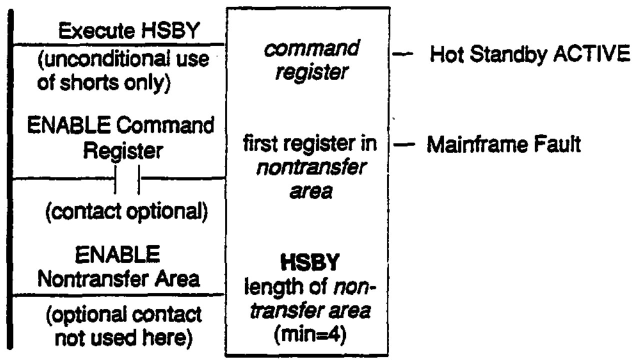
Chapter 2
The HSBY Function Block

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.

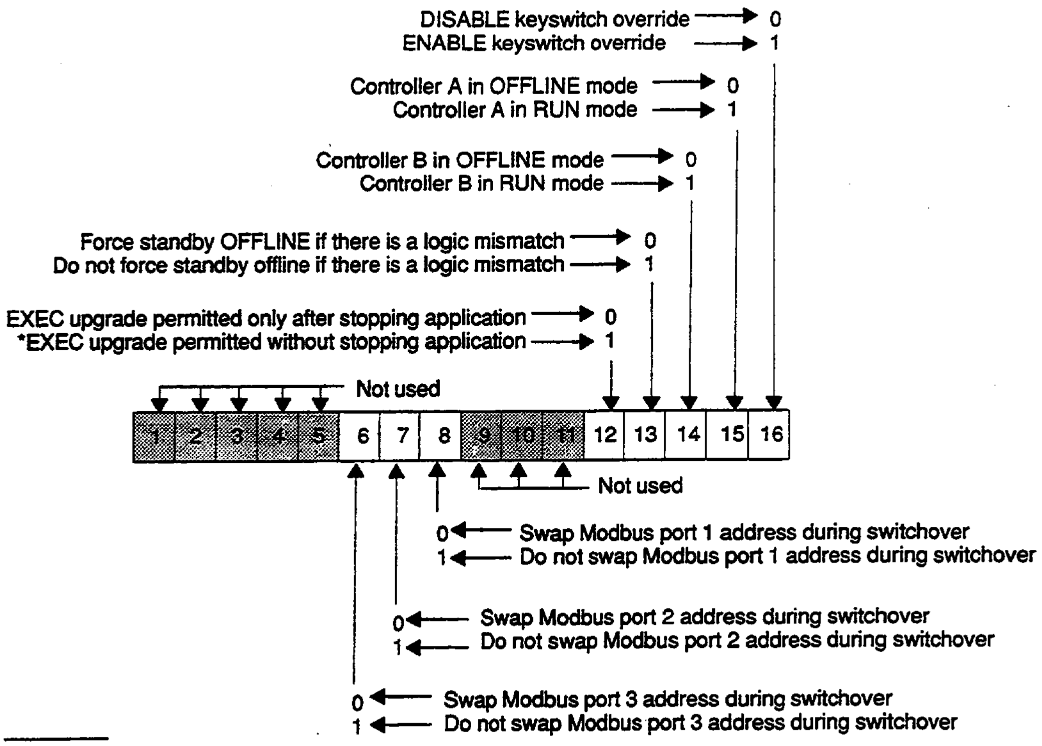
HSBY Output Lines

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

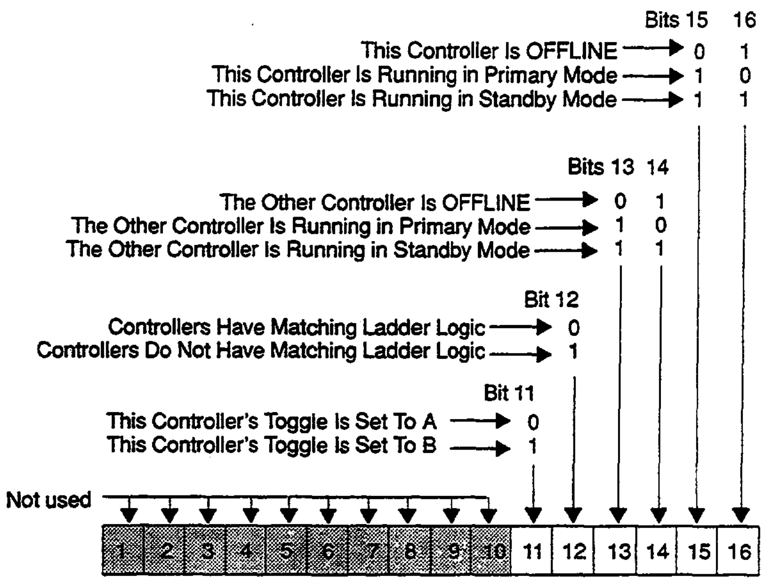
The actual nontransfer area begins at register 4*x* + 3.

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

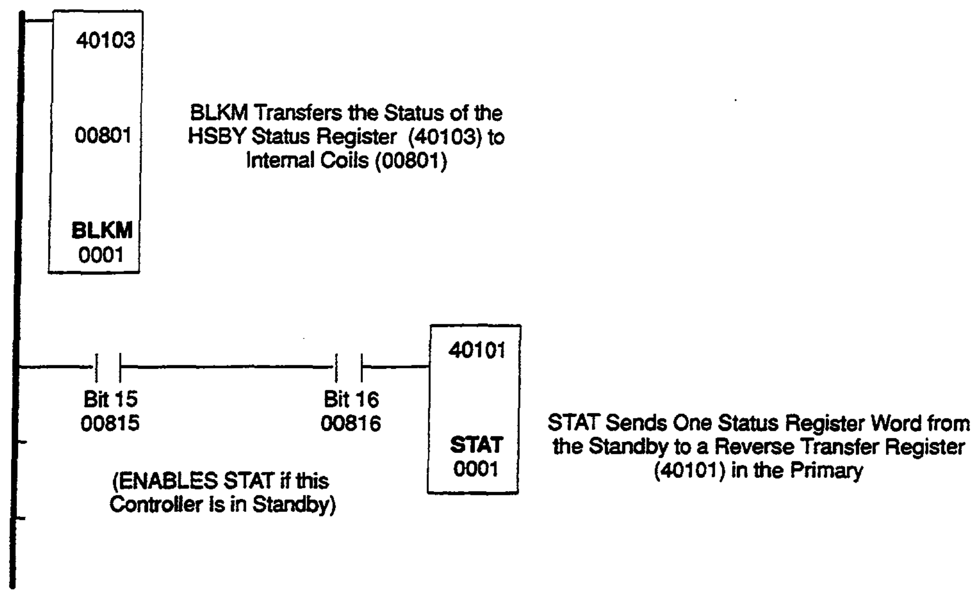
The logic at the top of Figure 5 is in network 2 of segment 1. This network contains a Block Move (BLKM) function and a System Status (STAT) function. The standby controller enables the STAT block. Bits 00815 and 00816 are controlled by bits 15 and 16 in the HSBY status register.

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
(ENABLES STAT if this Controller is in Standby)
STAT Sends One Status Register Word from the Standby to a Reverse Transfer Register (40101) in the Primary

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

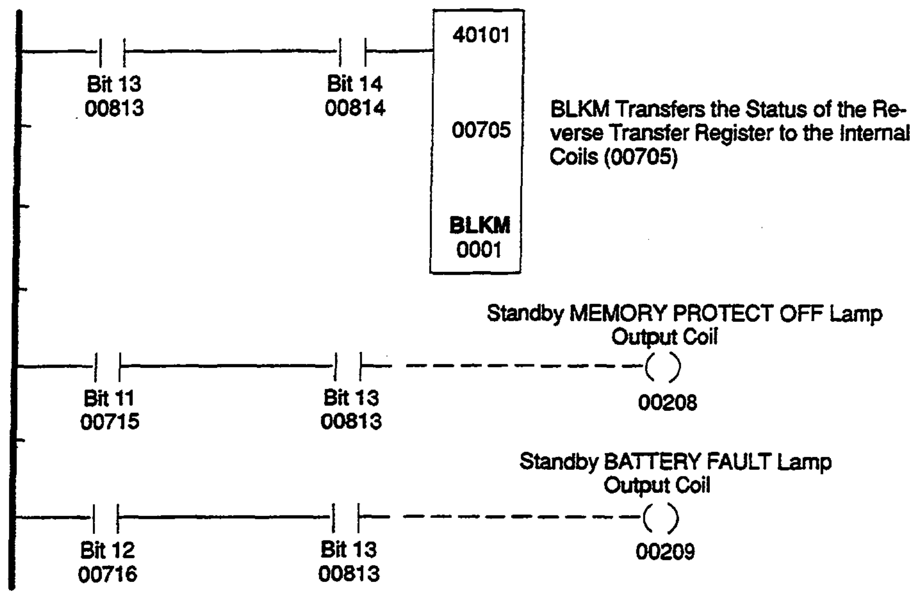

Bit 13
00813
Bit 14
00814
40101
00705
BLKM
0001
BLKM Transfers the Status of the Reverse Transfer Register to the Internal Coils (00705)
Standby MEMORY PROTECT OFF Lamp Output Coil
Bit 11
00715
Bit 13
00813
00208
Standby BATTERY FAULT Lamp Output Coil
Bit 12
00716
Bit 13
00813
00209

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 switchover. 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:

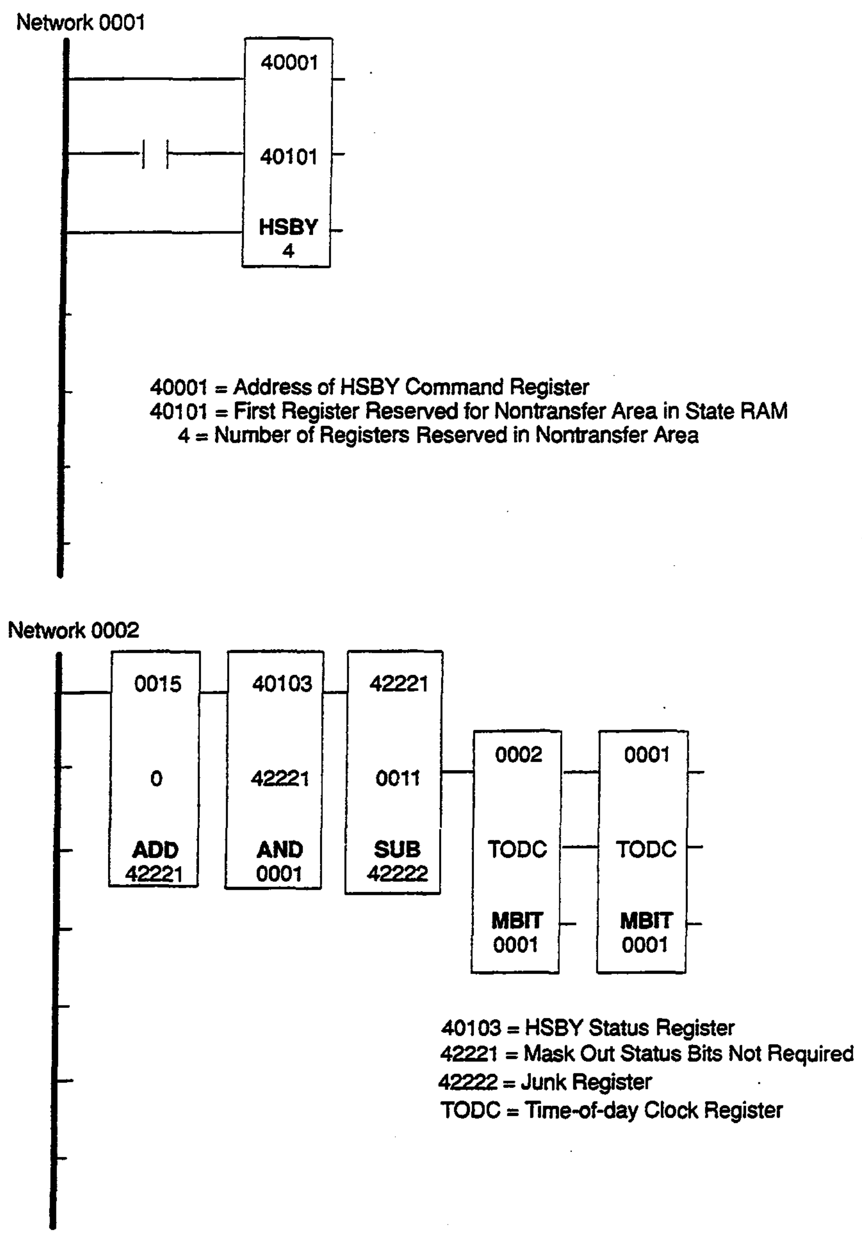
- Which controller is the standby
- That the remaining controller is the primary
- That both controllers are running the same logic

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