# Contents

# Control Processor

The control processor is made up of eight cards (16K-word storage positions that can be addressed): six cards for the processor and two cards for storage. The control processor:

- Controls system input/output (I/O) operations
- Controls assigning of tasks
- Moves data between the I/O devices and the main storage processor
- Handles some of the system control programming
- Moves data between the control processor and the I/O devices that use the channel
- Performs channel command functions (load and sense)
- Moves data between the control processor and the main storage processor
- Controls the main storage processor clock

Control storage contains 16K words; each word is 2 bytes long. Control storage can be addressed one word at a time. The control processor executes control storage instructions that are in control storage. The control processor functions are performed by the control storage program. The control storage program is loaded in control storage during the control storage initial program load (CSIPL) sequence. Control storage is loaded from the disk during normal operations or from the DIAGXX diskettes for diagnostic programs. The diagnostic programs control the routines and work done by transients that are not loaded at CSIPL time.

## DATA FLOW AND CLOCKS

### Data Flow

The control processor works with either 1 or 2 bytes of data at a time. The instruction being executed determines the number of bytes and the exact path of the data.

The 'system bus in' lines (channel SBI) from the channel are 1 byte wide plus parity (9 lines), but the byte can be either a high- or low-order byte in the control processor. If the data on the 'system bus in' lines is to be sent directly to the main storage processor, the control storage program sends 1 byte plus parity at a time. The control processor can also address main storage and main storage processor registers.

*Parity Checking*

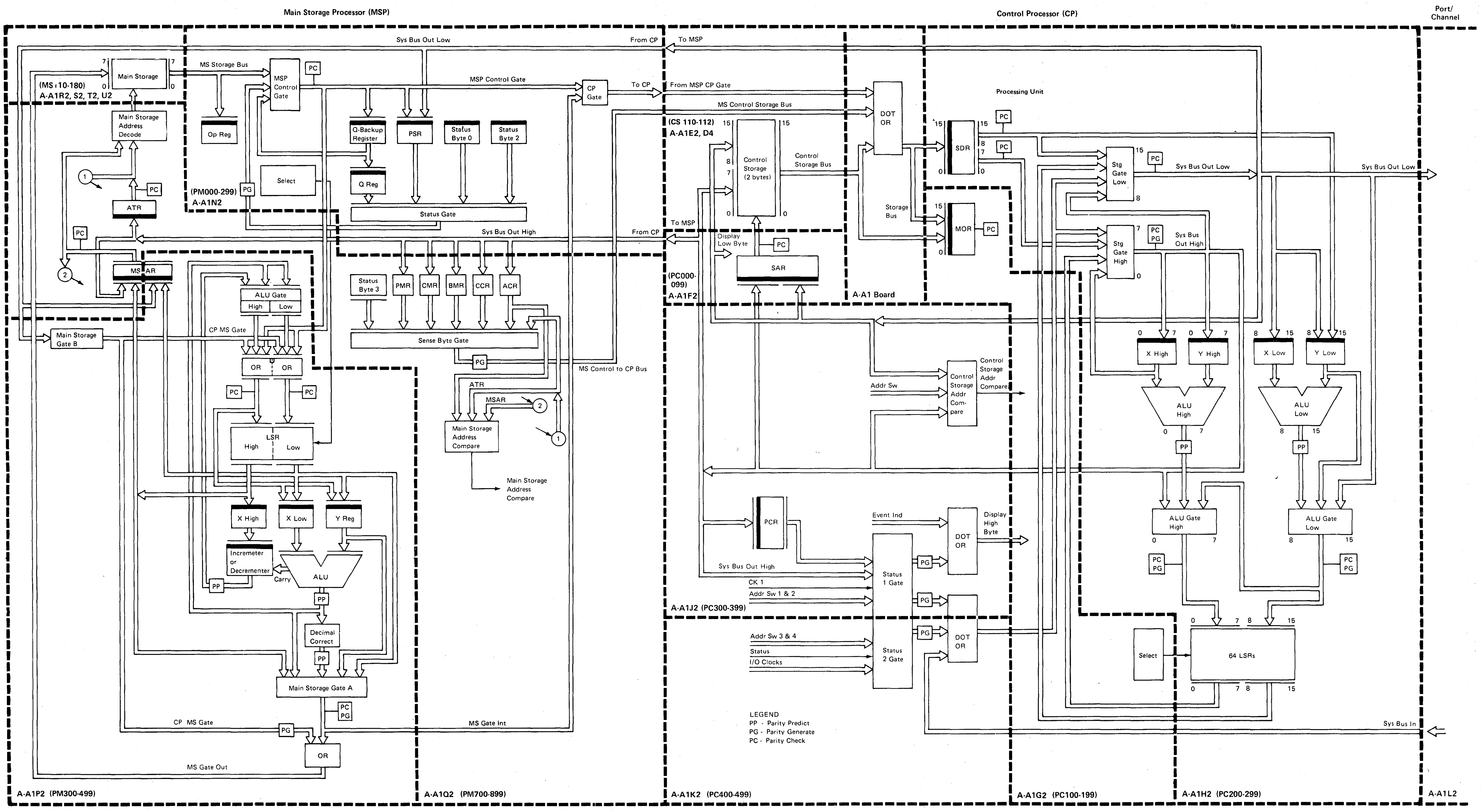
Odd parity by byte is maintained in the data flow. To ensure correct parity, System/34 has checking and generating stations. Parity is checked at the storage address register (SAR), storage data register (SDR), storage gates high and low, arithmetic and logic unit (ALU) gates high and low, micro-operation register (MOR), and on the channel data lines.

Parity generating stations are supplied for the status register, the control panel, switch bytes, and other internally generated data pertaining to the control processor (storage gate high and ALU gates high and low).

*Default Conditions*

If no hardware conditions are specified for the control processor, the control processor has automatic selections and functions that are default conditions. The default conditions for the functional units in the control processor are as follows:

| Unit | Default Selection |
|---|---|
| Storage gate high | LSR high |
| Storage gate low | LSR low |
| ALU gate high | ALU high |
| ALU gate low | ALU low |
| ALU function | X-register plus 1 |



*Data flow bus lines may not pass through FRUs as shown

## Clocks

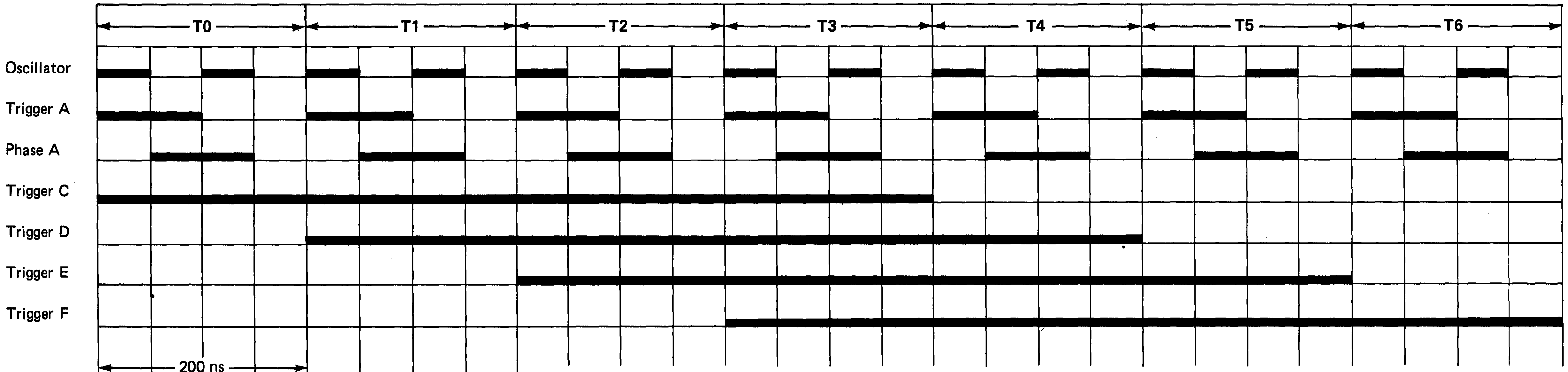
### System

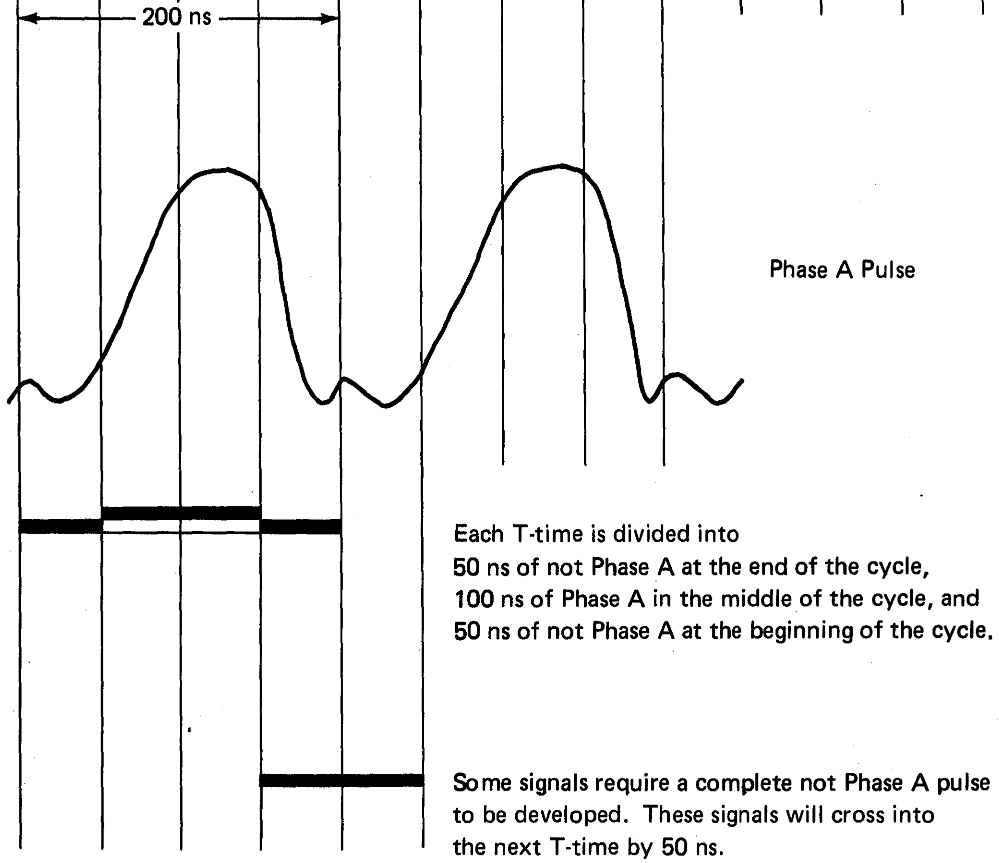
The control processor has a 100-nanosecond oscillator that runs continuously, supplying the 10-megahertz frequency needed for the clock pulses. The rise of this oscillator output causes 'trigger A' to change condition, while the fall of this oscillator output causes the 'phase A' line to change condition.

Four control processor clock triggers (C, D, E, and F) are decoded to determine control processor clock times T0 through T6.

When the current instruction is decoded, the control processor determines if some of the control processor clock times are needed and controls the gating of the triggers to skip the times that are not needed.

**T-Time and Phase A Relationship**

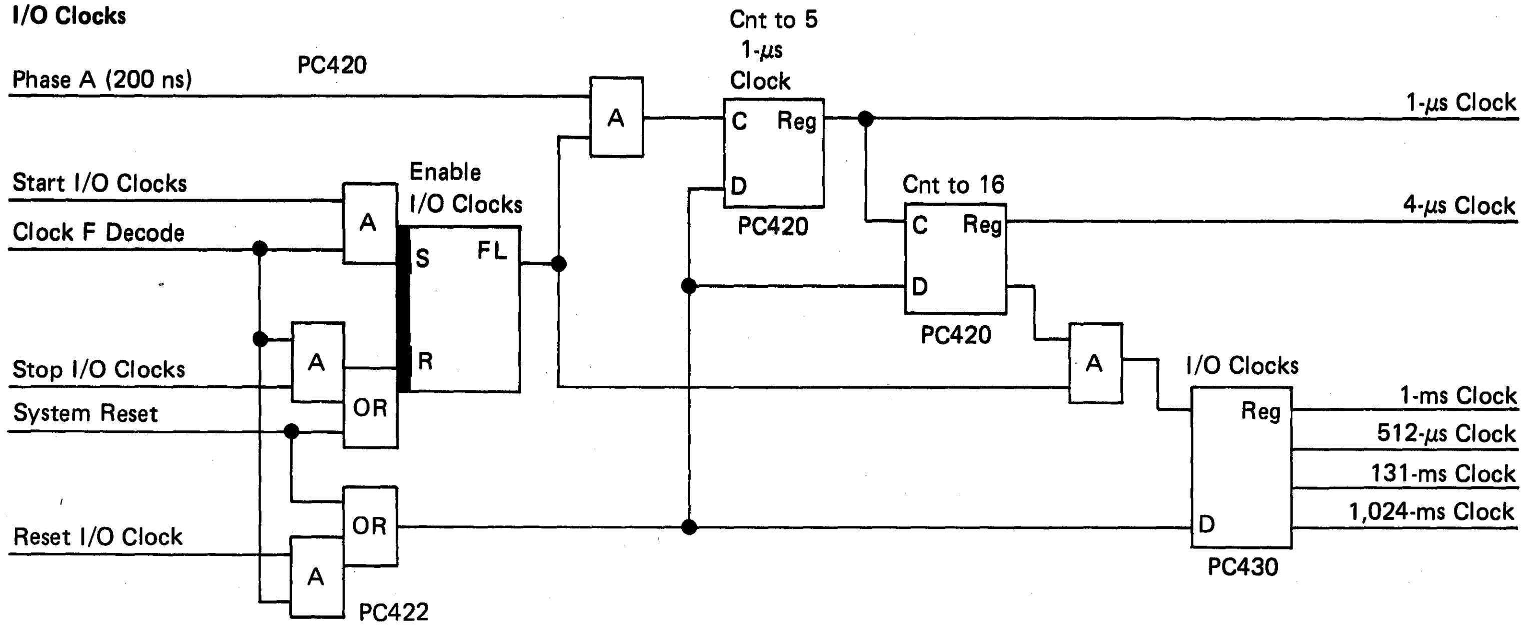




Each T-time is divided into
50 ns of not Phase A at the end of the cycle,
100 ns of Phase A in the middle of the cycle, and
50 ns of not Phase A at the beginning of the cycle.

Some signals require a complete not Phase A pulse to be developed. These signals will cross into the next T-time by 50 ns.

**I/O Clocks**



### I/O Attachment and Controller

The control processor has eight continuously running clocks that are used by the I/O attachments and controllers. Seven of these clocks can be stopped and started for diagnostic testing. The 100-nanosecond, free-running internal oscillator generates the 'phase A' line which, in turn, generates the other seven clocks. Clock triggers are used to count the time needed in the generation of the seven clocks. The times of the clocks are:

- 100 nanoseconds (oscillator)
- 1 microsecond
- 4 microseconds
- 1 millisecond
- 512 microseconds
- 16 milliseconds
- 131 milliseconds
- 1,024 milliseconds

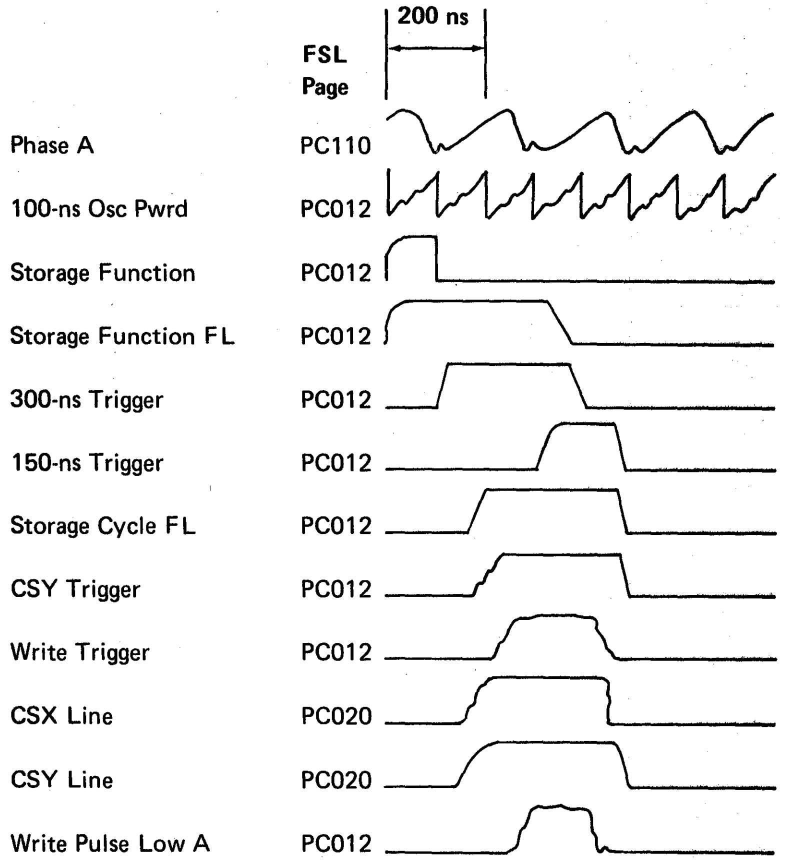
These clocks, except for the 100-nanosecond oscillator, are sensed by the I/O immediate instruction. The clocks must be in a stop condition before a program can execute an I/O immediate instruction (B76R or B66R).

### Storage

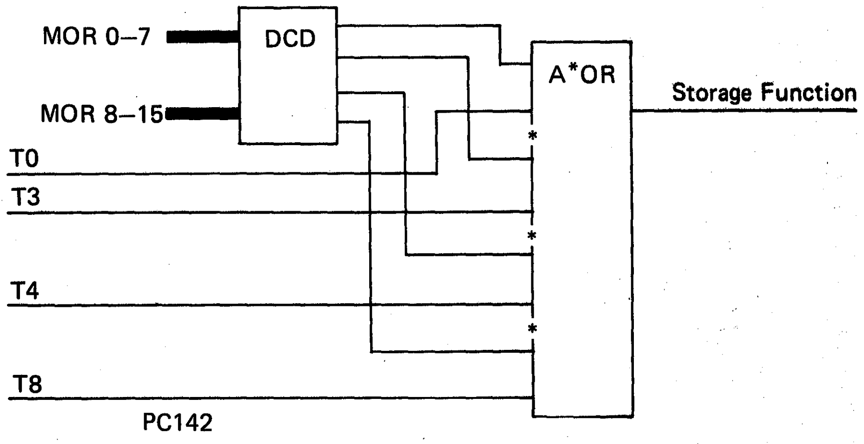
During instruction times T0 through T2, time T0 fetches the microaddress register (MAR) contents from the local storage register (LSR) stack and places this data into the storage address register (SAR). Time T0 also starts the storage clocks for the storage access. During times T1 and T2, storage is addressed to read the instruction.

The storage clocks are also used during burst-cycle-steal-mode operations and base-cycle-steal-mode operations. When an I/O device activates the 'disk/dskt block processor clock' line or the 'base cycle steal request' line, the control processor completes the instruction it is working on and then goes to the T7 condition where it is held until the 'disk/dskt block processor clock' line is not active. The rise of the 'disk/dskt (load) BC req' line[1] while the 'disk/dskt block processor clock' line is active generates a 'storage cycle request' line which, in turn, generates time T8 (clock SAR and X reg); time T8 is then used to load the storage address in the main storage address register (MSAR) or control storage address register (SAR). After the operation is completed, the 'disk/dskt block processor clock' line is not active and the control processor clocks are permitted to run. (See *Burst Cycle Steal Mode* in the *Channel* section of this manual.) The control processor storage clocks can also control main storage. (See *Control Processor and Main Storage Processor Communication* in the *Interrupts and Cycle Steal Requests* section of this manual.)

**Storage Access Timings**



**Storage Function**



[1] BC is a burst cycle request.

# OPERATIONS

## IPL-Customer User Programs

*MSIPL Switch in Disk Position; CSIPL Switch in Disk Position:* Initial program load (IPL) is completed in three major stages from the time the Load key is pressed until the SYSTEM CONSOLE message is displayed on the system console display screen. Loading is done from the disk. The three stages of IPL are as follows:

| IPL | | |
|---|---|---|
| CSIPL | | MSIPL |
| **Stage 1** | **Stage 2** | **Stage 3** |
| Control storage is loaded three times to run diagnostic routines and check hardware circuits (see Section 99 of the *5340 System Unit Maintenance Manual*). | The control storage program loaded includes IPL routines that overlay stage 1 and are executed (see the *Control Storage Logic Manual*). | Main storage initialization is loaded in three phases (see the *SSP Logic Manual: System*). |

### Control Storage Initial Program Load (CSIPL)

*Stage 1*

Stage 1 of the control storage initial program load (CSIPL) sequence loads control storage three times and performs a basic system check of the control processor and I/O functions. Nine display lights (display byte 0, bits P0 and 0 through 7), and the Load light on the CE panel are set to on by pressing the Load key. These lights are reset to off at various stages of the CSIPL by both hardware and software as programs are loaded and executed.

*First Load:* Load 16 sectors (2K words) that contain control processor diagnostic routines 1 through 19.

Then, perform the following tasks:

1. Load control storage from disk by a burst-cycle-steal-mode operation.
2. Load 2K words (4,096 bytes) into control storage at hexadecimal addresses 0000 through 07FF.
3. Reset the microaddress register (MAR) for machine check (local storage register hexadecimal 0A) to hexadecimal 0000 and execute any machine check log routines for control processor errors using interrupt level 0.
4. Software set the microaddress register (MAR) for main program level to hexadecimal 0292, branch to hexadecimal 00FF, and execute instructions for diagnostic routines 1 through 19.

If all tests run correctly, the following lights are reset to off in the sequence: bits P0, 0, and 1 of display byte 0, the Load light, and bit 2 of display byte 0.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).

See *Error Indications* or *Display Light Sequence* later in this section.

*Second Load:* Load 16 sectors (2K words) that contain control processor diagnostic routines 20 through 70.

Then, perform the following tasks:

1. Load control storage from disk by a burst-cycle-steal-mode operation.
2. Load 2K words (4,096 bytes) into control storage at hexadecimal addresses 0800 through 0FFF.
3. Software set the microaddress register (MAR) for main program level to hexadecimal 0800 and execute instructions for diagnostic routines 20 through 70.

If all tests run correctly, bits 3 and 4 of display byte 0 are reset to off.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).

See *Error Indications* or *Display Light Sequence* later in this section.

*Third Load:* Load 28 sectors (3.5K words) that contain control processor diagnostic routines 71 through 79 and device wrap loader tests.

Then, perform the following tasks:

1. Load control storage from disk by a burst-cycle-steal-mode operation.
2. Load 3.5K words (7,168 bytes) into control storage at hexadecimal addresses 0080 through 0E7F.
3. Software set the microaddress register (MAR) to hexadecimal 0080 and execute instructions for diagnostic routines 71 through 79.

The wrap loader calls in each device wrap test and executes that test before it calls in the next wrap test.

If all tests run correctly, bits 5, 6, and 7 of display byte 0 are reset to off.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- The Console Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).
- Error messages are stored in control storage at hexadecimal locations 07A0 through 07BF and may also appear on the system console display screen.

See *Error Indications* or *Display Light Sequence* later in this section.

*Stage 2*

Stage 2 of the control storage initial program load (CSIPL) sequence loads the control storage program that contains the routines necessary to load:

- The work station controller program
- The printer controller program
- The main storage nucleus initialization program (#MSNIP)

Then, perform the following tasks:

1. Load control storage from disk by a burst-cycle-steal-mode operation.
2. Load 62 sectors (9.75K words) into control storage at hexadecimal addresses 0000 through 26FF.
3. Software set the microaddress register (MAR) to hexadecimal 1E00 and the control processor takes control.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- The Console Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).
- Error messages are stored in control storage at hexadecimal locations 07A0 through 07BF and may also appear on the system console display screen.

See *Error Indications* or *Display Light Sequence* later in this section.

## Main Storage Initial Program Load (MSIPL)

*Stage 3*

Initialization of main storage completes the hardware and software tasks necessary to load the System Support Program Product (SSP) and ready the system for customer user program requests. The initialization is performed in three phases.

*Phase 1:* The main storage module (#MSNIP) initializes main storage. This module is the basic first step for all other modules that will be used during the main storage initial program load (MSIPL) sequence. The main functions of #MSNIP are to:

- Initialize the system communications area
- Assemble the resident library format 1
- Determine the bad main storage locations
- Initialize the transient/transfer control table
- Determine the disk addresses as needed
- Set the command processor task control block (TCB) to indicate any bad 2K storage blocks
- Increase the size of the assign/free area to permit assigning of main storage
- Load and pass control to software module #MSTWA (phase 2)

*Phase 2:* Software module #MSTWA initializes the task and work areas in main storage. The main functions of #MSTWA are to:

- Initialize the transfer control table for the resident routine
- Initialize the task work area index
- Initialize the terminal unit blocks
- Initialize the task work areas for each work station
- Assemble the device allocate table
- Load and pass control to software module #MSIPL (phase 3)

*Phase 3:* This phase controls the last main storage initial program load (MSIPL) and includes a group of software modules under the control of software module #MSIPL. The main functions of #MSIPL are to:

- Perform the main storage initial program load sign-on request
- Process the override information if necessary
- Initialize the print spool function
- Complete the nucleus initialization

Before MSIPL is complete, the #MSIPL module updates the instruction address register (IAR) in the request block (RB) stack to pass control to the command processor resident router. The supervisor task attach transient then attaches a task control block (TCB) to run file rebuild. Control then passes to the control processor resident router. The IPL SIGN-ON message is displayed on the system console display screen while phase 3 is completing many of the last tasks.

Initial program load is complete when SYSTEM CONSOLE DISPLAY appears on the display screen or COMMAND DISPLAY appears at one of the work stations. The customer now has an operational system and can process job requests.

Errors that occur during main storage initial program load cause two types of not normal terminations (abends):

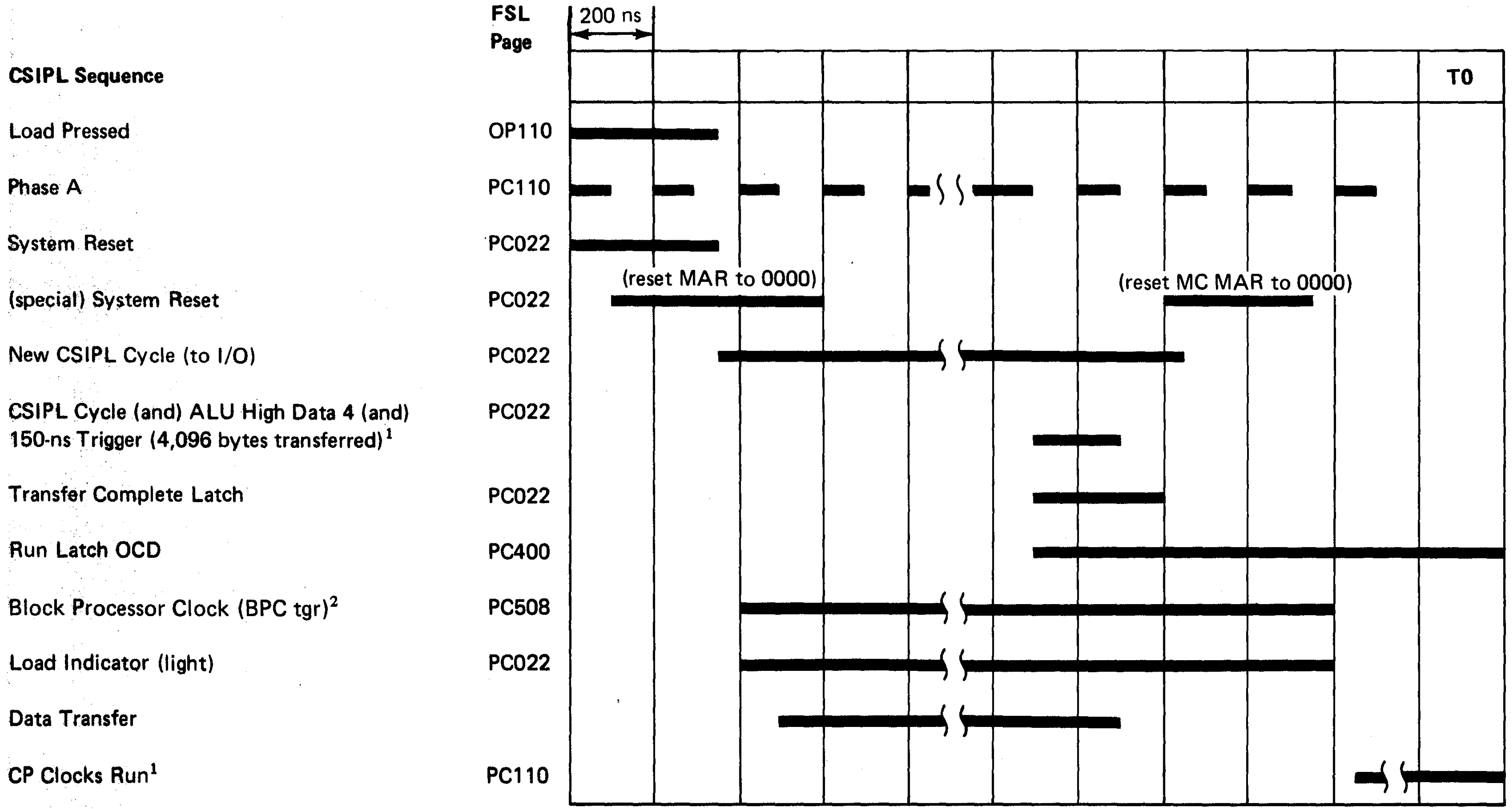
- Task-associated abends do not stop the system (except for the command processor task), but a dump of main or control storage is written to disk and only the error task is terminated while other tasks continue.
- System-associated abends are so severe that they do not permit any task to continue. The system must be stopped immediately so the damage can be contained and diagnosed. Two types of processor checks that cause system-related abends are:
  - Hardware generated–The specified error is shown in the command processor unit status word indicators. (Set the Mode Selector switch on the CE panel to the Dply Chks position.)
  - Software generated–Activated by the System Support Program Product when an error occurs that cannot permit the operation to continue. (A display of selected local storage registers describes the error more fully.)

For detailed information on errors, see Appendix G. *Troubleshooting Aids* and Appendix I. *Hardware Diagnostic Information* in the *Data Areas Handbook*.

To run a complete test of the I/O devices, run the SYSTST program. SYSTST checks all the mechanical parts of all the I/O devices, the system program, and the I/O routines.

### IPL Timing Sequence

Pressing and releasing the Load switch starts the control storage initial program load (CSIPL) sequence and the Load light is set to on. The CSIPL, along with the ALU high 'data 4' and the '150-ns tgr' lines, causes the 'transfer complete' line to be activated.



*Note:* The Load light continues to be set on if: (1) the block processor clock is not de-activated, (2) the disk is not ready, or (3) a processor check occurs.

[1] This line cannot be probed.
[2] The 'block processor clock' line is active as shown for 62EH disk drives. The line will be pulsing if 62PC disk drives are installed.

### Display Light Sequence (Byte 0)

The Load light and all nine display lights (display byte 0 on the CE panel) are set to on when the operator presses the Load switch. When the Load switch is released, the control storage initial program load (CSIPL) sequence starts and 2K words are moved into control storage (from either the disk or a diskette). At the end of the move of 2K words, the Load light is reset to off if no error was sensed. The lights are reset to off as described below as the sequence advances. If CSIPL is not completed, the lights that represent the part of CSIPL that was not completed continue to be set on. The Mode Selector switch must be in the Proc Run position for the lights to appear when set to on (clock running).

If during the CSIPL, the system has a processor check, and byte 0 bits P0, 0, 1, and 2 are reset to off, and either bit 3, 4, 5, 6, or 7 is set to on, this indicates that the control processor has failed in one of its bring-up diagnostic routines. To determine which routine failed (for routine numbers larger than 08), display work register 3 low. This register will contain the hexadecimal number that identifies the failing routine. See Section 99 of the *5340 System Unit Maintenance Manual* for routine numbers.

Each light is reset to off and remains off as follows:

P0[1] The adapter has received the 'load' signal and made active the 'disk/dskt block processor clock' signal to start data transmission by a burst-cycle-steal-mode operation.

0[1] The first cycle steal request was received and data transmission was started (write trigger).

1[1] The transmission of 4,096 bytes of data was completed.

Load[1] The data transmission was completed with no data check.

2 The branch and branch-on-condition routines have completed. Parity checks are reset during routine 2.

3 The second load of control storage was completed and the first instruction was executed.

4 The control storage test was run correctly.

5 The third load of control storage was completed and the first instruction was executed.

6 The main storage test ran correctly. Start executing the wrap loader control program.

7 The System Support Program Product or the diagnostic supervisor was loaded. After loading, the initial program load sequence is complete and the system is ready to run user programs or diagnostic programs.

[1] Reset by hardware controls. The other lights are reset by control storage instructions.

### Disk Operation

When the operator presses the Load switch, the control storage initial program load (CSIPL) sequence does three partial control storage loads. Then, it loads the control storage program from cylinder 0, track 1, sector 0-3B, and takes control at hexadecimal location 1E00 of control storage.

The control storage program has routines that load and control the main storage initialization along with loading the System Support Program Product.

*First Load:* Hardware loads 2K words into control storage at hexadecimal locations 0000 through 07FF. These words contain the following:

| | Words | Addresses (Hex) |
|---|---|---|
| Direct area (the unit definition table and addresses) | 128 | 0000-007F |
| Control processor instruction tests | 1,408 | 0080-05FF |
| Disk loader | 512 | 0600-07FF |

*Second Load:* The disk loader loads 2K words into control storage at hexadecimal locations 0800 through 0FFF. These words contain the following:

| | Words | Addresses (Hex) |
|---|---|---|
| Remainder of control processor instruction tests | 1,792 | 0000-0EFF |
| Control storage tests | 256 | 0F00-0FFF |

*Third Load:* The disk loader loads 3.5K words into control storage at hexadecimal locations 0080 through 0E7F. These words contain the following:

| | Words | Addresses (Hex) |
|---|---|---|
| Main storage processor basic tests | 640 | 0080-027F |
| Wrap loader and control program subroutines | 128 | 0280-02FE |
| Wrap loader and control program | 512 | 02FF-047F |
| Wrap device identification and location table | 256 | 0480-057F |
| Wrap device and unit definition table | 256 | 0580-067F |
| Additional subroutines | 128 | 0680-06FF |
| Reserved | 128 | 0700-077F |
| Wrap error storage area | 128 | 0780-07FF |
| Work station display routine and CSIPL wrap error message | 640 | 0800-0A7F |
| CSIPL device wrap tests | 1,024 | 0A80-0E7F |

*Fourth Load:* The disk loader loads 9.75K words into control storage at hexadecimal locations 0000 through hexadecimal 26FF. Control is passed to hexadecimal location 1E00.

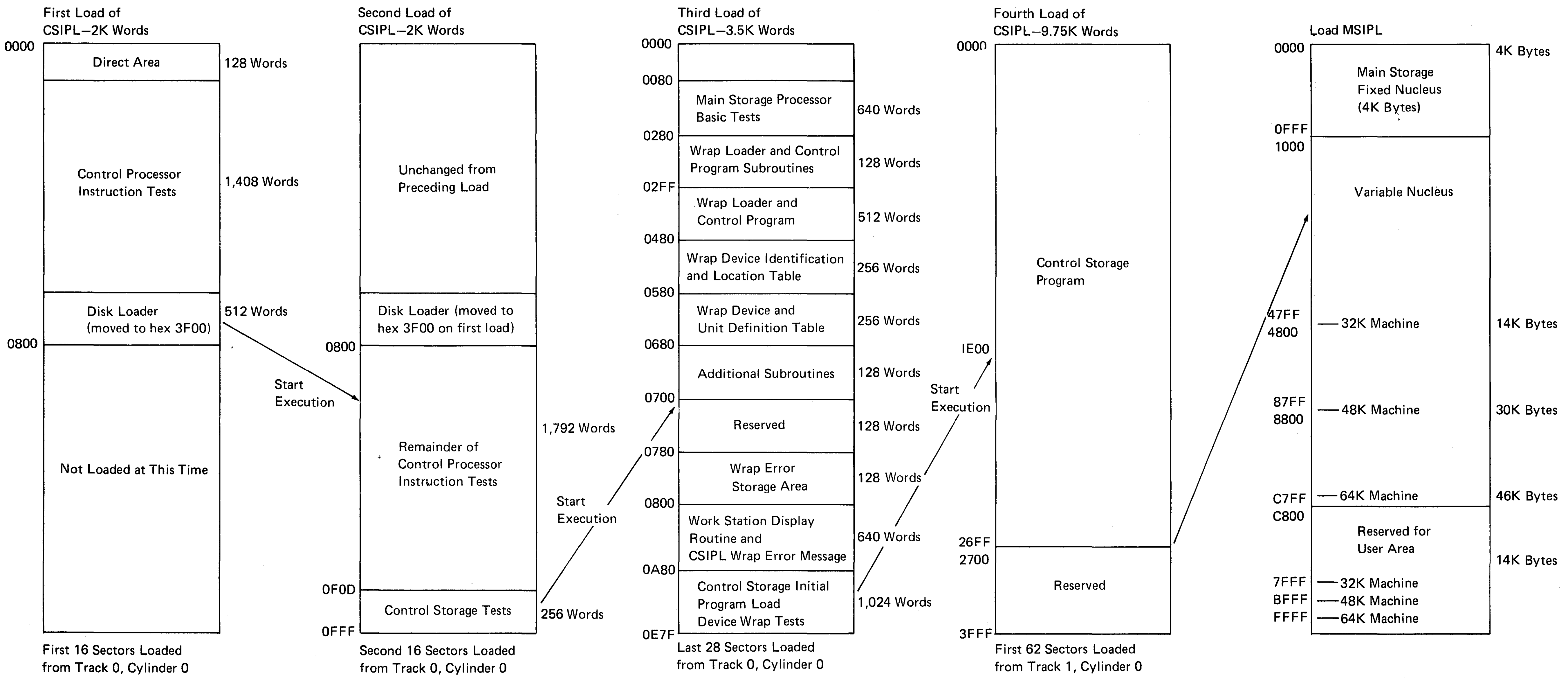
For more information on control storage initial program load, see Section 1 of the *Control Storage Logic Manual.*

*Load MSIPL:* The last CSIPL load routine of the main storage fixed nucleus and the variable nucleus (under control of the control storage program) are loaded into main storage. The size of the variable nucleus will rely on the system configuration.

| Start Address (Hex) | Function | Words Assigned (Decimal) |
|---|---|---|
| 0000 | System Communication Area | 208 |
| 00D0 | Termination Dump IOB | 32 |
| 00F0 | Termination Dump ACE | 16 |
| 0100 | ACE Queue Headers | 192 |
| 01C0 | Multipurpose IOB | 32 |
| 01E0 | CS Transient Loader IOB | 32 |
| 0200 | Command Processor TCB | 128 |
| 0280 | Task Work Area Index | 24 |
| 0298 | System Diskette IOB | 60 |
| 02D4 | Disk Error Request Block | 12 |
| 02E0 | #Library Format 1 | 32 |
| 0300 | Alter/Display ACE | 16 |
| 0310 | Alternative Sector ACE | 16 |
| 0320 | Statistical Logout ACE | 16 |
| 0330 | Interval Timer ACE | 16 |
| 0340 | MS Processor Check ACE | 16 |
| 0350 | Swap ACE | 16 |
| 0360 | MS Transient Loader ACE | 16 |
| 0370 | Diskette ERP ACE | 16 |
| 0380 | Error Task-to-Task ACE | 16 |
| 0390 | Dispatcher TQE | 8 |
| 0398 | Midnight TQE | 8 |
| 03A0 | Statistics Logging TQE | 8 |
| 03A8 | System Queue Space/Failure TQE | 8 |
| 03B0 | MSIPL Free Area | 848 |
| 0700 | Minimum Trace Buffer | 256 |
| | or Alter Display Work Area | 170 |
| | and | |
| 07C0 | CSIPL Error Log Work Area | 64 |
| 0800 | Main Storage Transient Area | 2,048 |
| 1000 | Variable Nucleus | |
| | Terminal Unit Blocks | |
| | Command Processor Work Area | |
| | Command Processor Matrix Image | |
| | Command Processor Mainline | |
| | Disk Data Management | |
| | Task-to-Task Communications | |
| | Device Allocate Table | |
| | Command Processor Error ACE | |
| | Command Processor Task-to-Task ACE | |
| | Command Processor JCB | |
| | Spool Intercept | |
| | Spool Intercept Buffer | |
| | Spool Write Buffer | |
| | Display Station Data Management | |
| | Work Station Queue Space | |
| | System Queue Space | |
| 2000 | Load Address for IPL Diskette | |

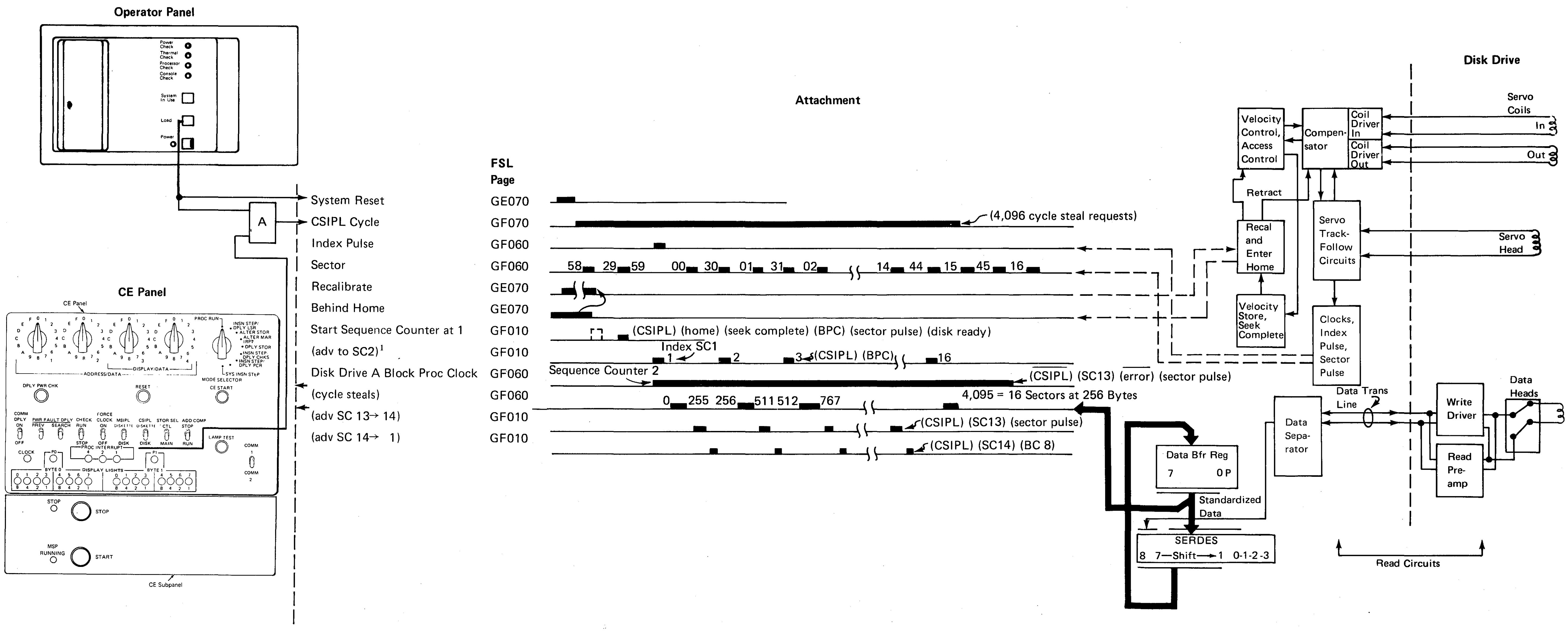
See the *SSP Logic Manual: System.*

**Disk Sequence**



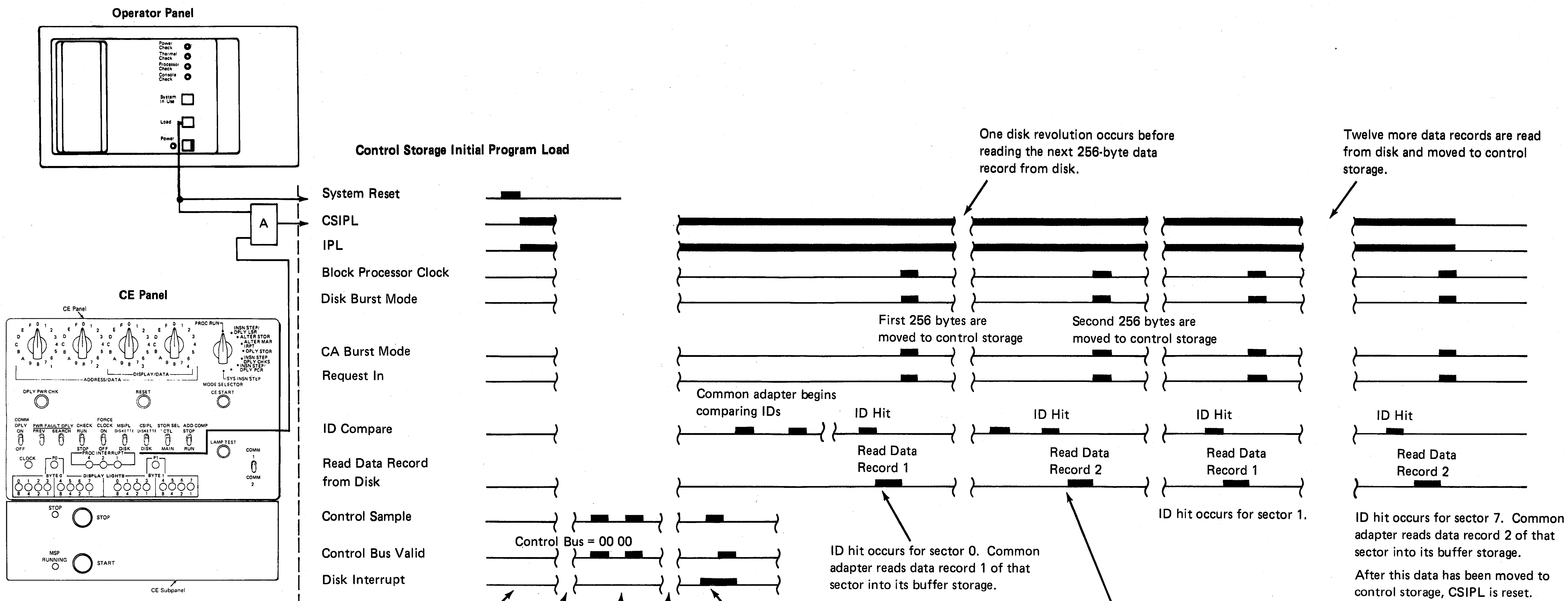
Page is intentionally left blank.

## 62EH Disk Timing



[1] Data transfers operate like read data or read diagnostic operations. Sector hit is forced.

## 62PC Disk Timing



One disk revolution occurs before reading the next 256-byte data record from disk.

Twelve more data records are read from disk and moved to control storage.

First 256 bytes are moved to control storage

Second 256 bytes are moved to control storage

Common adapter begins comparing IDs

ID hit occurs for sector 1.

ID hit occurs for sector 7. Common adapter reads data record 2 of that sector into its buffer storage.

After this data has been moved to control storage, CSIPL is reset.

ID hit occurs for sector 0. Common adapter reads data record 1 of that sector into its buffer storage.

Interrupt from the disk, indicating seek has been completed.

ID hit occurs for sector 0. Common adapter reads data record 2 of that sector into its buffer storage.

Control storage initial program load starts when the Load switch is pressed while disk CSIPL is selected.

Disk drive performs a seek to cylinder 0 and selects head 0.

During power on, the common adapter will perform a seek calibration sequence after the disk becomes ready and is recalibrated. There will be a 20-second delay from power on to the start of the CSIPL sequence.

CSIPL sequence is started by sending a seek command to disk drive A, to seek to cylinder 0 and select head 0.

CSIPL reads the first 16 data records (4,096 bytes) from head 0, cylinder 0 of disk drive A into control storage.

## IPL-Customer SSP from Diskettes

*MSIPL Switch in Diskette Position; CSIPL Switch in Disk Position:* Initial program load (IPL) is completed in three major stages from the time the Load key is pressed until the system operator does another IPL from disk and IPL SIGN-ON and SYSTEM CONSOLE DISPLAY have been displayed on the system console display screen. This type of IPL is necessary when the customer must update his SSP with a new release or exchange an existing SSP because it has been damaged.

*Stage 1*

The control storage initial program load (CSIPL) sequence loads control storage three times from the disk and performs a basic check of the control processor and I/O functions. Stage 1 is the same as the IPL operation on customer user programs. Nine display lights (display byte 0, bits P0 and 0 through 7) and the Load light on the CE panel are set to on by pressing the Load key. These lights are reset to off at various stages of CSIPL by both hardware and software as programs are loaded and executed.

*Stage 2*

CSIPL loads the control storage program from the disk that contains the routines necessary to load:

- The work station controller program

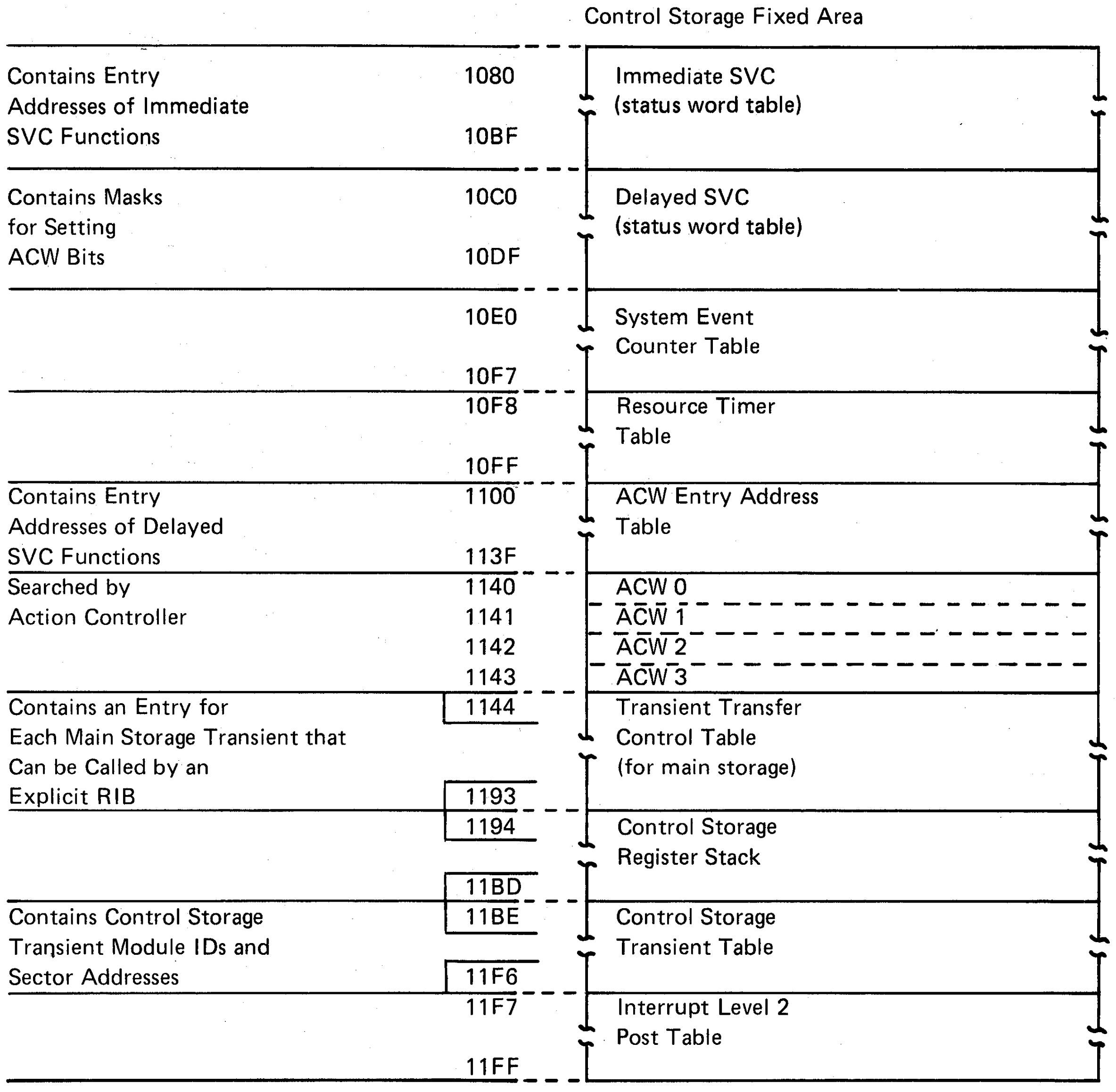
- The printer controller program

The MSIPL switch in the Diskette position is checked by the program and causes two operations to occur: 1) the main storage initialization is not done as before (IPL of customer user programs); 2) the IPL sign-on display is bypassed.

*Stage 3*

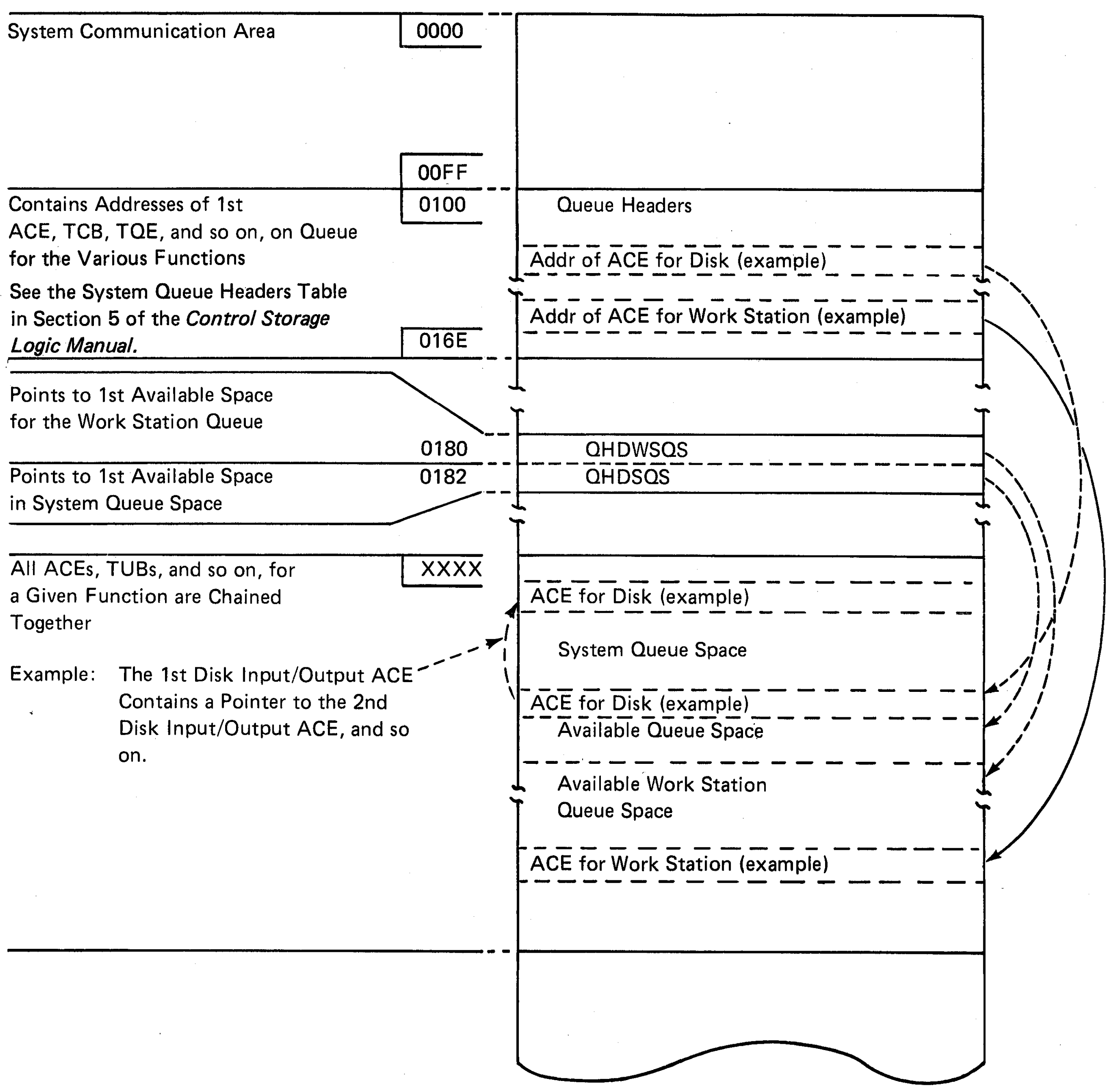
The first load program from diskette is loaded into main storage and starts executing the load routines. The SYSTEM CONSOLE DISPLAY message gives prompting messages to the operator to control the inserting of all necessary diskettes as they are again loaded on the disk. If the correct sequence is followed and all diskettes have been loaded, the COMPLETE message informs the operator that the programs are all loaded and the system is now ready for a normal customer IPL. Some additional prompting messages inform the operator to: 1) reset the CSIPL and MSIPL switches to the Disk position and 2) press the Load key to perform an IPL for customer user programs.

**Control Storage Layout**



|  |  | Control Storage Fixed Area |
|---|---|---|
| Contains Entry Addresses of Immediate SVC Functions | 1080<br>10BF | Immediate SVC (status word table) |
| Contains Masks for Setting ACW Bits | 10C0<br>10DF | Delayed SVC (status word table) |
|  | 10E0<br>10F7 | System Event Counter Table |
|  | 10F8<br>10FF | Resource Timer Table |
| Contains Entry Addresses of Delayed SVC Functions | 1100<br>113F | ACW Entry Address Table |
| Searched by Action Controller | 1140 | ACW 0 |
|  | 1141 | ACW 1 |
|  | 1142 | ACW 2 |
|  | 1143 | ACW 3 |
| Contains an Entry for Each Main Storage Transient that Can be Called by an Explicit RIB | 1144<br>1193 | Transient Transfer Control Table (for main storage) |
|  | 1194<br>11BD | Control Storage Register Stack |
| Contains Control Storage Transient Module IDs and Sector Addresses | 11BE<br>11F6 | Control Storage Transient Table |
|  | 11F7<br>11FF | Interrupt Level 2 Post Table |

**Main Storage Layout**



|  |  |  |
|---|---|---|
| System Communication Area | 0000<br>00FF |  |
| Contains Addresses of 1st ACE, TCB, TQE, and so on, on Queue for the Various Functions<br>See the System Queue Headers Table in Section 5 of the *Control Storage Logic Manual.* | 0100<br>016E | Queue Headers<br>Addr of ACE for Disk (example)<br>Addr of ACE for Work Station (example) |
| Points to 1st Available Space for the Work Station Queue | 0180 | QHDWSQS |
| Points to 1st Available Space in System Queue Space | 0182 | QHDSQS |
| All ACEs, TUBs, and so on, for a Given Function are Chained Together | XXXX | ACE for Disk (example)<br>System Queue Space<br>ACE for Disk (example)<br>Available Queue Space<br>Available Work Station Queue Space<br>ACE for Work Station (example) |

Example: The 1st Disk Input/Output ACE Contains a Pointer to the 2nd Disk Input/Output ACE, and so on.

## IPL-CE Diagnostics

*MSIPL Switch Not Used; CSIPL Switch in Diskette Position:* When the Load switch is pressed, the control storage initial program load (CSIPL) sequence does three partial loads and then loads the diagnostic control program. The four control storage loads are as follows:

| IPL | | | |
|---|---|---|---|
| CSIPL | | | |
| **Load 1** | **Load 2** | **Load 3** | **Load 4** |
| Loads control processor diagnostic routines 1-19 and executes | Loads control processor diagnostic routines 20-70 and executes | Loads control processor diagnostic routines 71-79, then executes and wraps | Loads CE diagnostic supervisor and executes |

See Section 99 of the *5340 System Unit Maintenance Manual.*

*Load 1:* Hardware loads 2K words (16 sectors from track 0) into control storage at hexadecimal addresses 0000 through 07FF. These words contain the following information for control processor diagnostic routines 1 through 19:

| | Words | Addresses (Hex) |
|---|---|---|
| Direct area (the unit definition table and addresses) | 128 | 0000-007F |
| Control processor instruction tests | 1,408 | 0080-05FF |
| Diskette loader | 512 | 0600-07FF |

Then, perform the following tasks:

1. Load control storage from diskette by a burst-cycle-steal-mode operation.
2. Load 2K words (4,096 bytes) into control storage at hexadecimal addresses 0000 through 07FF.
3. Software set the microaddress register (MAR) to hexadecimal 0000 and execute the instructions for diagnostic routines 1 through 19.

If all tests run correctly, the following lights are reset to off in the sequence: bits P0, 0, and 1 of display byte 0, the Load light, and bit 2 of display byte 0.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).

See *Error Indicators* or *Display Light Sequence* in this section.

*Load 2:* The diskette loader loads 2K words (8 sectors from track 1) into control storage at hexadecimal addresses 0800 through 0FFF. These words contain the following information for control processor diagnostic routines 20 through 70:

| | Words | Addresses (Hex) |
|---|---|---|
| Remainder of control processor instruction tests | 1,792 | 0800-0EFF |
| Control storage tests | 256 | 0F00-0FFF |

Then, perform the following tasks:

1. Load control storage from diskette by an interrupt-level-mode operation.
2. Load 2K words (4,096 bytes) into control storage at hexadecimal addresses 0800 through 0FFF.
3. Software set the microaddress register (MAR) to hexadecimal 0800 and execute the instructions for diagnostic routines 20 through 70.

If all tests run correctly, bits 3 and 4 of display byte 0 are reset to off.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).

See *Error Indications* or *Display Light Sequence* in this section.

*Load 3:* The diskette loader loads 3.5K words (track 2 and six sectors of track 3) into control storage at hexadecimal addresses 0080 through 0E7F. These words contain the following information for control processor diagnostic routines 71 through 79 and device wrap tests:

| | Words | Addresses (Hex) |
|---|---|---|
| Main storage processor basic tests | 640 | 0080-027F |
| Wrap loader and control program subroutines | 128 | 0280-02FE |
| Wrap loader and control program | 512 | 02FF-047F |
| Wrap device identification and location table | 256 | 0480-057F |
| Wrap device and unit definition table | 256 | 0580-067F |
| Additional subroutines | 128 | 0680-06FF |
| Reserved | 128 | 0700-077F |
| Wrap error storage area | 128 | 0780-07FF |
| Work station display routine and CSIPL wrap error message | 640 | 0800-0A7F |
| CSIPL device wrap tests | 1,024 | 0A80-0E7F |

Then, perform the following tasks:

1. Load control storage from diskette by an interrupt-level-mode operation.

2. Load 3.5K words into control storage at hexadecimal addresses 0080 through 0E7F.

3. Software set the microaddress register (MAR) to hexadecimal 0080 and execute the instruction for diagnostic routines 71 through 79 and device wrap tests.

If all tests run correctly, bits 5, 6, and 7 of display byte 0 are reset to off.

To indicate a failure, one or more of the following occur:

- The Processor Check light is set to on.
- The Console Check light is set to on.
- Display byte 0 does not contain correct results.
- The system goes into a loop during CSIPL (display byte 0 lights show the sequence of advance).
- Error messages are stored in control storage at hexadecimal addresses 07A0 through 07BF and may also appear on the system console display screen.

See *Error Indications* or *Display Light Sequence* in this chapter.

*Load 4:* The diskette loader loads 15.75K words (8 tracks) into control storage at hexadecimal addresses 0000 through 3EFF. These words contain the diagnostic supervisor necessary to run selected diagnostic device tests.

Then, perform the following tasks:

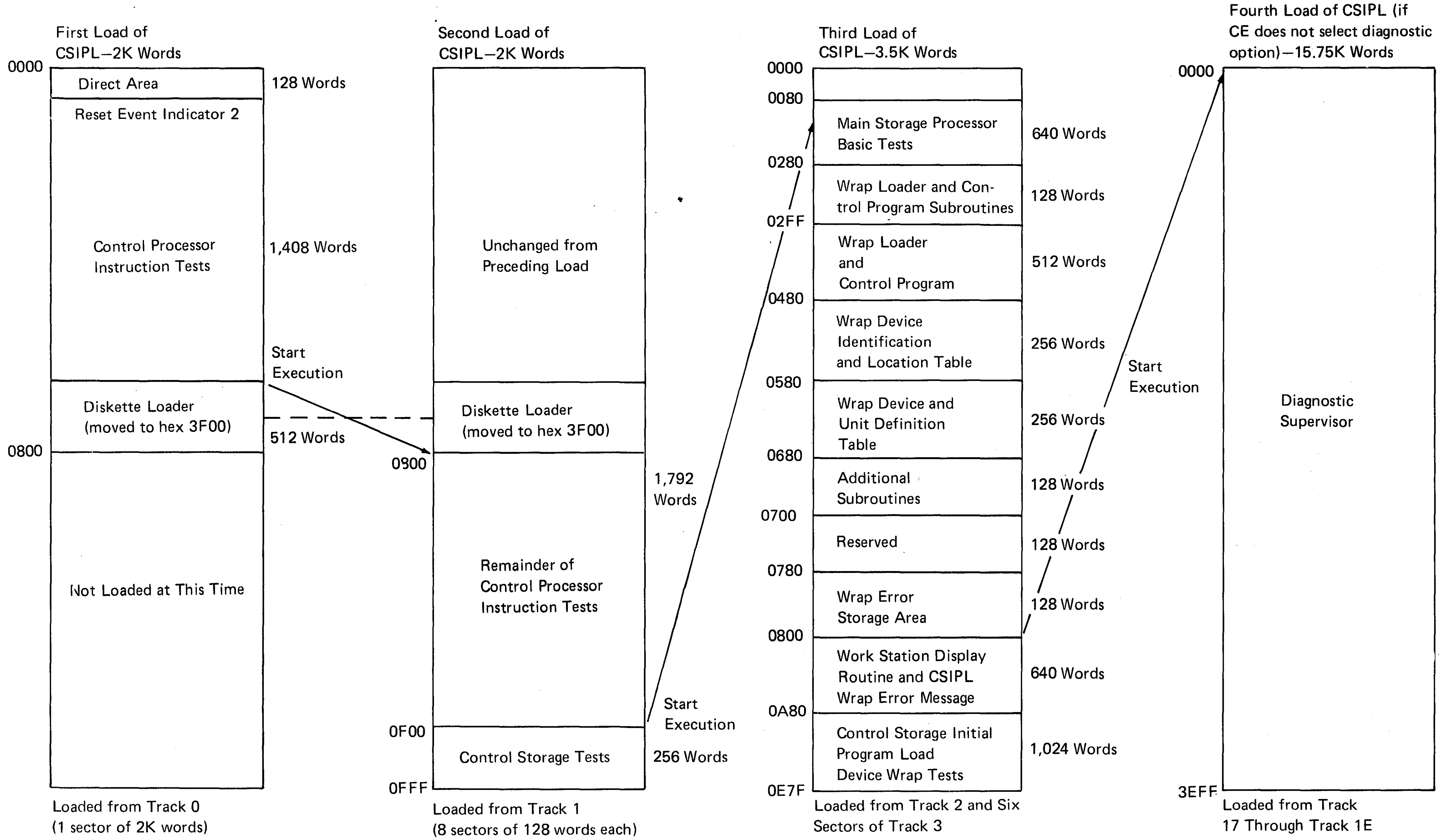
1. Load control storage from diskette by an interrupt-level-mode operation.

2. Software set the microaddress register (MAR) to hexadecimal 0000 and execute the instructions to initialize the diagnostic supervisor.

3. When the load operation is complete, the MAIN MENU message appears on the system console display screen.

Diagnostic options may be selected by the CE by using the address switches on the CE panel. For various switch settings and options, see *CSIPL Switch Options* later in this section.

## Diskette CSIPL Diagnostic Sequence



**Diskette Timing (Level 1 Attachment)**

The following charts show the sequence of events on a diskette control storage initial program load operation.

For all scope probes, ground AA2-L2J12 (DL510) + CSIPL to Dskt. Grounding this pin prevents the reset of the 'seek counter' latch and can be used to hold the head on one track.

Set the CSIPL switch to the Diskette position.

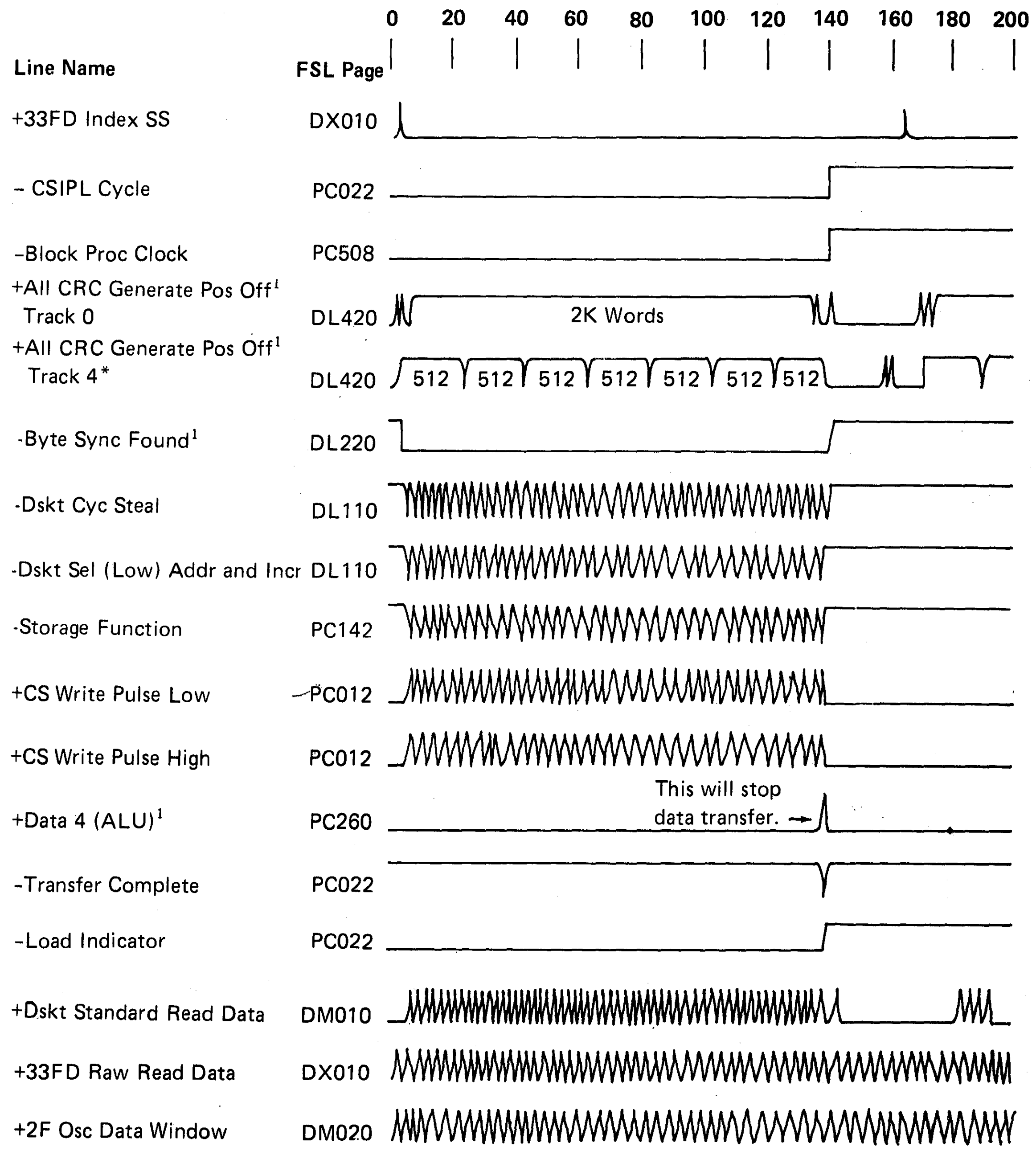
Set the Mode Selector switch to the Insn Step/Dply LSR position.

Set the Store Sel switch to the Ctl position and the Add Comp switch to the Stop position.

Set the four Address/Data switches to zero.

Sync scope Ext/DC (-) AA2-L2G07 20 ms/div (DL110) - Dskt Cyc Steal.

Press the Load switch repeatedly.



*Manually crank head to track 4.

[1] These lines cannot be probed.

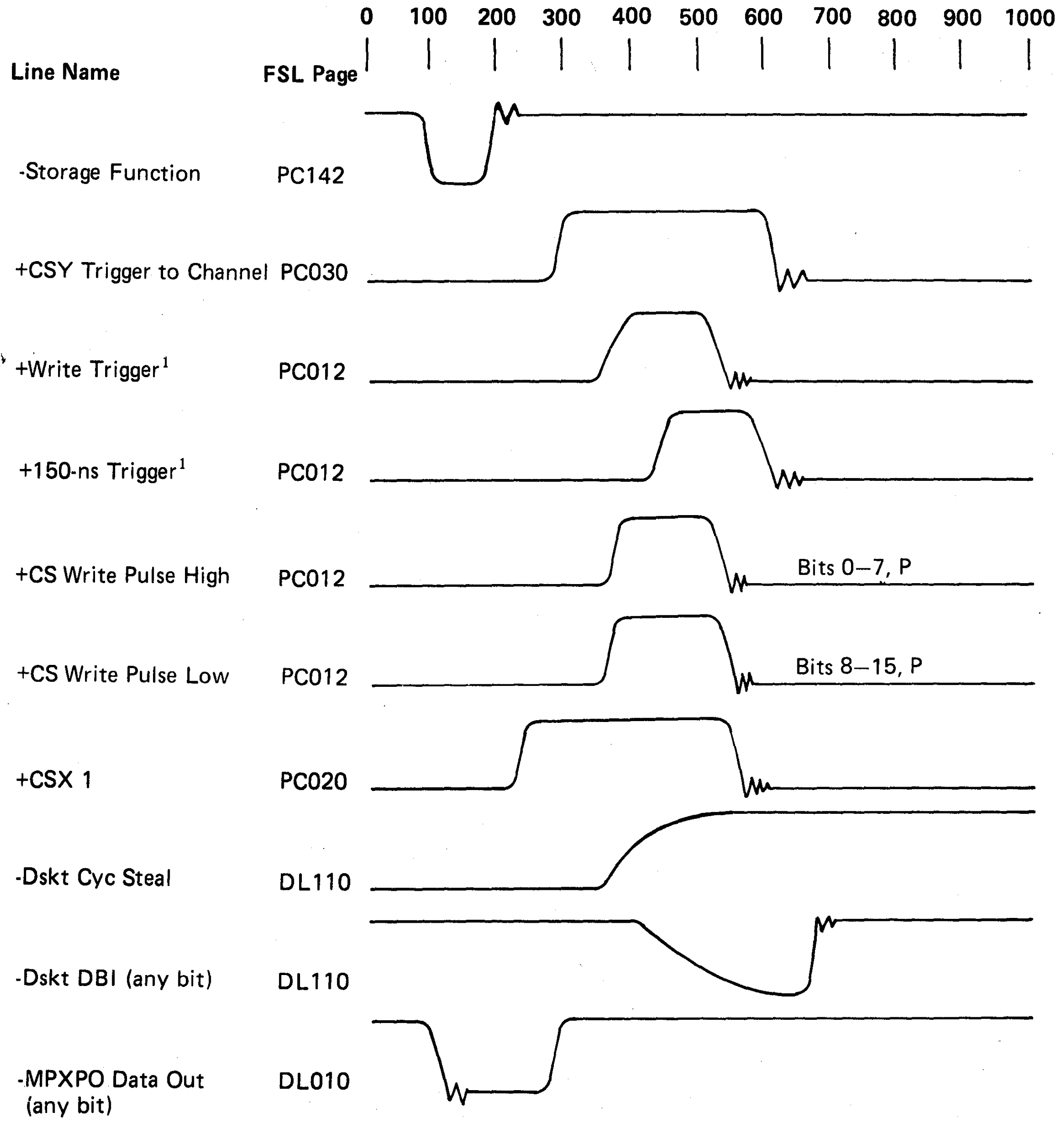
*Storage Cycle for Diskette*

Jumper AA1-H2S07 to ground (+ carry in) (PC260), which causes all data to be loaded into control storage at hexadecimal location 0000.

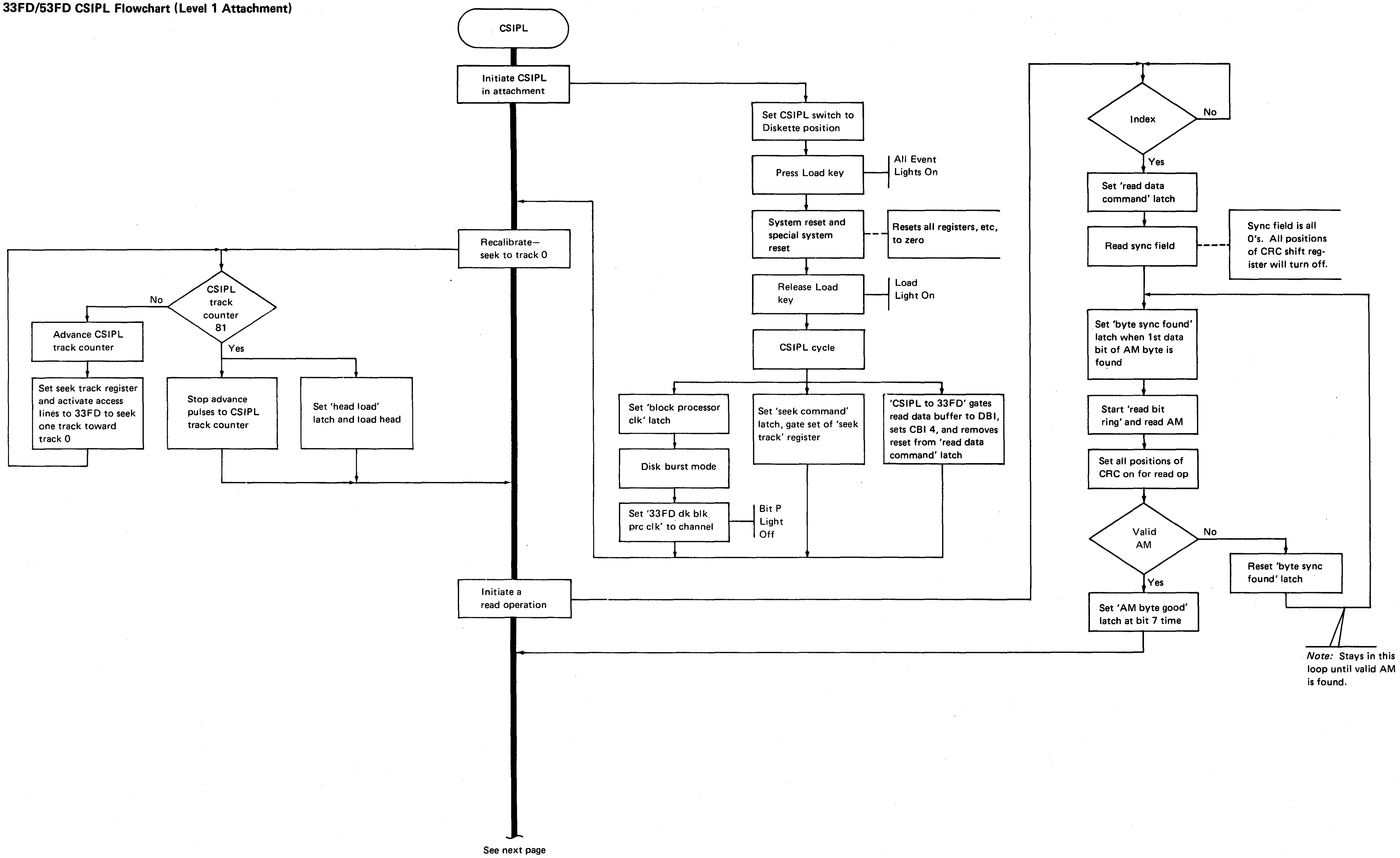
Jumper AA2-L2J12 to ground (DL510).

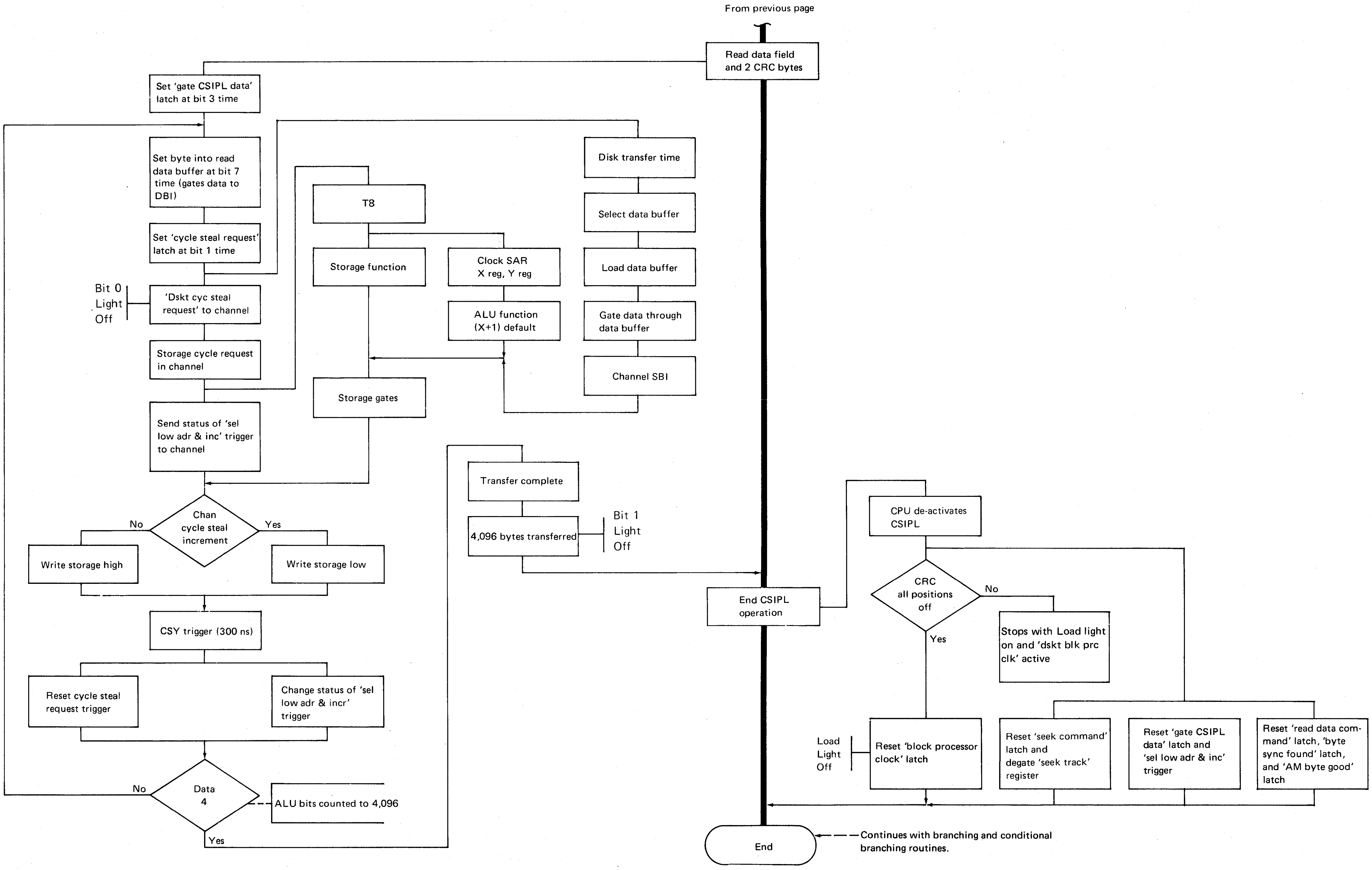
Set the CSIPL switch to the Diskette position.

Sync scope Ext/DC (-) AA1-F2J05 - storage function 100 ns/div, 2V/div.



**33FD/53FD CSIPL Flowchart (Level 1 Attachment)**


CSIPL
Initiate CSIPL in attachment
Set CSIPL switch to Diskette position
Press Load key
All Event Lights On
System reset and special system reset
Resets all registers, etc, to zero
Release Load key
Load Light On
CSIPL cycle
Set 'block processor clk' latch
Disk burst mode
Set '33FD dk blk prc clk' to channel
Bit P Light Off
Set 'seek command' latch, gate set of 'seek track' register
'CSIPL to 33FD' gates read data buffer to DBI, sets CBI 4, and removes reset from 'read data command' latch
Recalibrate—seek to track 0
CSIPL track counter 81
No
Yes
Advance CSIPL track counter
Set seek track register and activate access lines to 33FD to seek one track toward track 0
Stop advance pulses to CSIPL track counter
Set 'head load' latch and load head
Initiate a read operation
Index
No
Yes
Set 'read data command' latch
Read sync field
Sync field is all 0's. All positions of CRC shift register will turn off.
Set 'byte sync found' latch when 1st data bit of AM byte is found
Start 'read bit ring' and read AM
Set all positions of CRC on for read op
Valid AM
No
Yes
Reset 'byte sync found' latch
Set 'AM byte good' latch at bit 7 time
Note: Stays in this loop until valid AM is found.
See next page


From previous page
Read data field and 2 CRC bytes
Set 'gate CSIPL data' latch at bit 3 time
Set byte into read data buffer at bit 7 time (gates data to DBI)
Set 'cycle steal request' latch at bit 1 time
Bit 0 Light Off
'Dskt cyc steal request' to channel
Storage cycle request in channel
Send status of 'sel low adr & inc' trigger to channel
T8
Storage function
Clock SAR X reg, Y reg
ALU function (X+1) default
Storage gates
Disk transfer time
Select data buffer
Load data buffer
Gate data through data buffer
Channel SBI
No
Chan cycle steal increment
Yes
Write storage high
Write storage low
CSY trigger (300 ns)
Reset cycle steal request trigger
Change status of 'sel low adr & incr' trigger
No
Data 4
ALU bits counted to 4,096
Yes
Transfer complete
4,096 bytes transferred
Bit 1 Light Off
End CSIPL operation
CPU de-activates CSIPL
CRC all positions off
No
Stops with Load light on and 'dskt blk prc clk' active
Yes
Load Light Off
Reset 'block processor clock' latch
Reset 'seek command' latch and degate 'seek track' register
Reset 'gate CSIPL data' latch and 'sel low adr & inc' trigger
Reset 'read data command' latch, 'byte sync found' latch, and 'AM byte good' latch
End
Continues with branching and conditional branching routines.

**Diskette Timing (Level 2 Attachment)**

The following charts show the sequence of events on a diskette control storage initial program load operation.

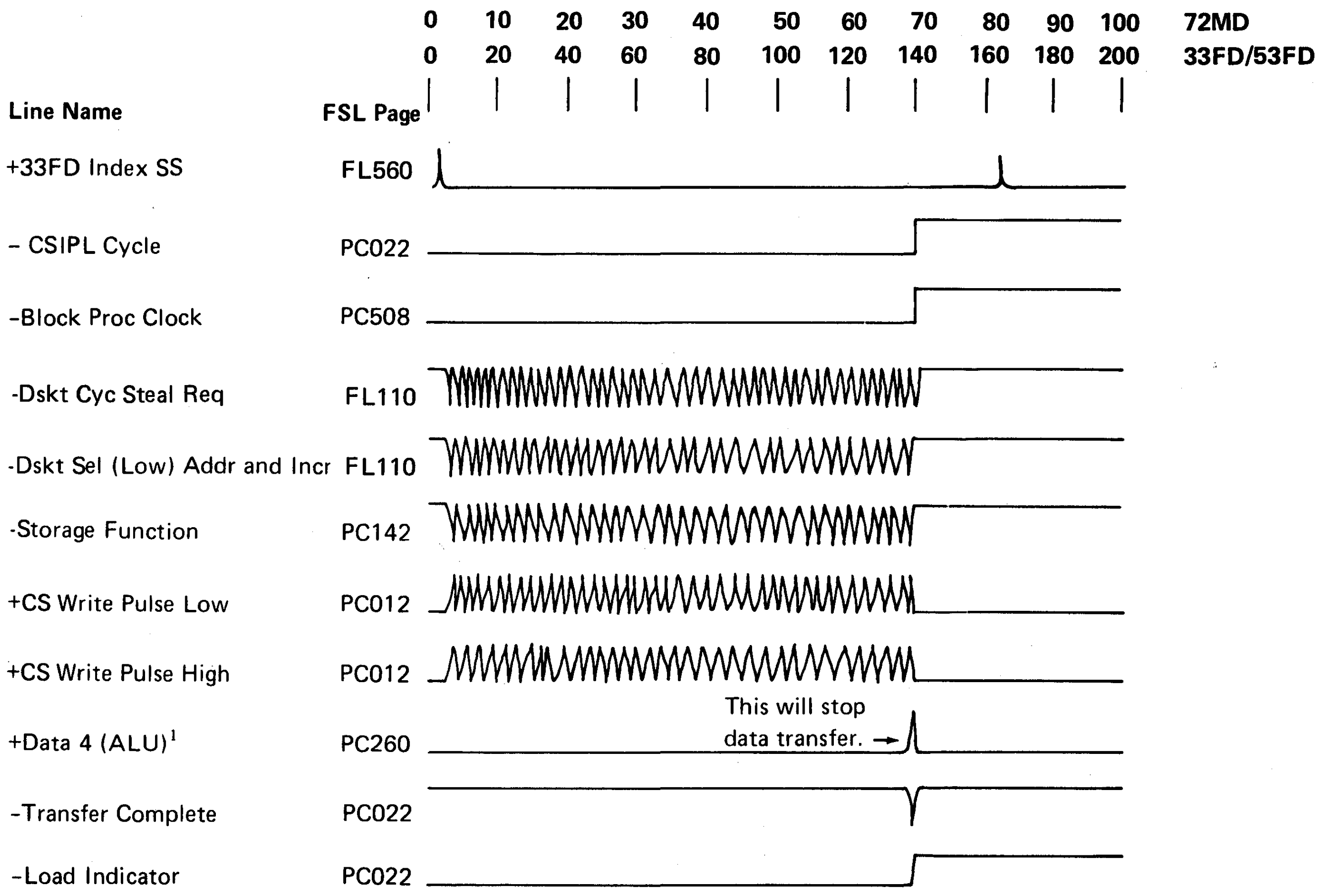
Set the CSIPL switch to the Diskette position.

Set the Store Sel switch to the Ctl position and the Add Comp switch to the Stop position.

Set the four Address/Data switches to zero.

Sync score Ext/DC (-) AA2-L2G07 10 ms/div (72MD) or 20 ms/div (33FD/53FD) (FL110)–Dskt Cyc Steal Req.

Press the Load switch repeatedly.



*Storage Cycle for Diskette*

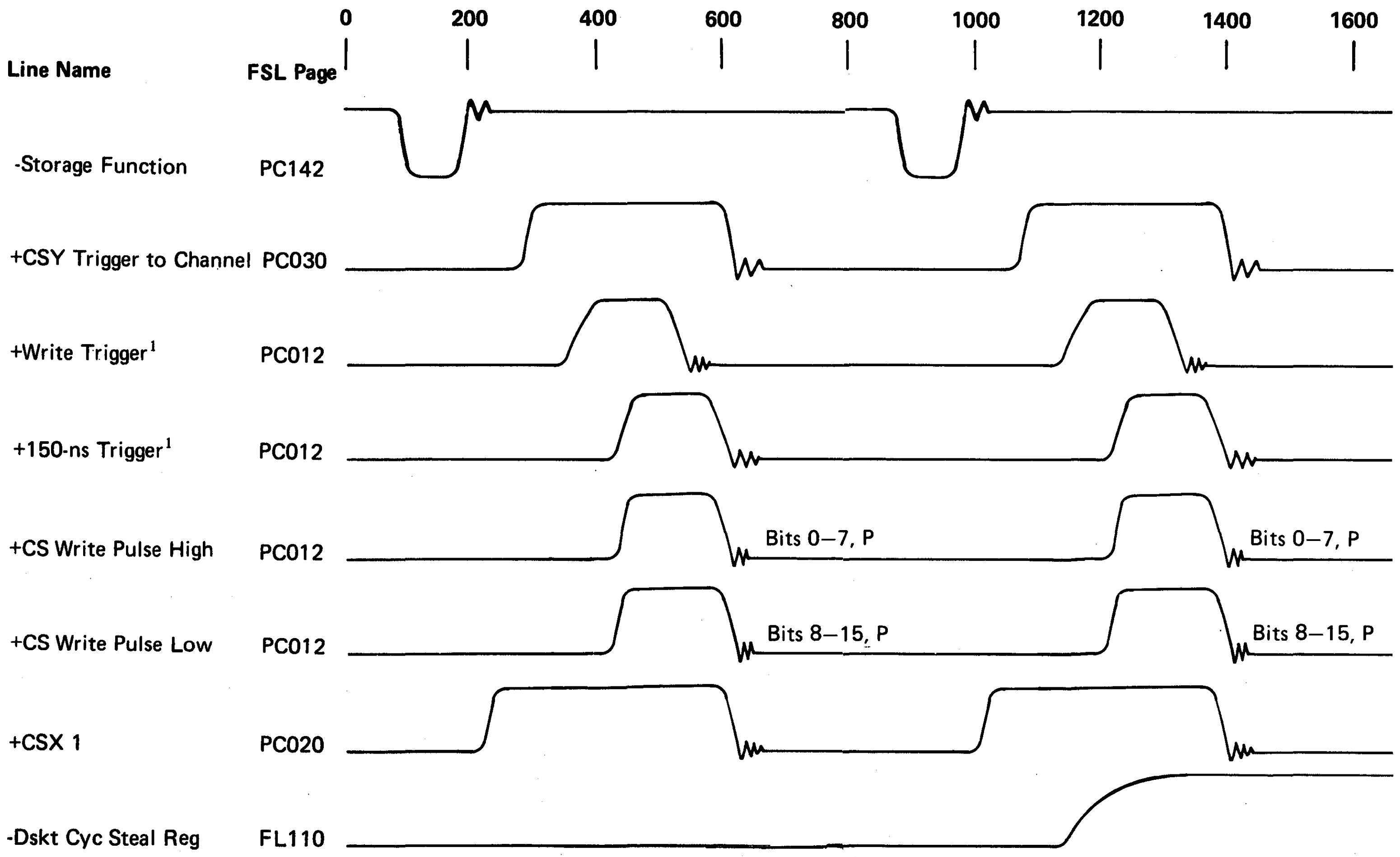
Jumper AA1-H2S07 to ground (+ carry in) (PC260), which causes all data to be loaded into control storage at hexadecimal location 0000.

Set the CSIPL switch to the Diskette position.

Set the Add Comp switch to the Run position.

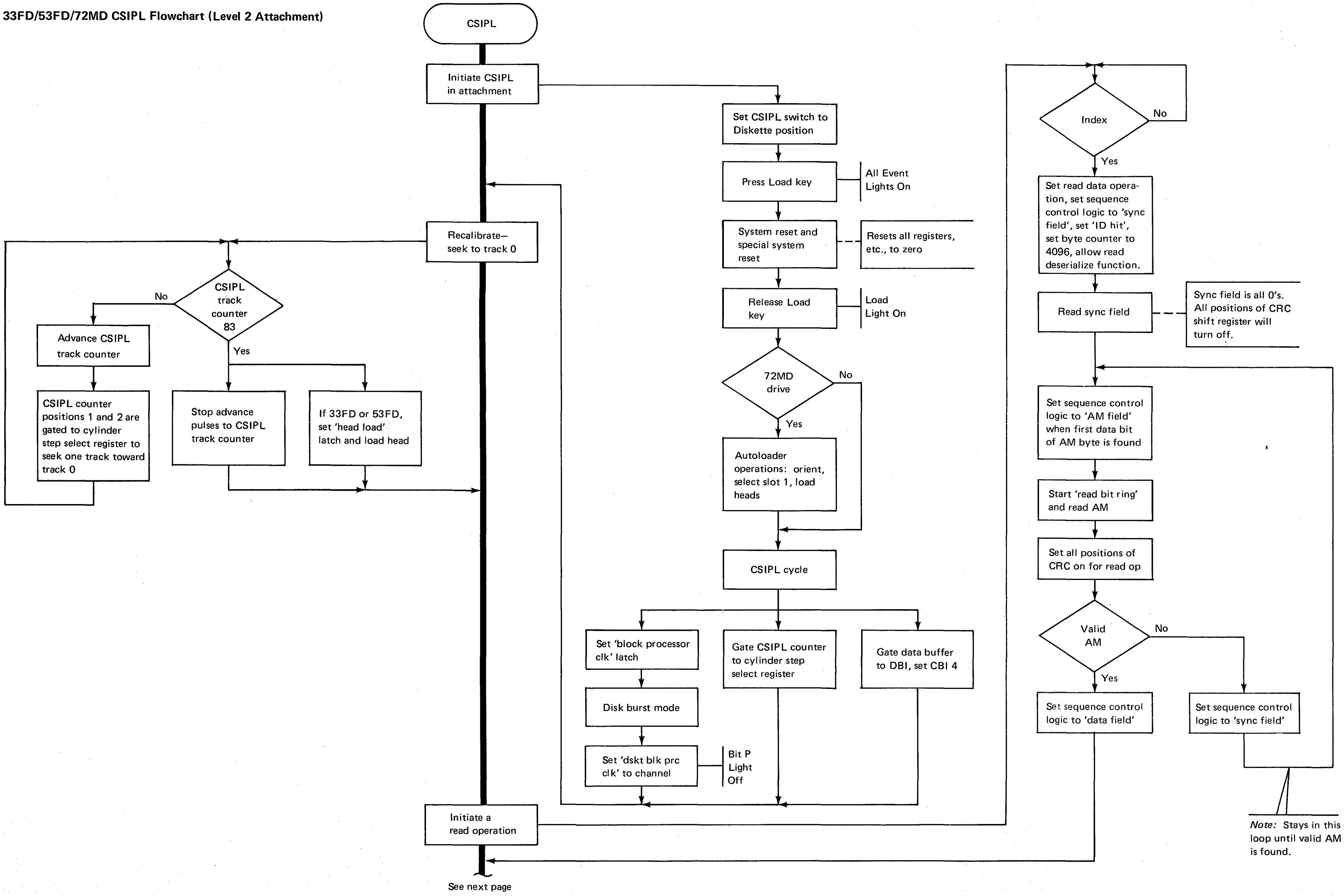
Sync scope Ext/DC (-) AA1-F2J05 - storage function 200 ns/div, 2V/div.

Press the Load switch.

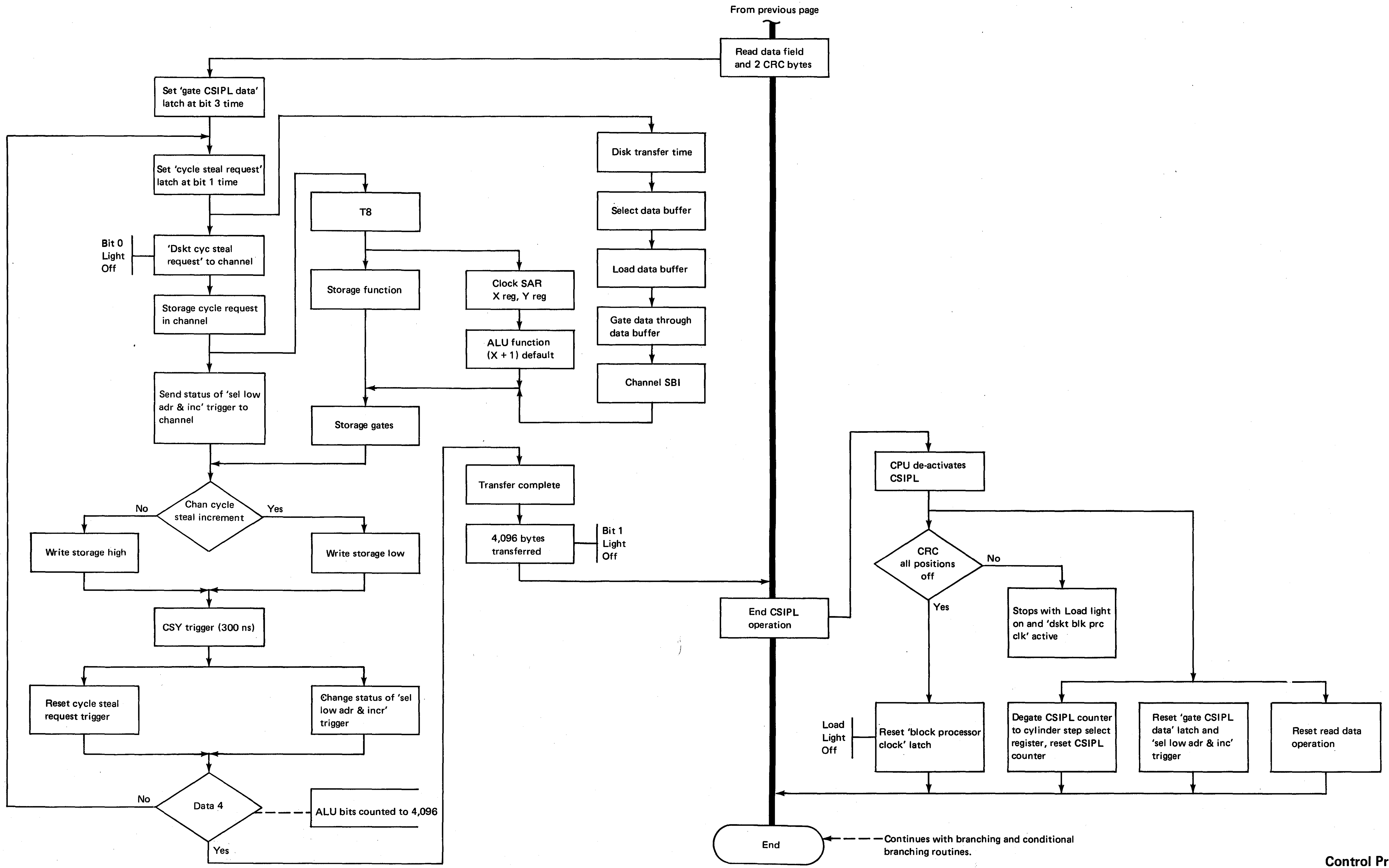


[1] These lines cannot be probed.

## 33FD/53FD/72MD CSIPL Flowchart (Level 2 Attachment)


CSIPL
Initiate CSIPL in attachment
Recalibrate– seek to track 0
CSIPL track counter 83
No
Yes
Advance CSIPL track counter
CSIPL counter positions 1 and 2 are gated to cylinder step select register to seek one track toward track 0
Stop advance pulses to CSIPL track counter
If 33FD or 53FD, set 'head load' latch and load head
Set CSIPL switch to Diskette position
Press Load key
All Event Lights On
System reset and special system reset
Resets all registers, etc., to zero
Release Load key
Load Light On
72MD drive
No
Yes
Autoloader operations: orient, select slot 1, load heads
CSIPL cycle
Set 'block processor clk' latch
Disk burst mode
Set 'dskt blk prc clk' to channel
Bit P Light Off
Gate CSIPL counter to cylinder step select register
Gate data buffer to DBI, set CBI 4
Initiate a read operation
See next page
Index
No
Yes
Set read data operation, set sequence control logic to 'sync field', set 'ID hit', set byte counter to 4096, allow read deserialize function.
Read sync field
Sync field is all 0's. All positions of CRC shift register will turn off.
Set sequence control logic to 'AM field' when first data bit of AM byte is found
Start 'read bit ring' and read AM
Set all positions of CRC on for read op
Valid AM
No
Yes
Set sequence control logic to 'data field'
Set sequence control logic to 'sync field'

*Note:* Stays in this loop until valid AM is found.


From previous page
Read data field and 2 CRC bytes
Set 'gate CSIPL data' latch at bit 3 time
Set 'cycle steal request' latch at bit 1 time
Bit 0 Light Off
'Dskt cyc steal request' to channel
Storage cycle request in channel
Send status of 'sel low adr & inc' trigger to channel
Chan cycle steal increment
No
Yes
Write storage high
Write storage low
CSY trigger (300 ns)
Reset cycle steal request trigger
Change status of 'sel low adr & incr' trigger
Data 4
No
Yes
ALU bits counted to 4,096
T8
Storage function
Clock SAR X reg, Y reg
ALU function (X + 1) default
Storage gates
Disk transfer time
Select data buffer
Load data buffer
Gate data through data buffer
Channel SBI
Transfer complete
4,096 bytes transferred
Bit 1 Light Off
End CSIPL operation
End
Continues with branching and conditional branching routines.
CPU de-activates CSIPL
CRC all positions off
No
Yes
Stops with Load light on and 'dskt blk prc clk' active
Load Light Off
Reset 'block processor clock' latch
Degate CSIPL counter to cylinder step select register, reset CSIPL counter
Reset 'gate CSIPL data' latch and 'sel low adr & inc' trigger
Reset read data operation

### Error Indications

If you press the Load switch and the correct display does not appear in the specified time (less than 90 seconds) and the display lights do not reset to off, you should suspect a machine failure.

Check the setting of the CE panel switches; then, check that the correct diskette is inserted in the machine correctly.

There are two types of machine errors: wrap test errors and processor check errors.

*Wrap test errors:* If the control storage initial program load diagnostic wrap test finds an error in a device adapter, the system console usually can be used to display the error as shown.

If this display appears, the same information is in the main program level work registers (1-6) and in control storage at hexadecimal locations 07A0-07BF.

If the display option is not taken, the wrap test errors remain in control storage at hexadecimal locations 07A0-07BF.

```
1 WRAP ERROR DISPLAY
2 AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD
3 AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD AABBCCDD
4  PRESS 'ENTER' TO CONTINUE SYS—0019 ERROR
```

*Note:* Initial program load uses only the top four lines on the console display.

Header decode for wrap errors is as follows:

| | |
|---|---|
| AA | Device Identification |
| BB | Device Address |
| CC | Unit Address |
| DD | Wrap Module Number (for diskette and line printer, DD equals the TU that failed) |

| Wrap Error AABBCCDD | Device |
|---|---|
| 020000XX | Main Storage Processor |
| A0A000XX | 62EH Disk Drive A |
| A0B000XX | 62EH Disk Drive B |
| A1A000XX | 62PC Disk Drive A |
| A1A001XX | 62PC Disk Drive B |
| CAC000XX | Work Station Attachment |
| C0C0XXXX | Work Station |
| D0D000XX | Diskettes (Level 1) |
| D1D000XX | Diskettes (Level 2) |
| E2E000XX | 3262 Line Printer |
| E0E000XX | 5211 Line Printer |
| 80XX01XX | Data Communications |
| 525000XX | 1255 Magnetic Character Reader |

*Processor Check Errors:* The control storage initial program load diagnostic tests find an error and force a processor check (processor check halt instruction). The low-order byte of work register 3 contains the number of the failing routine when the failure occurs in routine 09 or above. Check the display lights (byte 0 on the CE panel) to determine when the failure occurred. See Section 99 of the *5340 System Unit Maintenance Manual* to determine which specific function of the machine failed.

A control processor check during normal operation (running under control of SSP) will do a log operation of that error under the following conditions and sequence:

1. A control processor check occurs during normal operation (interrupt level 0 instruction stop condition).

2. The system operator does an IPL with no processor check (problem is intermittent).

3. Error information (from normal operation) is stored in control storage while executing the CSIPL (first load–2K word control processor diagnostic routines 1 through 19).

4. After the system operator has completed the IPL sign-on message task, the error information is logged from control storage to one complete sector on the disk.

5. IPL is completed with the log information on the disk as shown in the following error history table.

A main storage processor check causes an interrupt level 5 to the control processor and the error log operation is then executed by the control processor. The following information is contained in the error history table:

```
ERROR HISTORY TABLE FOR CONTROL PROCESSOR
        BYT BYT
PCR IL   0   1  WR0  WR1  WR2  WR3  WR4  WR5  WR6  WR7  MAR  MAB    DATE   TIME
.............................. HEX ..............................  YYMMDD HHMMSS
C2  07  24  00  0000 24C2 0A02 8000 0000 674D 9200 2020 0000 21B0  770518 150200
C2  07  04  00  0000 04C2 0A02 8000 0000 674D 9200 2020 0000 21B0  770518 145500
92  07  08  00  A2F1 0892 1140 0200 0000 0000 0000 0006 A2F2 21AA  770518 145200
C2  00  02  00  1000 02C2 1CAC 3800 0000 0000 A200 0008 159B 1597  770518 144800
01  07  80  00  00E0 8001 00F7 0080 0000 0000 9200 0008 0000 21AA  770518 144600
A2  07  02  00  23FF 02A2 033D 4078 1040 1043 1001 0040 23B0 231C  770518 143800
91  07  B0  00  0000 B091 EEA2 2000 0000 0041 A228 2027 0152 0146  770518 111500
92  07  08  00  88C0 0892 0000 0080 0000 0000 9228 0008 88C1 21AA  770518 101500
C2  07  38  00  011C 38C2 BEA3 BEA3 BEA3 BEA3 0000 227E 227E 21A9  770518 083400
A2  07  20  00  0840 20A2 5AFA A2F1 0000 0000 0000 2021 21D0 21AE  770518 082500
A2  07  20  00  0E00 20A2 0141 BEA3 0000 0000 0000 0000 21A8 21B0  770518 081500
22  07  08  00  0177 0822 0C00 0000 21B4 B180 21B5 C3F2 F3B5 21B4  770518 080300
****************************** END OF TABLE ************************************
```

PCR Processor Condition Register
IL Interrupt Level
Byte 0 Control Processor Check Byte Information
Byte 1 Channel Check Byte Information
WR0
WR1
WR2
WR3 Contents of the Work Registers Specified by the Interrupt Level Value
WR4
WR5
WR6
WR7
MAR Microaddress Register contents of present Interrupt Level
MAB Microaddress Backup Register contents of present Interrupt Level

*Note:* The 16 most current errors are stored.

```
ERROR HISTORY TABLE FOR MAIN STORAGE PROCESSOR
                                 ATRS        PROG  STATUS   FAIL.
IAR  ARR  XR1  XR2  OP1  OP2  IR 01 02 OP Q  MR SR 0  2  3  ADDR.  DATE   TIME
............................. HEX ................................ YYMMDD HHMMSS
C801 0003 C818 D270 D24C CB56 01 0D 08 00 01 06 01 04 8A 03 000000 770519 150550
C801 0003 0F11 D678 0639 0F10 01 00 01 FF 01 06 01 04 A8 00 000000 770519 150518
******************************* END OF TABLE ***********************************
```

IAR Instruction Address Register
ARR Address Recall Register
XR1 Index Register 1
XR2 Index Register 2
OP1 Operand 1
OP2 Operand 2
ATRS/IR Address Translation Register used by the Instruction Address Register
ATRS/01 Address Translation Register used by Operand 1
ATRS/02 Address Translation Register used by Operand 2
OP Operation Code
Q Q-byte Register Contents
PROG/MR Program Mode Register
PROG/SR Program Status Register
STATUS/0 Main Storage Processor Register Status Byte 0
STATUS/2 Main Storage Processor Register Status Byte 2
STATUS/3 Main Storage Processor Register Status Byte 3

*Note:* The 16 most current errors are stored.

This page intentionally left blank.

### CSIPL Switch Options

See Section 99 of the *5340 System Unit Maintenance Manual* for references given below. CSIPL options that can be changed by use of the Address/Data switches are:

| Address/Data Switch Settings | Function Performed |
|---|---|
| F100 | Bypasses wrap tests and executes work station MDI MAPs (see paragraph 99-062). |
| F101 | Bypasses wrap tests and executes work station TU select (see paragraph 99-064). |
| F180 | Runs work station diagnostics and prints results (step mode) (see paragraph 99-062). |
| F181 | Bypasses wrap tests, executes work station TU select, and prints results (see paragraph 99-064). |
| F800 | Loads the diagnostic supervisor from disk (use this option to run MDI tests for the diskette). |
| FA01 | Stops after the first load and permits changing of the Address/Data switches to FB01 or FC01. |
| FA02 | Stops after the second load and permits changing of the Address/Data switches to FB02 or FC02. |
| FB01 | Loops on CSIPL number 1 and stops on errors. |
| FB02 | Loops on CSIPL number 2 and stops on errors. |
| FC01 | Loops on CSIPL number 1 and bypasses errors. |
| FC02 | Loops on CSIPL numbers 1 and 2 and bypasses errors. |
| FDXX | Loops on CSIPL routine xx (routines 9 through 64 only) (see paragraph 99-020). |
| FEXX | Loops on CSIPL routine xx and bypasses errors (except errors in routines that test control storage or main storage) (see paragraph 99-020 for a list of valid routine xx numbers). |
| FFXX | Bypasses selected wrap tests indicated for the device with an identification of xx (see paragraph 99-060 for a complete list as shown on this page). |
| EE00 | Loads and executes the main storage processor MAP diagnostic integration programs (see paragraph 99-015). |
| FF00 | Bypasses all wrap tests and skips control processor tests that are affected by the system configuration. Used to do a special load from a diskette that has not been configured. Use this setting if a CE diskette from another system with a different storage or system configuration is used. Also use this setting if additional storage is being added to the system and the CE diskette has not yet been given the correct configuration. |
| 0000 | Normal position–runs all wrap tests. |

| Option for FFXX Device Address (Hexadecimal) | Functions |
|---|---|
| 00 | Bypasses configuration tests and wrap tests |
| 02 | Main storage processor |
| 52 | 1255 (MICR) |
| 80 | Communications |
| A0 | 62EH disk A wrap test |
| A1 | 62PC disk A or B wrap test |
| B0 | 62EH disk B wrap test |
| C0 | Work station wrap test |
| CA | Work station controller wrap test |
| D0 | Diskette wrap test |
| E0 | Printer wrap test |

## Instructions

The System/34 control storage program performs the following functions:

- Reads, decodes, and operates on data and system instructions in main storage that are not executed by the main storage processor. The supervisor call (SVC) instruction executed by the main storage processor sets interrupt level 5 in the control processor.
- Performs I/O operations for the system attachments.
- Performs console operations.
- Performs diagnostic operations.
- Performs task management functions.

The control storage program performs functions in the system operation. Each function has many instruction steps and may use several routines or part of a routine to complete its task. These instructions are executed in a specific sequence. To change the sequence, a branch-and-link instruction can be used to permit branching to another routine. A branch-and-link instruction stores a link address, which is the address of the next sequential instruction to be executed in the branched-from routine. The program can then return to the instruction after the branch-and-link instruction. Jump instructions and branch instructions are also used to change the instruction sequence. These instructions are described later in this section.

Each instruction is a 16-bit word that represents a machine instruction. This instruction has specific fields specified for controlling data flow of the system. A zone digit is the hexadecimal value represented in the 4 high-order bits (bits 0-3) of a byte. A numeric digit is the hexadecimal value represented in the 4 low-order bits (bits 4-7) of a byte. System/34 uses 20 basic instructions. Bits 0-3 of the instruction identify the type of instruction. The 20 instructions are described under *Instruction Execution* later in this section.

## Instruction Times

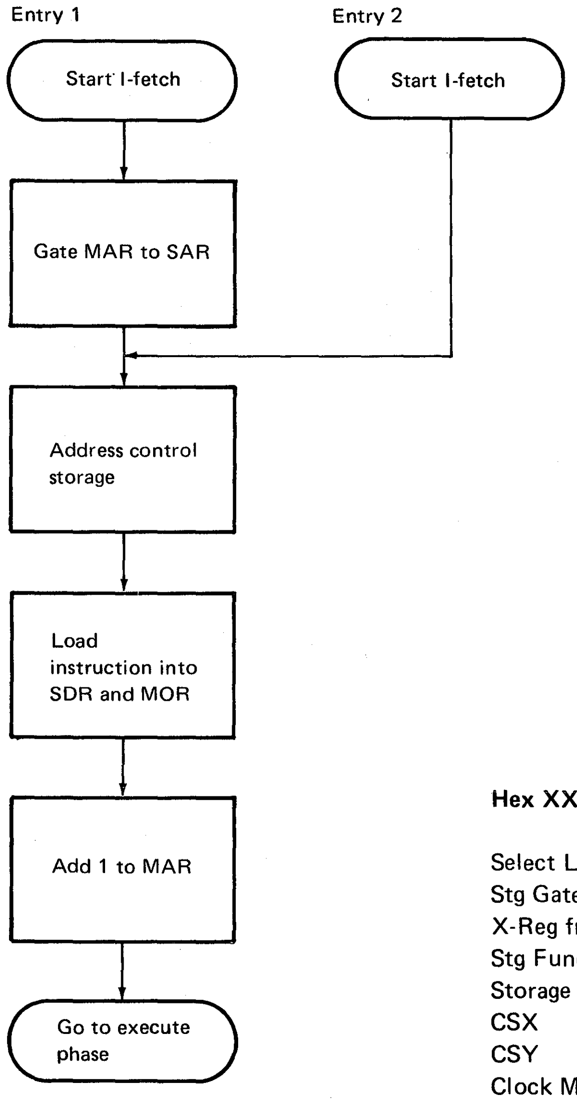
Instructions are executed in two times: an instruction fetch time (I-time) and an execution time (E-time). During I-time, the control processor:

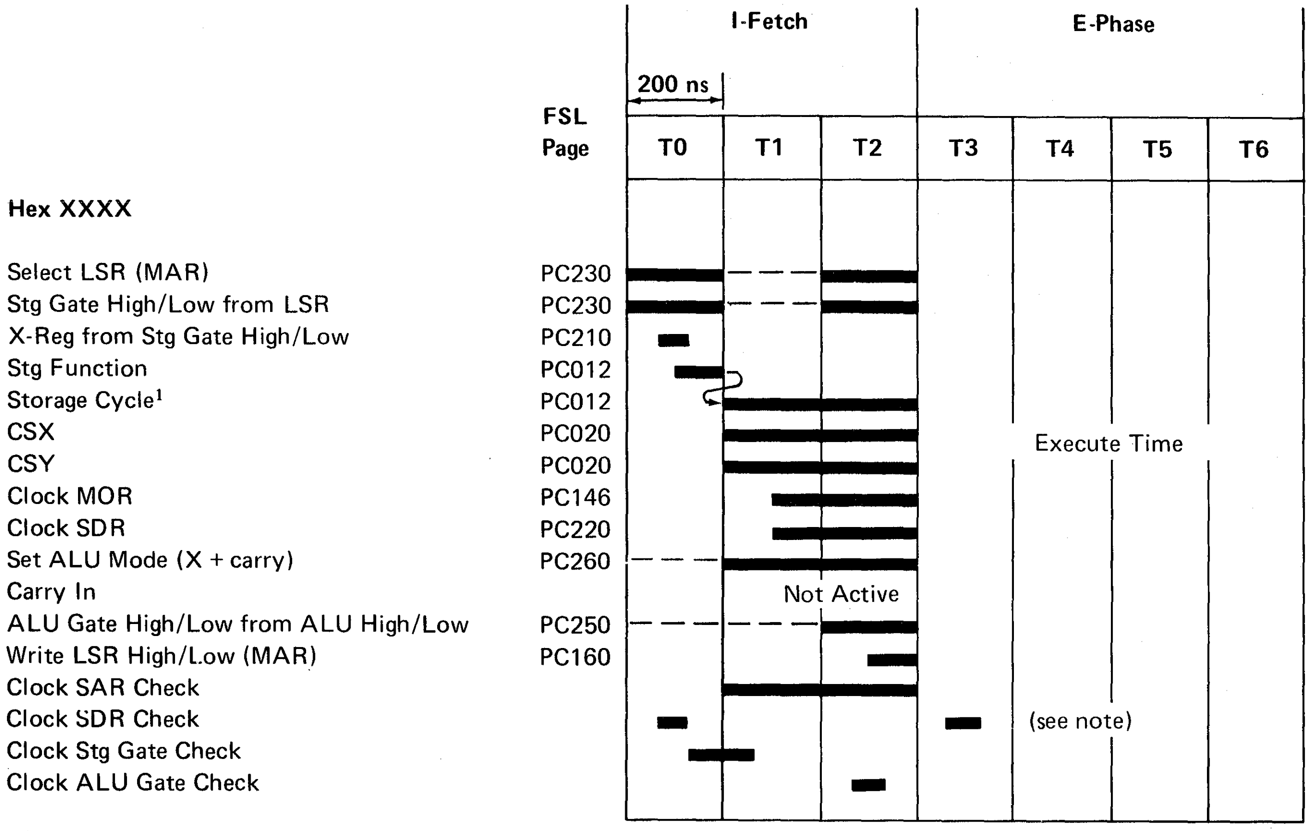
- Loads a control storage address from the microaddress register (MAR) into the storage address register (SAR).
- Addresses the control storage address in SAR.
- Gates the instruction from this address into the storage data register (SDR) and micro-operation register (MOR).
- Adds 1 to the microaddress register (MAR).

For specific events that occur during the execution time, see the specific instruction description later in this section.

### Sequence and Timing

A printout of the instructions may be obtained by using the diagnostic utilities program (see paragraph 99-055 of the *5340 System Unit Maintenance Manual*). Module name identification may be indexed by using Section 4 of the *Control Storage Logic Manual.* Shown below is a sample printout.





[1] This line cannot be probed.

*Note:* SDR check after T2 actually is gated during E-cycle time T3.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | 50FF | TM |
| 01 | 50FF | TM * |
| 02 | 0000 | B |

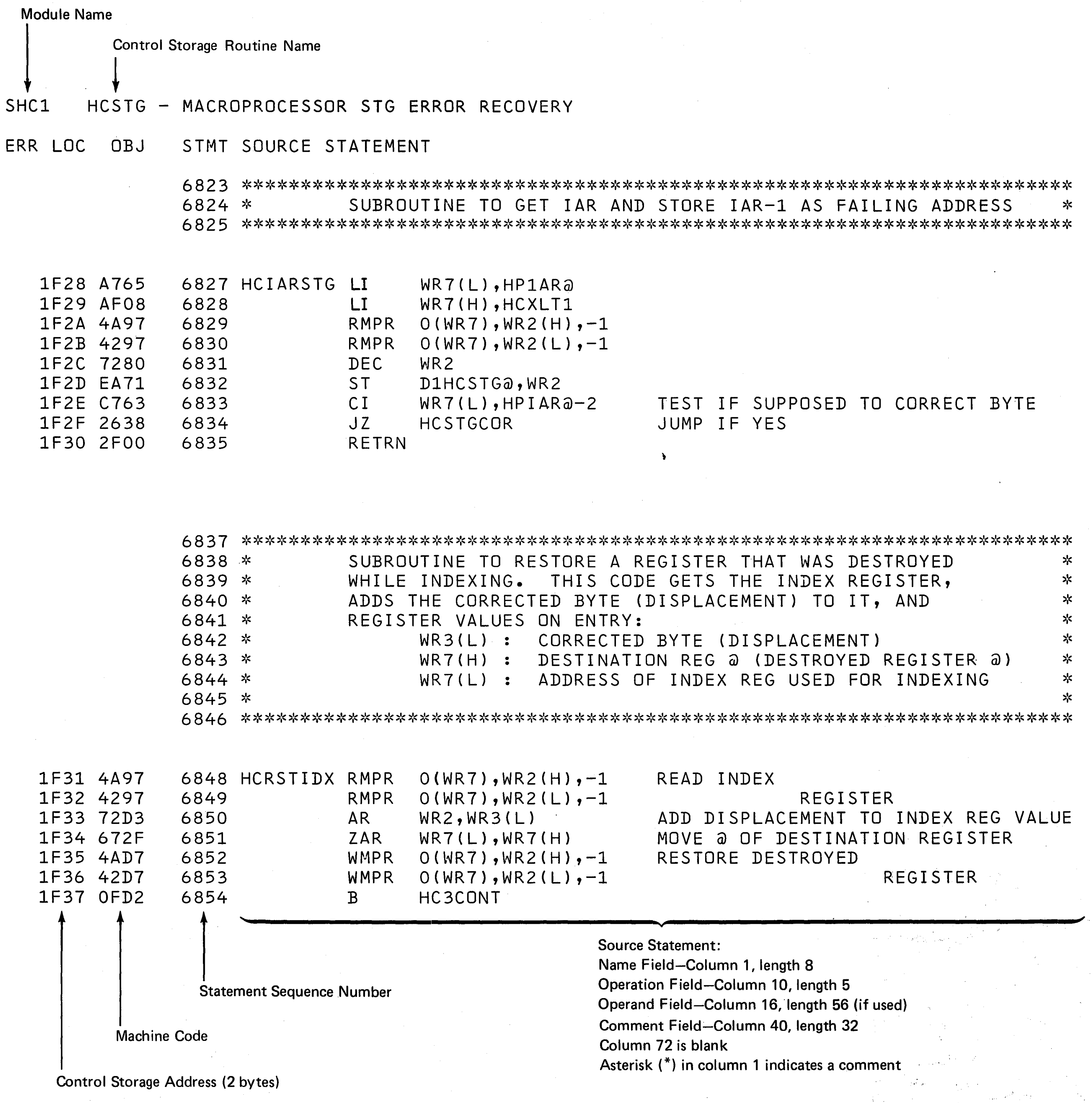
**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Routine Printout



Module Name

Control Storage Routine Name

```
SHC1   HCSTG - MACROPROCESSOR STG ERROR RECOVERY

ERR LOC  OBJ   STMT SOURCE STATEMENT

               6823 ***********************************************************************
               6824 *        SUBROUTINE TO GET IAR AND STORE IAR-1 AS FAILING ADDRESS     *
               6825 ***********************************************************************


   1F28 A765   6827 HCIARSTG LI    WR7(L),HP1AR@
   1F29 AF08   6828          LI    WR7(H),HCXLT1
   1F2A 4A97   6829          RMPR  0(WR7),WR2(H),-1
   1F2B 4297   6830          RMPR  0(WR7),WR2(L),-1
   1F2C 7280   6831          DEC   WR2
   1F2D EA71   6832          ST    D1HCSTG@,WR2
   1F2E C763   6833          CI    WR7(L),HPIAR@-2     TEST IF SUPPOSED TO CORRECT BYTE
   1F2F 2638   6834          JZ    HCSTGCOR            JUMP IF YES
   1F30 2F00   6835          RETRN


               6837 ***********************************************************************
               6838 *        SUBROUTINE TO RESTORE A REGISTER THAT WAS DESTROYED          *
               6839 *        WHILE INDEXING.  THIS CODE GETS THE INDEX REGISTER,          *
               6840 *        ADDS THE CORRECTED BYTE (DISPLACEMENT) TO IT, AND            *
               6841 *        REGISTER VALUES ON ENTRY:                                    *
               6842 *              WR3(L) :  CORRECTED BYTE (DISPLACEMENT)                *
               6843 *              WR7(H) :  DESTINATION REG @ (DESTROYED REGISTER @)     *
               6844 *              WR7(L) :  ADDRESS OF INDEX REG USED FOR INDEXING       *
               6845 *                                                                     *
               6846 ***********************************************************************

   1F31 4A97   6848 HCRSTIDX RMPR  0(WR7),WR2(H),-1    READ INDEX
   1F32 4297   6849          RMPR  0(WR7),WR2(L),-1                REGISTER
   1F33 72D3   6850          AR    WR2,WR3(L)          ADD DISPLACEMENT TO INDEX REG VALUE
   1F34 672F   6851          ZAR   WR7(L),WR7(H)       MOVE @ OF DESTINATION REGISTER
   1F35 4AD7   6852          WMPR  0(WR7),WR2(H),-1    RESTORE DESTROYED
   1F36 42D7   6853          WMPR  0(WR7),WR2(L),-1                       REGISTER
   1F37 0FD2   6854          B     HC3CONT
```

Statement Sequence Number

Machine Code

Control Storage Address (2 bytes)

Source Statement:
Name Field–Column 1, length 8
Operation Field–Column 10, length 5
Operand Field–Column 16, length 56 (if used)
Comment Field–Column 40, length 32
Column 72 is blank
Asterisk (*) in column 1 indicates a comment

## Mnemonic Listing

| Instruction | Mnemonic | Operation Code | Function or Instruction Definition |
|---|---|---|---|
| Branch | B | 0 | |
| Branch and link | BAL | 1 | |
| Jump on condition (includes a group of instruction sets) | | 2 | Bits 4-7 specify the jump condition |
| Jump on carry | JCY | | 0000 |
| Jump on high | JH | | 0001 |
| Jump on low | JL | | 0010 |
| Jump on equal | JE | | 0011 |
| Jump on positive | JP | | 0100 |
| Jump on all ones | JO | | 0100 |
| Jump on negative | JN | | 0101 |
| Jump on mixed | JM | | 0101 |
| Jump on zero | JZ | | 0110 |
| Jump on flag | JFLG | | 0111 |
| Jump on service request | JSR | | 1000 |
| Jump on not high | JNH | | 1001 |
| Jump on not low | JNL | | 1010 |
| Jump on not equal | JNE | | 1011 |
| Jump on not positive | JNP | | 1100 |
| Jump on not negative | JNN | | 1101 |
| Jump on not zero | JNZ | | 1110 |
| Return | RETRN | | 1111 |
| Jump on input/output condition | JIO | 3 | |
| Input/output storage | | 4 | Bit 8 = 0 |
| Write to control storage high/low from input/output | WTCH/L | | |
| Read from control storage high/low to input/output | RDCH/L | | |
| Write to main storage from input/output | WTM | | |
| Read from main storage to input/output | RDM | | |
| Storage | | 4 | Bit 8 = 1 |
| Load from control storage | LC | | |
| Store to control storage | STC | | |

| Instruction | Mnemonic | Operation Code | Function or Instruction Definition |
|---|---|---|---|
| Storage (continued) | | | |
| Load from main storage | LM | | |
| Store to main storage | STM | | |
| Register control | | 4 | Bit 8 = 1 |
| Load main storage processor register | WMPR | | Bits 9-12 = 1010 |
| Sense main storage processor register | RMPR | | Bits 9-12 = 0010 |
| Test mask | TM | 5 | |
| Logical/arithmetic 1 | | 6 | Bits 8-11 specify the function |
| Zero and add register | ZAR | | 0010 |
| Exclusive OR | XR | | 0001 |
| OR | OR | | 0011 |
| AND register | NR | | 0110 |
| AND complement | NCR | | 0101 |
| OR complement | OCR | | 0111 |
| Decrement register by 1 | DEC | | 1000 |
| Add registers with carry | ACYR | | 1001 |
| Subtract register | SR | | 1100 |
| Add register | AR | | 1011 |
| Shift left logical | SLL | | 1011 |
| Subtract with borrow | SCYR | | 1110 |
| Increment register by 1 | INC | | 1111 |
| Logical/arithmetic 2 | | 7 | Bits 8-11 specify the function |
| Zero and add register | ZAR | | 0010 |
| Exclusive OR | XR | | 0001 |
| OR | OR | | 0011 |
| AND register | NR | | 0110 |
| AND complement | NCR | | 0101 |
| OR complement | OCR | | 0111 |

| Instruction | Mnemonic | Operation Code | Function or Instruction Definition | |
|---|---|---|---|---|
| Logical/arithmetic 2 (continued) | | | | |
| Decrement register by 1 | DEC | 7 | 1000 | |
| Add registers with carry | ACYR | | 1001 | |
| Subtract register | SR | | 1100 | 2 bytes from 2 bytes |
| | SR | | 1010 | 1 high or low byte from 2 bytes<br>Bit 12 = 0: Low<br>Bit 12 = 1: High |
| Add register | AR | | 1011 | 2 bytes to 2 bytes<br>Bit 12 = 0: Low<br>Bit 12 = 1: High |
| | AR | | 1101 | 1 high or low byte to 2 bytes |
| Shift left logical double | SLLD | | 1011 | |
| Subtract with borrow | SCYR | | 1110 | |
| Increment register by 1 | INC | | 1111 | |
| Set bits off | SBF | 8 | | |
| Set bits on | SBN | 9 | | |
| Load immediate | LI | A | | |
| Input/output immediate | | B | Bits 8-11 specify the function | |
| Input/output load | IOL | | 0000 | |
| Input/output control load | IOCL | | 1000 | |
| Input/output sense | IOS | | 0100 | |
| Input/output control sense | IOCS | | 1100 | |
| Control processor load function | MPLF | | 1010 | |
| Control processor sense | MPS | | 0110 | |
| Compare immediate | CI | C | | |
| Subtract immediate | SI | D | | |
| Add immediate | AI | D | Assembler mnemonic only | |

| Instruction | Mnemonic | Operation Code | Function or Instruction Definition |
|---|---|---|---|
| Storage direct | | E | |
| Load register | L | | Bit 4 = 0<br>Bit 8 = 0 |
| Store register | ST | | Bit 4 = 1<br>Bit 8 = 0 |
| Move local storage register | MVR | E | Bit 8 = 1 |
| Hexadecimal branch | | F | |
| Branch numeric | HBN | | Bit 15 = 1 |
| Branch zone | HBZ | | Bit 15 = 0 |
| Hexadecimal move | | F | |
| Shift right logical | SRL | | Bits 9, 10 = 00 |
| Shift right logical double | SRLD | | Bits 9, 10 = 01 |
| Move zone to zone | MZZ | | Bits 9, 10 = 10 |
| Move zone to numeric | MZN | | Bits 9, 10 = 11 |

# INSTRUCTION EXECUTION

## Signals, Gating Lines, and Logical Functions for Timing Charts

*Local Storage Registers (High and Low)*

- Selected by the '+LSR address bit 0-5' lines
- See FSL page PC230
- Active and can be probed at the T-time(s) when an LSR is selected for reading or writing
- Loaded by the '-write LSR high' or '-write LSR low' lines

*Storage Gates (High and Low)*

- Selected by the '+stg gt lo/hi bit 0-1' lines
- See FSL page PC230
- Active and can be probed at the T-time(s) when the storage gates are ready and receiving input from the system
- Decoded as follows:

| Storage Gate High | | +Storage Gate Hi Bit 0 Fixed A1H2G03<br>+Storage Gate Hi Select Bit 1 A1H2G08 | |
|---|---|---|---|
| **Bit 0** | **Bit 1** | **Register Gated Through** | |
| 0 | 0 | LSR High 0-7,P (G1) | SBO High 0-7,P |
| 0 | 1 | SDR High 0-7,P (G2) | |
| 1 | 0 | SBI Bits 8-15,P (G3) | |
| 1 | 1 | Bits 0-3 X-Reg Hi (G4)<br>4-7 SDR<br>P Storage Gate Hi Generate P Bit | |

| Storage Gate Low | | +Storage Gate Lo Bit 0 A1H2D06<br>+Storage Gate Lo Bit 1 A1H2D11 | |
|---|---|---|---|
| **Bit 0** | **Bit 1** | **Register Gated Through** | |
| 0 | 0 | LSR Low 8-15,P (G1) | SBO Low 8-15,P |
| 0 | 1 | SDR Low 8-15,P (G2) | |
| 1 | 0 | SBI Bits 8-15,P (G3) | |
| 1 | 1 | Storage Gate Hi 0-7,P (G4) | |

*Micro-Operation Register and Storage Data Registers (High and Low)*

- Clocked by the '+CSY trg new' line
- See FSL page PC146

*X-Registers (High and Low)*

- Clocked by the '+clock SAR and X reg' line
- See FSL page PC210

*Y-Registers (High and Low)*

- Clocked by the '+T3 and phase A' line
- See FSL page PC210

*Status 1 Gate*

- Gated out by selecting the '+status function 0-1' and '+status sel 0-2' lines
- See FSL page PC314
- Decoded as follows:

| Status Gate High | | | |
|---|---|---|---|
| **Card**<br>**0** | **Function**<br>**01** | **Select**<br>**012** | **Lines Gated Through** |
| 0 | 00 | 000 | Display Storage Gate High |
| 0 | 00 | 001 | Spare |
| 0 | 00 | 010 | Display Control Processor Check |
| 0 | 00 | 011 | Display Processor Condition Register (PCR) |
| 0 | 00 | 100 | Default Display Events (if not single cycle) |
| 0 | 00 | 101 | Sense Console Switches 1 and 2 |
| 0 | 00 | 110 | Sense Control Processor Check |
| 0 | 00 | 111 | Sense Processor Condition Register (PCR) |
| 0 | 01 | XXX | I/O Control |
| 0 | 10 | XXX | Clock Processor Condition Register (1-3) |
| 0 | 11 | XXX | Clock Processor Condition Register (1-7) |
| 1 | 1X | XXX | Display Storage Gate High |
| 1 | 0X | X01 | Display Console Switches 1 and 2 |

| | |
|---|---|
| + Status Sel 2 | A1-J2B05 |
| + Status Sel 1 | A1-J2D07 |
| + Status Sel 0 | A1-J2D05 |
| + Status Function 1 | A1-J2D04 |
| + Status Function 0 | A1-J2B03 |

*Status 2 Gate*

- Gated out by selecting the '+status function 0-1' and '+status sel 0-2' lines
- See FSL pages PC402 and PC404
- Decoded as follows:

**Status Gate Low**

| Card<br>0 | Function<br>01 | Select<br>012 | Lines Gated Through |
|---|---|---|---|
| 1 | 00 | X00 | Sense Console Status |
| 1 | 00 | X01 | Sense/Display Console Switches 3 and 4 |
| 1 | 00 | X10 | Sense Clock Low |
| 1 | 00 | 011 | Sense Clock High |
| 1 | 00 | 111 | Sense Interrupt Level Backup Byte |
| 1 | 01 | XXX | I/O Load |
| 1 | 1X | XXX | Gate Switches 3 and 4 (bits 4-7) PC422 |

| | |
|---|---|
| + Status Sel 2 | A1-K2M06 |
| + Status Sel 1 | A1-K2M05 |
| + Status Sel 0 | A1-K2P06 |
| + Status Function 1 | A1-K2P04 |
| + Status Function 0 | A1-K2G12 |

*Arithmetic and Logic Unit*

- Gated out by selecting the '+ALU func bit 0-3' lines
- See FSL page PC260
- Decoded as follows:

**Select ALU Mode**

| Bits<br>0-3 | Function Gated Through |
|---|---|
| 0000 | Not Used—Force ALU Hi/Lo, Not Carry |
| 0001 | X OE Y |
| 0010 | Y |
| 0011 | X OR Y |
| 0100 | Not Used |
| 0101 | X and Not Y |
| 0110 | X and Y |
| 0111 | X or Not Y |
| 1000 | X-1 +Carry |
| 1001 | X+Y +Carry |
| 1010 | X–Y 16/8 |
| 1011 | X+Y 16-X or 8-Y |
| 1100 | X–Y 16 or 8 |
| 1101 | X+Y 16/8 |
| 1110 | X-Y-1 +Carry |
| 1111 | X+Carry |

| | |
|---|---|
| +ALU Func Bit 3 | A1H2B09 |
| +ALU Func Bit 2 | A1H2D09 |
| +ALU Func Bit 1 | A1H2D10 |
| +ALU Func Bit 0 | A1H2B03 |
| +Carry In | A1H2S07 |

*Arithmetic and Logic Unit Gates (High and Low)*

- Gated out by selecting the '+ALU gate hi/lo sel 0-2' lines
- See FSL page PC250
- Decoded as follows:

**ALU Gate Low**

| Bits<br>0-2 | Gate | Function Gated Through |
|---|---|---|
| 000 | G0 | ALU Lo 8-15, Predict P Lo |
| 001 | G1 | SBO Lo 8-15, SBO Lo P1 |
| 010 | G2 | ALU Hi 7, ALU Lo 8-14, ALU Lo P Gen |
| 011 | G3 | ALU Lo 8-14, ALU Lo P Gen |
| 100 | G0 | ALU Lo 8-15, Predict P Lo |
| 101 | G1 | SBO Lo 8-15, SBO Lo P1 |
| 110 | G6 | Gate Lo 8-11 from Y Lo 8-11/Gate Lo 12-15 from ALU Lo 8-11/ALU Lo P (ZZ) |
| 111 | G7 | Y Reg 8-11, ALU 12-15, ALU Lo P (ZN) |

**ALU Gate High**

| Bits<br>0-2 | Gate | Function Gated Through |
|---|---|---|
| 000 | G0 | ALU Hi 0-7, Predict P Hi |
| 001 | G1 | SBO Hi 0-7, SBO Hi P |
| 010 | G2 | ALU Hi 0-6, ALU Hi P Gen |
| 011 | G3 | ALU Gate Lo 8-15, P |
| 100 | G3 | ALU Gate Lo 8-15, P |
| 101 | G3 | ALU Gate Lo 8-15, P |
| 110 | G3 | ALU Gate Lo 8-15, P |
| 111 | G3 | ALU Gate Lo 8-15, P |

| | |
|---|---|
| +ALU Gate Hi/Lo Sel 2 | A1H2G07 |
| +ALU Gate Hi/Lo Sel 1 | A1H2G06 |
| +ALU Gate Hi/Lo Sel 0 | A1H2G04 |

*Note:* The storage gates, the ALU, and the ALU gates have default data paths when no-bit select values are used. During default operations, the gating times are a function of the data present at the circuit input.

| Unit | Default Selection |
|---|---|
| Storage gate high | LSR high |
| Storage gate low | LSR low |
| ALU gate high | ALU high |
| ALU gate low | ALU low |
| ALU function | X-register plus 1 |

In the following instruction descriptions, lines in the timing charts that cannot be probed are included so that a better understanding of the data flow and the circuit timings can be maintained. These lines are noted with a superscript number.

## Branch (B)

| 0 0 0 0 | Branch Address |
|---|---|
| 0 3 | 4 15 |

This instruction is used for an unconditional branch operation. It permits branching to any one of the 4,096 word addresses in one control storage segment. There are four 4K-word segments in control storage:

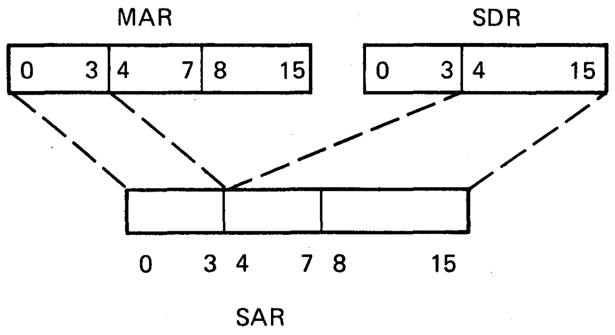
Segment 0–hexadecimal addresses 0000 through 0FFF

Segment 1–hexadecimal addresses 1000 through 1FFF

Segment 2–hexadecimal addresses 2000 through 2FFF

Segment 3–hexadecimal addresses 3000 through 3FFF

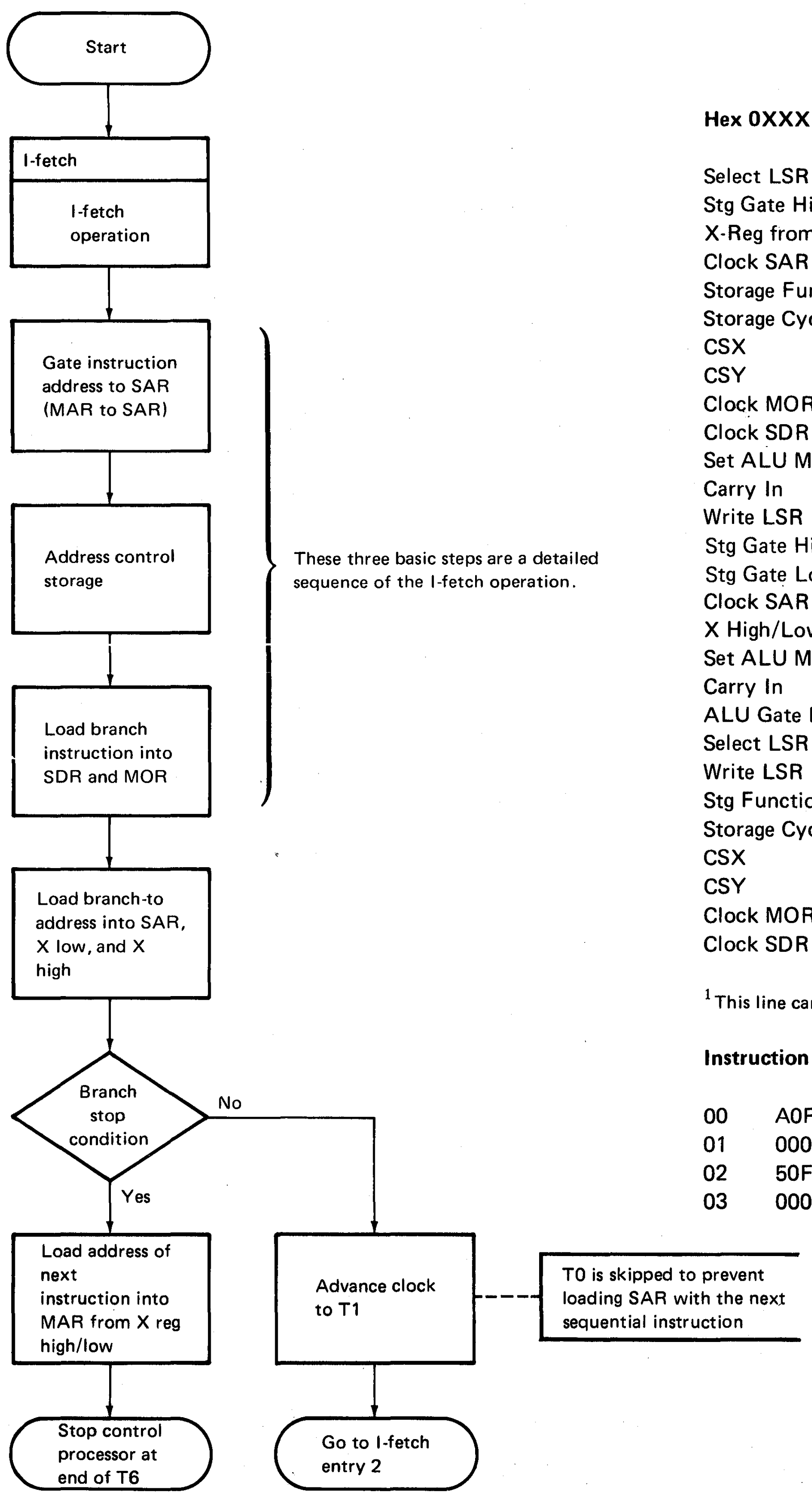
*Branch Address (Bits 4-15):* This is a 12-bit branch address. These 12 bits and X-high bits 0-3 replace the comparable 16 bits in the storage address register (SAR), and the branch address becomes the address of the next sequential instruction. The microaddress register (MAR) is then updated during time T2 of the next cycle.

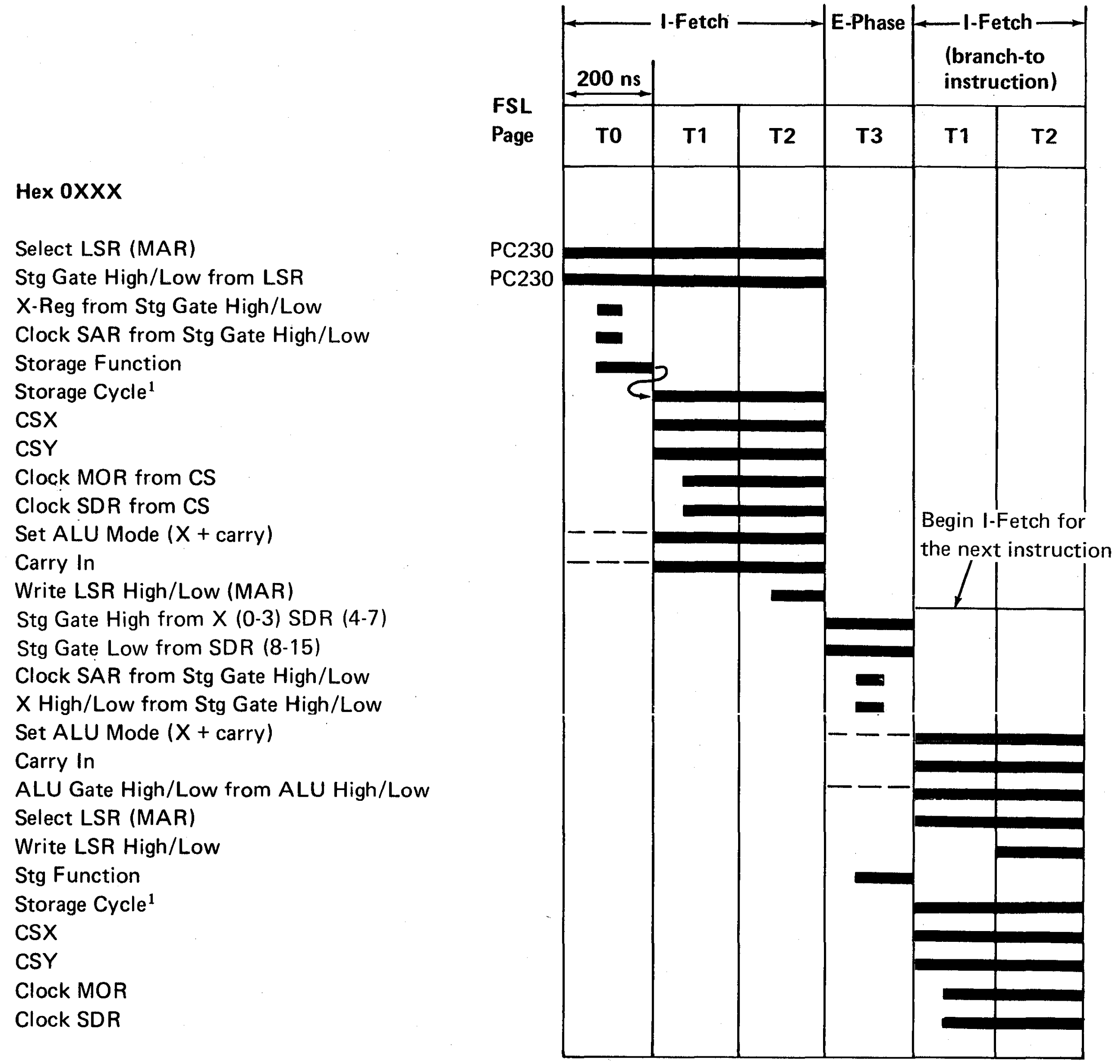


*Condition Code*

No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Gate instruction address to SAR (MAR to SAR)
Address control storage
Load branch instruction into SDR and MOR
These three basic steps are a detailed sequence of the I-fetch operation.
Load branch-to address into SAR, X low, and X high
Branch stop condition
No
Yes
Load address of next instruction into MAR from X reg high/low
Advance clock to T1
T0 is skipped to prevent loading SAR with the next sequential instruction
Stop control processor at end of T6
Go to I-fetch entry 2



[1] This line cannot be probed.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A0FF | LI |
| 01 | 0002 | B * |
| 02 | 50FF | TM |
| 03 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Branch (Stop Condition) (B)

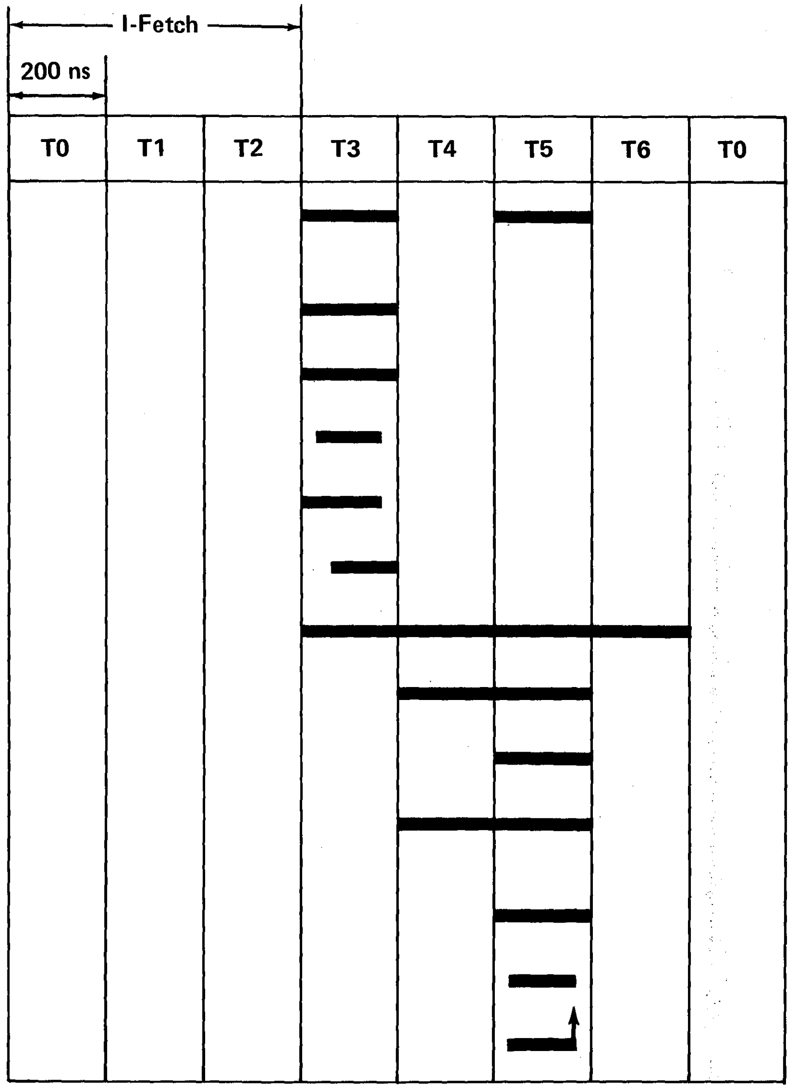
Clock times T4, T5, and T6 can be taken if the control processor is executing a branch instruction and the 'run' latch is reset (branch stop condition) by one of the following:

- A control storage address compare with the Add Comp switch on the CE panel set to the Stop position
- Instruction step mode selected by setting the Mode Selector switch to any Insn Step position (not process condition)
- Processor check stop condition as a result of a processor check

Setting the Mode Selector switch to the Insn Step/Dply LSR position permits single stepping through a branch instruction. Any attempt to single step through a branch that is located in the last valid address of control storage causes a not valid control address check.

*Timing of Control Processor Functions for Branch (Stop Condition)*

| Timing of CP Functions | FSL Page |
|---|---|
| Select LSR (MAR) | PC230 |
| Select Storage Gate High [from X high (0-3)/ from SDR high (4-7)] | PC230 |
| Select Storage Gate Low (from SDR low) | PC230 |
| Clock Low and X High (SAR, don't care) | PC210 |
| Clock Storage Gate Check | PC230 |
| Select Storage Gate Check | PC230 |
| Control Storage Access | PC130 |
| Storage Cycle | PC012 |
| Clock SDR | PC220 |
| ALU Function (pass) | PC260 |
| Select ALU Gate High/Low (from ALU high/low) | PC250 |
| Write LSR High/Low | PC160 |
| Clock ALU Gate Check | PC160 |



## Branch and Link (BAL)

| 0 0 0 1 | Branch Address |
|---|---|
| 0 3 | 4 15 |

This instruction is used for an unconditional branch-and-link operation. It permits branching to any address inside a 4,096-word address block in a control storage segment. Each segment is 4K words long, and there are four 4K-word segments in control storage of 16K words:

Segment 0–hexadecimal addresses 0000 through 0FFF

Segment 1–hexadecimal addresses 1000 through 1FFF

Segment 2–hexadecimal addresses 2000 through 2FFF

Segment 3–hexadecimal addresses 3000 through 3FFF

*Branch Address (Bits 4-15):* This is a 12-bit branch address that replaces the comparable 12 bits in the microaddress register (MAR).

When this instruction is executed, the address in the microaddress register (of the next sequential instruction) is kept in the microaddress backup register (MAB). The address in the microaddress backup register is the link address. The 12-bit branch address in the branch-and-link instruction replaces the address in the microaddress register. The address placed in the microaddress register is the next instruction that is to be executed.

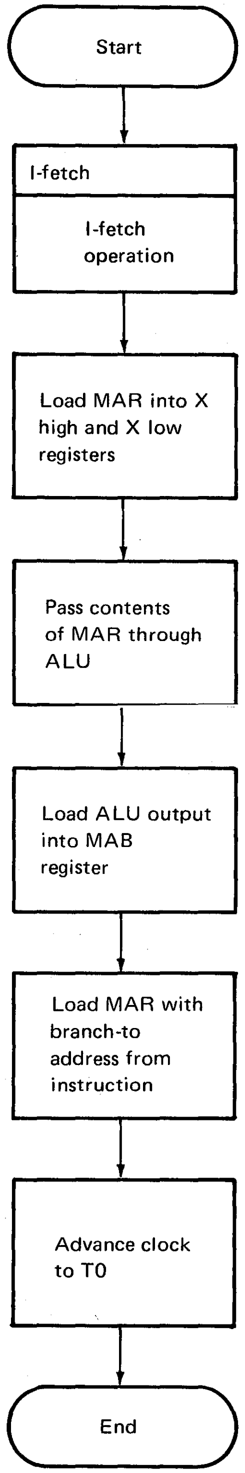
A return instruction is used to return to the next sequential instruction following the branch-and-link instruction. The return instruction causes the address kept in the microaddress backup register to be placed into the microaddress register.

The microaddress register now contains the instruction following the branch-and-link instruction.

*Condition Code*

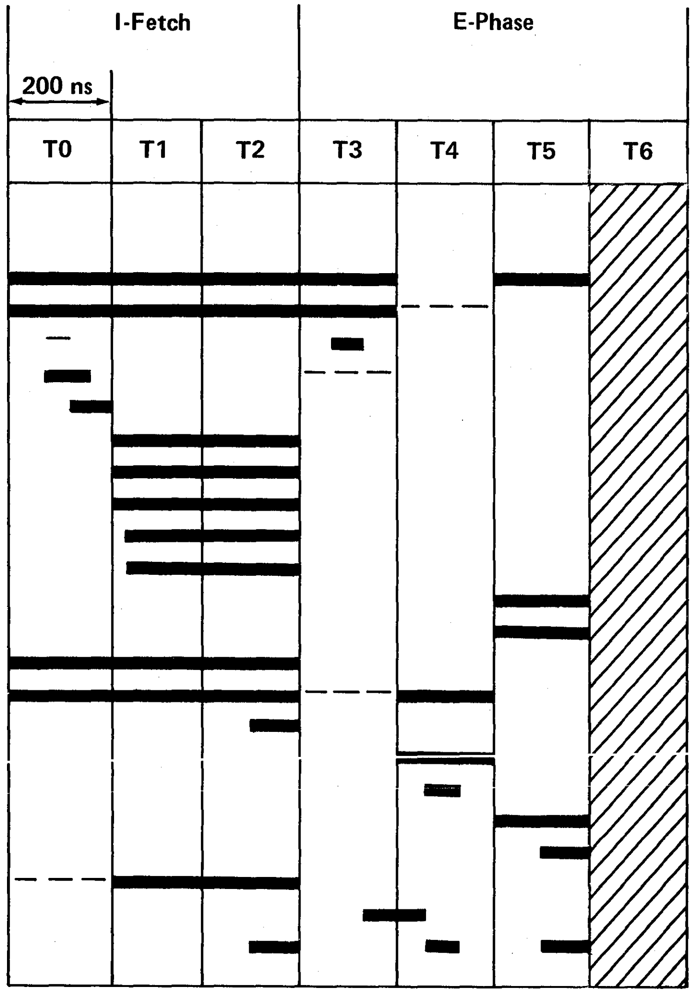
No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Load MAR into X high and X low registers
Pass contents of MAR through ALU
Load ALU output into MAB register
Load MAR with branch-to address from instruction
Advance clock to T0
End

**Hex 1XXX**

| | FSL Page |
|---|---|
| Select LSR (MAR) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Clock SAR from Stg Gate High/Low | |
| Storage Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| Clock MOR | PC146 |
| Clock SDR | PC220 |
| Stg Gate High from X (0-3) SDR (4-7) | PC230 |
| Stg Gate Low from SDR (8-15) | PC230 |
| Set ALU Mode (X + carry) | PC260 |
| ALU Gate High/Low from ALU High/Low | PC250 |
| Write LSR High/Low (MAR) | |
| Select LSR (MAB) | PC230 |
| Write LSR High/Low (MAB) | PC160 |
| ALU Gate High/Low from Stg Gate High/Low | PC250 |
| Write LSR High/Low (MAR) | PC160 |
| Carry In | |
| Clock Stg Gate Check Gated | |
| Clock ALU Gate Check Trigger | |



[1] This line cannot be probed.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A0FF | LI |
| 01 | 50FF | TM |
| 02 | 1000 | BAL * |

**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

This page intentionally left blank.

## Jump on Condition (JC)

| | | |
|---|---|---|
| 0 0 1 0 | Condition | Page Address |
| 0 3 | 4 7 | 8 15 |

This instruction permits branching inside a page boundary (256-word limit of hex 00 through hex FF) (specified by bits 8-15) if the condition specified by bits 4-7 is met. If the condition is met, the 8-bit page address replaces the comparable bits in the microaddress register (MAR) and the storage address register (SAR) to form the address of the next instruction to be executed.

*Condition Tested (Bits 4-7):* Indicates the function to be tested as follows:

| Bits 4-7 | Mnemonic | Test Condition |
|---|---|---|
| 0000 | JCY | Carry |
| 0001 | JH | High (condition code bit 5) |
| 0010 | JL | Low (condition code bit 6) |
| 0011 | JE | Equal (condition code bit 7) |
| 0100 | JP | Positive (condition code bit 1) |
| 0100 | JO | All ones (condition code bit 1) |
| 0101 | JN | Negative (condition code bit 2) |
| 0101 | JM | Mixed (condition code bit 2) |
| 0110 | JZ | Zero (condition code bit 3) |
| 0111 | JFLG | Flag |
| 1000 | JSR | Service request |
| 1001 | JNH | Not high |
| 1010 | JNL | Not low |
| 1011 | JNE | Not equal |
| 1100 | JNP | Not positive |
| 1101 | JNN | Not negative |
| 1110 | JNZ | Not zero |
| 1111 | RETRN | Return |

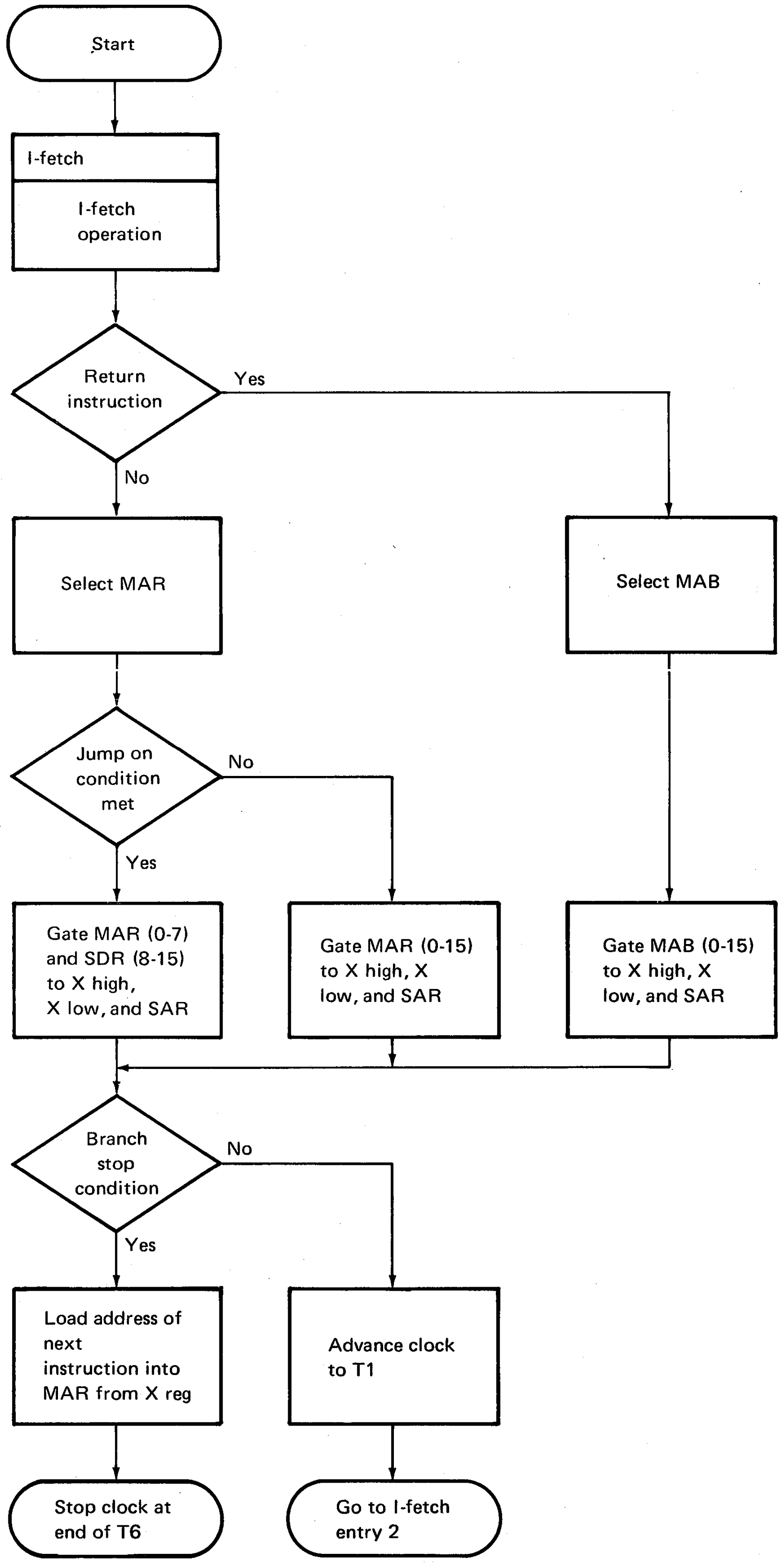
*Page Address (Bits 8-15):* Permits branching inside a page boundary (256-word limit of hex 00 through hex FF) in control storage only. The page address replaces the 8 low-order bits in the microaddress register when the tested condition is met.

*Note:* For the return condition (bits 4-7 equal 1111), the page address is not used. In this case, the microaddress backup register is selected for the address of the next instruction to be executed.

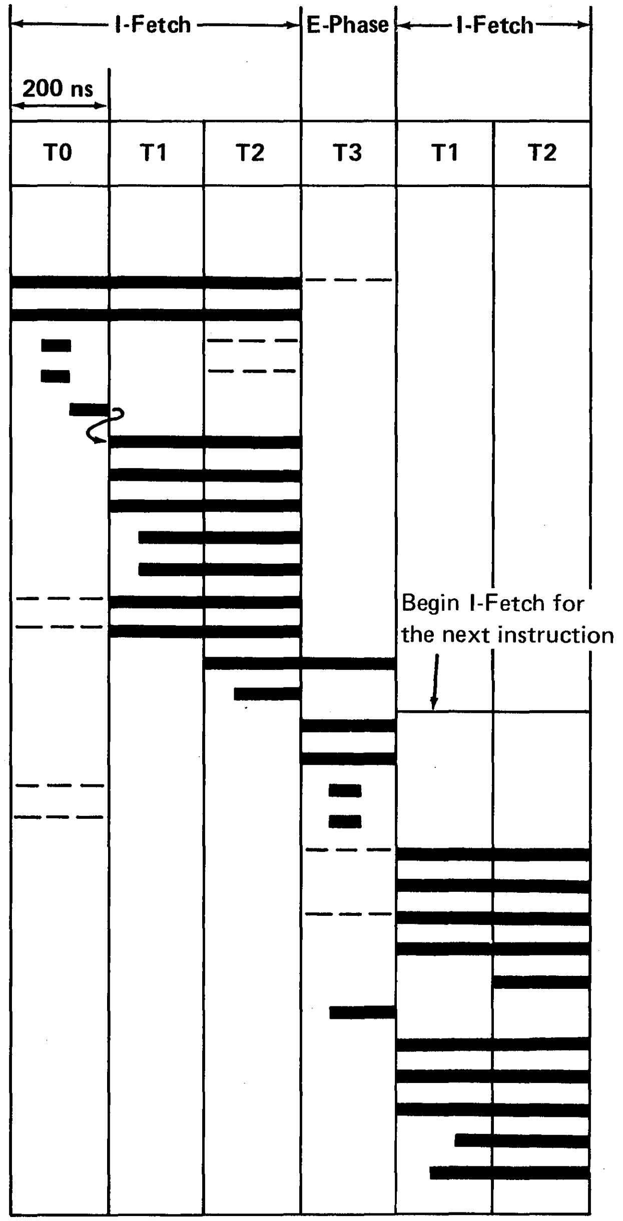
*Condition Code*

No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Return instruction
Yes
No
Select MAR
Select MAB
Jump on condition met
No
Yes
Gate MAR (0-7) and SDR (8-15) to X high, X low, and SAR
Gate MAR (0-15) to X high, X low, and SAR
Gate MAB (0-15) to X high, X low, and SAR
Branch stop condition
No
Yes
Load address of next instruction into MAR from X reg
Advance clock to T1
Stop clock at end of T6
Go to I-fetch entry 2

| Hex 2XXX | FSL Page |
|---|---|
| Select LSR (MAR) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Clock SAR from Stg Gate High/Low | |
| Storage Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| Clock MOR from CS | PC146 |
| Clock SDR from CS | PC220 |
| Set ALU Mode (X + carry) | PC260 |
| Carry In | |
| CPU Branch Condition Met | PC304 |
| Write LSR High/Low (MAR) | PC160 |
| Stg Gate High from X (0-3) SDR (4-7) | PC230 |
| Stg Gate Low from SDR (8-15) | PC230 |
| Clock SAR from Stg Gate High/Low | |
| X High/Low from Stg Gate High/Low | |
| Set ALU Mode (X + carry) | PC260 |
| Carry In | |
| ALU Gate High/Low from ALU High/Low | PC250 |
| Select LSR (MAR) | |
| Write LSR High/Low | |
| Stg Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| Clock MOR | PC146 |
| Clock SDR | PC220 |



[1] This line cannot be probed.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A0FF | LI |
| 01 | 50FF | TM |
| 02 | 2304* | JE |
| 03 | BEA3 | Check Halt |
| 04 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = – address compare' looking at the instruction referenced with an asterisk (*).

## Jump on Condition (Stop Condition) (JC)

Clock times T4, T5, and T6 can be taken if the control processor is executing a jump-on condition and the 'run' latch is reset by one of the following:

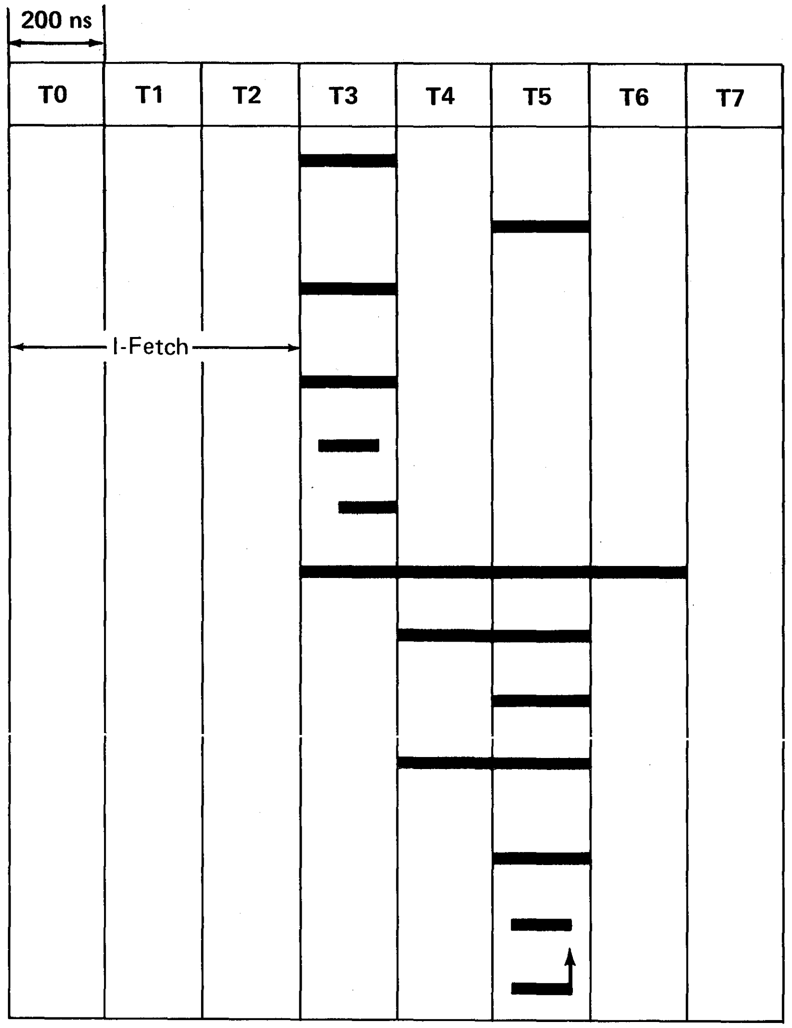
- A control storage address compare with the Add Comp switch on the CE panel set to the Stop position
- Instruction step mode selected by setting the Mode Selector switch to any Insn Step position (not process condition)
- Processor check stop condition as a result of a processor check

Setting the Mode Selector switch to the Insn Step/Dply LSR position permits single stepping through a jump-on-condition instruction. Any attempt to single step through a jump-on condition that is located in the last valid address of control storage, control storage segment, or 256-byte block (hex 00 through hex FF), causes a not valid control address check.

The function of the condition tested (bits 4-7) is the same as for the jump-on-condition instruction.

*Timing*

| Timing of CP Functions | FSL Page |
|---|---|
| Select LSR (MAR: no return; | PC240 |
| MAB: return) | PC240 |
| Select Storage Gate High (from LSR high) | PC230 |
| Select Storage Gate Low (from LSR low: not met; from SDR low: low and X high met) | PC230 |
| Clock X (SAR, don't care) | PC210 |
| Clock Storage Gate Check | PC230 |
| Control Storage Access | PC010 |
| Storage Cycle | PC012 |
| Clock SDR | PC002 |
| ALU Function (pass) | PC260 |
| Select ALU Gate High/Low (from ALU high/low) | PC250 |
| Write LSR High/Low | PC160 |
| Clock ALU Gate Check | PC160 |



This page intentionally left blank.

## Logical/Arithmetic 1 (XR, ZAR, OR, NCR, NR, OCR, DEC, ACYR, SR, AR, SCYR, INC)

| 0 1 1 0 | H1 | Reg 1 | Function | H2 | Reg 2 |
|---|---|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 11 | 12 | 13 15 |

This instruction performs logical and arithmetic type functions that are performed in the arithmetic and logic unit (ALU). The logical/arithmetic 1 instruction is for 1-byte operations only.

*H1 (Bit 4):* Indicates which byte of the selected local storage register (register 1) is to be used in the current function:

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected register is operand 1 of the function and is changed at the end of the function.

*Function (Bits 8-11):* Determines the basic logical or arithmetic function to be performed.

*H2 (Bit 12):* Indicates which byte of the selected local storage register (register 2) is to be used in the current function:

H2 = 0: Low-order byte

H2 = 1: High-order byte

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected register is operand 2 of the function. The selected register is not changed by the operation being performed.

**Condition Code for Logical Operations**

On logical operations, two actions are performed:

- The logical operation (OR, AND, exclusive OR, and so on) is performed.
- Register 1 contents are combined, using an OR operation, with the ones complement of register 2 contents. This is shown as (register 1 or not register 2).

The condition code is set as follows to show the results of *both* operations, except when the result of the logical operation is zeros (bit 3 of the processor condition register):

- Positive (bit 1 of the processor condition register)–Set if the result of the logical operation is not equal to zero, and (register 1 or not register 2) is equal to all ones. Reset if the result of the logical operation is equal to all zeros, or (register 1 or not register 2) is not equal to all ones.
- Negative (bit 2 of the processor condition register)–Set if the result of the logical operation is not equal to all zeros, and (register 1 or not register 2) is not equal to all ones. Reset if the result of the logical operation is equal to all zeros, or (register 1 or not register 2) is equal to all ones.
- Zero (bit 3 of the processor condition register)–Set if the result of the logical operation is equal to all zeros. Reset if the result of the logical operation is not equal to all zeros.

**Condition Code for Arithmetic Operations**

*Note:* Borrow and carry in the processor condition register have the following meanings:

| | | |
|---|---|---|
| Borrow | = | No carry |
| Carry | = | No borrow |

- Positive (bit 1 of the processor condition register)–Set if the result of the arithmetic operation is not equal to zero and has a carry. Reset if the result is zero or there is no carry.
- Negative (bit 2 of the processor condition register)–Set if the result of the arithmetic operation is not equal to zero and has no carry. Reset if the result is zero or there is a carry.
- Zero (bit 3 of the processor condition register)–Set if the result of the arithmetic operation is equal to zero. Reset if the result is not equal to zero.
- Carry (bit 4 of the processor condition register)–Set if the arithmetic operation results in a carry. Reset by the I/O immediate instruction (reset carry-set equal function), by system reset, or if the operation results in no carry.
- High (bit 5 of the processor condition register)–Same as positive (bit 1).
- Low (bit 6 of the processor condition register)–Same as negative (bit 2).
- Equal (bit 7 of the processor condition register)–Reset if the result of the operation is not equal to zero. Set only by the I/O immediate instruction (reset carry-set equal function), or by system reset.

**Logical/Arithmetic Functions**

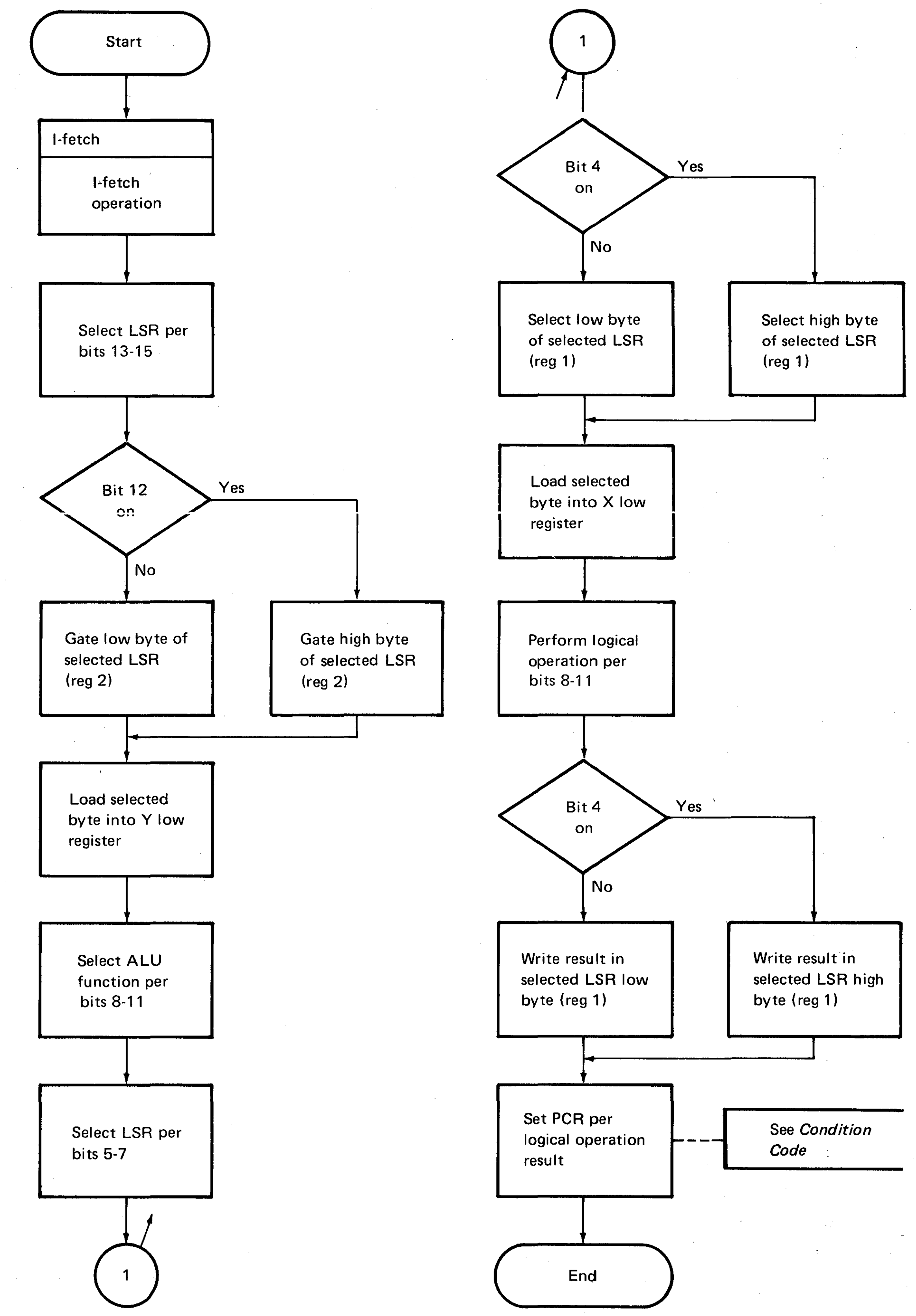
| Bits<br>8 9 10 11 | Mnemonic | Function | Description | Example | |
|---|---|---|---|---|---|
| 0 0 0 0 | | Not used | | | |
| 0 0 0 1 | XR | R1 (XOR) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an exclusive OR function and the result is placed in the R1 location. | R1<br>R2<br>R1 | 10111100<br>00110101<br>10001001 |
| 0 0 1 0 | ZAR | R2 + 0 → R1 | The contents of R2 are placed in the R1 location. | R2<br>+0<br>R1 | 10111100<br>00000000<br>10111100 |
| 0 0 1 1 | OR | R1 (OR) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an OR function and the result is placed in the R1 location. | R1<br>R2<br>R1 | 10111100<br>00110101<br>10111101 |
| 0 1 0 0 | | Not used | | | |
| 0 1 0 1 | NCR | R1 (AND) $\overline{R2}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU complements the Y register (R2), performs an AND function on the X and Y registers, and the result is placed in the R1 location. | R1<br>R2<br>$\overline{R2}$<br>R1 | 10111100<br>00110101<br>11001010<br>10001000 |
| 0 1 1 0 | NR | R1 (AND) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an AND function on the X and Y registers, and the result is placed in the R1 location. | R1<br>R2<br>R1 | 10111100<br>00110101<br>00110100 |
| 0 1 1 1 | OCR | R1 (OR) $\overline{R2}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU complements the Y register contents (R2), performs an OR function on the X and Y registers, and the result is placed in the R1 location. | R1<br>R2<br>$\overline{R2}$<br>R1 | 10111100<br>00110101<br>11001010<br>11111110 |
| 1 0 0 0 | DEC | R1 - 1 → R1 | The contents of R1 are placed in the X register. This data is gated in the ALU. The ALU performs an X minus 1 function and the result is placed in the R1 location. | R1<br>-1<br>R1 | 10111100<br>00000001<br>10111011 |
| 1 0 0 1 | ACYR | R1 + R2 + C → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The contents of the X and Y registers are added together and then added to the result of the carry trigger from a previous operation. The result is placed in the R1 location. | R1<br>R2<br><br>+C<br>R1 | 10111100<br>00110101<br>11110001<br>00000001<br>11110010 |
| 1 0 1 0 | | Not used | | | |
| 1 0 1 1[1] | AR | R1 + R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The contents of the X and Y registers are added together in the ALU and the result is placed in the R1 location. | R1<br>R2<br>R1 | 10111100<br>00110101<br>11110001 |
| 1 1 0 0 | SR | R1 - R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The Y register contents are subtracted from the X register contents, and the result is placed in the R1 location. | R1<br>R2<br>R1 | 10111100<br>00110101<br>10000111 |
| 1 1 0 1 | | Not used | | | |
| 1 1 1 0[2] | SCYR | R1 - R2 - $\overline{C}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The Y register contents are subtracted from the X register contents. The carry trigger from a previous operation is complemented and then subtracted from the result. The final result is placed in the R1 location. | R1<br>R2<br><br>C<br>$-\overline{C}$<br>R1 | 10111100<br>00110101<br>10000111<br>1 ←<br>0<br>10000111 |
| 1 1 1 1 | INC | R1 + 1 → R1 | The contents of R1 are placed in the X register. The 'carry in' line is activated by the instruction and 1 is added to the contents of the X register by the ALU. The result is placed in the R1 location. | R1<br>+1<br>R1 | 10111100<br>00000001<br>10111101 |

From a previous operation

[1] By adding a register to itself (R1 + R1 → R1), the shift left logical function can be executed. This function causes the 8 bits to be shifted one position to the left and the low-order bit (bit 7) to be replaced with a zero. Mnemonic = SLL.

[2] $\overline{C}$ is the same as a borrow.

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 13-15
Bit 12 on
Yes
No
Gate low byte of selected LSR (reg 2)
Gate high byte of selected LSR (reg 2)
Load selected byte into Y low register
Select ALU function per bits 8-11
Select LSR per bits 5-7
1
1
Bit 4 on
Yes
No
Select low byte of selected LSR (reg 1)
Select high byte of selected LSR (reg 1)
Load selected byte into X low register
Perform logical operation per bits 8-11
Bit 4 on
Yes
No
Write result in selected LSR low byte (reg 1)
Write result in selected LSR high byte (reg 1)
Set PCR per logical operation result
See *Condition Code*
End

**Hex 6132**

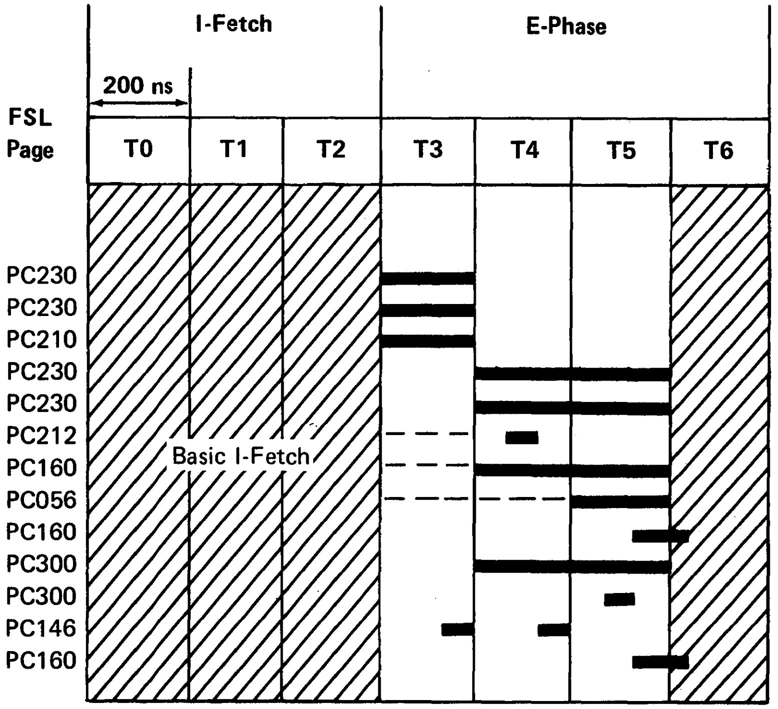
Select LSR (operand 2)
Stg Gate High/Low from LSR
X Low from Stg Gate Low (Y high, don't care)
Select LSR (operand 1)
Stg Gate High/Low from LSR
Y Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X or Y) (see Note 1)
ALU Gate Low from ALU Low (ALU gate high, don't care)
Write LSR Low
Clock PCR (bits 1, 2, 3)
Clock PCR (bits 4, 5, 6, 7)
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

```
00    A0FF    LI
01    6132    LA1 (OR) * (see Note 2)
02    0000    B
```

*Notes:*

1. ALU mode setting will vary with the setting of the function bits (8-11).
2. This instruction uses the low byte of each operand.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

**Hex 693A**

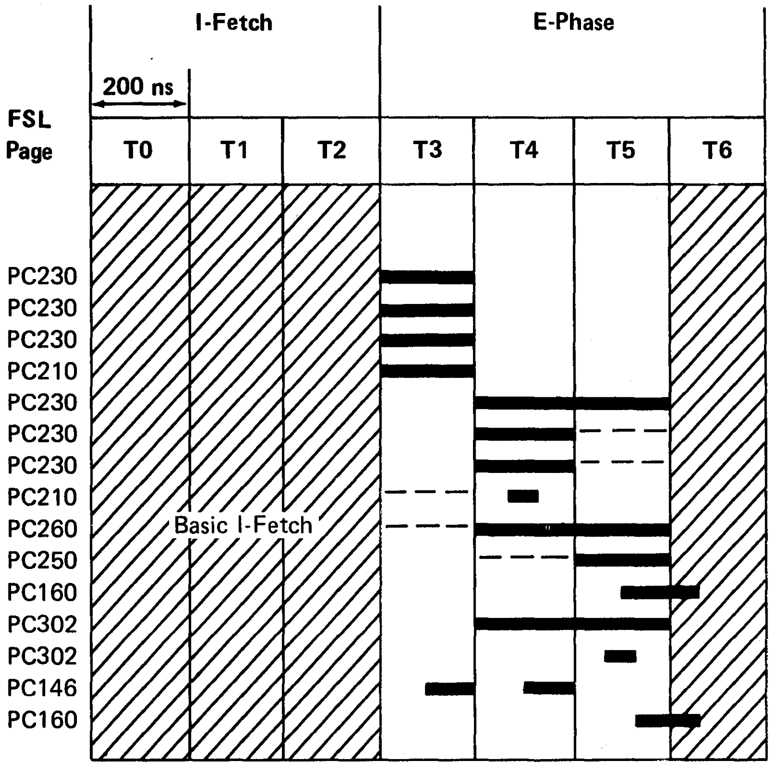
Select LSR (operand 2)
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
Y Low from Stg Gate Low (Y high, don't care)
Select LSR (operand 1)
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X or Y) (see Note 1)
ALU Gate High/Low from ALU High/Low
Write LSR High
Clock PCR (bits 1, 2, 3)
Clock PCR (bits 4, 5, 6, 7)
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

```
00    A0FF    LI
01    693A    LA1 (OR) * (see Note 2)
02    0000    B
```

*Notes:*

1. ALU mode setting will vary with the setting of the function bits (8-11).
2. This instruction uses the high byte of each operand.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Logical/Arithmetic 2 (XR, ZAR, OR, NCR, NR, OCR, DEC, ACYR, SR, AR, SCYR, INC)

| | | | | | |
|---|---|---|---|---|---|
| 0 1 1 1 | | Reg 1 | Function | H2 | Reg 2 |
| 0 3 | 4 | 5 7 | 8 11 | 12 | 13 15 |

This instruction performs logical and arithmetic type functions. The logical/arithmetic 2 instruction always uses both bytes of operand 1 and one or both bytes of operand 2, as determined by the function. Both bytes of operand 2 are used unless the instruction is SR with a function modifier of hexadecimal A, or the instruction is AR with a function modifier of hexadecimal B. In the exception instructions, the selected byte (hi or lo) of operand 2 performs a logical or arithmetic operation on the low-order byte of operand 1.

When the operand 2 high byte is selected, the high byte of data is moved into the low-order data position of the storage gate. Then, Stg Gate Lo is moved to Y Reg Lo and Y Reg Hi is not gated.

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. Both bytes of the selected local storage register represent operand 1. The selected local storage register is changed at the end of the function being performed.

*Function (Bits 8-11):* Determines the basic logical or arithmetic function to be performed.

*H2 (Bit 12):* Indicates which byte of the selected local storage register (register 2) is to be used in the current function:

H2 = 0: Low-order byte

H2 = 1: High-order byte

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected local storage register is operand 2 of the function. The selected local storage register is not changed by the operation being performed.

**Condition Code for Logical Operations**

On logical operations, two actions are performed:

- The logical operation (OR, AND, exclusive OR, and so on) is performed.

- Register 1 contents are combined, using an OR operation, with the ones complement of register 2 contents. This is shown as (register 1 or not register 2).

The condition code is set as follows to show the results of *both* operations, except when the result of the logical operation is zeros (bit 3 of the processor condition register):

- Positive (bit 1 of the processor condition register)–Set if the result of the logical operation is not equal to zero, and (register 1 or not register 2) is equal to all ones. Reset if the result of the logical operation is equal to all zeros, or (register 1 or not register 2) is not equal to all ones.

- Negative (bit 2 of the processor condition register)–Set if the result of the logical operation is not equal to all zeros, and (register 1 or not register 2) is not equal to all ones. Reset if the result of the logical operation is equal to all zeros, or (register 1 or not register 2) is equal to all ones.

- Zero (bit 3 of the processor condition register)–Set if the result of the logical operation is equal to all zeros. Reset if the result of the logical operation is not equal to all zeros.

**Condition Code for Arithmetic Operations**

*Note:* Borrow and carry in the processor condition register have the following meanings:

| | | |
|---|---|---|
| Borrow | = | No carry |
| Carry | = | No borrow |

- Positive (bit 1 of the processor condition register)–Set if the result of the arithmetic operation is not equal to zero and has a carry. Reset if the result is zero or there is no carry.

- Negative (bit 2 of the processor condition register)–Set if the result of the arithmetic operation is not equal to zero and has no carry. Reset if the result is zero or there is a carry.

- Zero (bit 3 of the processor condition register)–Set if the result of the arithmetic operation is equal to zero. Reset if the result is not equal to zero.

- Carry (bit 4 of the processor condition register)–Set if the arithmetic operation results in a carry. Reset by the I/O immediate instruction (reset carry-set equal function), by system reset, or if the operation results in no carry.

- High (bit 5 of the processor condition register)–Same as positive (bit 1).

- Low (bit 6 of the processor condition register)–Same as negative (bit 2).

- Equal (bit 7 of the processor condition register)–Reset if the result of the operation is not equal to zero. Set only by the I/O immediate instruction (reset carry-set equal function) or by system reset.

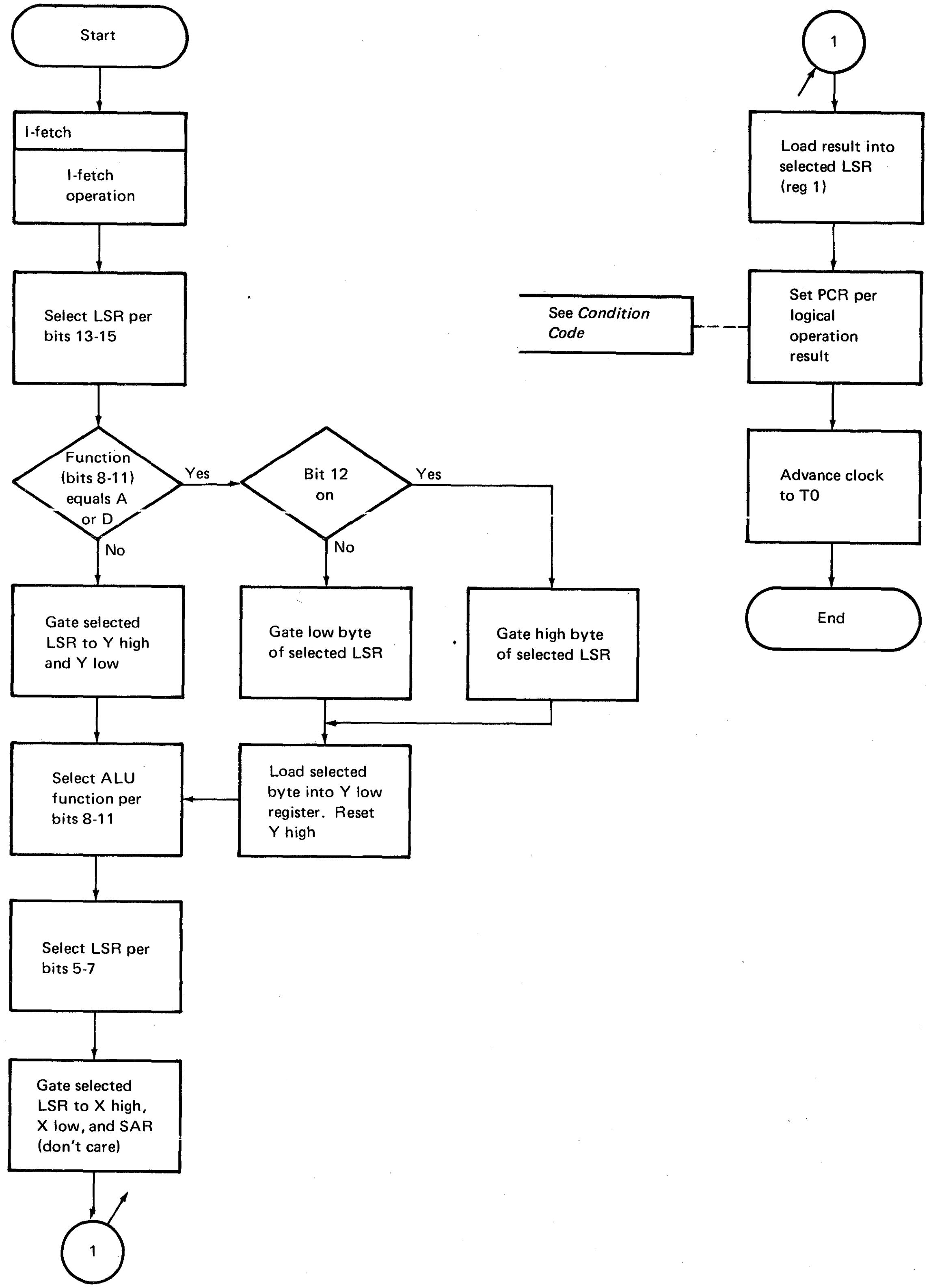
**Logical/Arithmetic Functions**

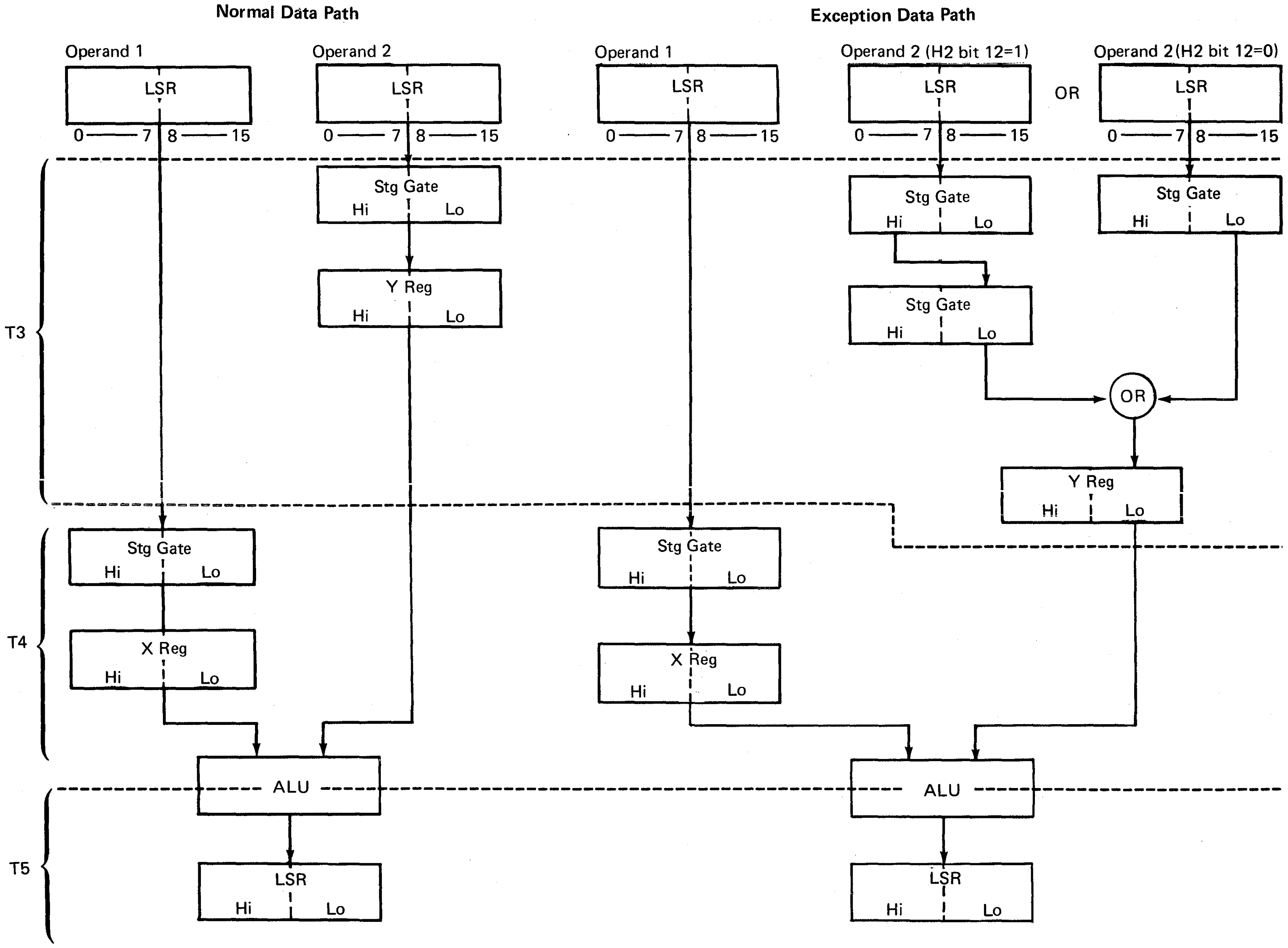
| Bits<br>8 9 10 11 | Mnemonic | Function | Description | Example |
|---|---|---|---|---|
| 0 0 0 0 | | Not used | | |
| 0 0 0 1 | XR | R1 (XOR) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an exclusive OR function and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>R1 1000100101100100 |
| 0 0 1 0 | ZAR | R2 + 0 → R1 | The contents of R2 are placed in the R1 location. | R2 1011110011001101<br>+0 0000000000000000<br>R1 1011110011001101 |
| 0 0 1 1 | OR | R1 (OR) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an OR function and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>R1 1011110111101101 |
| 0 1 0 0 | | Not used | | |
| 0 1 0 1 | NCR | R1 (AND) $\overline{R2}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU complements the Y register contents (R2), performs an AND function on the register contents, and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>$\overline{R2}$ 1100101001010110<br>R1 1000100001000100 |
| 0 1 1 0 | NR | R1 (AND) R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU performs an AND function and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>R1 0011010010001001 |
| 0 1 1 1 | OCR | R1 (OR) $\overline{R2}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The ALU complements the Y register contents (R2), performs an OR function on the register contents, and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>$\overline{R2}$ 1100101001010110<br>R1 1111111011011111 |
| 1 0 0 0 | DEC | R1 - 1 → R1 | The contents of R1 are placed in the X register. This data is gated in the ALU. The ALU performs an X minus 1 function and the result is placed in the R1 location. | R1 1011110011001101<br>-1 0000000000000001<br>R1 1011110011001100 |
| 1 0 0 1 | ACYR | R1 + R2 + C → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The contents of the two registers are added together and added to the result of the carry trigger from a previous operation. The result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>1111001001110110<br>+C 1<br>R1 1111001001110111 |
| 1 0 1 0 | SR | $R1 \underbrace{- R2}_{\text{1 byte}} \rightarrow R1$ | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The Y register contents are subtracted from the X register contents and the result is placed in the R1 location. | R1 1011110011001101<br>R2 10101001<br>R1 1011110000100100 |
| 1 0 1 1[1] | AR | R1 + R2 → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The contents of the two registers are added together in the ALU and the result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>R1 1111001001110110 |
| 1 1 0 0 | SR | R1 - R2 → R1 | Same as (1010) SR. | R1 1011110011001101<br>R2 0011010110101001<br>R1 1000011100100100 |
| 1 1 0 1 | AR | $R1 \underbrace{+ R2}_{\text{1 byte}} \rightarrow R1$ | Same as (1011) AR. | R1 1011110011001101<br>R2 10101001<br>R1 1011110101110110 |
| 1 1 1 0[2] | SCYR | R1 − R2 − $\overline{C}$ → R1 | The contents of R1 are placed in the X register; the contents of R2 are placed in the Y register. The Y register contents are subtracted from the X register contents; the carry trigger from a previous operation is complemented and then subtracted from the result. The final result is placed in the R1 location. | R1 1011110011001101<br>R2 0011010110101001<br>1000011100100100<br>C_ 0 ← From a previous operation<br>$-\overline{C}$ 1<br>R1 1000011100100011 |
| 1 1 1 1 | INC | R1 + 1 → R1 | The contents of R1 are placed in the X register. The 'carry in' line is activated by the instruction, and this is added to the contents of the X register by the ALU. The result is placed in the R1 location. | R1 1011110011001101<br>+1 0000000000000001<br>R1 1011110011001110 |

[1] By adding a register to itself (R1 + R1 → R1), the shift left logical double function can be executed. This function causes the 16 bits to be shifted one position to the left and the low-order bit (bit 15) to be replaced with a zero. Mnemonic = SLLD.

[2] $\overline{C}$ is the same as a borrow.

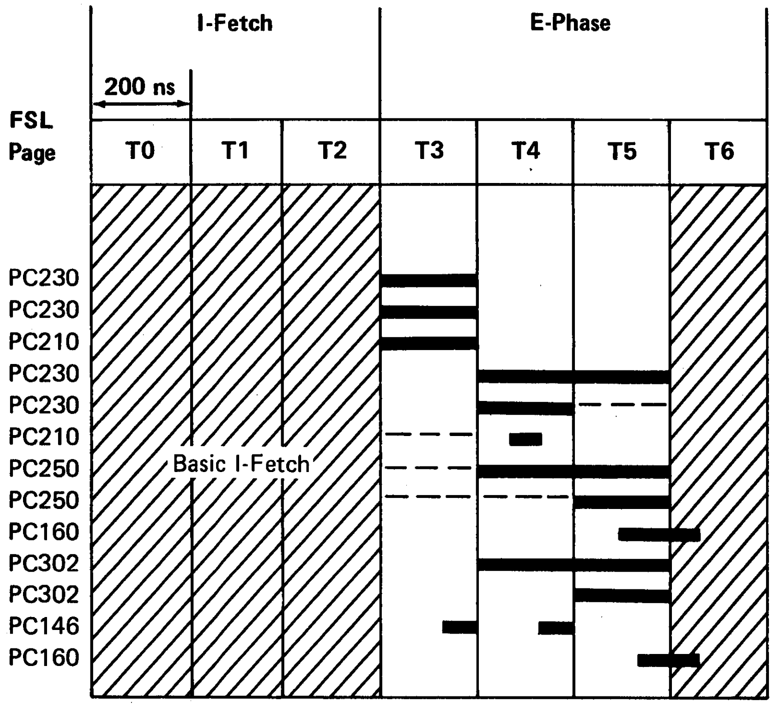
*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 13-15
Function (bits 8-11) equals A or D
Yes
No
Bit 12 on
Yes
No
Gate selected LSR to Y high and Y low
Gate low byte of selected LSR
Gate high byte of selected LSR
Select ALU function per bits 8-11
Load selected byte into Y low register. Reset Y high
Select LSR per bits 5-7
Gate selected LSR to X high, X low, and SAR (don't care)
1
1
Load result into selected LSR (reg 1)
See *Condition Code*
Set PCR per logical operation result
Advance clock to T0
End



**Hex 7132**

Select LSR (operand 2)
Select Stg Gate High/Low from LSR
Y High/Low from Stg Gate High/Low
Select LSR (operand 1)
Stg Gate High/Low from LSR
X High/Low from Stg Gate High/Low
Set ALU Mode (X or Y) (see Note 1)
ALU Gate High/Low from ALU High/Low
Write LSR High/Low
Clock PCR (bits 1, 2, 3)
Clock PCR (bits 4, 5, 6, 7)
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | 50FF | TM |
| 01 | 7132 | LA2 (OR) * (see Note 2) |
| 02 | 0000 | B |

*Notes:*

1. ALU mode setting will vary with the setting of the function bits (8-11).
2. This instruction uses both bytes of both operands.

**Scope Setup**

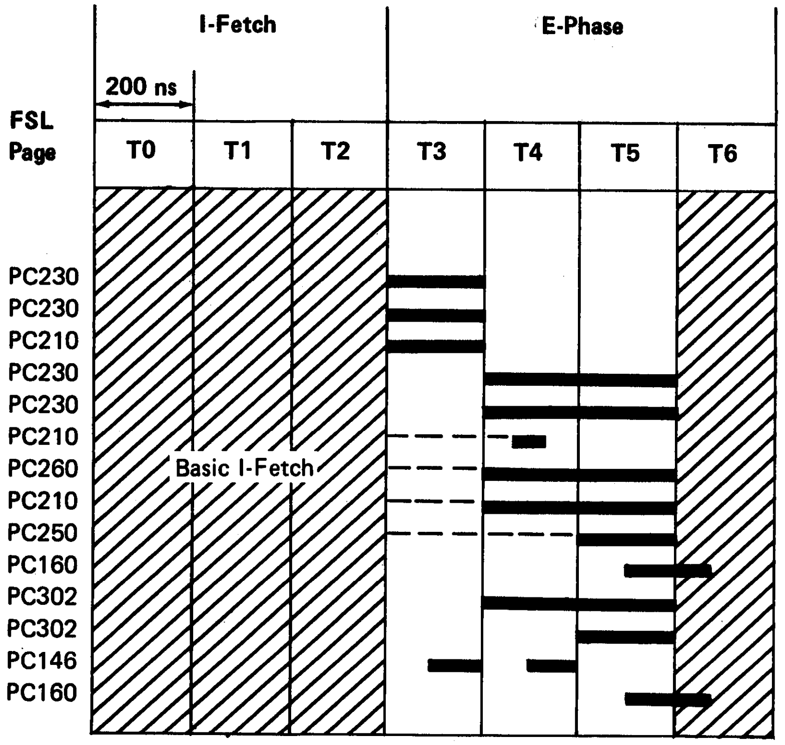
Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Hex 71D2**

Select LSR (operand 2)
Stg Gate High/Low from LSR
Y High/Low from Stg Gate High/Low
Select LSR (operand 1)
Stg Gate High/Low from LSR
X High/Low from Stg Gate High/Low
Set ALU Mode (X+Y) (see Note 1)
Reset Y High
ALU Gate High/Low from ALU High/Low
Write LSR High/Low
Clock PCR (bits 1, 2, 3)
Clock PCR (bits 4, 5, 6, 7)
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | 50FF | TM |
| 01 | 71D2 | LA2 (X+Y) * (see Note 2) |
| 02 | 0000 | B |

*Notes:*

1. ALU mode setting will be either X+Y or X–Y.
2. These are the only two LA2 instructions that use only 1 byte from operand 2.

**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Load Immediate (LI)

| 1 0 1 0 | H1 | Reg 1 | Immediate Byte |
|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 15 |

This instruction takes the data in the immediate byte (bits 8-15) and loads the data directly into the selected register of the local storage register stack. Data can be placed into the high- or low-order byte of the selected register.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used:

H1 = 0: Low-order byte

H1 = 1: High-order byte

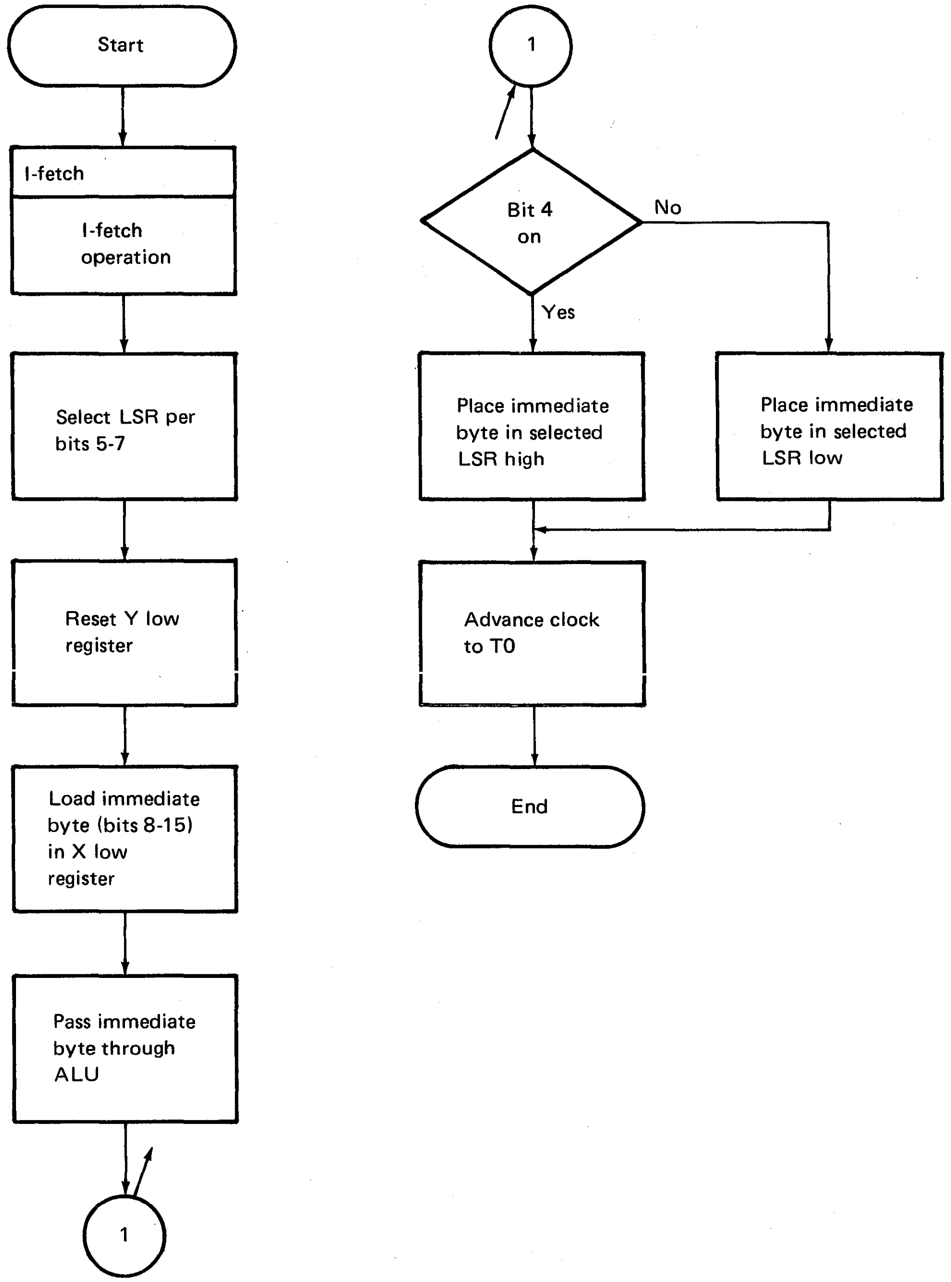
*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level.

*Immediate Byte (Bits 8-15):* The immediate byte of the instruction is loaded into the selected local storage register.

*Condition Code*

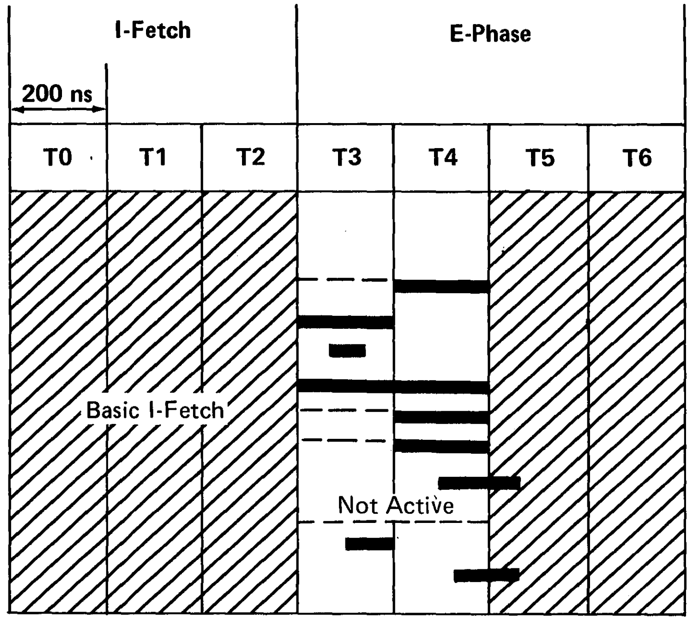
No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 5-7
Reset Y low register
Load immediate byte (bits 8-15) in X low register
Pass immediate byte through ALU
1
1
Bit 4 on
No
Yes
Place immediate byte in selected LSR high
Place immediate byte in selected LSR low
Advance clock to T0
End

**Hex A9XX**

| | FSL Page |
|---|---|
| Select LSR | PC230 |
| Stg Gate High/Low from SDR High/Low | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Set ALU Mode (X + carry) | PC260 |
| ALU Gate Low from ALU Low | PC250 |
| ALU Gate High from ALU Gate Low | PC250 |
| Write LSR High | PC160 |
| Carry In | |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A0FF | LI |
| 01 | A9FF | LI* (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.

**Scope Setup**

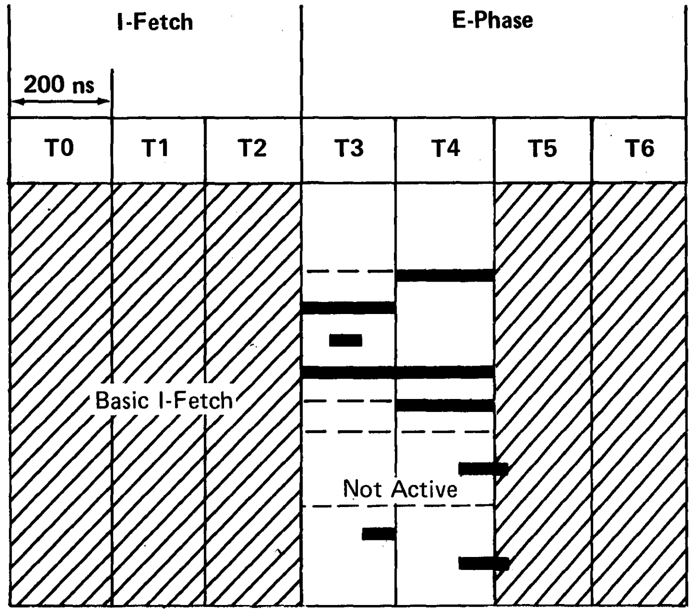
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

**Hex A1XX**

| | FSL Page |
|---|---|
| Select LSR | PC230 |
| Stg Gate High/Low from SDR High/Low | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Set ALU Mode (X + carry) | PC250 |
| ALU Gate Low from ALU Low | PC250 |
| ALU Gate High from ALU Gate Low | PC250 |
| Write LSR Low | PC160 |
| Carry In | |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A0FF | LI |
| 01 | A1FF | LI * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Compare Immediate (CI)

| 1 1 0 0 | H1 | Reg 1 | Immediate Byte |
|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 15 |

This instruction compares the 8 bits of data in the selected local storage register with the comparable 8 bits of data in the immediate byte. The results of the compare are set in the processor condition register. The selected local storage register is not changed by the compare immediate instruction.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used in the compare:

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level.

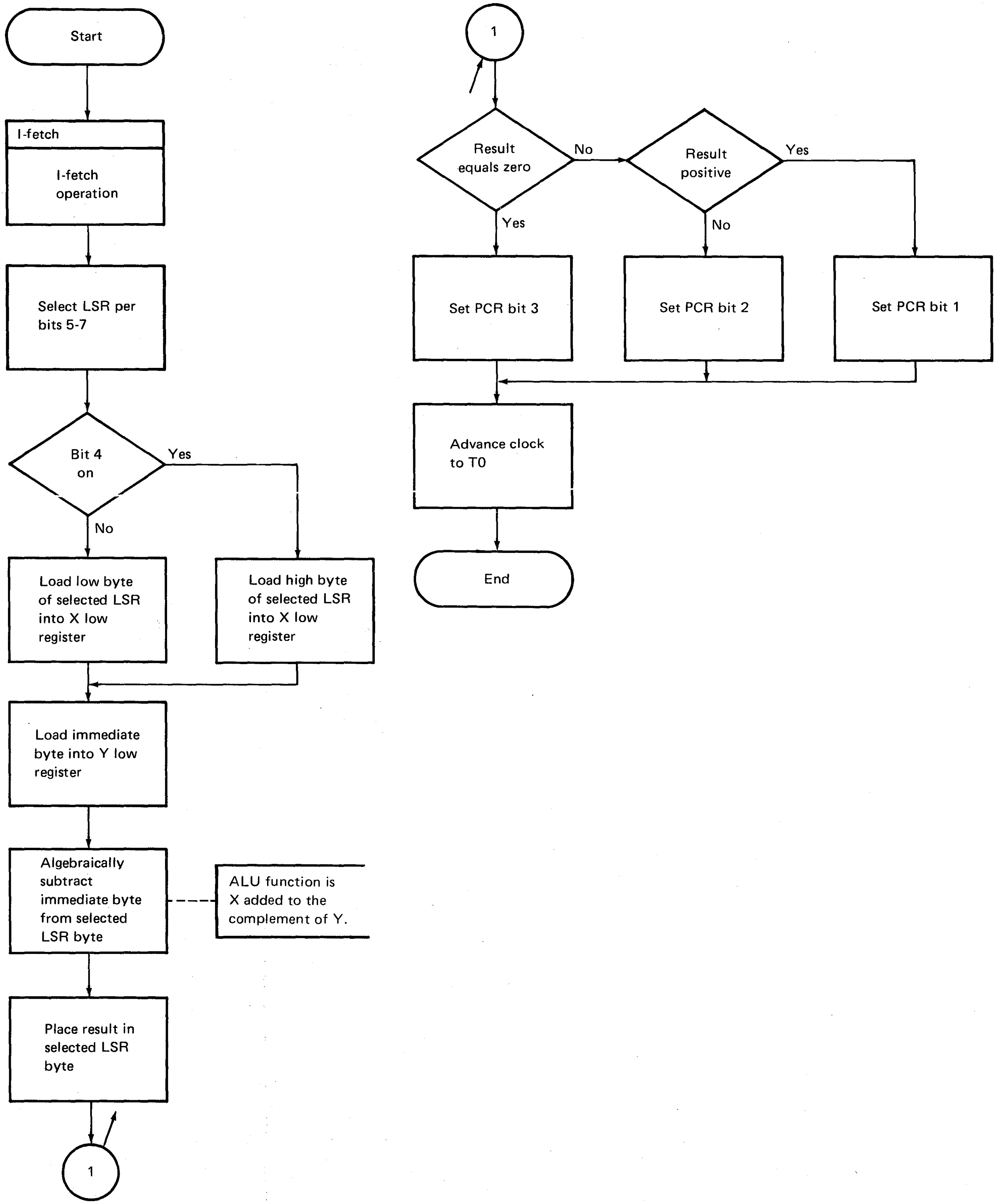
*Immediate Byte (Bits 8-15):* Contains the data to be compared with the data in the selected local storage register.

*Condition Code*

The condition code is set as follows:

- Positive (bit 1 of the processor condition register)–Register data is larger than the data field.
- Negative (bit 2 of the processor condition register)–Register data is less than the data field.
- Zero (bit 3 of the processor condition register)–Register data is equal to the data field.

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 5-7
Bit 4 on
Yes
No
Load low byte of selected LSR into X low register
Load high byte of selected LSR into X low register
Load immediate byte into Y low register
Algebraically subtract immediate byte from selected LSR byte
ALU function is X added to the complement of Y.
Place result in selected LSR byte
1
1
Result equals zero
No
Yes
Result positive
Yes
No
Set PCR bit 3
Set PCR bit 2
Set PCR bit 1
Advance clock to T0
End

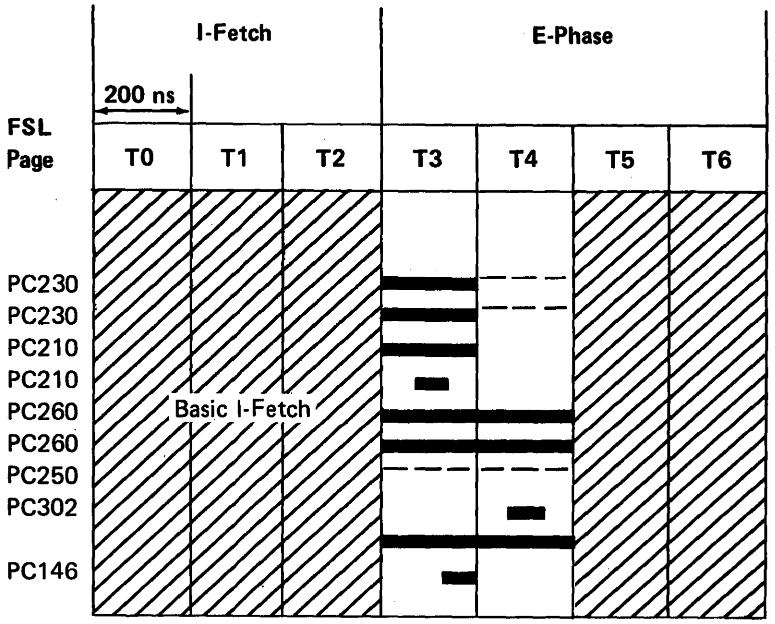
**Hex C1XX**

Select LSR
Stg Gate High/Low from LSR
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X–Y–1+carry)
ALU Gate Low from ALU Low
ALU Gate High from ALU Low
Clock PCR (bits 1, 2, 3)
Carry
Clock Stg Gate Check

**Instruction Loop**

```
00    A0FF    LI
01    C1FF    CI * (see note)
02    0000    B
```

*Note:* This instruction uses the low byte of the LSR.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

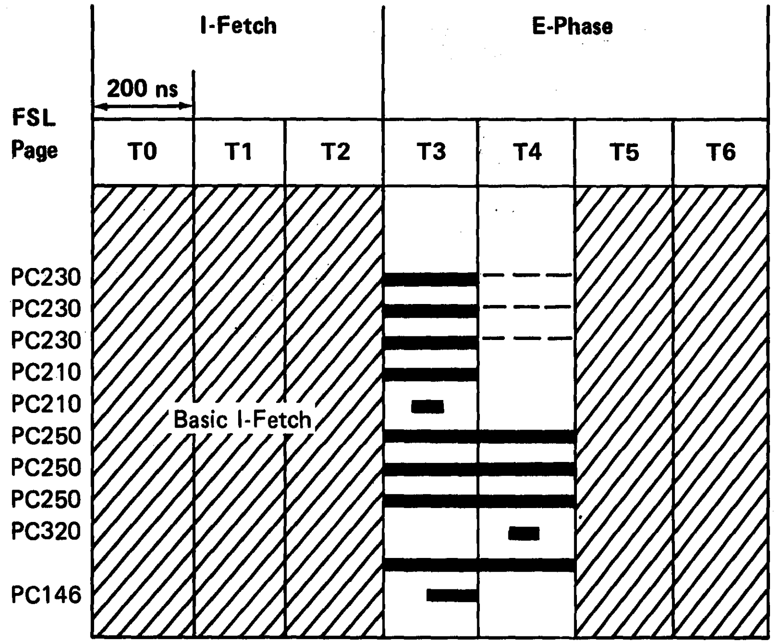
**Hex C9XX**

Select LSR
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X–Y–1+carry)
ALU Gate Low from ALU Low (don't care)
ALU Gate High from ALU Gate Low (don't care)
Clock PCR (bits 1, 2, 3)
Carry
Clock Stg Gate Check

**Instruction Loop**

```
00    A0FF    LI
01    C9FF    CI * (see note)
02    0000    B
```

*Note:* This instruction uses the high byte of the LSR.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Subtract Immediate/Add Immediate (SI, AI)

| | | | |
|---|---|---|---|
| 1 1 0 1 | H1 | Reg 1 | Immediate Byte |
| 0 3 | 4 | 5 7 | 8 15 |

The data in the immediate byte of this instruction is subtracted from the data in the specified local storage register (register 1).

The add immediate instruction is valid for the control storage program only. To add immediate, the immediate data must be complemented by the assembler and then inserted in the immediate field of the instruction (complement subtract = addition). The immediate field then becomes a constant and is coded before assembly with the value to be used.

These instructions can also be used to compare two operands by testing the condition code after executing the instruction.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used in the subtract operation:

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level.

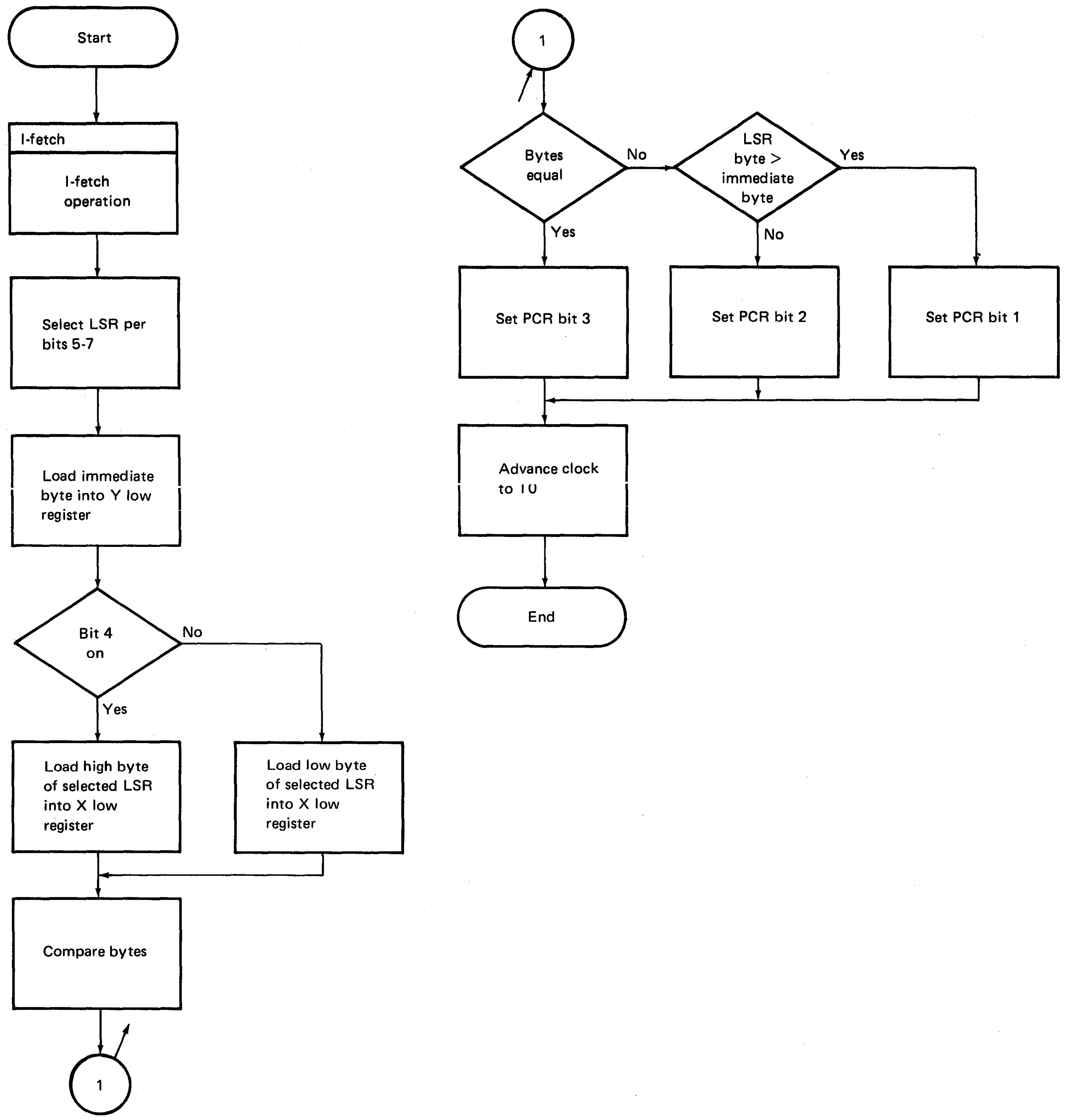
*Immediate Byte (Bits 8-15):* Contains the data to be subtracted from the data in the selected local storage register.

*Condition Code*

The condition code is set as follows:

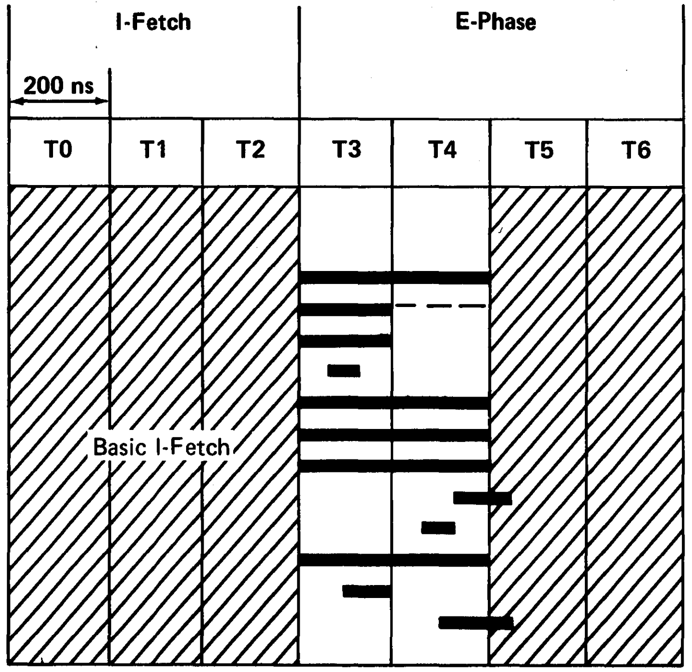
- Positive (bit 1 of the processor condition register)–Register data is larger than the data field.
- Negative (bit 2 of the processor condition register)–Register data is less than the data field.
- Zero (bit 3 of the processor condition register)–Register data and the data field are equal.

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 5-7
Load immediate byte into Y low register
Bit 4 on
No
Yes
Load high byte of selected LSR into X low register
Load low byte of selected LSR into X low register
Compare bytes
1
1
Bytes equal
No
Yes
LSR byte > immediate byte
Yes
No
Set PCR bit 3
Set PCR bit 2
Set PCR bit 1
Advance clock to 10
End

### Hex D1XX

| | FSL Page |
|---|---|
| Select LSR | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| Y Low from SDR Low (Y high, don't care) | PC210 |
| X Low from Stg Gate Low (X high, don't care) | PC210 |
| Set ALU Mode (X–Y–1+carry) | PC260 |
| ALU Gate Low from ALU Low | PC250 |
| ALU Gate High from ALU Gate Low (don't care) | PC250 |
| Write LSR Low | PC160 |
| Clock PCR (bits 1, 2, 3) | PC320 |
| Carry | |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | D100 | SI * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.

**Scope Setup**

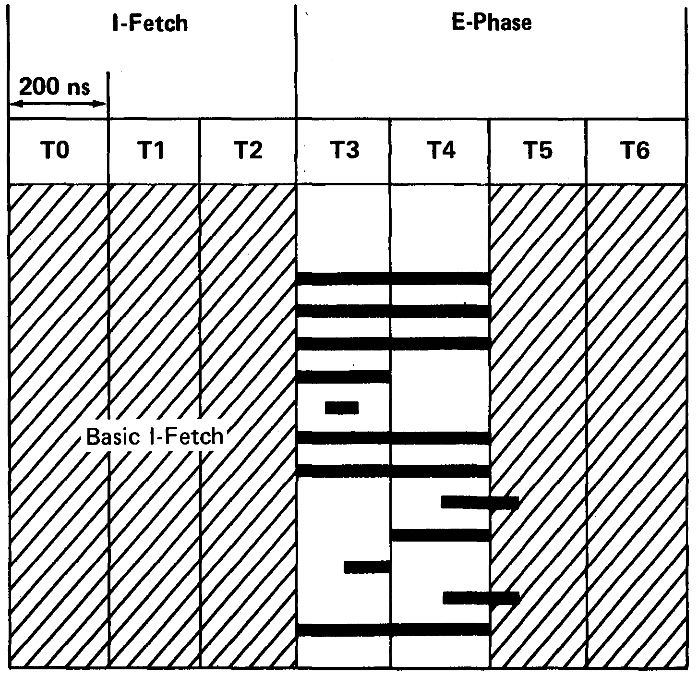
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' ' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

### Hex D9XX

| | FSL Page |
|---|---|
| Select LSR | PC230 |
| Stg Gate High from LSR | PC230 |
| Stg Gate Low from Stg Gate High | PC230 |
| Y Low from SDR Low (Y high, don't care) | PC210 |
| X Low from Stg Gate Low (X high, don't care) | PC210 |
| Set ALU Mode (X–Y–1+carry) | PC260 |
| ALU Gate High/Low from ALU High/Low | PC260 |
| Write LSR High | PC160 |
| Clock PCR (bits 1, 2, 3) | PC302 |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |
| Carry | |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A9FF | LI |
| 01 | D900 | SI * (see note) |
| C2 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' ' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Test Mask (TM)

| | | | |
|---|---|---|---|
| 0 1 0 1 | H1 | Reg 1 | Mask |
| 0 3 | 4 | 5 7 | 8 15 |

This instruction tests the bits in 1 byte of a work register. A mask byte in the instruction identifies the bits to be tested. As a result of this test, one of the three following conditions will be found and this condition is set in the processor condition register:

- Positive = Ones–The tested bits are all equal to 1 (processor condition register bit 1 is set on).
- Negative = Mixed–The tested bits are a combination of ones and zeros (processor condition register bit 2 is set on).
- Zero = Zeros–The tested bits are all equal to 0 (processor condition register bit 3 is set on).

*H1 (Bit 4):* Selects the low- or high-order byte of the register:

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level.

*Mask (Bits 8-15):* Any bit set to 1 indicates that the comparable bit in the selected byte is to be tested. Any bit set to 0 indicates that the comparable bit is to be ignored.

*Condition Code*

| Result of Test | Condition Code |
|---|---|
| Tested bits all = 1 | Positive |
| Tested bits are mixed | Negative |
| Tested bits all = 0 | Zero |

Example:

H1 = 0
Reg 1 = 011

Interrupt level = 0

Mask = 0 0 1 0 1 0 0 1    Bits 2, 4, and 7 in LSR 3 are to be tested.

LSR 3 = 0 1 1 0 1 1 0 1

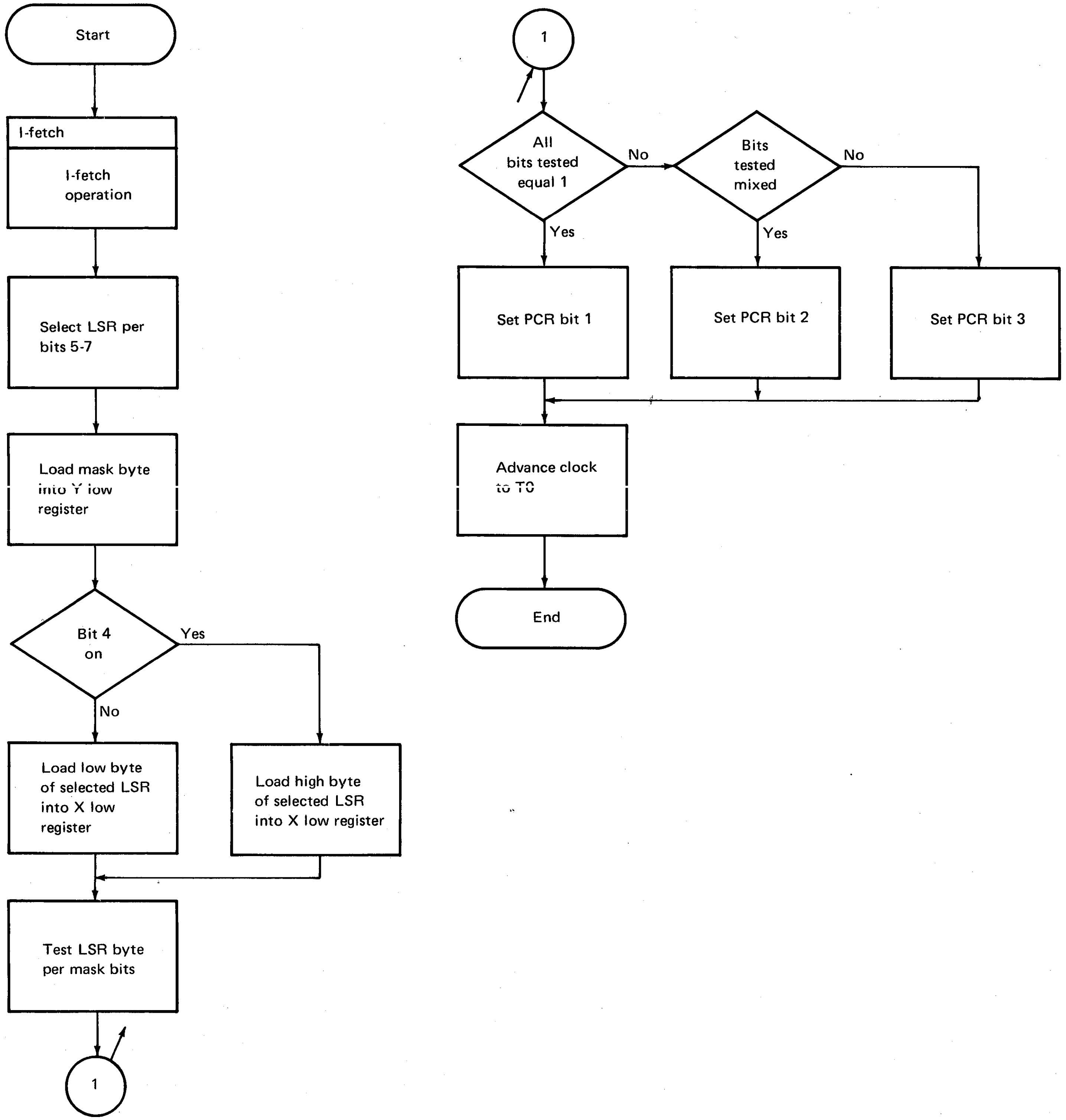
Condition Code Set: Positive

PCR = 01000000

Bits tested all equal 1.

| Bits | Selected LSR (Hexadecimal) | | | | | |
|---|---|---|---|---|---|---|
| | Interrupt Level | | | | | |
| 5 6 7 | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 0 0 | 0 | 10 | 18 | 20 | 30 | 38 |
| 0 0 1 | 1 | 11 | 19 | 21 | 31 | 39 |
| 0 1 0 | 2 | 12 | 1A | 22 | 32 | 3A |
| 0 1 1 | 3 | 13 | 1B | 23 | 33 | 3B |
| 1 0 0 | 4 | 14 | 1C | 24 | 34 | 3C |
| 1 0 1 | 5 | 15 | 1D | 25 | 35 | 3D |
| 1 1 0 | 6 | 16 | 1E | 26 | 36 | 3E |
| 1 1 1 | 7 | 17 | 1F | 27 | 37 | 3F |

*Sequence and Timing*



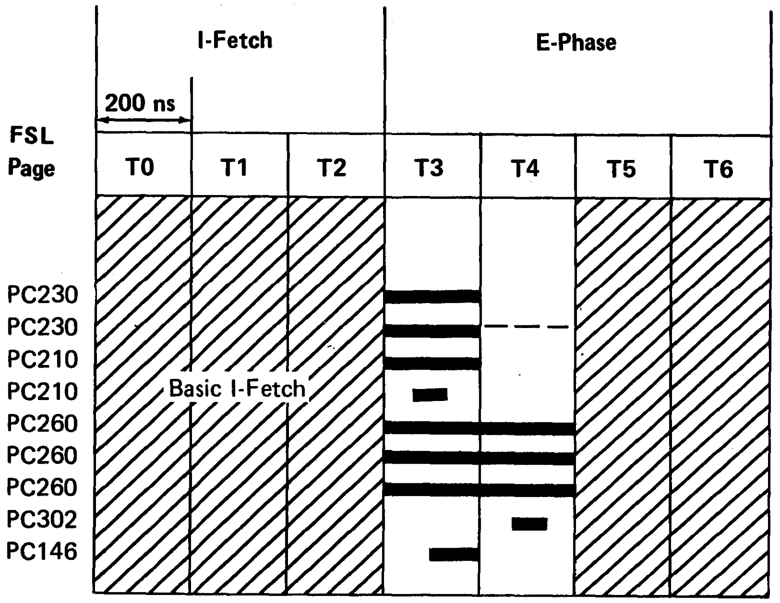
**Hex 51XX**

Select LSR
Stg Gate High/Low from LSR
Y Low from SDR Low (X high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X and Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Gate Low
Clock PCR (bits 1, 2, 3)
Clock Stg Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | 51FF | TM * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

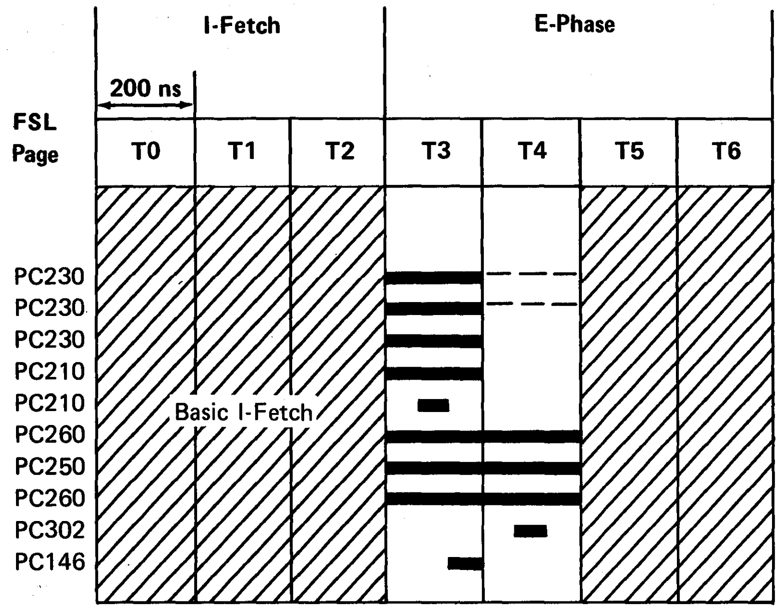
**Hex 59XX**

Select LSR
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X and Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Gate Low
Clock PCR (bits 1, 2, 3)
Clock Stg Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A9FF | LI |
| 01 | 59FF | TM * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Set Bits On (SBN)

| 1 0 0 1 | H1 | Reg 1 | Data |
|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 15 |

This instruction sets bits in the high- or low-order byte of the selected local storage register to 1.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used:

H1 = 0: Low-order byte

H1 = 1: High-order byte

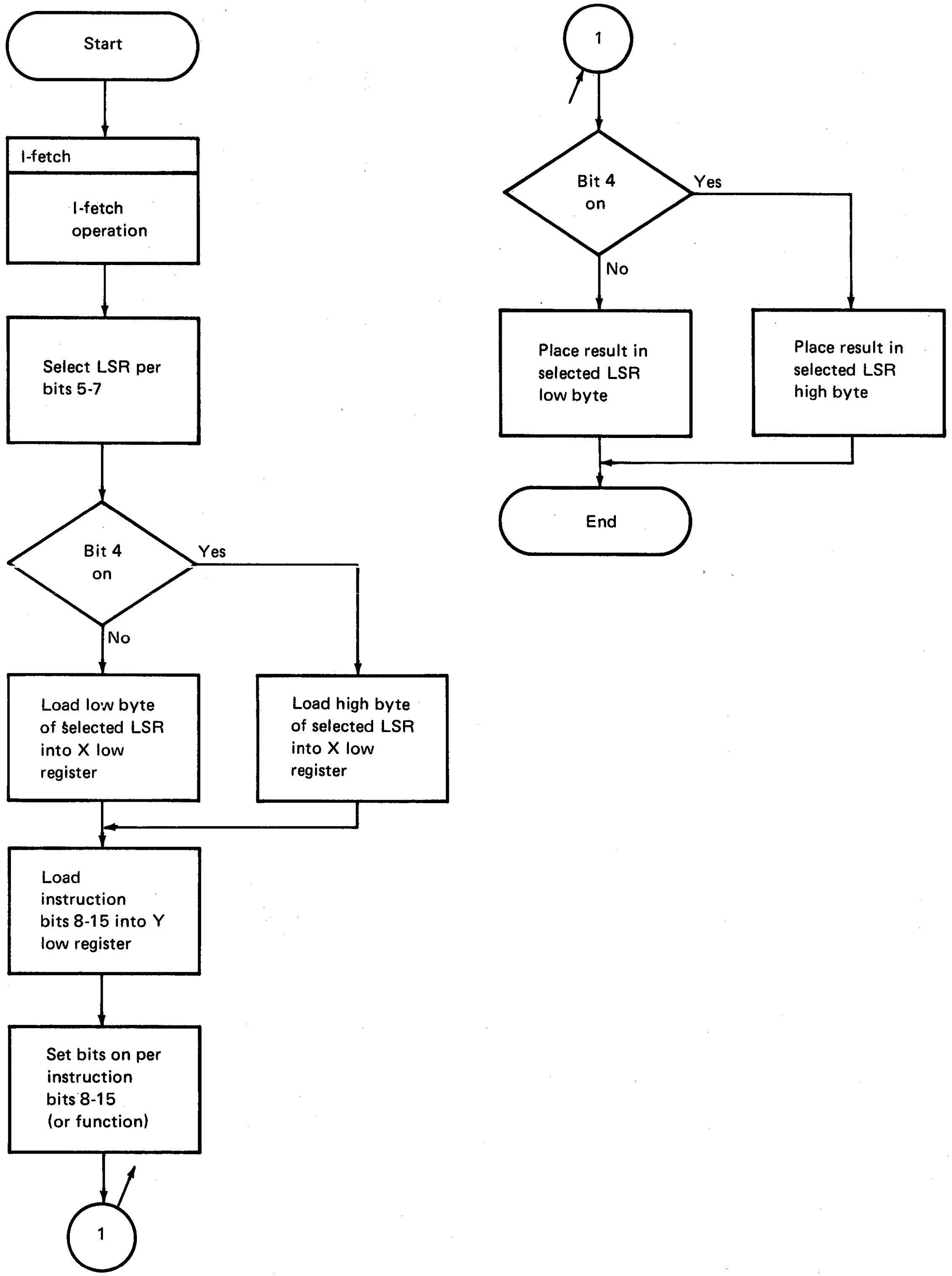
*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The byte of the register is combined, using an OR operation, with the data in the data field.

*Data (Bits 8-15):* The 8 bits of this field are compared with the 8 bits in the selected register. Any bit in the data field that is set to 1 causes the same bit in the selected register to be set to 1. Any bits in the data field that are set to 0 do not affect any bits in the selected register.

*Condition Code*

No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 5-7
Bit 4 on
Yes
No
Load low byte of selected LSR into X low register
Load high byte of selected LSR into X low register
Load instruction bits 8-15 into Y low register
Set bits on per instruction bits 8-15 (or function)
1
1
Bit 4 on
Yes
No
Place result in selected LSR low byte
Place result in selected LSR high byte
End

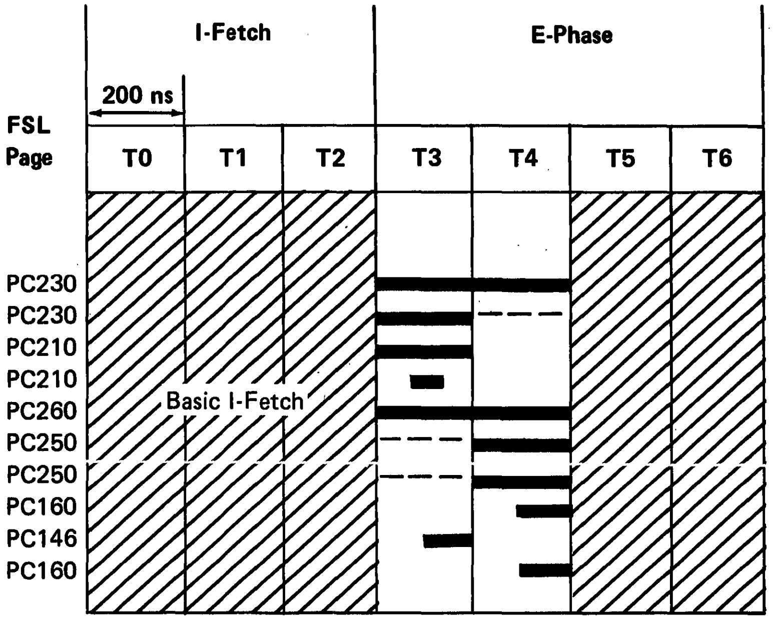
**Hex 91XX**

Select LSR
Stg Gate High/Low from LSR
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X or Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Gate Low (don't care)
Write LSR Low
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | 91FF | SBN * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.



**Scope Setup**

Horizontal = 0.1 µs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

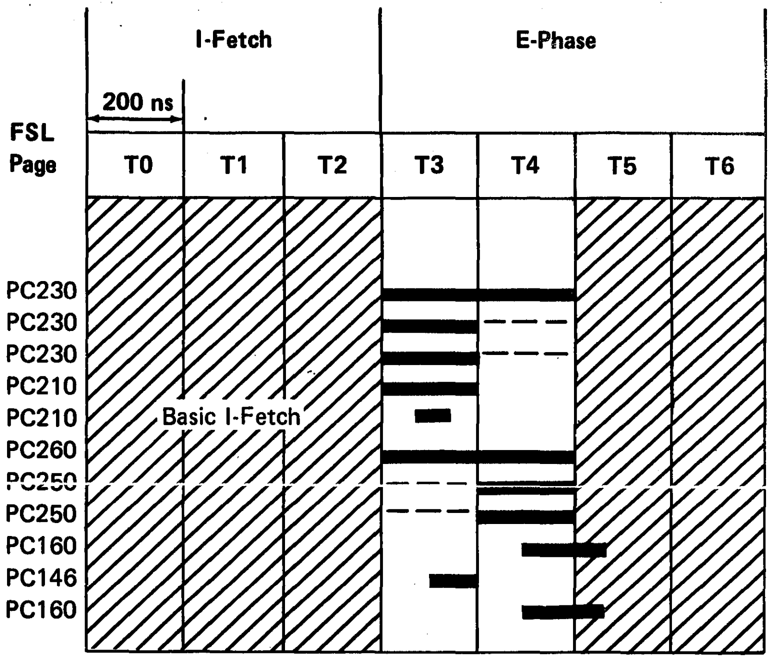
**Hex 99XX**

Select LSR
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X or Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Low
Write LSR High
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A9FF | LI |
| 01 | 99FF | SBN * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.



**Scope Setup**

Horizontal = 0.1 µs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Set Bits Off (SBF)

| 1 0 0 0 | H1 | Reg 1 | Data |
|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 15 |

This instruction resets bits in the high- or low-order byte of the selected register in the local storage register stack.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used:

H1 = 0: Low-order byte

H1 = 1: High-order byte

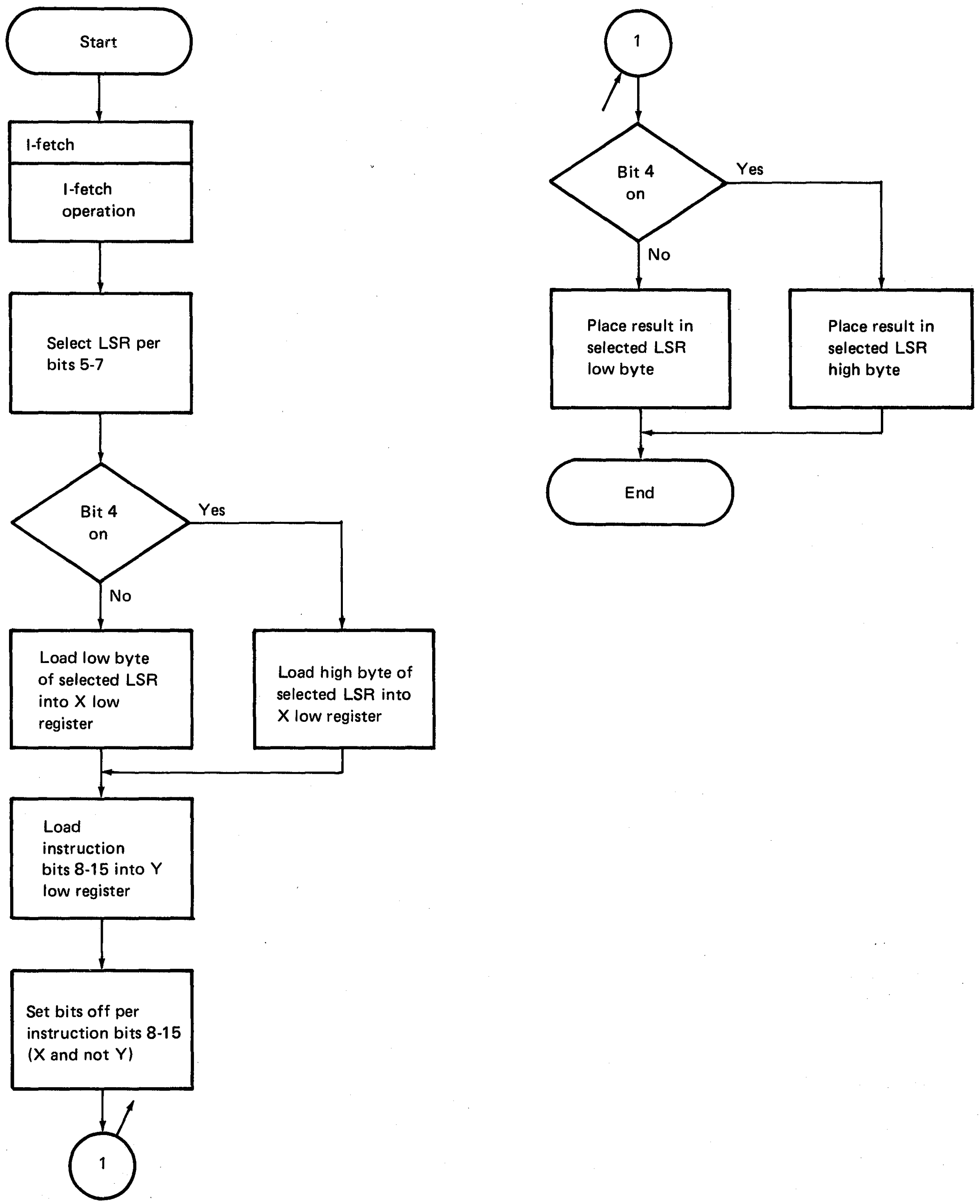
*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The contents of the register are ANDed with the complement of the data in the data field.

*Data (Bits 8-15):* The 8 bits in this field are compared with the 8 bits of the selected register. Any bit in the data field that is set to 1 causes the same bit in the selected register to be set to 0. Any bits in the data field that are set to 0 do not affect any bits in the selected register.

*Condition Code*

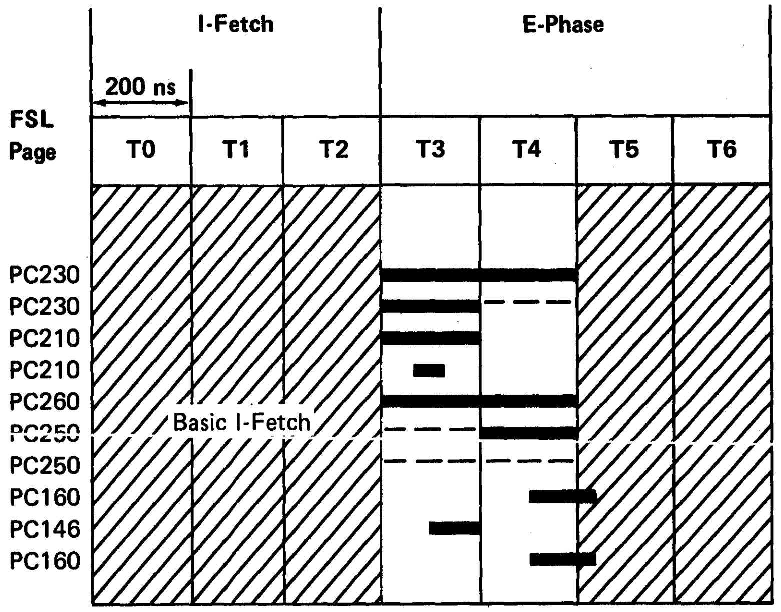
No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Select LSR per bits 5-7
Bit 4 on
Yes
No
Load low byte of selected LSR into X low register
Load high byte of selected LSR into X low register
Load instruction bits 8-15 into Y low register
Set bits off per instruction bits 8-15 (X and not Y)
1
1
Bit 4 on
Yes
No
Place result in selected LSR low byte
Place result in selected LSR high byte
End

**Hex 81XX**

Select LSR
Stg Gate High/Low from LSR
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X and not Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Gate Low
Write LSR Low
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | 81FF | SBF * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.

**Scope Setup**

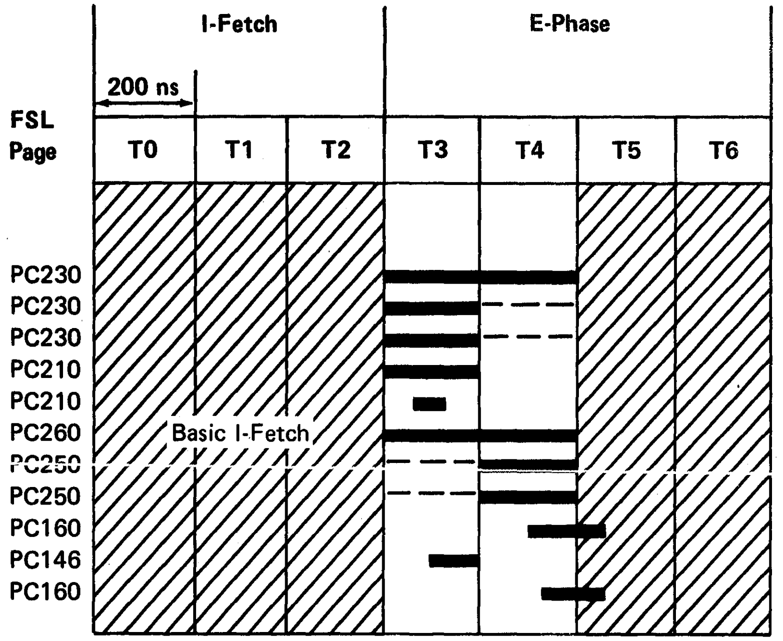
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Hex 89XX**

Select LSR
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
Y Low from SDR Low (Y high, don't care)
X Low from Stg Gate Low (X high, don't care)
Set ALU Mode (X and not Y)
ALU Gate Low from ALU Low
ALU Gate High from ALU Gate Low
Write LSR High
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A9FF | LI |
| 01 | 89FF | SBF * (see note) |
| 02 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

This page intentionally left blank.

## Storage (LC, LM, STC, STM)

| | |
|---|---|
| LC | (load from control storage) |
| STC | (store to control storage) |
| LM | (load from main storage) |
| STM | (store to main storage) |

| 0 1 0 0 | H1 | Reg 1 | 1 | W | C | D | V | Reg 2 |
|---|---|---|---|---|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

This instruction permits access to either control storage or main storage. Data can be moved to or from the local storage registers.

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used:

H1 = 0: Low-order byte

H1 = 1: High-order byte

Bit 4 is not used when bit 10 is on. When bit 10 is on, both the high- and low-order bytes are selected.

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. Data is moved to or from this register.

*Bit 8:* If bit 8 = 1, the operation code (bits 0-3) of the control storage instruction is changed. If bit 8 = 0, the instruction is an I/O storage instruction.

*W (Bit 9):* Identifies the direction the data is to be moved:

W = 0: Read from storage and move to the local storage register stack

W = 1: Move from the local storage register stack and write to storage

*C (Bit 10):* Selects main storage or control storage:

C = 0: Selects main storage

C = 1: Selects control storage

*D (Bit 11):* Indicates if the address in the local storage register (specified by bits 13-15) should be increased or decreased:

D = 0: Increase the selected local storage register by the value of field V

D = 1: Decrease the selected local storage register by the value of field V

*V (Bit 12):* Indicates the amount the address in the local storage register (specified by bits 13-15) should be increased or decreased:

V = 0: The selected local storage register is not changed (register 2).

V = 1: The address in the selected local storage register is increased or decreased by 1 as determined by the bit setting of field D (register 2).

*Register 2 (Bits 13-15):* Selects one of the eight work registers assigned to the current interrupt level that contains the storage address of the data. The address in the specified local storage register may be updated as specified by bit 11 (field D) and bit 12 (field V).

**Instruction List**

| Bits | | | | | | Mne- | |
|---|---|---|---|---|---|---|---|
| **4** | **8** | **9** | **10** | **11** | **12** | **monic** | **Description** |
| X | 1 | 0 | 1 | 0 | 1 | LC | Load from control storage, increase register 2 by 1. |
| X | 1 | 0 | 1 | 1 | 1 | LC | Load from control storage, decrease register 2 by 1. |
| X | 1 | 0 | 1 | 0 | 0 | LC | Load from control storage, no change to register 2. |
| X | 1 | 1 | 1 | 0 | 1 | STC | Store to control storage, increase register 2 by 1. |
| X | 1 | 1 | 1 | 1 | 1 | STC | Store to control storage, decrease register 2 by 1. |
| X | 1 | 1 | 1 | 0 | 0 | STC | Store to control storage, no change to register 2. |
| H | 1 | 0 | 0 | 0 | 1 | LM | Load from main storage, increase register 2 by 1. |
| H | 1 | 0 | 0 | 1 | 1 | LM | Load from main storage, decrease register 2 by 1. |
| H | 1 | 0 | 0 | 0 | 0 | LM | Load from main storage, no change to register 2. |
| H | 1 | 1 | 0 | 0 | 1 | STM | Store to main storage, increase register 2 by 1. |
| H | 1 | 1 | 0 | 1 | 1 | STM | Store to main storage, decrease register 2 by 1. |
| H | 1 | 1 | 0 | 0 | 0 | STM | Store to main storage, no change to register 2. |

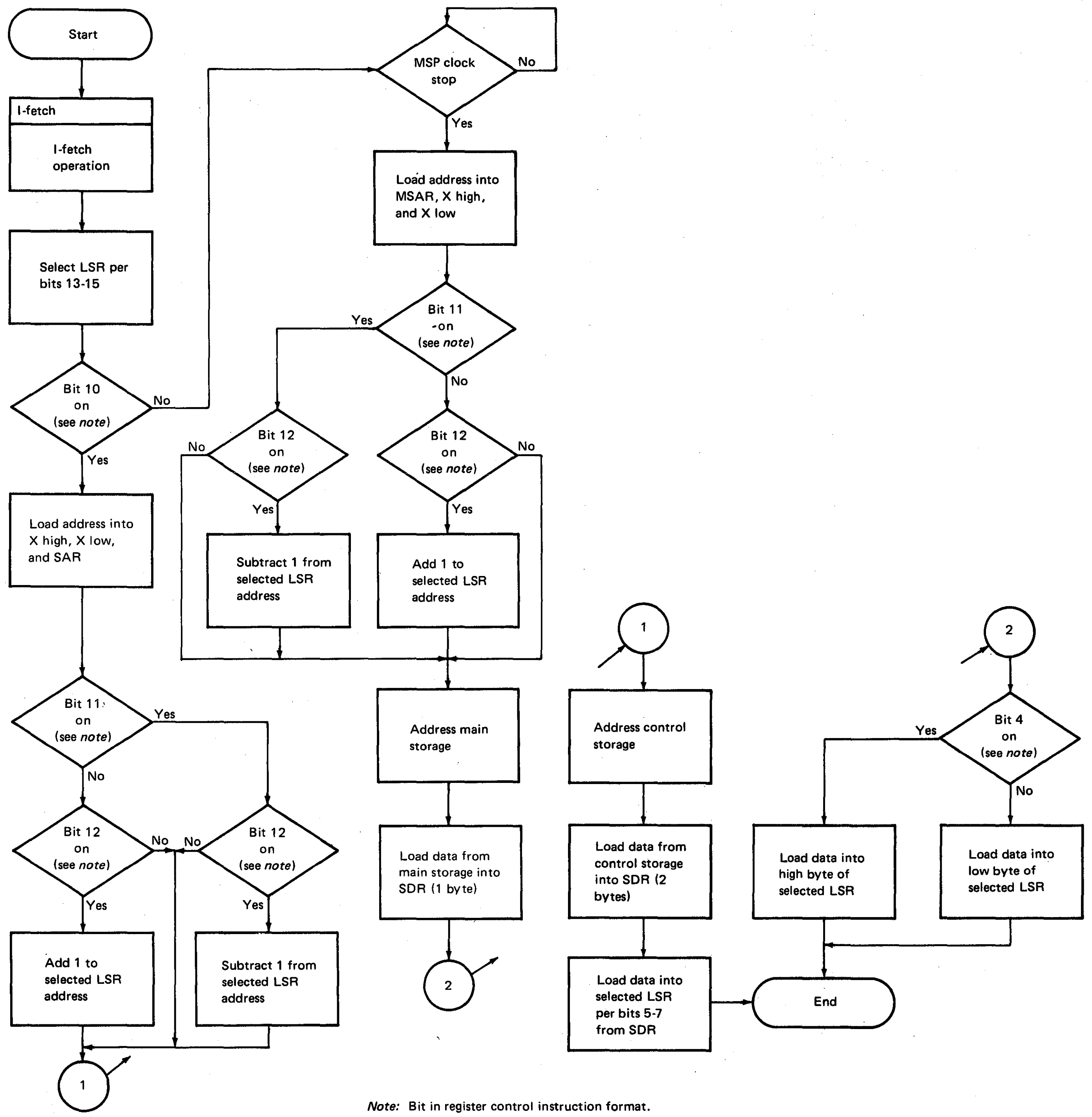
Legend for Bit 4:

| | |
|---|---|
| X: | Not used |
| H = 0: | Low-order byte |
| H = 1: | High-order byte |

*Condition Code*

No change

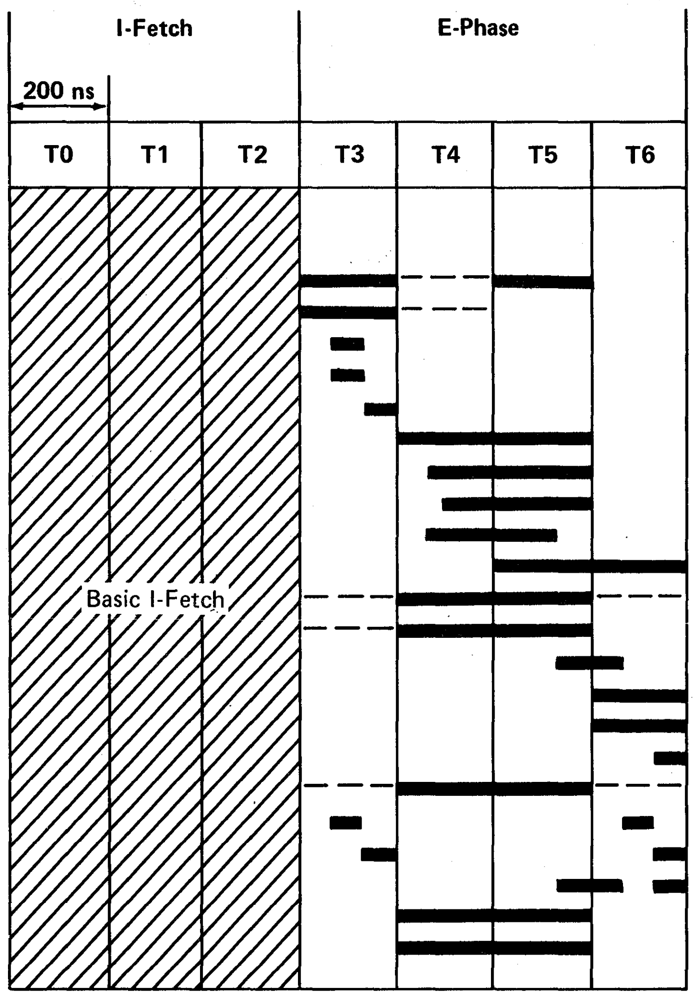
*Sequence and Timing for Reading from Storage (LC, LM)*


Start
I-fetch
I-fetch operation
Select LSR per bits 13-15
Bit 10 on (see *note*)
No
Yes
Load address into X high, X low, and SAR
MSP clock stop
No
Yes
Load address into MSAR, X high, and X low
Bit 11 -on (see *note*)
Yes
No
Bit 12 on (see *note*)
No
Yes
Subtract 1 from selected LSR address
Bit 12 on (see *note*)
No
Yes
Add 1 to selected LSR address
Address main storage
Load data from main storage into SDR (1 byte)
2
Bit 11 on (see *note*)
Yes
No
Bit 12 on (see *note*)
No
Yes
Add 1 to selected LSR address
Bit 12 on (see *note*)
No
Yes
Subtract 1 from selected LSR address
1
1
Address control storage
Load data from control storage into SDR (2 bytes)
Load data into selected LSR per bits 5-7 from SDR
End
2
Bit 4 on (see *note*)
Yes
No
Load data into high byte of selected LSR
Load data into low byte of selected LSR

*Note:* Bit in register control instruction format.

**Hex 41AA**

| | FSL Page |
|---|---|
| Select LSR (address) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X High/Low from Stg Gate High/Low | PC210 |
| SAR from Stg Gate High/Low | |
| Stg Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| Clock SDR from CS | PC220 |
| Stg Gate High/Low from SDR High/Low | PC230 |
| Set ALU Mode (X + carry) (see Note 1) | PC260 |
| ALU Gate High/Low from ALU High/Low | PC250 |
| Write LSR High/Low (address) | PC160 |
| ALU Gate High/Low from Stg Gate High/Low | PC250 |
| Select LSR (data) | PC230 |
| Write LSR High/Low (data) | PC160 |
| Carry | |
| Clock SDR Check | PC220 |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |
| Ctl Storage Address Check | |
| Ctl Storage SAR P Check | |



[1] This line cannot be probed.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A2FF | LI |
| 01 | AA01 | LI |
| 02 | 41AA | LC * (see Note 2) |
| 03 | 0000 | B |

*Notes:*
1. ALU mode setting may be pass or X–1 carry, depending on the instruction.
2. Control storage operation uses a forced 2-byte data path.

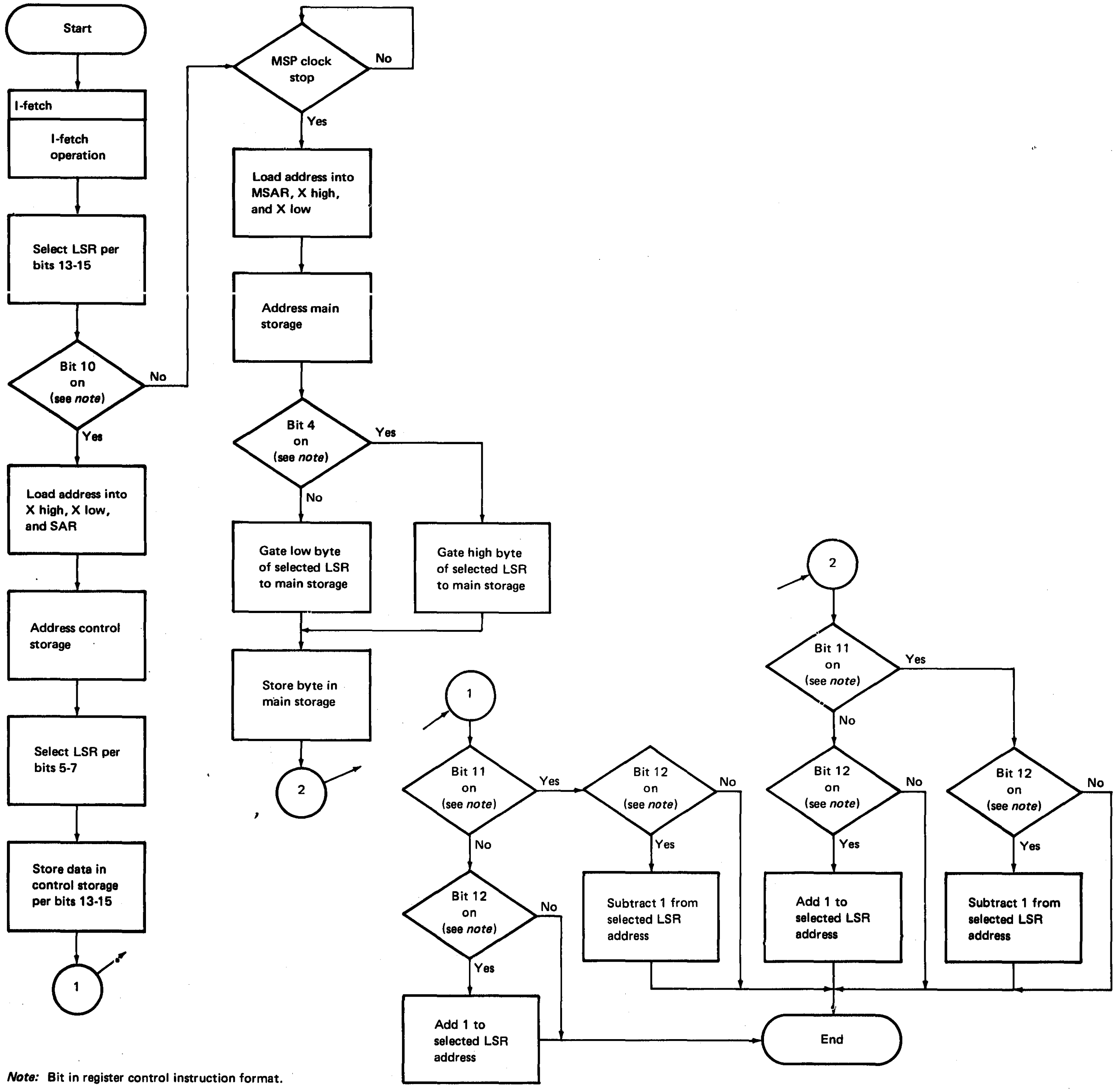
**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

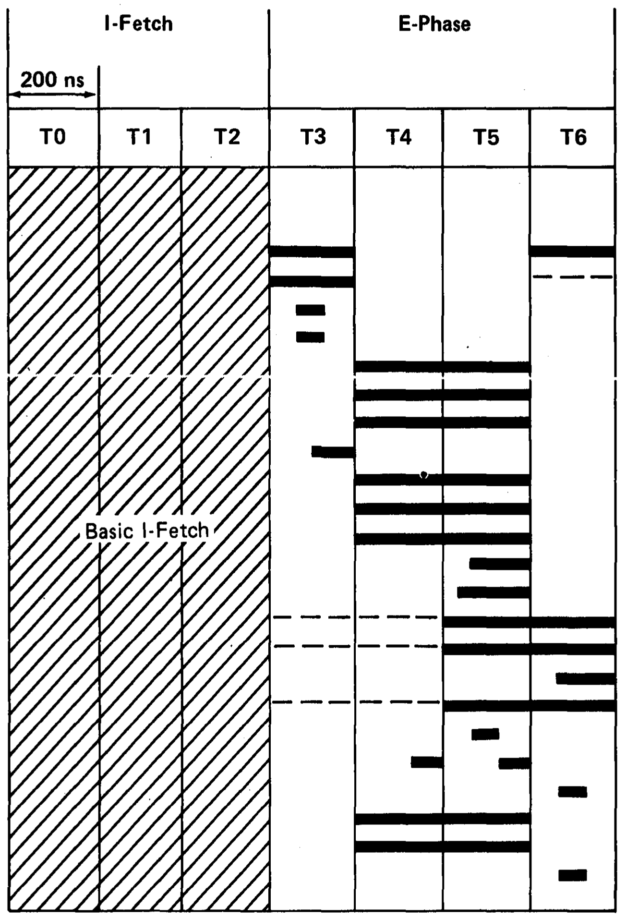
## *Sequence and Timing for Writing into Storage (STC, STM)*


Start
I-fetch
I-fetch operation
Select LSR per bits 13-15
Bit 10 on (see *note*)
No
Yes
Load address into X high, X low, and SAR
Address control storage
Select LSR per bits 5-7
Store data in control storage per bits 13-15
1
MSP clock stop
No
Yes
Load address into MSAR, X high, and X low
Address main storage
Bit 4 on (see *note*)
Yes
No
Gate low byte of selected LSR to main storage
Gate high byte of selected LSR to main storage
Store byte in main storage
2
1
Bit 11 on (see *note*)
Yes
No
Bit 12 on (see *note*)
No
Yes
Add 1 to selected LSR address
Bit 12 on (see *note*)
No
Yes
Subtract 1 from selected LSR address
2
Bit 11 on (see *note*)
Yes
No
Bit 12 on (see *note*)
No
Yes
Add 1 to selected LSR address
Bit 12 on (see *note*)
No
Yes
Subtract 1 from selected LSR address
End

***Note:*** Bit in register control instruction format.

**Hex 41EA**

| | FSL Page |
|---|---|
| Select LSR (address) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low (address) | PC230 |
| SAR from Stg Gate High/Low | |
| Select LSR (data) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| CS from Stg Gate High/Low | |
| Stg Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| +CS Write Pulse High | |
| +CS Write Pulse Low | |
| Set ALU Mode (X + carry) (see Note 1) | PC260 |
| ALU Gate High/Low from ALU High/Low | PC260 |
| Write LSR High/Low (address) | PC160 |
| Carry | |
| Clock SDR (echo check) | PC220 |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |
| Ctl Storage Address Check | |
| Ctl Storage SAR P Check | |
| Clock SDR Check | |



[1] This line cannot be probed.

**Instruction Loop**

| | | | |
|---|---|---|---|
| 00 | A2FF | LI | |
| 01 | AA01 | LI | |
| 02 | A100 | LI | |
| 03 | A900 | LI | |
| 04 | 41EA | STC * | (see Note 2) |
| 05 | 0000 | B | |

*Notes:*

1. ALU mode setting may be X+carry or X−1+carry, depending on the instruction.
2. Control storage operation uses a forced 2-byte data path.

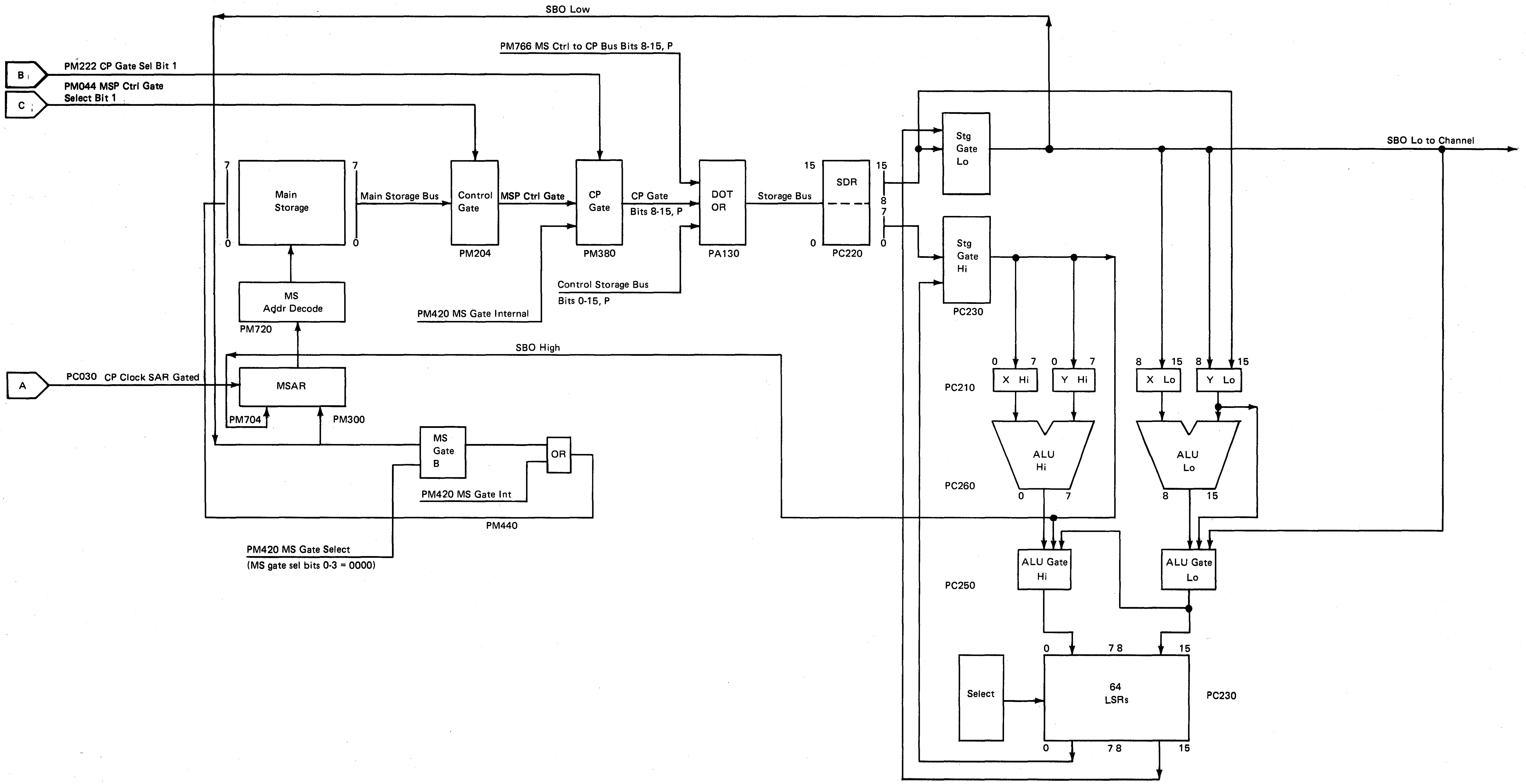
**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

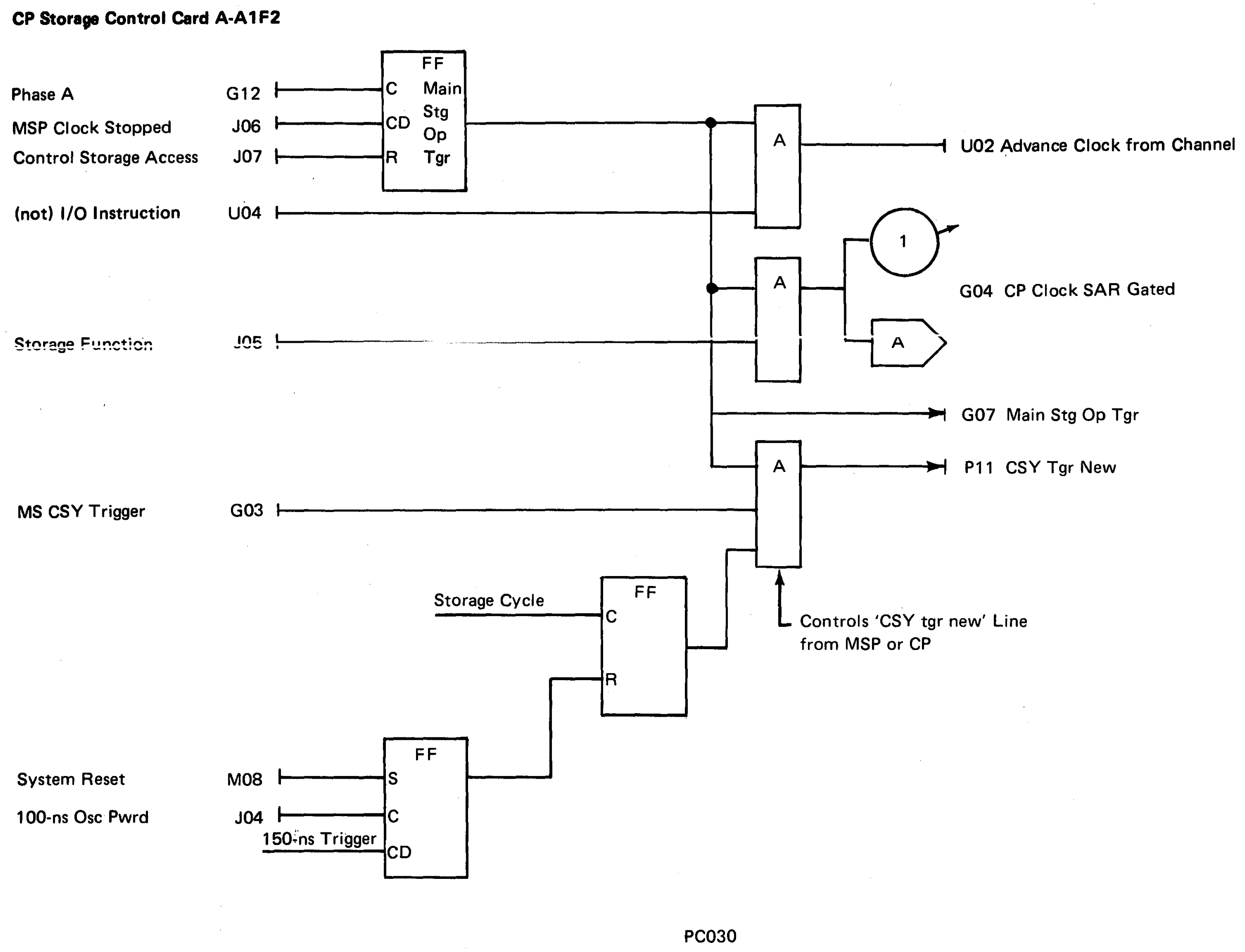
Vertical = 0.2V/div using X10 probes.

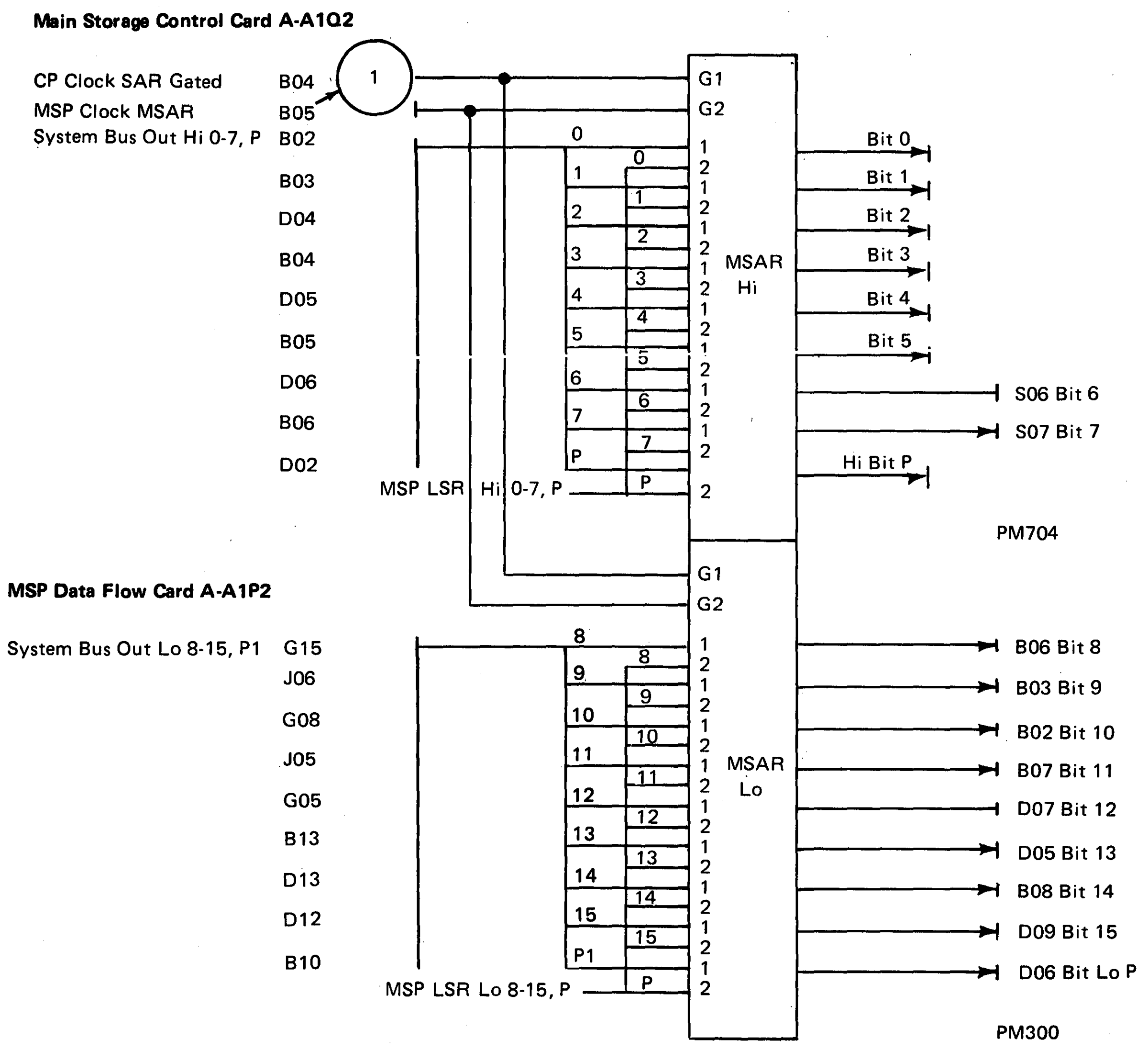
Sync External = −'address compare' looking at the instruction referenced with an asterisk (*).

## Main Storage Access by Control Processor



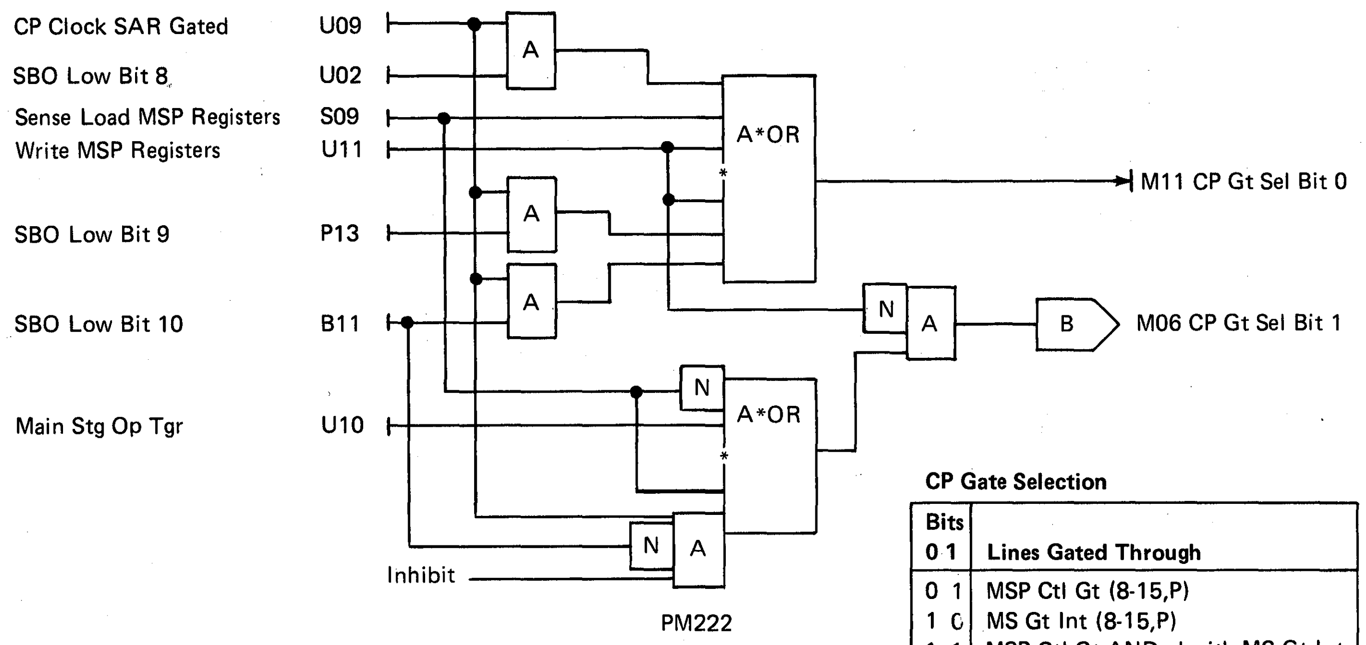
### Control Processor Control of MSAR





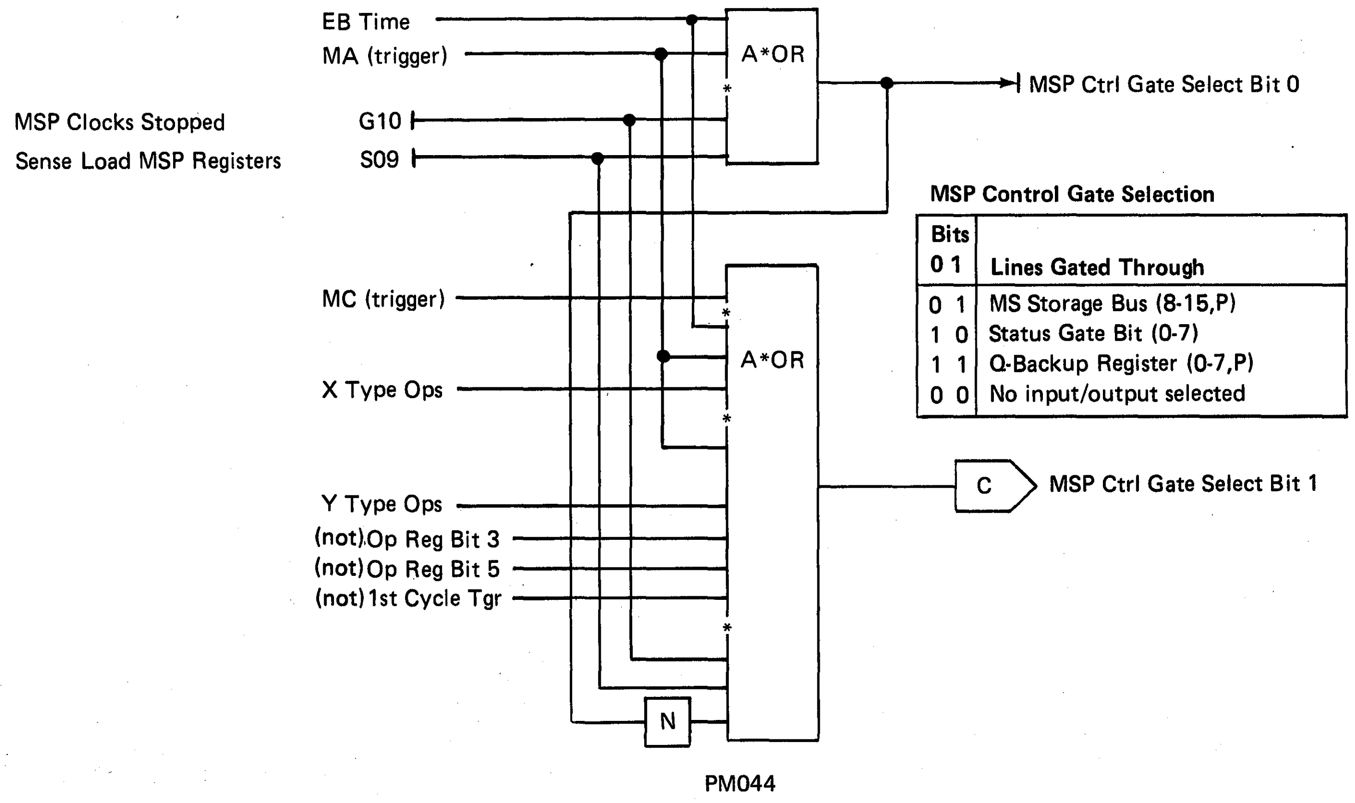
## MSP Bus Line Control

### MSP Control Card A-A1N2



**CP Gate Selection**

| Bits | |
|---|---|
| 0 1 | Lines Gated Through |
| 0 1 | MSP Ctl Gt (8-15,P) |
| 1 0 | MS Gt Int (8-15,P) |
| 1 1 | MSP Ctl Gt ANDed with MS Gt Int |
| 0 0 | No input/output selected |

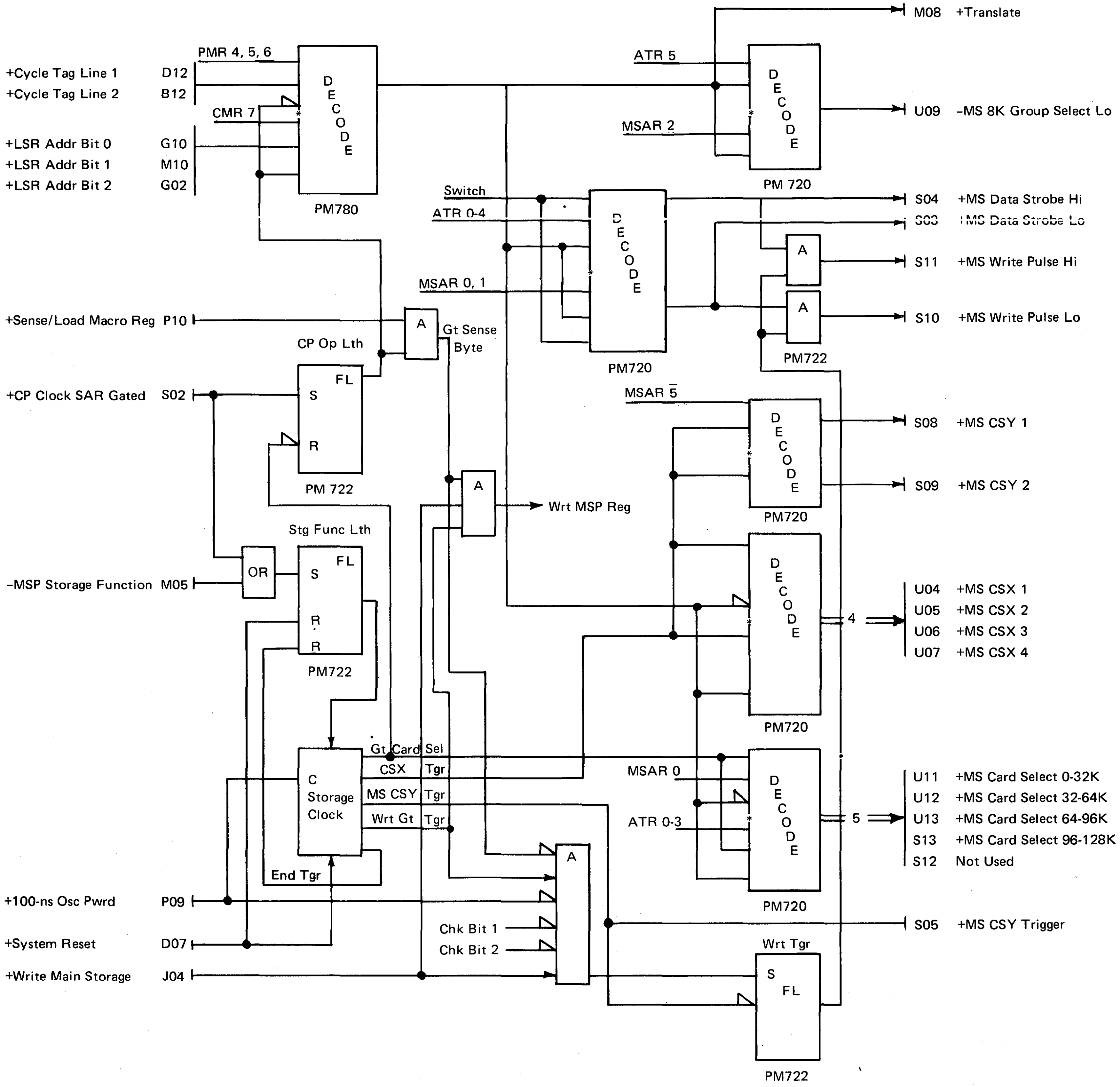


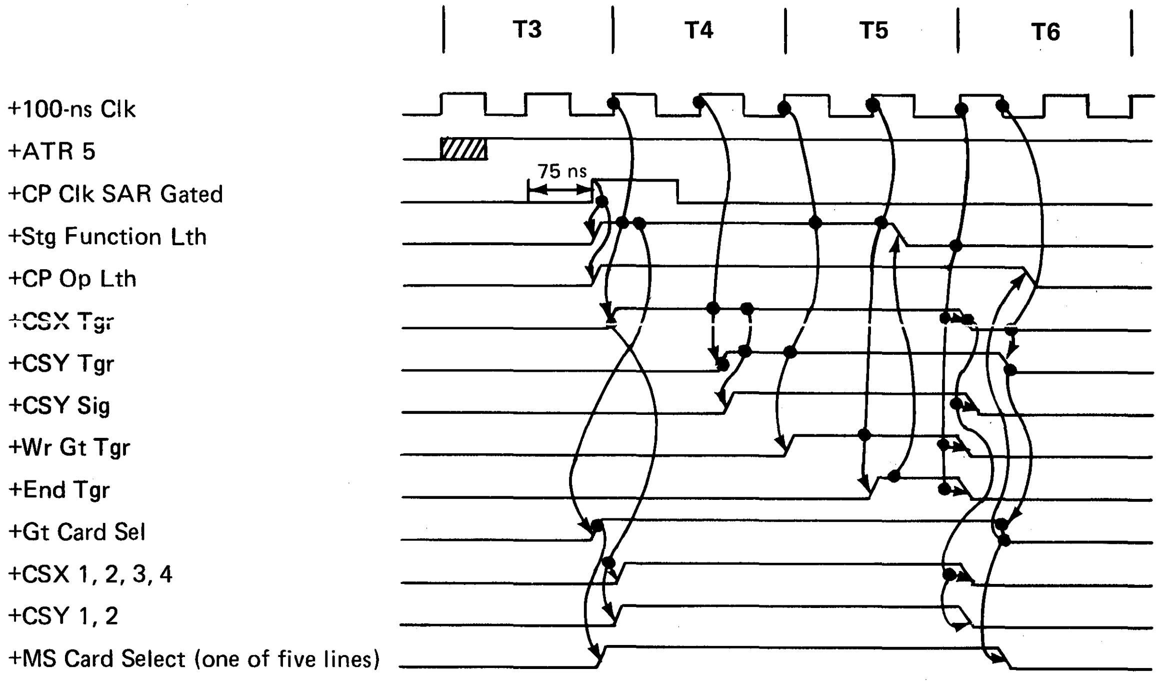
**MSP Control Gate Selection**

| Bits | |
|---|---|
| 0 1 | Lines Gated Through |
| 0 1 | MS Storage Bus (8-15,P) |
| 1 0 | Status Gate Bit (0-7) |
| 1 1 | Q-Backup Register (0-7,P) |
| 0 0 | No input/output selected |

### Main Storage Address Decoding

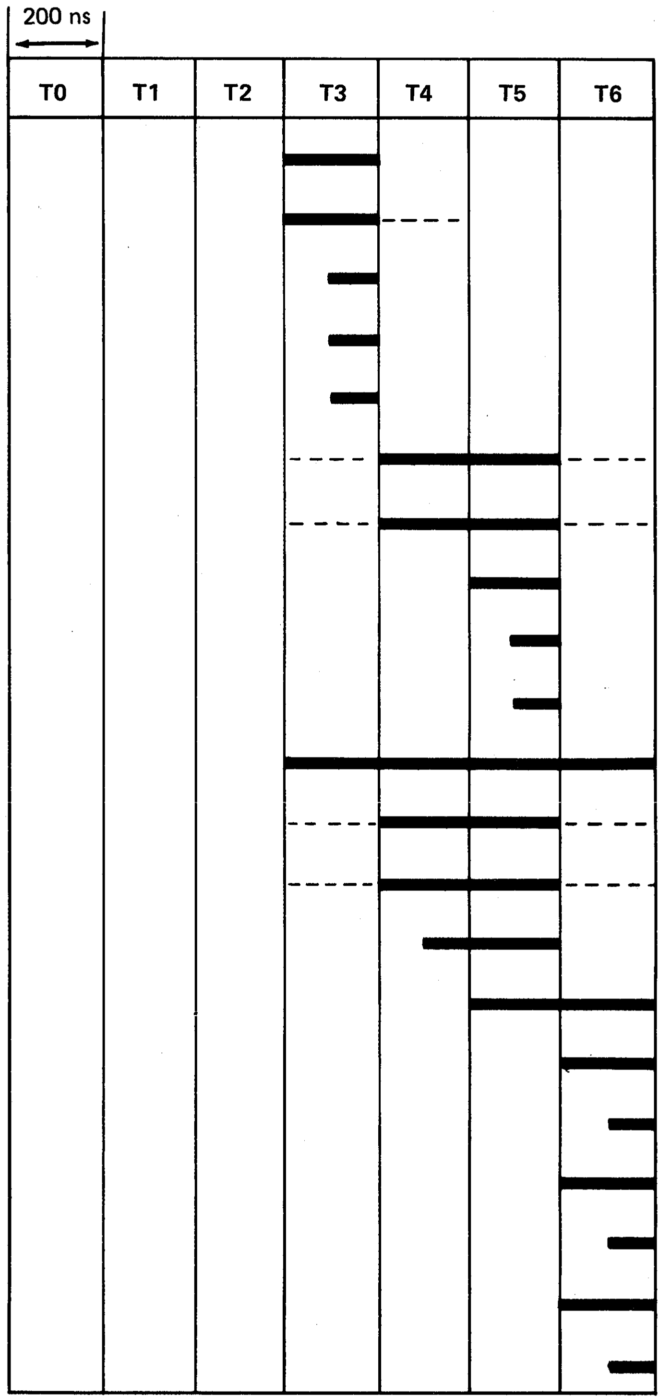
MSP Storage Control Card A-A1Q2





**Load from Main Storage (LM)**

| Step | | FSL Page |
|---|---|---|
| 1 | Select LSR (addr) (bits 13-15) | PC230 |
| 2 | Select Stg Gate Hi/Lo (from LSR hi/lo) | PC230 |
| 3 | Clock Stg Gate Check | PC146 |
| 4 | Clock X Hi, X Lo, SAR | PC210 |
| 5 | Clock MSAR | PC030 |
| 6 | Select ALU Mode (X+carry) | PC230 |
| 7 | ALU Gate Hi/Lo (from ALU hi/lo) | PC250 |
| 8 | Select LSR (addr) (bits 13-15) | PC230 |
| 9 | Write LSR Hi/Lo (address) | PC160 |
| 10 | Clock ALU Gate Check | PC160 |
| 11 | Main Stg Op Trigger | PC030 |
| 12 | Control Gate from Main Storage | PM204 |
| 13 | CP Gate from Control Gate | PM380 |
| 14 | Clock SDR (from CP gate) | PC220 |
| 15 | Select Stg Gate Hi/Lo (from SDR hi/lo) | PC230 |
| 16 | Select ALU Gate Lo (from stg gate lo) | PC250 |
| 17 | Clock Stg Gate Check | PC146 |
| 18 | Select ALU Gate Hi (from ALU gate lo) | PC250 |
| 19 | Clock ALU Gate Check | PC160 |
| 20 | Select LSR (data) (bits 5-7) | PC230 |
| 21 | Write LSR Hi if Bit 4=1, Lo if Bit 4=0 | PC230 |



**Instruction Loop**

| | | | |
|---|---|---|---|
| 00 | AA01 | LI | Load Main Storage Address |
| 01 | A200 | LI | |
| 02 | 418A | LM* | Load from Main Storage and Increment Address (reg 2) |
| 03 | 0000 | B | Branch |

**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2 V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Storage**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 1 0 0 | H1 | Reg 1 | 1 | W | C | D | V | Reg 2 |
| 0 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

LM = 418A Load from Main Storage and Increment Address (reg 2)

*Register 1 (Bits 5-7):* Selects an LSR, for the current interrupt level, that the main storage data will be written to.

*Register 2 (Bits 13-15):* Selects an LSR, for the current interrupt level, that contains the main storage address.

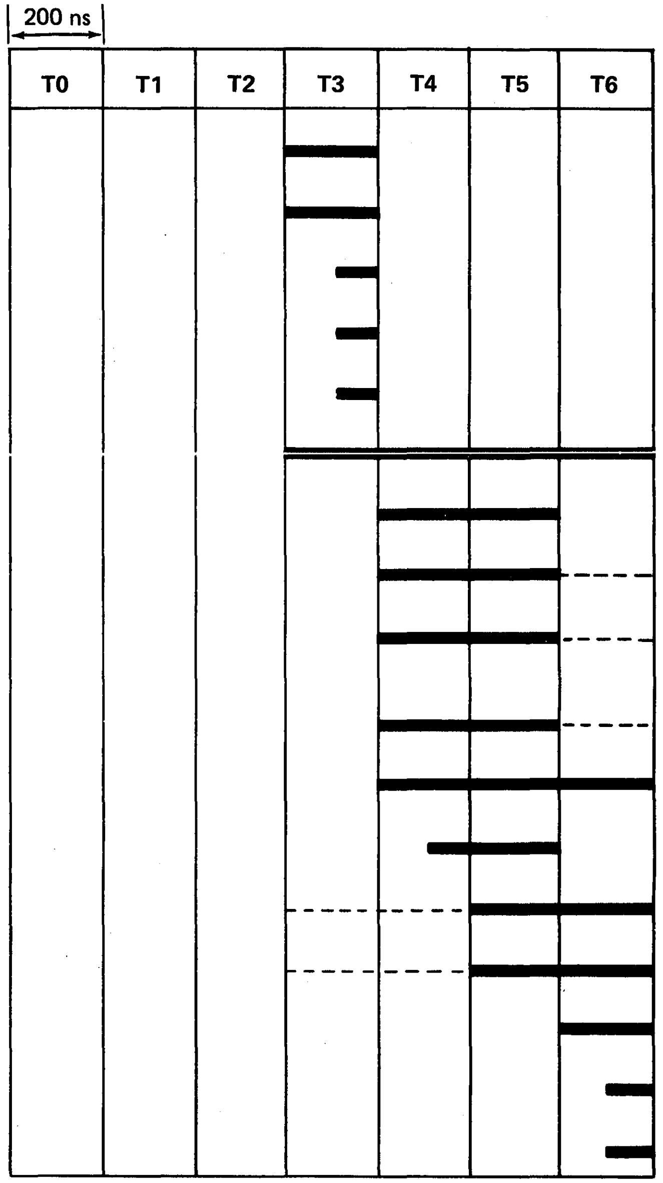
Steps 1-5 clock the main storage address from the selected LSR (reg 2) to MSAR.

Steps 6-10 increment the address (reg 2).

Steps 11-21 gate the main storage data to the selected LSR (reg 1).

### Store to Main Storage (STM)

| Step | | FSL Page |
|---|---|---|
| 1 | Select LSR (addr) (bits 13-15) | PC230 |
| 2 | Select Stg Gate Hi/Lo (from LSR hi/lo) | PC230 |
| 3 | Clock Stg Gate Check | PC146 |
| 4 | Clock X Hi, X Lo, SAR | PC210 |
| 5 | Clock MSAR | PC030 |
| 6 | Main Storage Op Trigger | PC030 |
| 7 | Select LSR (data) (bits 5-7) | PC240 |
| 8 | Select Stg Gate Hi (from LSR hi) | PC230 |
| 9 | Select Stg Gate lo (from LSR lo if bit 4=0, from stg gate hi if bit 4=1) | PC230 |
| 10 | Select MS Gate B (from SBO) | PM440 |
| 11 | Write Main Storage | PC030 |
| 12 | CSY Trigger New | PC030 |
| 13 | Set ALU Mode (X+carry) | PC230 |
| 14 | ALU Gate Hi/Lo (from ALU hi/lo) | PC250 |
| 15 | Select LSR (addr) (bits 13-15) | PC230 |
| 16 | Write LSR Hi/Lo (addr) | PC160 |
| 17 | Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | | |
|---|---|---|---|
| 00 | A155 | LI | Load Data into WR 1 (L) |
| 01 | AA01 | LI | Load MS Address into WR 2 |
| 02 | A200 | LI | |
| 03 | 41CA | STM* | Store to Main Storage and Increment Address (WR 2) |
| 04 | 0000 | B | Branch |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2 V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Storage**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 1 0 0 | H1 | Reg 1 | 1 | W | C | D | V | Reg 2 |
| 0 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

STM = 41CA Store to Main Storage and Increment Address (reg 2)

*Register 1 (Bits 5-7):* Selects an LSR, for the current interrupt level, that contains the data to be written into main storage.

*Register 2 (Bits 13-15):* Selects an LSR, for the current interrupt level, that contains the main storage address where register 1 will be written.

Steps 1-4 clock the main storage address from the selected LSR (reg 2) to MSAR.

Steps 5-9 gate the data from the selected LSR (reg 1) to main storage.

Steps 10-12 clock the data into main storage.

Steps 13-17 increment the address (reg 2).

This page intentionally left blank.

## Register Control (WMPR, RMPR)

WMPR (load main storage processor register)
RMPR (sense main storage processor register)

| 0 1 0 0 | H1 | Reg 1 | 1 | W | 0 | 1 | 0 | Reg 2 |
|---|---|---|---|---|---|---|---|---|
| 0 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

This instruction moves 1 byte of data between a local storage register and a main storage processor register.

*H1 (Bit 4):* Selects the low- or high-order byte of the local storage register specified by bits 5-7 (register 1):

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Note:* Specific main storage processor registers can be loaded only from the high-order byte.

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. Data is moved to or from a main storage processor register.

*Bit 8:* Changes the operation code (bits 0-3). Bit 8 is always a 1.

*W (Bit 9):* Identifies the direction the data is to be moved:

W = 0: Move the data from the selected main storage processor register to the selected local storage register

W = 1: Move the data from the selected local storage register to the selected main storage processor register

*Bit 10:* Bit 10 is always a 0.

*Bit 11:* Bit 11 is always a 1. Therefore, register 2 is always decreased by 1.

*Bit 12:* Bit 12 is always a 0. Bits 11 and 12 change the operation code (bits 0-3). For this instruction, register 2 is always decreased by 1.

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack used by the present interrupt level that contains the address of the main storage processor register to load data into or to read data from.

**Instruction List**

| Bits<br>4 | <br>8 | <br>9 | <br>10 | <br>11 | <br>12 | Mne-<br>monic | Description |
|---|---|---|---|---|---|---|---|
| H | 1 | 1 | 0 | 1 | 0 | WMPR | Load main storage processor register, decrease register 2 by 1. |
| H | 1 | 0 | 0 | 1 | 0 | RMPR | Sense main storage processor register, decrease register 2 by 1. |

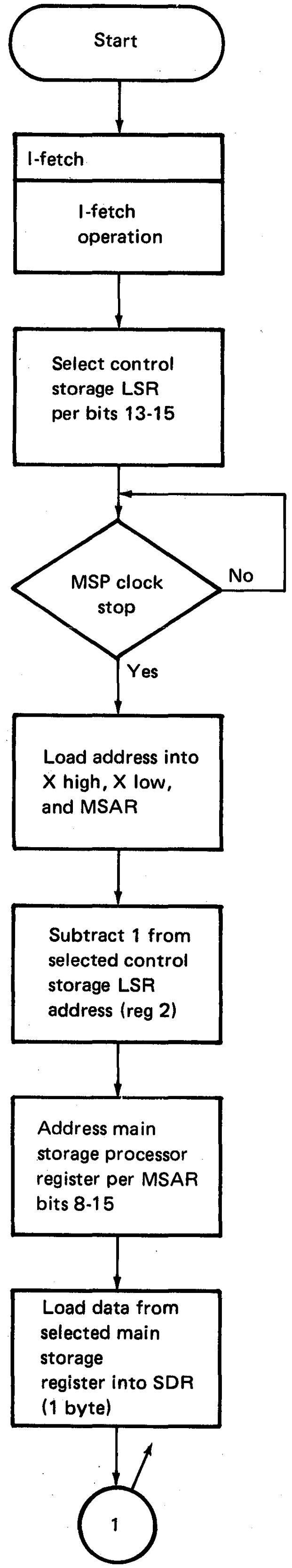
*Condition Code*

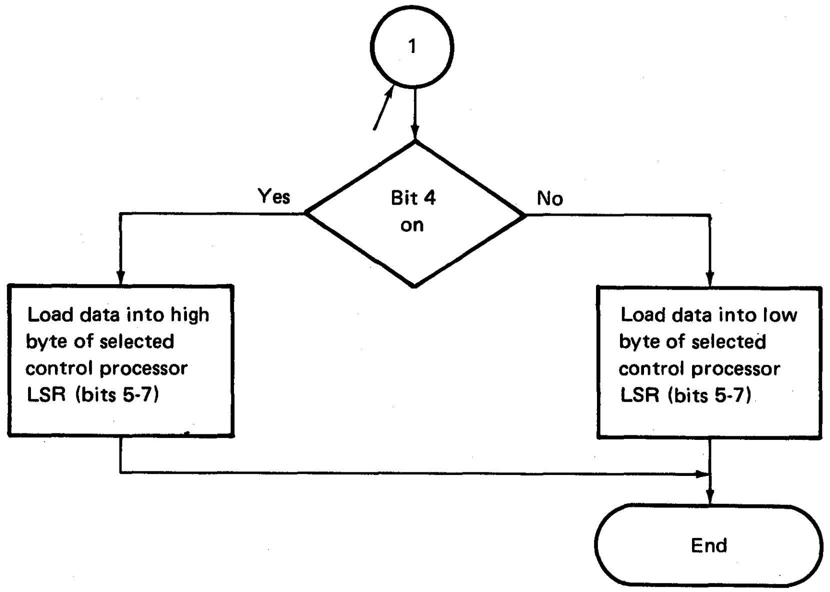
No change

| MSAR Low Byte | Bits 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Restrictions |
|---|---|---|---|---|---|---|---|---|---|
| Storage Control Commands | 0 | 0 | 1 | 1 | 1 | X | X | X | |
| CCR (configuration control register) | | | | | | 0 | 0 | 0 | Load high only[1] |
| ACR-Low (address compare register) | | | | | | 0 | 0 | 1 | Load high only[1] |
| ACR-High (address compare register) | | | | | | 0 | 1 | 0 | Load high only[1] |
| ACR-E (address compare extend) | | | | | | 0 | 1 | 1 | Load high only[1] |
| BMR (backup mode register) | | | | | | 1 | 0 | 0 | Load high only[1] |
| Status byte 3 | | | | | | 1 | 0 | 1 | Sense only |
| CMR (control mode register) | | | | | | 1 | 1 | 0 | Load high only[1] |
| PMR (program mode register) | | | | | | 1 | 1 | 1 | Load high only[1] |
| Status Registers | 0 | 1 | 0 | 0 | 0 | 0 | X | X | |
| PSR (program status register) | | | | | | | 0 | 0 | |
| Status byte 0 | | | | | | | 0 | 1 | Sense only |
| Status byte 1 | | | | | | | 0 | 1 | Load only |
| Status byte 2 | | | | | | | 1 | 0 | Sense only |
| Q-byte (real) | | | | | | | 1 | 1 | Sense only |
| Main Storage Processor Registers | 0 | 1 | 1 | 0 | X | X | X | X | |
| Operand 1 | | | | | 0 | 0 | 0 | H | H = 1 specifies |
| Operand 2 | | | | | 0 | 0 | 1 | H | the high byte |
| IAR (instruction address register) | | | | | 0 | 1 | 0 | H | |
| Q-register | | | | | 0 | 1 | 1 | 0 | |
| Op register | | | | | 0 | 1 | 1 | 1 | |
| XR1 | | | | | 1 | 0 | 0 | H | |
| XR2 | | | | | 1 | 0 | 1 | H | |
| ARR (address recall register) | | | | | 1 | 1 | 0 | H | |
| PSR (program status register address)<br>not a valid PSR | | | | | 1 | 1 | 1 | 0 | |
| LCRR (length count recall register)<br>R-byte if not executable | | | | | 1 | 1 | 1 | 1 | |
| Expanded ATRs | 1 | 0 | X | X | X | X | X | X | Load high only[1] |
| Task ATRs | 1 | 1 | 0 | X | X | X | X | X | Load high only[1] |
| I/O ATRs | 1 | 1 | 1 | X | X | X | X | X | Load high only[1] |

[1] Data is loaded into MSAR from the selected control processor LSR high byte only.

### Sense MSP Register


Start
I-fetch
I-fetch operation
Select control storage LSR per bits 13-15
MSP clock stop
No
Yes
Load address into X high, X low, and MSAR
Subtract 1 from selected control storage LSR address (reg 2)
Address main storage processor register per MSAR bits 8-15
Load data from selected main storage register into SDR (1 byte)
1


1
Bit 4 on
Yes
No
Load data into high byte of selected control processor LSR (bits 5-7)
Load data into low byte of selected control processor LSR (bits 5-7)
End

**Register Control**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | H1 | Reg 1 | 1 | W | 0 | 1 | 0 | Reg 2 |
| 0 | | | 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

RMPR = 4X9X Sense MSP Register

*Register 1 (Bits 5-7):* Selects an LSR, for the current interrupt level, that will store the data from the selected MSP register.

*Register 2 (Bits 13-15):* Selects an LSR, for the current interrupt level, that contains the address of the MSP register to be sensed.

Steps 1-5 clock the MSP register address (reg 2) to MSAR.

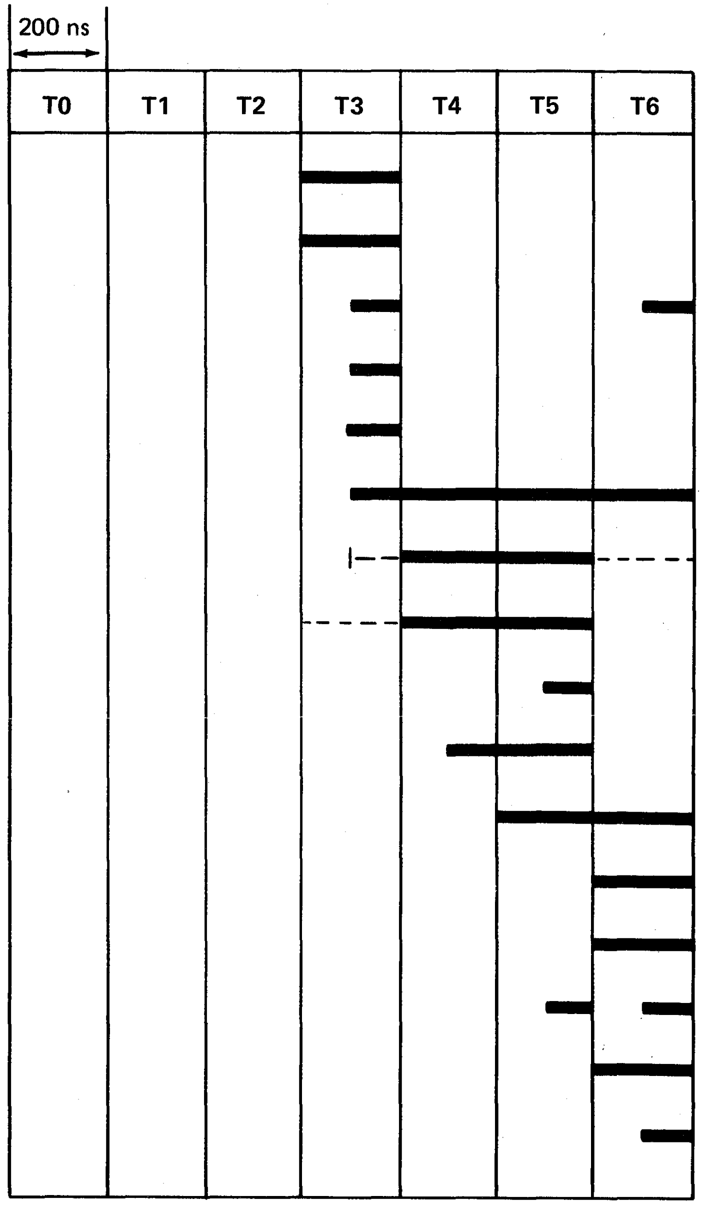
Steps 6-10 clock the contents of the selected MSP register into SDR.

Steps 7-9 decrement the address in the LSR (reg 2).

Steps 11-16 move the MSP register contents from the SDR to the selected LSR (reg 1).

**Sense from MSP Register**

| Step | | FSL Page |
|---|---|---|
| 1 | Select LSR (bits 13-15) | PC230 |
| 2 | Storage Gate Hi/Lo (from LSR hi/lo) | PC230 |
| 3 | Clock Stg Gate Check | PC146 |
| 4 | Clock X Hi, X Lo, SAR | PC210 |
| 5 | Clock MSAR | PC030 |
| 6 | Sense Load MSP Regs | PC030 |
| 7 | ALU Mode (X-1) | PC260 |
| 8 | ALU Gate Hi/Lo (from ALU hi/lo) | PC250 |
| 9 | Write LSR Hi/Lo (address) | PC160 |
| 10 | Clock SDR (CSY trigger) | PC220 |
| 11 | Select Stg Gate Hi/Lo (from SDR hi/lo) | PC230 |
| 12 | Select ALU Gate Lo (from stg gate lo) | PC250 |
| 13 | Select ALU Gate Hi (from ALU gate lo) | PC250 |
| 14 | Clock ALU Gate Check | PC160 |
| 15 | Select LSR (bits 5-7) | PC240 |
| 16 | Write LSR (hi if bit 4=1, lo if bit 4=0) (data) | PC230 |



**Instruction Loop**

| | | | |
|---|---|---|---|
| 00 | A238 | LI | Load MSP Reg Addr into WR 2 (L) Hex 38=CCR |
| 01 | 4192 | RMPR* | Read CCR into WR 1 (L) |
| 02 | 0000 | B | Branch |

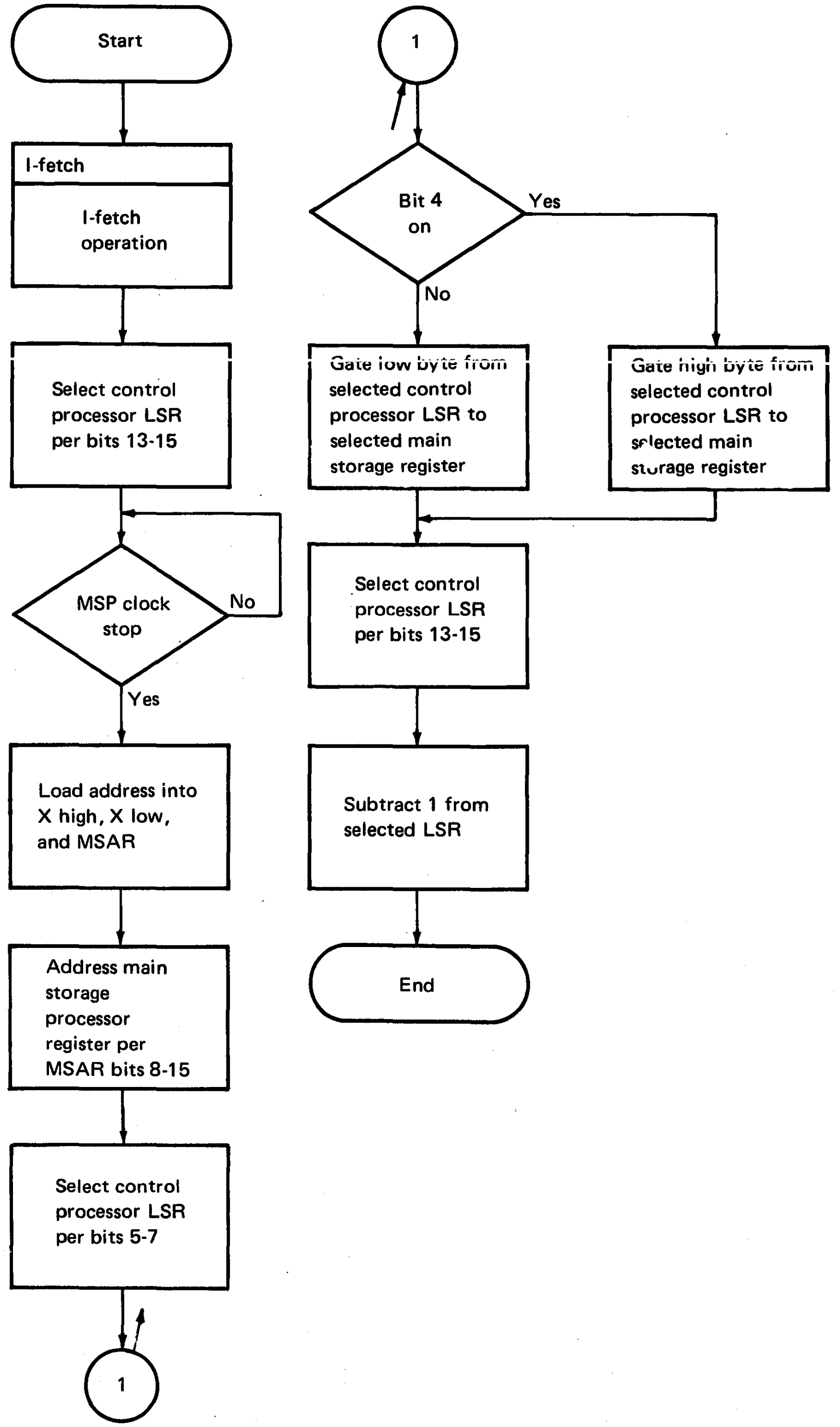
**Scope Setup**

Horizontal = 0.1 *µ*s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2 V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Load MSP Register


Start
I-fetch
I-fetch operation
Select control processor LSR per bits 13-15
MSP clock stop
No
Yes
Load address into X high, X low, and MSAR
Address main storage processor register per MSAR bits 8-15
Select control processor LSR per bits 5-7
1
1
Bit 4 on
Yes
No
Gate low byte from selected control processor LSR to selected main storage register
Gate high byte from selected control processor LSR to selected main storage register
Select control processor LSR per bits 13-15
Subtract 1 from selected LSR
End

### Register Control

| 0 | 1 | 0 | 0 | H1 | Reg 1 | 1 | W | 0 | 1 | 0 | Reg 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 3 | 4 | 5 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

WMPR = 4XDX Load MSP Register

*Register 1 (Bits 5-7):* Selects an LSR, for the current interrupt level, that contains the data to be sent to the selected MSP register.

*Register 2 (Bits 13-15):* Selects an LSR, for the current interrupt level, that contains the address of an MSP register to be loaded.

Steps 1-5 clock the MSP register address from the LSR (reg 2) to MSAR.

Steps 6-10 gate data from the LSR (reg 1) to the selected MSP register.

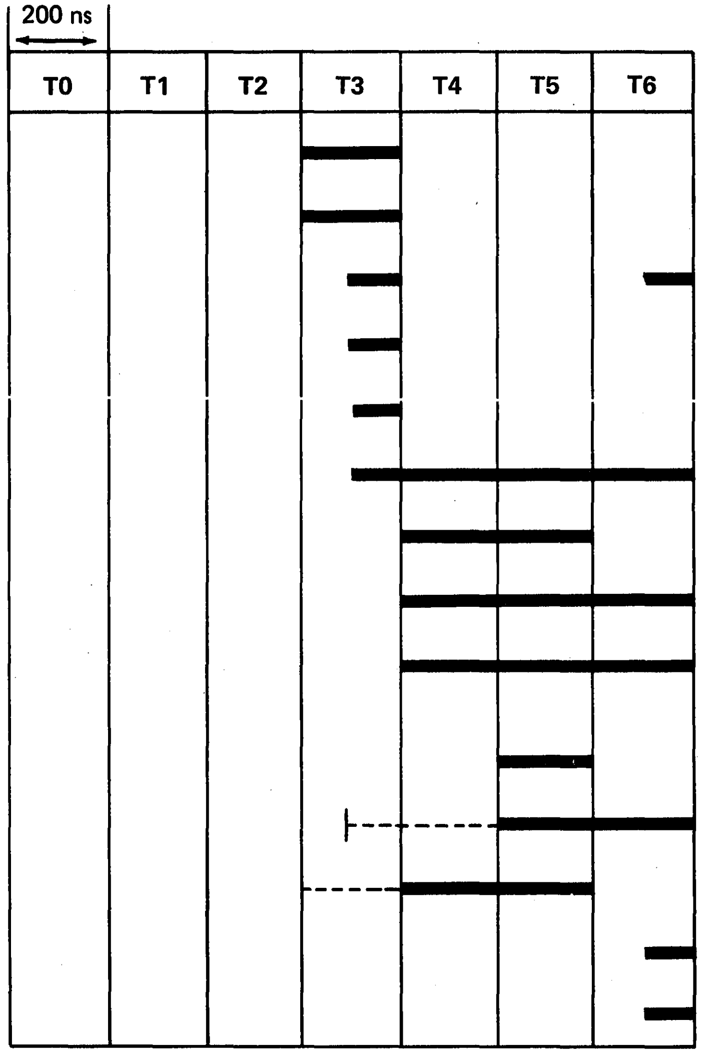
Steps 11-14 decrement the MSP register address (reg 2).

Data to be written to the following MSP registers must be written from reg 1 high:

1. ACR (Hi, Lo, or Ext)
2. CCR
3. BMR
4. CMR
5. PMR
6. ATRs

### Load MSP Register

| Step | | FSL Page |
|---|---|---|
| 1 | Select LSR Hi/Lo (bits 13-15) | PC230 |
| 2 | Select Stg Gate Hi/Lo (from LSR hi/lo) | PC230 |
| 3 | Clock Stg Gate Check | PC146 |
| 4 | Clock X Hi, X Lo, SAR | PC210 |
| 5 | Clock MSAR | PC030 |
| 6 | Sense Load MSP Reg | PC030 |
| 7 | Select LSR (bits 5-7) | PC240 |
| 8 | Select Stg Gate Hi (from LSR hi) | PC230 |
| 9 | Select Stg Gate Lo (from LSR lo if bit 4=0, from stg gate hi if bit 4=1) | PC230 |
| 10 | Write MSP Registers | PC030 |
| 11 | ALU Mode (X-1) | PC260 |
| 12 | ALU Gate Hi/Lo (from ALU hi/lo) | PC250 |
| 13 | Clock ALU Gate Check | PC160 |
| 14 | Write LSR Hi/Lo (address) | PC160 |



**Instruction Loop**

| | | | |
|---|---|---|---|
| 00 | A907 | LI | Load Data into WR 1 (H) (reg 1) |
| 01 | A238 | LI | Load MSP Reg Addr into WR 2 (L) (reg 2) Hex 38=CCR |
| 02 | 49D2 | WMPR* | Write to MSP Reg (CCR) |
| 03 | 0000 | B | Branch |

**Scope Setup**

Horizontal = 0.1 $\mu$s/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2 V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Storage Direct (L, ST)

L (load register)
ST (store register)

| | | | | |
|---|---|---|---|---|
| 1 1 1 0 | W | Reg 1 | 0 | SAR |
| 0 3 | 4 | 5 7 | 8 | 9 15 |

This instruction has direct access to any of the first 128 addresses in the current 4K-word block of addresses of control storage (the fixed storage area) during read or write operations, and moves 2 bytes of data to or from control storage.

*W (Bit 4):* Indicates if a read or write operation is to occur:

W = 0: Read from control storage to the selected register

W = 1: Write to control storage using the selected register for source

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. Moves 2 bytes of data between this register and control storage.

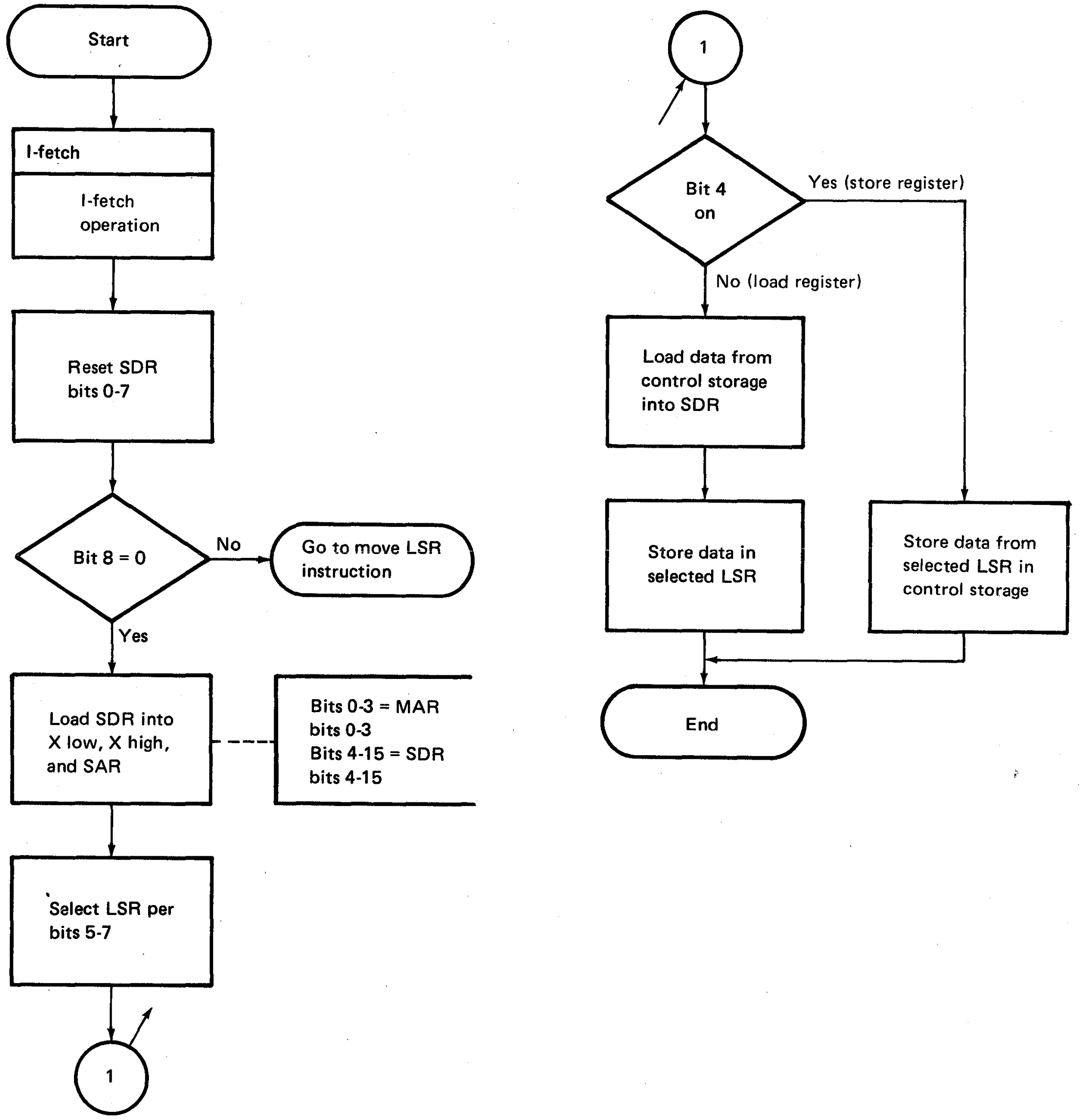
*Bit 8:* Changes the operation code (bits 0-3). Bit 8 is always a 0.

*Storage Address Register (Bits 9-15):* Specifies one of the first 128 locations of the 4K-word block in control storage in which the current instruction is loaded. These 7 bits replace the comparable 7 bits in the storage address register. Bits 4 through 8 of the storage address register are set to zero. Bits 0-3 are left as is and point to the current 4K-word block of addresses.

*Condition Code*

No change

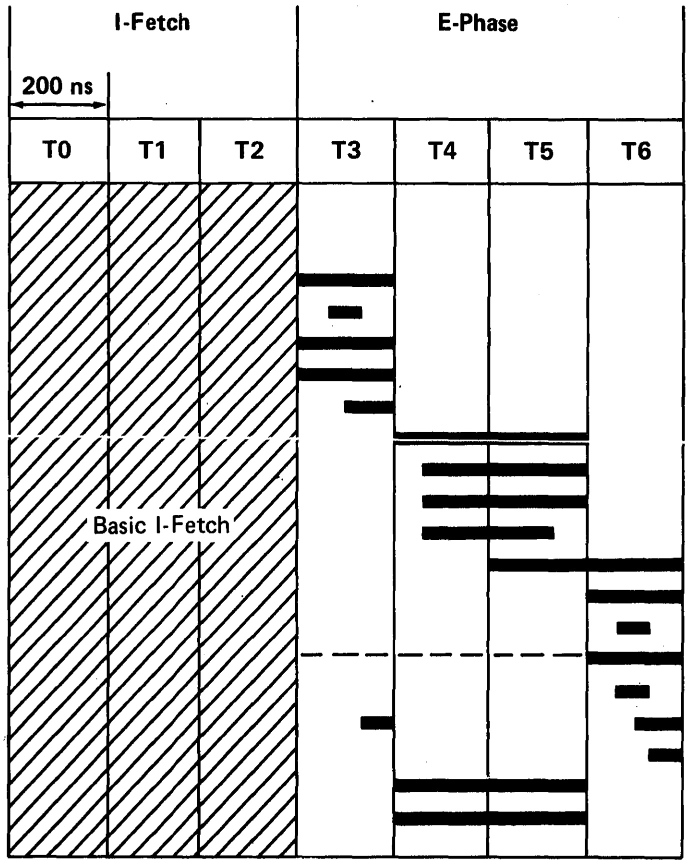
*Sequence and Timing for:*
*Reading from Control Storage—L (load register)*
*Writing into Control Storage—ST (store register)*


Start
I-fetch
I-fetch operation
Reset SDR bits 0-7
Bit 8 = 0
No
Go to move LSR instruction
Yes
Load SDR into X low, X high, and SAR
Bits 0-3 = MAR bits 0-3
Bits 4-15 = SDR bits 4-15
Select LSR per bits 5-7
1
1
Bit 4 on
Yes (store register)
No (load register)
Load data from control storage into SDR
Store data in selected LSR
Store data from selected LSR in control storage
End

**L (load register)**

**Hex E17F**

| | FSL Page |
|---|---|
| +Reset SDR High A1H2P07 | PC220 |
| Clock X High (MAR data from T0) | PC210 |
| Stg Gate High from X (0-3) SDR (4-7) | PC230 |
| Clock SAR from Stg Gate High/Low | |
| Stg Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| Clock SDR | PC220 |
| Stg Gate High/Low from SDR High/Low | PC230 |
| ALU Gate High/Low from Stg Gate High/Low | PC250 |
| Write LSR High/Low from ALU Gate High/Low | PC160 |
| Select LSR (data) | PC230 |
| Clock SDR Check | PC220 |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |
| Ctl Stg Address Check | |
| Ctl Stg SAR P Check | |



[1] This line cannot be probed.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | E17F | L * |
| 02 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

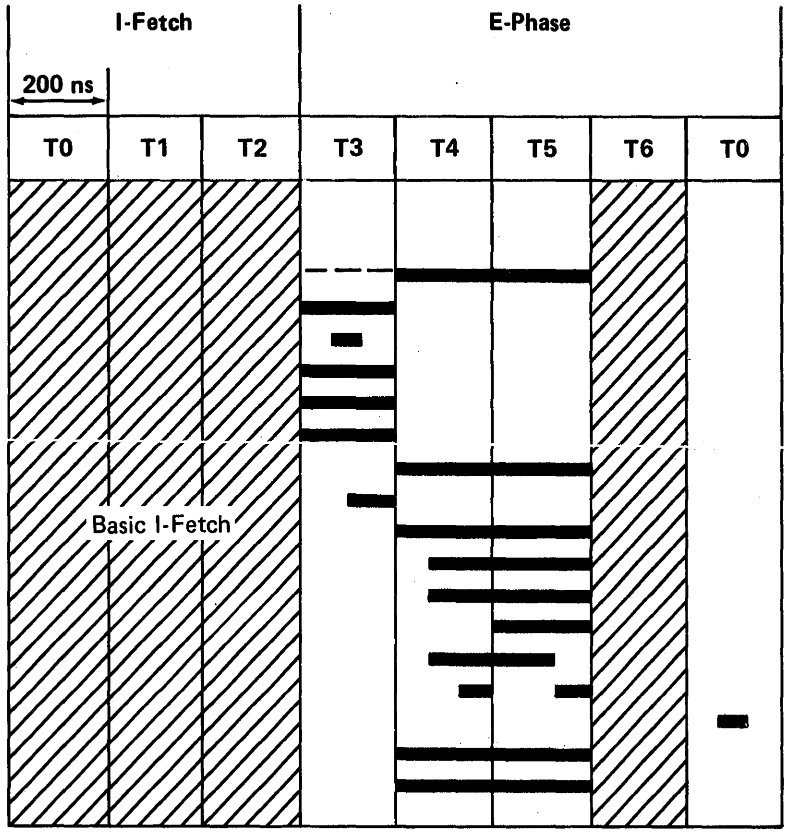
Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**ST (store register)**

**Hex E97F**

| | FSL Page |
|---|---|
| Select LSR (data) | PC230 |
| Reset SDR High A1-H2 P07 | PC220 |
| Clock X High (MAR data from T0) | PC210 |
| Stg Gate High from X (0-3) SDR (4-7) | PC230 |
| Stg Gate Low from SDR (8-15) | PC230 |
| Clock SAR from Stg Gate High/Low | |
| Stg Gate High/Low from LSR | PC230 |
| Stg Function | PC012 |
| Storage Cycle[1] | PC012 |
| CSX | PC020 |
| CSY | PC020 |
| CS Write Pulse High/Low | |
| Clock SDR (echo check) | PC146 |
| Clock Stg Gate Check | PC160 |
| Clock SDR Check (see note) | |
| Ctl Stg Address Check | |
| Ctl Stg SAR P Check | |



[1] This line cannot be probed.

*Note:* SDR check from this instruction is actually set at T0 of the next instruction.

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A155 | LI |
| 01 | A955 | LI |
| 02 | E97F | ST * |
| 03 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Move Local Storage Register (MVR)

| | | | | |
|---|---|---|---|---|
| 1 1 1 0 | Reg 1 | 1 | S | Reg 2 |
| 0 3 | 4 7 | 8 | 9 | 10 15 |

This instruction moves 2 bytes of data from one local storage register to another local storage register. This instruction permits access to any of the 64 local storage registers in the stack. For example, data can be moved either from register 1 to register 2 or from register 2 to register 1; data movement is controlled by the setting of bit 9.

*Register 1 (Bits 4-7):* Selects one of 16 local storage registers. The group selected is determined by the present interrupt level. Eight of these registers are always the microaddress register or the microaddress backup register stack (specified by bit 4 = 1). The other eight local storage registers are the work registers associated with the interrupt level selected. These registers are selected by specifying 0-7 in the register 1 field and then selecting from a group of eight registers assigned to each interrupt level (1-5) or the main program level interrupt. Hardware automatically selects stack 1 or stack 2 because of the interrupt level. Hardware then adds hex 00 or hex 10 for stack 1 and hex 20 or hex 30 for stack 2 to the register 1 bits to come up with the real hex address of the local storage register selected.

*Bit 8:* Changes the operation code (bits 0-3). Bit 8 is always a 1.

*S (Bit 9):* Indicates the direction the data is to be moved:

S = 0: Register 1 is the source register and 2 bytes of data are moved from register 1 to register 2.

S = 1: Register 2 is the source register and 2 bytes of data are moved from register 2 to register 1.

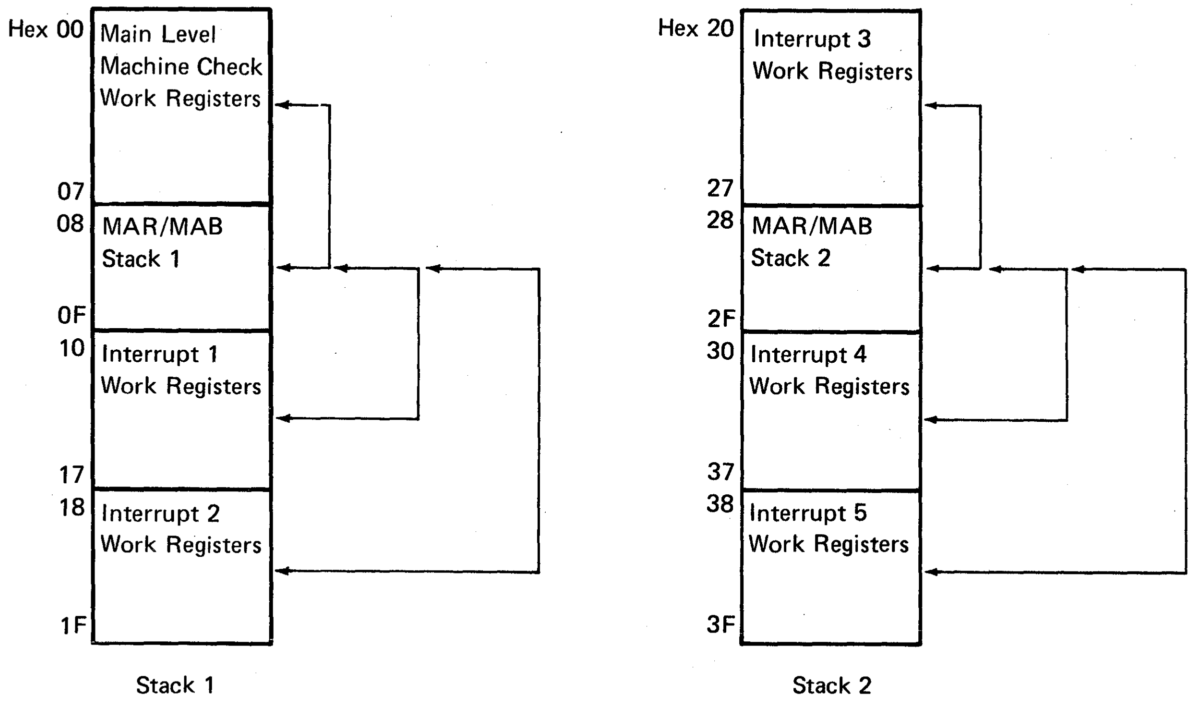
*Register 2 (Bits 10-15):* The 6 bits of this field select one of the 64 local storage registers in the data flow (bit 10 = 0). Two bytes of data are moved to or from this field, as determined by the bit setting of the S field.

*Condition Code*

No change

### Control Processor Local Storage Registers

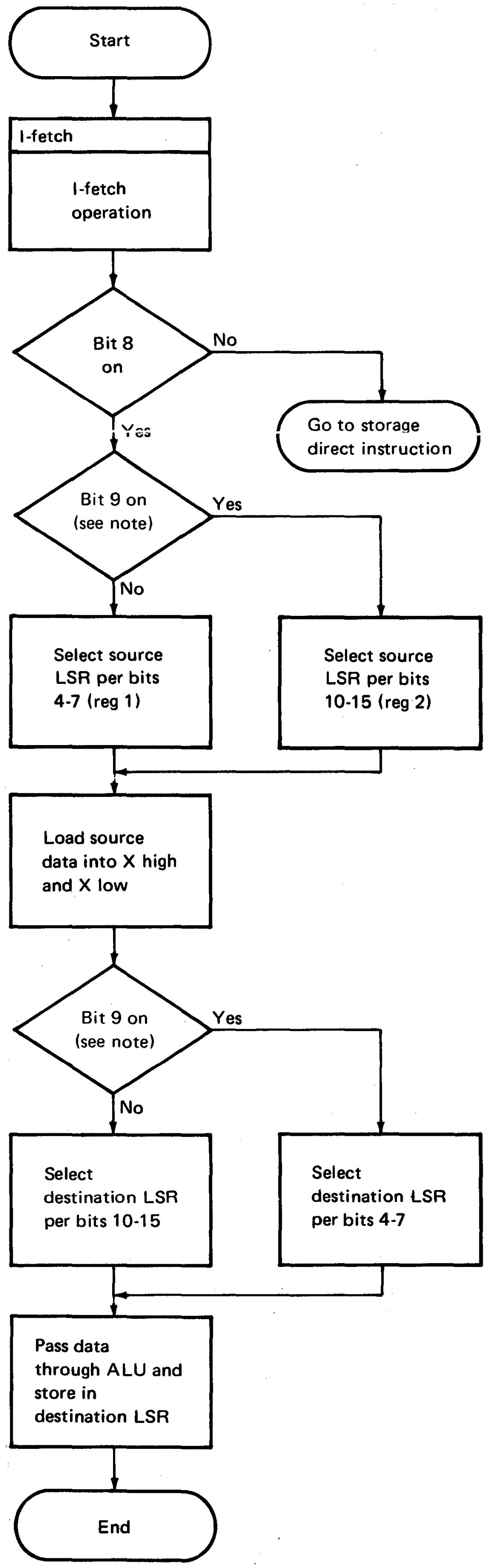
**Valid Field Register Specifications**



Valid combinations of local storage registers that can be specified in the register 1 field of the move local storage registers are:

- If in main level or machine check, registers 00-07 and 08-0B can be specified.
- If in interrupt level 1, registers 10-17 and 0C-0D can be specified.
- If in interrupt level 2, registers 18-1F and 0E-0F can be specified.
- If in interrupt level 3, registers 20-27 and 28-29 can be specified.
- If in interrupt level 4, registers 30-37 and 2C-2D can be specified.
- If in interrupt level 5, registers 38-3F and 2E-2F can be specified.

*Sequence and Timing*



Start
I-fetch
I-fetch operation
Bit 8 on
No
Go to storage direct instruction
Yes
Bit 9 on (see note)
Yes
No
Select source LSR per bits 4-7 (reg 1)
Select source LSR per bits 10-15 (reg 2)
Load source data into X high and X low
Bit 9 on (see note)
Yes
No
Select destination LSR per bits 10-15
Select destination LSR per bits 4-7
Pass data through ALU and store in destination LSR
End

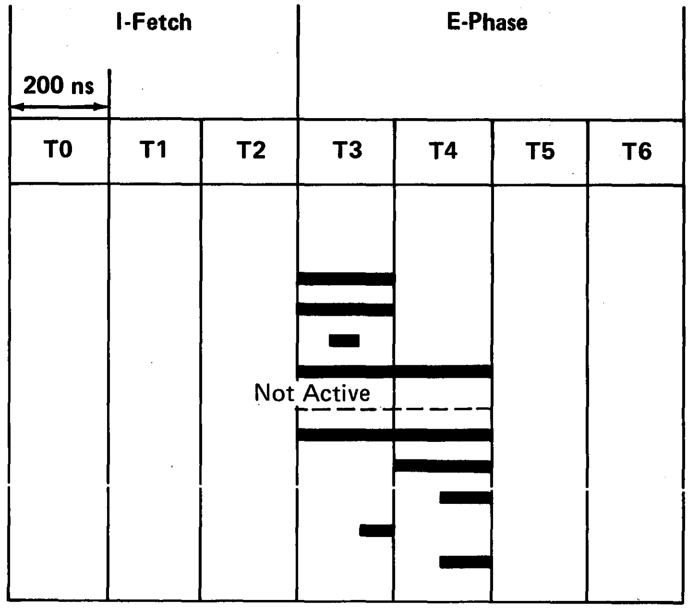
*Note:* Bit 9 determines the source field:

If bit 9 = 0, bits 4-7 specify the source LSR; bits 10-15 specify the destination LSR.

If bit 9 = 1, bits 10-15 specify the source LSR; bits 4-7 specify the destination LSR.

**Hex E182**

| | FSL Page |
|---|---|
| Select LSR (source) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Set ALU Mode (X + carry) | PC260 |
| Carry | |
| ALU Gate High/Low from ALU High/Low | PC250 |
| Select LSR (destination) | PC230 |
| Write LSR High/Low | PC160 |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A1FF | LI |
| 01 | A9FF | LI |
| 02 | E182 | MVR * |
| 03 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## Hexadecimal Branch (HBN, HBZ)

HBN (numeric)
HBZ (zone)

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 1 1 1 | H1 | Reg 1 | MAR' | 0 | ▨ | Z |
| 0 3 | 4 | 5 7 | 8 11 | 12 | 13 14 | 15 |

This instruction operates as a 16-way branch without prerequisites. Either the zone or digit part of the high- or low-order byte of the selected register replaces bits 12-15 of the control storage microaddress register. Bits 8-11 of the control storage microaddress register are replaced by the bits contained in the hexadecimal branch instruction (bits 8-11).

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used in the hexadecimal branch:

H1 = 0: Low-order byte

H1 = 1: High-order byte

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The zone or digit part of the selected register replaces bits 12-15 of the control storage microaddress register.

*MAR' (Bits 8-11):* Replaces bits 8-11 of the control storage microaddress register. Bits 0-7 of the control storage microaddress register are not changed by this instruction.

*Bit 12:* Changes the operation code (bits 0-3). Bit 12 is always a 0.

*Bits 13 and 14:* Not used in this instruction.

*Z (Bit 15):* Causes either the zone or numeric part of the selected register to be used in the hexadecimal branch function:

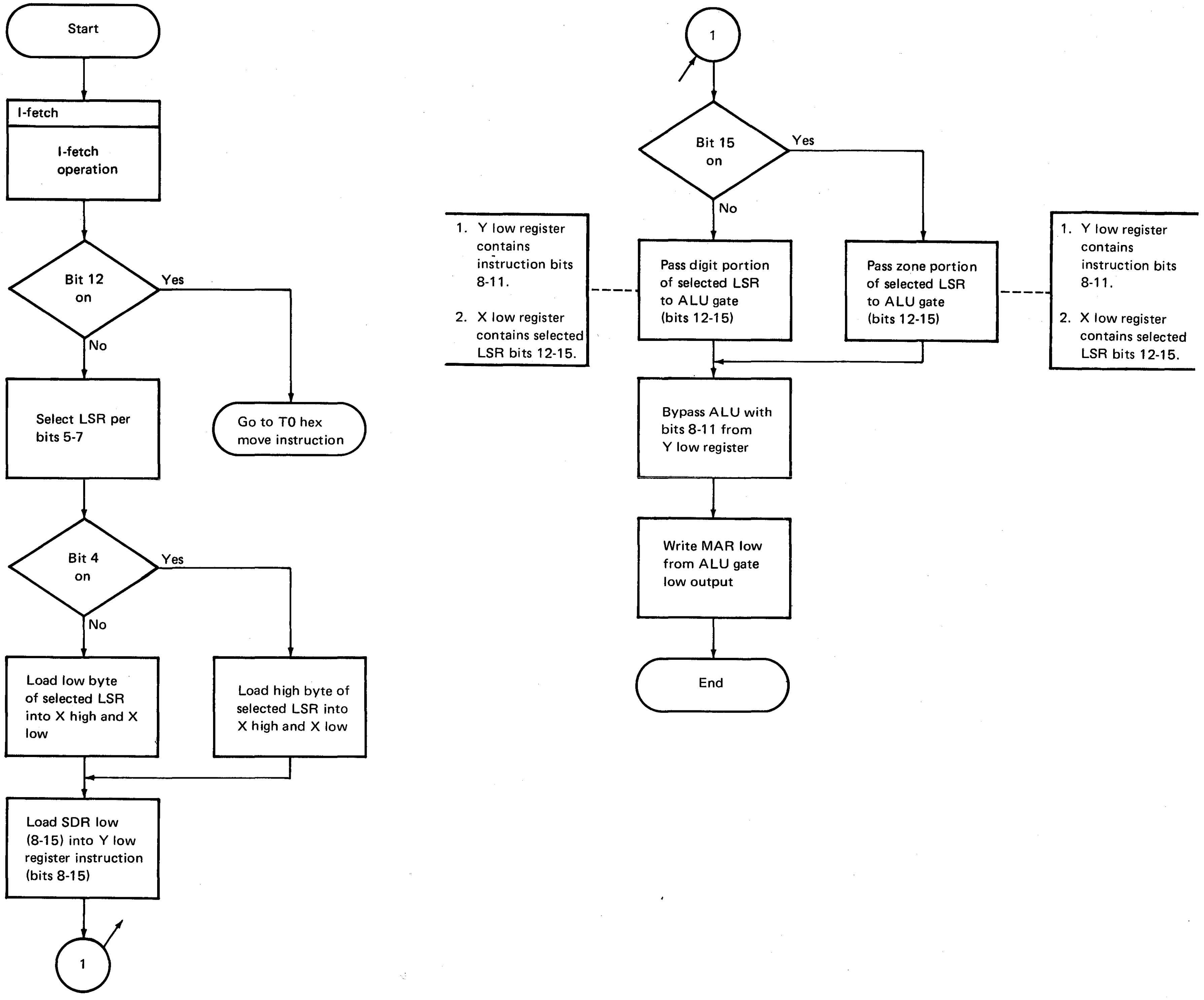
Z = 0: The numeric part of the data byte of the selected register replaces bits 12-15 of the control storage microaddress register.

Z = 1: The zone part of the data byte of the selected register replaces bits 12-15 of the control storage microaddress register.

*Condition Code*

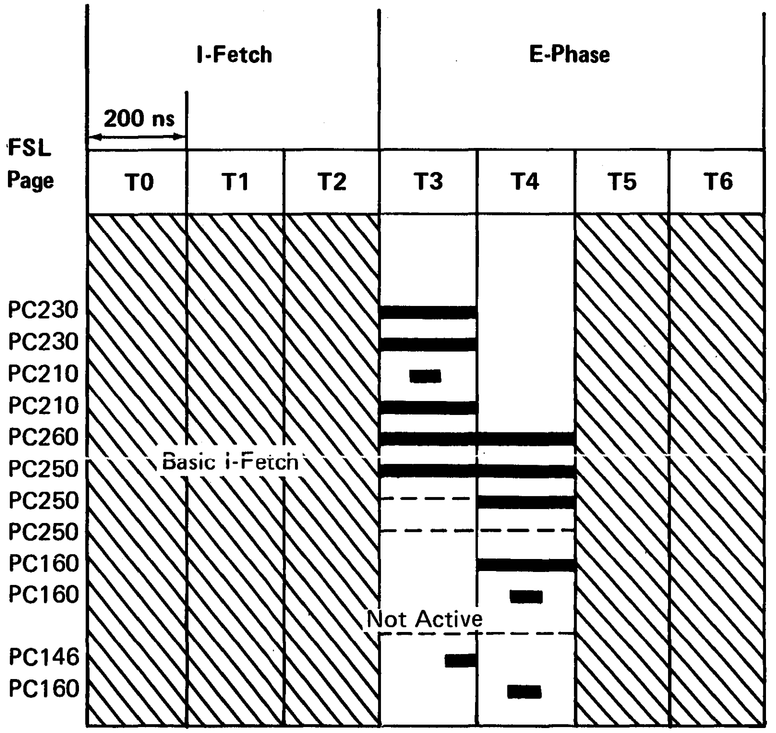
No change

*Sequence and Timing*


Start
I-fetch
I-fetch operation
Bit 12 on
Yes
No
Go to T0 hex move instruction
Select LSR per bits 5-7
Bit 4 on
Yes
No
Load low byte of selected LSR into X high and X low
Load high byte of selected LSR into X high and X low
Load SDR low (8-15) into Y low register instruction (bits 8-15)
1
1
Bit 15 on
Yes
No
1. Y low register contains instruction bits 8-11.
2. X low register contains selected LSR bits 12-15.
Pass digit portion of selected LSR to ALU gate (bits 12-15)
Pass zone portion of selected LSR to ALU gate (bits 12-15)
1. Y low register contains instruction bits 8-11.
2. X low register contains selected LSR bits 12-15.
Bypass ALU with bits 8-11 from Y low register
Write MAR low from ALU gate low output
End

**Hex F100**

Select LSR
Stg Gate High/Low from LSR
Clock X-Reg from Stg Gate High/Low
Clock Y Low from SDR Low (Y high, don't care)
Set ALU Mode (X + carry)
ALU Gate Low (8-11) from Y Low (8-11)
ALU Gate Low (12-15) from ALU Low (12-15)
ALU Gate High from ALU High
Select LSR (MAR)
Write LSR Low
Carry
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A103 | LI |
| 01 | F100 | HBN * |
| 02 | BEA3 | Proc |
| 03 | 0000 | B |

**Scope Setup**

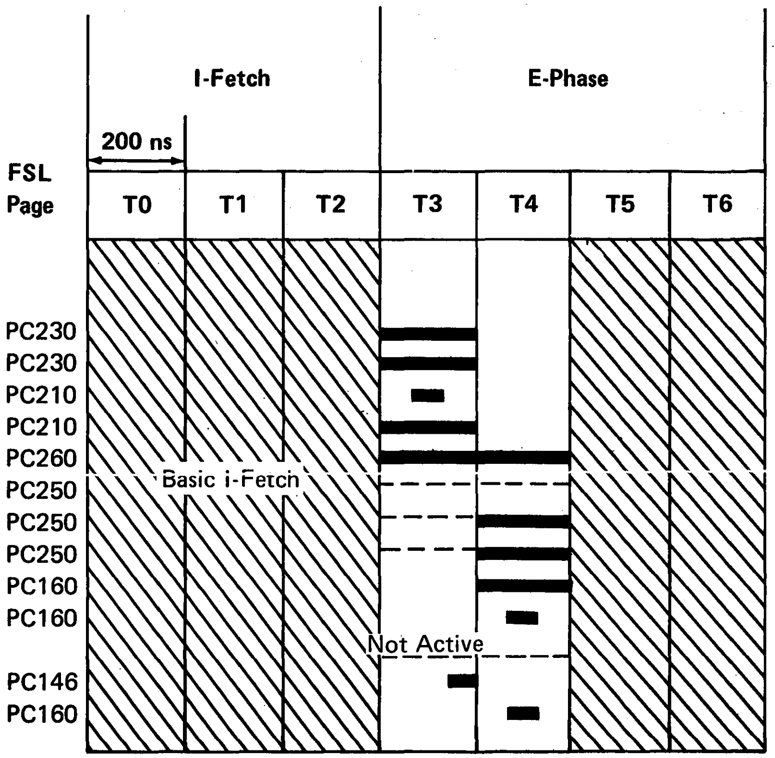
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Hex F101**

Select LSR
Stg Gate High/Low from LSR High/Low
Clock X-Reg from Stg Gate High/Low
Clock Y Low from SDR Low (Y high, don't care)
Set ALU Mode (X + carry)
ALU Gate High from ALU High
ALU Gate Low (8-11) from Y Low (8-11)
ALU Gate Low (12-15) from ALU Low (8-11)
Select LSR (MAR)
Write LSR Low
Carry
Clock Stg Gate Check
Clock ALU Gate Check



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A103 | LI |
| 01 | F101 | HBZ * |
| 02 | BEA3 | Proc |
| 03 | 0000 | B |

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

## Hexadecimal Move (SRL, SRLD, MZZ, MZN)

| | |
|---|---|
| SRL | (shift right logical) |
| SRLD | (shift right logical double) |
| MZZ | (link register 2 zone to register 1 numeric) |
| MZN | (link register 2 zone to register 1 zone) |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 1 1 1 | H1 | Reg 1 | ▨ | Function | H2 | 1 | Reg 2 |
| 0 3 | 4 | 5 7 | 8 | 9 10 | 11 | 12 | 13 15 |

This instruction performs the following functions:

- Shift right logical (SRL) 8 bits of register (register 1).

- Shift right logical double (SRLD) 16 bits of register (register 1).

- Link the zone part of register 2 to the numeric part of register 1 (MZZ) and put the results into register 1 in the following format:

| | |
|---|---|
| Reg2<br>Zone | Reg1<br>Numeric |

- Link the zone part of register 2 to the zone part of register 1 (MZN) and put the results into register 1 in the following format:

| | |
|---|---|
| Reg2<br>Zone | Reg1<br>Zone |

*H1 (Bit 4):* Indicates which byte of the selected register in the local storage register stack is to be used:

H1 = 0: Low-order byte

H1 = 1: High-order byte

The H1 field is not used for shift-right-logical-double instructions.

*Register 1 (Bits 5-7):* Selects one of the eight work registers in the local storage register stack for the current interrupt level.

*Bit 8:* Not used in this instruction.

*Function (Bits 9 and 10):* Specifies one of the following functions:

- Bits 9 and 10 = binary 00: Register 1 shift right logical (SRL). The 8 bits of the selected byte are moved one position to the right. The high-order (leftmost) bit is replaced with a 0. The register 2 and H2 fields of the hexadecimal move instruction are not used for shift-right-logical functions.

- Bits 9 and 10 = binary 01: Register 1 shift right logical double (SRLD). The 16 bits of the selected register are moved one position to the right. The high-order bit (bit 0) is replaced with a 0. The H1, H2, and register 2 fields of the hexadecimal move instruction are not used for shift-right-logical-double functions.

- Bits 9 and 10 = binary 10: Link the zone part of register 2 to the zone part of register 1 (MZN). The zone digit of the register specified in register 2 is moved to the zone position of the register specified by register 1, and the zone digit of the register specified in register 1 is moved to the numeric position of the register specified in register 1. The results are put in the register specified by register 1 and have the following format:

| | |
|---|---|
| Reg2<br>Zone | Reg1<br>Zone |

| | | |
|---|---|---|
| Example: | Register 1 | 0110 1000 |
| | Register 2 | 1111 0010 |
| | Result | 1111 0110 |

- Bits 9 and 10 = binary 11: Link the zone part of register 2 to the numeric part of register 1 (MZZ). The zone digit of the register specified in register 2 is moved to the zone position of the register specified in register 1, and the numeric digit of the register specified by register 1 remains the same. The results are put in the register specified by register 1 and have the following format:

| | |
|---|---|
| Reg2<br>Zone | Reg1<br>Numeric |

| | | |
|---|---|---|
| Example: | Register 1 | 0110 1001 |
| | Register 2 | 1111 0010 |
| | Result<br>(register 1) | 1111 1001 |

*H2 (Bit 11):* Indicates which byte of the selected register (specified by register 2) in the local storage register stack is to be used:

H2 = 0: Low-order byte

H2 = 1: High-order byte

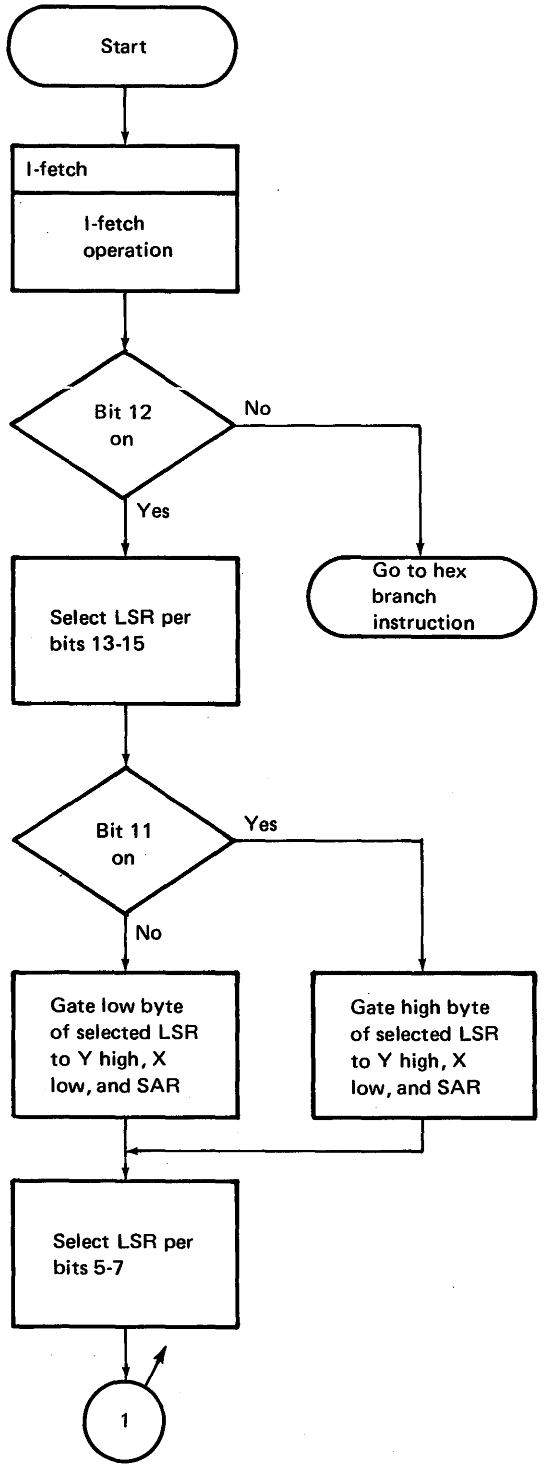
The H2 field is not used in the shift-right-logical and shift-right-logical-double functions.

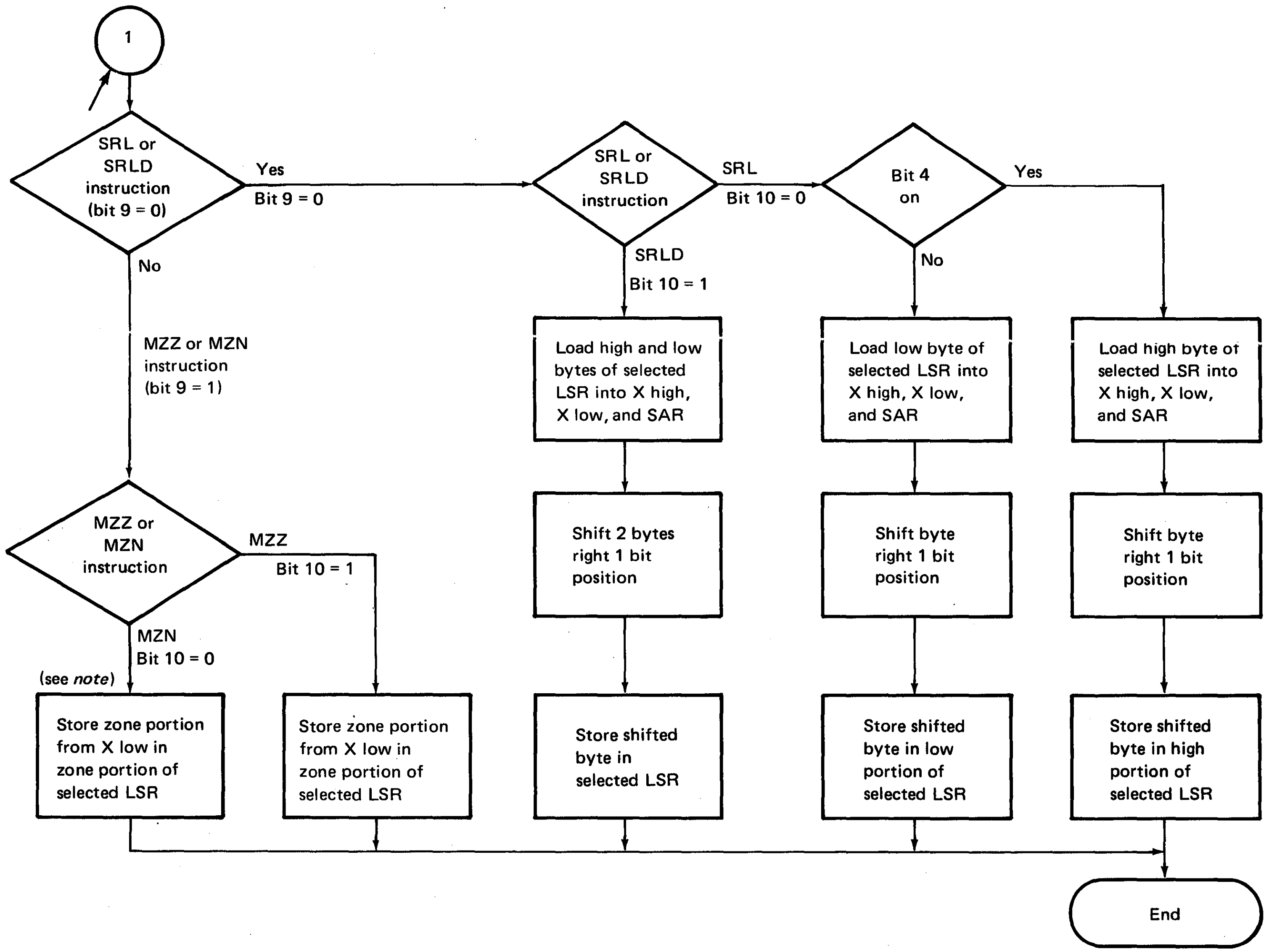
*Bit 12:* Changes the operation code (bits 0-3). Bit 12 is always a 1.

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The register 2 field is not used in the shift-right-logical and shift-right-logical-double functions.

*Condition Code*

No change


Start
I-fetch
I-fetch operation
Bit 12 on
No
Yes
Go to hex branch instruction
Select LSR per bits 13-15
Bit 11 on
Yes
No
Gate low byte of selected LSR to Y high, X low, and SAR
Gate high byte of selected LSR to Y high, X low, and SAR
Select LSR per bits 5-7
1


1
SRL or SRLD instruction (bit 9 = 0)
Yes
Bit 9 = 0
No
MZZ or MZN instruction (bit 9 = 1)
SRL or SRLD instruction
SRL
Bit 10 = 0
SRLD
Bit 10 = 1
Bit 4 on
Yes
No
Load high and low bytes of selected LSR into X high, X low, and SAR
Load low byte of selected LSR into X high, X low, and SAR
Load high byte of selected LSR into X high, X low, and SAR
MZZ or MZN instruction
MZZ
Bit 10 = 1
MZN
Bit 10 = 0
(see *note*)
Shift 2 bytes right 1 bit position
Shift byte right 1 bit position
Shift byte right 1 bit position
Store zone portion from X low in zone portion of selected LSR
Store zone portion from X low in zone portion of selected LSR
Store shifted byte in selected LSR
Store shifted byte in low portion of selected LSR
Store shifted byte in high portion of selected LSR
End

*Note:* The zone portion of the selected LSR is also placed in the numeric portion of the selected LSR.

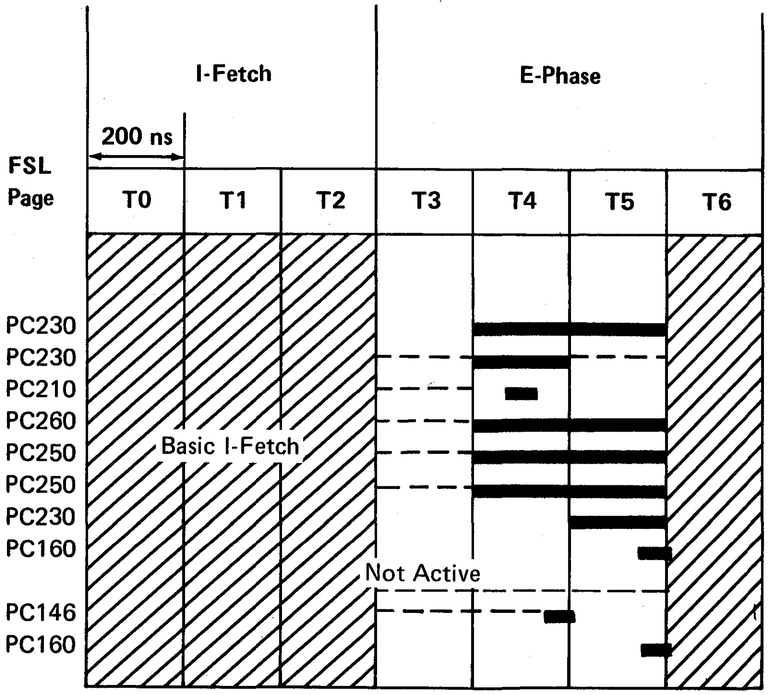
**Hex F108**

Select LSR
Stg Gate High/Low from LSR
X-Reg from Stg Gate High/Low
Set ALU Mode (X + carry)
ALU Gate Low (9-15) from ALU Low (8-14)
ALU Gate Low (8) from ALU Low P Gen (0 bit)
Select LSR (MAR)
Write LSR Low
Carry
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A155 | LI |
| 01 | A000 | LI |
| 02 | F108 | SRL * (see note) |
| 03 | 0000 | B |

*Note:* This instruction uses the low byte of the LSR.



**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

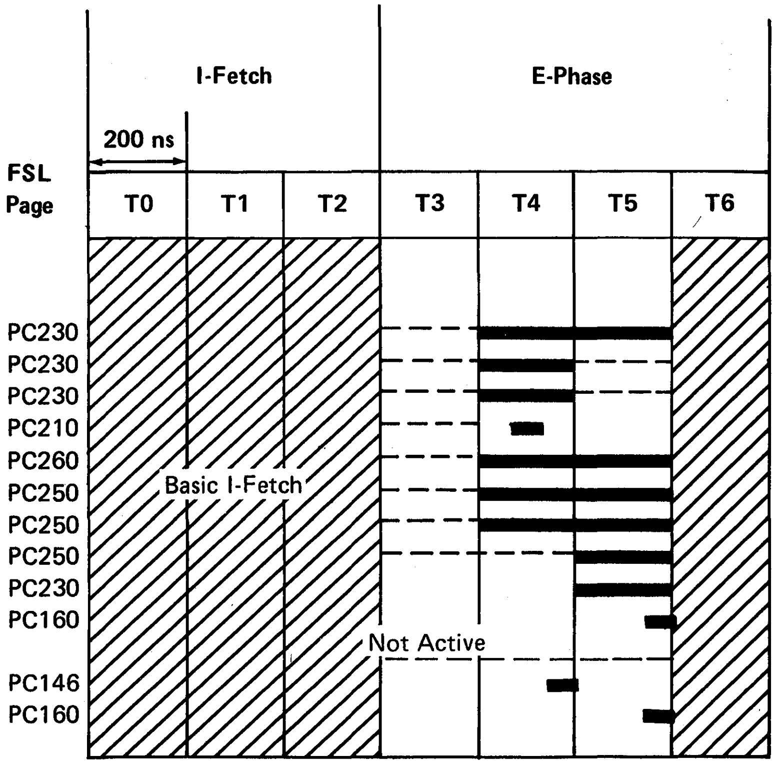
**Hex F908**

Select LSR
Stg Gate High from LSR
Stg Gate Low from Stg Gate High
X-Reg from Stg Gate High/Low
Set ALU Mode (X + carry)
ALU Gate Low (9-15) from ALU Low (8-14)
ALU Gate Low (8) from ALU Low P Gen (0 bit)
ALU Gate High from ALU Gate Low
Select LSR (MAR)
Write LSR High
Carry
Clock Stg Gate Check
Clock ALU Gate Check

**Instruction Loop**

| | | |
|---|---|---|
| 00 | A155 | LI |
| 01 | A900 | LI |
| 02 | F908 | SRL * (see note) |
| 03 | 0000 | B |

*Note:* This instruction uses the high byte of the LSR.



**Scope Setup**

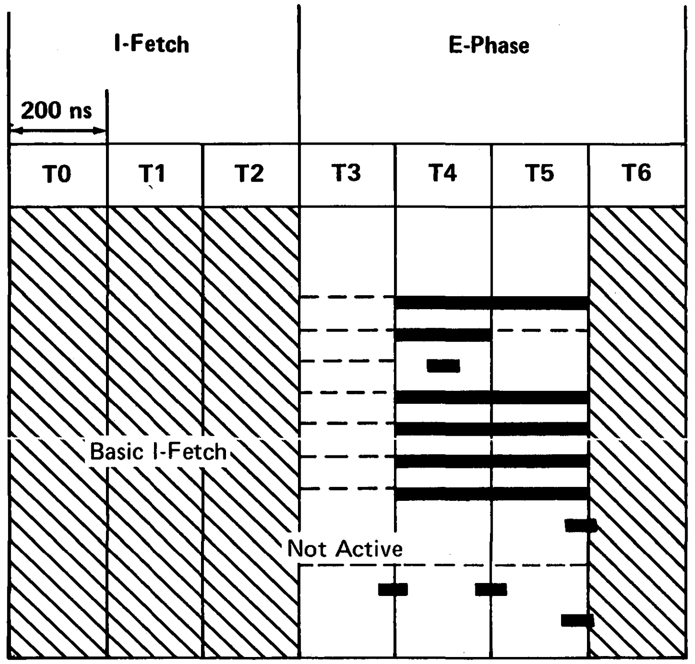
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

**Hex F128**

| | FSL Page |
|---|---|
| Select LSR | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Set ALU Mode (X + carry) | PC260 |
| ALU Gate Low (9-15) from ALU Low (8-14) | PC250 |
| ALU Gate Low (8) from ALU High (7) | PC250 |
| ALU Gate High (1-7) from ALU High (0-6) | PC250 |
| Write LSR High | PC160 |
| Carry | |
| Clock Storage Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

| | | |
|---|---|---|
| 00 | A155 | LI |
| 01 | A900 | LI |
| 02 | F128 | SRLD * |
| 03 | 0000 | B |

**Scope Setup**

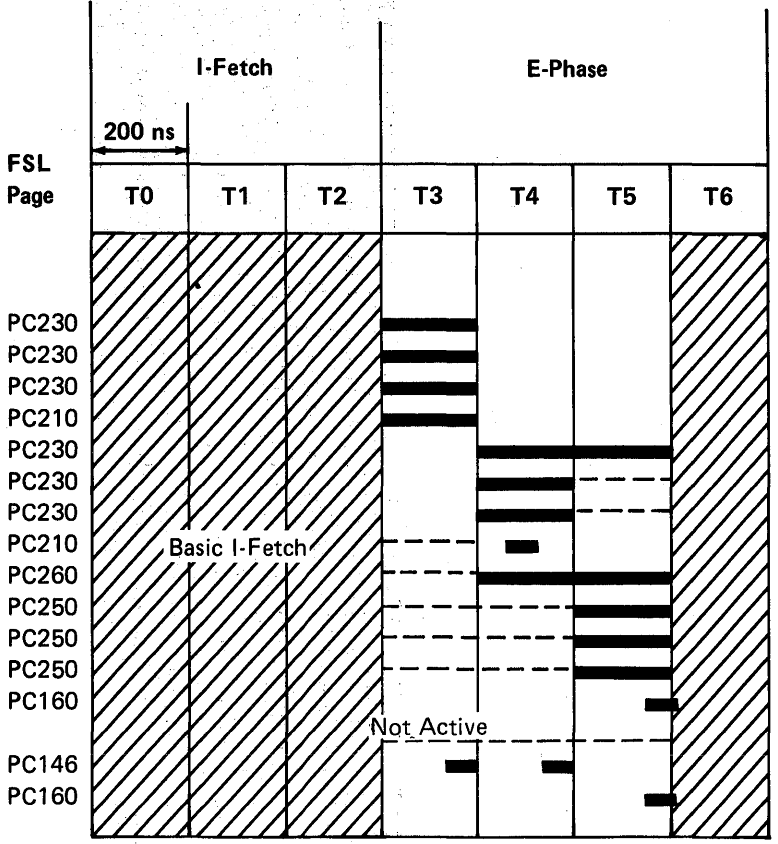
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

### Hex F95A

Select LSR (operand 2)
Stg Gate High from LSR High
Stg Gate Low from Stg Gate High
Y-Reg from Stg Gate High/Low
Select LSR (operand 1)
Stg Gate High from LSR High
Stg Gate Low from Stg Gate High
X-Reg from Stg Gate High/Low
Set ALU Mode (X + carry)
ALU Gate Low (8-11) from Y Low (8-11)
ALU Gate High (12-15) from X Low (8-11)
ALU Gate High from ALU High
Write LSR High
Carry
Clock Storage Gate Check
Clock ALU Gate Check



**Instruction Loop**

```
00    A9C1    LI
01    ·AAE1   LI
02    F95A    MZZ * (see note)
03    0000    B
```

*Note:* This instruction uses the high byte of both operands.

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

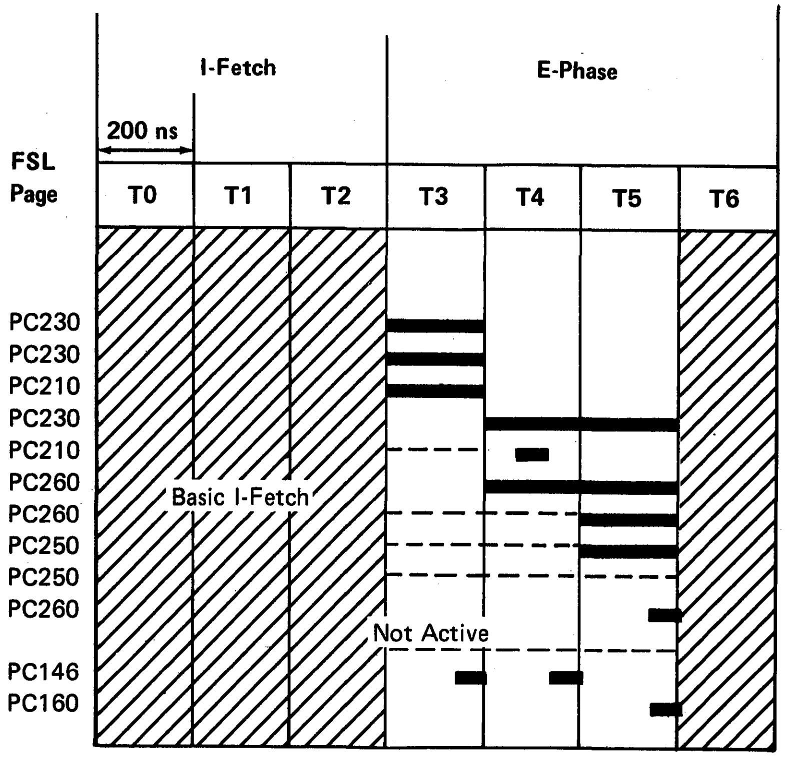
### Hex F14A

Select LSR (operand 2)
Stg Gate High/Low from LSR
Y-Reg from Stg Gate High/Low
Select LSR (operand 1)
X-Reg from Stg Gate High/Low
Set ALU Mode (X + carry)
ALU Gate Low (8-11) from Y Low (8-11)
ALU Gate Low (12-15) from ALU Low (8-11)
ALU Gate High from ALU Gate Low
Write LSR Low
Carry
Clock Storage Gate Check
Clock ALU Gate Check

**Instruction Loop**

```
00    A1C1    LI
01    A2E1    LI
02    F14A    MZZ * (see note)
03    0000    B
```

*Note:* This instruction uses the low byte of both operands.



**Scope Setup**

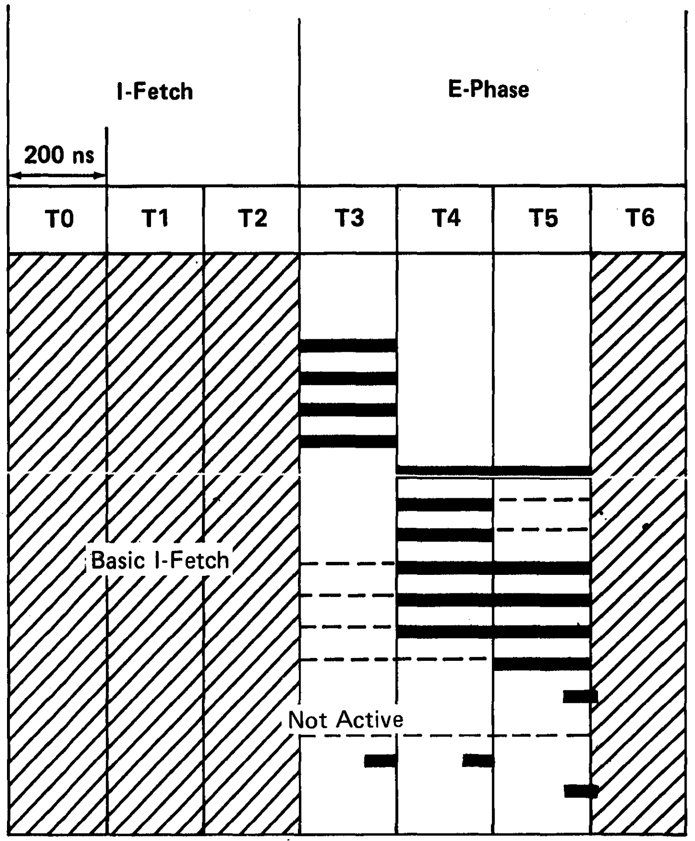
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = —'address compare' looking at the instruction referenced with an asterisk (*).

**Hex F97A**

| | FSL Page |
|---|---|
| Select LSR (operand 2) | PC230 |
| Stg Gate High from LSR | PC230 |
| Stg Gate Low from Stg Gate High | PC230 |
| Y-Reg from Stg Gate High/Low | PC210 |
| Select LSR (operand 1) | PC230 |
| Stg Gate High from LSR | PC230 |
| Stg Gate Low from Stg Gate High | PC230 |
| Set ALU Mode (X + carry) | PC260 |
| ALU Gate Low (8-11) from Y Low (8-11) | PC250 |
| ALU Gate Low (12-15) from ALU Low (12-15) | PC250 |
| ALU Gate High from ALU Gate Low | PC250 |
| Write LSR High | PC160 |
| Carry | |
| Clock Storage Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

```
00   A9C1   LI
01   AAE1   LI
02   F97A   MZN * (see note)
03   0000   B
```

*Note:* This instruction uses the high byte of both operands.

**Scope Setup**

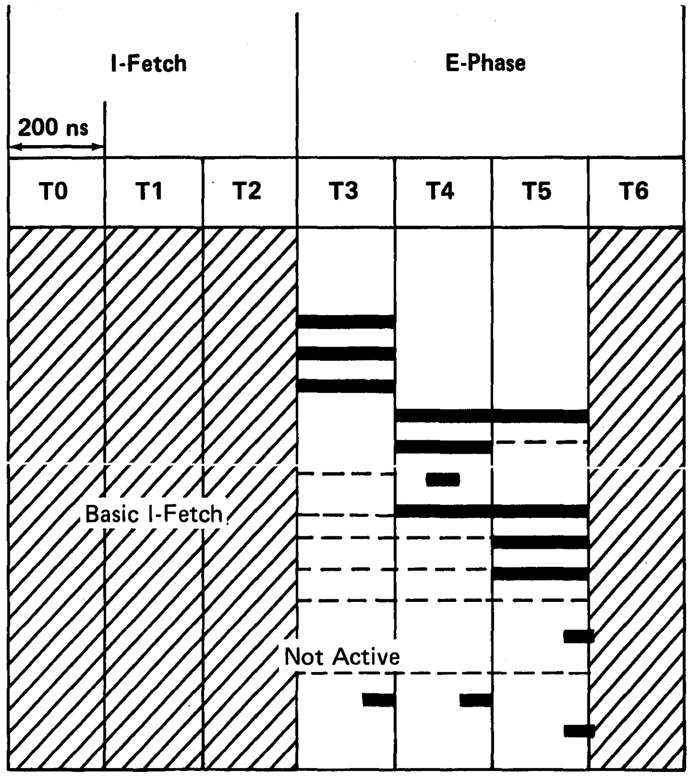
Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

**Hex F16A**

| | FSL Page |
|---|---|
| Select LSR (operand 2) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| Y-Reg from Stg Gate High/Low | PC210 |
| Select LSR (operand 1) | PC230 |
| Stg Gate High/Low from LSR | PC230 |
| X-Reg from Stg Gate High/Low | PC210 |
| Set ALU Mode (X + carry) | PC260 |
| ALU Gate Low (8-11) from Y Low (8-11) | PC250 |
| ALU Gate Low (12-15) from ALU Low (12-15) | PC250 |
| ALU Gate High from ALU Gate Low | PC250 |
| Write LSR Low | PC160 |
| Carry | |
| Clock Stg Gate Check | PC146 |
| Clock ALU Gate Check | PC160 |



**Instruction Loop**

```
00   A1C1   LI
01   A2E1   LI
02   F16A   MZN * (see note)
03   0000   B
```

*Note:* This instruction uses the low byte of both operands.

**Scope Setup**

Horizontal = 0.1 μs/div uncalibrated to display one 'phase A' cycle per division on chan 2.

Vertical = 0.2V/div using X10 probes.

Sync External = –'address compare' looking at the instruction referenced with an asterisk (*).

## I/O Immediate

| 1 0 1 1 | Modifier | Function | H2 | Reg 2 |
|---|---|---|---|---|
| 0 3 | 4 7 | 8 11 | 12 | 13 15 |

The I/O immediate instruction has four main functions:

- Move 1 byte of data between the local storage registers and the I/O devices
- Direct control of the channel and the I/O functions that may or may not include data movement
- Direct control of the control processor functions
- Direct control of the main storage processor functions

*Modifier (Bits 4-7):* The modifier bits rely on the device usage and are sent to the I/O attachment. These bits, along with the command bus out (CBO) bits, specify what is to be done.

*Function (Bits 8-11):* The function bits are sent to the port where they are decoded as one of the following commands: load, sense, control load, or control sense. This command is then sent to the I/O attachment on the 'command bus out' lines.

If bits 10 and 11 = binary 10, the command does not go to the port but remains in the control processor. For a bit definition of the sense information, see the control processor sense chart in this section.

*H2 (Bit 12):* Selects the high- or low-order byte of the selected local storage register.

H2 = 0: Low-order byte

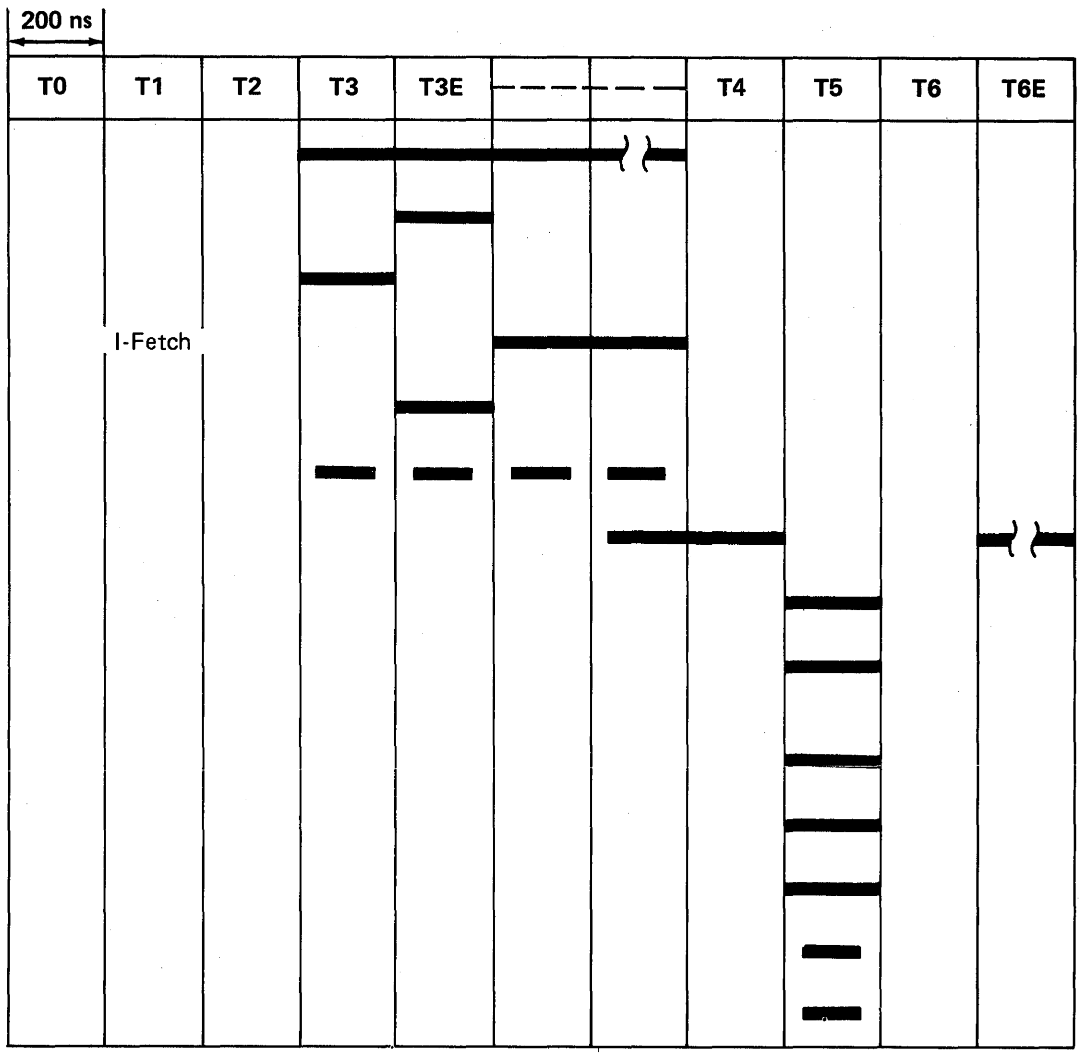
H2 = 1: High-order byte

*Reg 2 (Bits 13-15):* This field selects one of the eight work registers in the local storage register stack for the current interrupt level. This register is used for the byte of data or control information that is to be sent or received.

*Note:* For control processor control instructions, bits 12-15 are used as a second set of modifier bits.

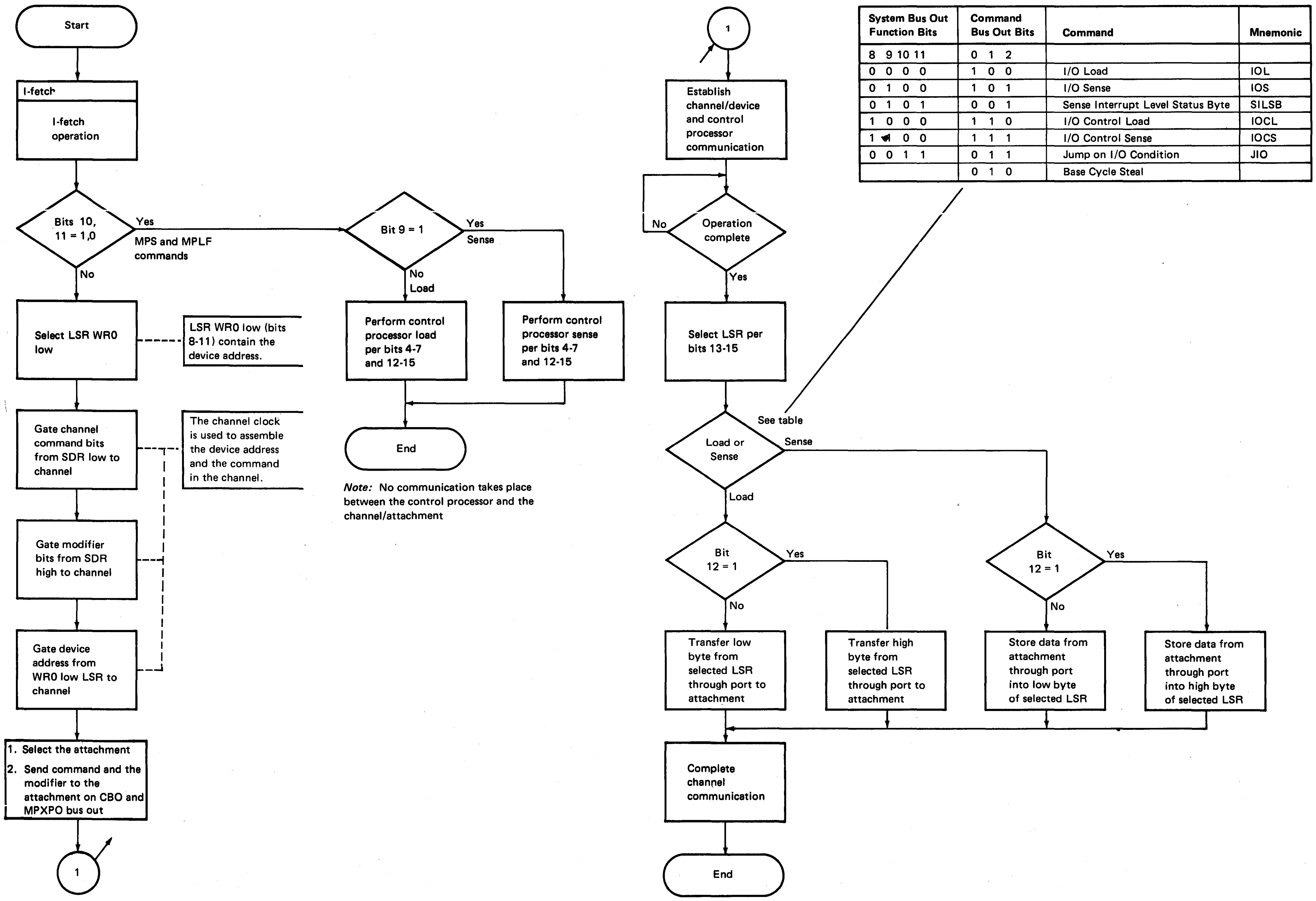
*Timing of CP Functions*

| | FSL Page |
|---|---|
| Select LSR (WR0) | PC240 |
| SDR High | |
| SDR Low | |
| LSR Low | PC230 |
| Select Storage Gate High (from SDR high) | PC230 |
| Clock X Low, X High, SAR | PC210 |
| Advance Time | PC518 |
| Select LSR (bits 13, 14, 15) | PC240 |
| Select Storage Gate High (from LSR high) | PC230 |
| Select Storage Gate Low (from channel bus: 9=1; from LSR: 9=0) | PC230 |
| Select ALU Gate Low (from storage gate low) | PC250 |
| Select ALU Gate High (from ALU gate low) | PC250 |
| Write LSR Low (9=1, 12=0) | PC160 |
| Write LSR High (9, 12=1) | PC160 |



*Note:* A more complete description of the I/O immediate commands can be found under *Commands* in the *Channel* section of this manual.

*Sequence of Port Communications*


Start
I-fetch
I-fetch operation
Bits 10, 11 = 1,0
Yes
MPS and MPLF commands
No
Select LSR WR0 low
LSR WR0 low (bits 8-11) contain the device address.
Gate channel command bits from SDR low to channel
The channel clock is used to assemble the device address and the command in the channel.
Gate modifier bits from SDR high to channel
Gate device address from WR0 low LSR to channel
1. Select the attachment
2. Send command and the modifier to the attachment on CBO and MPXPO bus out
1
Bit 9 = 1
Yes
Sense
No
Load
Perform control processor load per bits 4-7 and 12-15
Perform control processor sense per bits 4-7 and 12-15
End
1
Establish channel/device and control processor communication
No
Operation complete
Yes
Select LSR per bits 13-15
See table
Load or Sense
Sense
Load
Bit 12 = 1
Yes
No
Bit 12 = 1
Yes
No
Transfer low byte from selected LSR through port to attachment
Transfer high byte from selected LSR through port to attachment
Store data from attachment through port into low byte of selected LSR
Store data from attachment through port into high byte of selected LSR
Complete channel communication
End

*Note:* No communication takes place between the control processor and the channel/attachment

| System Bus Out Function Bits | Command Bus Out Bits | Command | Mnemonic |
|---|---|---|---|
| 8 9 10 11 | 0 1 2 | | |
| 0 0 0 0 | 1 0 0 | I/O Load | IOL |
| 0 1 0 0 | 1 0 1 | I/O Sense | IOS |
| 0 1 0 1 | 0 0 1 | Sense Interrupt Level Status Byte | SILSB |
| 1 0 0 0 | 1 1 0 | I/O Control Load | IOCL |
| 1 1 0 0 | 1 1 1 | I/O Control Sense | IOCS |
| 0 0 1 1 | 0 1 1 | Jump on I/O Condition | JIO |
| | 0 1 0 | Base Cycle Steal | |

**I/O Immediate Instructions**

| Op Code<br>0 1 2 3 | Modifier<br>4 5 6 7 | Function<br>8 9 10 11 | Address of LSR Used by Instruction<br>12 13 14 15 | WR0 Device Address | Device Type | |
|---|---|---|---|---|---|---|
| 1 0 1 1 | X X X X | 0 0 0 0<br>I/O Load<br>(IOL) | Z Z Z Z<br>(WR0 = 00)<br>Y R R R<br>(WR0 ≠ 00) | 00 =<br>50 =<br>80 =<br>A0 =<br>B0 =<br>C0 =<br>D0 =<br>E0 = | Channel (see instruction list)<br>Unit record (MICR) 1255<br>Communications<br>Disk A<br>Disk B<br>Work station<br>Diskette<br>Line printer | See note |
| 1 0 1 1 | X X X X | 0 1 0 0<br>I/O Sense<br>(IOS) | Y R R R | 00 =<br>50 =<br>80 =<br>A0 =<br>B0 =<br>C0 =<br>D0 =<br>E0 = | Channel (see chart and instr list)<br>Unit record (MICR) 1255<br>Communications<br>Disk A<br>Disk B<br>Work station<br>Diskette<br>Line printer | See note |
| 1 0 1 1 | X X X X | 0 1 0 1<br>Sense Interrupt Level Status Byte<br>(SILSB) | Y R R R | *0 = | Channel (see chart and instr list)<br>*Interrupt level of data | |
| 1 0 1 1 | X X X X | 0 1 1 0<br>Control Processor Sense (MPS) | Y R R R | N/A | (see chart and instruction list) | |
| 1 0 1 1 | X X X X | 1 0 0 0<br>I/O Control Load (IOCL) | X X X X | 00 =<br>50 =<br>80 =<br>A0 =<br>B0 =<br>C0 =<br>D0 =<br>E0 = | Channel (see instruction list)<br>Unit record (MICR) 1255<br>Communications<br>Disk A<br>Disk B<br>Work station<br>Diskette<br>Line printer | See note |
| 1 0 1 1 | X X X X | 1 0 1 0<br>Control Processor Load Function<br>(MPLF) | X X X X | N/A (see instruction list) | | |

| Op Code<br>0 1 2 3 | Modifier<br>4 5 6 7 | Function<br>8 9 10 11 | Address of LSR Used by Instruction<br>12 13 14 15 | WR0 Device Address | Device Type |
|---|---|---|---|---|---|
| 1 0 1 1 | X X X X<br>For diagnostic purposes only | 1 1 0 0<br>I/O Control Sense (IOCS) | Y R R R | 00 =<br>50 =<br>80 =<br>A0 =<br>B0 =<br>C0 =<br>D0 =<br>E0 = | Channel<br>Unit record (MICR) 1255<br>Communications<br>Disk A<br>Disk B<br>Work station<br>Diskette<br>Line printer |

Legend:
X = Dependent on specific function
Y = High or low byte of selected LSR
Z = Not required or used
R = Selected LSR value
* = Interrupt level

Note: See *Commands* in the appropriate attachment section of this manual.

**Control Processor Sense (MPS)**

The contents of these bytes or switches are moved to an LSR. This data can then be used by the program.

| 4 5 6 7 | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|---|---|
| 0 0 1 1<br>Interrupt status | Invalid logout | | | | | Interrupt code | Interrupt code | Interrupt code |
| 0 1 0 0<br>Console status byte | Stop key | Main storage address compare | Overlap off | MSIPL device select switch | I/O request | Sys step mode | Go flag | Micro-interrupt check |
| 0 1 0 1<br>Address/ Data switches 3 and 4 | Switch 3<br>8 | Switch 3<br>4 | Switch 3<br>2 | Switch 3<br>1 | Switch 4<br>8 | Switch 4<br>4 | Switch 4<br>2 | Switch 4<br>1 |
| 0 1 1 0<br>I/O clocks low byte | 8.19 ms | 16.38 ms | 32.77 ms | 65.54 ms | 131.1 ms | 262.1 ms | 524.3 ms | 1s |
| 0 1 1 1<br>I/O clocks high byte | 32 μs | 64 μs | 128 μs | 256 μs | 512 μs | 1.02 ms | 2.05 ms | 4.10 ms |
| 1 0 0 1<br>Address/ Data switches 1 and 2 | Switch 1<br>8 | Switch 1<br>4 | Switch 1<br>2 | Switch 1<br>1 | Switch 2<br>8 | Switch 2<br>4 | Switch 2<br>2 | Switch 2<br>1 |
| 1 0 1 0<br>CPU error byte | SDR P check | MOR P check | Storage gate P check | ALU gate P check | Control storage invalid addr/ SAR check | Microloop time-out/ SAR check | Main storage invalid addr/ MSAR check | Main storage exception/ MSAR check |
| 1 0 1 1<br>PCR | Flag | Plus | Minus | Zero | Carry log | High log | Low log | Equal log |

*Control Processor Sense (Interrupt Status/Code)*

The interrupt code indicates which hardware interrupt level the control processor was executing on when the error occurred that caused the logout. A decode of the interrupt code in terms of a hardware interrupt level is as follows:

| **Interrupt Code (Bits 5-7) (Hex)** | **Hardware Interrupt Level** |
|---|---|
| 0 | 5 |
| 1 | 4/Base cycle steal |
| 2 | Base cycle steal/Burst cycle steal |
| 3 | 3 |
| 4 | 2 |
| 5 | 1/Burst cycle steal |
| 7 | 0/Main level |

*I/O Sense (IOS)*

By checking channel check byte bit 6 = 1 (cycle steal check), the CE can determine if the interrupt was caused by a hardware level or a cycle steal operation.

IOS (Channel/Port)

| | 4 5 6 7 | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|---|---|---|
| Sense Port Register | 0 0 0 0 | Data | | | | | | | |
| Sense Port Error Byte | 0 0 0 1 | MPXPO bus out | Invalid device address | DBI P check | I/O time-out check | CBI/DBI not zero | System bus out P check | Cycle steal check | Invalid port |

**Valid I/O Immediate Instructions (Numeric Sequence by Instruction)**

| Numeric | Op Code | Mod | Funct | Reg 2 | Description | Mnemonic |
|---|---|---|---|---|---|---|
| B00X | B | 0 | 0 | X | Start fixed interval timer | IOL |
| B04R | B | 0 | 4 | R | Sense port register | IOS |
| B05R | B | 0 | 5 | R | Sense interrupt level status byte | SILSB |
| B06R | B | 0 | 6 | R | Reserved | MPS |
| B08X | B | 0 | 8 | X | Disable extended time-out | IOCL |
| B0AR | B | 0 | A | R | Load PCR (from high byte of register only) | MPLF |
| B10X | B | 1 | 0 | X | Disable main storage processor level 5 interrupt | IOL |
| B14R | B | 1 | 4 | R | Sense port error byte | IOS |
| B16R | B | 1 | 6 | R | No-op | MPS |
| B18R | B | 1 | 8 | R | Load port register | IOCL |
| B1AX | B | 1 | A | X | Reset carry-set equal | MPLF |
| B20X | B | 2 | 0 | X | Reset main storage processor level 5 interrupt | IOL |
| B26R | B | 2 | 6 | R | No-op | MPS |
| B28X | B | 2 | 8 | X | Reset port check | IOCL |
| B2AX | B | 2 | A | X | Reset event light 2 | MPLF |
| B30X | B | 3 | 0 | X | Reset fixed interval timer interrupt | IOL |
| B36R | B | 3 | 6 | R | Sense interrupt status | MPS |
| B38X | B | 3 | 8 | X | Enable extended time-out | IOCL |
| B3AX | B | 3 | A | X | Reset event light 3 | MPLF |
| B40X | B | 4 | 0 | X | Enable main storage processor level 5 interrupt | IOL |
| B46R | B | 4 | 6 | R | Console status byte | MPS |
| B48X | B | 4 | 8 | X | Reset interrupt level 5 request | IOCL |
| B4AX | B | 4 | A | X | Reset event light 4 | MPLF |
| B50X | B | 5 | 0 | X | Stop fixed interval timer | IOL |
| B56R | B | 5 | 6 | R | Address/Data switches 3-4 | MPS |
| B58X | B | 5 | 8 | X | Set channel odd parity | IOCL |
| B5AX | B | 5 | A | X | Reset event light 5 | MPLF |
| B60X | B | 6 | 0 | X | No-op | IOL |
| B66R | B | 6 | 6 | R | I/O clocks low byte | MPS |
| B68X | B | 6 | 8 | X | Set channel even parity | IOCL |
| B6AX | B | 6 | A | X | Reset event light 6 | MPLF |
| B70X | B | 7 | 0 | X | Set main storage processor level 5 interrupt | IOL |
| B76R | B | 7 | 6 | R | I/O clocks high byte | MPS |
| B78X | B | 7 | 8 | X | Set interrupt level 5 request | IOCL |
| B7AX | B | 7 | A | X | Reset event light 7 | MPLF |
| B80X | B | 8 | 0 | X | Disable main storage processor level 5 request | IOL |
| B8AX | B | 8 | A | X | Set flag | MPLF |
| B90X | B | 9 | 0 | X | Enable main storage processor interrupt level 5 request | IOL |
| B96R | B | 9 | 6 | R | Address/Data switches 1-2 | MPS |
| B9AF | B | 9 | A | F | No-op | MPLF |
| BA0F | B | A-F | 0 | F | No-op | IOL |
| BA6R | B | A | 6 | R | Common processor check byte 0 | MPS |
| BAAF | B | A | A | F | No-op | MPLF |
| BB6R | B | B | 6 | R | Control processor condition reg (PCR) | MPS |
| BBAF | B | B | A | F | Reset flag | MPLF |
| BCAF | B | C | A | F | No-op | MPLF |
| BDAF | B | D | A | F | No-op | MPLF |
| BEA0 | B | E | A | 0 | No-op | MPLF |
| BEA1 | B | E | A | 1 | Set I/O service request | MPLF |
| BEA2 | B | E | A | 2 | Reset I/O service request | MPLF |
| BEA3 | B | E | A | 3 | Processor check halt | MPLF |
| BEA4 | B | E | A | 4 | Disable checks | MPLF |
| BEA5 | B | E | A | 5 | Enable interrupts | MPLF |
| BEA6 | B | E | A | 6 | Disable interrupts | MPLF |
| BEA7 | B | E | A | 7 | Enable checks | MPLF |
| BEA8 | B | E | A | 8 | Reset main storage processor | MPLF |
| BEA9 | B | E | A | 9 | Turn on System In Use light | MPLF |
| BEAA | B | E | A | A | Turn off System In Use light | MPLF |
| BEAB | B | E | A | B | Start main storage processor | MPLF |
| BEAC | B | E | A | C | No-op | MPLF |
| BEAD | B | E | A | D | No-op | MPLF |
| BEAE | B | E | A | E | No-op | MPLF |
| BFA0 | B | F | A | 0 | Set control processor working | MPLF |
| BFA1 | B | F | A | 1 | Reset 'stop' latch | MPLF |
| BFA2 | B | F | A | 2 | Reset 'machine check interrupt' latch | MPLF |
| BFA3 | B | F | A | 3 | Reset 'go' latch | MPLF |
| BFA4 | B | F | A | 4 | Enable control processor loop time-out | MPLF |
| BFA5 | B | F | A | 5 | Set 'stop' latch | MPLF |
| BFA6 | B | F | A | 6 | Reset 'retry' latch, reset control processor loop time-out, and set go | MPLF |
| BFA7 | B | F | A | 7 | Set retry | MPLF |
| BFA8 | B | F | A | 8 | Enable I/O clocks | MPLF |
| BFA9 | B | F | A | 9 | No-op | MPLF |
| BFAA | B | F | A | A | Reset I/O clocks | MPLF |
| BFAB | B | F | A | B | Disable I/O clocks | MPLF |
| BFAC | B | F | A | C | No-op | MPLF |
| BFAD | B | F | A | D | Reset control processor working | MPLF |
| BFAE | B | F | A | E | Processor wait | MPLF |

Reg 2 Legend
X = Don't care
R = Specify register

**Valid I/O Immediate Instructions (Alphabetic Sequence by Description)**

| Numeric | Op Code | Mod | Funct | Reg 2 | Description | Mnemonic |
|---|---|---|---|---|---|---|
| B96R | B | 9 | 6 | R | Address/Data switches 1-2 | MPS |
| B56R | B | 5 | 6 | R | Address/Data switches 3-4 | MPS |
| BA6R | B | A | 6 | R | Common processor check byte 0 | MPS |
| B46R | B | 4 | 6 | R | Console status byte | MPS |
| BB6R | B | B | 6 | R | Control processor condition reg (PCR) | MPS |
| BEA4 | B | E | A | 4 | Disable checks | MPLF |
| BFAB | B | F | A | B | Disable I/O clocks | MPLF |
| BEA6 | B | E | A | 6 | Disable interrupts | MPLF |
| B10X | B | 1 | 0 | X | Disable main storage processor level 5 interrupt | IOL |
| B80X | B | 8 | 0 | X | Disable main storage processor level 5 request | IOL |
| B08X | B | 0 | 8 | X | Disable extended time-out | IOCL |
| BEA7 | B | E | A | 7 | Enable checks | MPLF |
| BFA4 | B | F | A | 4 | Enable control processor loop time-out | MPLF |
| B38X | B | 3 | 8 | X | Enable extended time-out | IOCL |
| BFA8 | B | F | A | 8 | Enable I/O clocks | MPLF |
| BEA5 | B | E | A | 5 | Enable interrupts | MPLF |
| B40X | B | 4 | 0 | X | Enable main storage processor level 5 interrupt | IOL |
| B90X | B | 9 | 0 | X | Enable main storage processor interrupt level 5 request | IOL |
| B76R | B | 7 | 6 | R | I/O clocks high byte | MPS |
| B66R | B | 6 | 6 | R | I/O clocks low byte | MPS |
| B0AR | B | 0 | A | R | Load PCR (from high byte of register only) | MPLF |
| B18R | B | 1 | 8 | R | Load port register | IOCL |
| BAAF | B | A | A | F | No-op | MPLF |
| BA0F | B | A-F | 0 | F | No-op | IOL |
| BCAF | B | C | A | F | No-op | MPLF |
| BDAF | B | D | A | F | No-op | MPLF |
| BEAC | B | E | A | C | No-op | MPLF |
| BEAD | B | E | A | D | No-op | MPLF |
| BEAE | B | E | A | E | No-op | MPLF |
| BEA0 | B | E | A | 0 | No-op | MPLF |
| BFAC | B | F | A | C | No-op | MPLF |
| BFA9 | B | F | A | 9 | No-op | MPLF |
| B60X | B | 6 | 0 | X | No-op | IOL |
| B9AF | B | 9 | A | F | No-op | MPLF |
| BEA3 | B | E | A | 3 | Processor check halt | MPLF |
| BFAE | B | F | A | E | Processor wait | MPLF |
| B06R | B | 0 | 6 | R | Reserved | MPS |
| B16R | B | 1 | 6 | R | No-op | MPS |
| B26R | B | 2 | 6 | R | No-op | MPS |
| B1AX | B | 1 | A | X | Reset carry-set equal | MPLF |
| BFAD | B | F | A | D | Reset control processor working | MPLF |
| B2AX | B | 2 | A | X | Reset event light 2 | MPLF |
| B3AX | B | 3 | A | X | Reset event light 3 | MPLF |
| B4AX | B | 4 | A | X | Reset event light 4 | MPLF |
| B5AX | B | 5 | A | X | Reset event light 5 | MPLF |
| B6AX | B | 6 | A | X | Reset event light 6 | MPLF |
| B7AX | B | 7 | A | X | Reset event light 7 | MPLF |
| B30X | B | 3 | 0 | X | Reset fixed interval timer interrupt | IOL |
| BBAF | B | B | A | F | Reset flag | MPLF |
| BFA3 | B | F | A | 3 | Reset 'go' latch | MPLF |
| BFAA | B | F | A | A | Reset I/O clocks | MPLF |
| BEA2 | B | E | A | 2 | Reset I/O service request | MPLF |
| B48X | B | 4 | 8 | X | Reset interrupt level 5 request | IOCL |
| BFA2 | B | F | A | 2 | Reset 'machine check interrupt' latch | MPLF |
| BEA8 | B | E | A | 8 | Reset main storage processor | MPLF |
| B20X | B | 2 | 0 | X | Reset main storage processor level 5 interrupt | IOL |
| B28X | B | 2 | 8 | X | Reset port check | IOCL |
| BFA6 | B | F | A | 6 | Reset 'retry' latch, reset control processor loop time-out, and set go | MPLF |
| BFA1 | B | F | A | 1 | Reset 'stop' latch | MPLF |
| B05R | B | 0 | 5 | R | Sense interrupt level status byte | SILSB |
| B36R | B | 3 | 6 | R | Sense interrupt status | MPS |
| B14R | B | 1 | 4 | R | Sense port error byte | IOS |
| B04R | B | 0 | 4 | R | Sense port register | IOS |
| B68X | B | 6 | 8 | X | Set channel even parity | IOCL |
| B58X | B | 5 | 8 | X | Set channel odd parity | IOCL |
| BFA0 | B | F | A | 0 | Set control processor working | MPLF |
| B8AX | B | 8 | A | X | Set flag | MPLF |
| BEA1 | B | E | A | 1 | Set I/O service request | MPLF |
| B78X | B | 7 | 8 | X | Set interrupt level 5 request | IOCL |
| B70X | B | 7 | 0 | X | Set main storage processor level 5 interrupt | IOL |
| BFA7 | B | F | A | 7 | Set retry | MPLF |
| BFA5 | B | F | A | 5 | Set 'stop' latch | MPLF |
| B00X | B | 0 | 0 | X | Start fixed interval timer | IOL |
| BEAB | B | E | A | B | Start main storage processor | MPLF |
| B50X | B | 5 | 0 | X | Stop fixed interval timer | IOL |
| BEAA | B | E | A | A | Turn off System In Use light | MPLF |
| BEA9 | B | E | A | 9 | Turn on System In Use light | MPLF |

### I/O Load or I/O Control Load (IOL, IOCL)

| 1011 | Modifier | Function | H2 | Reg 2 |
|---|---|---|---|---|
| 0 3 | 4 7 | 8 11 | 12 | 13 15 |

This part of the I/O immediate instruction moves 1 byte of data or control information from a local storage register to the I/O attachment.

*Modifier (Bits 4-7):* The modifier bits are specified for the device and are sent to the I/O attachment with the command. These bits specify what is to be done with the data byte.

*Function (Bits 8-11):* The function bits are sent to the channel where they are decoded as either the load or control load command. This command is then sent to the I/O attachment on the 'command bus out' lines.

Bits 8-11 = 0000 for IOL

Bits 8-11 = 1000 for IOCL

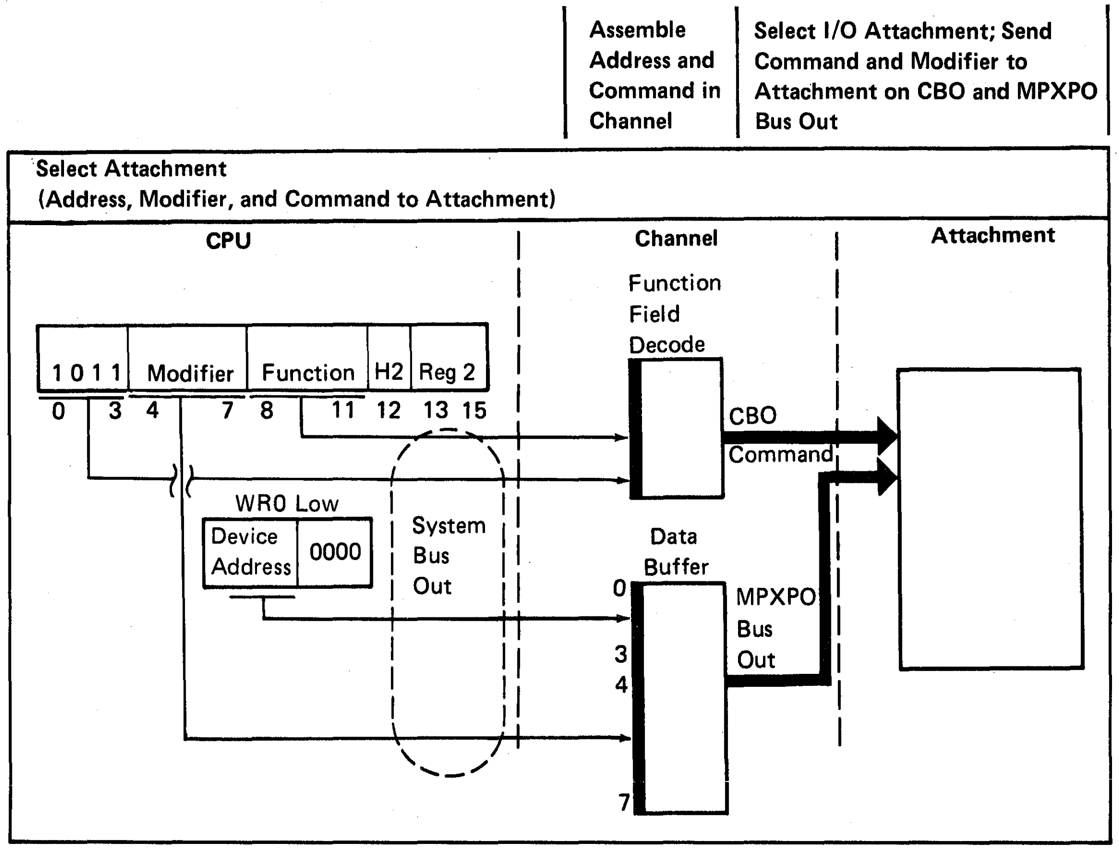
*H2 (Bit 12):* Selects the low- or high-order byte of the selected local storage register of the current interrupt level:

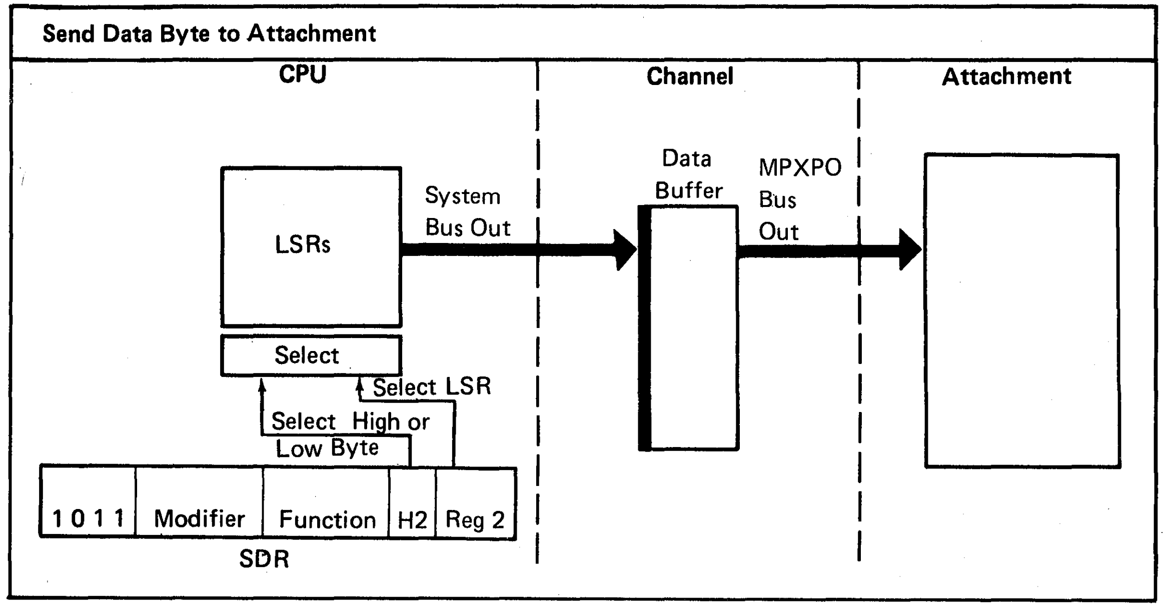
H2 = 0: Low-order byte

H2 = 1: High-order byte

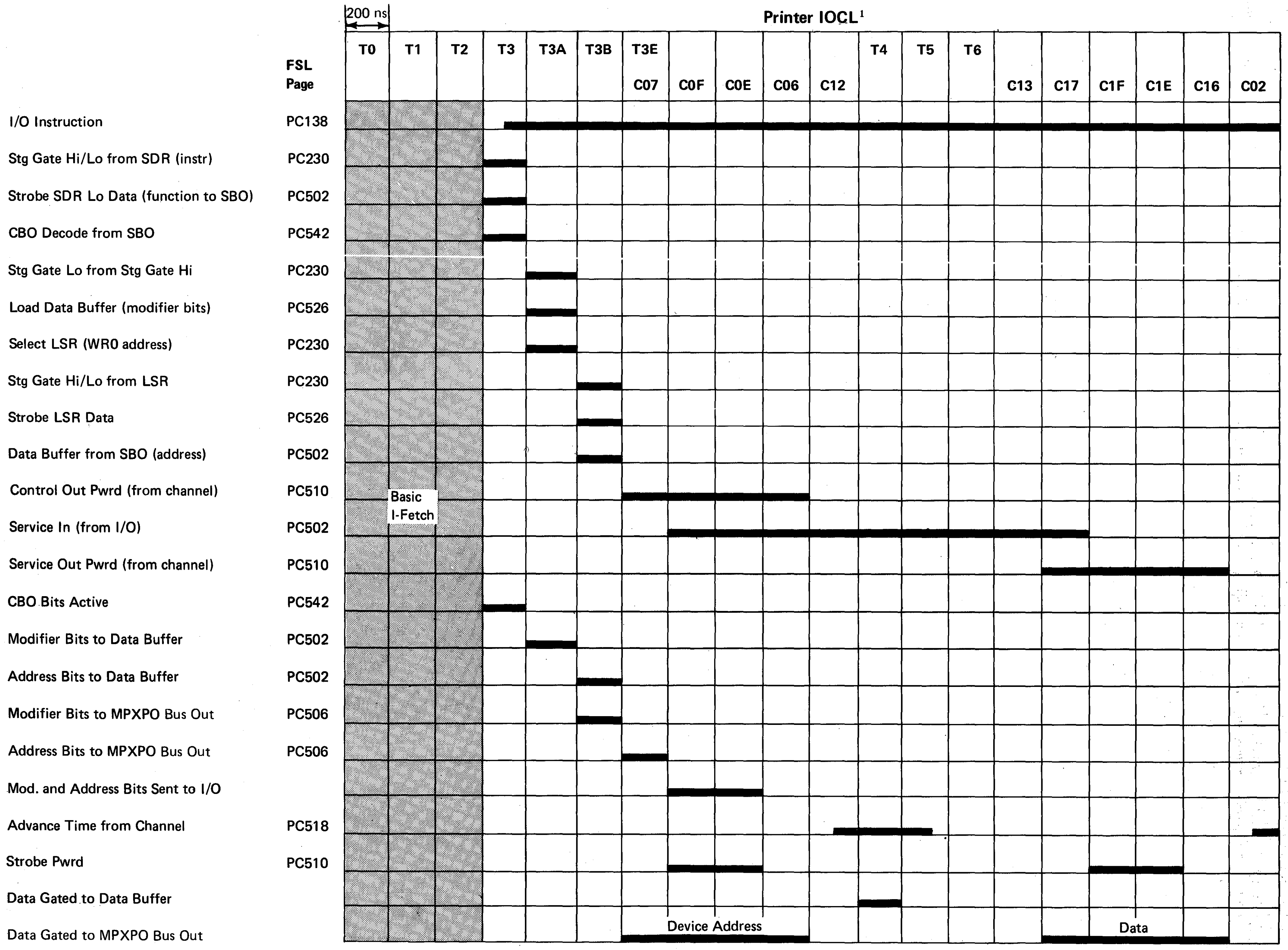
*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected register contains the byte of data or control information that is to be sent to the I/O attachment.

*Note:* A more complete description of the I/O load and I/O control load commands may be found under *Commands* in the *Channel* section of this manual.





*Timing of CP/Channel Functions*


200 ns
Printer IOCL[1]
FSL Page
T0
T1
T2
T3
T3A
T3B
T3E
C07
C0F
C0E
C06
C12
T4
T5
T6
C13
C17
C1F
C1E
C16
C02
I/O Instruction PC138
Stg Gate Hi/Lo from SDR (instr) PC230
Strobe SDR Lo Data (function to SBO) PC502
CBO Decode from SBO PC542
Stg Gate Lo from Stg Gate Hi PC230
Load Data Buffer (modifier bits) PC526
Select LSR (WR0 address) PC230
Stg Gate Hi/Lo from LSR PC230
Strobe LSR Data PC526
Data Buffer from SBO (address) PC502
Control Out Pwrd (from channel) PC510
Service In (from I/O) PC502
Service Out Pwrd (from channel) PC510
CBO Bits Active PC542
Modifier Bits to Data Buffer PC502
Address Bits to Data Buffer PC502
Modifier Bits to MPXPO Bus Out PC506
Address Bits to MPXPO Bus Out PC506
Mod. and Address Bits Sent to I/O
Advance Time from Channel PC518
Strobe Pwrd PC510
Data Gated to Data Buffer
Data Gated to MPXPO Bus Out
Basic
I-Fetch
Device Address
Data

[1] See *Channel Exerciser Loop Program* in the *Channel* section of this manual for a program that can be used with this command.

The first 'strobe pwrd' pulse after the rise of the 'control out pwrd' line signals the I/O attachment that the device address and the command information on the 'command bus out' and 'MPXPO bus out' lines are valid. The rise of the 'service in' line signals the port that the I/O attachment has taken the information from the 'command bus out' and 'MPXPO bus out' lines and is ready to receive data.

The first 'strobe pwrd' pulse after the rise of the 'service out pwrd' line signals the I/O attachment that the data byte on the 'data bus out' lines is valid. The fall of the 'service in' line signals the port that the I/O attachment has taken the data byte from the 'MPXPO bus out' lines.

### I/O Sense or I/O Control Sense (IOS, IOCS)

| 1 0 1 1 | Modifier | Function | H2 | Reg 2 |
|---|---|---|---|---|
| 0 3 | 4 7 | 8 11 | 12 | 13 15 |

This part of the I/O immediate instruction moves 1 byte of data or status type information from the I/O attachment to a local storage register.

*Modifier (Bits 4-7):* The modifier bits are specified by the device and are sent to the I/O attachment with the command. These bits specify which data byte is to be sent.

*Function (Bits 8-11):* The function bits are sent to the port where they are decoded as either the sense or control sense command. This command is then sent to the I/O attachment on the 'command bus out' lines.

Bits 8-11 = 0100 for IOS

Bits 8-11 = 1100 for IOCS

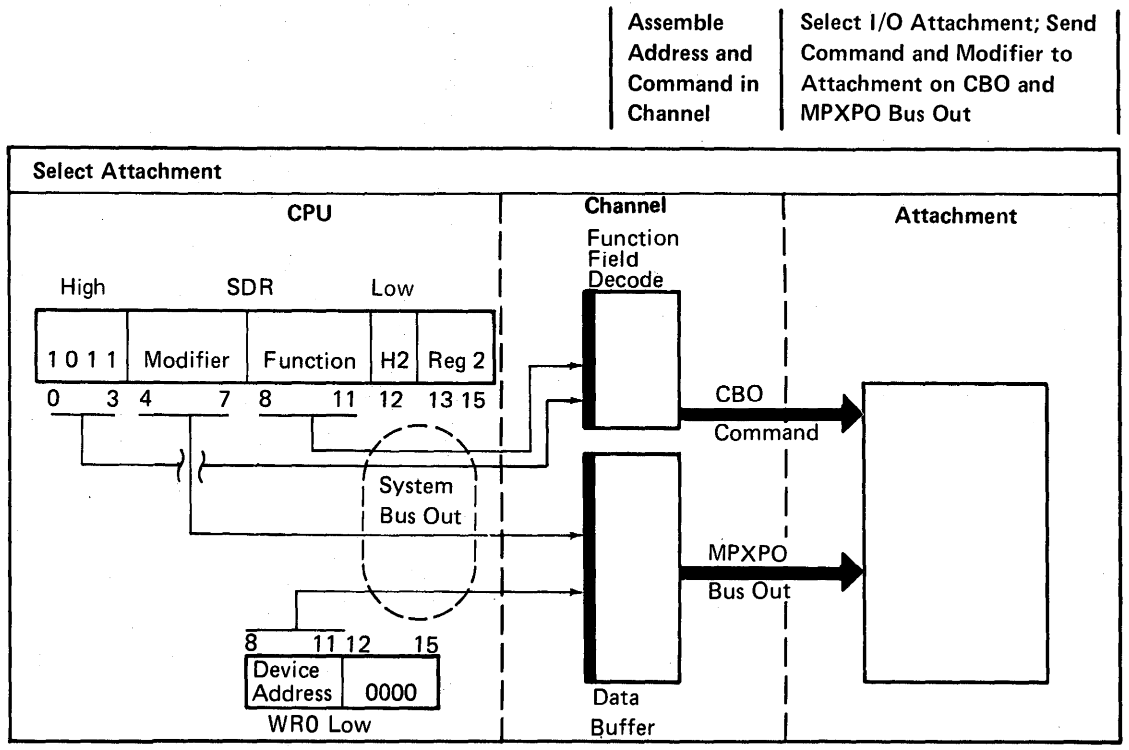
*H2 (Bit 12):* This bit is used to select the low- or high-order byte of the selected local storage register:

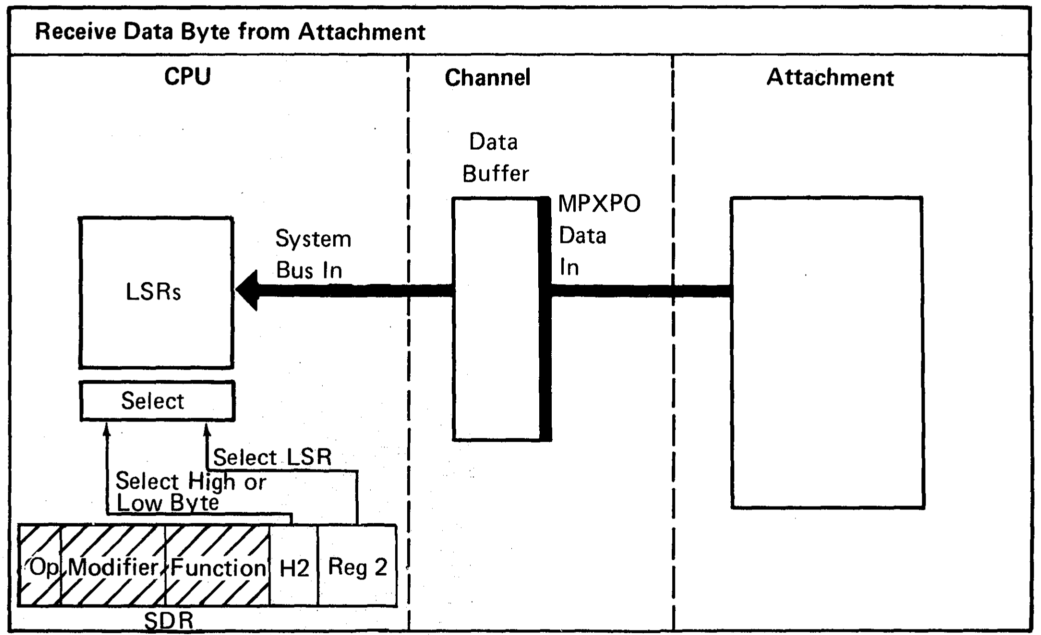
H2 = 0: Low-order byte

H2 = 1: High-order byte

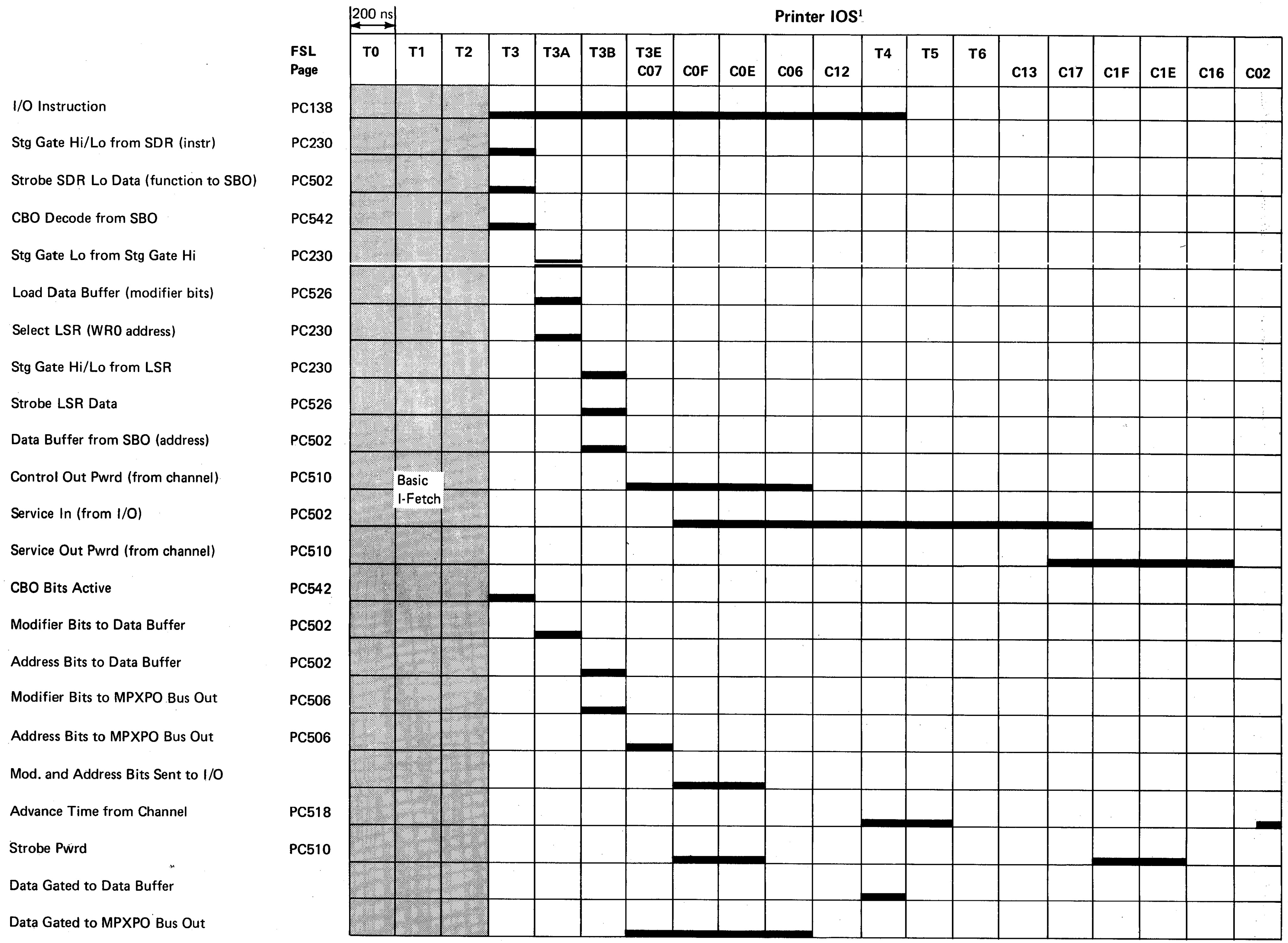
*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The byte of data being sent from the I/O attachment is placed in this local storage register.

*Note:* A more complete description of the I/O sense and I/O control sense commands may be found under *Commands* in the *Channel* section of this manual.





*Timing of CP/Channel Functions*



[1] See *Channel Exerciser Loop Program* in the *Channel* section of this manual for a program that can be used with this command.

The first 'strobe pwrd' pulse after the rise of the 'control out pwrd' line signals the I/O attachment that the device address and the command information on the 'command bus out' and 'MPXPO bus out' lines are valid.

The rise of the 'service in' line signals the port that the I/O attachment has taken the information from the 'command bus out' and 'MPXPO bus out' lines. The rise of the 'service in' line also signals the port that the data byte on the 'MPXPO data in' lines is valid. The rise of the 'service out pwrd' line signals the I/O attachment that the channel has taken the byte from the 'MPXPO data in' lines.

### Sense Interrupt Level Status Byte (SILSB)

| | | | | |
|---|---|---|---|---|
| 1011 | Modifier | Function | H2 | Reg 2 |
| 0   3 | 4   7 | 8   11 | 12 | 13   15 |

This function of the I/O immediate instruction moves 1 byte of status information from the I/O attachment to the selected local storage register. This status byte determines which devices are requesting service on a given interrupt level.

*Modifier (Bits 4-7):* The modifier bits are specified for each device and are sent to the I/O attachment with the command. These bits specify what is to be done with the data byte.

*Function (Bits 8-11):* The function bits are sent to the channel where they are decoded along with the operation code as a sense interrupt level status byte command. This command is then sent to the I/O attachment on the 'command bus out' lines.

Bits 8-11 = 0101

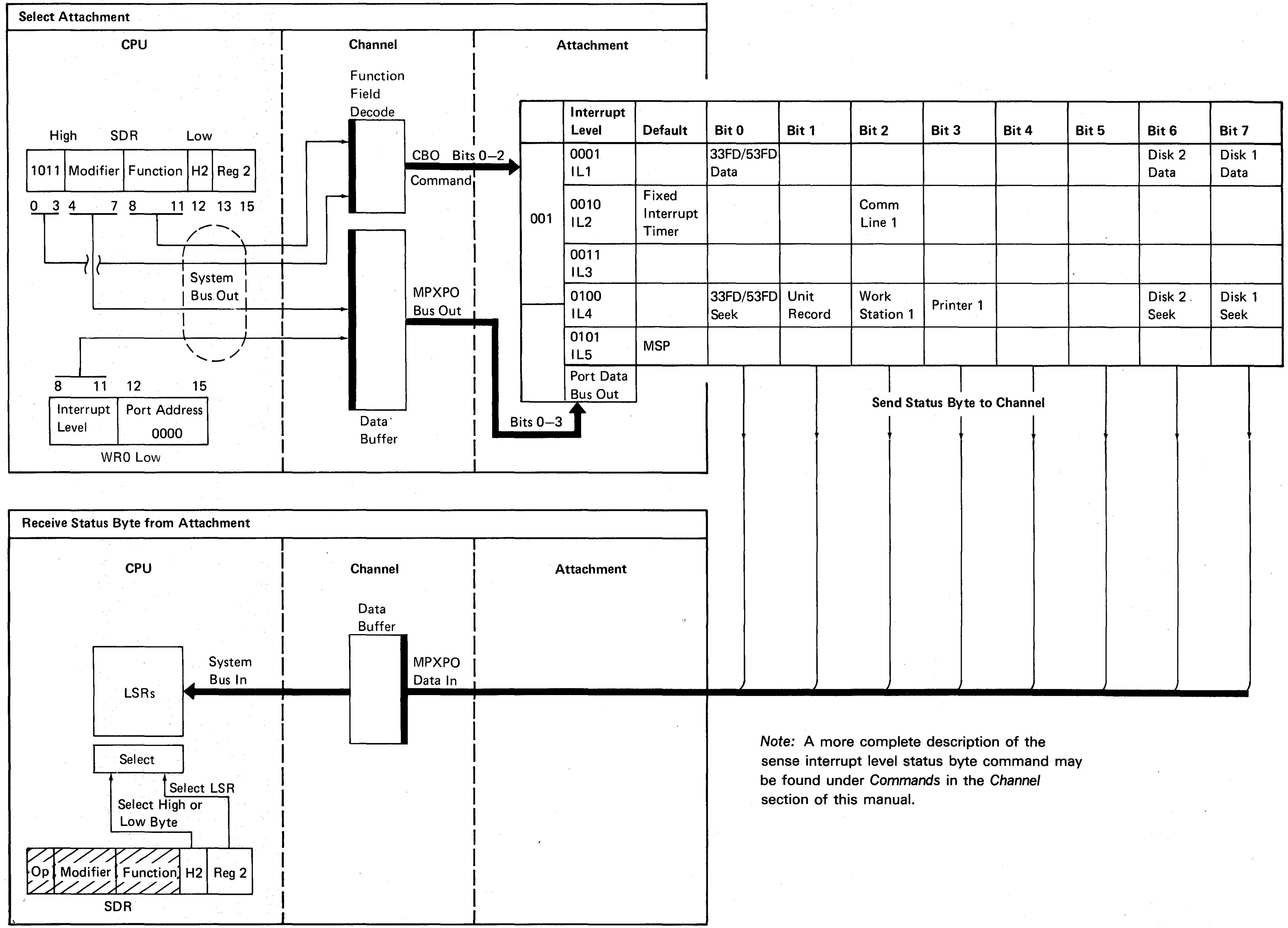
*H2 (Bit 12):* Selects the low- or high-order byte of the selected LSR of the current interrupt level:

H2 = 0: Low-order byte

H2 = 1: High-order byte

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected register stores the byte of status (information containing the device causing the interrupt level) received from the I/O attachment.

*WR0 Low (Bits 8-11):* Contains the interrupt level hexadecimal value used by the I/O attachment to select the status byte of information to be stored in the selected local storage register.



| | Interrupt Level | Default | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 001 | 0001 IL1 | | 33FD/53FD Data | | | | | | Disk 2 Data | Disk 1 Data |
| | 0010 IL2 | Fixed Interrupt Timer | | | Comm Line 1 | | | | | |
| | 0011 IL3 | | | | | | | | | |
| | 0100 IL4 | | 33FD/53FD Seek | Unit Record | Work Station 1 | Printer 1 | | | Disk 2 Seek | Disk 1 Seek |
| | 0101 IL5 | MSP | | | | | | | | |
| | Port Data Bus Out | | | | | | | | | |

*Note:* A more complete description of the sense interrupt level status byte command may be found under *Commands* in the *Channel* section of this manual.

### Control Processor Load Function (MPLF)

| 1011 | Modifier | Function | H2 | Reg 2 |
|---|---|---|---|---|
| 0 3 | 4 7 | 8 11 | 12 | 13 15 |

This function of the I/O immediate instruction does not go to the channel but remains in the control processor. It performs functions (such as loading registers), sets/resets conditions, and enables/disables conditions.

*Modifier (Bits 4-7):* Specifies the type of load function to be performed by the command.

*Function (Bits 8-11):* Decoded by the control processor as an internal load function when bits 10 and 11 are equal to binary 10.

*Modifier 2 (Bits 12-15):* Combines with bits 4-7 to specify the type of load function to be performed by the command.

### Control Processor Sense (MPS)

| 1011 | Modifier | Function | H2 | Reg 2 |
|---|---|---|---|---|
| 0 3 | 4 7 | 8 11 | 12 | 13 15 |

This function of the I/O immediate instruction does not go to the channel but remains in the control processor. A byte of data is moved to a local storage register to be used by the program. The byte contains one of the following:

Console status

Address/Data switches 1-4

Processor condition register

Interrupt status

*Modifier (Bits 4-7):* Selects the byte of data or status to be moved to the selected local storage register.

*Function (Bits 8-11):* Decoded by the control processor as an internal sense function when bits 10 and 11 are equal to binary 10.

*H2 (Bit 12):* Selects the low- or high-order byte of the selected local storage register for the current interrupt level:

H2 = 0: Low-order byte

H2 = 1: High-order byte

*Register 2 (Bits 13-15):* Selects one of the eight work registers in the local storage register stack for the current interrupt level. The selected register stores the byte of data to be used by the program.

The contents of these bytes or switches are moved to an LSR. This data can then be used by the program.

**Control Processor Sense (MPS)**

| 4 5 6 7 | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|---|---|
| 0 0 1 1 Interrupt status | Invalid logout | | | | | Interrupt code | Interrupt code | Interrupt code |
| 0 1 0 0 Console status byte | Stop key | Main storage address compare | Overlap off | MSIPL device select switch | I/O request | Sys step mode | Go flag | Micro-interrupt check |
| 0 1 0 1 Address/Data switches 3 and 4 | Switch 3 8 | Switch 3 4 | Switch 3 2 | Switch 3 1 | Switch 4 8 | Switch 4 4 | Switch 4 2 | Switch 4 1 |
| 0 1 1 0 I/O clocks low byte | 8.19 ms | 16.38 ms | 32.77 ms | 65.54 ms | 131.1 ms | 262.1 ms | 524.3 ms | 1s |
| 0 1 1 1 I/O clocks high byte | 32 μs | 64 μs | 128 μs | 256 μs | 512 μs | 1.02 ms | 2.05 ms | 4.10 ms |
| 1 0 0 1 Address/Data switches 1 and 2 | Switch 1 8 | Switch 1 4 | Switch 1 2 | Switch 1 1 | Switch 2 8 | Switch 2 4 | Switch 2 2 | Switch 2 1 |
| 1 0 1 0 CPU error byte | SDR P check | MOR P check | Storage gate P check | ALU gate P check | Control storage invalid addr/ SAR check | Microloop time-out/ SAR check | Main storage invalid addr/ MSAR check | Main storage exception/ MSAR check |
| 1 0 1 1 PCR | Flag | Plus | Minus | Zero | Carry log | High log | Low log | Equal log |

## I/O Storage (WTCL, WTCH, RDCL, RDCH, WTM, RDM)

WTCL (I/O load from control storage low)

WTCH (I/O load from control storage high)

RDCL (I/O store to control storage low)

RDCH (I/O store to control storage high)

WTM (I/O load from main storage)

RDM (I/O store to main storage)

| 0 1 0 0 | Modifier | 0 | W | C | D | V | Reg 2 |
|---|---|---|---|---|---|---|---|
| 0 3 | 4 7 | 8 | 9 | 10 | 11 | 12 | 13 15 |

This instruction moves 1 byte of data between either main storage or control storage and the I/O attachment.

*Modifier (Bits 4-7):* Specifies the control field for the I/O attachment. The field is moved to the attachment through the port. Bit 4 of this field is used in the control processor to select the high- or low-order byte of control storage. When main storage is being addressed, bit 4 is not used by the control processor.

*Bit 8:* Changes the operation code (bits 0-3). Bit 8 is always a 0.

*W (Bit 9):* Identifies the direction the data is to be moved.

W = 0: Read data from storage and move it to the I/O attachment

W = 1: Write data to storage from the I/O attachment

*C (Bit 10):* Selects main storage or control storage.

C = 0: Main storage

C = 1: Control storage

*D (Bit 11):* Indicates if the address in the local storage register (specified by bits 13-15) is to be increased or decreased.

D = 0: Increase the selected local storage register by the value of field V

D = 1: Decrease the selected local storage register by the value of field V

*Note:* Bits 8-11 are sent to the port where they are decoded as either the load command or the sense command. The command is then sent to the I/O attachment on the 'command bus out' lines.

*V (Bit 12):* Indicates the amount the address in the local storage register (specified by bits 13-15) should be increased or decreased. If V = 0, the address in the selected local storage register is not changed. If V = 1, the address in the selected local storage register is decreased or increased by 1, as specified by the bit setting of field D.

*Register 2 (Bits 13-15):* Selects one of the eight local storage registers assigned to the current interrupt level that contains the storage address needed to move the data. The address in the specified local storage register may be updated as specified by bit 11 (field D) and bit 12 (field V).

*Condition Code*

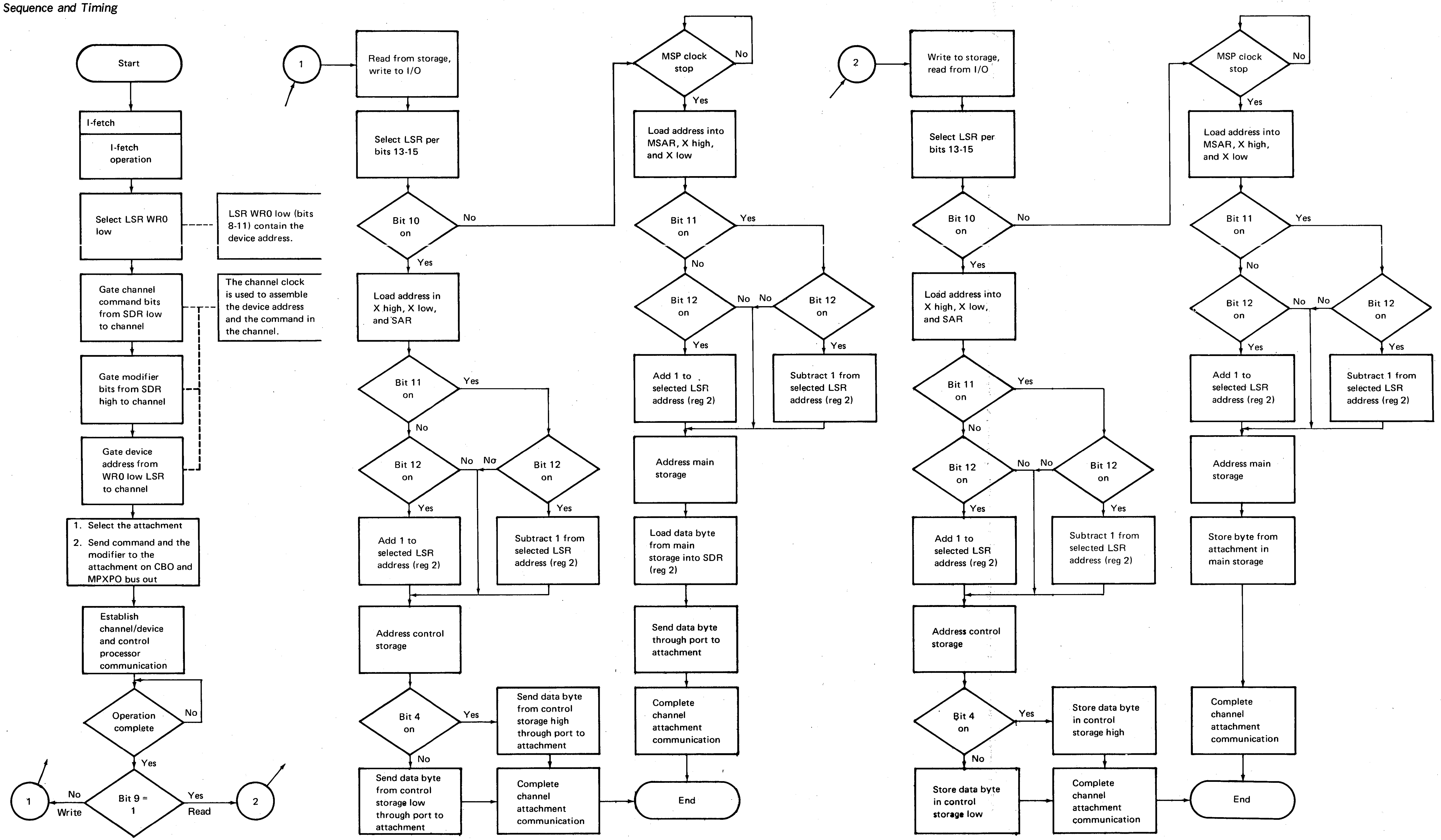
No change

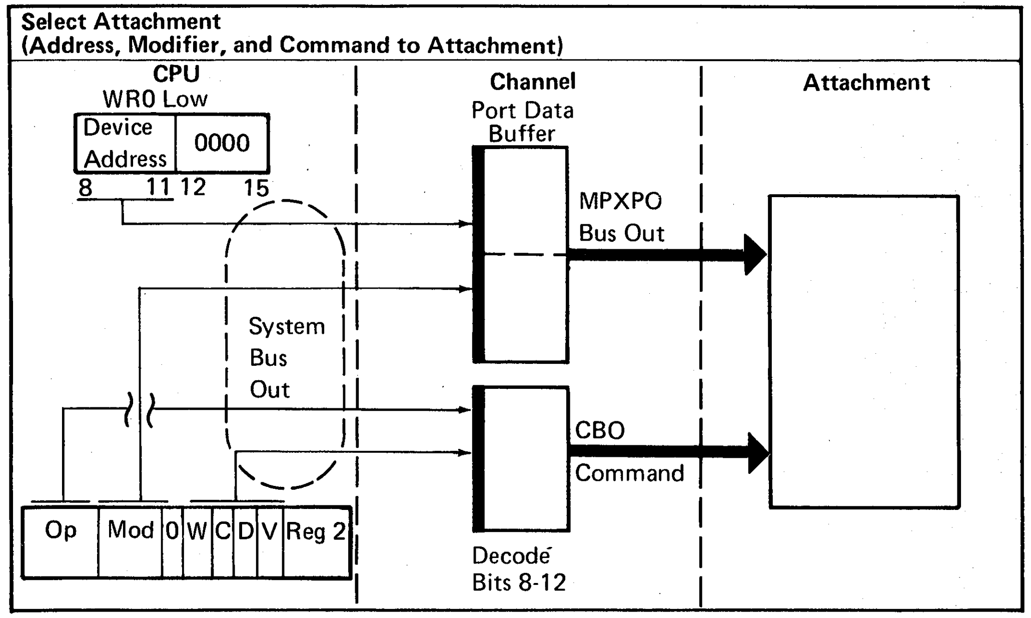
*Note:* A more complete description of the I/O storage commands may be found under *Commands* in the *Channel* section of this manual.

**Instruction List**

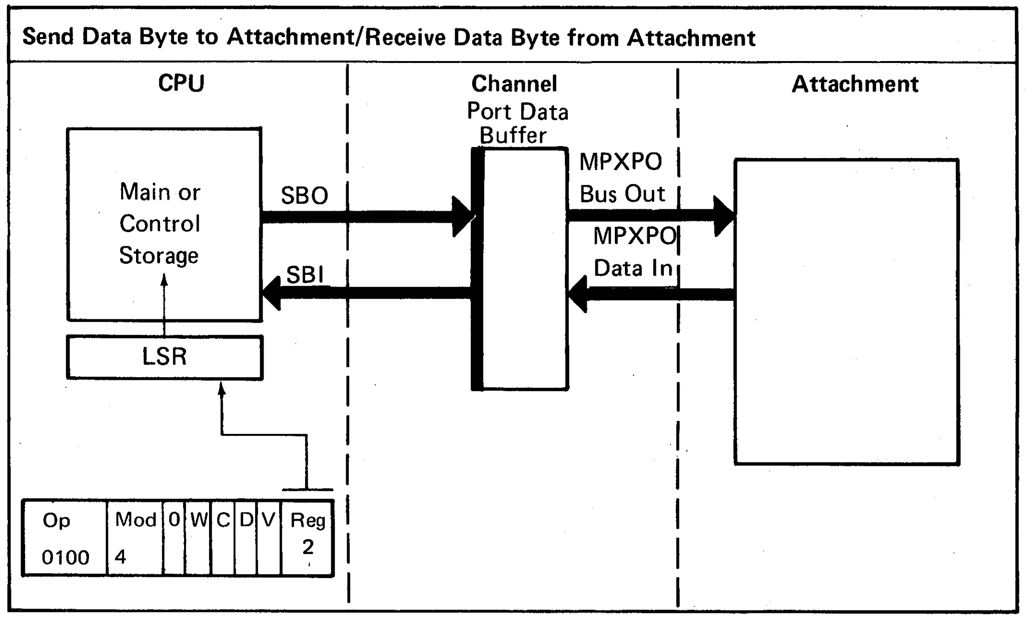
| **Bits**<br>**4 8 9 10 11 12** | **Mnemonic** | **Description** |
|---|---|---|
| 0 0 1 1 0 1 | RDCL | I/O store to control storage, increase register 2 by 1 |
| 1 0 1 1 0 1 | RDCH | |
| 0 0 1 1 1 1 | RDCL | I/O store to control storage, decrease register 2 by 1 |
| 1 0 1 1 1 1 | RDCH | |
| 0 0 1 1 0 0 | RDCL | I/O store to control storage, no change to register 2 |
| 1 0 1 1 0 0 | RDCH | |
| 0 0 0 1 0 1 | WTCL | I/O load from control storage, increase register 2 by 1 |
| 1 0 0 1 0 1 | WTCH | |
| 0 0 0 1 1 1 | WTCL | I/O load from control storage, decrease register 2 by 1 |
| 1 0 0 1 1 1 | WTCH | |
| 0 0 0 1 0 0 | WTCL | I/O load from control storage, no change to register 2 |
| 1 0 0 1 0 0 | WTCH | |
| X 0 1 0 0 1 | RDM | I/O store to main storage, increase register 2 by 1 |
| X 0 1 0 1 1 | RDM | I/O store to main storage, decrease register 2 by 1 |
| X 0 1 0 0 0 | RDM | I/O store to main storage, no change to register 2 |
| X 0 0 0 0 1 | WTM | I/O load from main storage, increase register 2 by 1 |
| X 0 0 0 1 1 | WTM | I/O load from main storage, decrease register 2 by 1 |
| X 0 0 0 0 0 | WTM | I/O load from main storage, no change to register 2 |

Legend for Bit 4:
X: Not used


Start
I-fetch
I-fetch operation
Select LSR WR0 low
LSR WR0 low (bits 8-11) contain the device address.
Gate channel command bits from SDR low to channel
The channel clock is used to assemble the device address and the command in the channel.
Gate modifier bits from SDR high to channel
Gate device address from WR0 low LSR to channel
1. Select the attachment
2. Send command and the modifier to the attachment on CBO and MPXPO bus out
Establish channel/device and control processor communication
Operation complete
No
Yes
Bit 9 = 1
No
Write
1
Yes
Read
2
1
Read from storage, write to I/O
Select LSR per bits 13-15
Bit 10 on
No
Yes
Load address in X high, X low, and SAR
Bit 11 on
Yes
No
Bit 12 on
No
No
Bit 12 on
Yes
Yes
Add 1 to selected LSR address (reg 2)
Subtract 1 from selected LSR address (reg 2)
Address control storage
Bit 4 on
Yes
No
Send data byte from control storage high through port to attachment
Send data byte from control storage low through port to attachment
Complete channel attachment communication
MSP clock stop
No
Yes
Load address into MSAR, X high, and X low
Bit 11 on
Yes
No
Bit 12 on
No
No
Bit 12 on
Yes
Yes
Add 1 to selected LSR address (reg 2)
Subtract 1 from selected LSR address (reg 2)
Address main storage
Load data byte from main storage into SDR (reg 2)
Send data byte through port to attachment
Complete channel attachment communication
End
2
Write to storage, read from I/O
Select LSR per bits 13-15
Bit 10 on
No
Yes
Load address into X high, X low, and SAR
Bit 11 on
Yes
No
Bit 12 on
No
No
Bit 12 on
Yes
Yes
Add 1 to selected LSR address (reg 2)
Subtract 1 from selected LSR address (reg 2)
Address control storage
Bit 4 on
Yes
No
Store data byte in control storage high
Store data byte in control storage low
Complete channel attachment communication
MSP clock stop
No
Yes
Load address into MSAR, X high, and X low
Bit 11 on
Yes
No
Bit 12 on
No
No
Bit 12 on
Yes
Yes
Add 1 to selected LSR address (reg 2)
Subtract 1 from selected LSR address (reg 2)
Address main storage
Store byte from attachment in main storage
Complete channel attachment communication
End

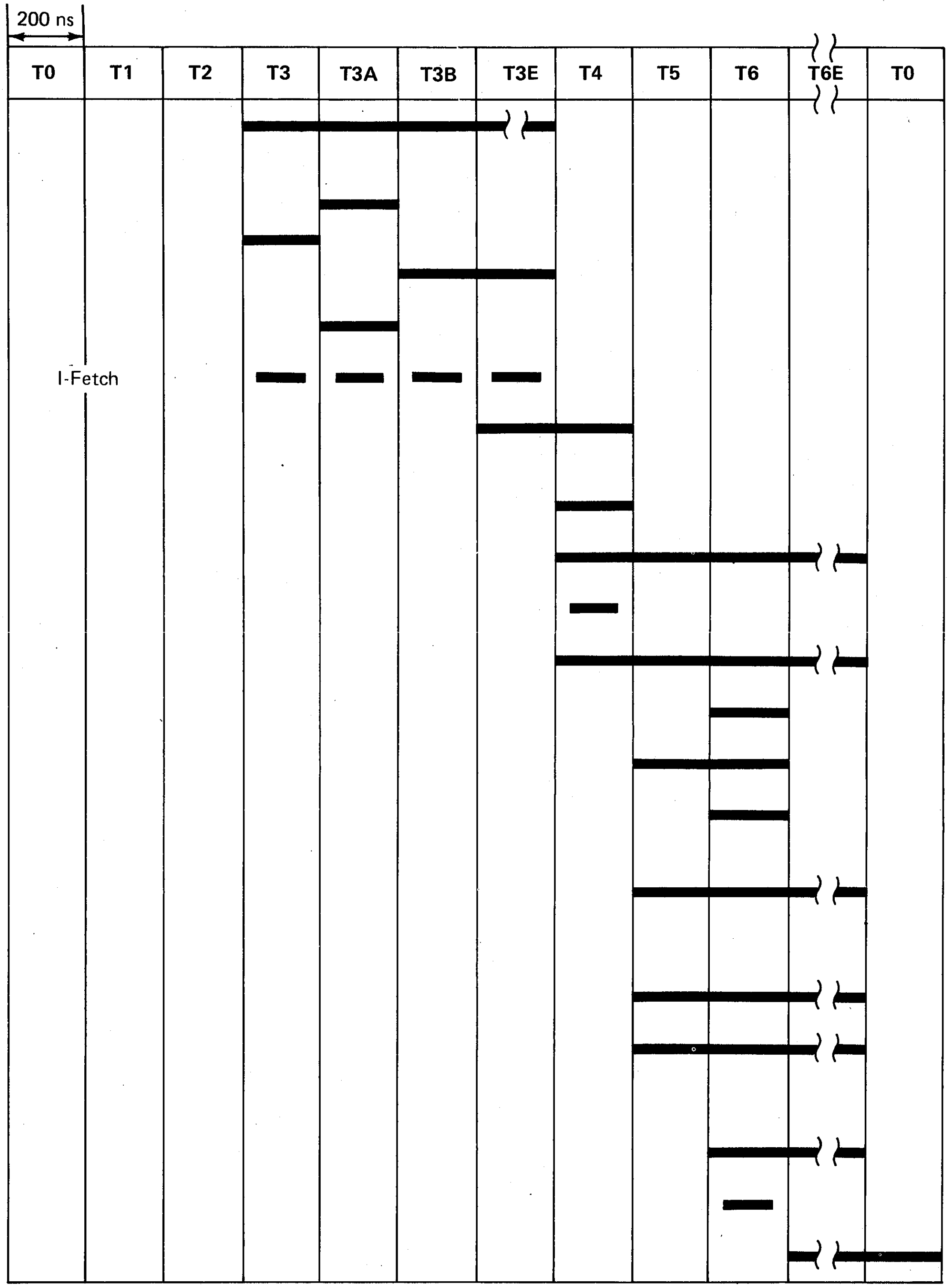


Select Attachment
(Address, Modifier, and Command to Attachment)

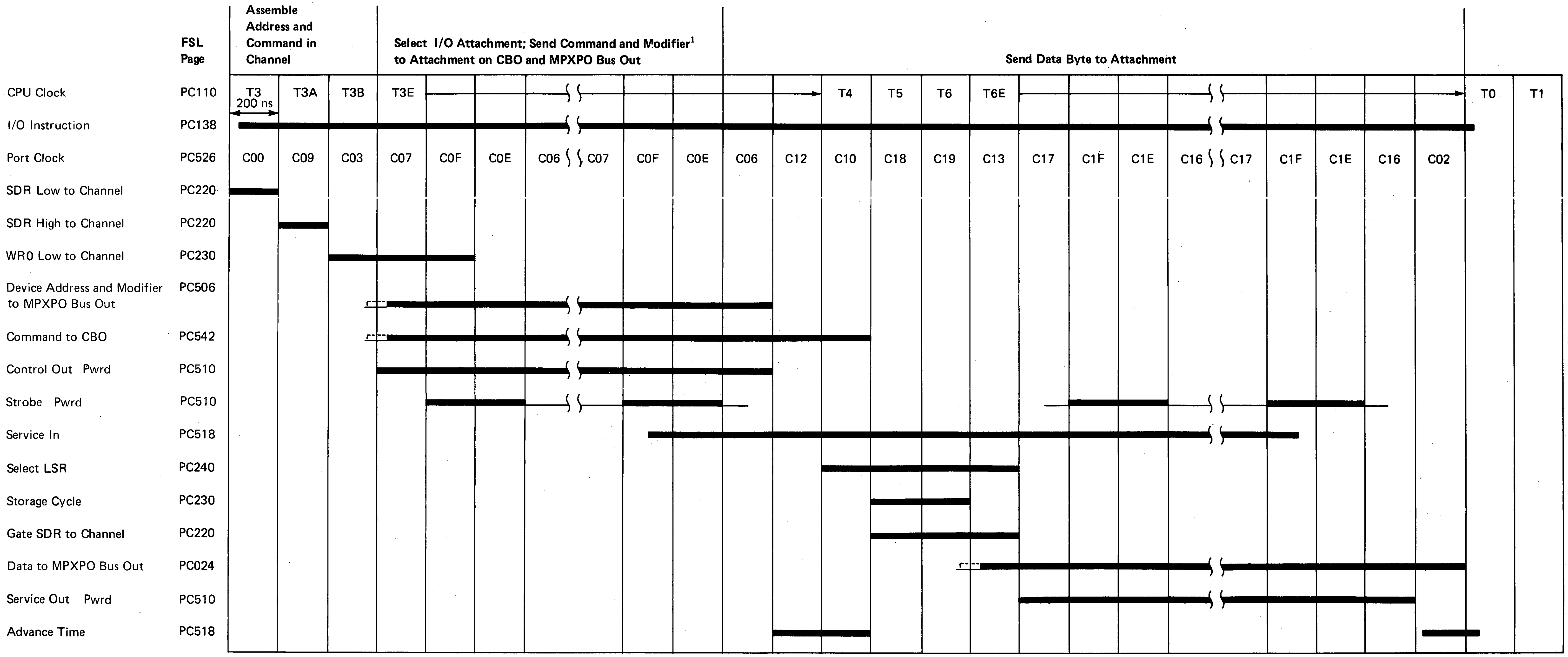


Send Data Byte to Attachment/Receive Data Byte from Attachment

| Timing of CP Functions | FSL Page |
|---|---|
| Select LSR(WR0) | PC240 |
| Select Storage Gate Low (from storage gate high, from SDR low, from LSR low) | PC230 |
| Select Storage Gate High (from SDR high) | PC230 |
| Clock X Low, X High, SAR | PC210 |
| Advance Time | PC518 |
| Select Storage Gate High/Low (from LSR high/low) | PC240 |
| Select LSR (bits 13, 14, 15) | PC230 |
| Clock X Low, X High, SAR | PC210 |
| ALU (± 1 or pass) | PC260 |
| ALU Gate High/Low (from ALU high/low) | PC250 |
| Storage Cycle | PC012 |
| Clock SDR (write trigger) | PC220 |
| Select Storage Gate High (from SDR high: 9=0; from channel bus: 9=1) | PC230 |
| Select Storage Gate Low (from channel bus: 9=1; from low: 4, 9=0 or 9, 10=0; from storage gate high: 4, 10=1; 9=0) | PC230 |
| Write Storage High (4, 9, 10=1) | PC134 |
| Write Storage Low (4=0, 9=1 or 9=1, 10=0) | PC134 |
| ALU Gate High/Low (from ALU high/low) | PC250 |
| Write LSR High/Low (write trigger/phase B) | PC160 |
| Advance Time | PC518 |



*Timing of CP/Channel Functions*



[1] See *Channel Exerciser Loop Program* in the *Channel* section of this manual for a program that can be used with this command.

## Jump on I/O Condition (JIO)

| 0 0 1 1 | Modifier | Page Address |
|---|---|---|
| 0 3 | 4 7 | 8 15 |

This instruction tests I/O conditions. If the condition tested is active, this instruction causes a jump to the address specified by the page address (bits 8-15). If the condition tested is not active, the next sequential instruction is executed.

The operation code (bits 0-3) is sent to the port where the bits are decoded as a jump-on-I/O-condition command. This command is then sent to the I/O attachment through the port.

*Modifier (Bits 4-7):* Specifies the control field for the I/O devices. The I/O device being used determines how this field is used. The modifier field is moved to the I/O attachment through the port.

Some of the modifier combinations make a common code for those conditions that are used by most I/O attachments. The modifier usage is specified as follows:

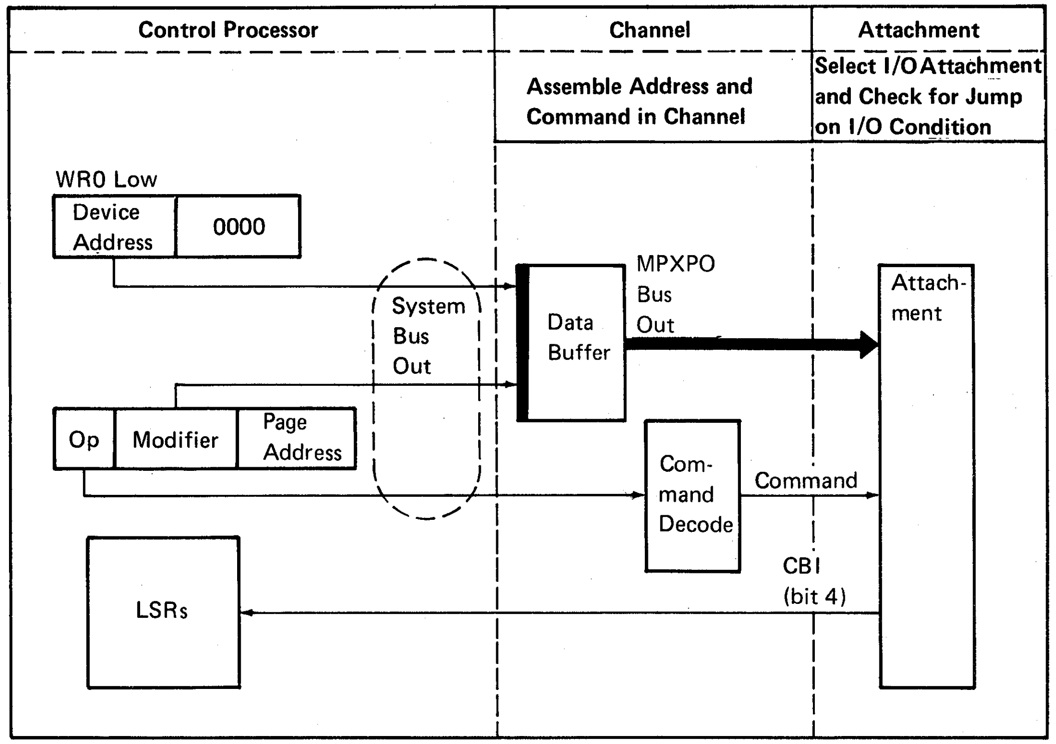
| **Modifier Field Setting 4 5 6 7** | **Description** |
|---|---|
| 0 0 0 0 | Adapter check |
| 0 0 0 1 | Adapter not ready |
| 0 0 1 0 | Busy condition 1 |
| 0 0 1 1 | Busy condition 2 |
| 0 1 0 0 | Interrupt enabled |
| 0 1 0 1 | Diagnostic real |
| 0 1 1 0 | Diagnostic not real |
| 0 1 1 1 through 1 1 1 1 | Available for I/O attachment needs |

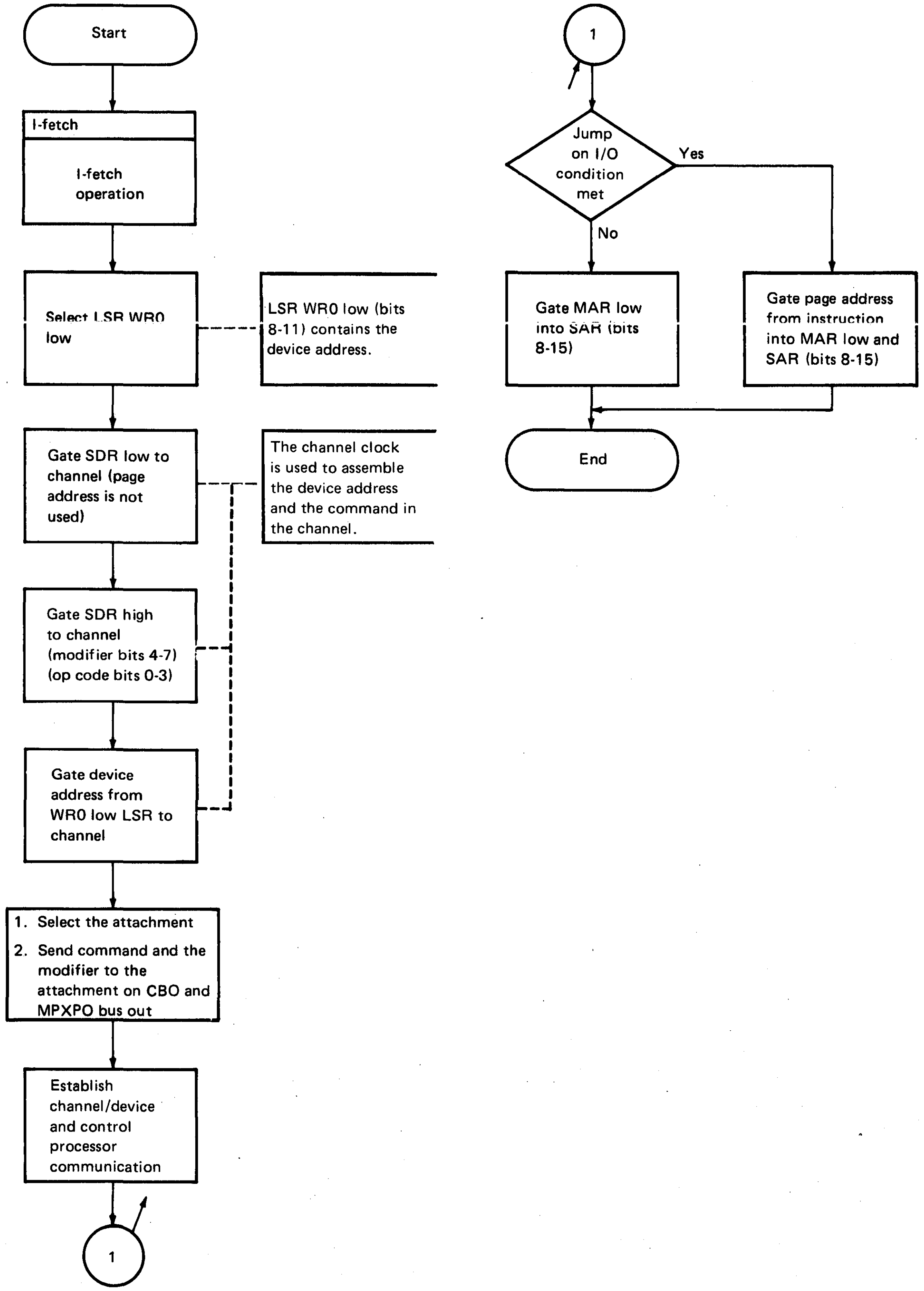
*Page Address (Bits 8-15):* Permits a jump inside a page boundary (256-word limits hex 00 through hex FF) in control storage only. The page address must be located on the same page boundary as the jump on I/O condition. This field replaces the 8 low-order bits in the microaddress register if the I/O device indicates that the jump condition is met. The 'CBI bit 4' port line determines if the I/O condition is met.

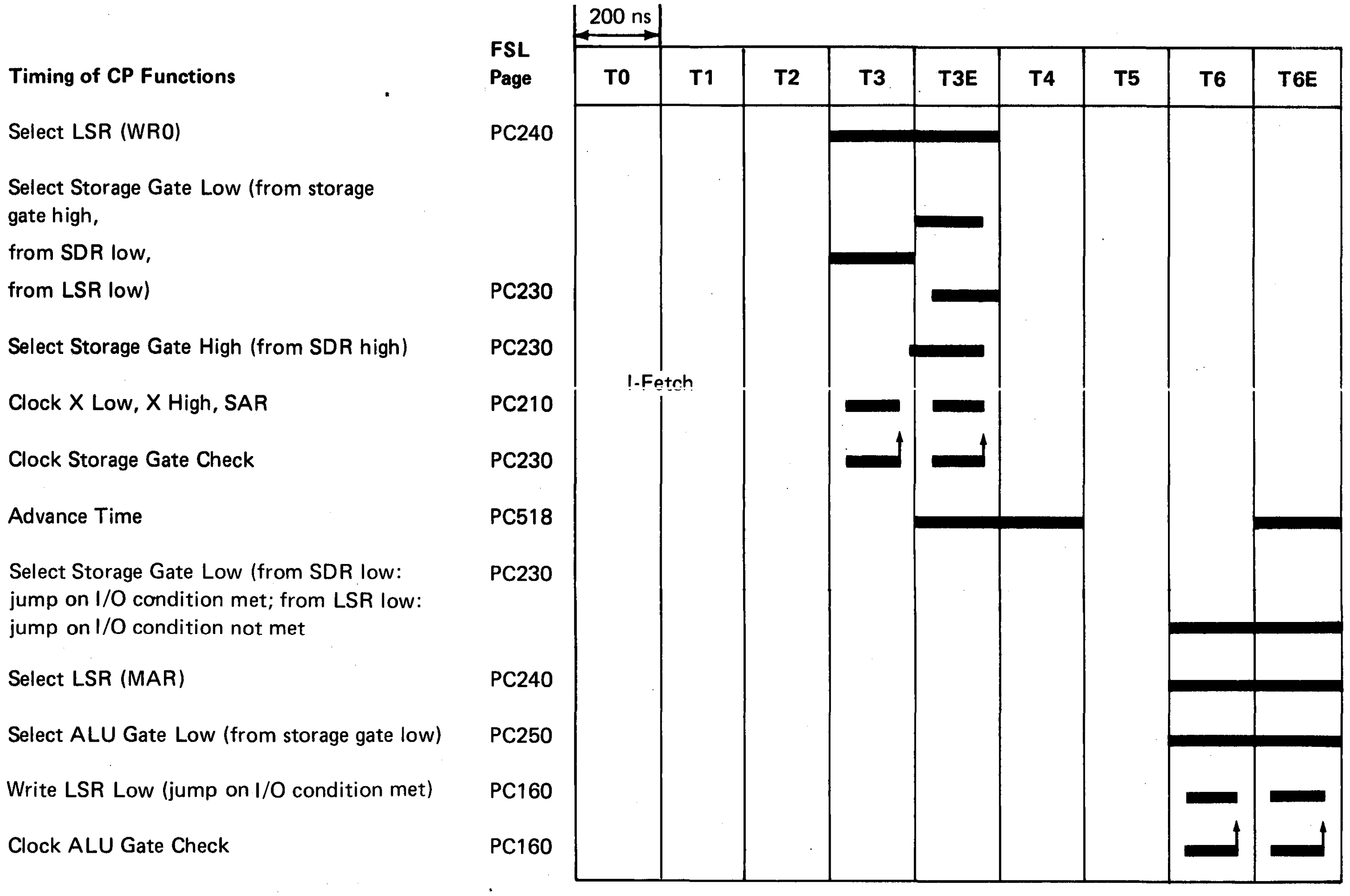
*Condition Code*

No change

*Note:* A more complete description of the jump-on-I/O-condition command may be found under *Commands* in the *Channel* section of this manual.

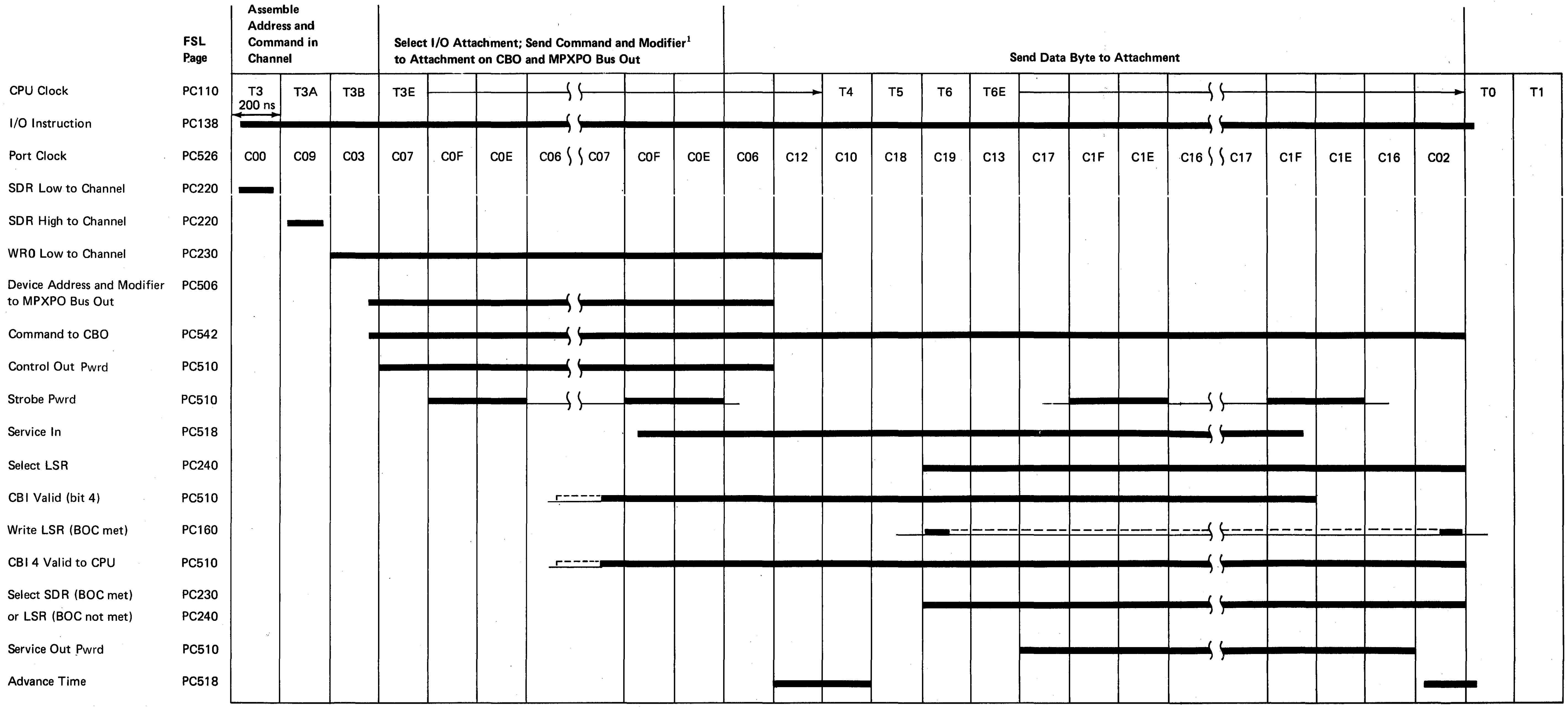



Start
I-fetch
I-fetch operation
Select LSR WR0 low
LSR WR0 low (bits 8-11) contains the device address.
Gate SDR low to channel (page address is not used)
The channel clock is used to assemble the device address and the command in the channel.
Gate SDR high to channel (modifier bits 4-7) (op code bits 0-3)
Gate device address from WR0 low LSR to channel
1. Select the attachment
2. Send command and the modifier to the attachment on CBO and MPXPO bus out
Establish channel/device and control processor communication
1
1
Jump on I/O condition met
Yes
No
Gate MAR low into SAR (bits 8-15)
Gate page address from instruction into MAR low and SAR (bits 8-15)
End



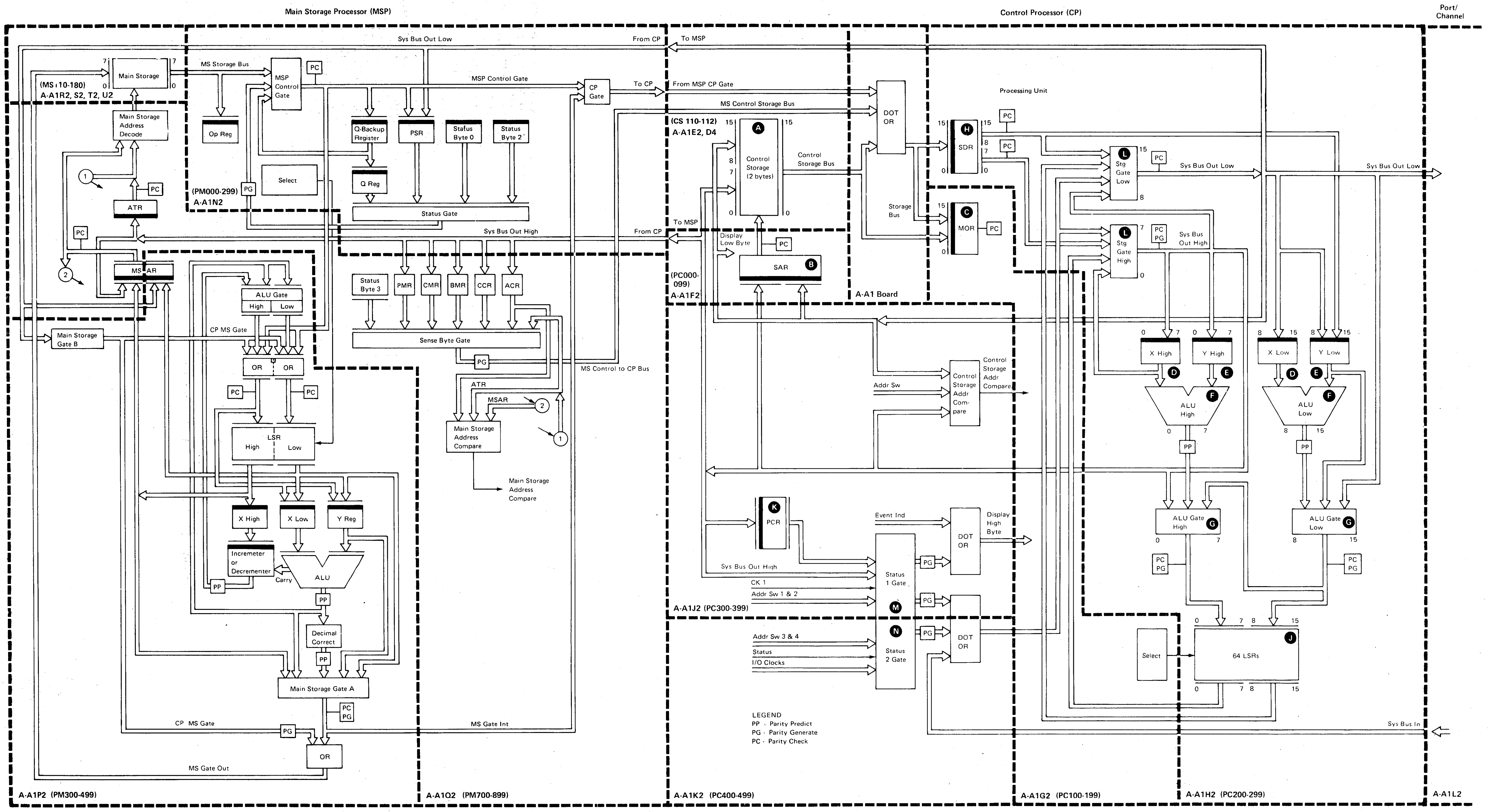
| Timing of CP Functions | FSL Page |
|---|---|
| Select LSR (WR0) | PC240 |
| Select Storage Gate Low (from storage gate high,<br>from SDR low,<br>from LSR low) | PC230 |
| Select Storage Gate High (from SDR high) | PC230 |
| Clock X Low, X High, SAR | PC210 |
| Clock Storage Gate Check | PC230 |
| Advance Time | PC518 |
| Select Storage Gate Low (from SDR low: jump on I/O condition met; from LSR low: jump on I/O condition not met | PC230 |
| Select LSR (MAR) | PC240 |
| Select ALU Gate Low (from storage gate low) | PC250 |
| Write LSR Low (jump on I/O condition met) | PC160 |
| Clock ALU Gate Check | PC160 |

This page intentionally left blank.


FSL Page
Assemble Address and Command in Channel
Select I/O Attachment; Send Command and Modifier[1] to Attachment on CBO and MPXPO Bus Out
Send Data Byte to Attachment
CPU Clock PC110
T3 T3A T3B T3E T4 T5 T6 T6E T0 T1
200 ns
I/O Instruction PC138
Port Clock PC526
C00 C09 C03 C07 C0F C0E C06 C07 C0F C0E C06 C12 C10 C18 C19 C13 C17 C1F C1E C16 C17 C1F C1E C16 C02
SDR Low to Channel PC220
SDR High to Channel PC220
WR0 Low to Channel PC230
Device Address and Modifier to MPXPO Bus Out PC506
Command to CBO PC542
Control Out Pwrd PC510
Strobe Pwrd PC510
Service In PC518
Select LSR PC240
CBI Valid (bit 4) PC510
Write LSR (BOC met) PC160
CBI 4 Valid to CPU PC510
Select SDR (BOC met) PC230
or LSR (BOC not met) PC240
Service Out Pwrd PC510
Advance Time PC518

[1] See *Channel Exerciser Loop Program* in the *Channel* section of this manual for a program that can be used with this command.

# FUNCTIONAL UNITS



LEGEND
PP - Parity Predict
PG - Parity Generate
PC - Parity Check

*Data flow bus lines may not pass through FRUs as shown

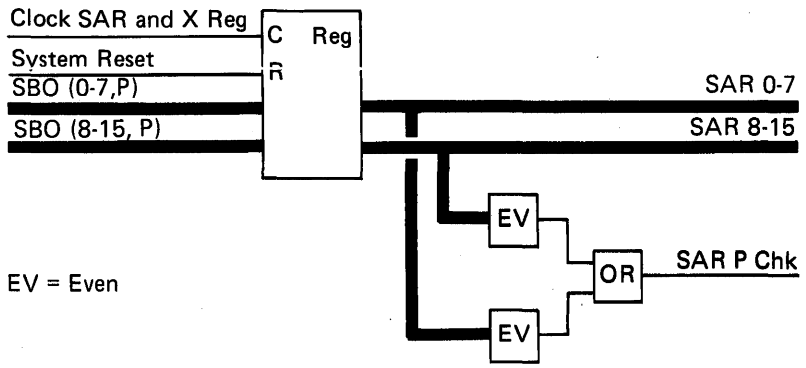
## Control Storage Ⓐ

Control storage contains 16K addresses; each address is 2 bytes wide. Control storage is loaded from the disk during normal operations, or from the diskette when diagnostic programs are being run. When control storage is loaded, it contains the control storage programs that are used to support system programs.

## Storage Address Register Ⓑ

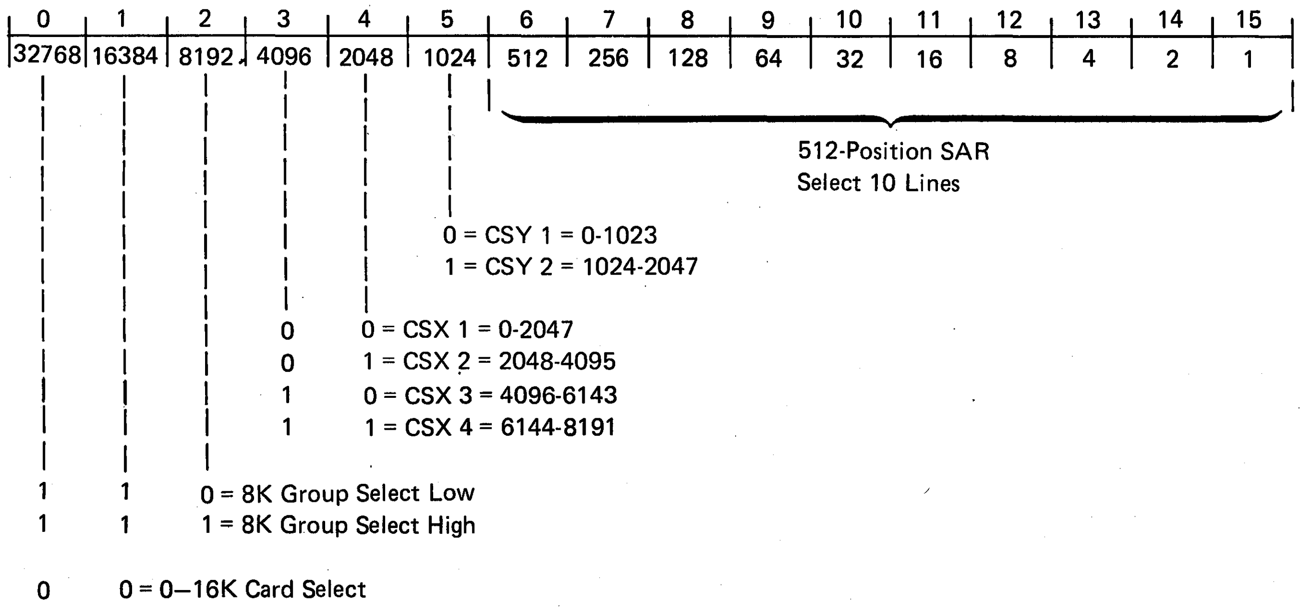
The storage address register (SAR) is a 16-bit register used to address control storage. This register holds all storage addresses that are moved from the local storage register or generated from local storage register, X high register, or storage data register data. The data moved into the storage address register does not change during the storage cycle.

**Storage Address Register**



PC002

**SAR Decoding Control Storage**



## Micro-Operation Register Ⓒ

The micro-operation register (MOR) is a 16-bit register that holds each control storage instruction as it is taken from control storage. The instruction is decoded to control the data flow (for example, gate selection, arithmetic and logic unit operations, local storage register selection, and setting the processor condition register).

## X-Registers and Y-Registers Ⓓ and Ⓔ

These four registers are the buffer input to the two control processor arithmetic and logic units (ALU). The X-high and Y-high registers are input to ALU high and the X-low and Y-low registers are input to ALU low.

The X-registers are buffers for base constants into the ALU.

The Y-registers are buffers for changing constants into the ALU.

## Arithmetic and Logic Units Ⓕ

There are two arithmetic and logic units (ALUs) in the control processor. ALU high uses bits 0-7 when 2-byte data fields are used. ALU low uses bits 8-15 when either 1-byte or 2-byte data fields are used. The ALUs always send 2 bytes of data to the local storage register (LSR) input bus. When 2 bytes are used in the ALU operation, both bytes (high and low) are placed on the LSR input bus and are, at the same time, written into bits 0-7 and bits 8-15 of the LSR. When the ALU output is only 1 byte, the byte is sent to both the high and low LSR input bus lines. In these cases, the instruction selects only 1 byte to be written into an LSR. The ALU performs the following functions:

| Function | Function Bits | | | | Carry In |
|---|---|---|---|---|---|
| | F0 | F1 | F2 | F3 | |
| Y→X(ZAR) | 0 | 0 | 0 | 0 | * |
| X XR Y | 0 | 0 | 0 | 1 | * |
| X OR Y | 0 | 0 | 1 | 1 | * |
| X AND (not) Y | 0 | 1 | 0 | 1 | * |
| X AND Y | 0 | 1 | 1 | 0 | * |
| X OR (not) Y | 0 | 1 | 1 | 1 | * |
| X minus one | 1 | 0 | 0 | 0 | 0 |
| X plus Y plus-carry | 1 | 0 | 0 | 1 | C |
| X minus Y (16/8) | 1 | 0 | 1 | 0 | 1 |
| X plus Y (16 or 8) | 1 | 0 | 1 | 1 | 0 |
| X minus Y minus (16 or 8) | 1 | 1 | 0 | 0 | 1 |
| X plus Y (16/8) | 1 | 1 | 0 | 1 | 0 |
| X minus Y | 1 | 1 | 1 | 0 | C |
| X plus one (carry in) | 1 | 1 | 1 | 1 | 1 |

Legend for Function:

16/8–First field 16 bits, second field 8 bits

16 or 8–Both fields 16 bits or both fields 8 bits

Legend for Carry In:

C = Carry used (carry trigger from an earlier operation)

1 = Force carry to 1 (by hardware, T-times, and instruction)

0 = Force carry to 0

* = Not used

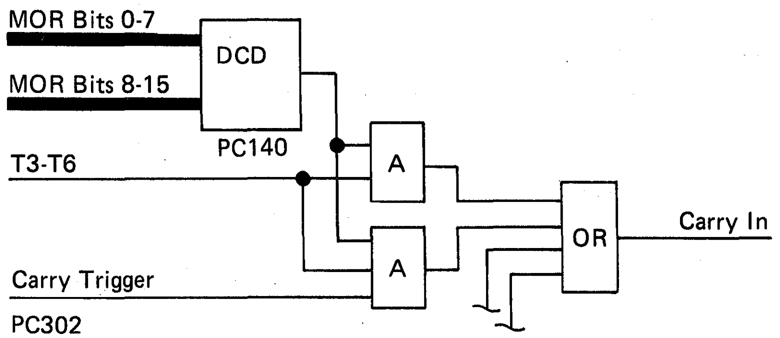
Any data sent to the ALU is first loaded into the X-high and Y-high registers for the low bytes. The X-registers supply the data for one operand, and the Y-registers supply the data for the other operand that is used in the current ALU operation. The instruction and its function determine if 1 or 2 bytes are affected by the ALU.

The ALU does arithmetic operations with two 16-bit words, one 16-bit word plus or minus one 8-bit byte, or one 8-bit byte plus or minus one 8-bit byte. The instruction, logical/arithmetic 1, is used for 8-bit by 8-bit arithmetic operations. The logical/arithmetic 2 instruction is used for 16-bit by 16-bit arithmetic and 16-bit by 8-bit arithmetic operations. In 16-bit by 8-bit arithmetic, the 'reset Y high reg' line (generated on the data flow card) resets the 8 bits of the Y-register that are not used.

Instructions that cause an increase or decrease of the X-register contents are executed by resetting the Y-high and Y-low registers and then forcing a carry in to the ALU. This causes only the X-register to be affected by the instruction.

The output of the ALU always sends 2 bytes of data to the LSR stack input bus. If 2 bytes are needed by the ALU operation, both bytes are placed directly on the LSR input bus and are, at the same time, written into the LSR stack. If only 1 byte was operated on by the ALU, the result (1 byte) is sent to both the high and low input buses. Only the byte selected by the instruction is written into the LSR stack.

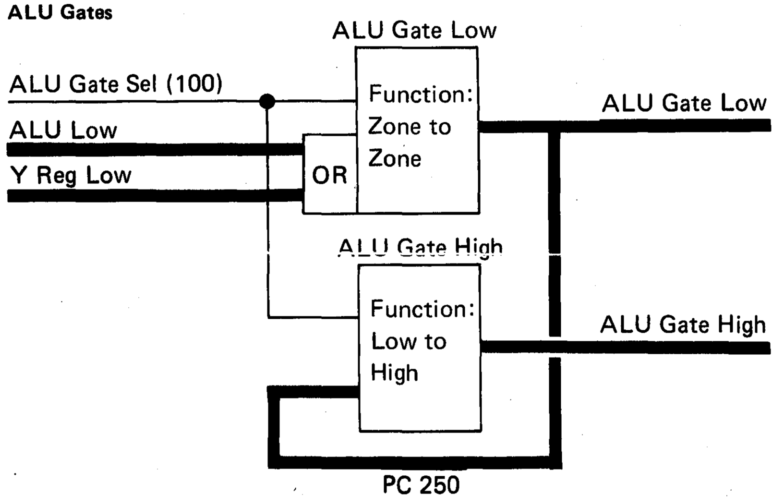
Carry In



### Arithmetic and Logic Unit Gates G

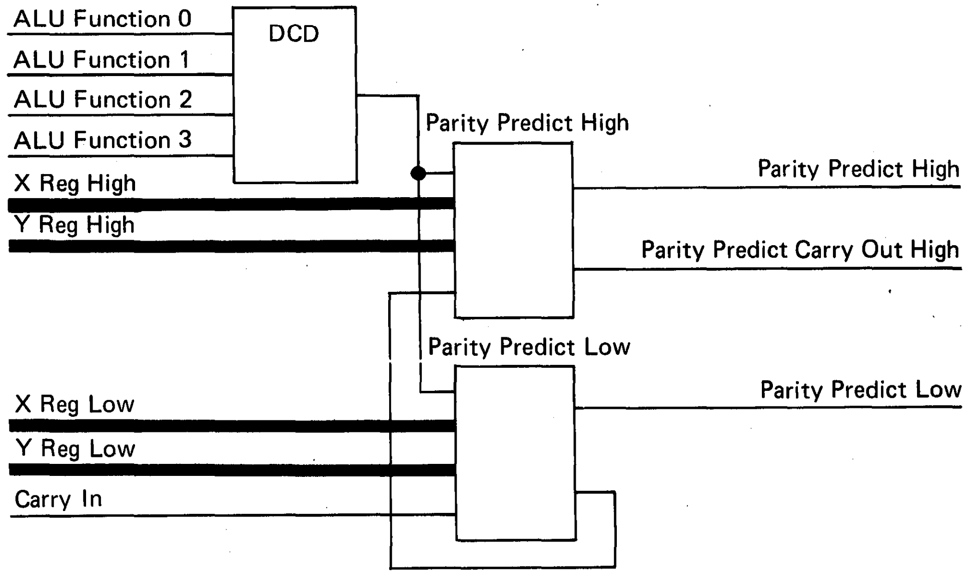
ALU gate high and ALU gate low control the path of the ALU data. The lines that select the data path are generated by a decode of the micro-operation register bits and the T-times.

ALU Gates



### Arithmetic and Logic Unit Parity Predict

Parity Predict Circuits



Parity predict circuits predict the parity of the result of the ALU operation. This predicted parity is compared against the parity generated. If there is a difference, a parity check results.

## Storage Data Register Ⓗ

The storage data register (SDR) is a 16-bit register that is an intermediate buffer for all instructions and data bytes taken from control storage and main storage (under control of the control processor or I/O operations). Each instruction is 2 bytes wide and, therefore, uses all 16 bit positions.

The storage data register high-order bits (0-7) are gated through the storage gate high register to the high-order X-register and Y-register and then to the arithmetic and logic unit (ALU). The storage data register low-order bits (8-15) are gated to the low-order X-register and Y-register and then to the ALU.

## Local Storage Registers ●J

The control processor uses the local storage registers (LSRs) as:

- Data buffers and address registers for control storage
- Operand registers for internal calculations
- I/O control registers that can be loaded from the I/O attachments or from which data can be sent to the I/O attachments

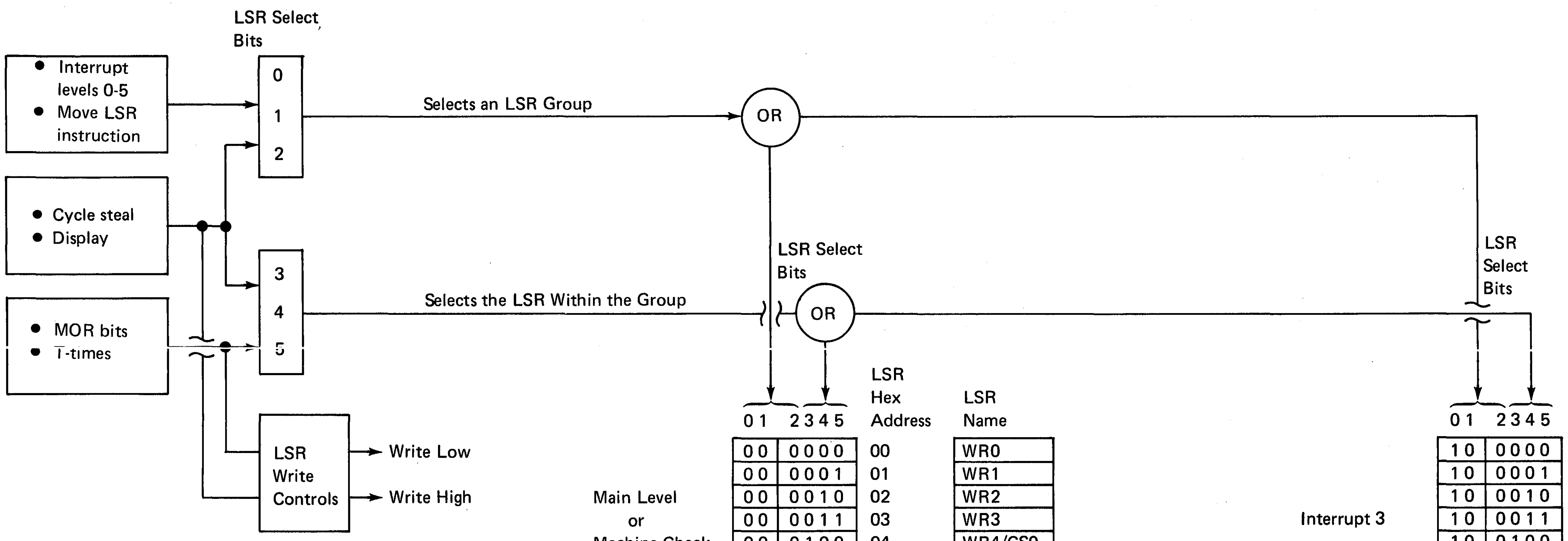
The local storage register stack contains 64 two-byte registers. Bits 0-7 of each register are the high local storage register and bits 8-15 of each register are the low local storage register.

The 64 local storage registers are divided into seven interrupt level groups. The current interrupt level determines which group is used.

The interrupt levels associated with the local storage registers are:

| Interrupt Level | Microaddress Register (Hex) | Microaddress Backup Register (Hex) | Work Register Physical (Hex) | Work Register Logical (Hex) |
|---|---|---|---|---|
| 0 | 0A | 0B | 00–07 | W0–W7 |
| 1 | 0C | 0D | 10–17 | W0–W7 |
| 2 | 0E | 0F | 18–1F | W0–W7 |
| 3 | 28 | 29 | 20–27 | W0–W7 |
| 4 | 2C | 2D | 30–37 | W0–W7 |
| 5 | 2E | 2F | 38–3F | W0–W7 |
| MPL | 08 | 09 | 00–07 | W0–W7 |

*Note:* Interrupt levels are shown in priority order.



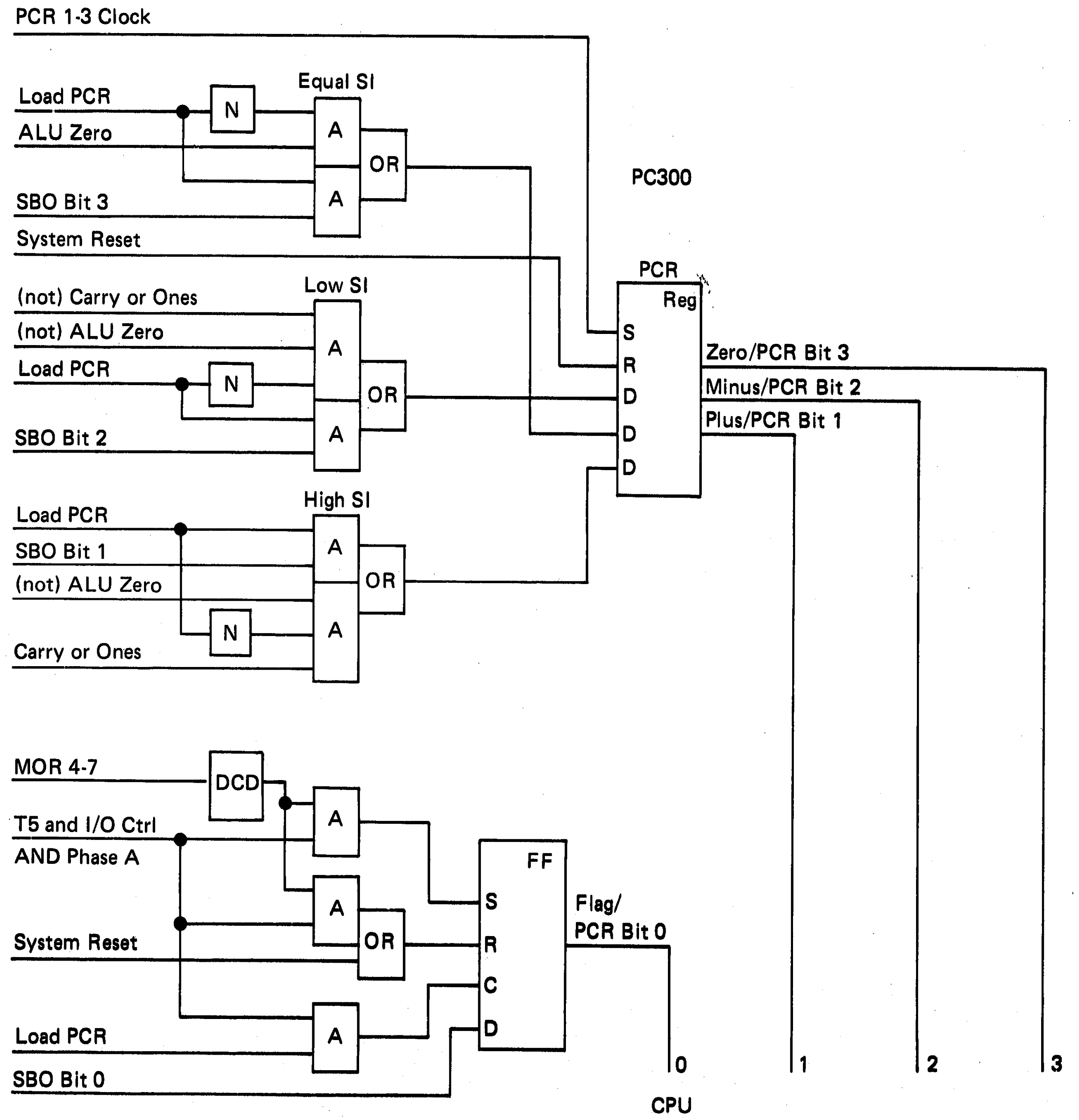
| | 0 1 | 2 3 4 5 | LSR Hex Address | LSR Name |
|---|---|---|---|---|
| Main Level or Machine Check 0 | 0 0 | 0 0 0 0 | 00 | WR0 |
| | 0 0 | 0 0 0 1 | 01 | WR1 |
| | 0 0 | 0 0 1 0 | 02 | WR2 |
| | 0 0 | 0 0 1 1 | 03 | WR3 |
| | 0 0 | 0 1 0 0 | 04 | WR4/CS0 |
| | 0 0 | 0 1 0 1 | 05 | WR5/CS1 |
| | 0 0 | 0 1 1 0 | 06 | WR6/CS2 |
| | 0 0 | 0 1 1 1 | 07 | WR7/CS3 |
| MAR/MAB Stack 1 | 0 0 | 1 0 0 0 | 08 | MAR |
| | 0 0 | 1 0 0 1 | 09 | MAB |
| | 0 0 | 1 0 1 0 | 0A | MAR (MC) |
| | 0 0 | 1 0 1 1 | 0B | MAB (MC) |
| | 0 0 | 1 1 0 0 | 0C | MAR-1 |
| | 0 0 | 1 1 0 1 | 0D | MAB-1 |
| | 0 0 | 1 1 1 0 | 0E | MAR-2 |
| | 0 0 | 1 1 1 1 | 0F | MAB-2 |
| Interrupt 1 | 0 1 | 0 0 0 0 | 10 | WR0 |
| | 0 1 | 0 0 0 1 | 11 | WR1 |
| | 0 1 | 0 0 1 0 | 12 | WR2 |
| | 0 1 | 0 0 1 1 | 13 | WR3 |
| | 0 1 | 0 1 0 0 | 14 | WR4/CS0 |
| | 0 1 | 0 1 0 1 | 15 | WR5/CS1 |
| | 0 1 | 0 1 1 0 | 16 | WR6/CS2 |
| | 0 1 | 0 1 1 1 | 17 | WR7/CS3 |
| Interrupt 2 | 0 1 | 1 0 0 0 | 18 | WR0 |
| | 0 1 | 1 0 0 1 | 19 | WR1 |
| | 0 1 | 1 0 1 0 | 1A | WR2 |
| | 0 1 | 1 0 1 1 | 1B | WR3 |
| | 0 1 | 1 1 0 0 | 1C | WR4/CS0 |
| | 0 1 | 1 1 0 1 | 1D | WR5/CS1 |
| | 0 1 | 1 1 1 0 | 1E | WR6/CS2 |
| | 0 1 | 1 1 1 1 | 1F | WR7/CS3 |

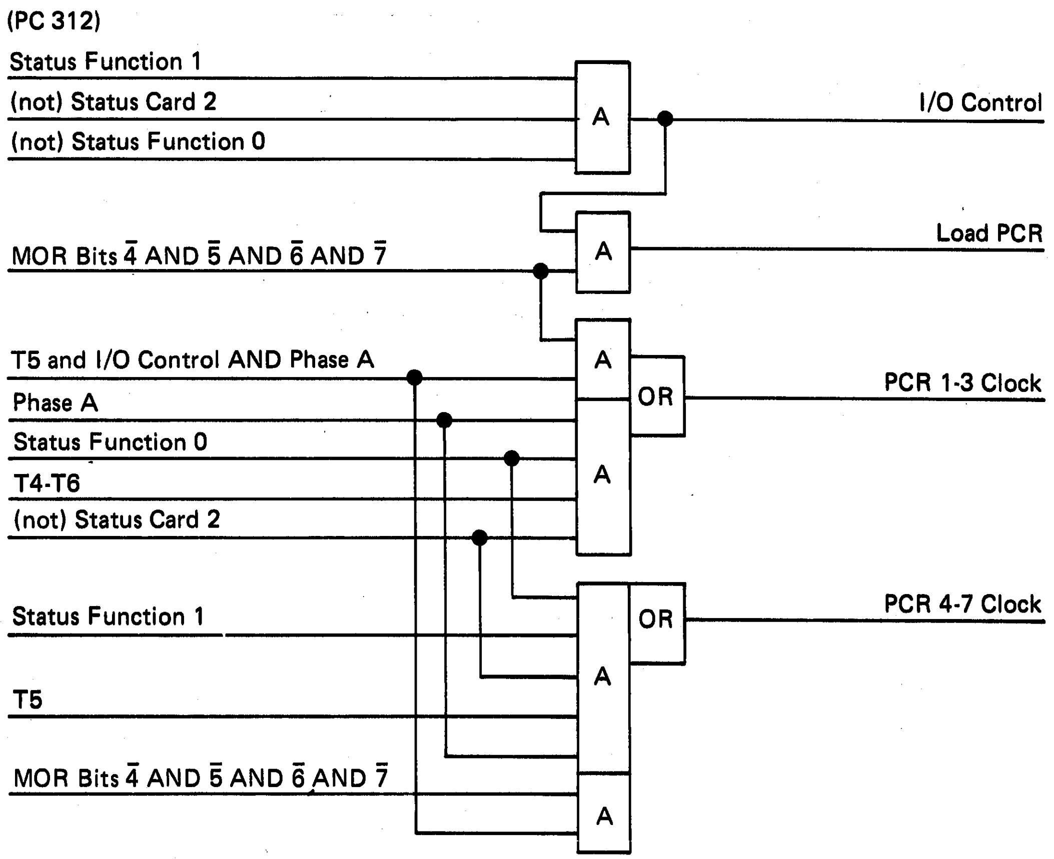
| | 0 1 | 2 3 4 5 | LSR Hex Address | LSR Name |
|---|---|---|---|---|
| Interrupt 3 | 1 0 | 0 0 0 0 | 20 | WR0 |
| | 1 0 | 0 0 0 1 | 21 | WR1 |
| | 1 0 | 0 0 1 0 | 22 | WR2 |
| | 1 0 | 0 0 1 1 | 23 | WR3 |
| | 1 0 | 0 1 0 0 | 24 | WR4/CS0 |
| | 1 0 | 0 1 0 1 | 25 | WR5/CS1 |
| | 1 0 | 0 1 1 0 | 26 | WR6/CS2 |
| | 1 0 | 0 1 1 1 | 27 | WR7/CS3 |
| MAR/MAB Stack 2 | 1 0 | 1 0 0 0 | 28 | MAR-3 |
| | 1 0 | 1 0 0 1 | 29 | MAB-3 |
| | 1 0 | 1 0 1 0 | 2A | Spare |
| | 1 0 | 1 0 1 1 | 2B | Spare |
| | 1 0 | 1 1 0 0 | 2C | MAR-4 |
| | 1 0 | 1 1 0 1 | 2D | MAB-4 |
| | 1 0 | 1 1 1 0 | 2E | MAR-5 |
| | 1 0 | 1 1 1 1 | 2F | MAB-5 |
| Interrupt 4 | 1 1 | 0 0 0 0 | 30 | WR0 |
| | 1 1 | 0 0 0 1 | 31 | WR1 |
| | 1 1 | 0 0 1 0 | 32 | WR2 |
| | 1 1 | 0 0 1 1 | 33 | WR3 |
| | 1 1 | 0 1 0 0 | 34 | WR4/CS0 |
| | 1 1 | 0 1 0 1 | 35 | WR5/CS1 |
| | 1 1 | 0 1 1 0 | 36 | WR6/CS2 |
| | 1 1 | 0 1 1 1 | 37 | WR7/CS3 |
| Interrupt 5 | 1 1 | 1 0 0 0 | 38 | WR0 |
| | 1 1 | 1 0 0 1 | 39 | WR1 |
| | 1 1 | 1 0 1 0 | 3A | WR2 |
| | 1 1 | 1 0 1 1 | 3B | WR3 |
| | 1 1 | 1 1 0 0 | 3C | WR4/CS0 |
| | 1 1 | 1 1 0 1 | 3D | WR5/CS1 |
| | 1 1 | 1 1 1 0 | 3E | WR6/CS2 |
| | 1 1 | 1 1 1 1 | 3F | WR7/CS3 |

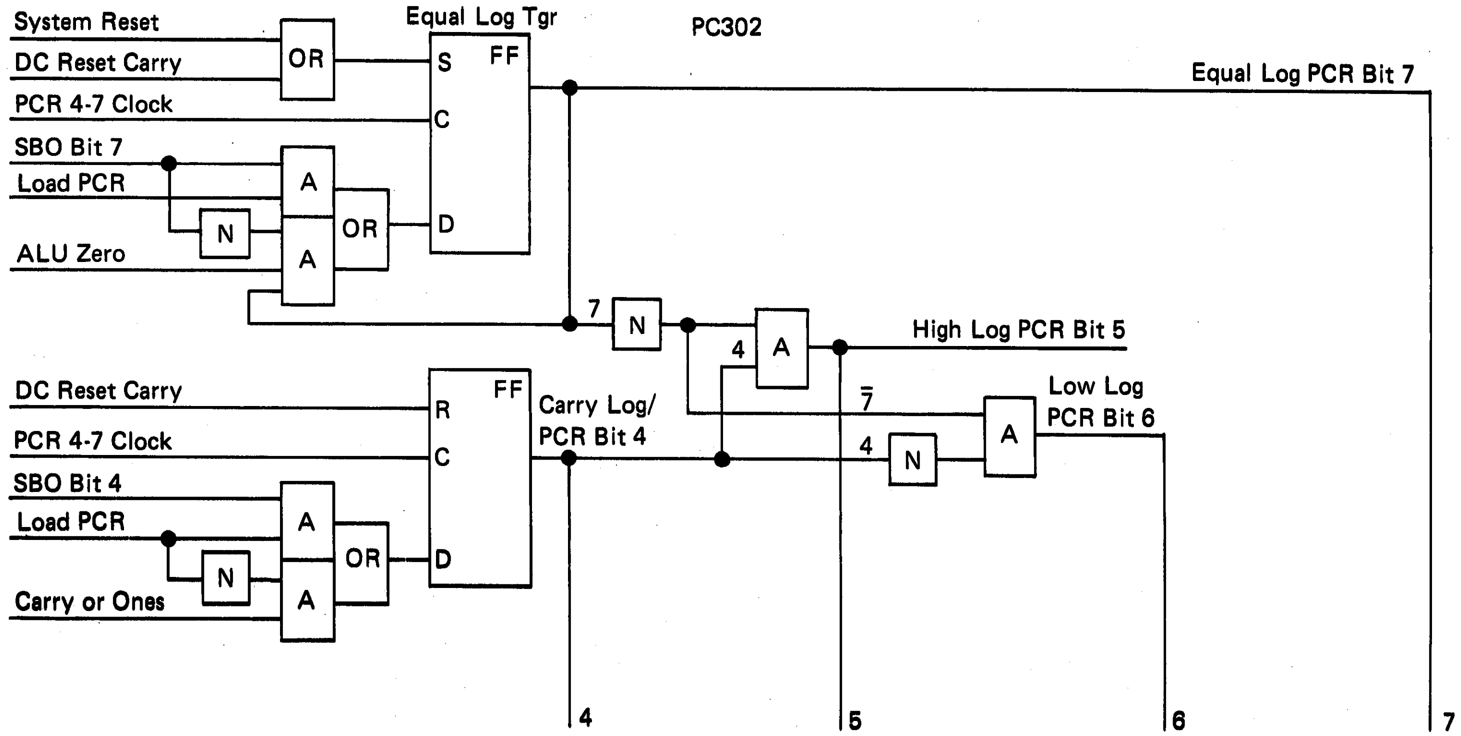
## Processor Condition Register Ⓚ

The processor condition register (PCR) contains the processor conditions that are tested by the jump-on-condition instruction. The processor condition register is changed by system reset, program loading, or instructions that change register bits. These conditions are changed by the instructions that perform the add, subtract, test mask, compare immediate, subtract immediate, and R1-linked-with-R2 functions.

The processor condition register clocks gate the data into the processor condition register.







## Processor Condition Register

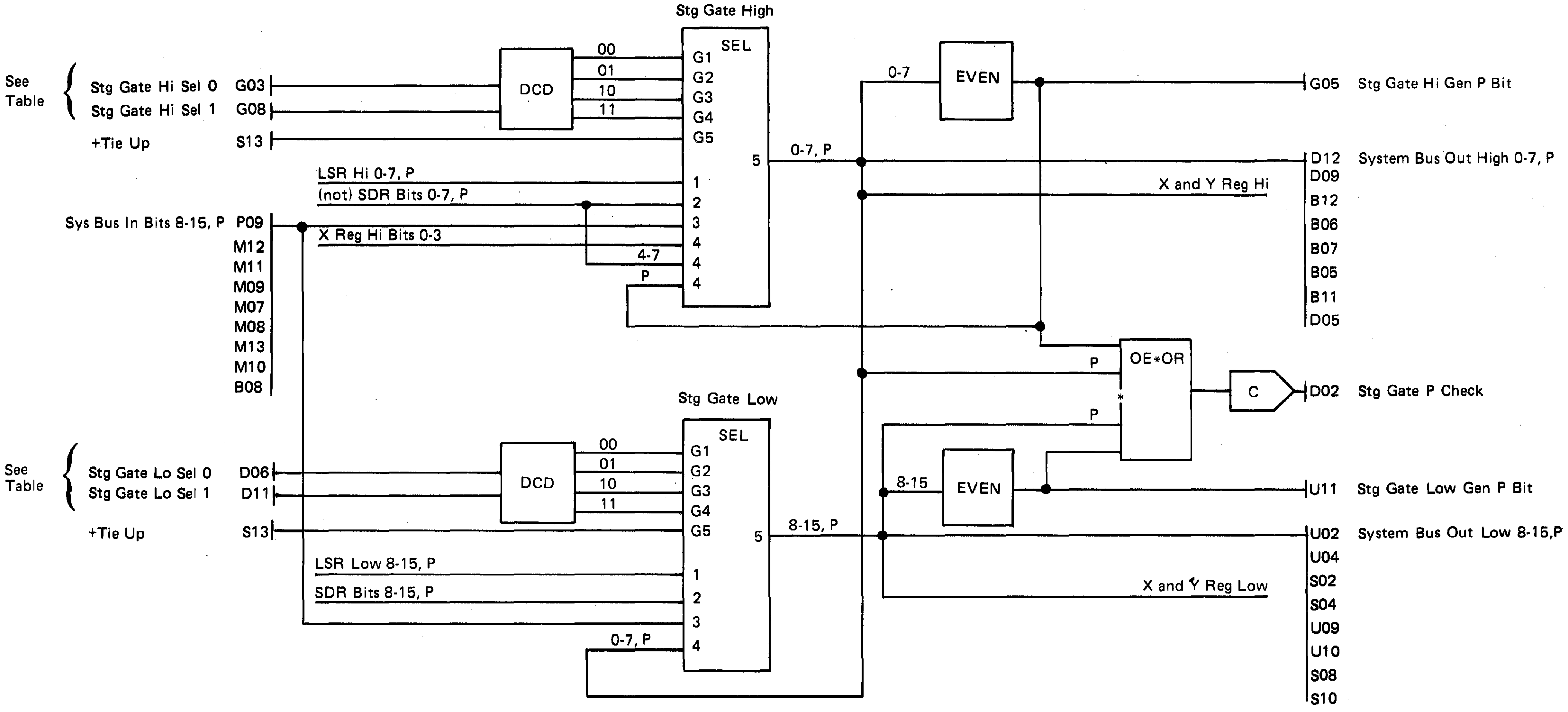
| PCR | | Flag Bit 0 | Positive Bit 1 | Negative Bit 2 | Zero Bit 3 | Carry Bit 4 | High Bit 5 | Low Bit 6 | Equal Bit 7 |
|---|---|---|---|---|---|---|---|---|---|
| L/A1 or L/A2 Logical | Set | | R1 or $\overline{R2}$ = all ones and result ≠ all zeros | Result ≠ all zeros and R1 or $\overline{R2}$ ≠ all ones | Results = all zeros | | | | |
| | Reset | | Result = all zeros or R1 or $\overline{R2}$ ≠ all ones | Result = all zeros or R1 or $\overline{R2}$ = all ones | Result ≠ all zeros | | | | |
| L/A1 or L/A2 Arithmetic | Set | | Result has a carry and result ≠ zero | Result has no carry and result ≠ zero | Result = zero | Result had a carry (add) ______<br>A borrow (sub) | Result has a carry and result ≠ zero | Result has no carry and result ≠ zero | |
| | Reset | | Result = no carry or result = zero | Result has a carry or result = zero | Result ≠ zero | No carry (add) result had a borrow (sub) | Result has no carry or result = zero | Result has carry or result = zero | Result ≠ zero |
| Test Mask | Set | | Tested bits = all ones | Tested bits ≠ all ones and tested bits ≠ all zeros | All tested bits = zero (or no bits tested) | | | | |
| | Reset | | Tested bits ≠ all ones or tested bits = all zeros | Tested bits = all ones or tested bits = all zeros | Tested bits ≠ zero or tested bits = all ones | | | | |
| Compare or Subtract Immediate | Set | | Register data is greater than immediate data | Register data is less than immediate data | Register data is equal to immediate data | | | | |
| | Reset | | Register data is not greater than immediate data | Register data is not less than immediate data | Register data is not equal to immediate data | | | | |
| I/O Immediate Reset Carry – Set Equal | Set | | | | | | | | Equal set on |
| | Reset | | | | | Carry set off | Decoded from carry and equal and set off | Decoded from carry and equal and set off | |
| I/O Immediate Load | Set | Loaded bit 0 is on | Loaded bit 1 is on | Loaded bit 2 is on | Loaded bit 3 is on | Loaded bit 4 is on | Loaded bit 4 is on and bit 7 off | Loaded bit 4 off and bit 7 off | Loaded bit 7 is on |
| | Reset | Loaded bit 0 is off | Loaded bit 1 is off | Loaded bit 2 is off | Loaded bit 3 is off | Loaded bit 4 is off | Loaded bit 4 off or loaded bit 7 on | Loaded bit 4 on or loaded bit 7 on | Loaded bit 7 is off |
| POR/Reset Key/Reset MCI | Set | | | | | | | | Equal set on |
| | Reset | Set off | Set off | Set off | Set off | Carry set off | Decoded from bits 4 and 7 and set off | Decoded from bits 4 and 7 and set off | |
| I/O Immediate Flag Latch | Set | Set on | | | | | | | |
| | Reset | Set off | | | | | | | |

## Storage Gate High/Low Ⓛ

The storage gates select data coming from the SDR, LSR, system bus in, and X-register available to system bus out and to the X- and Y-registers.

The selected bits are generated in the control processor control card by the MOR bits and T-times.

CP Data Flow Card A-A1H2



PC230

**Storage Gate High**

| Sel<br>0 1 | Lines Gated Through |
|---|---|
| 0 0 | LSR High |
| 0 1 | SDR Bits 0-7 |
| 1 0 | SBI Bits 8-15 |
| 1 1 | X-Reg High Bits 0-3 and SDR Bits 4-7 |

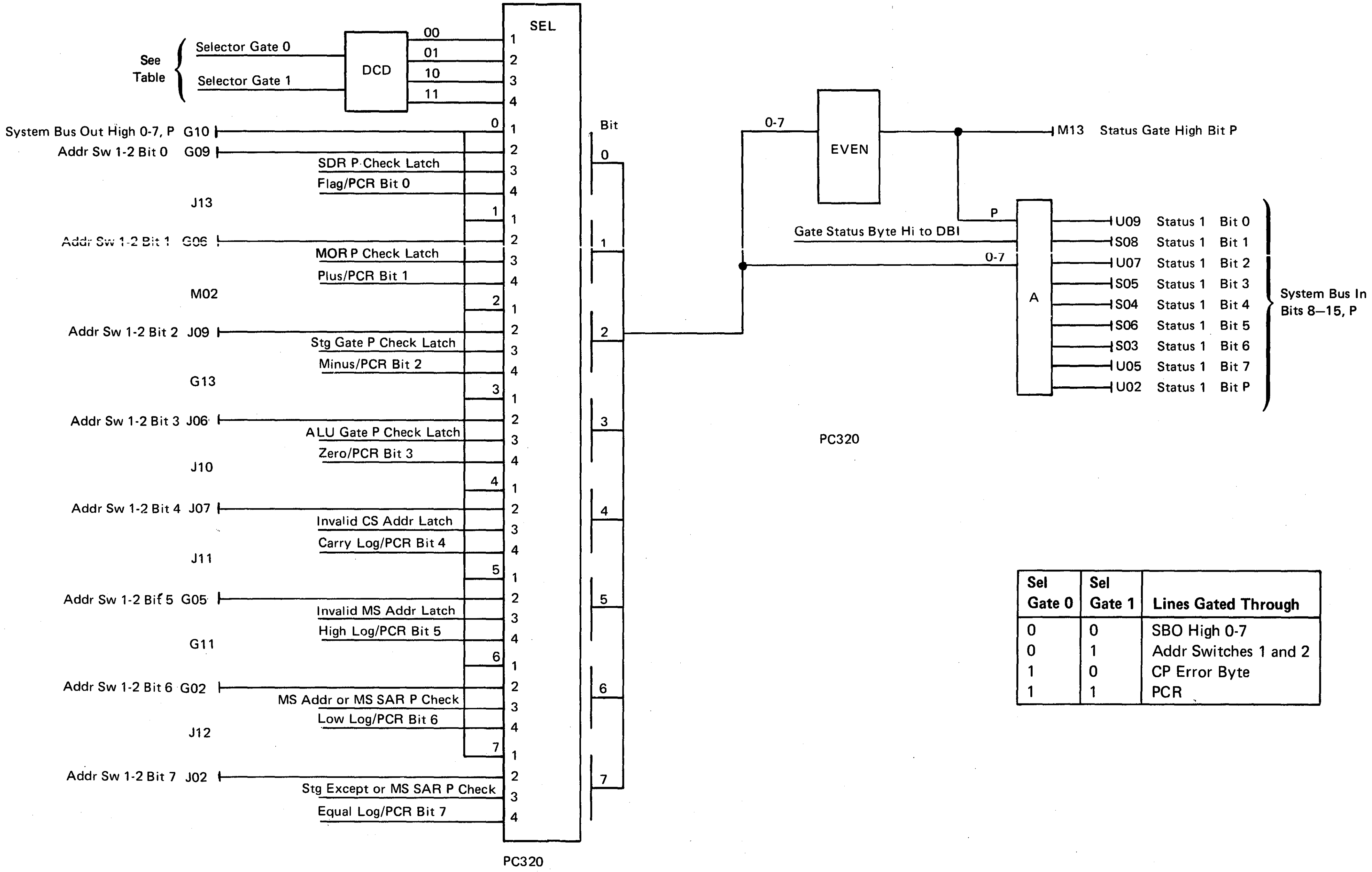
**Storage Gate Low**

| Sel<br>0 1 | Lines Gated Through |
|---|---|
| 0 0 | LSR Low |
| 0 1 | SDR Bits 8-15 |
| 1 0 | SBI Bits 8-15 |
| 1 1 | Stg Gate High Bits 0-7 |

## Status 1 Gate Ⓜ

The status 1 card gates the system bus out high 0-7 bits, address switches 1 and 2, CP checks error byte, and the processor condition register to the storage gates high/low. Also, the event indicators, display high byte bits 0-7 and P, branch on condition, and control storage address compare high logic are controlled by this card.

**Status 1 Gate Parity Generation**
**CP Status 1 Card A-A1J2**

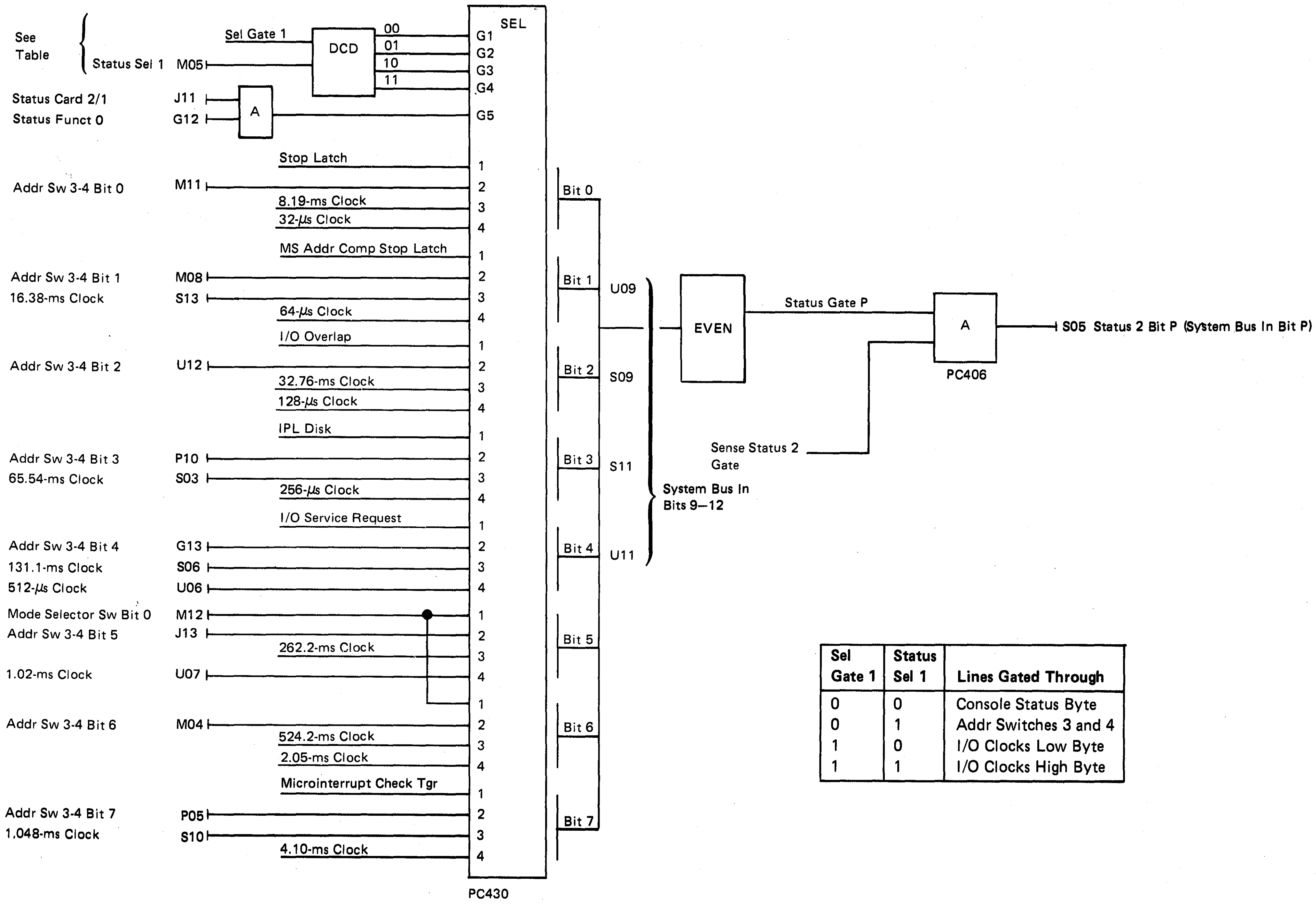


| Sel Gate 0 | Sel Gate 1 | Lines Gated Through |
|---|---|---|
| 0 | 0 | SBO High 0-7 |
| 0 | 1 | Addr Switches 1 and 2 |
| 1 | 0 | CP Error Byte |
| 1 | 1 | PCR |

## Status 2 Gate N

The status 2 card gates the address switches 3 and 4, console status byte, and I/O clocks, high/low byte. Also, the display low byte bits 12-15 and P, address compare low logic and sync, and start-stop-run logic are controlled by this card.

**Status 2 Gate Parity Generation**
**CP Status 2 Card A-A1K2**



| Sel Gate 1 | Status Sel 1 | Lines Gated Through |
|---|---|---|
| 0 | 0 | Console Status Byte |
| 0 | 1 | Addr Switches 3 and 4 |
| 1 | 0 | I/O Clocks Low Byte |
| 1 | 1 | I/O Clocks High Byte |

## ERROR CONDITIONS

The control processor program determines the cause of an I/O hardware error other than a control processor error. When an I/O error is found, the control processor attempts the operation again by executing the instruction, program, or task. However, some system errors stop the system. In some cases, a recovery is possible only by loading the system main storage programs again.

### Control Processor Checks

When a hardware error is found in the control processor (CP), a bit is set in the CP check register latches to indicate an error. This register can be sensed by an I/O immediate instruction (control processor sense–MPS). This instruction loads the contents of the CP check register into the specified LSR work register so the control processor check conditions can be sensed. These checks can also be displayed in the byte 0 lights on the CE panel by setting the Mode Selector switch to the Insn Step/Dply Chks position.

Any CP or port errors cause a CP machine check interrupt, processor check condition, and stops the MSP clocks.

MSP hardware checks cause an interrupt level 5 request to the CP and stop the MSP clock. Three MSP conditions cause the 'MSP hardware checks' line to become active. MSP status byte 2 must be sensed by a register control instruction (RMPR). Then, a CE panel display of the selected LSR work register can determine which of the three conditions caused the error. The conditions are as follows:

1. Control gate check (status byte 2 bit 1)
2. LSR gate check (status byte 2 bit 2)
3. Main storage gate check (status byte 2 bit 3)

**Processor Error Byte (Display Byte 0)**

| Bit | Error | Cause |
|---|---|---|
| 0 | Storage data register parity check | Parity in the storage data register is not correct. |
| 1 | Micro-operation register parity check | Parity in the micro-operation register is not correct. |
| 2 | Storage gate parity check | Parity at the output of the storage gate is not correct. |
| 3 | ALU gate parity check | The parity expected does not match the parity generated at the ALU gate. |
| 4 | Illegal control storage address/ storage address register | Control storage was addressed outside its limits. Bits 4 and 5 both on indicates that parity in the storage address register is not correct. |
| 5 | Control storage program check/storage address register | The control storage program remained in a loop for more than 7 seconds. Bits 4 and 5 both on indicates that parity in the storage address register is not correct. |
| 6 | Illegal main storage address/main storage address register | The real or translated main storage address used by the control storage program is greater than the main storage size of the system. Bits 6 and 7 both on indicates that parity in the main storage address register is not correct. |
| 7 | Storage exception/ main storage address register | The control storage program addressed a not valid address translation register; that is, an address translation register containing hexadecimal FF. Bits 6 and 7 both on indicates that parity in the main storage address register is not correct. |

**Decode of Bits 6 and 7**

| Bits 6 7 | CMR Bit 7 | PMR Bit 7 | Cause |
|---|---|---|---|
| 1 0 | 0 | * | Invalid main storage address (real) |
| 1 0 | 1 | * | Invalid main storage address (translate) |
| 0 1 | 1 | * | Storage protect |
| 0 1 | * | 1 | MSP tried to alter PMR while PMR bit 7 = 1 |
| 1 1 | * | * | MSAR parity check |
| 1 1 | 1 | * | ATR parity check |

Legend: * = don't care

**Processor Errors**

As a result of a control processor hardware error, the system programs must be loaded again. When the Load switch is pressed, special initial program load routines determine if the processor was in a processor check halt state before the Load switch was pressed. A routine then records the error information in the control processor error recording area and on the disk.

For each error, the following data is recorded:

- The processing level on which the error occurred
- The contents of the control processor microaddress register of the level on which the error occurred
- The contents of the microaddress backup register of the level on which the error occurred
- The contents of the work registers of the level on which the error occurred
- The contents of the processor condition register
- The processing unit checks byte
- The port checks byte
- The time and date of the logout

The recorded data does not change as a result of pressing the Load switch to load and run these special diagnostic routines after an error. Therefore, the recorded information indicates the state of the control processor when the error occurred, except for time and date.

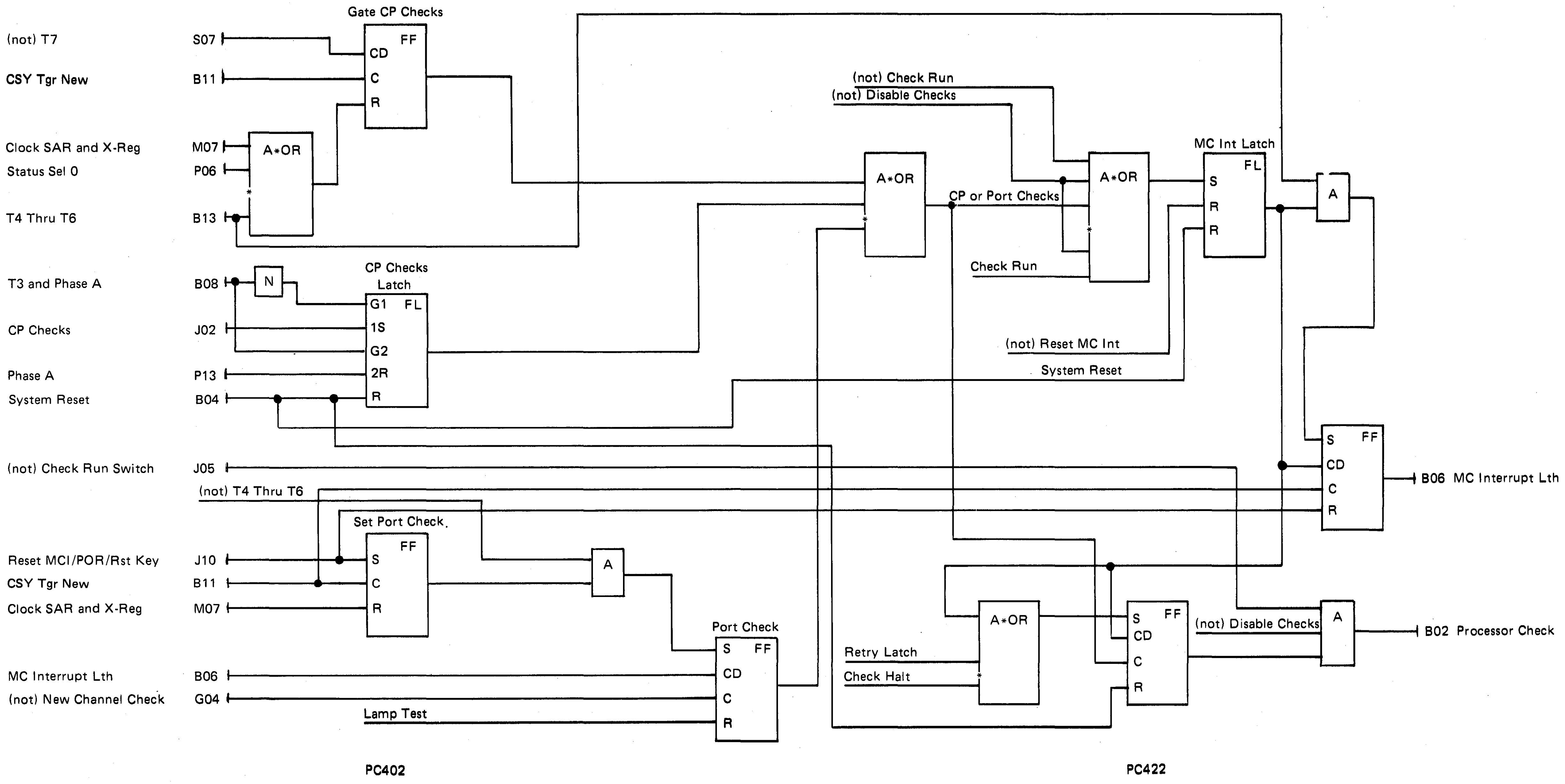
Examples of the error history tables for the control processor and the main storage processor can be found under *Error Indications* earlier in this section.

### Error Conditions (Second Level)

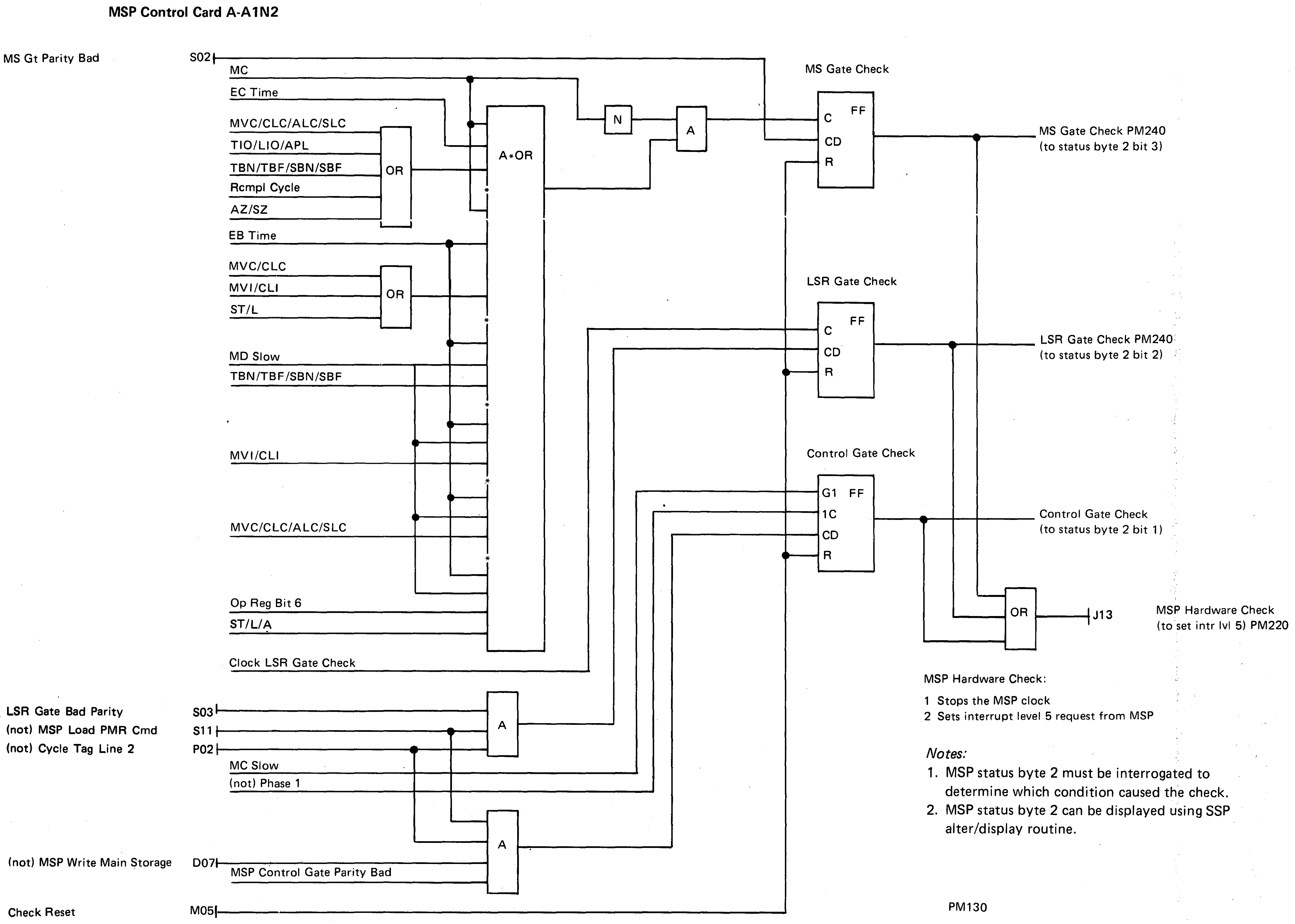
Errors associated with the main storage processor and the control processor are shown on the following pages. The control processor checks (second level) are also shown individually and are key-coded and referenced to the second-level diagram.

### Machine Check Interrupt and Processor Check Generation

CP Status 2 Card A-A1K2



**MSP Hardware Checks**



MSP Hardware Check:

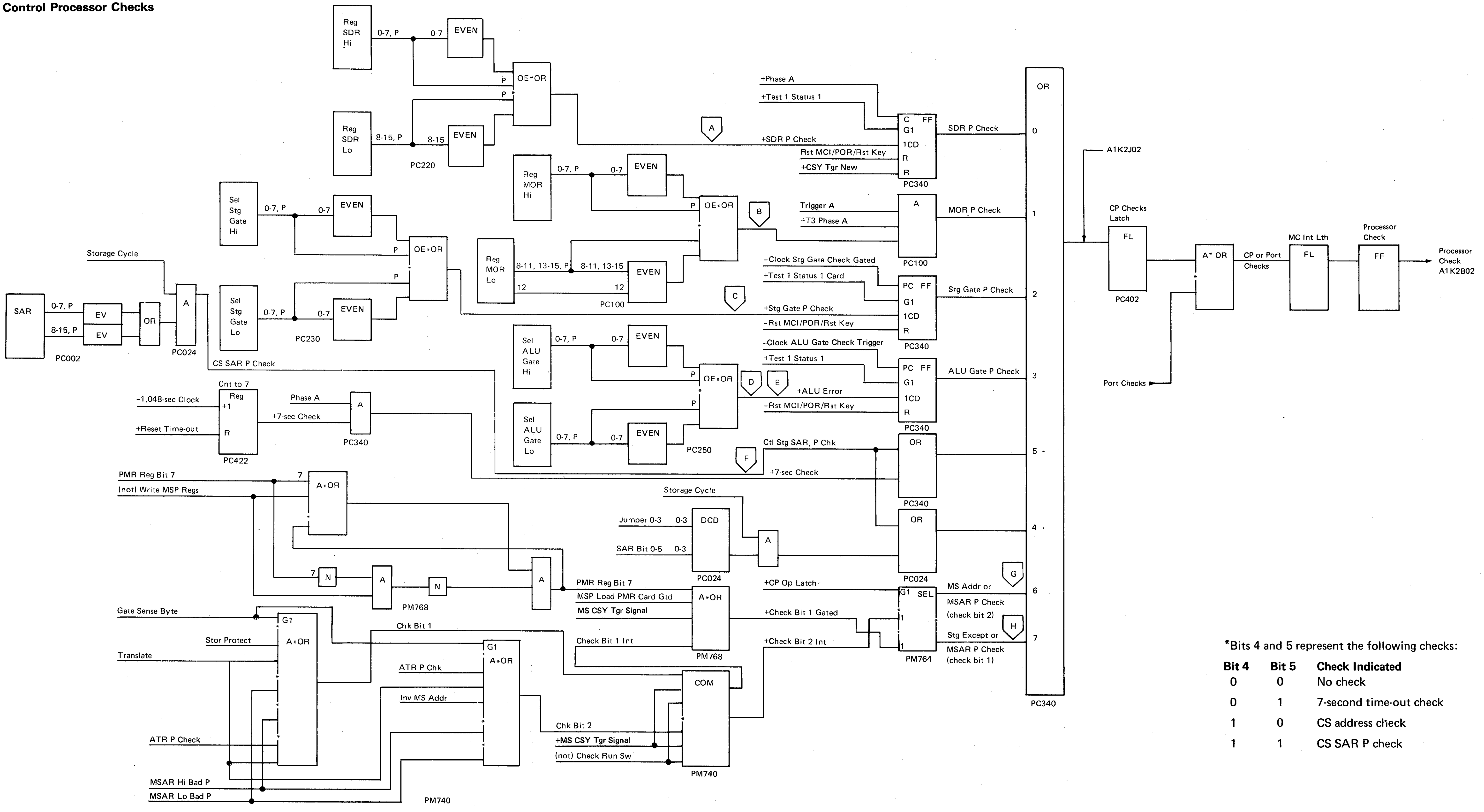
1 Stops the MSP clock

2 Sets interrupt level 5 request from MSP

*Notes:*

1. MSP status byte 2 must be interrogated to determine which condition caused the check.
2. MSP status byte 2 can be displayed using SSP alter/display routine.

## Control Processor Checks

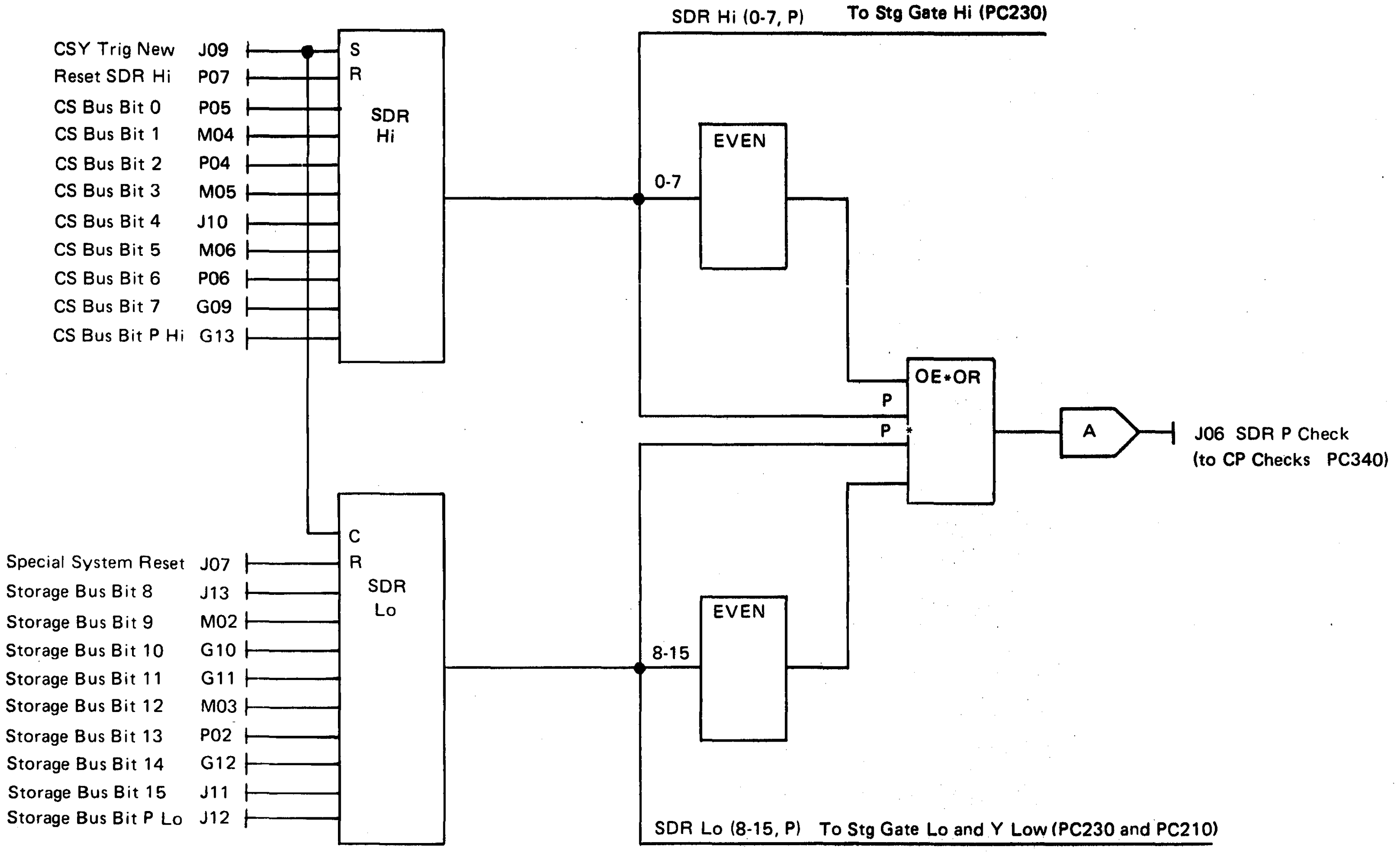


*Bits 4 and 5 represent the following checks:

| Bit 4 | Bit 5 | Check Indicated |
|---|---|---|
| 0 | 0 | No check |
| 0 | 1 | 7-second time-out check |
| 1 | 0 | CS address check |
| 1 | 1 | CS SAR P check |

## SDR Parity Check Generation

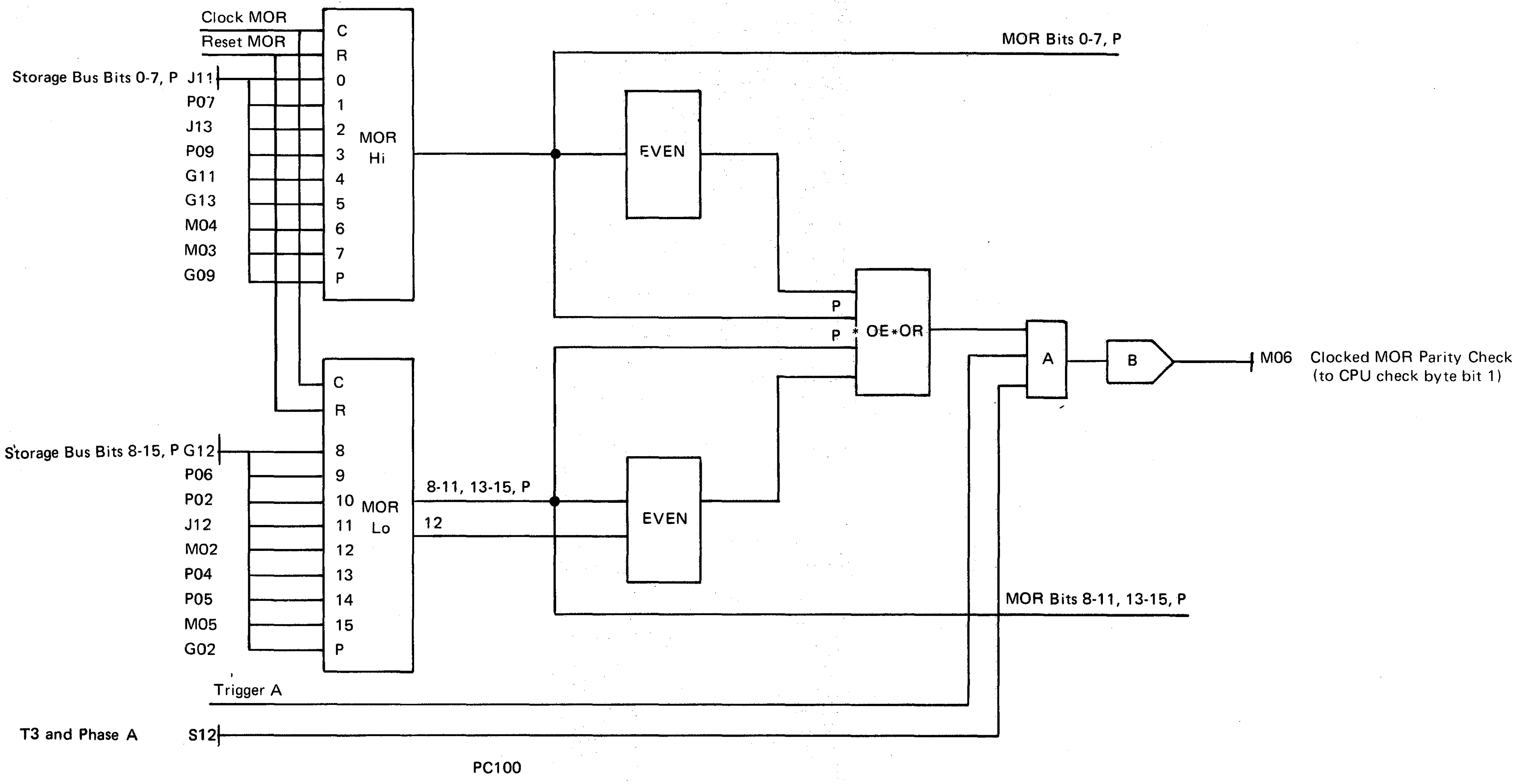
CP Data Flow Card A-A1H2



PC220

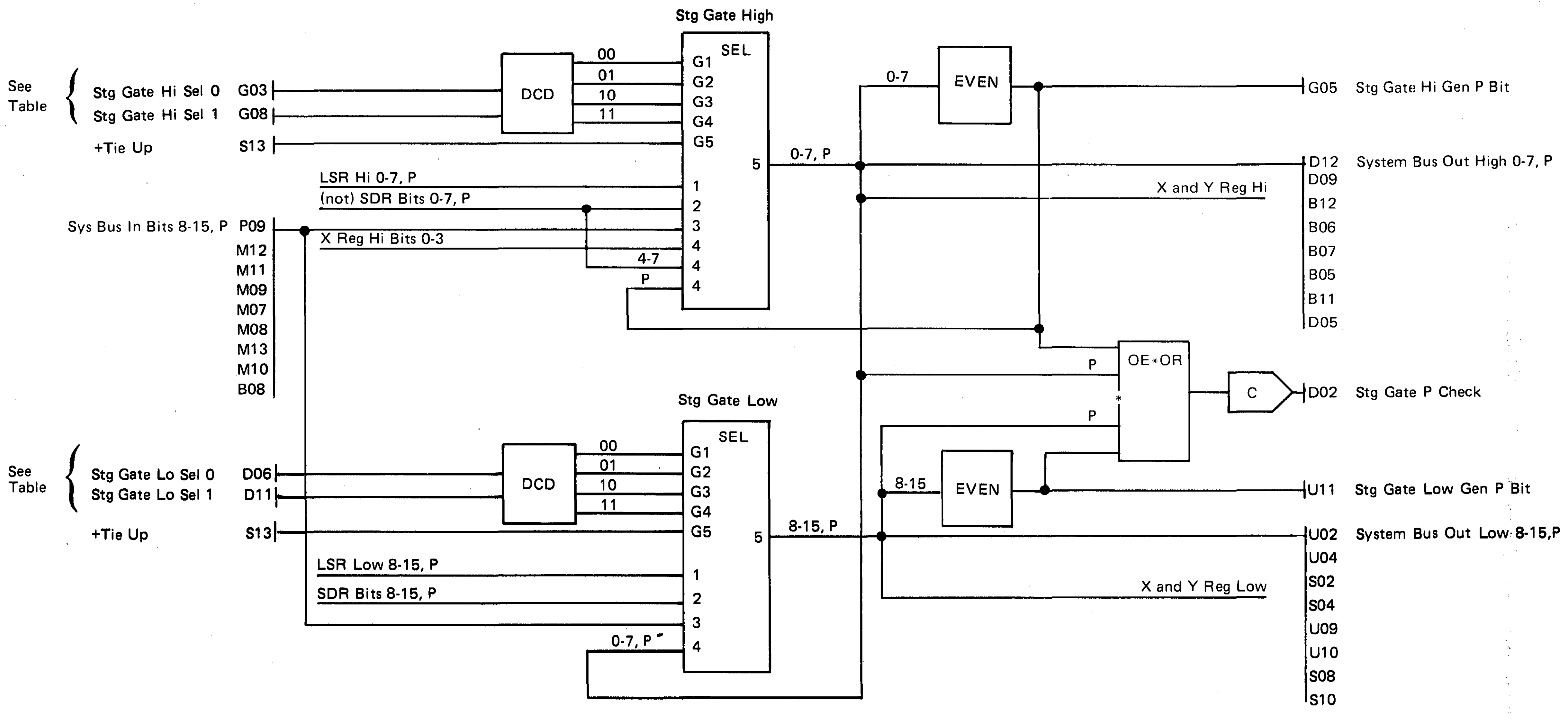
## MOR Parity Check

### CP Control Card A-A1G2



## Storage Gate High/Low Parity Check and Generation

CP Data Flow Card A-A1H2



PC230

**Storage Gate High**

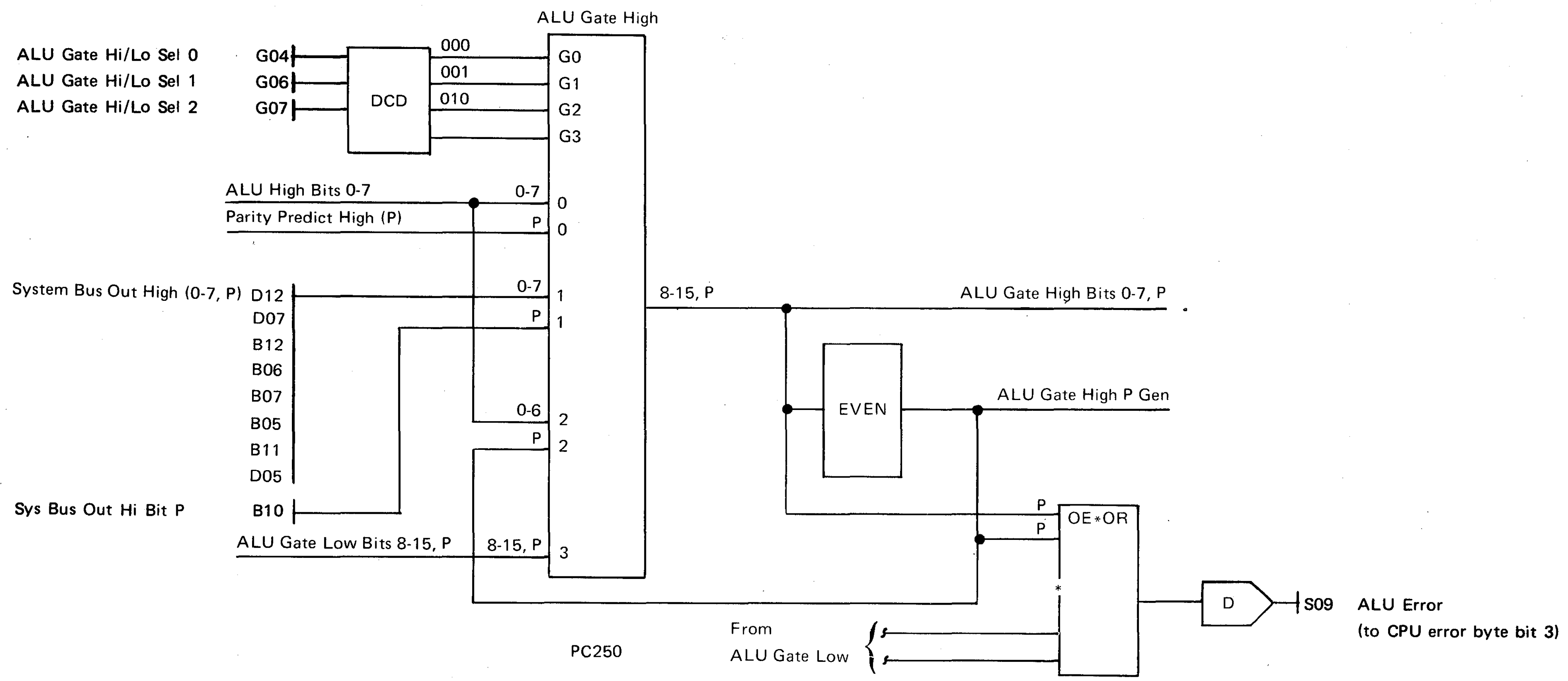
| Sel<br>0 1 | Lines Gated Through |
|---|---|
| 0 0 | LSR High |
| 0 1 | SDR Bits 0-7 |
| 1 0 | SBI Bits 8-15 |
| 1 1 | X-Reg High Bits 0-3 and SDR Bits 4-7 |

**Storage Gate Low**

| Sel<br>0 1 | Lines Gated Through |
|---|---|
| 0 0 | LSR Low |
| 0 1 | SDR Bits 8-15 |
| 1 0 | SBI Bits 8-15 |
| 1 1 | Stg Gate High Bits 0-7 |

## ALU Gate High Parity Check and Generation

### CP Data Flow Card A-A1H2

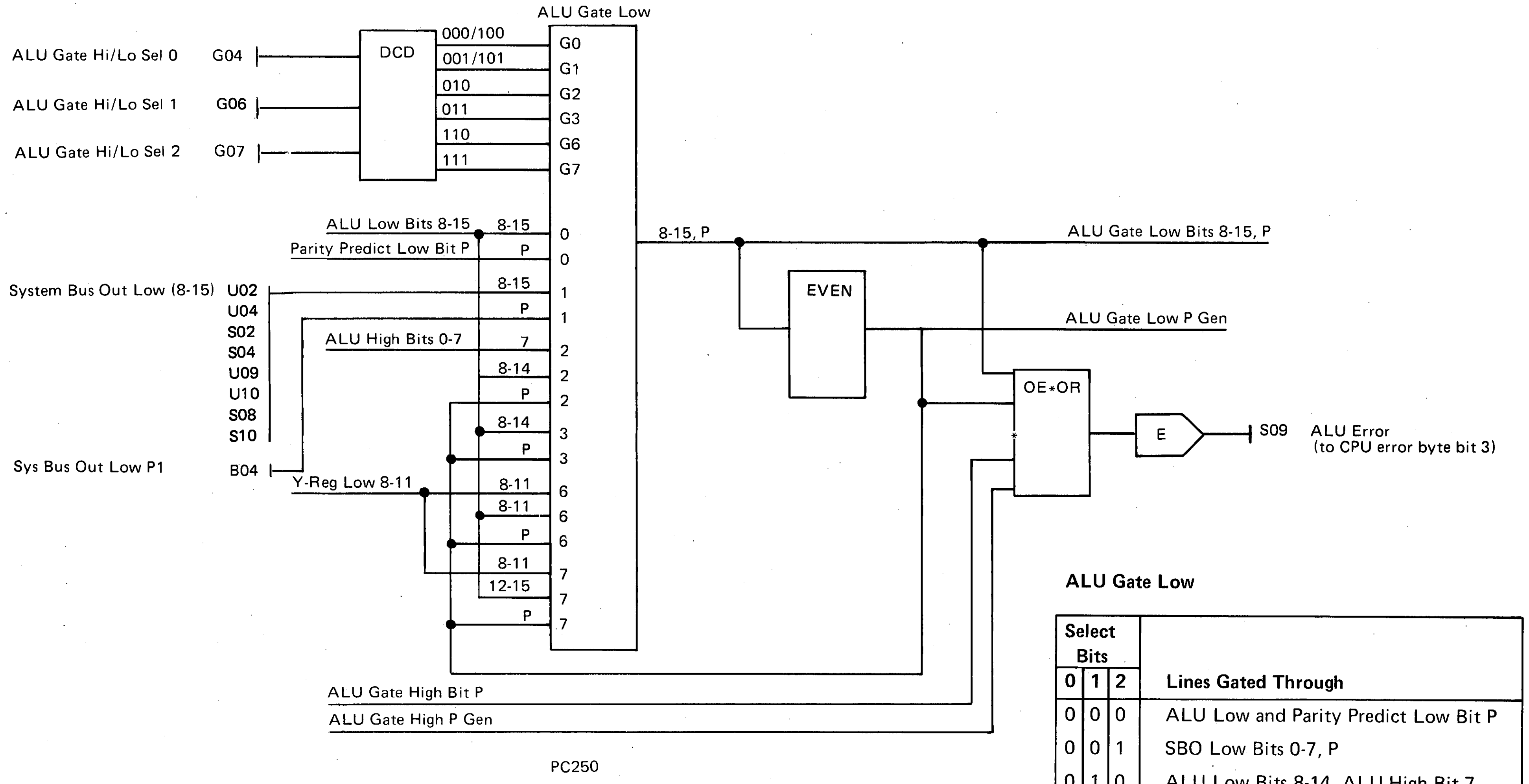


**ALU Gate High**

| Select Bits | | | Lines Gated Through |
|---|---|---|---|
| **0** | **1** | **2** | |
| 0 | 0 | 0 | ALU High Bits 0-7, Parity Predict |
| 0 | 0 | 1 | SBO 0-7, P |
| 0 | 1 | 0 | ALU High Bits 0-6, ALU Hi P Gen |
| 0 | 1 | 1 | ALU Gate Low Bits 0-7, P |
| 1 | 0 | 0 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

## ALU Gate Low Parity Check and Generation

### CP Data Flow Card A-A1H2

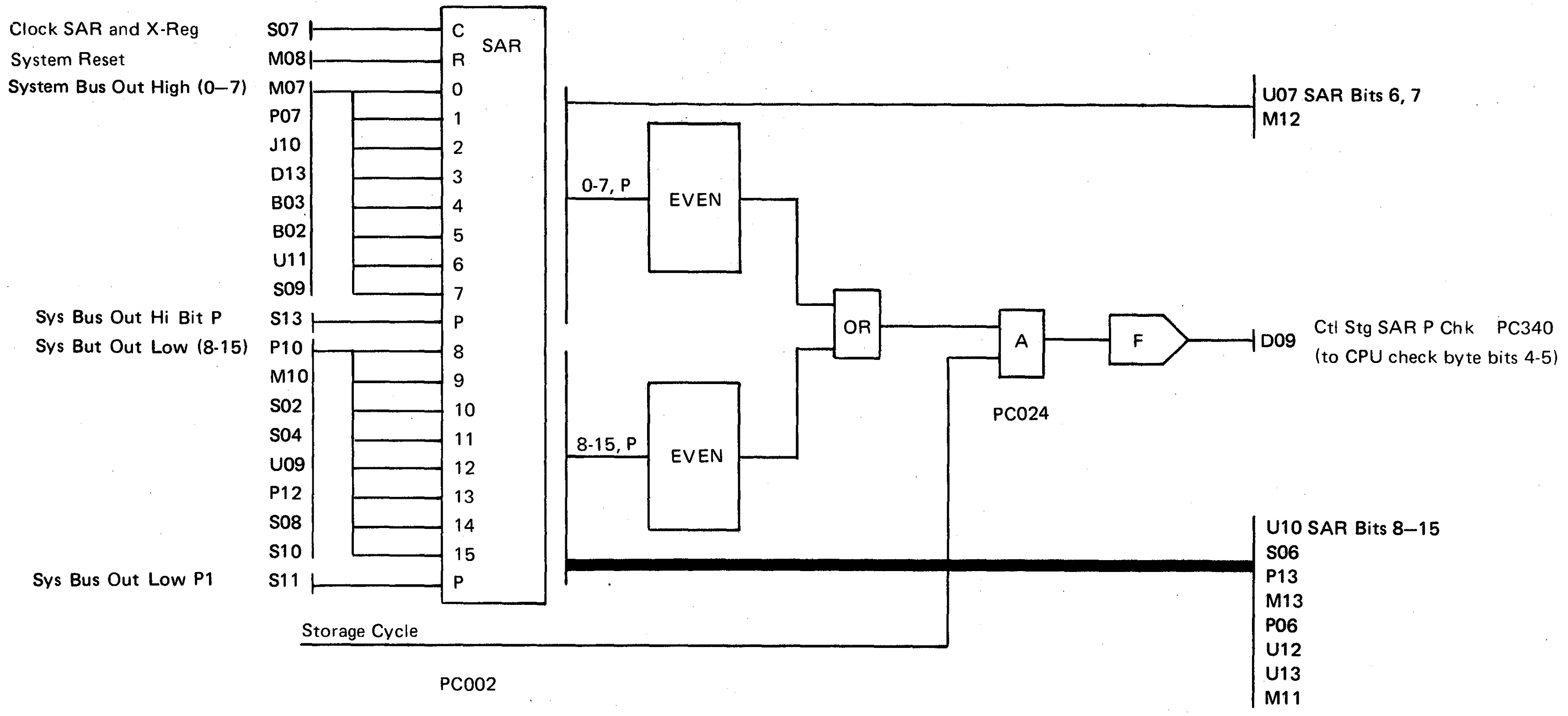


### ALU Gate Low

| Select Bits | | | Lines Gated Through |
|---|---|---|---|
| 0 | 1 | 2 | |
| 0 | 0 | 0 | ALU Low and Parity Predict Low Bit P |
| 0 | 0 | 1 | SBO Low Bits 0-7, P |
| 0 | 1 | 0 | ALU Low Bits 8-14, ALU High Bit 7, ALU P Gen. |
| 0 | 1 | 1 | ALU Low Bits 8-14, ALU Low P Gen |
| 1 | 1 | 0 | ALU Low 12-15, X-Reg |
| 1 | 1 | 1 | Y-Reg Low 8-11, ALU Low 12-15, ALU P Gen |

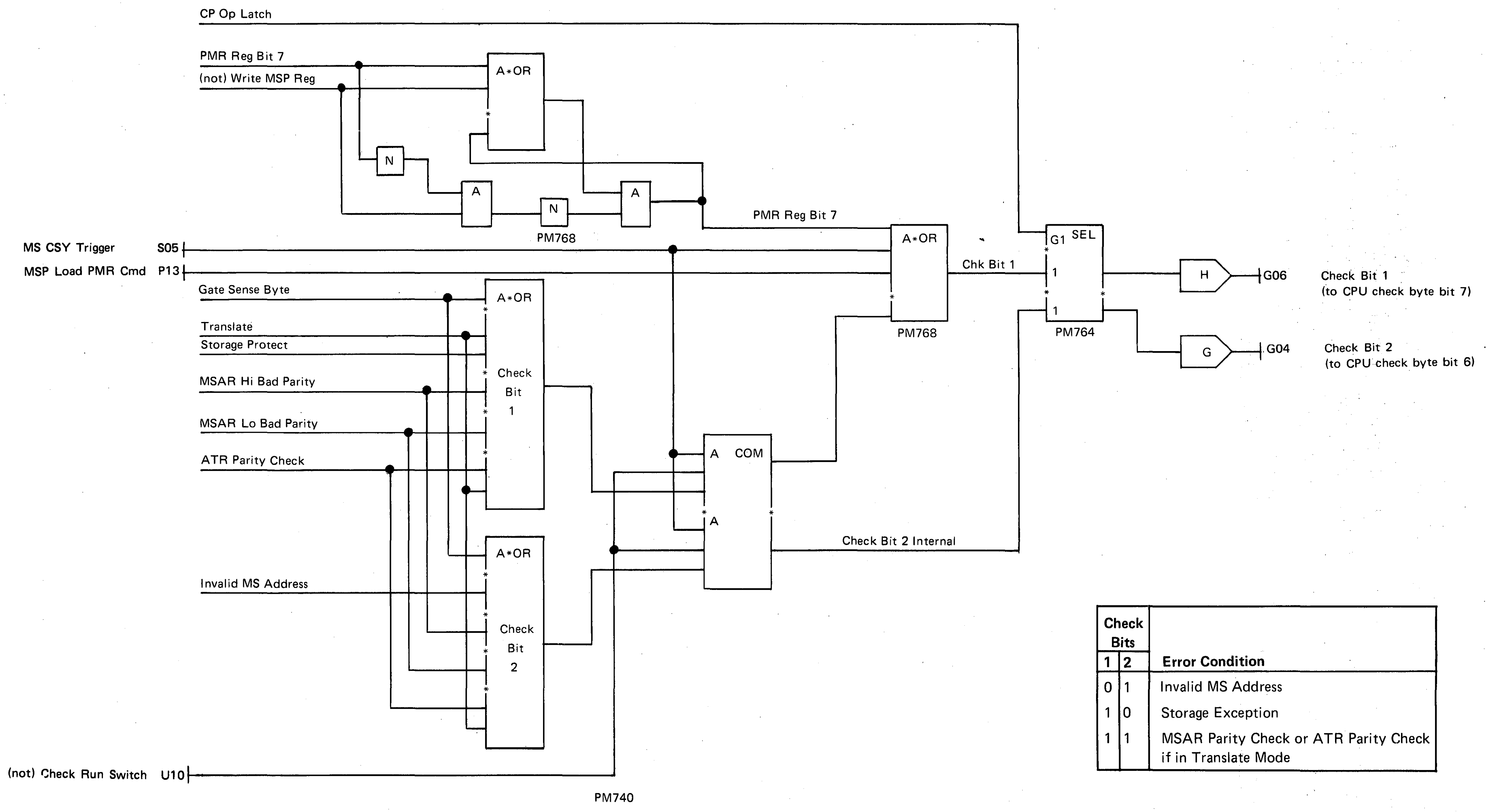
## Control Storage SAR Parity Check

### CP Storage Control Card A-A1F2



## MSP Check Bits 1 and 2

### MS Control Card A-A1Q2



| Check Bits | | |
|---|---|---|
| 1 | 2 | Error Condition |
| 0 | 1 | Invalid MS Address |
| 1 | 0 | Storage Exception |
| 1 | 1 | MSAR Parity Check or ATR Parity Check if in Translate Mode |

This page intentionally left blank.