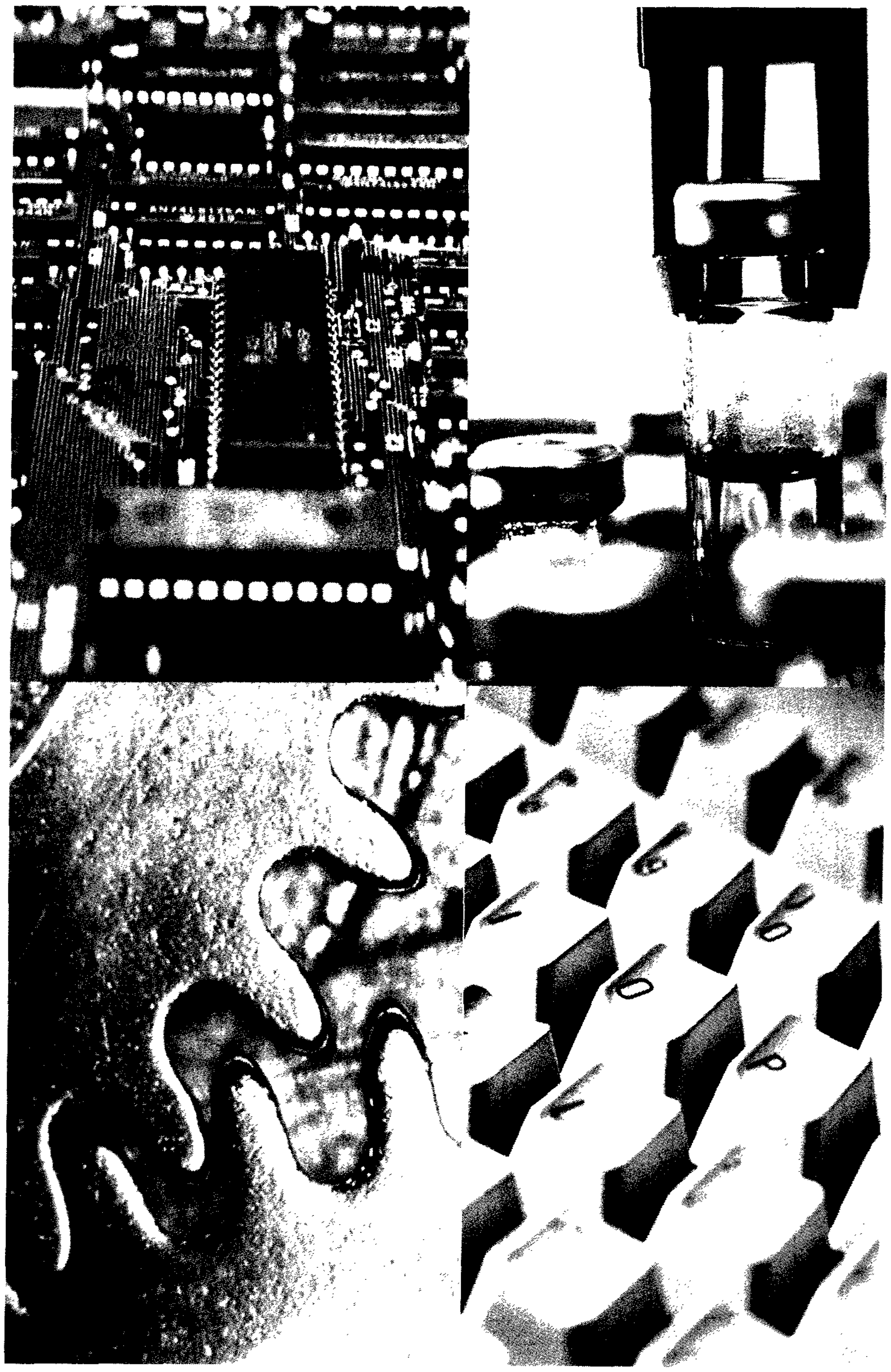
Modicon
984 Hot Standby System Programming Manual

GM-S911-002 Rev. D



GROUPE SCHNEIDER

■ Merlin Gerin ■ Modicon ■ Square D ■ Telemecanique

Modicon 984 Hot Standby System Programming Manual

GM-S911-002 Rev. D

January, 1993

MODICON, Inc., Industrial Automation Systems
One High Street
North Andover, Massachusetts 01845

Preface

⚠ **Caution All pertinent state, regional, and local safety regulations must be observed when installing and using this product.**
For reasons of safety and to ensure compliance with documented system data, repairs to components should be performed only by the manufacturer.

Table of Contents

Figures

Tables

Chapter 1 Introduction

- What a Hot Standby System Does
- Configuring Hot Standby Ladder Logic
- Hot Standby System Software Capabilities

What a Hot Standby System Does

The Hot Standby concept provides fault-tolerant, high-availability programmable control. The system consists of two identical 984 Programmable Controllers configured so that a failure in one transfers system control duties to the other in less than one scan.

Primary and Standby Control

One controller—the *primary*—controls all aspects of the remote 800 Series I/O drops. It constantly scans and solves all the configured segments of ladder logic. The primary controller updates the *standby* controller with the most recent values of the inputs and internal states. Only state memory is transmitted.

Primary and standby states are switchable; either controller in a Hot Standby system can be put in the primary state, but the other must be in the standby state. Remote I/O is always controlled by the primary controller.

The Hot Standby Function Block

The key software component of a Hot Standby System is the loadable hot standby function block—HSBY. The HSBY function block in the primary controller automatically captures a *snapshot* of the 984 state table. This snapshot is transmitted from the primary to the standby through S911 Hot Standby option modules in each controller. This sharing of process information enables the standby to assume control with up-to-date system data in the event of a failure in the primary system.

The HSBY function block lets you reserve several registers for Hot Standby activities. Included in these reseved register areas are a *command* register and a *status* register, which provide a software interface to the Hot Standby System. The command register lets you assume control of the system and override the keyswitch control mechanism. The status register provides you with current system status.

S911 Hot Standby Modules

Each mainframe contains an S911 Hot Standby Module that monitors its own mainframe and communicates with the S911 in the other mainframe.

The timing diagram in Figure 1 shows how the HSBY function blocks in the two CPUs interact with the two S911s to transfer the state table from the primary to the standby controller.

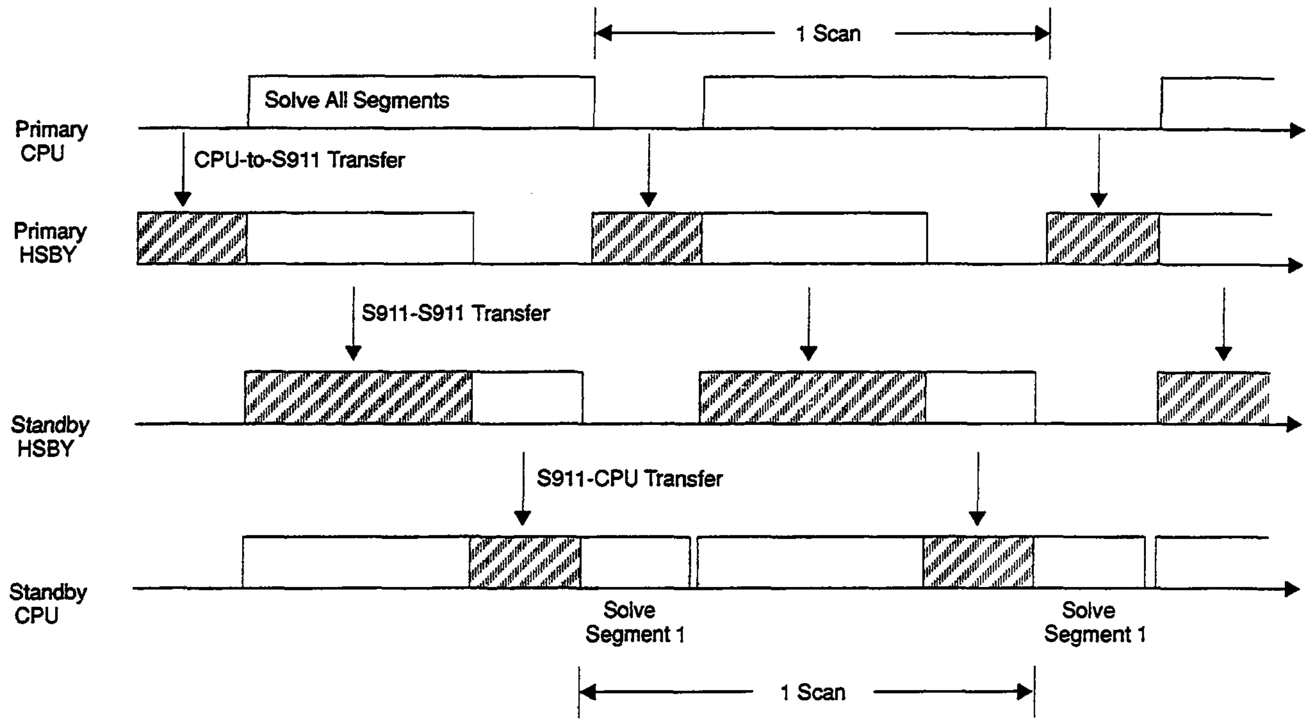


Figure 1 Hot Standby Scan Time Diagram

Hot Standby System Software Capabilities

For the Hot Standby System to function properly, some basic programming tasks must be performed: identical ladder logic must be installed in the primary and standby controllers, all ladder logic must be correct, and an HSBY function block must be loaded into both controllers. Both the primary and the standby units should be configured with a minimum of 1024 words of *0x* ... *4x* references, and the primary unit should contain enough user logic in segments 2 and above to guarantee that the standby controller can complete its tasks by the time the primary returns to begin another scan. An insufficient amount of configured references and user logic can cause system switchover to occur.

Application Control

Application control is through an HSBY function block that lets you influence operations by

- Designating a series of registers in the primary controller—the *nontransfer area*—that do not get transferred to the standby controller
- Specifying whether mismatched logic will fail the standby controller or simply report the mismatch
- Specifying whether the Modbus port addresses are offset between the primary and standby controllers—this mechanism provides an easy scheme for integrating a Hot Standby System into a Modbus network
- Forcing a switchover because of a user-defined fault condition in the primary controller

Manual Control

You can control a 984 Hot Standby System manually with the RUN/OFFLINE keyswitches on the S911 modules in your primary and standby controllers. If you switch the primary controller to OFFLINE, the standby controller changes to primary mode and functions as a standalone controller. If you switch the standby controller OFFLINE, the primary controller functions as a standalone.

You can use these keyswitches to force manual control for maintenance or troubleshooting.

The Effects of Switchover

Most operations in progress during a switchover will be completed or aborted without the unit's being aware that the switchover has occurred. The system that switches to primary mode maintains control over all the operations currently in progress and retains control unless or until another switchover occurs.

Modbus and Modbus Plus transactions or J892 ASCII operations in progress during a switchover typically return an error condition, then complete the transaction on the subsequent retry.

Keep in mind that the default drop holdup time in the Traffic Cop is 300 ms. User logic programs with long scan times and/or configurations with *reset Watchdog Timer* as part of the Segment Scheduler must insure that drop holdup is sufficient, particularly in the event of switchover.

Caution It is imperative that functions controlled by your logic monitor the error outputs and perform retries when an error occurs.

Caution Up to two Modbus Plus messages may be affected during switchover due to protocol restraints. A timeout error or routing failure may be detected by the originator. User logic should be programmed to perform retries on critical applications.

Note A Modbus Plus *No Response* status error cannot be transferred through a BM85 from Modbus Plus to Modbus. A Modbus *No Response* condition could be detected by the Modbus originator.

Configuring Hot Standby Ladder Logic

All ladder logic for *Hot Standby* functions should be in segment 1. Network 1 of segment 1 is reserved exclusively for the HSBY function block and ladder logic directly associated with it.

Segment 1

When your Hot Standby System is running, the primary controller scans all segments while the standby controller scans only segment 1 of the configured ladder logic program. This has three very important implications with respect to the way you configure the system logic:

- You must program all ladder logic specific to Hot Standby functions in segment 1
- You must not program I/O control logic in segment 1
- You must not schedule any I/O drops in segment 1

The standby controller in a Hot Standby System must never execute I/O control logic.

⚠ **Caution To help protect against damage to application I/O devices through unexpected system actions, do not reschedule segment 1 via the Segment Scheduler.**

Segment 1 may contain the ladder logic for diagnostics and optional Hot Standby functions—e.g., time-of-day clock updates or optional system control implementation.

As a minimum, segment 1 in a Hot Standby System must contain the HSBY function block and its associated ladder logic.

Network 1 of Segment 1

Load the HSBY function block to the logic programs in both the primary and standby controllers—in network 1 of segment 1. Then program input and output lines to and from the function block. The two HSBY blocks and all their associated ladder logic inputs must be *identical in both programmable controllers.*

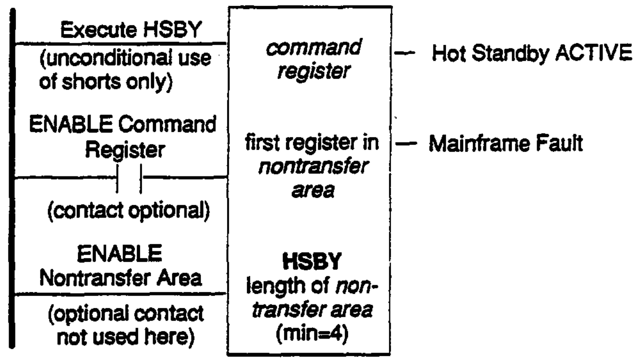
Chapter 2
The HSBY Function Block

The HSBY Function Block

The HSBY function block is a DX loadable ladder logic function that manages a Hot Standby System.

HSBY Block Structure

HSBY is a three-node function block. The block and its associated ladder logic must be programmed in network 1, segment 1 of a program that uses Hot Standby functionality.



Execute HSBY
(unconditional use of shorts only)
ENABLE Command Register
(contact optional)
ENABLE Nontransfer Area
(optional contact not used here)
command register
first register in nontransfer area
HSBY
length of nontransfer area (min=4)
Hot Standby ACTIVE
Mainframe Fault

Figure 2 Hot Standby Function Block

HSBY Input Lines

The top input line, *EXECUTE HSBY*, must be unconditional—i.e., it must be built with *only* horizontal shorts (one or more). It must never contain contacts or other instructions that could cause the function block to disable.

The middle input line, *ENABLE command register*, can be built with one or more horizontal shorts and/or one or more contacts. If your application never requires the command register to be disabled—as is the case most often—you may construct your middle input line with only horizontal shorts. If your application calls for the command register to be disabled at some time, use a contact in the middle input line. If this contact disables the command register, the system functions as if all command register bits are zeros (regardless of the value in the top-node register).

The bottom input line, *ENABLE nontransfer area of state RAM*, can be built with one or more horizontal shorts and/or one or more contacts.

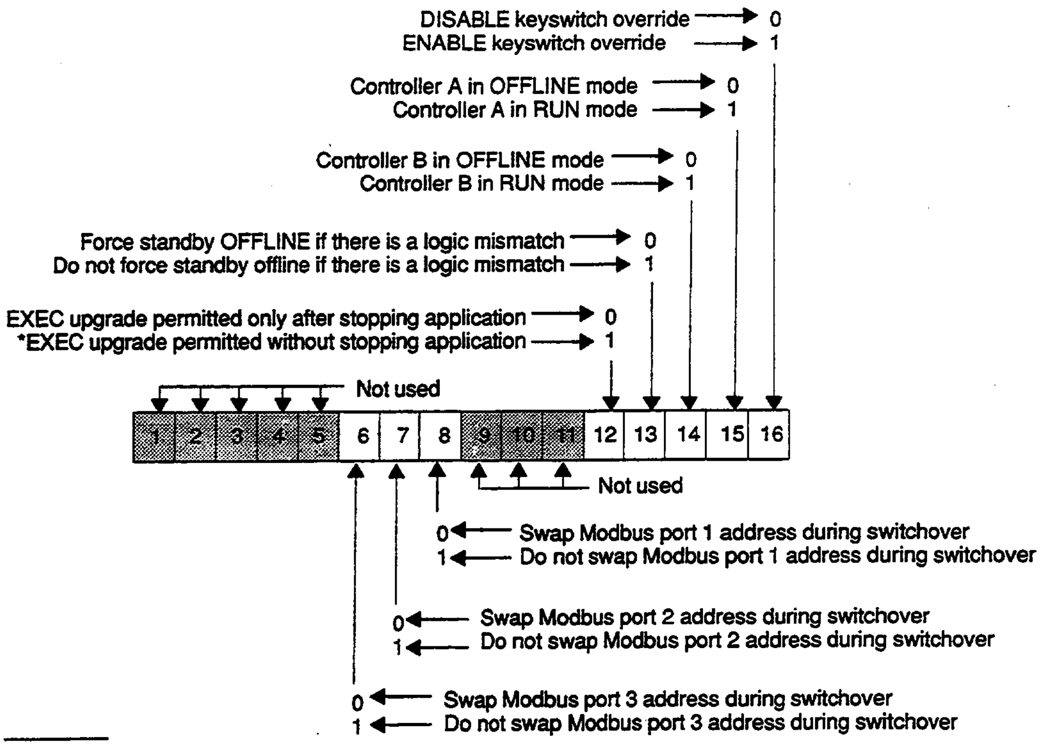
HSBY Output Lines

The top output line, *ACTIVE*, has logic power flow if the S911 hot standby module in this controller is functioning correctly.

The middle output line, *MAINFRAME FAULT*, has logic power flow if the controller cannot communicate with its S911 hot standby module.

The HSBY Command Register

The top node of the HSBY block is a 4*x* holding register that stores the address of the HSBY command register. This command register contains eight bits that let you configure and/or control various Hot Standby functions:



DISABLE keyswitch override → 0
ENABLE keyswitch override → 1
Controller A in OFFLINE mode → 0
Controller A in RUN mode → 1
Controller B in OFFLINE mode → 0
Controller B in RUN mode → 1
Force standby OFFLINE if there is a logic mismatch → 0
Do not force standby offline if there is a logic mismatch → 1
EXEC upgrade permitted only after stopping application → 0
*EXEC upgrade permitted without stopping application → 1
Not used
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Not used
0 ← Swap Modbus port 1 address during switchover
1 ← Do not swap Modbus port 1 address during switchover
0 ← Swap Modbus port 2 address during switchover
1 ← Do not swap Modbus port 2 address during switchover
0 ← Swap Modbus port 3 address during switchover
1 ← Do not swap Modbus port 3 address during switchover

*Requires Rev. C PROMS on the S911 Modules.

Figure 3 HSBY Command Register

You can configure and/or control the command register bits through a standard 984 programming panel (a P190, a P230, or a standard DOS-based PC). As an alternative, you can access the command register bits for configuration and/or control with optional system control techniques (see Chapter 4 for details).

Optional System Control

Bit 16 in the HSBY command register lets you select RUN and OFFLINE modes with software instead of with the keyswitches on the S911 modules in each controller. If bit 16 = 1, you must select RUN or OFFLINE modes through bits 15 and 14. If bit 16 = 0, you must use the S911 keyswitches to select RUN and OFFLINE modes.

If bit 16 = 1, you must use bit 15 to select either the RUN (bit 15 = 1) or OFFLINE (bit 15 = 0) mode for controller A (set by the S911 toggle switch). Bit 15 is ignored if bit 16 = 0.

If bit 16 = 1, you must use bit 14 to select either the RUN (bit 14 = 1) or OFFLINE (bit 14 = 0) mode for controller B (set by the S911 toggle switch). Bit 14 is ignored if bit 16 = 0.

System Reactions to Mismatched Ladder Logic

Bit 13 in the HSBY command register lets you determine how the Hot Standby System will react when it detects mismatched ladder logic in the primary and standby controllers. If bit 13 = 1, the standby continues to function in the RUN mode when the logic checksum does not match. If bit 13 = 0, the standby controller switches to the OFFLINE mode if mismatched logic is encountered. If the mismatch is corrected, the standby unit will return to the standby mode.

Note Setting bit 13 = 1 allows only user logic to be different in the primary and standby units. Configurations in the units must be the same for proper operation.

Upgrading a System EXEC Without Stopping the Application

If you set bit 12 in the HSBY command register to 1, you can upgrade system EXECs without having to stop the application. You need only power down the standby controller to change its EXEC pack; the primary controller continues to run as a standalone. This capability requires that the S911s in your Hot Standby System have Rev. C or higher PROMs.

STOP **Warning Reset bit 12 to 0 as soon as you have completed the EXEC upgrade—*do not leave bit 12 set to 1.* To enable the functionality of the exchange bit, normal checks for validity of the standby controller are ignored. Without these checks in place, the standby unit can remain online when critical parameters in the primary and standby units are not the same. Because of this, unplanned I/O points can turn ON and OFF in the event of switchover. These unplanned actions could result in injury to personnel, disruption of system operation, or damage to plant equipment.**

Modbus Port Addressing

Bits 8, 7, and 6 in the HSBY command register let you determine whether the Modbus port addresses in the primary and standby controllers are the same or offset. Bit 8 handles Modbus port 1, bit 7 handles Modbus port 2, and bit 6 handles Modbus port 3. (Modbus II uses the address from Modbus port 3.)

If one of these bits = 1, the corresponding Modbus port addresses are the same in the primary and standby controller. If the bit = 0, the Modbus port in the standby controller is set to an address that is 128 greater than the Modbus port address in the primary.

In the event of a switchover, the addresses swap between controllers. Because of this, a Modbus master can always access the primary controller.

For example, if bit 8 = 0 and the Modbus port 1 address of the primary controller is 1, then the Modbus port 1 address of the standby controller is 129.

Note If you change the states of bits 8, 7, or 6 while the system is running, the Modbus port address offsets are not entered or cancelled until a switchover occurs.

Table 1 shows the bit values and the Modbus port addresses they represent.

Table 1 Bit Values for Modbus Port Addresses

Bit	Bit Value	Primary Address	Standby Address
8	0	a	a + 128
8	1	a	a
7	0	b	b + 128
7	1	b	b
6	0	c	c + 128
6	1	c	c

a, b, and c are independent whole number values between 1 and 119

Modbus Plus Port Addressing

If you are using two controllers with Modbus Plus in a Hot Standby system, you must set the address switches on both controllers to the same network address. When the network is in active operation, both controllers will become active and addressable on the network. The primary controller will use the network address you have assigned, and the standby will assume an address that is offset by 32 (decimal) from the primary address, within the range 1 ... 64. The standby offset will be either 32 addresses higher or lower than the primary's network address within the 1 ... 64 range.

For example, if you set the address switches on both units to 10, the primary unit will function at address 10 and the standby will function at address 42. If you set the address switches on both units to 40 (a +32 offset), the primary unit will function at address 40 and the standby will function at address 8 (a –32 offset).

The two controllers automatically exchange addresses if switchover occurs—this maintains consistency in your application, as the current primary controller is always accessible at the same address regardless of which physical unit is serving as the the primary.

When you set the address switches, you must avoid duplication with other Modbus Plus nodes at both the primary and the standby offset addresses. For example, if you set the address switches to 10, you must make sure that there will be no addressing conflicts at both address 10 and at address 42 on the network.

Because Modbus Plus treats the two controllers as distinct nodes on the network, the application is able to interrogate either controller independently in order to acquire statistics.

☞ **Note** The two Hot Standby controllers must both be running before they are physically connected to a Modbus Plus network.

Setting Up a Nontransfer Area in State RAM

Within the HSBY function block, you must program a *nontransfer area* that will be reserved in the programmable controller's state RAM for Hot Standby functionality. The nontransfer area protects a serial group of registers in the standby controller from being modified by the primary controller—e.g., the standby STAT block area. You may be able to reduce overall system scan time by defining and enabling a certain amount of nontransfer area.

When you define and enable nontransfer area, you set aside a register block that does not get transferred to the standby controller every scan. Hot Standby Systems inevitably increase the system scan time—using nontransfer area can help minimize scan time increase.

Enabling the HSBY Nontransfer Area

The middle node of the HSBY block contains the 4*x* holding register that is the first of several consecutive registers in the nontransfer area. The bottom node of the HSBY block contains a number representing the length of the nontransfer area—i.e., the total number of consecutive 4*x* registers—reserved in the state table.

The minimum nontransfer length in all cases is four words. The maximum nontransfer length for a 16-bit CPU (the 984A, 984X, or 984-680) is 255 words. The maximum length for a 24-bit CPU (the 984B, the 984-780, or -785) is 8000 words.

Special Registers in the Nontransfer Area

The first three registers in the nontransfer area are reserved for special purposes:

- 4*x* is the first of two consecutive reverse transfer registers—the HSBY function block uses these two registers to pass information from the standby to the primary controller
- 4*x* + 1 is the second of two consecutive reverse transfer registers
- 4*x* + 2 is the HSBY status register

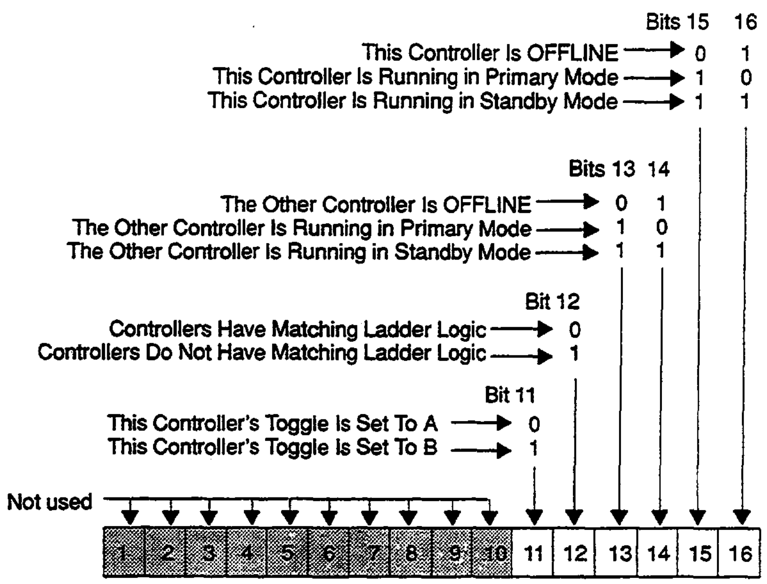
The actual nontransfer area begins at register 4*x* + 3.

Implementing the Reverse Transfer Registers
When you enable an HSBY nontransfer area, references 4*x* and 4*x* + 1 are copied from the standby controller to the primary controller—this is opposite from the normal *forward* state table transfer, which is from the primary to the standby. You can use reverse transfer registers to transmit diagnostic data from the standby controller to the primary controller.

Remember that the standby controller solves logic in all the networks in segment 1. Use network 2 (or greater) of segment 1 for ladder logic that loads diagnostic data to the reverse transfer registers. The primary controller, which scans and solves the logic in all segments, can then monitor the data in the reverse transfer registers.

The HSBY Status Register

The HSBY status register contains six bits—bits 11 ... 16—that are coded to describe the current status of the primary and standby controllers:



Bits 15 16
This Controller Is OFFLINE 0 1
This Controller Is Running in Primary Mode 1 0
This Controller Is Running in Standby Mode 1 1
Bits 13 14
The Other Controller Is OFFLINE 0 1
The Other Controller Is Running in Primary Mode 1 0
The Other Controller Is Running in Standby Mode 1 1
Bit 12
Controllers Have Matching Ladder Logic 0
Controllers Do Not Have Matching Ladder Logic 1
Bit 11
This Controller's Toggle Is Set To A 0
This Controller's Toggle Is Set To B 1
Not used
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Figure 4 HSBY Status Register Bit Functions

You can use ladder logic to monitor the six bits in the HSBY status register:

- The combination of bit patterns in bits 15 and 16 in the HSBY status register tell you whether the controller you are attached to is in the primary, standby, or OFFLINE mode
- The combination of bit patterns in bits 13 and 14 in the HSBY status register tell whether the other controller is in the primary, standby, or OFFLINE mode
- Bit 12 in the HSBY status register tells you whether or not the ladder logic in the two controllers is identical
- Bit 11 in the HSBY status register tells you whether the S911 module in a controller has its toggle switch set to position A or B

Status data can be monitored through panel software, Modbus, or I/O.

A Reverse Transfer Example

The example in Figure 5 shows I/O ladder logic for a primary controller that monitors two fault lamps and the reverse transfer logic that sends status data from the standby controller to the primary controller.

The Application

One fault lamp turns ON if the standby memory protect is OFF; the other lamp turns ON if the memory backup battery fails in the standby.

Internal coil bit 00715 (status bit 11) controls the STANDBY MEMORY PROTECT OFF lamp. Internal coil bit 00716 (status bit 12) controls the STANDBY BATTERY FAULT lamp.

Reverse Transfer Logic

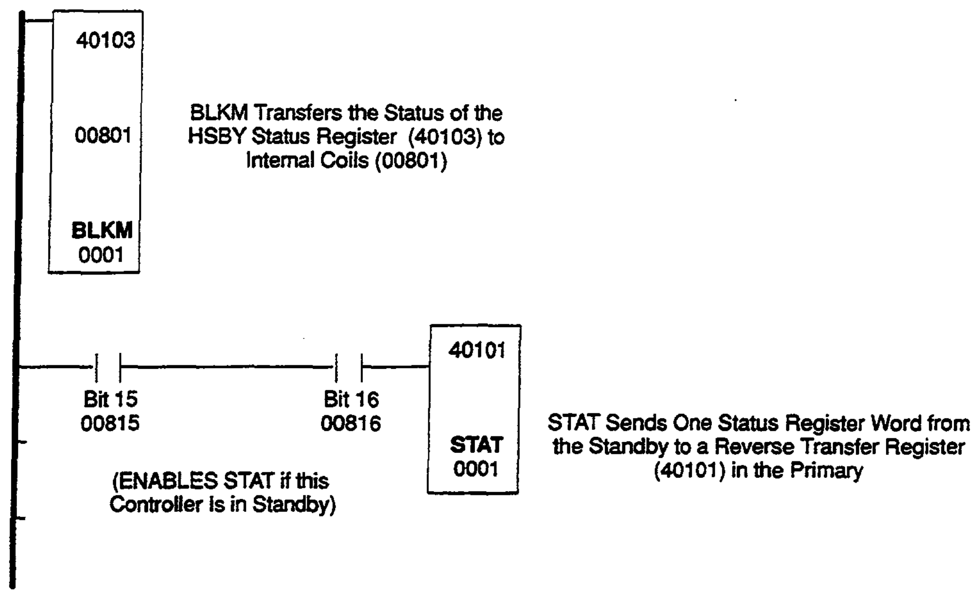
The logic at the top of Figure 5 is in network 2 of segment 1. This network contains a Block Move (BLKM) function and a System Status (STAT) function. The standby controller enables the STAT block. Bits 00815 and 00816 are controlled by bits 15 and 16 in the HSBY status register.

The STAT block sends one status register word to 4yyyy; this word initiates a reverse transfer to the primary controller.

Remote I/O Logic

The logic at the bottom of Figure 5 must be in segment 2 or greater, and therefore only the primary controller scans it. Bits 00813 and 00814 enable the BLKM block that transfers status data to the internal coils at reference 00705.

This Logic Must Be in Network 2 of Segment 1


40103
00801
BLKM
0001
BLKM Transfers the Status of the HSBY Status Register (40103) to Internal Coils (00801)
Bit 15
00815
Bit 16
00816
40101
STAT
0001
STAT Sends One Status Register Word from the Standby to a Reverse Transfer Register (40101) in the Primary
(ENABLES STAT if this Controller is in Standby)

This Logic Cannot Be in Segment 1—It Must Be in Another Segment

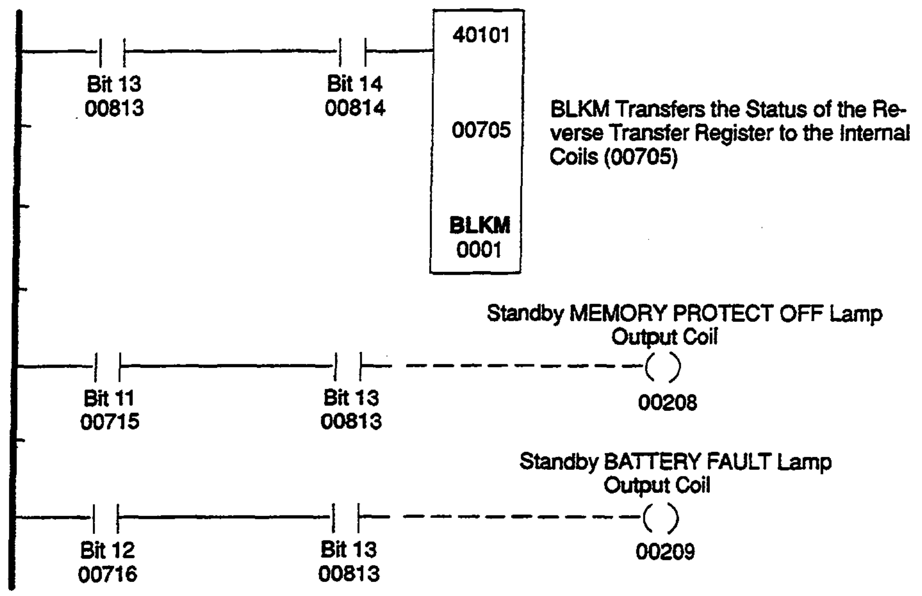

Bit 13
00813
Bit 14
00814
40101
00705
BLKM
0001
BLKM Transfers the Status of the Reverse Transfer Register to the Internal Coils (00705)
Standby MEMORY PROTECT OFF Lamp Output Coil
Bit 11
00715
Bit 13
00813
00208
Standby BATTERY FAULT Lamp Output Coil
Bit 12
00716
Bit 13
00813
00209

Figure 5 Reverse Transfer Example

Synchronizing Time-of-day Clocks

In a Hot Standby System, the primary and standby controllers have their own time-of-day clocks, and they are *not* synchronized. At switchover, the time of day changes by the difference between the two clocks. This could cause problems when you are controlling a time-critical application.

☞ **Note** In Hot Standby Systems, lockout will occur on the first scan only after switch-over. This is because real time clocks are not synchronized between primary and standby controllers. The dissimilarity between clocks will cause the loop to reset itself and implement the new (formerly standby) value in the algorithm solution.

Solving the Synchronization Problem

The problem of unsynchronized time-of-day clocks can be solved by programming the standby controller to reset its clock from the state table provided by the primary controller.

Put the logic for time synchronization in segment 1, but do not put it in network 1. Network 1 must contain *only* the HSBY function block.

Since both controllers run the same program, you must read HSBY status register bits 12 ... 16 to make sure that only the standby clock is resetting. If bits 12 ... 16 = 01011, you know three things:

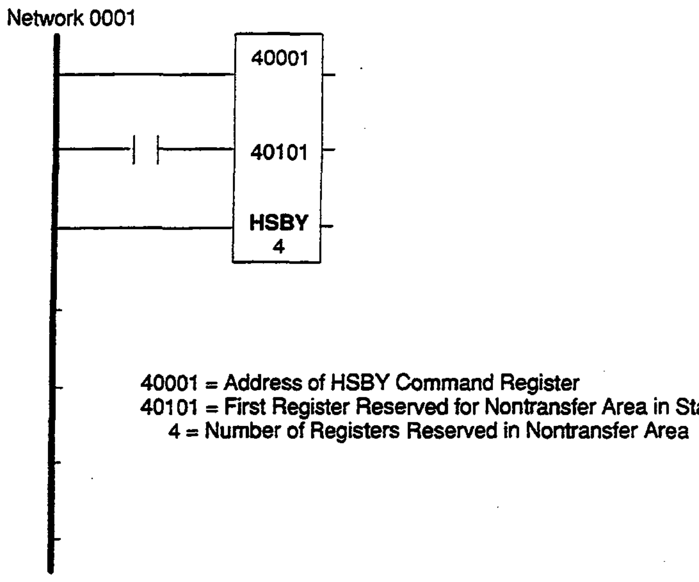
- Which controller is the standby
- That the remaining controller is the primary
- That both controllers are running the same logic

If these conditions are true, then the logic should clear bit 2 and set bit 1 of the time-of day control register.

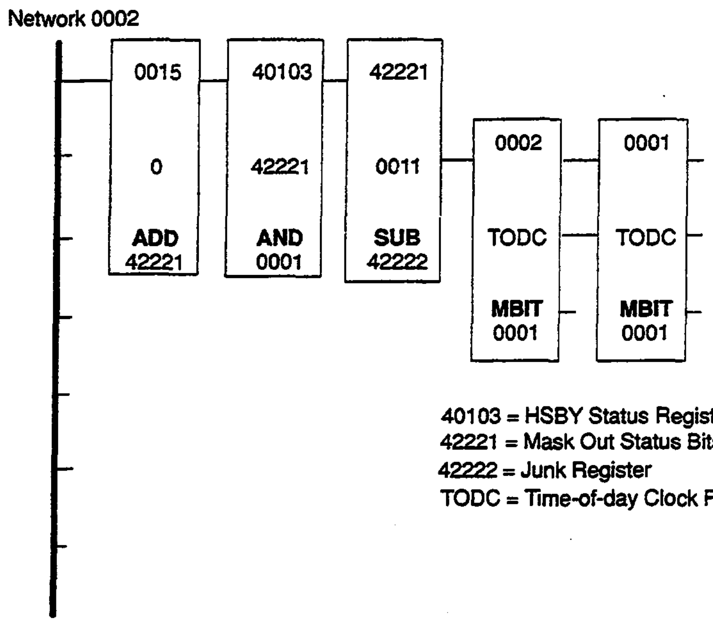
For more information, refer to the time-of-day clock discussion in the *984 Programmable Controllers Systems Manual* (GM-0984-SYS).

The clock in the standby controller will be reset from the state table of the primary controller at the end of a scan, and bit 1 will be cleared.

Figure 6 shows sample logic for synchronizing time-of-day clocks:



40001 = Address of HSBY Command Register
40101 = First Register Reserved for Nontransfer Area in State RAM
4 = Number of Registers Reserved in Nontransfer Area



40103 = HSBY Status Register
42221 = Mask Out Status Bits Not Required
42222 = Junk Register
TODC = Time-of-day Clock Register

Figure 6 Logic for Synchronizing Time-of-day Clocks

Chapter 3
Hot Standby Scan Time

Two Factors that Influence Hot Standby Scan Time

A Hot Standby capability necessarily adds more scan time to a comparable standalone 984 controller. The way you program your HSBY function block and I/O ladder logic can help to minimize this increased scan time.

Minimizing the Scan Time Increase

The amount of scan time increase varies according to two things:

- The size of the state RAM table
- The length of the nontransfer area in the controller's state RAM, as defined in the HSBY function block

While programming your Hot Standby System, you can help reduce the increase in scan time by not utilizing unnecessary area in the state table and by defining as much nontransfer area as possible.

Contents of the State RAM Table

The state RAM table contains all current I/O references (0*x* coils, 1*x* discrete inputs, 3*x* register inputs, and 4*x* holding registers), history bits, and DISABLE tables. You can minimize the time it takes to transfer state data by defining the state table with the smallest possible amount of state RAM area. Do not use all available state RAM unless your particular application requires it.

If you have16-bit CPUs in your Hot Standby system (a pair of 984A, 984X, 984-680, or -685 Programmable Controllers), the largest possible state RAM area is 3K words. If you have 24-bit CPUs in your Hot Standby System (a pair of 984B, 984-780, or -785 Programmable Controllers), the largest possible state RAM area is 12.8K words.

Nontransfer Area

You can minimize the time it takes to transfer the state table by defining the largest possible nontransfer area—and therby reducing the amount of state RAM data that does transfer.

If you have 16-bit CPUs in your Hot Standby system (a pair of 984A, 984X, 984-680, or -685 Programmable Controllers), the largest possible nontransfer area you can define is 255 words. If you have 24-bit CPUs in your Hot Standby system (a pair of 984B, 984-780, or -785 Programmable Controllers), the largest possible nontransfer area you can define is 8000 words.

⚠ **Caution   Make sure the nontransfer area you define contains no critical data. No data in the nontransfer area is transferred to the standby controller, and all data in the nontransfer area is lost at switchover.**

The Effects of State Table Transfer on Scan Time

A Hot Standby System transfers state table data from the primary controller to the standby controller while the primary controller scans and solves I/O ladder logic. There are two steps in this transfer process—CPU-to-S911 transfer and S911-to-S911 transfer.

CPU-to-S911 State Table Transfer

As soon as the primary controller executes its HSBY function block, it starts transferring the current state table from its CPU to its S911 Hot Standby module. Ladder logic scanning and I/O servicing halt while this data transfer takes place.

CPU-to-S911 state transfer takes 1.3 ms/K words that transfer—add another 3 ms for overhead. As soon as the CPU-to-S911 transfer finishes, the primary controller resumes scanning ladder logic and servicing I/O.

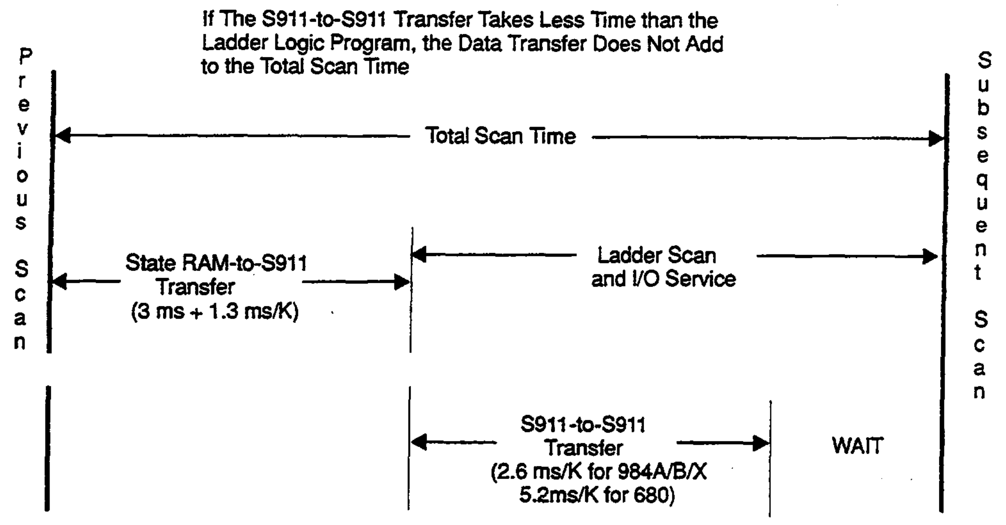
S911-to-S911 State Table Transfer

As the ladder logic scan resumes, the primary controller starts to simultaneously transfer the state table from its S911 to the S911 in the standby controller through the W911 cable link.

If your Hot Standby System uses chassis mount (984A, 984B, or 984X) Programmable Controllers, the S911-to-S911 transfer time is 2.6 ms/K words. If your Hot Standby system uses slot mount (984-680, -685, -780, or -785) Programmable Controllers, the S911-to-S911 data transfer takes 5.2 ms/K words.

When the I/O ladder logic program being scanned by the primary controller is longer than the S911-to-S911 data transfer, the transfer does *not* increase total scan time. However, if your logic ladder program is relatively short, the ladder scan will finish before the S911-to-S911 data transfer. In this case, S911-to-S911 data transfer increases total scan time.

Figure 7 shows the possible effects of state transfer time on total system scan time.


If The S911-to-S911 Transfer Takes Less Time than the Ladder Logic Program, the Data Transfer Does Not Add to the Total Scan Time
Previous Scan
Total Scan Time
State RAM-to-S911 Transfer (3 ms + 1.3 ms/K)
Ladder Scan and I/O Service
S911-to-S911 Transfer (2.6 ms/K for 984A/B/X 5.2ms/K for 680)
WAIT
Subsequent Scan

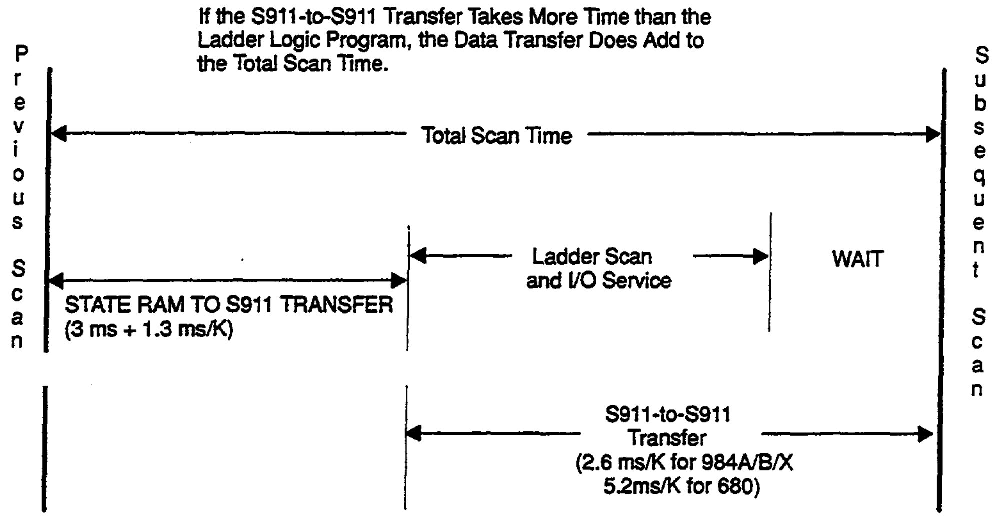

If the S911-to-S911 Transfer Takes More Time than the Ladder Logic Program, the Data Transfer Does Add to the Total Scan Time.
Previous Scan
Total Scan Time
Ladder Scan and I/O Service
WAIT
STATE RAM TO S911 TRANSFER (3 ms + 1.3 ms/K)
S911-to-S911 Transfer (2.6 ms/K for 984A/B/X 5.2ms/K for 680)
Subsequent Scan

Figure 7 Hot Standby System Scan Times

Chapter 4
Optional System Control Techniques

Using Software for Optional System Control

You can monitor your Hot Standby System through bits in the HSBY status register (register 3 in the nontransfer area table specified in the middle node of the HSBY function block). You can control various Hot Standby functions with bits in the HSBY command register.

Logic for Optional Monitoring Indicators

You can design ladder logic that monitors any or all of the six bits in the HSBY status register. Status data can then be read by panel software, Modbus, or I/O.

Logic for Remote Control of Keyswitch Functions

You can also design ladder logic to monitor inputs from Modbus or I/O, and thereby control any or all of the seven bits in the HSBY command register. You can use these bits to remotely control functions usually performed by the keyswitches on the S911 Hot Standby modules. For example, by remotely manipulating command register bits 14, 15, and 16, you can take the current primary controller OFFLINE and thereby force switchover.

The Value of Optional System Control

Optional system control is very useful in situations where the Hot Standby controllers are positioned in hard-to-get-to places. Optional system control can also be used to automatically recognize a fault and force a switchover under conditions not handled by a standard hot standby configuration—e.g., faults in option processors such as a C986/C996 Coprocessor or an S975 Modbus II.

An Optional System Control Panel Example

An example of the ladder logic for an optional system control panel is shown on the following pages. The logic is based on eight assumptions:

- RUN or OFFLINE modes are specified via keyswitches on an optional control panel
- RUN/OFFLINE control capabilities are transferred to the optional control panel via bit 16 in the command register—bit 16 is connected to input 10413
- RUN/OFFLINE control of Controller A (command register bit 15) is through input 10414
- RUN/OFFLINE control of Controller B (command register bit 14) is through input 10415
- You want offsets of 128 between both the Modbus addresses in ports 2 and 3 of the primary and standby controllers; you do not want any offset between the Modbus port 1 addresses in the primary and standby controllers
- 41009 is the command register reference

- 42001 is the start of the nontransfer area in state RAM—i.e., it is specified in the middle node and is reserved as the first of two reverse transfer registers
- A FAULT lamp (output 00207) on the optional control panel will turn ON if the stand-by controller goes OFFLINE.

Figure 8 shows the HSBY block set-up for this example:

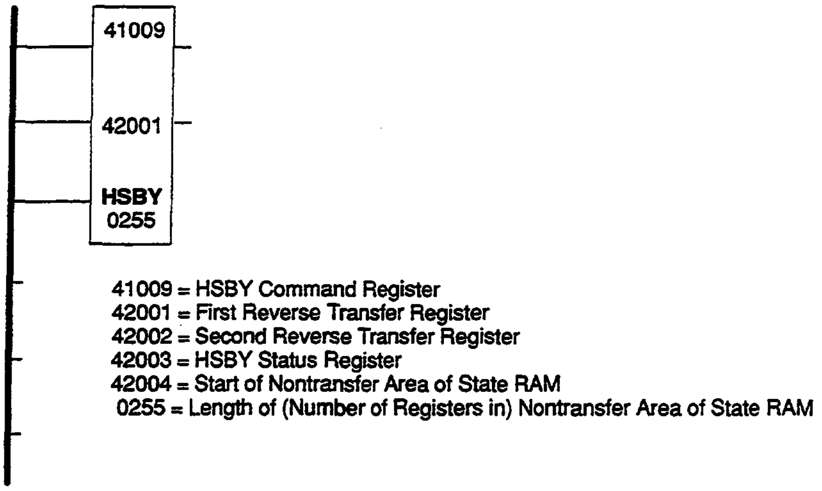

41009
42001
HSBY
0255
41009 = HSBY Command Register
42001 = First Reverse Transfer Register
42002 = Second Reverse Transfer Register
42003 = HSBY Status Register
42004 = Start of Nontransfer Area of State RAM
0255 = Length of (Number of Registers in) Nontransfer Area of State RAM

Figure 8 HSBY Block for an Optional Control Panel

Since you do not want to leave command register bits 16, 15, and 14 at 0, you must control them through ladder logic, as shown in Figure 9. You can control bits 6 and 7 through ladder logic or set them through the 984 programmer tape (or disk file).

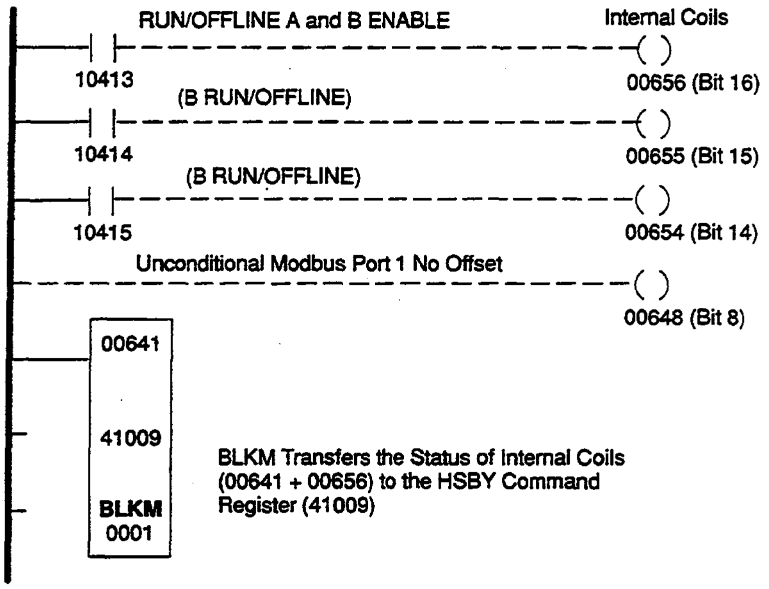

RUN/OFFLINE A and B ENABLE
Internal Coils
10413
00656 (Bit 16)
(B RUN/OFFLINE)
10414
00655 (Bit 15)
(B RUN/OFFLINE)
10415
00654 (Bit 14)
Unconditional Modbus Port 1 No Offset
00648 (Bit 8)
00641
41009
BLKM
0001
BLKM Transfers the Status of Internal Coils (00641 + 00656) to the HSBY Command Register (41009)

Figure 9 Example of Input Logic for an Optional Control Panel

Option Processors in a 984 Hot Standby System

The *soft control* capability of the 984 Hot Standby System allows the ladder logic to automatically override the S911 keyswitches and force a transfer of control to the active standby unit. This operation, although not common in typical applications, is necessary to support redundancy while using other 984 option processors.

Other 984 Option Processors

Other 984 option processors include the S975 Modbus II, the S985 Modbus Plus Interfaces, and the C986/C996 Coprocessors.

Forcing a Soft Control Switchover

A 984 Hot Standby System will not automatically switch over in the event of an option processor failure.

To force a controller OFFLINE in the event of a fault in one of its option processors, the logic shown in Figure 10 should exist in the controller. We recommend that identical logic exist in both controllers so that fault detection can be supported in both.

If the primary controller is forced OFFLINE, the standby will take control of the system; if the standby is forced OFFLINE, the primary will continue to operate as a standalone controller.

Make sure that bits 1 ... 13 are preset in 40200 (the HSBY command register) before running the system.

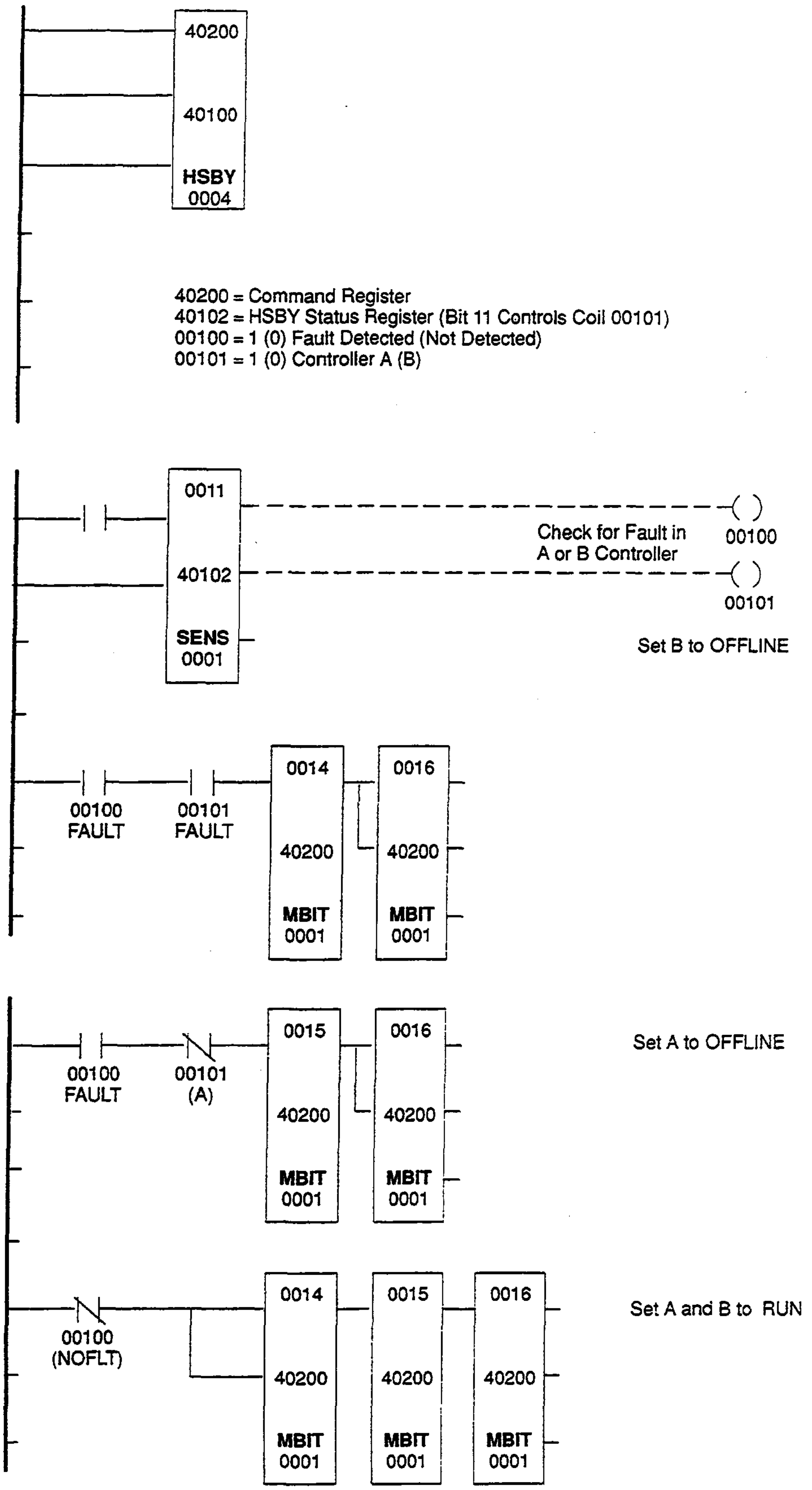


Figure 10 Switchover Logic for an Option Processor

Chapter 5
Programming a New Hot Standby System

Follow the programming procedures in this chapter to install a new Hot Standby System. These procedures assume that you have properly installed your 984 Hot Standby System according to the *984 Hot Standby System Installation and Maintenance Manual* (GM-S911-001).

If you are upgrading an existing 984 controller to give it Hot Standby capability, refer to Chapter 6.

Hot Standby Programming Procedures

Procedure **Programming a New Hot Standby System**

☞ **Note** Software label keys are shown in brackets [ ], and other keys are shown in **bold** type.

Step 1 Select either controller, and connect the programming panel—e.g., P190, P230—to Modbus port 1. Place the S911 keyswitch on the controller in the RUN position. Place the keyswitch on the other controller in the OFFLINE position.

Step 2 Load the 984 Configurator tape or disk file. Make sure the controller is in STOP mode before proceeding.

Step 3 Use the software label keys to configure the controller for your application. Then use [MODULES] to load the HSBY loadable module from tape or disk file. Then write the configuration to the controller using [WRITE CONFIG].

Step 4 Load the 984 Traffic Cop tape or disk file. Use the software label keys to configure the 800 Series I/O addressing.

☞ **Note** Local I/O capabilities—such as those available in the 9894X and the 984-680/-685/-780/-785 Controllers—are not supported in Hot Standby Systems.

Step 5 Load the 984 Programmer tape or disk file.

Step 6 Enter the HSBY DX function block into network 1 of segment 1. Optionally, you may program ladder logic in network 2 of segment 1, if that logic is not used to control I/O. If the HSBY function block does not appear in the various DX menus, return to step 3.

☞ **Note** Make sure that only horizontal shorts are used in the *Execute HSBY* input line to the HSBY function block. If contacts of any kind are used that cause the function block to disable, the integrity of the Hot Standby System will be upset.

☞ **Note** You may set or reset certain bits in the HSBY command register through the [FULL REFERENCE] screen of your programmer.

Step 7 Enter application ladder logic for segment 2 and all other segments.

Step 8 Start the selected controller and debug the application program. Make sure all application devices operate according to design.

Step 9 Load the 984 Tape Loader tape or disk file. Record your application program with a [DUMP 984] operation.

Step 10 Connect the programming panel to Modbus port 1 on the other controller. Make sure the controller is in the STOP mode before proceeding.

Step 11 Load the application program into this controller with a [LOAD 984] operation.

Step 12 Start the controller.

Step 13 Turn the keyswitch on the S911 to RUN. Verify the proper LED indications against Table 2.

Table 2 Hot Standby Normal Indications

Module	Led Indicator	Primary Controller	Standby Controller
S911	READY	ON	ON
	COMM ERROR	OFF	OFF
	PRIMARY	ON	OFF
	STANDBY	OFF	ON
S908/S929	READY	ON	ON
	COMM ACTIVE	ON	Blinking
	COMM ERROR	OFF	OFF
C916/C924	RUN	ON	ON
	READY/SAFE 84	OFF	OFF
984-680/-685/ -780/-785	RUN	ON	ON
	READY	ON	ON
	POWER OK	ON	ON

Your new 984 Hot Standby System is now set up. For further operating instructions, refer to the *984 Hot Standby System Installation and Maintenance Manual* (GM-S911-001).

Chapter 6
Programming a Retrofit Hot Standby System

Follow the programming procedures in this chapter to upgrade an existing standalone 984 controller to a Hot Standby System. These procedures assume that you have properly installed your 984 Hot Standby System according to the *984 Hot Standby System Installation and Maintenance Manual* (GM-S911-001).

If you are installing a new Hot Standby System, refer to Chapter 5.

☞ **Note** Local I/O capabilities—available in the 984X chassis mount and in the 984-680/-685/-780/-785 slot mount Controllers—are not supported in Hot Standby Systems.

Retrofit Hot Standby Programming Procedures

Procedure **Programming a Retrofit Hot Standby System**

☞ **Note** Software label keys are shown in brackets [ ], and other keys are shown in **bold** type.

Step 1 Connect a programming panel—e.g., a P190 or a P230—to Modbus port 1 on the original controller, which contains your application program. Turn the keyswitch on its S911 Hot Standby module to the RUN position. Make sure that the keyswitch on the other controller is in the OFFLINE position.

Step 2 Load the 984 Tape Loader tape or disk file. Record your application program with a [DUMP 984] operation.

Step 3 Load the 984 Configurator tape or disk file. Make sure the controller is in the STOP mode before proceeding.

Step 4 Use [MODULES] to load the HSBY loadable module from tape or disk file. Write the configuration to the controller using [WRITE CONFIG].

Step 5 Load the 984 Traffic Cop tape or disk file. Follow the appropriate software label keys to configure the 800 Series I/O addressing. (The Traffic Cop is lost when the system is reconfigured.)

Step 6 Load the 984 Tape Loader tape or disk file. Reload your application program with a [RELOCATE LOGIC] operation.

Step 7 Load the 984 Programmer tape or disk file.

Step 8 Insert the HSBY DX function block into network 1 of segment 1. All existing logic automatically moves to the next network of segment 1. If the HSBY function block does not appear in the various DX menus, return to Step 3.

☞ **Note** Make sure that only horizontal shorts are used in the top (*Execute HSBY*) input line to the HSBY function block. If contacts of any kind are used that cause the function block to disable, the integrity of the Hot Standby System will be upset.

☞ **Note** You may set or reset certain bits in the HSBY command register through the [FULL REFERENCE] screen of your programmer.

Step 9 Start the selected controller and debug the application program. Make sure all application devices operate according to design.

Step 10 Load the 984 Tape Loader tape or disk file. Record your application program with a [DUMP 984] operation. This copy will supersede your previous backup copy.

Step 11 Connect theprogramming panel to the Modbus port 1 of the other controller. Make sure the controller is in the STOP mode before proceeding.

Step 12 Load the application program into this controller with a [LOAD 984] operation.

Step 13 Start the controller.

Step 14 Turn the keyswitch on the S911 to RUN. Verify the proper LED indications against Table 3.

Table 3 Hot Standby Normal Indications

Module	Led Indicator	Primary Controller	Standby Controller
S911	READY	ON	ON
	COMM ERROR	OFF	OFF
	PRIMARY	ON	OFF
	STANDBY	OFF	ON
S908/S929	READY	ON	ON
	COMM ACTIVE	ON	Blinking
	COMM ERROR	OFF	OFF
C916/C924	RUN	ON	ON
	READY/SAFE 84	OFF	OFF
984-680/-685/ -780/-785	RUN	ON	ON
	READY	ON	ON
	POWER OK	ON	ON

Your new 984 Hot Standby System is now set up. For further operating instructions, refer to the *984 Hot Standby System Installation and Maintenance Manual* (GM-S911-001).

Chapter 7
On-Line Editing

After you have put your Hot Standby System in operation, you may occasionally need to edit the ladder logic to modify application functions. This chapter considers editing procedures for Hot Standby Systems. First, consider three general guidelines for on-line editing:

- Changes in ladder logic can make the control system behave differently. Always use care when making on-line editing changes.
- We recommend that one Modbus port be set aside for programming and not used for networking—refer to Chapter 2 regarding Modbus port address handling. If you are unable to use a dedicated Modbus port for programming, make sure that you are aware of the individual Modbus port addresses at all times.
- You can make minor changes to ladder logic while the system is on-line and controlling your application, but if you need to make *major* changes, we recommend that you shut down the application first.

⚠ **Caution If you are changing the 984 Configuration or Traffic Cop, you must first shut down your control system.**

On-line Editing Procedures with Modbus

Procedure **Editing Ladder Logic on a Hot Standby System**

Step 1 Verify that both controllers are running, one in the primary mode and one in the standby mode.

Step 2 Connect a programming panel—e.g., a P190 or P230—to a Modbus port on the *primary* controller—which is running application logic.

Step 3 Load the 984 Tape Loader tape or disk file. Do not stop the controller before proceeding. Record your application program with a [DUMP 984] operation.

Step 4 Place the S911 keyswitch on the *standby* controller in the OFFLINE position.

☞ **Note** The primary controller will function as a standalone until the editing task is complete.

Step 5 Load the 984 Programmer tape or disk file.

Step 6 Carefully edit the selected networks of the primary controller according to design requirements.

⚠ **Caution To help protect against damage to application I/O devices, use care in editing application logic. Untested logic changes can have unexpected consequences.**

Step 7 Verify that the application responds correctly to the changes.

Step 8 Repeat steps 6 and 7 until all changes are complete.

Step 9 Load the 984 Tape Loader tape or disk file. Record your application program with a **DUMP 984** operation. This copy will supersede your previous backup copy.

Step 10 Connect the programming panel to the *standby* controller at Modbus port 1. Make sure the controller is in the STOP mode before proceeding.

Step 11 Load the application program to the standby controller with a [LOAD 984] operation.

Step 12 Start the controller.

Step 13 Turn the S911 keyswitch to RUN, and verify proper LED indications against Table 4.

Table 4 Hot Standby Normal Indications

Module	Led Indicator	Primary Controller	Standby Controller
S911	READY	ON	ON
	COMM ERROR	OFF	OFF
	PRIMARY	ON	OFF
	STANDBY	OFF	ON
S908/S929	READY	ON	ON
	COMM ACTIVE	ON	Blinking
	COMM ERROR	OFF	OFF
C916/C924	RUN	ON	ON
	READY/SAFE 84	OFF	OFF
984-680/-685/ -780/-785	RUN	ON	ON
	READY	ON	ON
	POWER OK	ON	ON

On-line Editing Procedures with Modbus Plus

Note Because Modbus Plus addresses are set by hardware switches and not controlled by software, the procedures for loading and modifying a program in a 984 Hot Standby Controller configuration are different than in a standard Modbus network implementation.

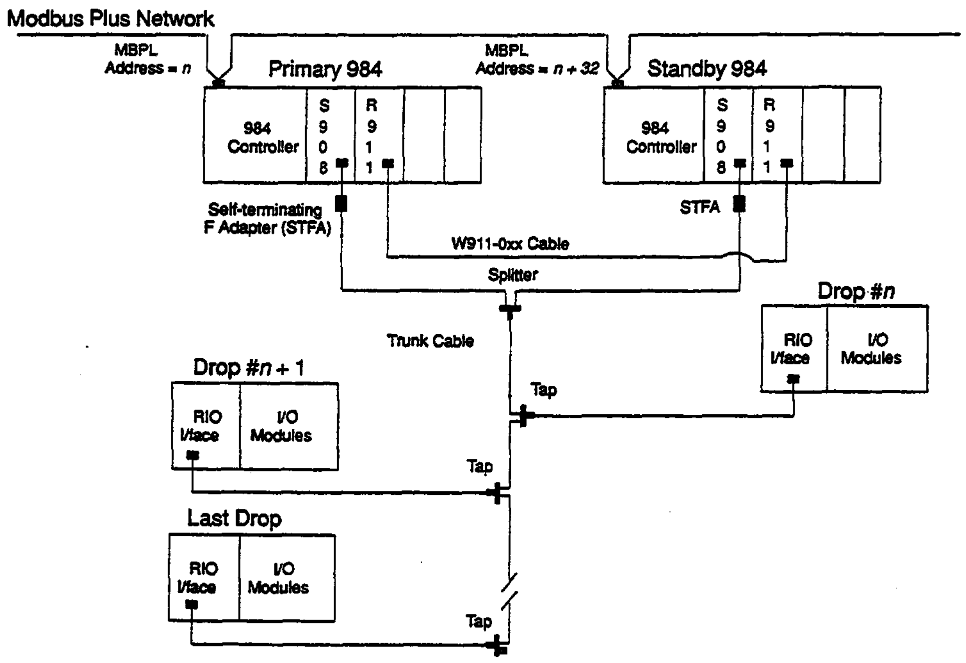


Figure 11 Hot Standby Control on Modbus Plus

Procedure **Initial Load Procedure**

Step 1 Set the Modbus Plus address switches identically on the primary and standby controllers (at address 32 or lower) and connect all cables.

Step 2 Connect *only one* of the programmable controllers to the Modbus Plus network.

Step 3 Download the control program into this controller.

Step 4 Start the controller—it should come up as the primary Hot Standby controller. When its **RUN** light illuminates, disconnect the Modbus Plus connector.

Step 5 Connect the second programmable controller to the Modbus Plus network.

Step 6 Download the control program into this controller and then start the controller. When its **RUN** light illuminates, the panel software (e.g., Modsoft)will return Modbus Error Code 818:

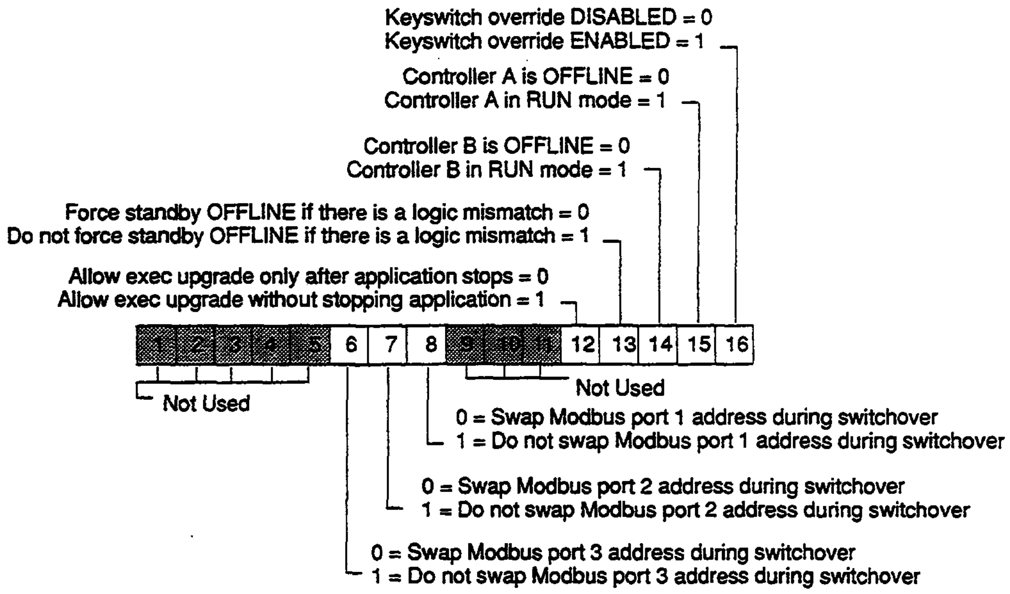
```
Modbus Plus: Remote PLC not listening for call
```

This is a normal response. The Modbus Plus address on this standby controller is now offset 32 addresses higher than the address you specified for the primary controller, while the program panel messages contain the address of the primary controller.

Step 7 Reconnect the primary controller to the Modbus Plus network. Both controllers should now be running and connected to Modbus Plus – i.e., the Modbus Plus LED on each unit should be blinking six times per second.

Procedure How to Modify a Program in a Hot Standby System

Caution Always work with the program uploaded—i.e., saved—from the standby controller. Never upload the program from the primary controller once the system is in operation.



Keyswitch override DISABLED = 0
Keyswitch override ENABLED = 1
Controller A is OFFLINE = 0
Controller A in RUN mode = 1
Controller B is OFFLINE = 0
Controller B in RUN mode = 1
Force standby OFFLINE if there is a logic mismatch = 0
Do not force standby OFFLINE if there is a logic mismatch = 1
Allow exec upgrade only after application stops = 0
Allow exec upgrade without stopping application = 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Not Used
Not Used
0 = Swap Modbus port 1 address during switchover
1 = Do not swap Modbus port 1 address during switchover
0 = Swap Modbus port 2 address during switchover
1 = Do not swap Modbus port 2 address during switchover
0 = Swap Modbus port 3 address during switchover
1 = Do not swap Modbus port 3 address during switchover

Figure 12 Hot Standby Command Register

Step 1 Make sure that bit 13 in the HSBY command register is set to 1 in both the primary and the standby controllers. Setting this bit equal to 1 causes the system to ignore changes to logic, a condition that would normally cause the standby to go OFFLINE; ignore differences in logic. Changes may be made to either controller in the ONLINE mode if bit 13 in the HSBY command register is set to 1.

Step 2 Set the program panel routing path to the address of the standby controller—i.e., 32 addresses higher than the address of the primary controller. You must work with the standby controller, using its Modbus Plus address.

Step 3 Upload the program from the standby controller tothe programming panel.

Step 4 Make the desired modifications to the standby program, then download the modified program back to the standby controller.

Step 5 Once the controller has resumed its ONLINE standby functions, force a failure in the primary controller—i.e., cause a switchover. This can be done by switching the primary unit to OFFLINE.

Step 6 The controller that was previously the primary controller now has the standby address. Download the modified program to the new standby controller and restart it.

Step 7 When the standby controller goes ONLINE in standby mode, the system is again redundant with the modified program installed. You may then wish to reset bit 13 in the HSBY command register to 0, its default setting.

Chapter 8
Locating Startup Errors

When you power up your Hot Standby System, the LEDs on the remote I/O processing modules and S911 Hot Standby modules display light patterns that tells you that the system is running properly or that some error has occurred. Error display patterns are described in the *984 Hot Standby Installation and Maintenance Manual* (GM-S911-001).

If, after carefully rechecking the hardware and software, you are unable to pinpoint the source of a startup error, you can access memory bits in the controller that indicate very specific startup error conditions. These startup error bits are the eight least significant bits in a word at a memory location.

This chapter describes a procedure you can follow to access these error codes.

Procedure for Locating Startup Error Codes

Procedure **Accessing Startup Error Codes**

Step 1 Connect a programming panel—e.g., a P190 or P230—to the controller flashing an error pattern. Load the Utility tape or disk file into the programmer.

Step 2 Set the assembly register (AR) to 00001, press [ATTACH], then press **EXIT**.

Step 3 Press [EXAMINE MEMORY], bring the cursor down to the AR line, and enter 0032 into the AR. Press **GET**.

This gets you to memory location 0032. A hexadecimal number will appear on the screen associated with that memory location. The number may vary depending on the type of programmable controllers your 984 Hot Standby System uses; call the number *nnnn* hex.

Step 4 Apply the following formula to *nnnn* hex:

nnnn hex – 8000 hex + 29 hex

Make the result of this calculation a five-digit hexadecimal number by placing F in the most significant position.

Step 5 Enter this five-digit hexadecimal number into the AR, and press **GET**. A hexadecimal number associated with this memory location appears on the screen.

Step 6 Convert the hexadecimal number on the screen to its binary representation. The eight least significant bits contain the startup error codes.

Step 7 Compare the bit values against those in Table 5. A "1" at any of these bit locations indicates that an error condition exists.

Table 5 Startup

Bit #	Set to 0	Set to 1
8	No error	Traffic Cop checksum error
7	No error	Segment Scheduler checksum error
6	No error	Configuration table checksum error
5	No error	Toggle switch position error
4	No error	UART error
3	No error	RAM address error
2	No error	RAM data error
1	No error	PROM checksum error

An Example of a Startup Error Condition

Here is an example of how you might use this startup error procedure to diagnose a problem.

Error Display

You are trying to power up your Hot Standby system, but the COMM ERROR and the PRIMARY lights on your primary controller are flashing an error pattern. You check your hardware and software configurations, but the source of the error is not obvious.

Applying the Startup Error Procedure

- Connect your programming panel to the primary controller and load the panel with the Utility tape.
- Attach the tape, press **EXIT**, press [EXAMINE MEMORY], and bring your cursor sown to the AR. You want to examine memory location 0032, so enter 32 into the AR and press **GET**. You see displayed on your screen:

  0032 = 9300
- 9300 is a hexadecimal number. Subtract 8000 hex from 9300 hex, then add 29 hex to it. The result is *1329 hex.* Make this hexadecimal number five digits by adding F to the most significant location:

  F1329
- Enter F1329 into the AR and press **GET**. You see displayed on your screen:

  F1329 = 0050
- Convert 0050 hex into its binary representation:

  0000/0000/0110/0000
- Now examine the eight least significant bits, and compare them against Table 5.

Interpreting the Startup Error Codes

Notice that 1's appears at the sixth and seventh bit locations. This tells you that you have checksum errors in the configuration table and the Segment Scheduler.

Schneider Automation Inc.
One High Street
North Andover, MA 01845
Tel: (1) 978-794-0800
Fax: (1) 978-795-9400

Schneider Automation GmbH
Steinheimer Strasse 117
D-63500 Seligenstadt
Tel: (49) 6182 81-2584
Fax: (49) 6182 81-2860

Schneider Automation S.A.
245, Route des Lucioles-BP147
F-06903 Sophia-Antipolis Cedex
Tel: (33) 92 96 20 00
Fax: (33) 93 65 37 15