A100   0040   0280   +00000  +00000  +00000  +00000    *
*                                                                           *
* Pressing the Accept key converts them to the decimal values shown below.  *
*****************************************************************************

  00042    R00350
+--] [---[                          BLOCK MOVE                          ]-  (  )
                  -24320  +00064  +00640  +00000  +00000  +00000  +00000

 << RUNG 5 >>

*****************************************************************************
*  DPREQ opens the communications window.  It should be tied to the left rail.  *
*****************************************************************************

 R0290
+[DPREQ]-                                                                  (  )

 << RUNG 6 >>

+[ENDSW]-

 << RUNG 7 >>

+[ENDSW]-
```

Chapter 7
Global Data

GFK-0171

This chapter explains how to program Global Data communications between CPUs on the same bus.

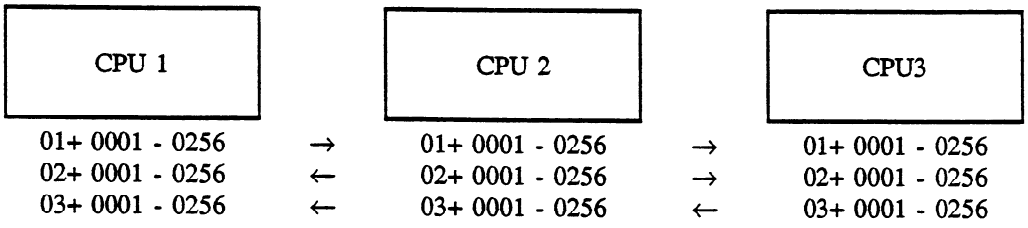
If registers in one CPU are required as data by any other CPUs on the bus, and the data must be constantly updated, use Global Data. Attempting to transfer register data using repeated Datagram messages will have a negative impact on system features, as previously described.

Each interface module can send Global Data to all other interface modules on the bus. A Global Data message may consist of up to 128 bytes of register data. Each interface module will receive all Global Data on the bus.

Global data differs from the Read Device and Write Device Datagrams (which can also be used to send 128 bytes of data) in several ways:

1. Global Data is used for automatic and repeated register data transfer; Datagrams are used to send specific messages.
2. After being initialized in the program, Global Data will be sent by the Bus Controller repeatedly, with minimal additional programming (Idle command) required. Each Datagram requires a DPREQ or WINDOW instruction both to send and to receive, and status must be monitored.
3. The Bus Controller sends all the Global Data it has received to the CPU once each CPU sweep. This requires only one DPREQ or WINDOW instruction to the Bus Controller. For Datagrams, each message received requires a separate window command to transfer the embedded data to the CPU.
4. Global Data cannot be read from or written to I/O memory (Datagrams can).

A Global Data message is always associated with a specified memory address in the sending CPU. If the receiving device is a Bus Controller in a Series Six PLC, the data is placed into the same registers it occupied in the sending CPU. (For this reason, Global Data *cannot be sent by two* or more Bus Controllers in the same Series Six PLC. The second Bus Controller in the PLC would always write Global Data received from the first into the same registers it was sent from, so the data in those registers would never change.) In this example, there are three Series Six CPUs on the same bus. Each CPU sends 16 registers of Global Data to both of the other CPUs.



If the receiving device is a bus interface module in another type of CPU, the data may be stored differently.

Programming to Send Global Data

If the Bus Controller will send Global data, use a Write Configuration command to the Bus Controller. Contents of the Command Block are shown below. For more information about using the Write Configuration command, see chapter 5.

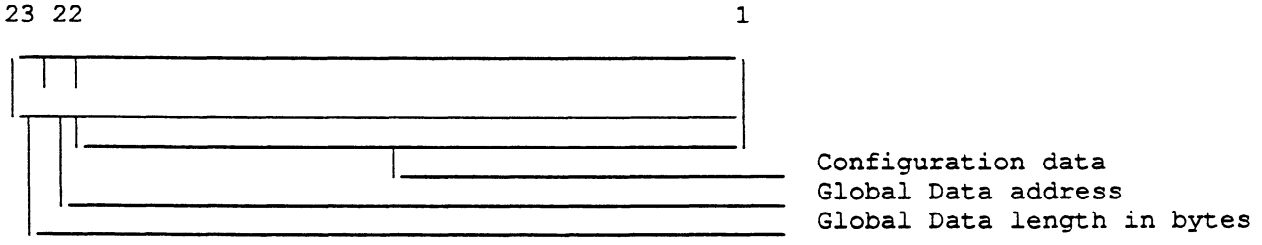
Global Data transmission will start, stop, or change 1.5 seconds after the command. During this 1.5 seconds, the Bus Controller's second LED goes off, and no outputs are updated. This occurs only when there is a Global Data command, not during routine transmission of Global Data.

Command Block for the Write Configuration Command to Send Global Data

The Command Block format for Global Data is:

Register 1:	Bus Controller Reference Number (plus 1000 for a DPREQ)
Register 2:	Command number 3
Register 3:	Status Code (supplied by the CPU)
Register 4:	Ten bit Reference Number of the resident Bus Controller (1-1024).
Register 5:	Pointer to the first register where the configuration data for Bus Controller begins.

The CPU location and length of data to be transferred using Global Data are determined by the content of the last two registers of the Bus Controller configuration table (stored in the Bus Controller).



Registers 1-21 contain configuration data, as described earlier (see "Data Returned by a Read Configuration Command to a Bus Controller" in chapter 5). Register 22 must contain the Global Data address.

The address specified is the same for all the sending and receiving CPUs. At powerup, the Bus Controller defaults register 22 to FFFF hexadecimal, which indicates no Global Data to be sent.

Register 23 should contain the number of bytes up to 128 (registers times 2) to be broadcast by the Bus Controller. At powerup, the Bus Controller defaults register 23 to 0.

Starting Global Data

To start Global Data, the beginning Global Data address and the message length should be loaded into registers 22 and 23, respectively, of the Bus Controller configuration table. The ladder logic for Global Data only needs to be executed once. After that, Global Data will be transferred between CPUs automatically and repetitively unless stopped, as described below.

Stopping Global Data

Global Data can be stopped by loading register 22 in the data buffer (R0322 in the example that follows) with the hexadecimal value FFFF and resetting the status register to 00. The status register (third register in the Command Block) should always be cleared before executing any new communications command.

GFK-0171

Example Ladder Logic for Global Data

The ladder logic that follows is an example of programming Global Data. This program initiates Global Data transfer of two bytes of data. The Global Data address for both CPUs is R0001.

```
+[   Start of Program    ]-
|
|  << RUNG 1 >>
|
|  ********************************************************************************
|  * Rung 1 uses a one-shot to create permissive logic used to load the Command   *
|  * Block and data registers in subsequent rungs.                                *
|  ********************************************************************************
|
|  I1008                                                                  O0042
+--] [-----------------------------------------------------------------------(OS)
|
|  << RUNG 2 >>
|
|  ********************************************************************************
|  * Rung 2 contains the Write Configuration command block:                       *
|  *              R0290 = Bus Controller I/O reference (+ 1000 decimal)           *
|  *              R0291 = Command Number (3)                                      *
|  *              R0292 = Communications status (supplied by CPU)                 *
|  *              R0293 = I/O reference of the Bus Controller. This is the same as *
|  *                      R0290 above, except that 1000 is not added for a DPREQ. *
|  *              R0294 = Pointer to the first register of the data storage       *
|  *                      buffer in the CPU (in this example, R0301).             *
|  ********************************************************************************
|
|   O0042    R00290
+--] [---[                          BLOCK MOVE                              ]-  (  )
|                   +01513  +00003  +00000  +00513  +00301  +00000  +00000
|
```

```
<< RUNG 3 >>

*****************************************************************************
* Rung 3 contains registers 22 (R0322) and 23 of the Bus Controller configuration*
* table.  Register R0322 contains the Global Data address (R00001).  R0323       *
* contains the number of data bytes to be transferred (2).                       *
*****************************************************************************

 00042    R00322
+--] [---[                          BLOCK MOVE                              ]-  (  )
                  +00001  +00002  +00000  +00000  +00000  +00000  +00000

<< RUNG 4 >>

*****************************************************************************
* Rung 4 can be used to stop the Global Data transfer.  If I0009 passes power to *
* the right, the value FFFF hex will be moved into R0322 beginning Global Data   *
* address register. Then the value 00 is moved into R0292 communications status. *
*****************************************************************************

I0009    Const          R00322          Const          R00292
+--] [----[   A    MOVE    B   ]-------[   A    MOVE    B   ]-                ( )
           FFFF                         0000

<< RUNG 5 >>

*****************************************************************************
* DPREQ instruction opens the communications window.  It should be tied          *
* to left rail.                                                                  *
*****************************************************************************

R0290
+[DPREQ]-                                                                    (  )

<< RUNG 6 >>

*****************************************************************************
* Rungs 6 and 7 implement the increment counter in R0001, which is transmitted   *
* on the bus as Global Data.                                                     *
*****************************************************************************

 Const                  R00288
+[   A      MOVE     B   ]-                                                  (  )
 +0001

<< RUNG 7 >>

R0001    R0288    R0001
+[   A   +    B  =    C  ]-                                                  (  )

<< RUNG 8 >>

+[ENDSW]-

<< RUNG 9 >>

+[ENDSW]-
```

Programming to Receive Global Data

For the Series Six PLC, all incoming Global Data is transferred from the Bus Controller to the CPU during the first open window to the Bus Controller that occurs during the CPU sweep. If the application program does not contain any commands to open a window between the CPU and the Bus Controller, a window must be opened. An Idle command can be used.

Be sure registers used for both outgoing and incoming Global Data are not assigned to any other use in the program, even if the CPU will not make use of Global Data it receives. If another device on the bus sends Global Data, it will always be received if the program opens a window to the Bus Controller (using a DPREQ or WINDOW instruction or the Computer Mailbox).

Maximum CPU Sweep Time Increase for Global Data

The Bus Controller will send all incoming Global Data to the CPU each sweep if the application program opens a window to the Bus Controller. If there are active window commands to the Bus Controller, there is no way for the Bus Controller to receive only part of the Global Data on the bus. It is possible to keep a CPU from receiving Global Data by completing all communications tasks during the startup period, then disabling the window commands during system operation. Datagrams may be preferable to Global Data in applications where the CPUs do not require all of the register message data on the bus.

The impact of incoming and outgoing Global Data on the CPU sweep can be estimated by adding all of the Global Data bytes sent and received for each bus in the Series Six PLC system:

FOR EACH BUS:					
	total Global Data Bytes Sent	x	.031mS	=	
+	total Global Data Bytes Received	x	.031mS	=	
+	total number of Global Data messages	x	.050mS	=	
+			1.20 mS	=	_____
	GLOBAL DATA			=	mS

Using Global Data to Check CPU Operation

Global Data is sometimes used to set up a "heartbeat" between two CPUs on the bus, enabling each to check the operation of the other. For this technique to work successfully, the sweep times of both CPUs must be similar. In addition, these CPU sweep times must be approximately twice as long as the scan time of the bus that connects the CPUs.

Chapter 8

8-1 Programming for Diagnostics Using Bus Controller Input References

GFK-0171

This chapter shows the format of the diagnostic data provided by the Bus Controller. You will not need this information if you are using the automatic diagnostics features of the Logicmaster 6 software. If the Logicmaster 6 software Expanded Functions are enabled as described in the chapter 4, the CPU can access this data automatically, and lengthy programming can be avoided.

The following diagnostic information is available from the Bus Controller:

Bus Controller OK	(all Bus Controllers)
Serial Bus Error	(all Bus Controllers)
I/O Circuit Fault	(Bus Controller with Diagnostics)
Loss or Addition of Block	(Bus Controller with Diagnostics)
Address Conflict	(Bus Controller with Diagnostics)
Pulse Test Active	(Bus Controller with Diagnostics)
HHM Force Active	(Bus Controller with Diagnostics)

Program logic can be used to monitor this information. In addition, the program can use the Bus Controller's output references to clear faults or to disable outputs (see chapter 9).

Additional Programming for Diagnostics

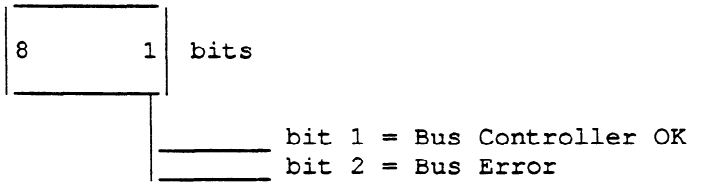
In addition to monitoring the Bus Controller data described in this chapter, the ladder logic can obtain diagnostics data directly from devices on the bus using the Read Diagnostics command. See chapter 5 for more information.

Bus Controller Input Reference Formats

A Bus Controller provides diagnostics data to the CPU in its assigned input references. The format of data in these references depends on whether the Bus Controller has diagnostics capabilities. Each diagnostic is available for exactly one CPU sweep.

Bus Controller without Diagnostics

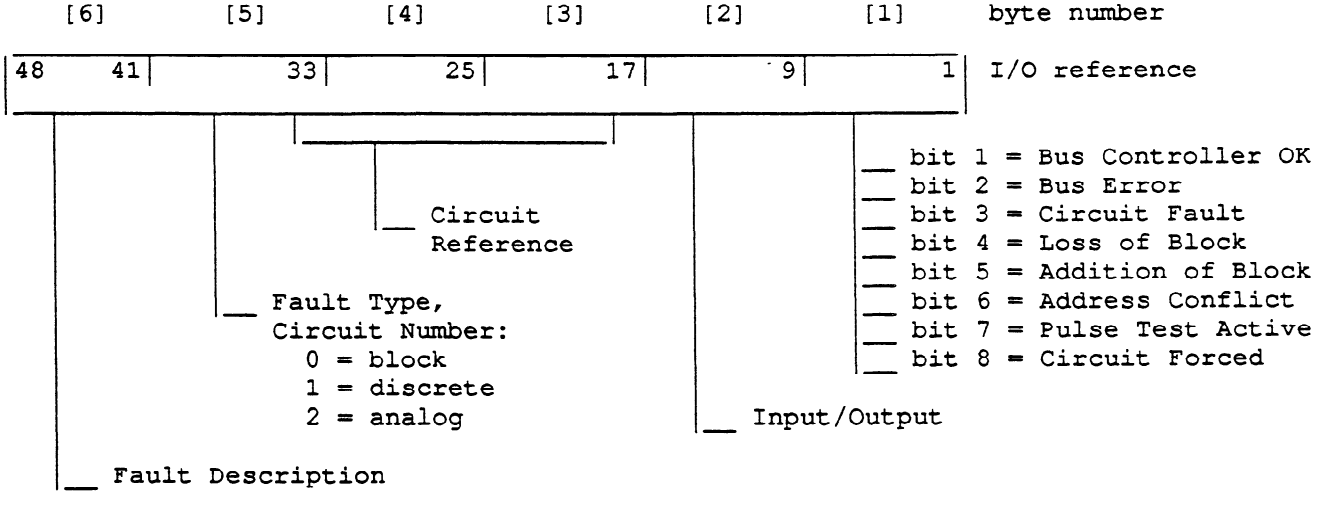
A Bus Controller without diagnostics (IC660CBB903) uses eight input references (one byte) to report its status and the status of the serial bus to the CPU once each CPU sweep.



Of these assigned references, bits 3-8 are not used

Bus Controller with Diagnostics

A Bus Controller with diagnostics (IC660CBB902) uses 48 input references (six bytes). The Bus Controller places fresh diagnostic data in these references every CPU sweep.



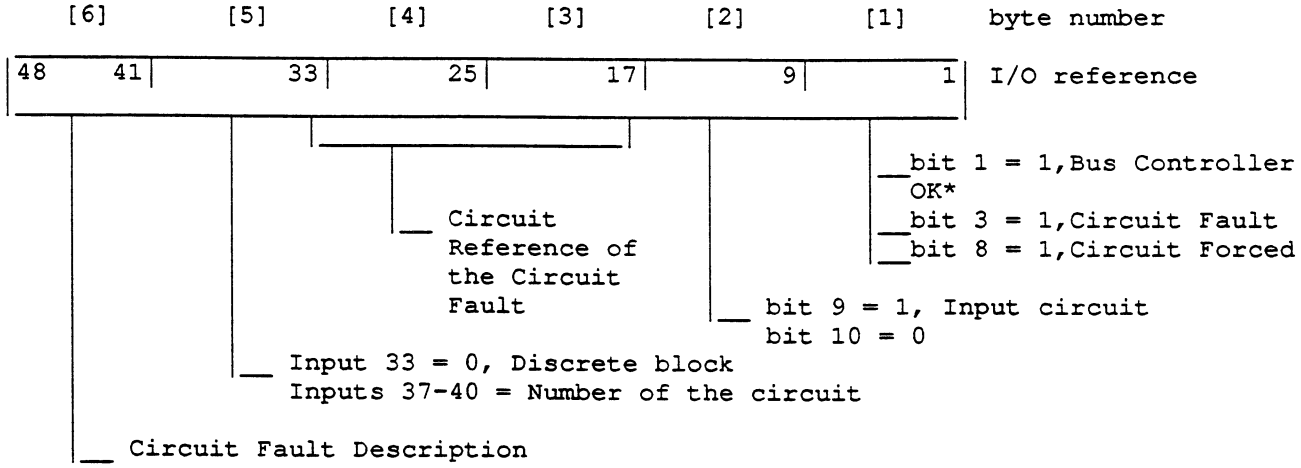
The program can monitor this data as explained below.

GFK-0171

Logic to Monitor Diagnostic Information

The ladder logic can monitor the Bus Controller input references for faults, and use Source-To-Table Moves or other program instructions to capture the data. A fault or system change is sent to the CPU for one sweep only. The Bus Controller buffers up to 60 faults and sends one at a time to the CPU every CPU sweep. Thus, input status bits change each CPU sweep when faults exist.

To use this diagnostic data, the program should look first at bits 1-8 of the Bus Controller input references. Bits 9-48 have meaning only if bit 3, 4, 5, or 6 is set to 1. Bits 3-6 can be 1 for only one sweep of the CPU, and only one bit between 3 and 6 can be 1 at a time. Bits 1, 2, 7, and 8 are independent. They can be 1 or 0 regardless of the state of the other inputs. For example:

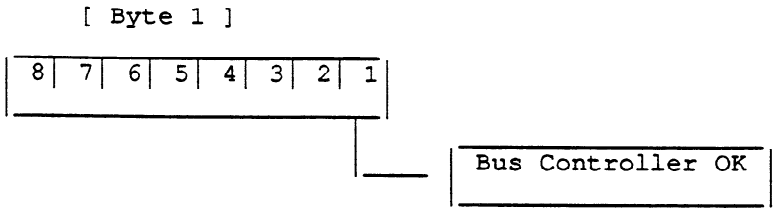


* Automatically cleared by the CPU if DIAGNOSTICS ENABLED is set to Y on the CPU Configuration Setup Menu.

In this example, three bits in status byte 1 are currently 1. They indicate that the Bus Controller is communicating with the CPU, at least one circuit is forced, and a circuit fault is being reported during this scan. Status bytes 2 through 6 describe the circuit fault.

Monitoring Bus Controller Status

Bus Controller input reference bit 1 (the least significant bit) indicates the status of the Bus Controller. It is set ON within one second of power-up if the Bus Controller passes its self-test.



If this bit fails to become 1 at powerup and remain 1 with power applied, the Bus Controller may need to be replaced. If bit 1 is 0, none of the other Bus Controller input reference bits have meaning.

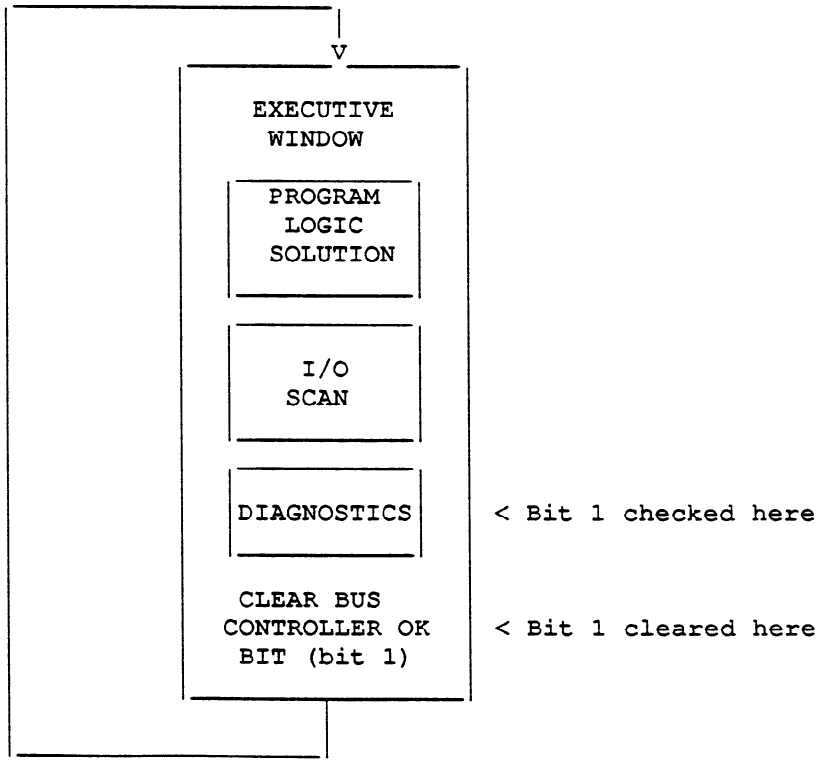
The way the program can monitor the Bus Controller status data will depend on whether DIAGNOSTICS ENABLED has been set to Y on the CPU Configuration Setup Menu, and whether the Bus Controller is included within the range of I/O specified for diagnostics scanning.

Monitoring Bus Controller Status: Diagnostics is NOT Enabled

If the Bus Controller is not set up for automatic diagnostics as defined above, the ladder logic can test bit 1 once every CPU sweep then clear it using a Bit Clear instruction.

Monitoring Bus Controller Status: Diagnostics IS Enabled

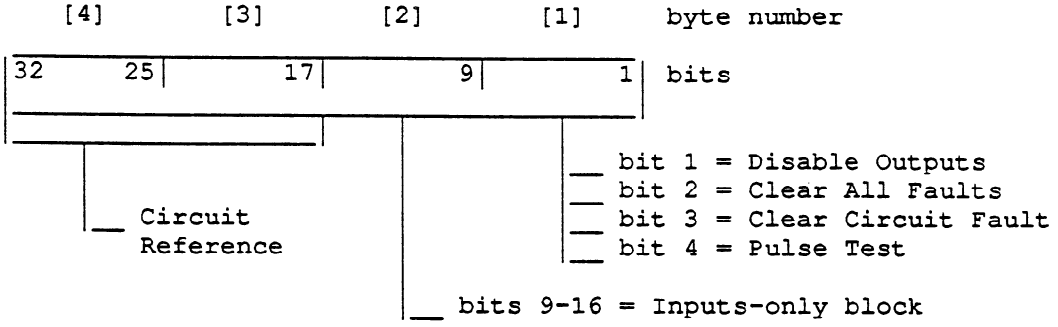
If the Bus Controller is set up for automatic diagnostics, the CPU automatically checks bit 1 during the diagnostics portion of the sweep.



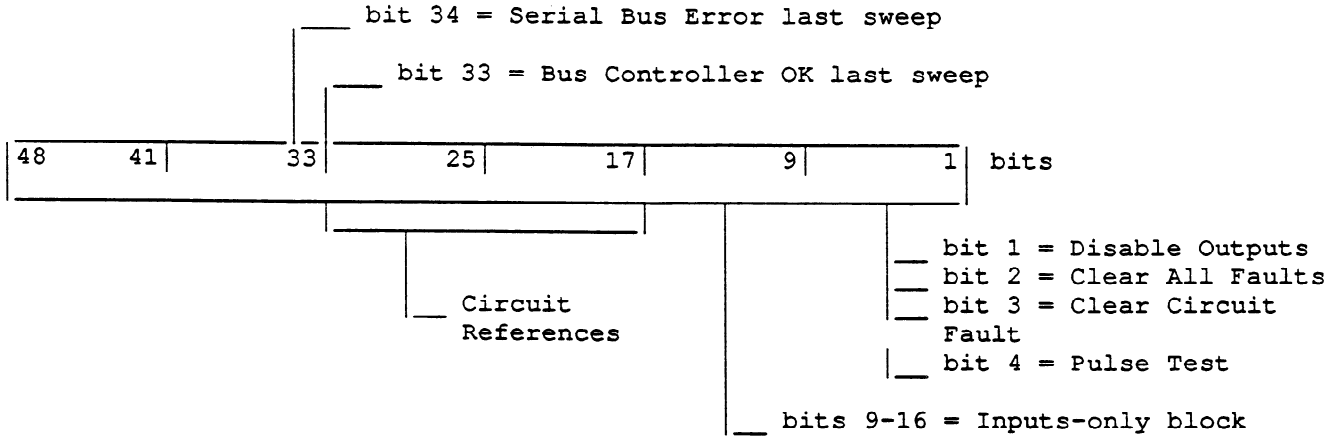
If bit 1 is 1 (indicating that the Bus Controller has passed its self-test), the CPU *automatically clears it* at the end of the CPU sweep. This sequence of diagnostics and bit clearing by the CPU means that bit 1 will always appear to be 0 during the program logic solution part of the sweep. Therefore, if automatic diagnostics are enabled for the CPU, program logic cannot monitor input reference bit 1 to determine the status of the Bus Controller. However, the program can monitor the status of Bus Controller output bit 33.

Monitoring Bus Controller Status using the Bus Controller OUTPUT Bit 33

A Bus Controller with Diagnostics uses 48 references in the output table. Of these 48 references, bits 1 through 32 are used to send commands to I/O blocks:



If DIAGNOSTICS ENABLED is set to Y on the CPU Configuration Setup Menu, the CPU also reserves Bus Controller output reference bits 33 through 48 for diagnostics information. Of these additional 16 bits, only bit 33 and bit 34 are used:

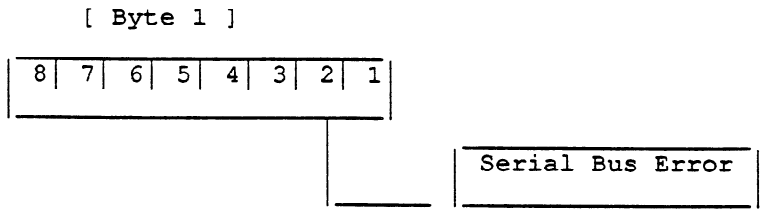


Before clearing Bus Controller input bit 1, the CPU copies it into Bus Controller output bit 33. On the next sweep, if the Bus Controller fails to reset input reference bit 1 to 1, the CPU looks at output reference bit 33. If bit 33 was 1 on the previous sweep, a Bus Controller fault is triggered. If it was 0 on the previous sweep, no additional fault occurs.

Because bit 33 is an accurate reflection of the Bus Controller state (on the previous CPU sweep), the program can monitor Bus Controller status by reading this bit. The program should NOT clear output bit 33.

Checking for Bus Errors

To have the program detect errors such as an open bus or excessive noise in the bus, use logic to monitor bit 2 of the Bus Controller input references.



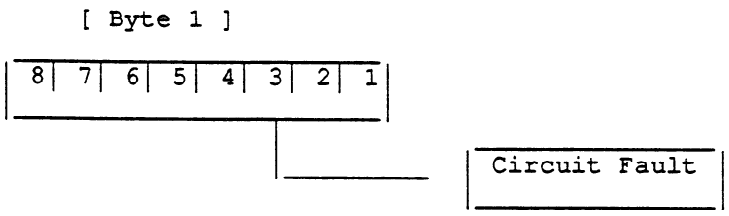
This bit is normally set to 0; it is set to 1 if:

1. the Bus Controller receives ten or more corrupted messages within a 10-second time period. It will be set to 0 again when the error rate falls below ten errors in a 10-second period. If bit 2 is continuously set to 1, there are multiple errors from one or more devices. The bus may continue to operate in spite of these errors.
2. the Bus Controller does not obtain its turn on the serial bus at least once every 500mS. This bit is 1 for at least one scan. It is set to 0 if the Bus Controller is able to transmit data on two successive scans.

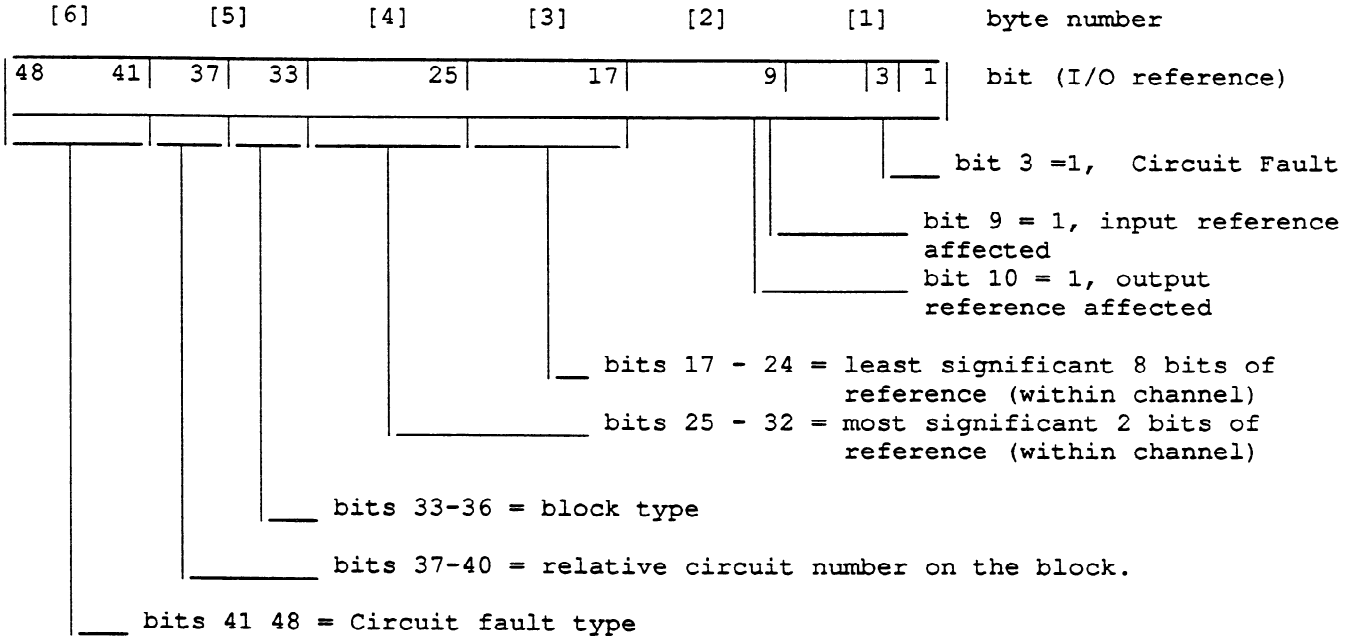
If bit 2 is equal to 1 for several scans, it means the Bus Controller cannot access the bus due to duplicate Device Number assignment, or bus scan greater than 500mS.

Detecting I/O Circuit Faults

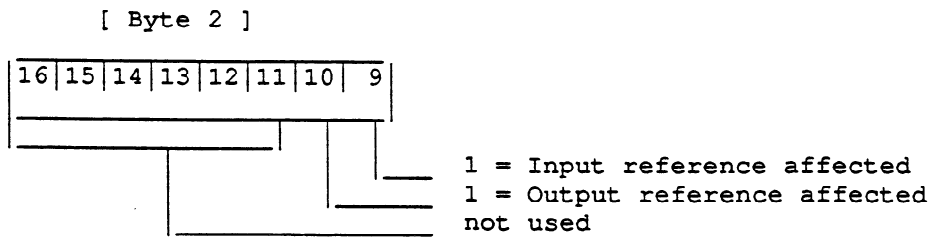
A Bus Controller with Diagnostics uses input reference bit 3 to report circuit faults. *Note that circuit diagnostics are obtained from devices on the bus that have been assigned Reference Numbers in I/O memory. Automatic diagnostics are NOT performed on devices assigned Reference Numbers in register memory. See chapter 3 for more information.*



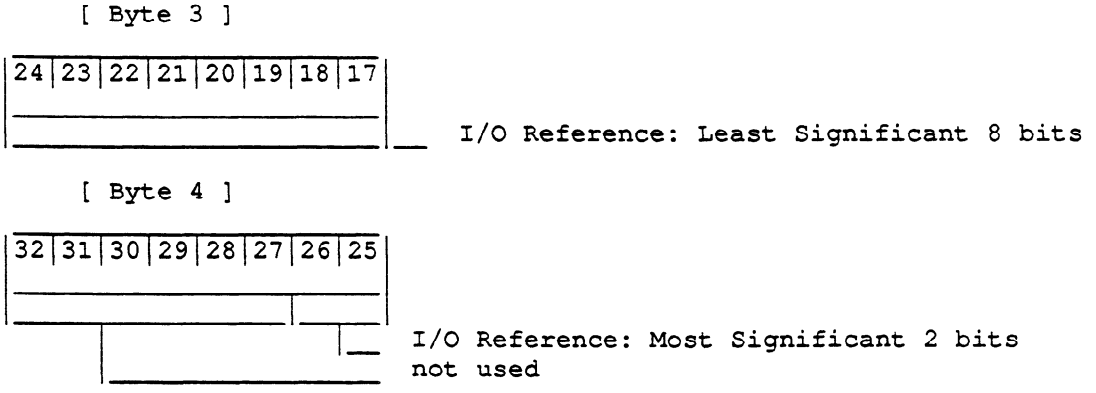
For each fault reported, bit 3 is set to 1 for one sweep . When this bit is 1, the other Bus Controller input references contain the following information:



Bits 9 through 16 indicate whether input or output references (or both) are affected. If bit 9 is 1, the circuit is an input. If bit 10 is 1, the circuit is an output. If both are 1, the circuit is an output with feedback, or the fault affects an entire block which is configured as a combination block.



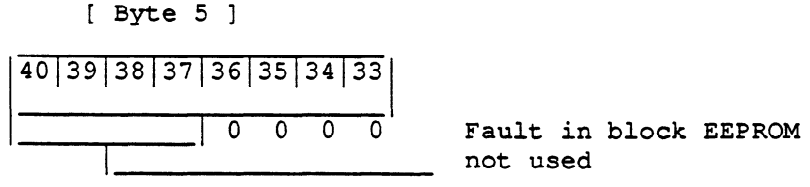
If the fault is a circuit fault, bytes 3 and 4 of the Bus Controller input references contain the I/O reference of the circuit (1-1000) within the channel.



For an analog or RTD block, the circuit is identified by the first six references assigned to the block. The following example shows circuit identification for a 4 Input/2 Output Analog block that starts at I/O references I0801 and O0801):

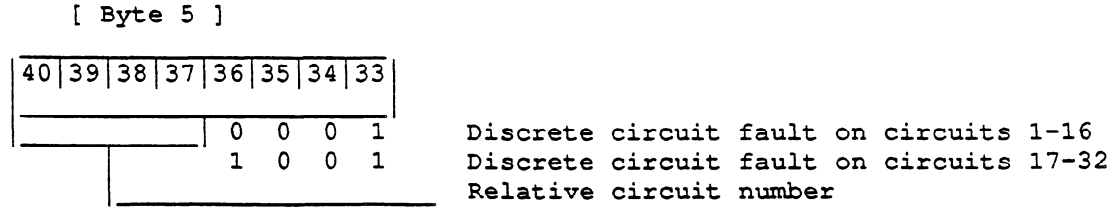
Analog Circuit	I/O Reference	Value
Input circuit 1	Starting reference	0801
Input circuit 2	Starting reference plus 1	0802
Input circuit 3	Starting reference plus 2	0803
Input circuit 4	Starting reference plus 3	0804
Output circuit 1	Starting reference plus 4	0805
Output circuit 2	Starting reference plus 5	0806

Byte 5 contains the block type and relative circuit number of the circuit where the fault has occurred. Content of bytes 5 and 6 depends on the block type specified in bits 33-36. If bits 33-36 are all 0, the fault is in the EEPROM in the block's Terminal Assembly.



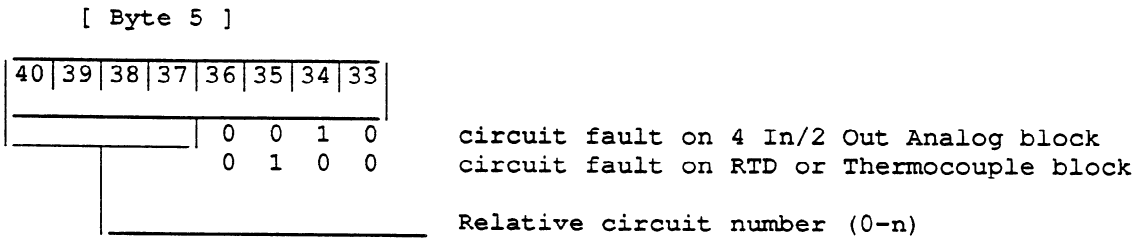
For a discrete block: bits 33-36 will be: 0 0 0 1 or 1 0 0 1.

If a fault has occurred on one of the upper sixteen circuits of a 32-point block, bits 33-36 are set to: 1 0 0 1. To find the correct relative number of the circuit, add 16 to the value in bits 37-40 (see below).



Bits 37-40 contain the relative number from 0 to 15 of the circuit where the fault occurred. The value zero represents the topmost circuit on the block. The higher value is the bottom circuit on the block (for example, 7 for a 8-circuit block).

For analog blocks: Bits 33-36 indicate the block type. Bits 37-40 indicate the relative circuit number (0 to n for inputs, 0 to n for outputs) where the fault occurred.

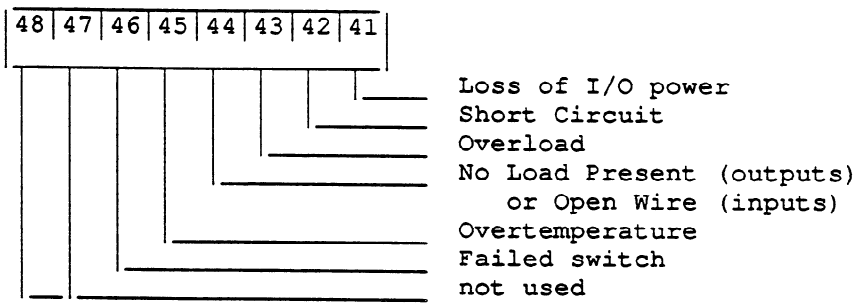


For the 4 Input/2 Output Analog block, bits 9 and 10 indicate whether the circuit is an input or output. Input circuits are numbered 0-3. Output circuits are numbered 0 and 1.

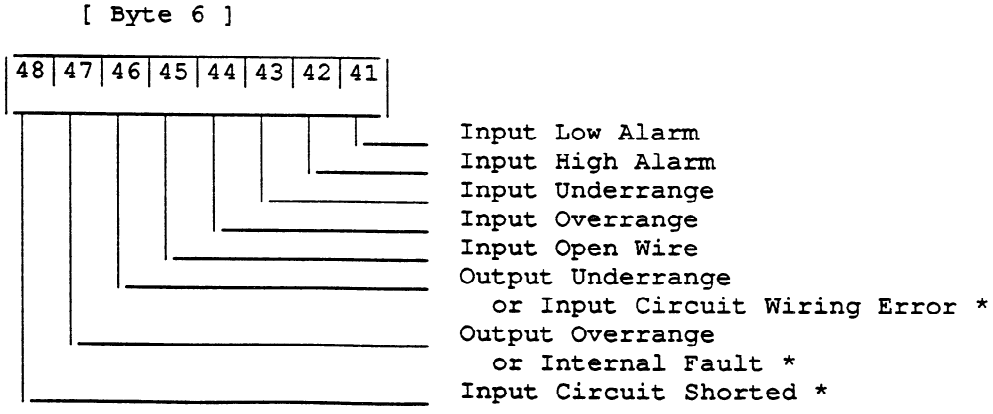
Fault Type

If the fault is a circuit fault, bits 41 to 48 of the Bus Controller input references identify the fault type. The meaning of these bits depends on whether the block is a discrete or analog block.

Fault type bits for a discrete block:



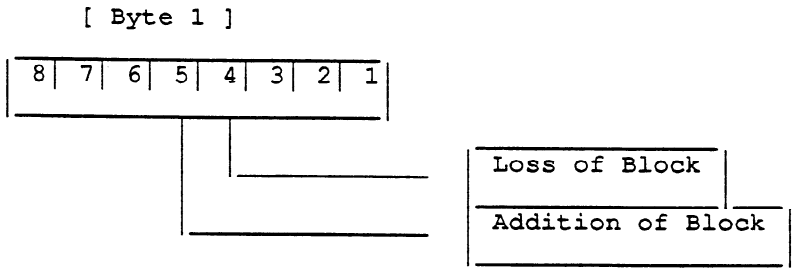
Fault type bits for an analog block:



* for RTD blocks only

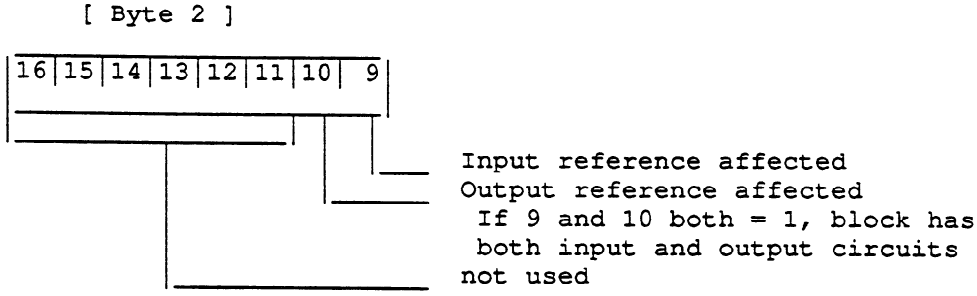
Detecting the Loss or Addition of a Block

To have the program detect the loss or addition of a block on the bus, use logic to monitor Bus Controller input reference bits 4 and 5.

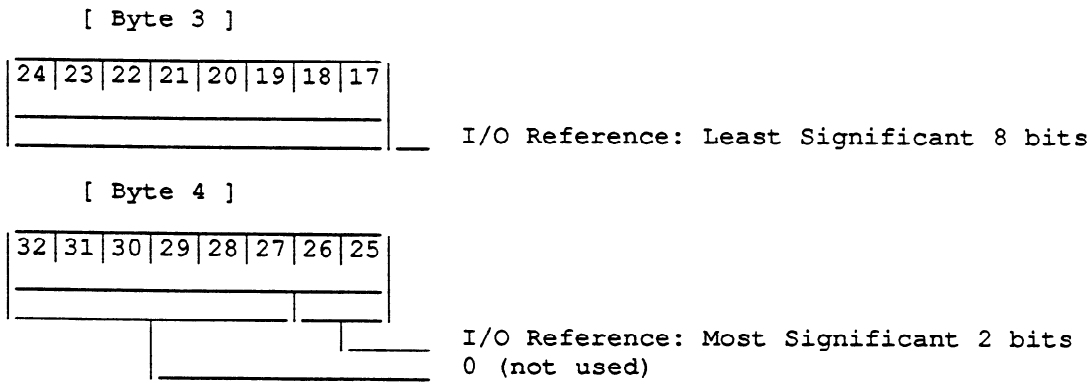


If bit 4 is 1, the Bus Controller has detected the loss of an I/O block that was previously operating. If bit 5 is 1, the Bus Controller has detected the addition of a block to the bus. Either of these bits may be 1 for one CPU sweep for each detected loss or addition; they are never both equal to 1 at the same time. If either of these bits is equal to 1, the other Bus Controller input reference bytes describe the block that was lost or added.

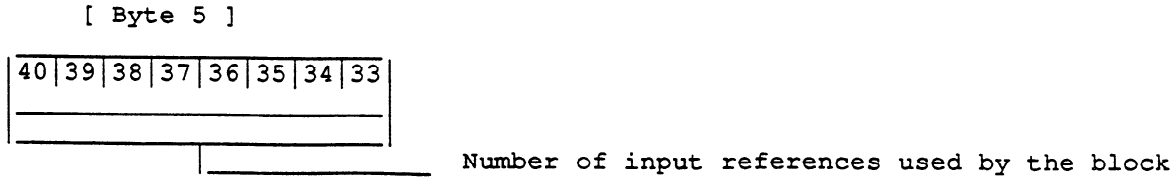
Byte 2 indicates the block's I/O type: all inputs, all outputs, or combination.



Bytes 3 and 4 contain the block's starting I/O reference:

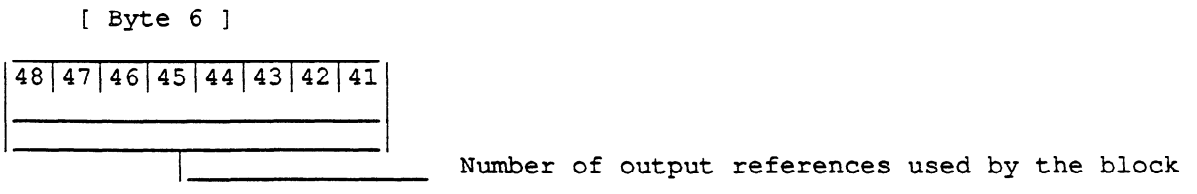


Bytes 5 and 6 contain the number of input and output references used by the block. Byte 5 contains the number of input references used by the block.



GFK-0171

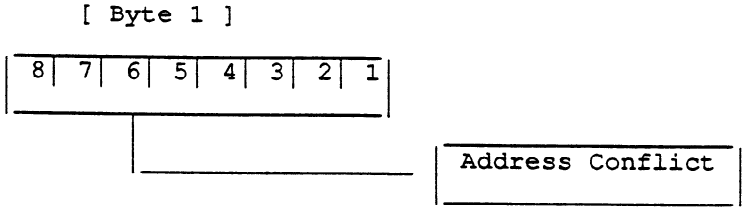
Byte 6 shows the number of output references used by the block.



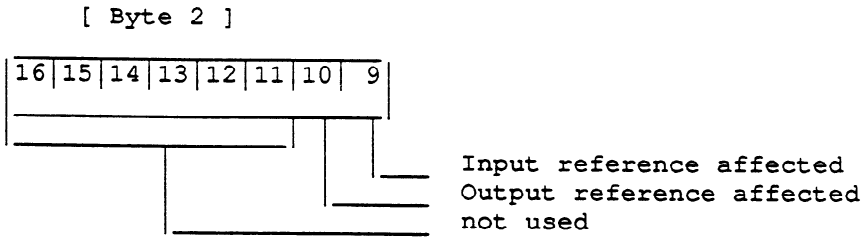
GFK-0171

Detecting Reference Number Conflict

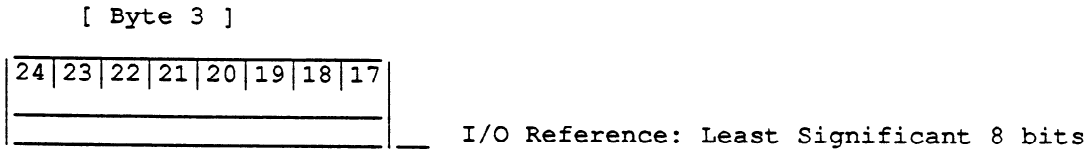
If the program should check for possible assignment of duplicate or overlapping Reference Numbers, monitor bit 6 of the Bus Controller input references. Bit 6 is set to 1 for one sweep for each conflict reported.

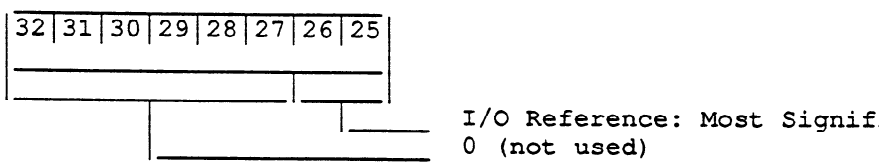


If bit 6 is 1, indicating a conflict, bits 9 and 10 indicate whether the conflict involves input references (input 9 is 1) or output references (input 10 is 1), or both inputs and outputs (both bits are ON).

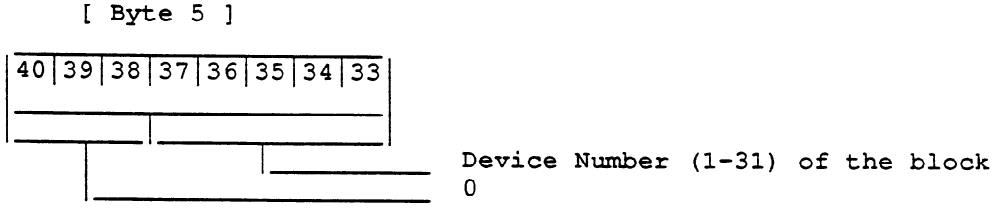


Bytes 3 and 4 contain the lowest reference (0-993) involved in the conflict.

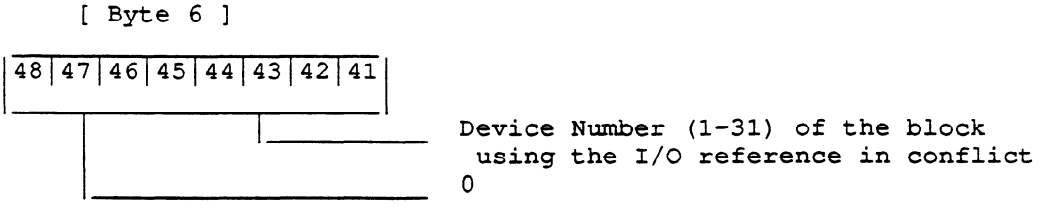




Byte 5 is the Device Number (the serial bus address) of the block that was not accepted. That block is left in a default state; no inputs are accepted and no outputs are activated.



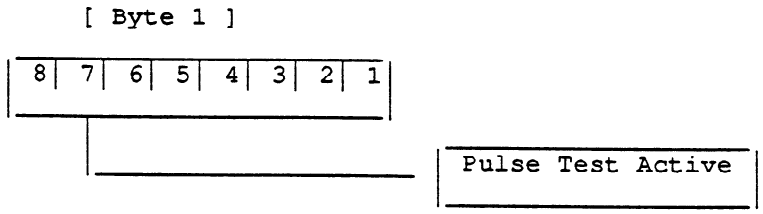
Byte 6 contains the Device Number (1-31) of the block that is already using the I/O references requested by the second block.



Detecting Execution of a Pulse Test

The program can periodically issue a command to Pulse Test discrete outputs on a bus (see chapter 6). Additional program logic can monitor execution of the Pulse Test, and determine whether the Pulse Test has located any faults.

To check execution of the Pulse Test, monitor Bus Controller input reference bit 7.

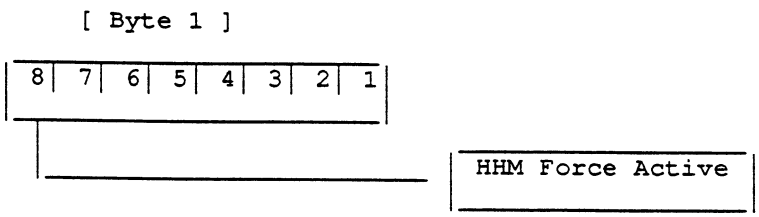


Bit 7 is set for one CPU sweep after a Pulse Test is commanded by the CPU. It remains 1 until all discrete I/O blocks configured for the Pulse Test have completed the test.

To determine whether any faults have been generated as the result of a Pulse Test (Failed Switch, Loss of I/O Power, Short Circuit, or No Load), monitor bit 3 (Circuit Fault).

Detecting a Force Condition on the Bus

The Hand-held Monitor can be used to force circuits, which removes them from program control. Such a force condition must also be removed with the Hand-held Monitor. The program can detect whether a force exists by monitoring Bus Controller input reference bit 8. If this bit changes to 1, the program can alert an operator that a forced condition exists on the bus. The operator can remove the force with a Hand-held Monitor.



Bit 8 is equal to 1 if any discrete or analog circuit is forced. No indication is provided as to which I/O block or reference contains the force condition.

If forces or unforces occur while the CPU is in Stop mode, or while outputs are disabled to the block being forced or unforced, this bit may not reflect changes, and may not be accurate when the CPU is returned to Run mode, or when outputs to the block are enabled. Some versions of block firmware are responsible for this effect.

Storing Diagnostic Information in CPU Registers

The CPU will automatically create a fault table in register memory if DIAGNOSTICS ENABLED is set to Y on the CPU Configuration Setup Menu (see chapter 4).

Alternatively, logic can be used to copy individual diagnostic reports from the Bus Controller's input references as described below. Because of its complexity, this method is not recommended.

Creating Diagnostic Tables

The information in the Bus Controller input references changes each CPU sweep. The following example shows program logic to copy the content of these references to register memory.

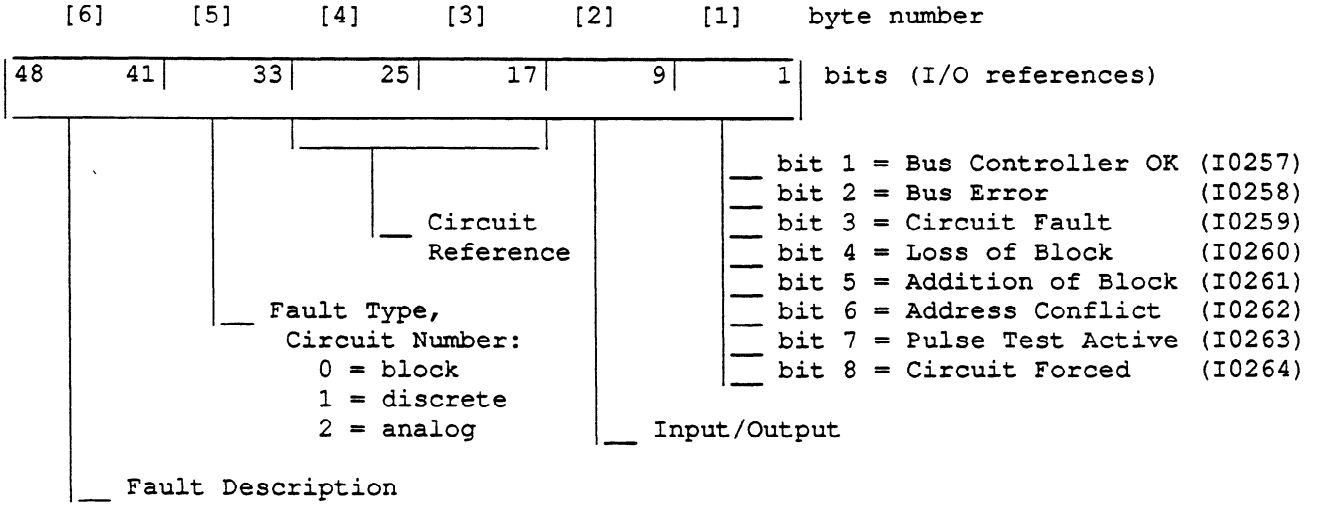
Ladder Logic Example

This example logic captures and displays diagnostic information from one Bus Controller for which automatic diagnostics is *not* enabled. For a system with multiple Bus Controllers, additional logic would be needed.

The logic creates a table which will store up to 19 faults at a time. The logic monitors Bus Controller input reference bit 3 (Circuit Fault), bit 4 (Loss of Block), bit 5 (Addition of Block) and bit 6 (Address Conflict). If any of these bits is equal to 1 during a CPU sweep, the program turns on an output which causes fault data to be logged into registers. The example program also monitors the Bus Controller OK and bus error bits, maintains an output which can be used to flash an indicator light, and clears the table pointer.

```
 << RUNG    0 >>

+[    Start of Program    ]-

 << RUNG     1 >>

 ********************************************************************************
 * This program will monitor fault diagnostic information from the Bus          *
 * Controller and establish a fault table in CPU register memory.  This         *
 * program assumes the Bus Controller's starting address is 257.                *
 ********************************************************************************

+[NO OP]-[NO OP]-[NO OP]-[NO OP]-[NO OP]-[NO OP]-[NO OP]-[NO OP]-[NO OP]-   (   )
```

First, the logic checks the condition of the Bus Controller by monitoring Bus Controller input bit 1 (the Bus Controller OK bit). This bit should always be equal to 1.



The Bus Controller's 48 assigned input references are I0257 through I0305, so bit 1 is located at I0257.

```
 << RUNG    2 >>

 ******************************************************************************
 * This rung monitors the Bus Controller OK bit from the Bus Controller and   *
 * the latched bus error output from the next rung.  If the Bus Controller    *
 * is NOT OK or if there are 10 or more bus communications errors within 10   *
 * seconds, output 1 will indicate the problem by turning off.                *
 ******************************************************************************

  BUS                                                                  GENIUS
CNTRLR    BUS                                                          BUS
OK BIT   ERROR                                                        IS OK

 I0257   00904                                                        00001
+--] [-----]/[-------------------------------------------------------------( )
|
```

Rung 3 checks for bus errors by monitoring Bus Controller input bit 2, located at I0258.

```
 << RUNG    3 >>

 ********************************************************************************
 * This rung latches the Bus Communications Error bit.  It will turn on if    *
 * 10 or more errors occur within 10 seconds.  Output 1 (Genius Bus OK) will  *
 * turn off if an error occurs.  (The previous rung).  Can reset output 1 by  *
 * turning on input 1.                                                        *
 ********************************************************************************

BUS COM
 ERROR                                                                  BUS
  BIT                                                                  ERROR

 I0258                                                                 O0904
+--] [-------------------------------------------------------[LATCH]---( L)
|                                                                       (  )
 RESET                                                                 (  )
BUS COM                                                                (  )
 ERROR                                                                 (  )
                                                                       (  )
 I0001                                                                 (  )
+--] [-------------------------------------------------------------------(UL)
 ERROR                                                                 ERROR
  BIT

 I0258                                                                 O0904
+--] [-------------------------------------------------------[LATCH]---( L)
|
```

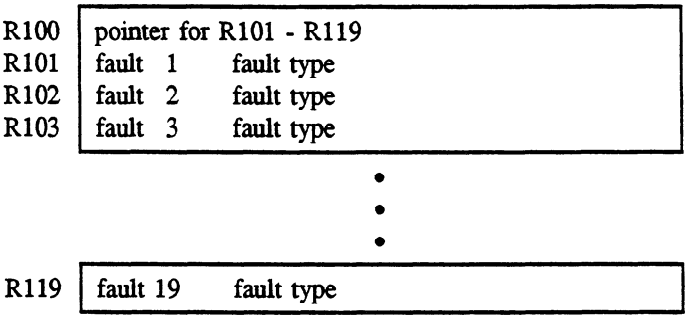
Rung 4 checks for the presence of any I/O circuit fault by monitoring input bit 3 (I0259), the addition or loss of a block (I0260 and I0261), or a reference number conflict.

```
 << RUNG    4 >>

 ************************************************************************
 * This rung monitors the Circuit Fault bit, Loss of Block bit, Addition of  *
 * Block bit, and Address Conflict bit from the Bus Controller.  If any of   *
 * these types of faults occur output 909 is turned on.  This is             *
 * used in the next several rungs to cause the fault information to be       *
 * logged into registers.                                                    *
 ************************************************************************

CIRCUIT   BUS                                                      A FAULT
 FAULT   CNTRLR                                                     HAS
  BIT    OK BIT                                                   OCCURRD

 I0259   I0257                                                      O0909
+--] [--+--] [-----------------------------------------------------------( )
|       |
|LOSS OF|
| BLOCK |
|  BIT  |
|       |
| I0260 |
+--] [--+
|       |
|ADD. OF|
| BLOCK |
|  BIT  |
|       |
| I0261 |
+--] [--+
|       |
|ADDRESS|
|CONFLCT|
|  BIT  |
|       |
| I0262 |
+--] [--+
```

If a fault occurs, rung 5 creates a table of 19 registers. Each register can contain the first 16 Bus Controller input references for one fault.

```
 << RUNG    5 >>

 ********************************************************************************
 * Registers 101 through 119 store the first two input status bytes for each    *
 * of 19 possible faults.  Each register can contain the following:             *
 *   In each register: - bit 1  = Bus Controller OK                             *
 *                       bit 2  = Bus Error                                     *
 *                       bit 3  = Circuit Fault                                 *
 *                       bit 4  = Loss of Block                                 *
 *                       bit 5  = Addition of Block                             *
 *                       bit 6  = Address Conflict                              *
 *                       bit 7  = Pulse Test Active                             *
 *                       bit 8  = Forced Circuit                                *
 *                       bit 9  = Input      If bits 9 and 10 are both on =     *
 *                       bit 10 = Output       Input and Output.                *
 *                    bits 11-16 = not used                                     *
 ********************************************************************************

A FAULT   BUS            FAULT
  HAS    CNTRLR          TYPE
OCCURRD  OK BIT         STORAGE

 00909    I0257          R00100   Const
+--] [---[ SRC ADD-TO-TOP  LIST      LEN]-                                ( )
|                                    019
```

The result of this rung is:



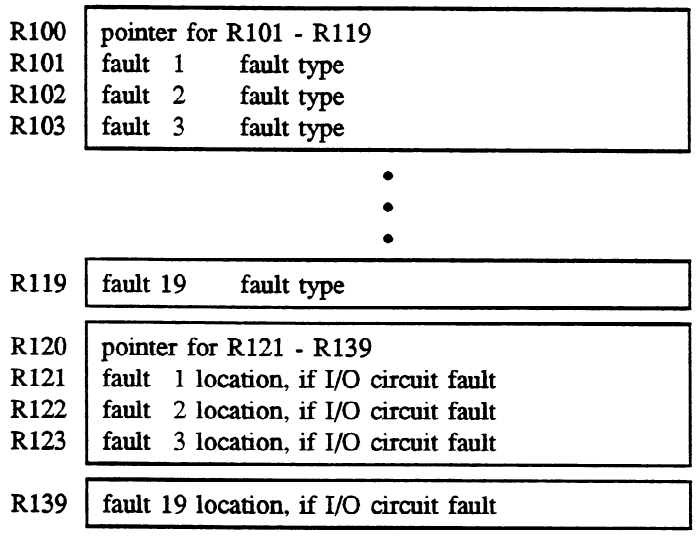
The first register assigned by the Add-to-Top function is a pointer to the rest of the entries in the list. The first fault information will be placed in R101.

These registers will show the fault type for each of the 19 faults.

If a fault occurs, rung 6 creates another table of 19 registers. Each register in this table can contain Bus Controller input references 17 - 32 for one fault. These references indicate the location of an I/O circuit fault.

```
 << RUNG    6 >>

 ********************************************************************************
 * Registers 121 through 139 store Input Status bytes 3 and 4 from the Bus      *
 * Controller for each fault that occurs.  This is the circuit reference        *
 * number in binary.                                                            *
 ********************************************************************************

A FAULT                  CIRCUIT
  HAS                     REF.
OCCURRD                  STORAGE

 00909    I0273          R00120   Const
+---] [---[ SRC ADD-TO-TOP  LIST      LEN]-                                ( )
|                                     019
|
```

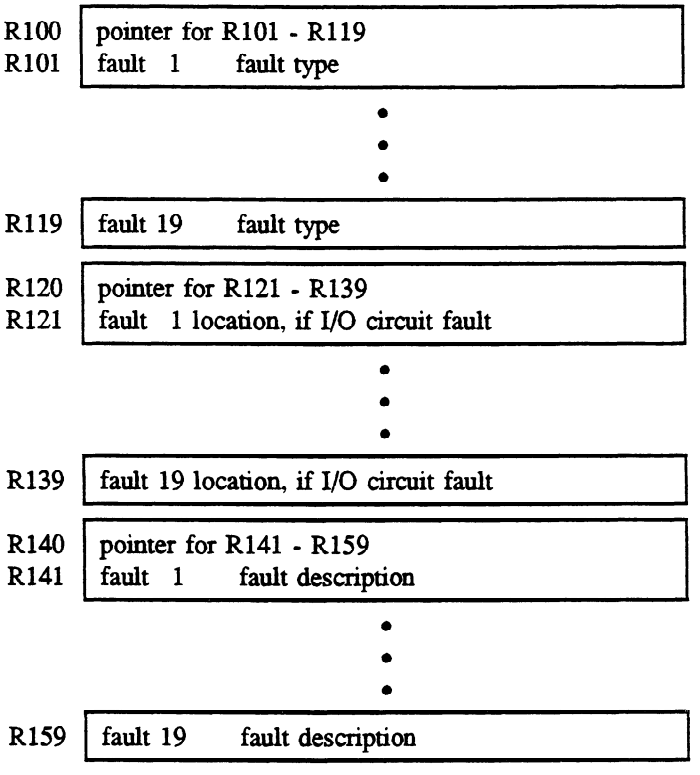
Now, the assignment of register memory is:



Similarly, rung 7 creates a table that stores additional information for each of the 19 faults.

```
 << RUNG    7 >>

 ****************************************************************************
 * Registers 141 through 159 store Input Status bytes 5 and 6 from the Bus   *
 * Controller for each fault that occurs.  The first eight bits store        *
 * whether the fault is internal to a block, a discrete circuit, or an       *
 * analog circuit.  Also the relative circuit number on the block is stored. *
 * The second eight bits of each register are a description of what type     *
 * of circuit fault occurred.                                                *
 ****************************************************************************

A FAULT                  FAULT
  HAS                   DESCRIP
OCCURRD                 STORAGE

 00909    I0289          R00140   Const
+--] [---[ SRC ADD-TO-TOP  LIST     LEN]-                                ( )
|                                   019
|
```

Now, the assignment of register memory is:



Rungs 8 through 12 of this example logic cause an indicator light to flash if any type of fault has occurred.

```
 << RUNG    8 >>

 ********************************************************************************
 * This instruction forces R23 to always contain 00000.                         *
 ********************************************************************************

                ALWAYS
                 ZERO

 Const           R00023
+[  A     MOVE      B ]-                                                  (  )
 +00000

 << RUNG    9 >>

 ********************************************************************************
 * This rung compares one of the storage list's pointers to determine if any *
 * faults exist.  If R120 compares with the 00000 in R23 no faults exist and *
 * output 910 is turned on.                                                     *
 ********************************************************************************

CIRCUIT                                                                  FAULT
 REF.   ALWAYS                                                            LIST
STORAGE  ZERO                                                           EMPTY

 R0120  R0023                                                           00910
+[  A  :    B ]-                                                         (  )

 << RUNG   10 >>

 ********************************************************************************
 * If any faults are stored in the fault storage registers, R120 will not    *
 * equal 00000 and output 910 will be off.  This allows output 6 to flash on *
 * and off to indicate that at least one fault exists.                          *
 ********************************************************************************

FAULT                                                                   GENIUS
LIST   FLASH                                                            FAULTS
EMPTY   ON                                                               EXIST

 00910  00905                                                           00005
+--]/[-----] [-------------------------------------------------------------(  )
```

```
<< RUNG   11 >>

*****************************************************************************
* This rung times the on portion of a flashing indicator light.             *
*****************************************************************************

                                                                      FLASH
                                                                        ON

                                                              Const   O0905
+-----------------------------------------------------------[PRESC]---(TT)
                                                                003     (  )
                                                                        (  )
FLASH                                                                   (  )
 OFF                                                                    (  )
                                                                        (  )
O0906                                                         R00002    (  )
+--] [-----------------------------------------------------------[ACCRG]---( R)


<< RUNG   12 >>

*****************************************************************************
* This rung times the off portion of a flashing indicator light.            *
*****************************************************************************

FLASH                                                                 FLASH
 ON                                                                    OFF

O0905                                                          Const   O0906
+--] [-----------------------------------------------------------[PRESC]---(TT)
                                                                003     (  )
                                                                        (  )
FLASH                                                                   (  )
 OFF                                                                    (  )
                                                                        (  )
O0906                                                         R00003    (  )
+--] [-----------------------------------------------------------[ACCRG]---( R)
```

Rungs 13 and 15 are used to reset (clear) registers and the Bus Controller OK bit. Rung 14 turns off the Bus Controller OK bit at the end of each sweep. This function is performed automatically by Logicmaster 6 software release 3.0 or later.

```
<< RUNG   13 >>

*****************************************************************************
* This rung insures that all of the fault storage registers are cleared if  *
* the CPU is restarted or if the fault list pointers are equal to 00000.    *
*****************************************************************************
POWER    FAULT    FAULT    FAULT
 UP      TYPE     TYPE     TYPE
RESET   STORAGE  STORAGE  STORAGE

O0908    R00100   R00100   R00100   Const
+--]/[--+[  A   EOR   B    =    C       LEN ]-                              (  )
        |                               060
 FAULT  |
  LIST  |
 EMPTY  |
        |
 O0910  |
+--] [--+

<< RUNG   14 >>

*****************************************************************************
* This rung turns off the Bus Controller OK bit at the end of each CPU      *
* sweep so that the Bus Controller can turn it back on.  It is then checked *
* to always be on in the program above.  If it is ever found to be off,     *
* output 1 is turned off to indicate the problem.                           *
*****************************************************************************
                 BUS
               CNTRLR
               OK BIT

  Const           I0257   Const
+[ BIT   CLEAR   MATRIX     LEN]-                                           (  )
 00001                      001

<< RUNG   15 >>

*****************************************************************************
* This rung generates the power up reset pulse used in the program above.   *
* Output 908 is always on except during the first sweep of the CPU when it  *
* is first powered up or restarted.                                         *
*****************************************************************************
                                                                    POWER
                                                                     UP
                                                                    RESET

                                                                    O0908
+[NO OP]-------------------------------------------------------------(  )

<< RUNG   16 >>

[ENDSW]-

<< RUNG   17 >>

[ENDSW]-
```

Displaying the Example Diagnostic Table

The Display Register Tables function of the Logicmaster 6 software can be used to display the information currently in the fault table registers assigned by the example program logic. The Register Tables display shows a full screen of register values in either decimal or hexadecimal format. Initially, register values are in decimal. Pressing the Change All (F7) key and the Hex Display (F3) key converts all values to hexadecimal:

```
                                                          L/M OFFLINE
REG                    REGISTER     00101  (      ) EQUALS 0000001100000101
00100      0002   01C3   0000   0148   0001   0000   0000   003E   0000   0000
00110      0000   0000   0000   0000   0000   0000   0000   0000   0205  |0305|
00120      0002   01C3   0000   0148   0001   0000   0000   003E   0000   0000

00130      0000   0000   0000   0000   0000   0000   0000   0000   0032   0031
00140      0000   0000   0000   0000   0000   0000   0000   0000   0000   0000
00150      0000   0000   0000   0000   0000   0000   0000   0000   0411   0801

00160      03E9   0000   0000   0000   0000   0000   0000   0000   0000   0000
00170      0002   0002   00C7   4420   4924   0000   0000   0000   4074   4074
00180      03E9   0000   000C   0000   0000   0000   0005   0000   0000   0000

00190      0000   0000   0000   0000   0000   0000   0FA0   0FA0   0000   0007
0200       0000   0000   0000   0000   0000   0000   0000   0000   0205   0000
00210      0002   01C3   0000   0148   0001   0000   0000   003E   0000   001E

00220      0000   0000   0000   0000   0000   0000   0000   0000   0205   0000
00230      0000   0000   0000   0000   0000   0000   0000   0000   0205   0000
00240      0002   01C3   0000   0148   0001   0000   0000   003E   0000   0000

  DEC    SIGNED   HEX    DBPREC        TEXT   FL PT   CHANGE  DISPLY
1DISPLY 2DISPLY 3DISPLY 4DISPLY      5DISPLY 6DISPLY 7 ALL   8REF TB
```

In addition to the Register Table display, the Mixed Reference Tables function could be used to create custom tables of register, I/O and text data. To create a Mixed Reference display, see *the Logicmaster 6 Software User's Manual* for instructions.

In the preceding example, register R100 is the pointer to registers R101 through R119, which store the fault types (see rung 5). The number currently in this pointer register is the number of faults that have occurred. Looking at register R100 shows that two faults are stored in the fault table:

```
                                                          L/M OFFLINE
REG                    REGISTER     00101  (      ) EQUALS 0000000000000010
00100      0002   01C3   0000   0148   0001   0000   0000   003E   0000   0000
             |
             |__ 2 faults
```

Registers R101 through R119 store the fault type for 19 possible faults. The information in these registers is copied from Bus Controller input references 1 through 16:

bit	1	=	Bus Controller OK
bit	2	=	Bus Error
bit	3	=	Circuit Fault
bit	4	=	Loss of Block
bit	5	=	Addition of Block
bit	6	=	Address Conflict
bit	7	=	Pulse Test Active
bit	8	=	Forced Circuit
bit	9	=	Input If bits 9 and 10 are both = 1,
bit	10	=	Output Input and Output
bits	11-16	=	not used

Moving the cursor to R101 would display its binary equivalent beside the word EQUALS:

```
                                                    L/M OFFLINE
 REG              REGISTER     00101  (       ) EQUALS 0000001100000101
00100   0002   01C3   0000   0148   0001   0000   0000   003E   0000   0000
00110   0000   0000   0000   0000   0000   0000   0000   0000   0205   0305
                                                                      __ R101
```

This is the fault type information for fault 1:

```
EQUALS 0000001100000101
```

Bits 1, 3, 9, and 10 are 1, showing that the Bus Controller is OK (bit 1 is 1) and there is a Circuit Fault (bit 3 is 1). Bits 9 and 10 are both equal to 1, so the circuit with the fault is an output with feedback.

If the cursor were moved to R102, the binary content of Bus Controller input bytes 1 and 2 for the second fault would appear beside the word EQUALS:

```
EQUALS 0000001000000101
```

Bits 1, 3, and 10 are equal to 1. The Bus Controller is OK and there is a Circuit Fault. Bit 10 is set to 1, showing that the circuit is an output.

Registers R121 through R139 each store Bus Controller input references 17 through 32 for one of the 19 faults (see rung 6). If a fault is an I/O circuit fault, its reference number will appear in the appropriate register.

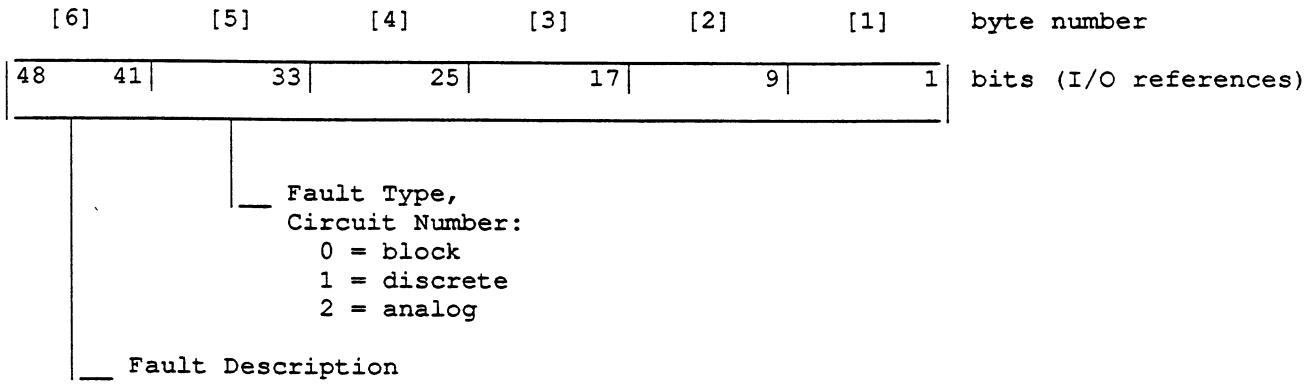
Register R101 shows that fault 1 is a circuit fault. Its reference number would be stored in register R121. In the previous screen illustration, the table was shown with hexadecimal values. Pressing the Decimal Display (F1) would show the content of register R101 in decimal:

```
00049
```

This is the I/O reference of the I/O circuit having the fault.

GFK-0171

Registers R141 through R156 can store the following information about each fault:



Fault 1 is an I/O circuit fault. Its fault type and description would be stored in R141. Moving the cursor to R141 would show the binary content of this register:

```
EQUALS 0000100000000001
```

Bit 33 (the first bit shown in status byte 5), is equal to 1. That means the fault is on a discrete block. The value 0 in bits 37-40 shows that the fault has occurred on the topmost circuit on the block. Finally, bit 45 is set to 1, showing that the fault is an OVERTEMPERATURE fault.

Chapter 9

9-1 Programming Commands to I/O Blocks Using the Bus Controller Output References

GFK-0171

This chapter explains how setting or clearing individual Bus Controller output reference bits can:

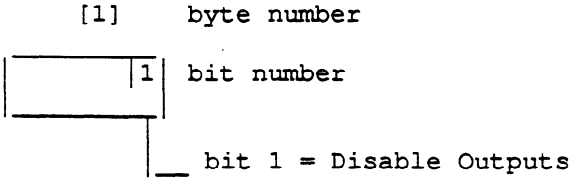
Disable all outputs on the bus	(all Bus Controllers)
Clear all faults	(Bus Controllers with Diagnostics only)
Clear faults on a specific circuit	(Bus Controllers with Diagnostics only)
Pulse Test discrete outputs	(Bus Controllers with Diagnostics only)

Bus Controller Output References

The length and content of the Bus Controller output references depend on whether the Bus Controller has diagnostics capabilities.

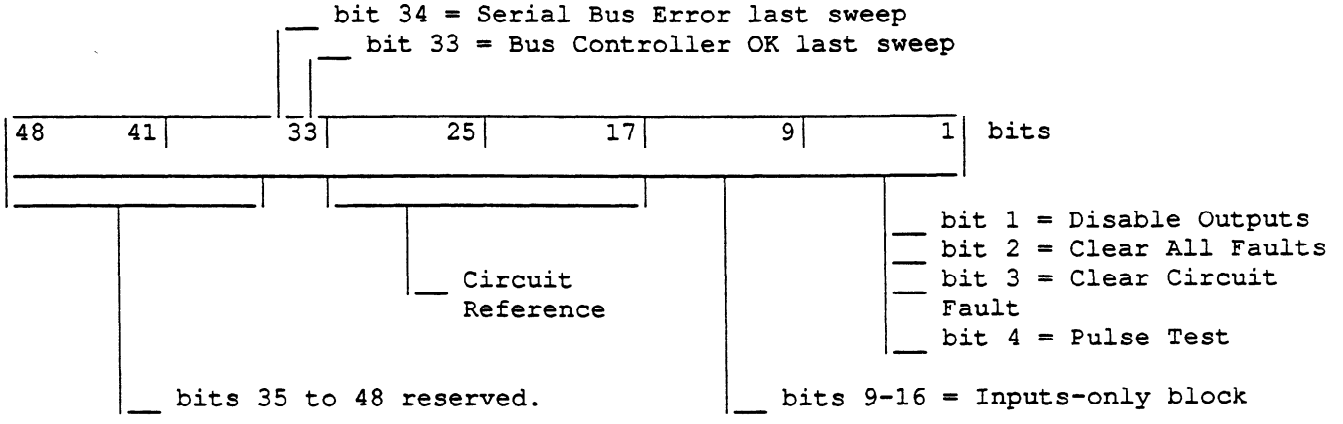
Bus Controller without Diagnostics

A Bus Controller without Diagnostics (IC660CBB903) has 8 output reference bits. Only the first bit is used. Bits 1 to 32 are used by the Bus Controller directly.



Bus Controller with Diagnostics

A Bus Controller with diagnostics (IC660CBB902) has 48 output reference bits. The first four bytes are used for commands to the blocks. Bits 1 to 32 are used by the Bus Controller directly. Bits 33 and 34 are controlled by the PLC if the Bus Controller has been entered on the Logicmaster 6 software Bus Controller Locations screen.

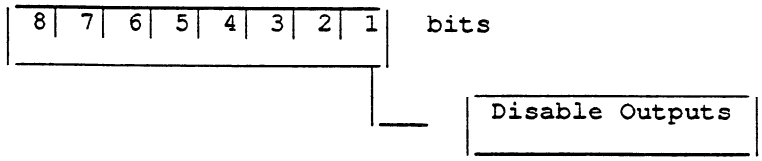


Bits 1 - 4 are used for commands. Bits 9 - 32 are used only for the Clear Circuit Fault command; they contain the circuit reference of the fault, and its I/O type.

Bit 33 can be monitored to detect a Bus Controller error (see chapter 8).

Enabling or Disabling all Outputs from the Bus Controller

Bus Controller output reference bit 1 can be used to enable or disable outputs to all I/O blocks on the bus. (To disable outputs on a block-by-block basis, see chapter 5).



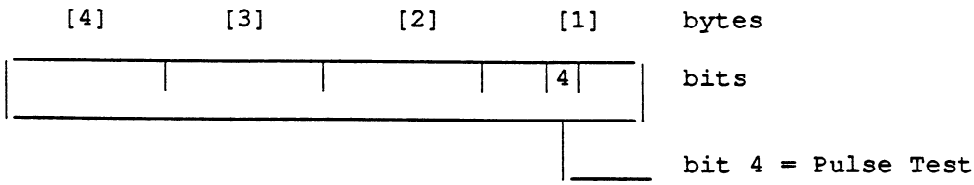
When this bit is set to 1, the blocks stop receiving outputs and their I/O Enabled LEDs go off. Each output either defaults to a previously-selected state (or value), or holds its last state or value, depending on how the block has been configured. The Bus Controller still accepts input data from the blocks, which remain "logged on" to the system; no Addition of Block or Loss of Block diagnostic is generated.

When bit 1 is set to 0, the outputs begin to receive data and the I/O Enabled LEDs go on. Blocks will drive the new states/values as soon as new output data is received.

Pulse Testing Outputs

Pulse Testing checks the ability of discrete outputs to change state. It also checks the continuity of switching devices, power sources, wiring, interposing devices such as fuses, and output devices. The program can issue a Pulse Test command periodically (for example, once a day) to check outputs that seldom change state during normal system operation. Executing a Pulse Test will not activate mechanical devices such as motor starters, relays, or solenoid valves. However, blocks that do not control sensitive loads can be configured to ignore a Pulse Test command. Pulse Testing can also be done using a Hand-held Monitor.

To have the program Pulse Test all discrete outputs on a bus, set Bus Controller output reference bit 4 to 1.

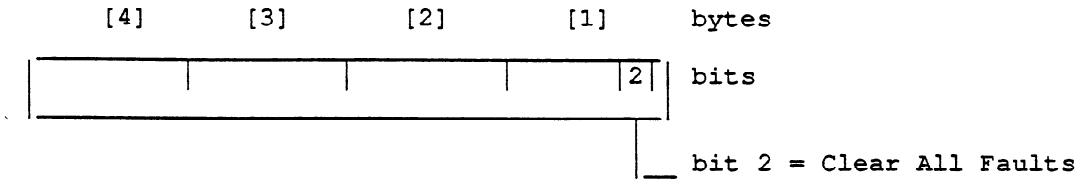


The Bus Controller will send the Pulse Test command to all discrete blocks on the bus. Discrete blocks that have been configured to disable Pulse Testing, inputs-only blocks, and analog I/O blocks will not perform the test. The test will begin at the block with the lowest Device Number, and proceed as quickly as the CPU receives test complete signals through all I/O blocks up to device 31. The Pulse Test should cause outputs that are OFF to go ON and outputs that are ON to go OFF momentarily. If an output does not switch, a fault (Failed Switch, Loss of I/O Power, Short Circuit, or No Load) is generated. The Bus Controller then sets the appropriate diagnostic bits in its assigned input references for one CPU sweep per fault. After all blocks have completed the Pulse Test, Bus Controller input reference bit 7 (Pulse Test Active) is set to 0.

If output bit 4 were still equal to 1 when the Pulse Test finished, another round of Pulse Tests would start immediately. Because outputs should not be pulsed continuously, the ladder logic should set the Pulse Test output bit (4) to 1 for one CPU sweep at a time. If repeated Pulse Testing is required, Pulse Test cycles should be at least 5 minutes apart.

Clearing all Faults on the Bus

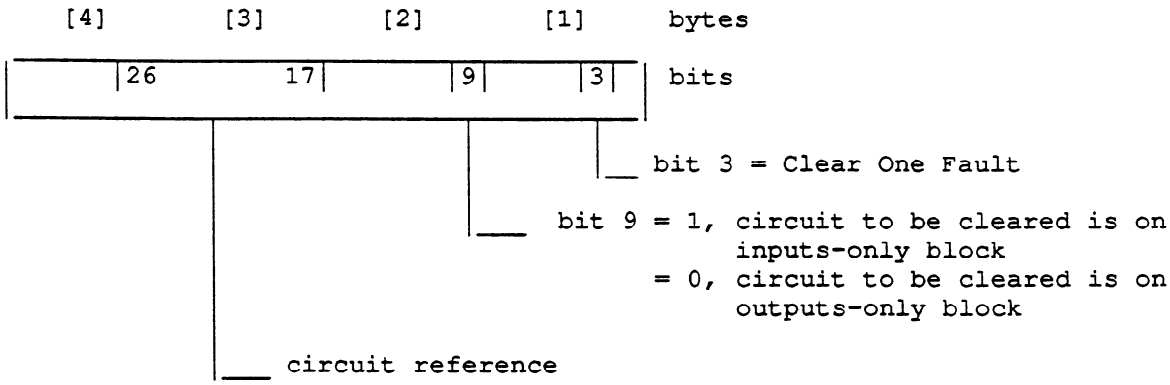
To have the program automatically clear all faults on the bus and all circuit faults buffered in the Bus Controller, set bit 2 for one CPU sweep. The bit must transition from 0 to 1 to clear faults.



The Clear All Faults command causes the blocks to attempt to clear their faults once. If a fault condition (for example, a short circuit) still exists, the fault is reported again.

Clearing a Specific Circuit Fault

To have the program automatically clear just one fault, set bit 3 for one CPU sweep. The bit must transition from 0 to 1 to clear the fault.



Use bit 9 to indicate whether the block has inputs only or outputs only. Bit 9 will be ignored if the block has both input and output circuits.

Specify the circuit reference and the block's I/O type in the remaining output references. *Individual circuit faults can be cleared for any block assigned a Reference Number in I/O memory. Circuit faults on blocks assigned a Reference Number in register memory cannot be cleared using this command. A Datagram command to the block can be used instead.*

Bits 17-26 contain the circuit reference (1 to 1000). Analog circuits are identified by the first six references assigned to the analog block. The following example shows circuit assignments for either an RDT or a 4 Input/2 Output Analog Block assigned I/O references I0801 and O0801:

RTD Circuit	Analog Circuit	I/O Reference	Value
Input circuit 1	Input circuit 1	Starting reference	0801
Input circuit 2	Input circuit 2	Starting reference plus 1	0802
Input circuit 3	Input circuit 3	Starting reference plus 2	0803
Input circuit 4	Input circuit 4	Starting reference plus 3	0804
Input circuit 5	Output circuit 1	Starting reference plus 4	0805
Input circuit 6	Output circuit 2	Starting reference plus 5	0806

Each time the Clear Circuit Fault command is received, any faults associated with the specified circuit are cleared. To clear another circuit (either discrete or analog), set bit 3 to 0 for one CPU sweep, change the values in Bus Controller bytes 2, 3 and 4 to reflect the individual circuit to be cleared and set bit 9 to reflect the I/O type of the circuit. Then set bit 3 to 1 again.

Chapter 10 Troubleshooting

This chapter lists errors that might occur, and suggests corrective actions you can take.

Errors are most likely when a new system is being started up. They are often caused by mistakes in cabling or field wiring, or by faulty logic in the CPU's application program. If problems occur, consult the troubleshooting information in this chapter. It will help you isolate any problem that originates in a Bus Controller. If these steps do not pinpoint the problem, the cause may lie in the CPU or programmer. You should refer to chapter 5 of the *Series Six Plus Installation and Maintenance Manual* (GEK-96602) for further troubleshooting information.

If you have questions that are not answered in this manual or in the other documentation for your system, contact your local authorized GE Fanuc distributor. After business hours, please don't hesitate to call the Programmable Control Emergency Service Number, (804) 978-5747 (DIAL COMM 8-227-5747). An automatic answering device will direct you to the home phone of one of our Programmable Control Service Personnel. Thus, you are never without backup help.

How to Begin

- Check the operating mode of the CPU and, if appropriate, the programmer.
- Check the status LEDs on the CPU.
  - If some of the CPU status LEDs are off, refer to the *Series Six Installation and Maintenance Manual.*
  - If all the CPU status LEDs are on but either of the Bus Controller LEDs is not, refer to the troubleshooting information on the following pages.
  - If all the CPU and Bus Controller LEDs are on, check cabling then proceed to I/O block troubleshooting. Refer to the *Genius I/O System User's Manual.*

Identifying the Problem

If a problem occurs, look for a description of the problem in the list that begins below. Then, refer to the troubleshooting suggestion with the same number on the pages that follow.

1. Both Bus Controller LEDs are off.
2. The Bus Controller BOARD OK LED is off and the COMM OK LED is on.
3. The BOARD OK LED is on and the COMM OK LED is off.
4. The BOARD OK and COMM OK LEDs are flashing in unison.
5. The Bus Controller is not communicating with the CPU. Intermittent or total lack of communications. No input data at CPU. No output data at block.
6. Window commands to the Bus Controller cause a syntax error.
7. Window commands to the Bus Controller don't show any status change.
8. The Bus Controller is not communicating on the Genius I/O serial bus.
9. The Bus Controller begins operating, but does not seem to be operating normally.
10. There are no functioning circuits on one bus, but other busses are working.
11. There are no functioning circuits on more than one bus.

12. The CPU system shuts down with parity errors after operating for a short time, or after changing the system configuration.
13. Communications on the bus are intermittent or lacking.
14. One of the following occurs:
    a. The Bus Controller COMM OK light flashes excessively.
    b. There are delays on the bus.
    c. Addition of Block/Loss of Block diagnostics occur repeatedly although no blocks are actually being added or removed.

Problems 1 - 4

1. Both Bus Controller LEDs are off.
   - The Bus Controller is probably not receiving enough power from the rack or the computer power supply. Be sure the board is seated properly.
2. The Bus Controller BOARD OK LED is off and the COMM OK LED is on.
   - Be sure the Bus Controller is installed in the correct slot.
   - Be sure the rack DIP switches are set for a Reference Number at or below 993 for a Bus Controller without Diagnostics, or at or below 953 for a Bus Controller with Diagnostics.
3. The BOARD OK LED is on and the COMM OK LED is off.
   - Check for correct cable type and length.
   - Check for correct terminating impedance at both ends of the bus.
   - Be sure the cable is daisy-chained.
   - Be sure wires to the Serial 1 and Serial 2 terminals are not crossed.
   - Look for a broken cable.

Problems 4 - 8

4. The BOARD OK LED and COMM OK LEDs are flashing in unison.
   - The Bus Controller detects another device on the same bus using the same Device Number. Change the Device Number of one of the two.
5. The Bus Controller is not communicating with the CPU.
   - This may indicate a programming or address assignment error. Also, check the CPU operating mode. Check particularly for overlapping addresses with other modules in the CPU rack. This includes blocks controlled by other Bus Controllers in the CPU rack.
6. Window commands to the Bus Controller cause a syntax error.
   - If the Bus Controller is located in a Remote I/O rack, DPREQ and WINDOW commands cannot be used. Please refer to chapter 1 for more information.
7. Window commands to the Bus Controller don't show any status change.
   - The Window Command is being sent to a non-existent Bus Controller. Power flows through the DPREQ or WINDOW instruction, but the status doesn't change.
8. The Bus Controller is not communicating on the Genius I/O serial bus.
   - Two devices on the same bus may have been configured with the same Device Number. Check this using the Hand-held Monitor. Note that a Phase B Bus Controller will not communicate on a bus if its assigned Device Number is already used by another device. However, both the Unit OK and the COMM OK LEDs will be blinking together.

- Be sure wires to the Serial 1/Serial 2 terminals on the module are not crossed or shorted together or to ground.
- Check the baud rate.
- Check the Device Number (serial bus address) assigned to the Bus Controller against the intended Device Number from your records of system configuration. New Bus Controllers are shipped from the factory already set up to use Device Number 31.
- Use the HHM to compare Device Numbers and Reference Numbers.
- Check the Bus Controller's Outputs Disabled bits using Read Configuration command to the Bus Controller. Also check Bus Controller output #1 (Disable All Outputs).

Problems 9 - 12

9. The Bus Controller begins operating, but does not seem to be operating normally.
   - Be sure serial bus wiring has been completed in a daisy chain fashion.
   - Make sure the communications cable is not close to high voltage wiring.
   - Look for a broken cable. Check for intermittent cable breaks and connections.
   - Ensure that cable shielding is properly installed and grounded (see chapter 6 of the *Genius I/O System User's Manual*).
10. There are no functioning circuits on one bus, but other busses are operating normally.
   - From the CPU, see if the Bus Controller has its Outputs Disabled. This selectable feature allows a module to receive inputs, but not to send outputs. See the chapter 1 for more information.
   - Check to see if the Bus Controller is properly installed, seated properly, and receiving power.
   - Check the on-board DIP switches, especially switch 4 at position U16.
   - Pull out the Bus Controller and reinsert.
   - Check for loose communications cable connections or breakage.
   - If necessary, replace the Bus Controller.
11. There are no functioning circuits on more than one bus.
   - Please refer to the documentation for the Series Six PLC for troubleshooting information.
12. The CPU system shuts down with parity errors after operating for a short time, or after changing the system configuration.
   - There may be duplicate or overlapping I/O references coming from different busses.
   - Unplug one Bus Controller, refer to the configuration worksheets, and use the HHM to read Reference Numbers. If necessary, check other buses the same way.
   - Verify that no conventional I/O module has reference numbers that overlap references assigned to Genius I/O devices.

Problems 13 - 14

13. Communications on the bus are intermittent or lacking.
   - This may be caused by mixed baud rates. To check this, power up blocks one at a time and look at their respective baud rates using HHM. If you find different baud rates, they must be changed. All devices on the bus must use the same baud rate. *Any change to baud rate in block will not take effect until block power is cycled.*
   - For Phase A devices, check for duplicate Block Numbers. Power devices up one at a time and confirm Block Numbers using the HHM.

- The terminating resistors on the bus may be missing or incorrectly chosen or placed. Check terminators at ends of the bus for correct resistance value; BSM cluster ''stubs'' should not be terminated.
- The cable may be too long. Shorten the cable or configure all devices on the bus to use a lower baud rate. Please refer to chapter 5 of the *Genius I/O System User's Manual* for more information about cabling and baud rate selection.
- Wires may be open, shorted, or reversed. Check all bus electrical connections.

14. The COMM OK light on the Bus Controller blinks excessively, and/or
there are propagation delays on the bus, and/or
the bus is operating, but the CPU repeatedly receives Addition of Block and/or
Loss of Block diagnostics.

- There is excessive ambient noise on the bus. This can be corrected by lowering the baud rate, re-routing the communications cable, or shielding the source of the electrical noise. The proper solution to these problems will depend on the application. Please refer to chapter 5 of the *Genius I/O System User's Manual* for information on cabling, baud rates, and ambient electrical noise.

Appendix A
Expanded I/O Addressing

The tables that follow show how Expanded I/O is mapped for CPUs with different amounts of memory.

Expanded I/O Addressing for 8K and 16K Registers

For a CPU with either 8K or 16K of register memory:

- Real I/O references O0+0001 to O0+1024, I0+0001 to I0+1024, O8+0001 to O8+1024, and I8+0001 to I8+1024 can not be used as program references. However, internal I/O references O0-0001 to O0-1024, I0-0001 to I0-1024, O8-0001 to O8-1024, and I8-0001 to I8-1024 can be used.
- Channel 1-7 real I/O references are for the Expanded Main I/O channels.
- Channel 9-F real I/O references are for the Expanded Auxiliary I/O channels.

REGISTERS		USED FOR	REGISTERS		USED FOR
Aux. I/O	R00001 to R00064	AO0001 to AO1024	Channel 0-	R02049 to R02112	O-0001 to O-1024
	R00065 to R00128	AI0001 to AI1024		R02113 to R02176	I-0001 to I-1024
Channel 1+	R00129 to R00192	O1+0001 to O1+1024	Channel 1-	R02177 to R02240	O1-0001 to O1-1024
	R00193 to R00256	I1+0001 to I1+1024		R02241 to R02304	I1-0001 to I1-1024
Channel 2+	R00257 to R00320	O2+0001 to O2+1024	Channel 2-	R02305 to R02368	O2-0001 to O2-1024
	R00321 to R00384	I2+0001 to I2+1024		R02369 to R02432	I2-0001 to I2-1024
Channel 3+	R00385 to R00448	O3+0001 to O3+1024	Channel 3-	R02433 to R02496	O3-0001 to O3-1024
	R00449 to R00512	I3+0001 to I3+1024		R02497 to R02560	I3-0001 to I3-1024
Channel 4+	R00513 to R00576	O4+0001 to O4+1024	Channel 4-	R02561 to R02624	O4-0001 to O4-1024
	R00577 to R00640	I4+0001 to I4+1024		R02625 to R02688	I4-0001 to I4-1024
Channel 5+	R00641 to R00704	O5+0001 to O5+1024	Channel 5-	R02689 to R02752	O5-0001 to O5-1024
	R00705 to R00768	I5+0001 to I5+1024		R02753 to R02816	I5-0001 to I5-1024
Channel 6+	R00769 to R00832	O6+0001 to O6+1024	Channel 6-	R02817 to R02880	O6-0001 to O6-1024
	R00833 to R00896	I6+0001 to I6+1024		R02881 to R02944	I6-0001 to I6-1024
Channel 7+	R00897 to R00960	O7+0001 to O7+1024	Channel 7-	R02945 to R03008	O7-0001 to O7-1024
	R00961 to R01024	I7+0001 to I7+1024		R03009 to R03072	I7-0001 to I7-1024
Gen. use	R01025 to R01088		Channel 8-	R03073 to R03136	O8-0001 to O8-1024
	R01089 to R01152			R03137 to R03200	I8-0001 to I8-1024
Channel 9+	R01153 to R01216	O9+0001 to O9+1024	Channel 9-	R03201 to R03264	O9-0001 to O9-1024
	R01217 to R01280	I9+0001 to I9+1024		R03265 to R03328	I9-0001 to I9-1024
Channel A+	R01281 to R01344	OA+0001 to OA+1024	Channel A-	R03329 to R03392	OA-0001 to OA-1024
	R01345 to R01408	IA+0001 to IA+1024		R03393 to R03456	IA-0001 to IA-1024
Channel B+	R01409 to R01472	OB+0001 to OB+1024	Channel B-	R03457 to R03520	OB-0001 to OB-1024
	R01473 to R01536	IB+0001 to IB+1024		R03521 to R03584	IB-0001 to IB-1024
Channel C+	R01537 to R01600	OC+0001 to OC+1024	Channel C-	R03585 to R03548	OC-0001 to OC-1024
	R01601 to R01664	IC+0001 to IC+1024		R03549 to R03712	IC-0001 to IC-1024
Channel D+	R01665 to R01728	OD+0001 to OD+1024	Channel D-	R03713 to R03776	OD-0001 to OD-1024
	R01729 to R01792	ID+0001 to ID+1024		R03777 to R03840	ID-0001 to ID-1024
Channel E+	R01793 to R01856	OE+0001 to OE+1024	Channel E-	R03841 to R03904	OE-0001 to OE-1024
	R01857 to R01920	IE+0001 to IE+1024		R03905 to R03968	IE-0001 to IE-1024
Channel F+	R01921 to R01984	OF+0001 to OF+1024	Channel F-	R03969 to R04032	OF-0001 to OF-1024
	R01985 to R02048	IF+0001 to IF+1024		R04033 to R04096	IF-0001 to IF-1024
				R04097 to R04112	Bus Ctr. bit map
				R04113 to R04116	User registers
				R04117 to R04119	Real time clock
				R04120	Fault table pointer
				R04121 to Rxxxxx	Fault table entries
				R08123 to R08192	Computer Mailbox 8K
				R16315 to R16384	Computer Mailbox 16K

Expanded I/O Addressing for 1K Registers

Registers can be used as register references, for Expanded I/O, or for Genius I/O diagnostics storage. For a CPU with 1K or 256 registers of memory, you should plan register use carefully, as mixing some or all of these may be difficult.

The table below shows how Expanded I/O is mapped into memory for a CPU with 1K register memory.

- Channel 0 I/O is scanned on the Main I/O chain.
- Expanded discrete references O1+0001 to O7+1024 and I1+0001 to I7+1024 are mapped into the register table.
- Auxiliary discrete references AO0001 to AO1024 and AI0001 to AI1024 are mapped into registers.

REGISTER RANGE	I/O ADDRESSES	REGISTER CONTENTS
R0001 - R0128	AO0001 - AO1024 AI0001 - AI1024	Auxiliary I/O
R0129 - R0256	O1-0001 - O1-1024 I1-0001 - I1-1024	GENIUS I/O point fault status. If enabled, associated with Main Chain I/O, 0001 - 1000 Bus Controller Status Bit Map
R0257		
R0258 - R0266	User Registers	If Genius I/O diagnostics is
R0267 - R-269	Real Time Clock	enabled, can be referenced as:
R0270	Fault Table Pointer	O2-0001 to O2-1000 I2-0001 to I2-1000
R0271 - Rxxxx	Fault Table Entries	through O7-0001 to O7-1000
R0955 - R1024	Computer Mail Box	I7-0001 to I7-1000

Expanded I/O Addressing for 256 Registers

The table below shows how Expanded I/O is mapped into memory for a CPU with 256-word register memory. Auxiliary and Expanded discrete references AO0001 through AO1024, A10001 through AI1024, O1+0001 through O1+1024, and I1+0001 through I1+1024 are mapped into the register table.

REGISTER RANGE I/O ADDRESSES	REGISTER CONTENTS
R0001 - R0128 (AO-0001 - AO-1024) (AI-0001 - AI-1024)	Either Auxiliary I/O or general use registers. If Genius diagnostics is enabled can be referenced as:
R0129 - R0160 (O1+0001 - O1+0512)	
R0161 Bus Controller Status Bit Map	O1+0513
R0162 - R0166 User Registers	
R0167 - R0169 Real Time Clock	to
R0170 Fault Table Pointer	
R0171 - Rxxxx Fault Table Entries	
Rxxxx - R0186 User Registers	O1+1000
R0187 - R0256 Computer Mail Box	(R0193) I1+0001 - I1+1000

Appendix B
Bus Controller Compatibility

This manual describes the operation and use of Phase B Genius I/O Bus Controllers. Basic differences between Phase B Bus Controllers and the earlier Phase A versions are explained below.

Feature	Phase A Bus Controller	Phase B Bus Controller
Power Supply and Rack Requirement	+5V and +12V req. CPU rack or High Capacity I/O Rack	+5 volts only May use High Capacity I/O rack, regular I/O rack, or CPU rack.
Bus Scan Time	Less than 250mS.	Less than 400mS.
Multiple Bus Controllers	Not supported.	Supported.
Bus Controller used as a Monitor.	Not supported.	Bus Controller outputs can be disabled allowing it to monitor inputs and diagnostics from other devices on the bus.
Baud Rate	Uses 153.6 Kbaud only	Selectable baud rates: 153.6 Kbaud standard, 153.6 Kbaud extended, 76.8 Kbaud or 38.4 Kbaud. Permits longer cables and better noise immunity.
Detection of Device Number Conflict	Duplicate Device Numbers not detected.	Tests for another device on bus with the same Device Number as the Bus Controller. If there is a Device Number Number conflict, both LEDs on the Bus Controller will flash in unison. The conflicting Bus Controller must be removed and assigned another Device Number. Then power to the Bus Controller must be cycled.
Powerup Tests	Will turn off both LEDs if communications test fails. Does not test MIT circuits.	Will flash both LEDS in unison if powerup communications test or MIT test fails.
Bus Error Detection	If 1 bus error is detected in a 250mS period, Comm OK LED turns off. Comm OK also goes off if the Bus Controller misses its turn on the bus.	If 10 bus errors are detected in 10 seconds, Comm OK LED goes off. Comm OK also goes off if Bus Controller misses its turn on the bus. Comm OK LED goes back on if no new errors are detected in 250 mS. The bus error count is cleared every 10 sec.
Clear All Faults Command	Clears all faults, and completely clears the fault queue.	Clears all faults, but does not remove Addition or Loss of Block or Bus Controller Address faults from the queue.
Use of 16/32-Circuit Blocks	Updates the input table during the programmer window part of the sweep with latest inputs from blocks. DO I/Os provide no updates.	Updates the input table during the regular I/O portion of the sweep. DO I/Os may be able to capture updates.
Use of Analog Blocks	Updates the I/O table during the programmer window. DO I/Os provide no update; same input, same data until next programmer window.	No change.

Feature	Phase A Bus Controller	Phase B Bus Controller
Number of Analog Blocks and Discrete Blocks on a Bus	If the number of analog blocks plus 16/32 ckt discrete blocks on bus exceeds 8, the program must prevent I/O scanning until the Bus Controller has updated all inputs. Idle DPREQ or WINDOW instruction provides automatic buffer time.	No restriction on the use of discrete blocks on one bus. If the number of analog blocks (of any type) exceeds 8, the program must prevent I/O scanning until the Bus Controller has updated all inputs. See chapter 3 for more information. Idle DPREQ or WINDOW provides automatic buffer time.
Data Coherency	Discrete blocks: 8-bit (1 byte) quantities only. Analog blocks: 16-bit channel values handled as coherent data.	Bus Controller version 1.6 (IC660CBB902G or CBB903G) or later is required to guarantee data coherency for future devices such as the High-speed Counter and PowerTRAC. Discrete blocks: same as phase A. Analog blocks: same as phase A.
Control Data Sent to Inputs-Only Blocks	Does not send control data message to inputs-only blocks. Block I/O Enabled LED goes on when logged in, never goes off.	Sends null message to any inputs-only block on each scan. This turns on I/O Enabled LED. If the block stops receiving this message for 3 scans, its I/O Enabled LED goes off.
Channelized I/O	Use of Expanded I/O channels requires channelized I/O Transmitter cards.	Bus Controller DIP switch used to select Expanded I/O and select channel unless down-stream of channelized I/O Transmitter card.
RTD Block Support	Not supported. Only supports analog blocks with 4 input channels (or fewer).	Supported - maximum number of channels is 128 for any analog block.
Login	Relatively slow. Must process login messages 1 per bus scan.	Fast. Can handle multiple login messages per scan. Phase A blocks won't recognize some of these, and will revert to phase A login procedure, which will go more slowly.
Switch BSM Message	Not supported.	Supports use of the Bus Switching Module and Switch BSM message. Additionally, version 1.5 or later supports BSM switching for BSMs controlled by analog blocks.
Programmed Communications	Specific Datagrams between the Bus Controller and blocks only.	Supports Datagrams and Global Data between Bus Controllers (CPUs) in addition to phase A features. Also Send/Receive Datagram functions for Datagrams without built-in support, or for devices assigned to register memory.
Send/Receive Datagrams	These DPREQ or WINDOW commands not supported. Syntax error.	Fully supported as either DPREQ or WINDOW commands. Used to send or request messages not covered by existing commands. Also used for CPU to CPU communications.

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

W

GE Fanuc Automation North America, Inc., Charlottesville, Virginia