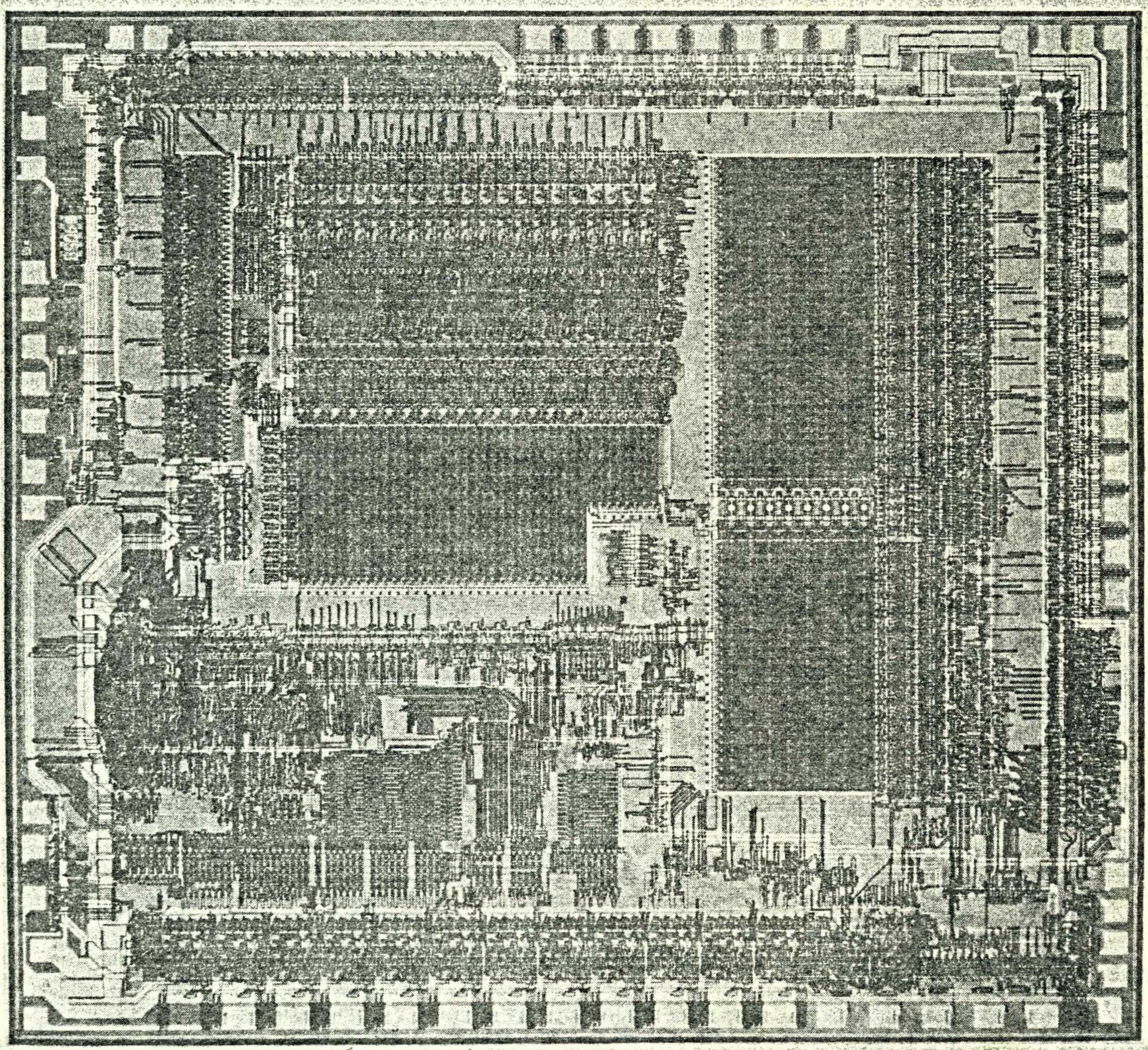
Honeywell

# LSI-6 User's Manual



# LSI-6 USER'S MANUAL

This document and the information contained herein are confidential to and the property of Honeywell Information Systems Inc. and are made available only to Honeywell employees for the sole purpose of maintaining Honeywell's products. This document, any copy thereof and the information contained herein shall be maintained in strictest confidence; shall not be copied in whole or in part except as authorized by the employee's manager; and shall not be disclosed or distributed (a) to persons who are not Honeywell employees, or (b) to Honeywell employees for whom such information is not necessary in connection with their assigned responsibilities. Upon request, or when the employee in possession of this document no longer has need for the document for the authorized Honeywell purpose, this document and any copies thereof shall be returned to the employee's manager. There shall be no exceptions to the terms and conditions set forth herein except as authorized in writing by the responsible Honeywell Vice President.

DOCUMENT NUMBER
71017559

File No.: 3S>3

February 1982

Honeywell

# CONTENTS

Page

SECTION 1 INTRODUCTION

SECTION 2 INTERFACE AND SIGNAL DESCRIPTIONS

SECTION 3 HARDWARE DESCRIPTION

# CONTENTS

# CONTENTS

Page

# ILLUSTRATIONS

Figure Page

# TABLES

# Section 1
# INTRODUCTION

The LSI-6 microprocessor is an NMOS, 16-bit chip capable of arithmetic, logic, and control operations, driven by a 48-bit external control mechanism Read Only Storage (ROS). The LSI-6 is a sophisticated processor whose design permits the execution of the Level 6 instruction repertoire (full software compatibility with Model 6/43), and allows for new architectural identities (emulation of foreign instruction sets). The LSI-6 is designed to directly control I/O and memory operations for ease in integrated system designs. The unique features available in the LSI-6 design that allow greater control and integration are:

A 48-bit external ROS that provides true horizontal microprogramming allowing up to 12 simultaneous micro-operations per 48-bit word

Eight external hardware interrupts that generate vectors to ROS microprogram routines (six of which can be user specified)

Five external software interrupts that are handled by firmware control (two can be specified by user)

Ten external monitor bits that are sensed and controlled by sophisticated test branch and major branch operations within the firmware.

## 1.1 SYSTEM FEATURES

- System Processor Bus

  20 address/data lines
  4 control lines

- External ROS

  Includes 1.5K words for Model 6/43 functionality (excludes I/O and control panel operation)

  Allows user 2.5K words for system integration

- Software Interrupt Capability

  Real-time clock
  Three levels of I/O interrupt
  Power failure interrupt

- Hardware Interrupts

  Nonexistent resource trap

  Memory access violation Memory Management Unit (MMU) (internal hardware interrupt)

  Uncorrectable memory error

  Memory refresh request

  Five levels of data transfer requests

- Monitor Inputs

  External events may be individually monitored and tested under firmware control

- Options

  Memory management with Model 6/43 functional MMU hardware resident on chip

  Level 6 Long Address Form (LAF) or Short Address Form (SAF) (20 bit/16 bit) address

  Level 6 CIP address syllable resolution

  Two additional user-definable options

## 1.2 HARDWARE FEATURES

- 16- and 20-Bit Arithmetic Logic Unit (ALU) With A- and B-Port Inputs

  A-port sourced by a 32-location register file with:

  16 data registers by 16 bits wide
  16 address registers by 20 bits wide

  B-port sourced by two hardware registers with:

  Q-register (7, 8, or 16 bit--right justified)
  G-register (20 bits)

- Full 16- or 32-Bit Shifting Capability

  16-bit register file
  16-bit Q-register
  32-bit register file, Q-register

- Program Counter With a 20-bit Increment/Decrement Feature

- Byte Swap Mechanism With Byte Manipulation

- Hardware Mask Generation

- Constant Generation to Internal Bus With Two Independent Sources to the Bus

- Level 6 7-Bit Indicator Register

- Mode Registers Using an 8 by 8 Register File

- MMU Hardware (Model 6/43 Functionality)

  31 by 32 register file (segment descriptors)
  12-bit relocation adder
  9-bit block size comparator
  Three 2-bit ring comparators
  Four memory violation flip-flops

- Next ROS Address Generation

  12-bit ROS address register
  Major branch matrixes
  Test branch matrixes
  4 by 12 push/pop stack
  Hardware interrupt priority network (Prinet)
  Software interrupt priority network (Prinet)

- 16-Bit Functional Register

  Software instruction register

- Various Storage Elements

  Monitor inputs<br>
  Software/hardware interrupt lines

## 1.3 FIRMWARE FEATURES

- Horizontal Microprogramming With:

  48-bit firmware word allowing up to 12 simultaneous micro-commands and internal/external simultaneous control operations

  4K word addressability for ROS generated internally; delivered to external hardware

- Subroutine Capability

  Push/pop micro-ops together with 4 by 12 stack allows up to four levels of subroutine

- Five ROS Address Generation Methods

  Jump (to absolute address)

  Test branch with:

  64 test conditions
  Two-way branch

  Major Branch with:

  15 test condition sets
  16-way branch

  Increment

  Return (Pop)

- Vectored Hardware Interrupt

## 1.4 PHYSICAL CHARACTERISTICS

68 pin--leadless package

## 1.5 ELECTRICAL CHARACTERISTICS

- Power

  VCC = +5V ± 5%
  VSS = 0V
  VBB = -5V ± 10%

- Temperature Range

  -10° C to +85° C

- Clock (two phase)

  Cycle Time: DC to 250 nanoseconds (refer to Appendix B for timing specifications).

- All Input/Output Signals are TTL Compatible

# Section 2
# INTERFACE AND SIGNAL DESCRIPTIONS

This section defines the LSI-6 interface and provides a description of the interface signals. The interface signals are divided into four groups according to pin assignment and phase relationship of the clock.

## 2.1 LSI-6 CLOCK

The LSI-6 clock is the primary element of the interface and it is composed of Phase A and Phase B as shown in Figure 2-1.

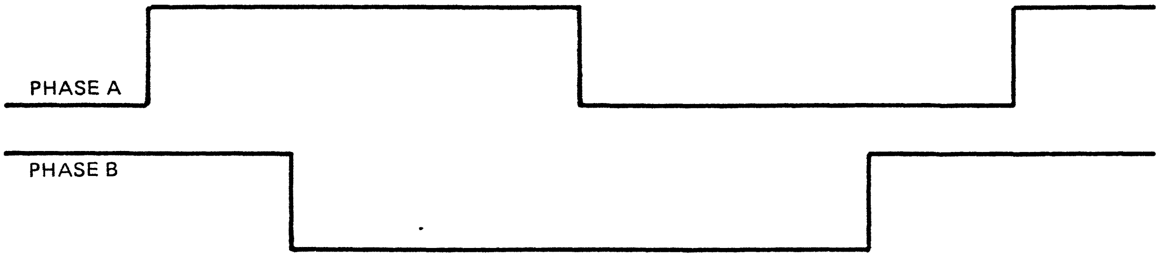


Figure 2-1. LSI-6 Clock Phase Relationship

The phase relationship between Phase A and Phase B determines the functions of the interface lines because there are 129 signals (excluding power, ground, and clock timing) shared among 57 input/output pins.

Phase A signal is used by the LSI-6 and the system to reverse the direction of the I/O bus drivers. When Phase A is true, the 48 ROS data lines and the five option lines are inputs to the LSI-6. When Phase A is false, all other shared signals are either inputs or outputs to the LSI-6.

Phase B signal is used to latch the signals that were gated with Phase A. When Phase B is going false, the ROS data and options are latched internal to the LSI-6. When Phase B is going true, all other shared signals are latched internal to the LSI-6.. For detailed timing, refer to Appendix B.

## 2.2 SIGNAL GROUPS

There are four signal groups defined by the LSI-6 interface. Each group, except group 1, consists of signals that have common pin assignments and clock phase relationship as shown in Figures 2-2 and 2-3.

Group 1 consists of nine interface lines that have unshared signals. They are: three voltage, two ground, and four timing signals.

Group 2 consists of 23 interface lines which control 69 signals, five of which are unused. During Phase A true, the 23 lines represent 21 ROS input data bits and two input option bits. During Phase A false, the 23 lines are bidirectional representing inputs of 16 data bus bits, 3 control signals, and 4 unused signals; or outputs of 20 address/data bus bits and 3 control signals.

Group 3 consists of 19 interface lines which control 38 input signals. During Phase A false, the 19 lines represent 4 software interrupt signals, 8 hardware interrupt signals, and 7 monitor bits. During Phase A true, the 19 lines represent 14 ROS input data bits, 3 input option bits, and 2 unused signals.

Group 4 consists of 13 interface lines which control 26 signals. During Phase A false, the 13 lines represent 12 ROS output address bits and an error signal (MEMKIL). During Phase A true, the 13 lines represent 13 ROS input data bits.

### 2.2.1 Group 1 Signals

Group 1 (unshared pins) consists of nine unshared interface lines. Table 2-1 lists the interface lines and the associated signal name. Table 2-2 provides a definition of group 1 signals.

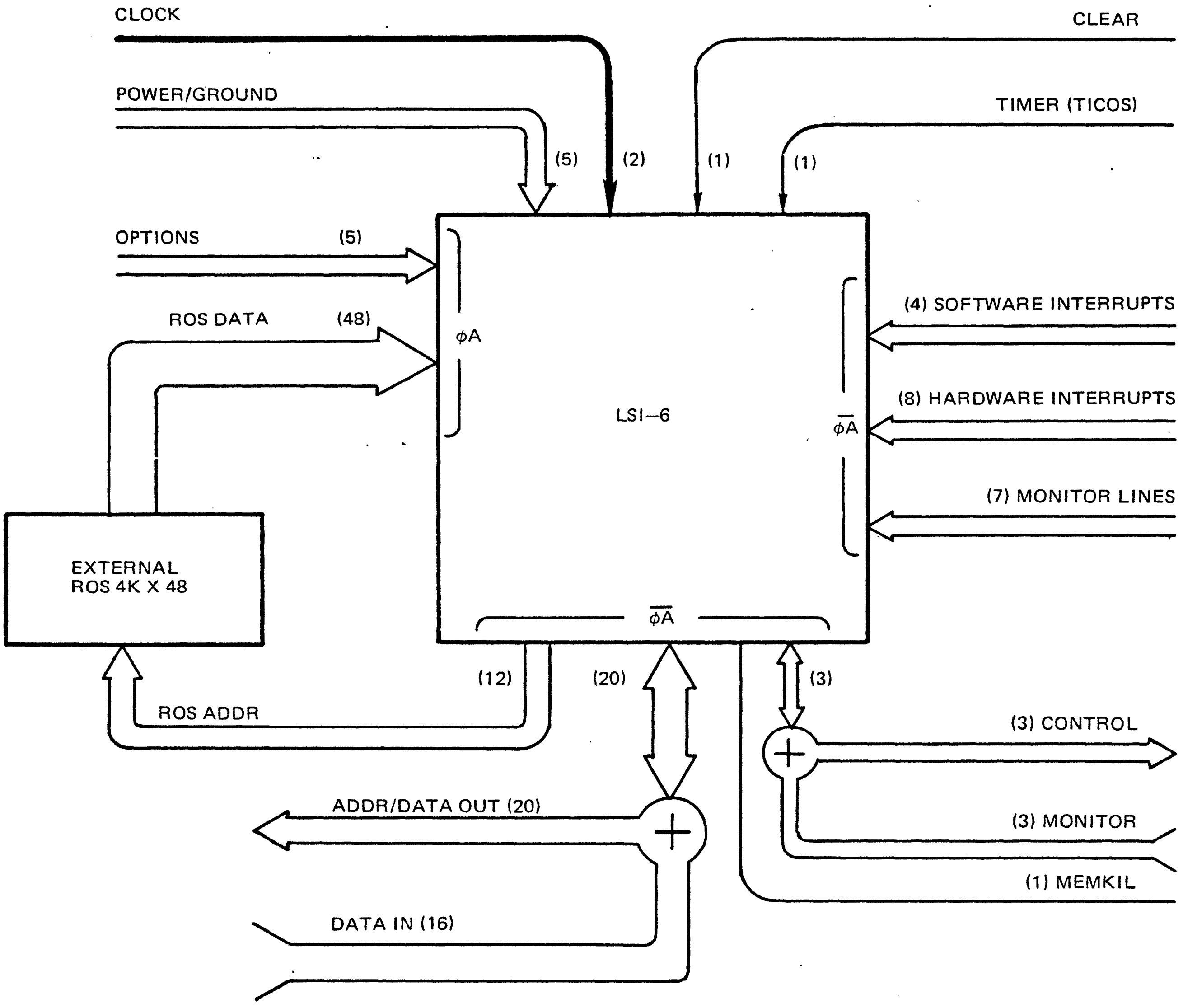


Figure 2-2. Interface Signals and Phase A Polarity

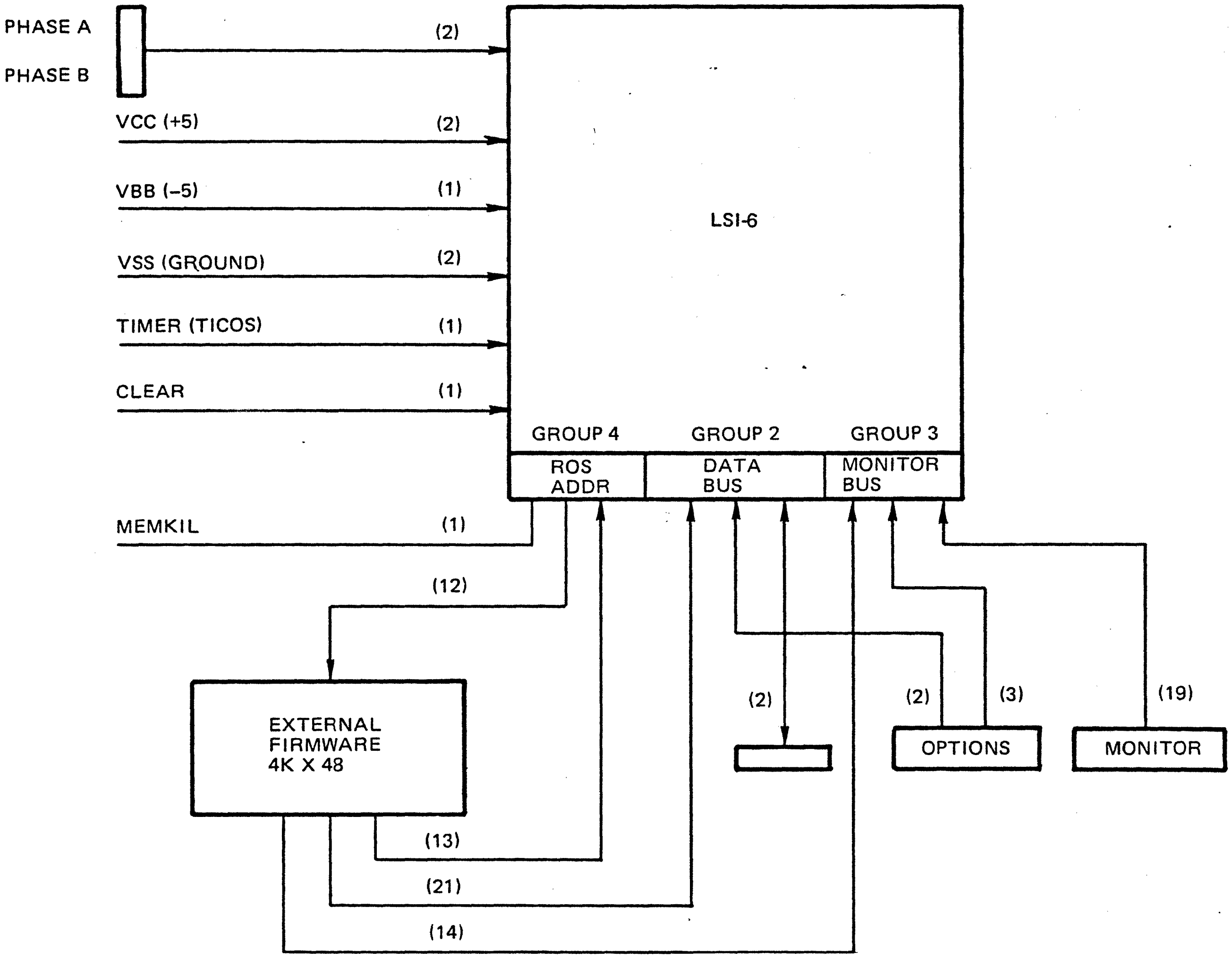


Figure 2-3. Basic LSI-6 With Interface Buffer

Table 2-1. Unshared Pins Interface/ Function Names

| Interface Name | Function (Signal Name) |
|---|---|
| VOLTA1 | VCC (+5V) |
| VOLTA2 | VCC (+5V) |
| VOLTA3 | VBB (-5V) |
| GRND01 | VSS (GND) |
| GRND02 | VSS (GND) |
| T1CK0S+0A | T1COS (Timer) |
| 0A | Clock Phase A |
| 0B | Clock Phase B |
| CLEARX-0A | Clear |

Table 2-2. Unshared Pins Signal Definitions

| Signal Name | Description |
|---|---|
| VCC (+5V) | +5 Volt Power Supply |
| VBB (-5V) | -5 Volt Power Supply |
| VSS | Ground |
| T1COS (Timer) | T1COS is a square wave input. The leading and trailing edges are detected on the chip and cause a software interrupt. It is during this interrupt that firmware updates the watchdog timer, the real-time clock, and the maintenance panel (assuming all are present). (This is a 60 hertz square wave--8.33 milliseconds between leading and trailing edges.) |
| CLOCK (Phase A)<br>(Phase B) | Phase A and Phase B is the two-phase clocking system for sequencing the processor chip. Voltage amplitudes are non-TTL levels. |
| CLEAR | CLEAR is a signal active low for a minimum of 10 microseconds. It forces the ROS address lines to location 0 and resets the following flip-flops: watchdog timer, real-time clock, MMU flip-flops 1, 2, 3, and 4. It also places 001 in all four stack locations. |

### 2.2.2 Group 2 Signals

Group 2 (processor bus) consists of 23 shared interface lines. Table 2-3 lists the interface lines and the associated signals. Table 2-4 provides a definition of group 2 signals.

Table 2-3. Processor Bus Interface/Function Names

| Interface Name | Function (Signal Name) | | |
|---|---|---|---|
| | $\overline{\phi}$A Input | $\overline{\phi}$A Output | ØA Input |
| DABS0D+18 | Unused | ADDR BUS 0D | ROS DATA 18 |
| DABS0C+19 | Unused | ADDR BUS 0C | ROS DATA 19 |
| DABS0B+20 | Unused | ADDR BUS 0B | ROS DATA 20 |
| DABS0A+21 | Unused | ADDR BUS 0A | ROS DATA 21 |
| DABS00+23 | DATA BUS 00 | ADDR/DATA BUS 00 | ROS DATA 23 |
| DABS00+24 | DATA BUS 01 | ADDR/DATA BUS 01 | ROS DATA 24 |
| DABS02+24 | DATA BUS 02 | ADDR/DATA BUS 02 | ROS DATA 25 |
| DABS03+35 | DATA BUS 03 | ADDR/DATA BUS 03 | ROS DATA 35 |
| DABS04+36 | DATA BUS 04 | ADDR/DATA BUS 04 | ROS DATA 36 |
| DABS05+37 | DATA BUS 05 | ADDR/DATA BUS 05 | ROS DATA 37 |
| DABS06+38 | DATA BUS 06 | ADDR/DATA BUS 06 | ROS DATA 38 |
| DABS07+39 | DATA BUS 07 | ADDR/DATA BUS 07 | ROS DATA 39 |
| DABS08+40 | DATA BUS 08 | ADDR/DATA BUS 08 | ROS DATA 40 |
| DABS09+41 | DATA BUS 09 | ADDR/DATA BUS 09 | ROS DATA 41 |
| DABS10+42 | DATA BUS 10 | ADDR/DATA BUS 10 | ROS DATA 42 |
| DABS11+43 | DATA BUS 11 | ADDR/DATA BUS 11 | ROS DATA 43 |
| DABS12+44 | DATA BUS 12 | ADDR/DATA BUS 12 | ROS DATA 44 |
| DABS13+45 | DATA BUS 13 | ADDR/DATA BUS 13 | ROS DATA 45 |
| DABS14+46 | DATA BUS 14 | ADDR/DATA BUS 14 | ROS DATA 46 |
| DABS15+47 | DATA BUS 15 | ADDR/DATA BUS 15 | ROS DATA 47 |
| CF1TA2-00 | MTESTA02 | CONTFP 01 | ROS DATA 00 |
| CF3TA0-P2 | MTESTA00 | CONTFP 03 | SIPE(OPT 2) |
| CF4TA1-P1 | MTESTA01 | CONTFP 04 | CIPE(OPT 1) |

Table 2-4. Processor Bus Signal Definitions

| Signal Name | Description |
|---|---|
| ADDR BUS (0D-0A) | Four high-order address lines gated out during Phase A low under control of bus control field (all bus output cycles). |
| ADDR/DATA OUT (00-15) | 16 lines (low order) used for address/data gated out during Phase A low under control of bus control field (all bus output cycles). |
| ADDR/DATA IN (00-15) | 16 data lines gated in during Phase A low under control of bus control field (DIN micro-op only). |
| CONTFP 01<br>CONTFP 03<br>CONTFP 04 | Control flip-flops 1, 3, and 4 are gated out of chip during Phase A low-order control of bus control field (all output bus cycles). |
| MTESTA00<br>MTESTA01<br>MTESTA02 | Three lines gated into monitor flip-flops during Phase A low when processor bus is not in an output cycle (i.e., bus control field equals DIN or NOP). |
| ROS DATA (XX) | ROS data lines are gated into the chip during Phase A high time. They are latched internally with the trailing edges of Phase B. |
| CIPE | Option 1 or Commercial Instruction Processor Enable (CIPE) signal. This signal, active low (steady state), signifies the presence of the CIP option to the chip. It is gated in when Phase A is high. |
| SIPE | Option 2 or Scientific Instruction Processor Enable (SIPE) signal. This signal, active low (steady state), signifies the presence of the SIP option to the chip. It is gated in when Phase A is high. |

### 2.2.3 Group 3 Signals

Group 3 consists of 19 shared interface lines. Table 2-5 lists the interface lines and the associated signals. Table 2-6 provides a definition of group 3 signals.

Table 2-5. Monitor/Interrupt and ROS Data/Option Interface/Function Names

| Interface Name | Function | |
|---|---|---|
| | $\overline{\phi}$A Input | $\phi$A Input |
| DATRQ0-32 | DTR0 | ROS DATA 32 |
| DATRQ1-33 | DTR1 | ROS DATA 33 |
| DATRQ2-27 | DTR2 | ROS DATA 27 |
| DATRQ3-26 | DTR3 | ROS DATA 26 |
| DATRQ4-34 | DTR4 | ROS DATA 34 |
| INTRQ2-28 | INTR2 | ROS DATA 28 |
| INTRQ1-29 | INTR1 | ROS DATA 29 |
| INTRQ0-30 | INTR0 | ROS DATA 30 |
| MTSTB0+13 | MTESTB00 | ROS DATA 13 |
| MTSTB1+14 | MTESTB01 | ROS DATA 14 |
| MTSTB2+15 | MTESTB02 | ROS DATA 15 |
| MTSTB3+16 | MTESTB03 | ROS DATA 16 |
| POWRON+22 | POWON | ROS DATA 22 |
| STSTB0+17 | STESTB00 | ROS DATA 17 |
| MTSTA3-0A | MTESTA03 | Unused |
| MEMPAR-P0 | MEMPAR | MMUE(OPT-0) |
| MEMPRES-P3 | MEMPRES | LAFE(OPT-3) |
| STSTA0-0A | STESTA00 | Unused |
| MREFSH-P4 | MEMFSH | MEMVAL(OPT-4) |

Table 2-6. Monitor/Interrupt and ROS Data/Option Signal Description (Sheet 1 of 2)

| Signal Name | Description |
|---|---|
| Data Request<br>DTR0 through DTR4 | Data Request is active low and is gated in during Phase A low. Data Request 0 (DTR0) is the highest priority data request and Data Request 4 (DTR4) is the lowest priority data request. |
| Interrupt Request<br>INTR0 through INTR2 | Interrupt Request is active low and is gated in during Phase A low. Interrupt Request 0 (INTR0) is the highest priority interrupt and Interrupt Request 2 is the lowest priority interrupt request. All interrupt requests are honored between instructions. |
| MTESTB00 through MTESTB03 | These are four lines gated in during Phase A low and are used as a source for a major branch based on some external events. User can define these events (such as CIP execution, SIP execution, additional opcode cracking, undefined I/O operations, etc.). They are also a source for test branch. |
| STESTB0/STESTA0 | These lines are gated in during Phase A low and can be used as a source for a test branch condition of external events. User can define these events. |
| MTESTA03 | This line can be used in conjunction with MTESTA00, MTESTA01, and MTESTA02 discussed in group 2 lines. Together they form another source for the major branch test groups.<br>NOTE<br>When taken as a group, this group can only sample external conditions on nonoutput bus cycles. MTESTB00 through MTESTB03 can sample external conditions on any cycle. MTESTA03 as an individual line can also test on any cycle.<br>This line is gated in during Phase A low and is a source for major branch and test branch conditions. |
| MEM PARITY ERROR | Memory parity error gated in during Phase A Signal is active low. It causes a memory error hardware interrupt. |

Table 2-6. Monitor/Interrupt and ROS Data/Option Signal Description (Sheet 2 of 2)

| Signal Name | Description |
|---|---|
| MEM PRESENT | Memory present signal is gated in during Phase A low. An active low signifies the memory is not present. It causes a nonexistent resource hardware interrupt. |
| POWON | POWON is a signal that becomes active high approximately 10 milliseconds after VCC (+5V) is in steady state, and becomes low approximately 2 milliseconds prior to VCC going off. This low going signal is used to cause a power failure interrupt. The state of this signal is sampled into the chip during Phase A low. |
| MEMRFRSH | Memory Refresh Request is an active low signal gated in during Phase A low. This signal will be generated at a rate equal to the need of the system's main memory refresh requirement. Receipt of this signal causes a hardware interrupt vector to a firmware routine. |
| ROS DATA (XX) | 14 ROS data lines are gated into the chip during Phase A high. They are latched internally with the trailing edge of Phase B. |
| MMUE | Option 0 or Memory Management Unit Enable (MMUE). This line, active low (steady state signal), signifies the presence of this option (including necessary firmware) to the chip; gated in during Phase A high. |
| LAFE | Option 3 or Long Address Format Enable (LAFE) signal. This signal, active low (steady state), signifies the presence of LAF mode to the chip. It is gated in when Phase A is high. |
| MEMVAL | Option 4 or Memory Valid Signal. This signal, active low (steady state), signifies the presence of a fully charged memory battery backup device. It is gated in when Phase A is high. |

### 2.2.4 Group 4 Signals

Group 4 consists of 13 shared interface lines. Table 2-7 lists the interface lines, and the associated signals. Table 2-8 provides a definition of group 4 signals.

Table 2-7. ROS Address and Data Bus Interface/Function Names

| Interface | $\overline{\phi}$A Output | ϕA Input |
|---|---|---|
| RSAD00+01 | ROS Address 00 | ROS Data 01 |
| RSAD01+02 | ROS Address 01 | ROS Data 02 |
| RSAD02+03 | ROS Address 02 | ROS Data 03 |
| RSAD03+04 | ROS Address 03 | ROS Data 04 |
| RSAD04+05 | ROS Address 04 | ROS Data 05 |
| RSAD05+06 | ROS Address 05 | ROS Data 06 |
| RSAD06+07 | ROS Address 06 | ROS Data 07 |
| RSAD07+08 | ROS Address 07 | ROS Data 08 |
| RSAD08+09 | ROS Address 08 | ROS Data 09 |
| RSAD09+10 | ROS Address 09 | ROS Data 10 |
| RSAD10+11 | ROS Address 10 | ROS Data 11 |
| RSAD11+12 | ROS Address 11 | ROS Data 12 |
| MEMKIL+31 | MEMKIL | ROS Data 31 |

Table 2-8. ROS Address and Data Bus Signal Description

| Signal Name | Description |
|---|---|
| ROS ADDRESS (00-11) | ROS address lines are gated out during Phase A low. They can be latched externally at the leading edge of Phase B. |
| ROS DATA (01-12,31) | ROS data lines are gated into the chip during Phase A high. They are latched internally with the trailing edge of Phase B. |
| MEMKIL | Signifies a memory error violation detected internal to the chip either as a result of MMU operation or a nonzero detected on BI (0D-0A) during SAF Mode (16-Bit Address mode). This signal is gated out of the chip during Phase A, and may be used to prevent alteration of memory contents. |

## 2.3 LEVEL 6 SIGNAL CROSS-REFERENCE

Table 2-9 provides a LSI-6 to Level 6 interface equivalency.

Table 2-9. LSI-6/Level 6 Interface Equivalency

| LSI-6 Mnemonic | Level 6 Mnemonic |
|---|---|
| CF1TA2-00 | BYTEXX-00 |
| CF3TA0-P2 | PROCED-P2 |
| CF4TA1-P1 | BUSYXX-P1 |
| MTESTA02 | BYTEX(IN) |
| MTESTA00 | PROCED(IN) |
| MTESTA01 | BUSY(IN) |
| CONTFP01 | BYTEX(OUT) |
| CONTFP03 | PROCED(OUT) |
| CONTFP04 | BUSY(OUT) |
| MTSTB0+13 | MIBGP0+13 |
| MTSTB1+14 | MIBGP1+14 |
| MTSTB2+15 | MIBGP2+15 |
| MTSTB3+16 | MIBGP3+16 |
| MTESTB00 | MIBGP0 |
| MTESTB01 | MIBGP1 |
| MTESTB02 | MIBGP2 |
| MTESTB03 | MIBGP3 |
| STSTB0+17 | TSTBR1+17 |
| MTSTA3-0A | ONBDCN-0A |
| STESTB00 | TESTBR1 |
| MTESTA03 | OBC |
| STSTA0-0A | MPLOCK-0A |
| STESTA00 | MPLOCK |

# Section 3
# HARDWARE DESCRIPTION

The LSI-6 microprocessor consists of five major internal hardware logic areas as shown in Figure 3-1. The five major logic areas, which will be discussed in the following subsections, are the Processor Bus (PB), the Internal Bus (BI), the data manipulation area (including ALU), the control area, and the Memory Management Unit (MMU).

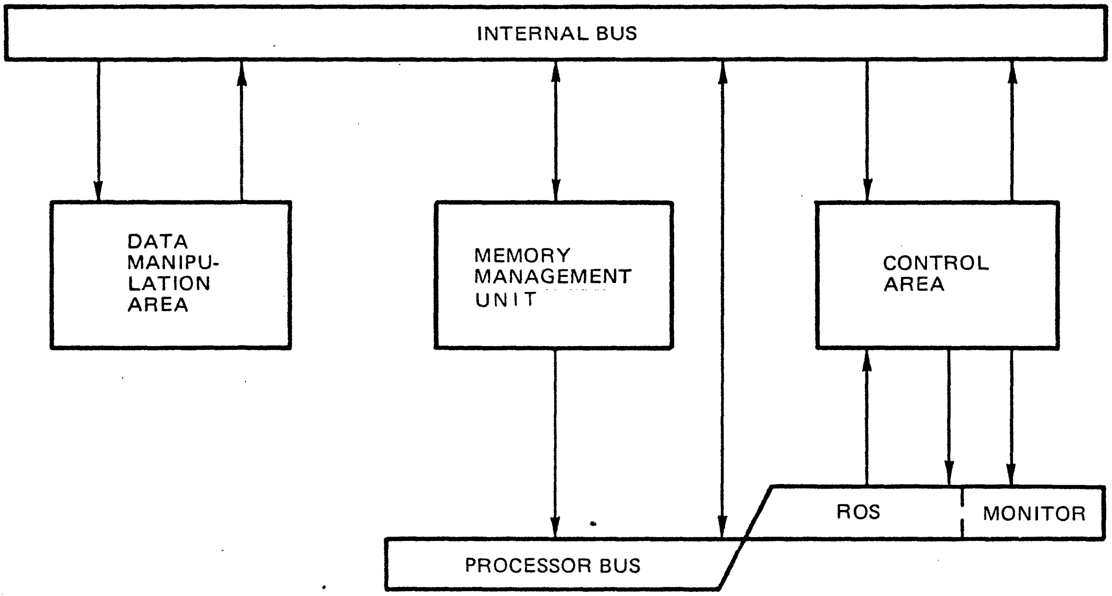


Figure 3-1. LSI-6 Major Block Diagram

### 3.1 PROCESSOR BUS

The processor bus consists of 20 address/data lines, one memory address violation line, and three general purpose control lines.

The address/data and the general purpose control lines are bidirectional during Phase A low. The direction and source (internal) for output operations is defined by the Bus Control Field (RDDT23 and RDDT25) of the current firmware word. The Memory Address Violation line (MEMKIL+31) is transferred out only during Phase A low.

### 3.1.1 Bus Address Lines DABS (0D-0A, 00-15)

When the 20 bus address/data lines are used as address lines (as defined by micro-ops: MMR, MWB1, MWB0 and MMW of the bus control field), the most significant 12 bits (D through A, 0 through 7) can originate from either the MMU (MMU adder output) or the internal bus (BI0D through BI0A, BI00 through BI07) depending on whether or not the MMU is enabled (MEMPAR-P0 activated low during Phase A high time [Option 0]). The low-order eight bits of the address (bits 8 through 15) will always originate from the internal bus (BI08 through BI15).

| MMU | From | To |
|---|---|---|
| | MMU ADDER (12 BITS) | DABS0D-0A, DABS00-07 |
| Enabled | BI08-15 | DABS08-15 |
| Disabled | BI0D-0A, BI00-15 | DABS0D-0A, DABS00-15 |

### 3.1.2 Bus Data Lines DABS (0D-0A, 00-15)

When the 20 bus address/data lines are used for data out (as designated by either the DOUT or DOUTM micro-ops), the output will be 20 bits wide. The origin of the data internal to the chip when using the DOUTM micro-op is as follows:

| MMU | From | To |
|---|---|---|
| | MMU ADDER (12 BITS) | DABS0D-0A, DABS00-07 |
| Enabled | BI08-15 | DABS08-15 |
| Disabled | BI0D-0A, BI00-15 | DABS0D-0A, DABS00-15 |

The origin of the data internal to the chip when using the DOUT micro-op is modified by other micro-ops TWNOUT and SWPOUT as shown below (both micro-ops are from a special command field).

| Micro-ops | | |
|---|---|---|
| Bus Control | Special Control | Origin to Data Out |
| DOUT | TWNOUT | Zeros -> DABS (0D-0A)<br>BI(08-15) -> DABS (00-07)<br>BI(08-15) -> DABS (08-15) |
| DOUT | SWPOUT | Zeros -> DABS (0D-0A, 00-11)<br>BI(0D-0A) -> DABS(12-15) |
| DOUT | -- | BI(0D-0A,00-15)->DABS(0D-0A,00-15) |

When DIN (bus control micro-op) is used, the data lines are directed into the chip. Data into the chip is 16 bits wide and the data path is as follows:

Zeros --> BI(0D-0A)

DABS (00-15) --> BI(00-15)

### 3.1.3 Memory Address Violation

The memory address violation signal (MEMKIL+31) is gated out when Phase A is low. It is used to prevent the destruction of main memory data when a memory address violation is detected internal to the chip. The source of this signal is either the detection of a nonzero value on BI(0D-0A) when 16-bit addressing mode is enabled or an MMU error is detected.

### 3.1.4 General Purpose Lines

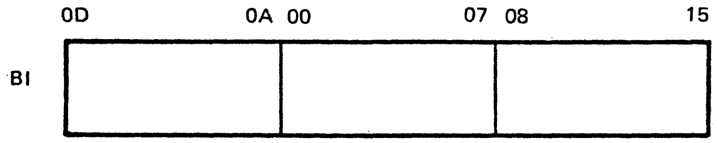
There are three bidirectional general purpose lines that can be used as control/response lines during input/output operations.

When used with processor bus output cycles (data or address out), they provide the contents of control flip-flops 1, 3, and 4 to the processor bus during Phase A low.

When the processor bus is not being transferred out (i.e., bus control bits RDDT 23 through 25 designated DIN or NOP), these lines are used as inputs to the major branch and test branch matrixes via monitor storage flip-flops MTESTA2, MTESTA0, and MTESTA1, respectively. These signals must be gated into the chip during Phase A low.

## 3.2 INTERNAL BUS

The Internal Bus (BI) is the major data path for communication of information between the other four areas of the processor chip. The internal bus is 20 bits wide and is designated as BIOD through BIOA, and BI00 through BI15. BIOD is the high-order bit and BI15 is the low-order bit as shown below.



There are 12 sources to the BI under control of 11 micro-ops. The ALU is a default source to BI if none of the other 11 defined micro-ops are used. A bus priority network (Prinet) is built into the LSI-6 to prevent damage to the chip if the user accidentally uses conflicting micro-ops to source the BI; however, resulting data to the BI is unspecified.

Tables 3-1 and 3-2 list the internal bus source micro-ops and load micro-ops.

## 3.3 DATA MANIPULATION AREA

The data manipulation area, as shown in Figure 3-2, performs arithmetic and logical operations on data and memory address generation. It is composed of the following eight elements.

P-Register
G-Register
Q-Register
Register File (Scratch Pad Memory)
ALU
Shifting Mechanism
Indicator Register
M-Register File

### 3.3.1 P-Register

The P-register is a 20-bit memory address register that contains either the program counter or a logical memory address. It is loaded from the internal bus, can be incremented or decremented, and the register output is transferred to the internal bus.

The following micro-ops are used with the P-register operation:

| Micro-Op | Remarks |
|---|---|
| LDP | BI -> P |
| PP1 | P+1 -> P |
| PM1 | P-1 -> P |
| BIFP | P -> BI |

Table 3-1. BI Source Micro-Ops

| Source | Micro-Op | Remarks |
|---|---|---|
| Processor Bus | DIN | Zeros -> BI(0D-0A);<br>DABS(00-15) -> BI(00-15) |
| P-Register | BIFP | P(0D-0A,00-15) -> BI(0D-0A,00-15) |
| I-Register | BIFI | Zeros -> BI(0D-0A,00-07)<br>I-register (00-07) -> BI(08-15) |
| M-Register File | BIFM | Ones -> BI(0D-0A,00-07)<br>M(x)REG(00-07) -> BI(08-15) |
| MMU Register File (Base Word) | BIFMMUH | Zeros -> BI(0D-0A)<br>MMURFx(00-15) -> BI(00-15) |
| MMU Register File (Size Word) | BIFMMUL | Zeros -> BI(0D-0A)<br>MMURFx(16-31) -> BI(00-15) |
| A-Port ALU (Register File) | BIFSP | A-Port (0D-0A,00-15) -> BI(0D-0A,00-15)<br>If selected register is 16 bits, then bit 0 is sign propagated through the upper four bits (D through A) |
| F-Register Mask | BIFMK | A One is sourced to bit position BI(00-15) dependent upon value of count in F-register (12-15). Zeros are sourced to all other positions.<br>BI(00) -> BI(0D-0A) |
| ROS Address Field (CONST) | INCK | RDDT(04) -> BI(0D-0A,00-07)<br>RDDT(05-12) -> BI(08-15) |
| Special Control Field (CONST) | RDL | RDDT(39) -> BI(0D-0A,00-07)<br>RDDT(40-47) -> BI(08-15) |
| Special Control Field (CONST) | RDH | RDDT(40) -> BI(0D-0A)<br>RDDT(40-47) -> BI(00-07)<br>RDDT(39) -> BI(08-15) |
| ALU Output | DEFAULT (none of the above) | ALU(0D-0A,00-15) -> BI(0D-0A,00-15) |

Table 3-2. BI Load Micro-Ops

| Load | Micro-Ops | Remarks |
|---|---|---|
| PB | DOUT | BI(0D-0A,00-15) -> DABS(0D-0A,00-15) |
|  | DOUTM<br>MMR<br>MWB1<br>MWB0<br>MMW | Relocated memory address-DABS |
| P-Register | LDP | BI(0D-0A,00-15) -> P(0D-0A,00-15) |
| I-Register | LDI | BI(08-15) -> I-register (00-07) |
| M-Register | LDM | BI(08-15) -> Mx(00-07) |
| MMU Register File (Base Word) | WMBS | BI(00-15) -> MMURFx(00-15) |
| MMURF (Size Word) | WMSZ | BI(00-15) -> MMURFX(16-31) |
| F-Register | LDFR | BI(00-11) -> F-register (00-11) |
| FCNT | LDFT | BI(11-15) -> FCNT |
| Register File (RF) (ALU-SPM) | SPW | BI(0D-0A,00-15) -> RFx(0D-0A,00-15)<br>See reference for other modifications |
| G-Register | LDG | BI(0D-0A,00-15) -> G(0D-0A,00-15) |
| Q-Register | LDQ | BI(00-15) -> Q(00-15) |

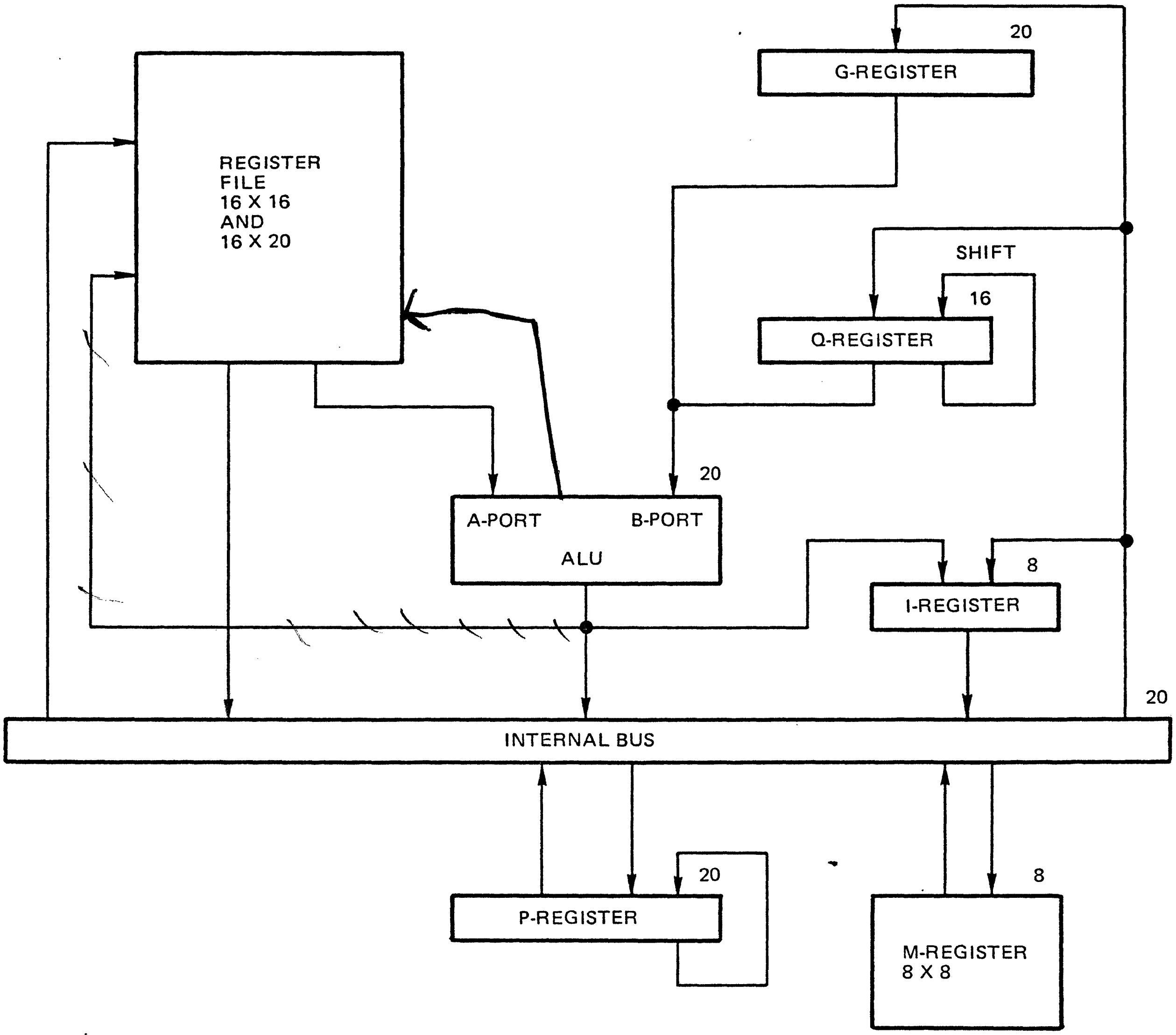


Figure 3-2. Data Manipulation Block Diagram

### 3.3.2 G-Register

The G-register is 20 bits wide and is used to hold addresses for ALU operations. It is loaded from the internal bus with the LDG micro-op and the output is sent to the B-port of the ALU.

### 3.3.3 Q-Register

The Q-register is a 16-bit register that provides operand shifts, and holds secondary operands for the ALU. It is loaded from the internal bus and the output is sent to the B-port of the ALU. Register output can be sign extended to 20 bits from Q0, Q8, or Q9 for ALU operations.

The Load Q-register micro-op (LDQ) is used for loading from BI when shifting micro-ops are not used.

### 3.3.4 Register File

The register file can be loaded from the ALU output or the BI. When loaded from BI, data comes direct, or with byte 0 (BI00 through BI07) swapped with byte 1 (BI08 through BI15), or BI12 through BI15 to RF0D through RF0A. Two write lines exist: one loading bits 00 through 15, the other loading bits 0D through 0A. The register file output feeds the A-port of the ALU. The register address is taken from either the firmware word register (RDDT) or the information in the F-register.

During Phase A high time, the address for the selected register file is generated under control of the register file address field. The selected register file is read out during Phase A high and its contents are latched in an output register during Phase A low. This output register is then available to two possible destinations, the A-port of the ALU (under control of the ALU control field, RDDT bits 18 through 22) and the internal bus (under control of special control field, RDDT bits 35 through 41).

The selected register file is written near the end of Phase A low time. The general command for writing the register file is supplied by RDDT bit 26 (register file load). This general command can be modified by certain special control field micro-ops or subcommands. Table 3-3 provides a list of possible write operations to the register file.

The register file description in subsections 3.3.4.1 through 3.3.4.9 are in conjunction with Level 6 Model 43 terminology. Figure 3-3 shows the register file topology and the hexadecimal locations of the registers.

Table 3-3. Register File Special Control Field Write Operations

| SPW | Special Control Field Micro-Ops | Resulting Operation |
|---|---|---|
| 0 | | No change in register file |
| 1 | IWDTA[a] | Write register file bits 00-16 |
| 1 | SPXDTA[b] | Write register file (0D-0A) from BI (12-15) |
| 1 | BIFSP | Write register file (all bits) from ALU |
| 1 | SPFAU | Write register file (all bits) from ALU |
| 1 | SPFBIX[c] | Write register file (0-7) from BI (8-15)<br>Write register file (8-15) from BI (0-7) |
| 1 | All register file Shift[d] Subcommands | Write register file (all bits) from shifted ALU result |
| 1 | None of the above special control field micro-ops | Write register file (all bits) from BI (all bits) |

[a] See special control field for complete definition of micro-ops.

[b] Normally used with 20-bit registers.

[c] Normally used with 16-bit registers. If a 20-bit register is selected BI12-BI15 to Register File RF0D-RF0A.

[d] Normally used with 16-bit registers. However, when used with a 20-bit register file, the effect is a 20-bit open left or 20-bit open right shift. Micro-op SFV0A supplies ALU (0A) to ALU (0D) for open right shifts.

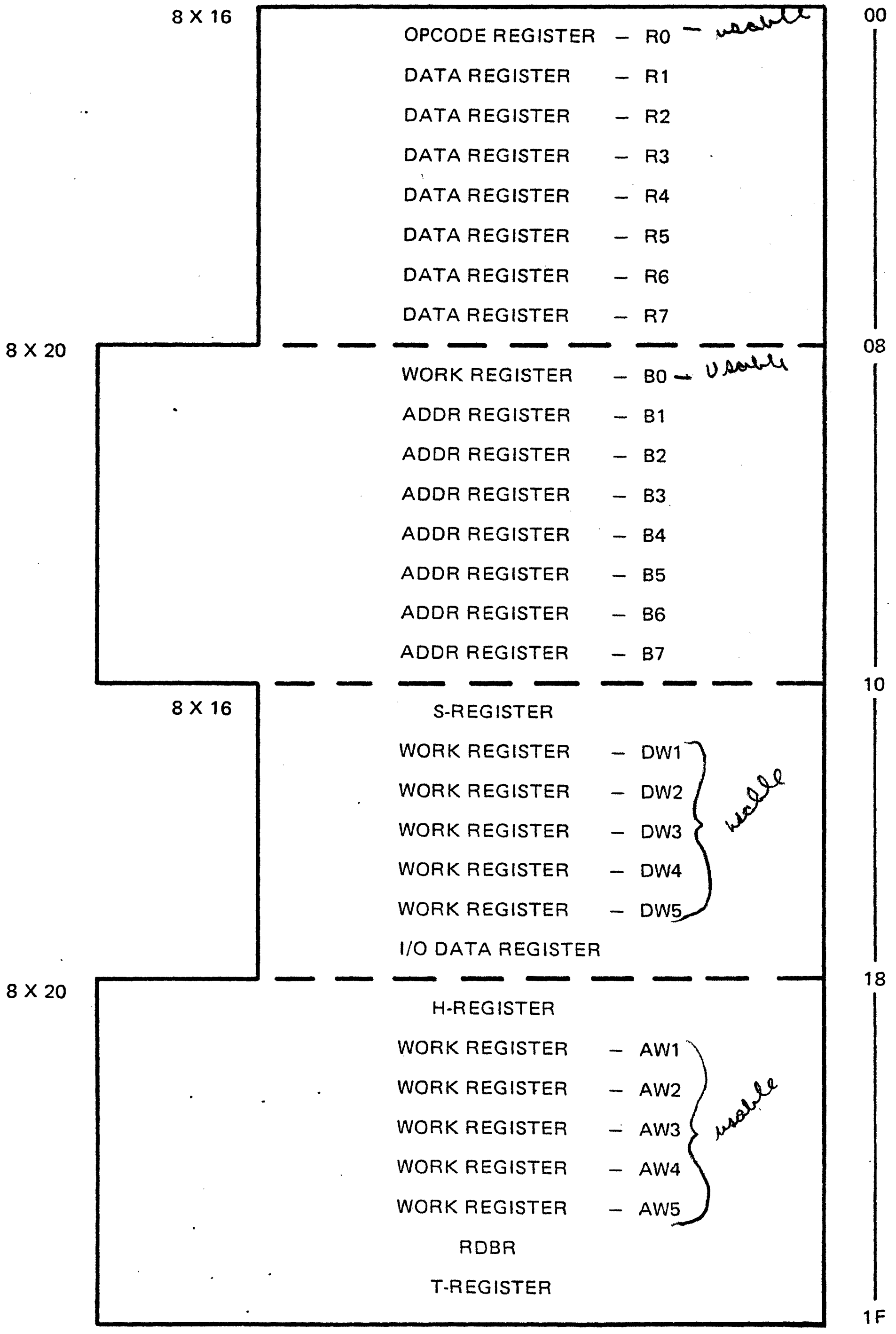


Figure 3-3. Register File Topology

### 3.3.4.1 DATA REGISTERS

There are seven 16-bit software addressable data registers, R1 through R7. They can be loaded from or stored into memory on either a word or byte basis. Each register can be used as operands in arithmetic, logical, and compare operations. (R0 is a 16-bit register that is used to hold a copy of the executing instruction.)

### 3.3.4.2 ADDRESS REGISTERS

There are seven software-addressable base registers, B1 through B7. They are 20 bits wide and can be used to hold main memory addresses. B0 is a 20-bit working register used in Panel mode.

### 3.3.4.3 WORK REGISTERS

Six 16-bit (DW1 through DW5) and five 20-bit (AW1 through AW5) registers are available for temporary storage of information during firmware operations.

NOTE

AW1 is normally used as the "effective address" storage element.

### 3.3.4.4 T-REGISTER

The T-register is the stack address register and is 20 bits wide.

### 3.3.4.5 S-REGISTER

System keys and processor security keys are contained in this 16-bit register. The bit fields are defined as follows:

| 0 | 1 2 | 3 5 | 6 9 | 10 15 |
|---|---|---|---|---|
| 0 | R | 000 | ID NO. | LEVEL |

R - Ring Number 00,01 = user states
10,11 = privilege states

ID No. - Each processor in the system has an identification number.

Level - The 6-bit level field defines the interrupt priority on which the processor is currently operating. Zero is the highest and 63 the lowest priority.

#### 3.3.4.6 H-REGISTER

The history (H) register is 20 bits wide. It contains the history of the program counter under firmware control.

#### 3.3.4.7 REMOTE DESCRIPTOR BASE REGISTER

The Remote Descriptor Base Register (RDBR) is used by the Commercial Instruction Processor (CIP). It is 20 bits wide.

#### 3.3.4.8 I/O DATA REGISTER

The I/O data register is a 16-bit data working register that can be used for temporary storage during I/O data transfer operations.

### 3.3.5 Arithmetic Logic Unit

The LSI-6 has an ALU with full 16- or 20-bit capabilities. Overflow and carry functions are generated out of both the 16th and 20th bit positions. The 16-bit version of the ALU is normally used when handling data, while the 20-bit version is normally reserved for address modifications and transfers.

The ALU has two ports for operand inputs, the A- and B-port. The A-port can accept either a 16- or 20-bit register file input. In the case where the register file selected is a 16-bit wide data register, the most significant bit is sign extended to 20 bits. The A-port can also select a value of zero as its input as specified by the ALU control field (RDDT bits 18 through 22). The B-port can select as its inputs: 20 bits from the G-register, 16, 8 or 7 bits right justified from the Q-register. The most significant bit of the field selected from the Q-register is always sign extended to present a 20-bit operand to the B-port. The B-port can also select a value of Zero as its input.

The output of the ALU can be directed to either the BI or the register file. When directed to the register file, its path can be direct (BIFSP or SPFAU, special control field micro-ops) or shifted left or right one bit position (also under control of special control field--all register file shift micro-ops). The ALU output is directed to the BI whenever there is no micro-op called to source the BI (it is the default source for the BI).

The carry and overflow conditions for both 16- and 20-bit operations are stored in temporary flip-flops (see subsection 4.1.2) for testing during the following cycle.

The following is a list of micro-ops used for ALU control.

| Micro-Op Name | | | |
|---|---|---|---|
| SPANDG | SPANDQ8 | ZERO | SPPQ8 |
| GNOT | Q8 | ZMG | SPMQ8 |
| G | SPXORQ8 | SPPG | SPPQ9 |
| SPORG | SPORQ8 | SPMG | SPMQ9 |
| SPANDQNOT | SPANDQ | MINUS1 | SPPQ |
| QNOT | Q | ZMQ | SPMQ |
| SPNOT | SPXORQ | SPM1 | SPMQM1 |
| SP | SPORQ | SPP1 | SPPQP1 |

COMMENTS

SP indicates register file (x)
G indicates G-register
Q indicates Q-register
Q8 indicates sign propagation from Q08
Q9 indicates sign propagation from Q09
Z indicates input to A-port is Zero
ZERO indicates output equals Zero
MINUS1 indicates output equals all Ones
AND, OR, NOT, XOR, P, and M are all operations

### 3.3.6 Shifting Mechanism

The LSI-6 has the ability to perform various shift operations (i.e., open/closed, arithmetic/logical, left/right) on either 16- or 32-bit operands.

Sixteen-bit data shifting can be accomplished in one of two ways. The first takes place from output of the ALU into the register file and the second takes place in the Q-register. These two operations can be concatenated to perform the 32-bit data shift operations.

Three B-field shift micro-ops (SHOPRI, SHOPRF, and SHOP) are used to implement the Level 6 software shift instructions. The shift micro-ops are combined with the F-register decode (the F-register contains the Level 6 software instruction word for this operation) to determine the shift type, direction, and necessary filler bits. The following chart shows the decoded F-register bits and the corresponding shift type.

| F-Register Bits 8 Through 11 | Shift Type | Level 6 Instruction |
|---|---|---|
| 0 | Single open left shift | SOL |
| 1 | Single closed left shift | SCL |
| 2 | Single arithmetic left shift | SAL |
| 3 | Double closed left shift | DCL |
| 4 | Single open right shift | SOR |
| 5 | Single closed right shift | SCR |
| 6 | Single arithmetic right shift | SAR |
| 7 | Double closed right shift | DCR |

| F-Register Bits 8 Through 11 | Shift Type | Level 6 Instruction |
|---|---|---|
| 8 | Double open left shift (≤15) | DOL |
| 9 | Double open left shift (≥16) | DOL |
| A | Double arithmetic left shift (≤15) | DAL |
| B | Double arithmetic left shift (≥16) | DAL |
| C | Double open right shift (≤15) | DOR |
| D | Double open right shift (≥16) | DOR |
| E | Double arithmetic right shift (≤15) | DAR |
| F | Double arithmetic right shift (≥16) | DAR |

F-register bits 1 through 3 contain the R-register to be shifted, and F-register bits 12 through 15 or 11 through 15 contain the number of positions to be shifted. When F-register bit 8 equals a Zero, the number of positions to be shifted is determined by F-register bits 12 through 15; and when F-register bit 8 equals a One, the number of positions to be shifted is contained in F-register bits 11 through 15. A special case exists when the count field contains a value of Zero. In this case, the number of positions to be shifted is contained in register file location R1 (general purpose register).

When a double word is selected (i.e., F-register bits 8 through 11 equals a hexadecimal 3, 7, 8, 9, A, B, C, D, E, or F), then F-register bits 1 through 3 must equal a 3, 5, or 7. This is necessary because it requires a combination of an implied even-numbered register and an explicitly addressed odd-numbered register to perform a double-word shift operation. When register R3 is explicitly addressed, register R2 is the implied addressed register. The even-numbered register contains the most significant bits of the double word.

### 3.3.7 Indicator (I) Register

The I-register is eight bits wide, containing various single bit indicators in the following format:

| 0 | | | | | | | 7 |
|---|---|---|---|---|---|---|---|
| OV | 0 | C | B | I | G | L | U |

OV - Overflow indicator: It is set when any of the data registers R1 through R7 overflow; e.g., when a 16-bit arithmetic result produced is larger than the capacity of the register (under op-code control, out of ALU).

0 - Always Zero.

C - Carry indicator: This is set when the logical capacity of a register is exceeded. The carry indicator is generated from the ALU.

B - Bit test indicator: This bit gives the state of the last bit tested (primarily for bit test operations).

I - Input/output indicator: It indicates whether the last I/O command was accepted by the I/O controller.

G - Greater than indicator.*

L - Less than indicator.*

U - Unlike sign indicator.*

The I-register can also be loaded from the internal bus (BI08 through BI15 to IR0 through IR7), and its output goes to the internal bus (IR0 through IR7 to BI08 through BI15, with Zeros to BI0D through BI07).

### 3.3.8 Mode (M) Register File

The LSI-6 has an M-register file that has eight registers, each eight bits wide (8 by 8). This file can be loaded from the BI using the LDM micro-op or sourced to BI using the BIFM micro-op. When using either of these micro-ops, the M-register address is supplied by the least significant three bits of the register file address register. When neither of these micro-ops are called, the M-register address defaults to location 1 (M1 register).

M-register bits 1 through 7 of any M-register are testable, with the register file address field (refer to subsection 4.2) selecting the desired M-register, and the desired bit within the register being selected by a code of 1 through 7 in F-register bits (0 through 3). This causes the setting of a temporary flip-flop (MREGBIT+00) when a One bit is detected (refer to subsection 3.4.2). Bit 0 of the selected M-register is sampled in each cycle by a temporary flip-flop (MDREGO+00). Testing of these temporary flip-flops is accomplished on the following cycle using micro-op TBMRGX and TBMRGO, respectively (refer to subsection 4.1.2).

For Level 6/43 functionality the M1, M3, M4, and M5 registers are specified as shown below. M2, M6, M7 are reserved for future Level 6 functionality.

*These three indicators are controlled by micro-ops during the compare instructions and contain the result of the last compare operation executed. Typically, the comparison involves a register and a word from memory. The indicators show whether the register contents are greater than or less than the memory word.

- The NATSAP pointer register (M0) is a 16-bit data working register used in trap routines to point to the Next Available Trap Save Area Pointer (NATSAP). This pointer will contain one of four values, determined by software. The pointer is initialized by firmware to a value of 0010.

- M1 Register is the trap enable mode control register.

| 0 | | | | | | | 7 |
|---|---|---|---|---|---|---|---|
| J | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

J - Trace Trap Enable Bit

0 = disable
1 = enable

1-7 - Overflow Trap Control Bits

0 = disable
1 = enable

When the J-bit is enabled, all jumps and branches that are executed in a program cause a trap to the trace entry location. The seven overflow trap control bits are associated with the seven data registers R1 through R7. If overflow in any one of these occurs and the corresponding trap control bit is set, an overflow trap pending flip-flop is set causing a software trap to occur during a major branch on SWINT. The overflow trap pending flip-flop can also be tested with a test branch condition TBOVTP (RDDT 3:6 = 18 hexadecimal) and can be reset using the RESOVT (A-field = 0, 1; B-field = 7) micro-op.

- M3 Register contains the control information for the Commercial Instruction Processor (CIP).

| 0 | 1 | 2 7 |
|---|---|---|
| OV | TR | UNUSED |

OV - Overflow Trap Mark

0 = disable
1 = enable

TR - Truncated Trap Mask

0 = disable
1 = enable

- M4 and M5 Registers: The M4 and M5 registers contain control information for the Scientific Instruction Processor (SIP). The bit fields are defined as follows:

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| M4 = | R/T | RFU | ML$_1$ | AL$_1$ | ML$_2$ | AL$_2$ | ML$_3$ | AL$_3$ |
|  |  |  | SA 1 |  | SA 2 |  | SA 3 |  |

where:

R/T - Round/Truncate Mode

0 = Truncate
1 = Round

SA 1 - Scientific Accumulator 1
SA 2 - Scientific Accumulator 2
SA 3 - Scientific Accumulator 3

ML - Memory Length - Length of main memory data field associated with this SA

0 = Two Words
1 = Four Words

AL - Accumulator Length - Length of the value in the scientific accumulator.

0 = Two Words
1 = Four Words

|  | 0 | 1 | 2 | 3 | 4 | 7 |
|---|---|---|---|---|---|---|
| M5 = | EUM | RFU | SEM | PEM | RFU |  |

EUM - Exponent Underflow Trap Mask

0 = Trap Disable
1 = Trap Enable

SEM - Significance Error Trap Mask

0 = Trap Disable
1 = Trap Enable

PEM - Precision Error Trap Mask

0 = Trap Disable
1 = Trap Enable

RFU - Reserved for Future Use - (Must be Zero)

## 3.4 CONTROL AREA

The control area (Figure 3-4) of the LSI-6 processor chip is logically divided into three areas:

1. Input latches for control
2. Testable registers
3. Next address generation

### 3.4.1 Input Latches for Control

There are three groups of latches that store information generated external to the chip, but used for internal control operations.

- ROS DATA REGISTER (RDDT 00-47): This is a 48-bit wide register which holds the firmware currently being executed. It is loaded from the external ROS during Phase A high.

- MONITOR REGISTER: This register holds 22 bits of system status. The system status bits listed below are dynamic in nature and are loaded during Phase A low. The bits marked with an asterisk will cause a hardware interrupt to occur, while the others are testable by firmware.

MEMPRES*
MEMPAR*
MEMRFSH*
DTR0*
DTR1*
DTR2*
DTR3*
DTR4*

INTR0
INTR1
INTR2
MTESTA00
MTESTA01
MTESTA02
MTESTA03

STESTA
STESTB
POWON
MTESTB00
MTESTB01
MTESTB02
MTESTB03

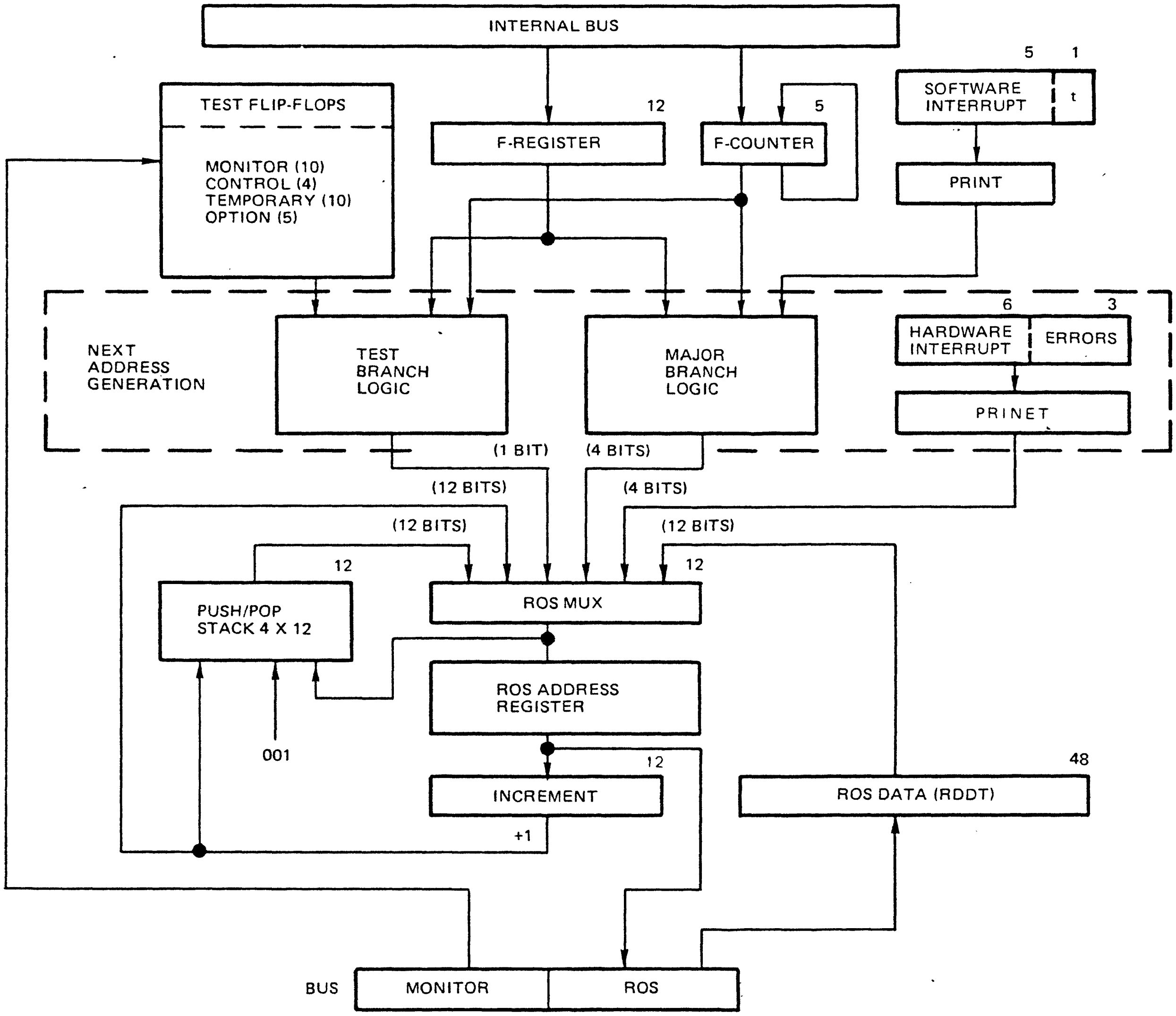


Figure 3-4. Control Major Block Diagram

- OPTION REGISTER: This register holds five bits which sample the status of the system options. These options should be thought of as static in nature. Option 0 (MMUE) and Option 3 (LAFE) are directly tied to control of certain hardware data paths and therefore their use is predetermined. All five option bits listed below are testable by firmware.

| Option | Level 6 Function | Remarks |
|---|---|---|
| 0 | MMUE | Enables MMU |
| 1 | CIPE | Refer to subsection 2.2.2 |
| 2 | SIPE | Refer to subsection 2.2.2 |
| 3 | LAFE | Enables 20-bit address mode when active low. (Refer to subsection 2.2.3.) |
| 4 | MEMVAL | Refer to subsection 2.2.3 |

### 3.4.2 Testable Registers

The following registers are testable by firmware using test branch and major branch micro-ops.

- F-REGISTERS (F-Counter): The F-Register is a 16-bit instruction register, loaded from the internal bus. All bits are testable by firmware. The low-order four bits also constitute the low-order four bits of a 5-bit counter (FCNT). These five bits can be loaded from the internal bus (BI11 through BI15 to FCNT11 through FCNT15), incremented and decremented. The four low-order bits are also decoded such that a 16-bit mask can be placed on the internal bus (BI00 through BI15) under firmware control. FCNT11 is held at Zero for shift-short instructions; i.e., F-register = ONNN 0000 0XXX XXXX where NNN is not 000 and X is a "don't care."

- TICOS: This flip-flop is set whenever the leading or trailing edge of an external square wave, nominally 60 hertz, is detected. It is reset with firmware. The flip-flop is testable by firmware via the software interrupt Prinet.

- WDT: Whenever set, this flip-flop indicates that the Watchdog Timer (WDT) is ON. It is set and reset with the WDTN and WDTF instructions, respectively, and is testable by the firmware.

- RTC: Whenever set, this flip-flop indicates that the Real-Time Clock (RTC) is ON. It is set and reset with the RTCN and RTCF instructions, respectively, and is testable by the firmware.

- CONTROL FLIP-FLOPS 1-4: Four individual flip-flops are available to be set or reset or to have a bit transferred under firmware control. They are testable by the firmware.

- TEMPORARY FLIP-FLOPS: Ten temporary flip-flops exist (TF0 through TF9) which are loaded during each firmware cycle with the following bits: BI00, BI00 through BI15 = 0, Carry (16), Overflow (16), BI0D through BI15 = 0, Carry (20), Overflow (20), BI15, BI00 through BI11 = 0, and BI0D through BI0A = 0. In addition, there are two M-register temporary flip-flops (refer to subsection 3.3.8) and they are MREGBIT+00 and MDREG+00.

- OVERFLOW TRAP FLIP-FLOP: If an overflow occurs in any of the data registers R1 through R7 of the register file and the corresponding overflow trap bit in the M1 register (bits M1 through M7) is set, then the overflow trap pending flip-flop is set. This flip-flop can be tested by the test branch micro-op TBOVTP or by a software interrupt Prinet via the major branch micro-op MBSWIP. The overflow flip-flop is reset using the RESOVT micro-op.

### 3.4.3 Next Address Generation

The next address generation section is the essence of the control area as it contains the hardware necessary for sequencing the Read Only Storage (ROS). Below are listed the major hardware registers and/or networks included in the LSI-6 chip for next address generation.

- ROS ADDRESS REGISTER - RAR(00-11): This is a 12-bit register which contains the address for the external ROS. During Phase A high time, this contains the value of the current ROS data word. During Phase A low, it contains the address of the ROS for the next microcycle and will be driven out of the chip on the ROS address bus.

- ROS ADDRESS INCREMENTER: This is a 12-bit incrementer which will be used to source the Push/Pop stack during Push micro-op, and provide the next RAR value for INC and INCK micro-ops.

- PUSH/POP STACK: This is a 4 by 12 Last In First Out (LIFO) array used for subroutines and hardware interrupts. It is initialized to a hexadecimal value of 001 during clear time, and its bottom location is set to 001 for each POP (return micro-operation). A Push micro-operation causes the top of the stack to be sourced by the ROS address incrementer, and a hardware interrupt causes the top of the stack to be sourced by the input network to the RAR.

- TEST BRANCH MATRIX: There are 64 test conditions resulting in a two-way branch address to the RAR.

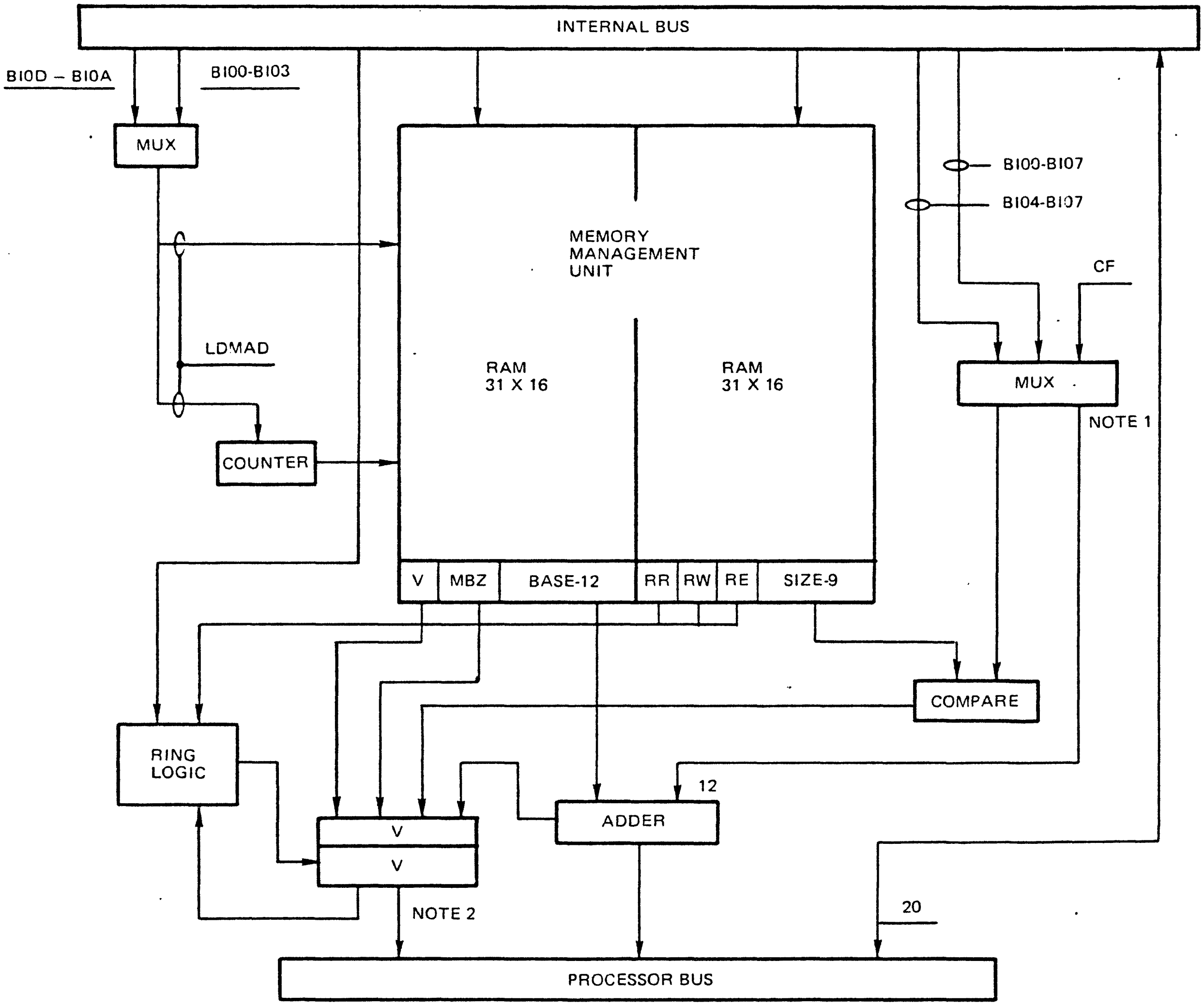
- MAJOR BRANCH MATRIX: There are 15 major branch test matrixes. The majority of the inputs to these matrixes are from the F-register (in various combinations). Other inputs are from monitor and option bits.

- HARDWARE INTERRUPT PRINET: There are 10 possible conditions that can cause a hardware interrupt. The hardware interrupt Prinet allows the prioritization and vector address generation for these conditions.

## 3.5 MEMORY MANAGEMENT UNIT

The Memory Management Unit (MMU) section of the LSI-6 (Figure 3-5) is composed primarily of (1) a register file (31 by 32) for holding an array of memory segment descriptors, (2) a 12-bit address for base relocation, (3) a 9-bit comparator for checking the size of a memory segment, (4) several 2-bit ring comparators for evaluating access rights to a given segment, and (5) storage flip-flops for indicating potential memory violations. For a detailed description of the MMU, refer to the Memory Management Unit manual, Order No. FN34-02 and Honeywell EPS-1, 60130079.

The MMU of the LSI-6 has the equivalent functionality of the Model 43 in the Level 6 family. The MMU is an optional feature of the LSI-6 which is activated by MEMPAR-P0 being held active low during Phase A high time (Option 0). Activation of the option (MMUE or Option 0) signifies the firmware exists in the ROS external to the chip to support this feature (Validate, Activate Segment Descriptor Instructions). Should this firmware not be installed, the MEMPAR-P0 lines should be driven high during Phase A high time to ensure deactivation of this feature, otherwise unspecified results can occur.



NOTES

1. THE LIMITED ACCESS CHECK EQUALS THE RANGE CHECK FOR VALIDATE, CIP, AND SIP INSTRUCTIONS.
2. HIGH-ORDER MEMORY ADDRESS BIT 21 (MEMKIL) IS SAMPLED BY ALL MEMORIES AND WILL PREVENT MEMORY ACCESS WHEN TRUE.

Figure 3-5. Memory Management Unit Block Diagram

### 3.5.1 MMU Theory of Operation

During any main memory address cycle (assuming the address is generated on chip), the BI is continually sampled by the MMU hardware. The address on the BI can be thought of as a logical address (as opposed to a physical memory address) containing a segment number, a block number, and an offset value. The MMU extracts the segment number for use in addressing its register file (containing segment descriptors for all possible segments). The proper descriptor is read from that addressed location. Next, the block number is extracted from the BI and added to a base (12 bits) value containing in the descriptor (effectively relocating the logical address). The block number is also checked against the size field of the segment descriptor. Depending upon the type of memory operation (read, write, or execute), the effective ring number is compared against one of three ring numbers contained in the segment descriptor. Precluding any violations, the output of the relocation adder is concatenated with the offset bits of the BI and delivered to the processor bus as a physical memory address.

### 3.5.2 MMU Hardware

- MMU REGISTER FILE ADDRESSING: The MMU register file addressing can be accomplished by two methods. The first is during normal main memory address cycles. The contents of the internal bus bits BI0D through BI0A and BI00 through BI03 are used to generate an address and load it in the MMU address latch during the trailing edge of Phase A. This latched address is then used to address the register file array.

  The second is via the MMU address register. The MMU address register is loaded with the contents of the internal bus bits BI0D through BI0A and BI00 through BI03 during the leading edge of Phase A using the LDMAD micro-op (refer to subsection 4.8.2). The contents of the MMU address register is then used to address the register file array. The second method can only be used to address the register file array during operations that read or write the register file array data (WMBS, WMSZ, BIFMMUH, and BIFMMUL) or during limited read/write access checks using LRA and LWA micro-ops.

  The MMU address generation for either of the methods described above is accomplished in the following manner. If BI0D through BI0A equals Zero, then the four least significant bits of either the MMU address latch or the MMU address register (determined by LDMAD) is loaded with the contents of BI00 through BI03 and the high-order bit (5) of either the latch or register is set to a Zero indicating a small segment. If BI0D through BI0A does not equal zero, then the four least significant bits of either the latch or register (determined by LDMAD) is loaded with the contents of BIOD through BIOA and the high-order bit (5) of either the latch or register is set to a One indicating a large segment.

When addressing the MMU using the second method, the MMU address register is incremented when either a WMSZ or BIFMMUL micro-op is used in the firmware word. The increment occurs at the leading edge of Phase A.

- <u>MMU DESCRIPTOR ARRAY</u>: A 31 by 32 register file contains segment descriptors for memory management. Locations 0 to 15 are used to describe small memory segments 0 to 15, each containing a maximum of 4K words of memory. Locations 17 to 31 describe 15 large segments each containing up to 64K maximum. Total address space is one megaword. The array is organized as follows:

| | 0 | 1 3 | 4 15 | 16 17 | 18 19 | 20 21 | 22 | 23 31 |
|---|---|---|---|---|---|---|---|---|
| 0 | V | MBZ | BASE | RR | RW | RE | MBZ | SIZE |
| | | | | | | | | |
| 15 | | | | | | | | |
| *17 | | | | | | | | |
| | | | | | | | | |
| 31 | | | | | | | | |

V - Validity bit; if this bit is a Zero, the segment is undefined.

MBZ - Must be all Zeros

BASE - 12-bit base address.

RR - Read ring value.

RW - Write ring value.

RE - Execute ring value.

SIZE - This 9-bit field defines the size of the memory segment 256-word blocks.

The array is loaded from the internal bus as two 16-bit words (BI00 through BI15 to MMURF00 through MMURF15 and BI00 through BI15 to MMURF16 through MMURF31). Likewise, the array can be read for test purposes (MMURF00 through MMURF15 to BI00 through BI15 and MMURF16 through MMURF31 to BI00 through BI15.

*Address location 16 does not exist as a physical entry in the array.

- MMU LATCHES: Two latches are provided to hold the processor's current ring number, and two for the effective ring number. The former is loaded from the internal bus (BI01 and BI02) while the latter is either a copy of the current ring, a copy of the write ring of the accessed segment descriptor, or set to Ones by the firmware.

- MMU FLIP-FLOPS 1-4: The MMU flip-flops, controlled by the MMU field (bits 31 through 33) of the ROS word, store MMU error conditions.

  - MMU FLIP-FLOP 1 AND 3 indicate a nonexistent resource violation (i.e., the address accesses a memory location which is not within allowable bounds), or the segment descriptor is undefined. When set, this will cause an immediate hardware trap unless the next firmware word contains a Clear Memory Error (CME) micro-op. MMU flip-flop 3 indicates the same function, but is testable by the firmware. It will not cause a hardware interrupt.

  - MMU FLIP-FLOP 2 AND 4 indicate an access violation; i.e., the processor does not have read, write or execute permission for the memory location addressed. This will cause an immediate hardware trap unless the next firmware word contains a Clear Memory Error (CME) micro-op. MMU flip-flop 4 indicates the same function, but is testable by the firmware. It will not cause a hardware interrupt.

- MMU ADDER: This is a 12-bit adder used to add the block number (from BI) to the base value contained in bits 4 through 15 of the MMURF . The output of this adder sources DABS (0D through 0A, 00 through 07) of the processor bus under control of the bus control field.

- MMU BLOCK COMPARATOR: This is a 9-bit comparator which is used to check the block number from BI against bits 23 through 31 of the MMURF for a size violation. This violation causes setting of MMU flip-flop 1 or 3 as noted above.

# Section 4
# FIRMWARE FORMAT AND DESCRIPTION

The LSI-6 has a 48-bit firmware word (RDDT00 through RDDT47) which is used to define up to 12 simultaneous operations or subcommands within a given microcycle. Figure 4-1 illustrates the coded fields within the ROS data word (firmware word) and Table 4-1 defines the coded fields.

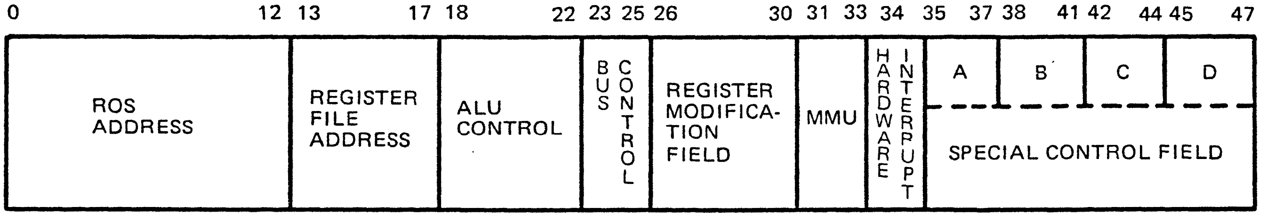


Figure 4-1. ROS Firmware Word Fields

Table 4-1. ROS Data Field Description

| Coded Fields | Description |
|---|---|
| ROS Address (RDDT00 through RDDT12) | (13-bit field) defines or describes the method for generating the next ROS address to be used in a given sequence. |
| Register File Address (RDDT13 through RDDT17) | (5-bit field) defines or describes the method for generating the register file address to be used during the present microcycle (firmware step). |
| ALU Control (RDDT18 through RDDT22) | (5-bit field) defines the ALU operations as well as specifying the sources of the inputs to the ALU during the present microcycle. |
| Bus Control (RDDT23 through RDDT25) | (3-bit field) defines the input/output operations of the memory (I/O) address/data bus during the present microcycle. |
| Register Modification Field (RDDT26 through RDDT30) | Bit-26 controls the write operation of the register file during the current microcycle. Bit 27 controls the loading of the Q-register during the current microcycle. Bits 28 through 30 define the operation to be performed on one of three registers (F-counter, P-counter, or G-register) during the current microcycle. |
| MMU (RDDT31 through RDDT33) | (3-bit field) defines the operation to be performed by the MMU during the present microcycle. |
| Hardware Interrupt (RDDT34) | (1-bit field) is used to prevent a memory refresh or data transfer request from interrupting the predetermined sequence between the present microcycle and the next microcycle (firmware step) as defined in ROS bits 0 through 12 (ROS address field) of the present firmware word. |
| Special Control Field (RDDT35 through RDDT47) | (13-bit field) is divided into four subfields (A, B, C, and D). The special control field is used to modify as well as supplement certain of the fields mentioned above. It can provide as many as three simultaneous operations or subcommands during the present microcycle. |

## 4.1 ROS ADDRESS FIELD

The ROS address field contains 13 bits (RDDT00 through RDDT12) and is used to generate the address of the next firmware step in a given sequence. The method for generating this address is defined by the first five bits of this field as shown below.

| RDDT Bits<br>0 1 2 3 4 | Operation |
|---|---|
| 1 X X X X | Jump |
| 0 1 X X X | Test Branch |
| 0 0 1 X X | Major Branch |
| 0 0 0 1 X | Increment With Constant |
| 0 0 0 0 1 | Increment Without Constant |
| 0 0 0 0 0 | Return (Pop stack) |

A Push operation can be used in conjunction with any of the first five operations listed above. This has the effect of calling a subroutine, from which it is expected the microprogrammer will wish to return (Pop stack) to the present ROS address plus One (RAR+1).

To facilitate this operation, the LSI-6 chip has built in hardware which allows the Push subcommand, when initiated, to push the contents of the present ROS address register incremented by One on top of the Push/Pop stack (see Figure 3-4). The Return (Pop stack) operation is then used in the last firmware step of the called subroutine to return to the caller's sequence.

One exception to the next address generation being defined by the above six operations is the hardware interrupt. When a hardware interrupt is initiated, the next ROS address will be provided as a hardware vector and the ROS address generated by the ROS address field of the present firmware word will be placed on top of the Push/Pop stack. If a Push subcommand (special control field) is used in conjunction with one of the first five operations shown above, then the microprogrammer must also initiate the hardware interrupt inhibit field (RDDT bit 34) in order to prevent conflicting Push operations. Since the hardware interrupt can be initiated (normally an external event) at any time prior to completion of a given firmware sequence, special consideration must be given to allow its use.

The branching capabilities of the six operations are referred to as page branching and bank branching. A page is defined as 64 consecutive memory locations, and a bank is defined as 1024 memory locations or 16 pages. As shown in Figure 4-2 the branch boundaries for the Test Branch operation is restricted to any location within a page. The branch boundaries for the major branch is restricted to any location within a bank. The remaining four operations are capable of branching or incrementing from one bank to another.

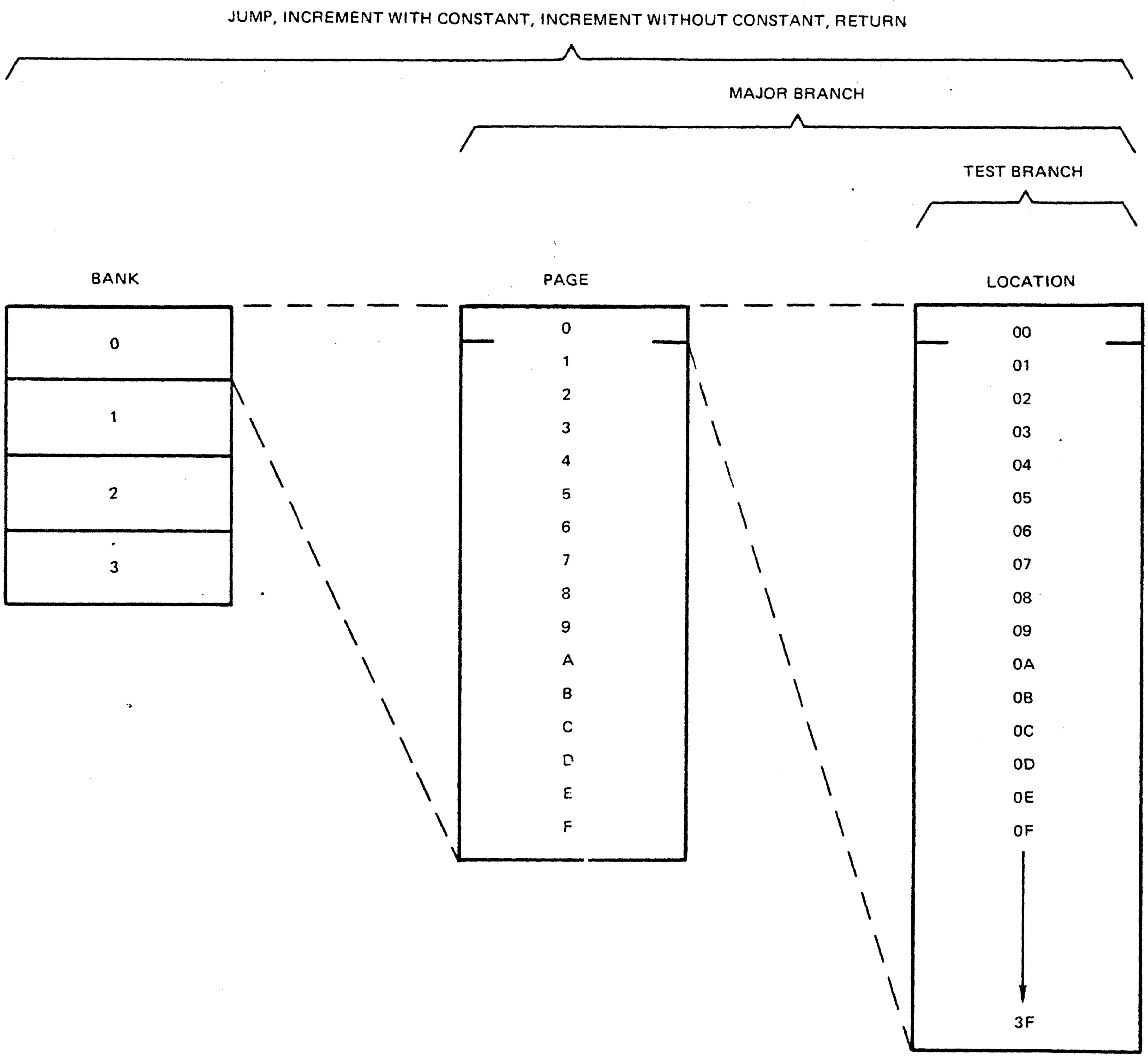
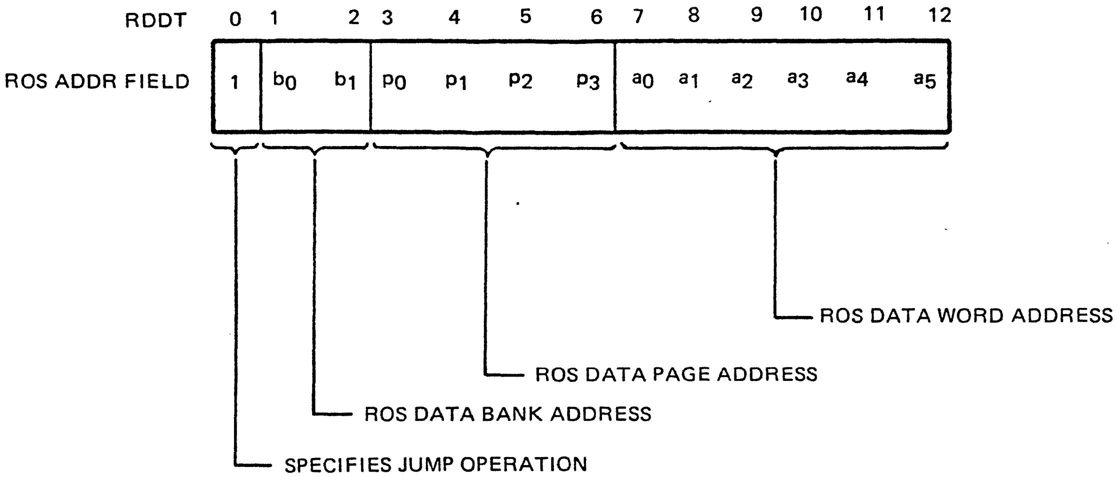


Figure 4-2. Firmware Instruction Branch Boundaries

### 4.1.1 Jump

The Jump operation is the only next address generation method that allows a branch to any of the possible 4096 locations of the ROS. This is accomplished by providing, within the ROS address field, a 12-bit direct address of the next firmware step in sequence.

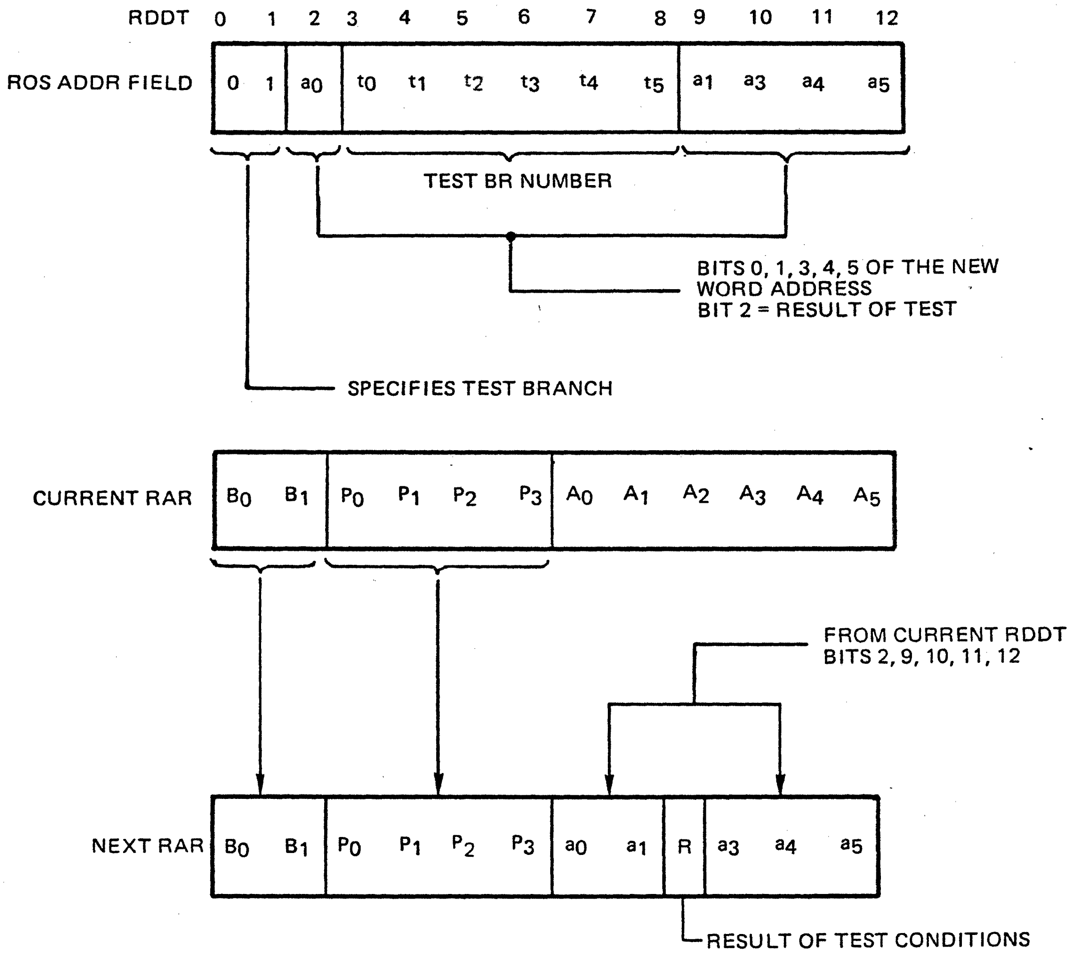


When RDDT bit 0 equals a One, RDDT bits 1 through 12 of the present ROS data word (firmware word) are delivered directly to the ROS address register (bits 0 through 11) as the next address in sequence, assuming of course, no intervening hardware interrupt. Should a hardware interrupt occur, this next address would be pushed on top of the Push/Pop stack and the generated hardware vector will replace it as the contents of the next RAR.

A Push operation (special control field) can be used along with the Jump operation. Hardware interrupt bit RDDT34 will be inhibited when the Push micro-op is used. The current RAR+1 will be pushed on top of Push/Pop stack.

### 4.1.2 Test Branch

The Test Branch operation is a two-way branch using the result of one of 64 test conditions specified as part of the ROS address field ($t_0$-$t_5$). All test branches are restricted to branching within the current page, that is, the next ROS address generated as a result of the test will always be one of two locations (depending on result of test; i.e., true or false) eight locations apart, but within the page (64 locations) currently being addressed by the ROS Address Register (RAR).


RDDT 0 1 2 3 4 5 6 7 8 9 10 11 12
ROS ADDR FIELD
0 1 a0 t0 t1 t2 t3 t4 t5 a1 a3 a4 a5
TEST BR NUMBER
BITS 0, 1, 3, 4, 5 OF THE NEW
WORD ADDRESS
BIT 2 = RESULT OF TEST
SPECIFIES TEST BRANCH
CURRENT RAR
B0 B1 P0 P1 P2 P3 A0 A1 A2 A3 A4 A5
FROM CURRENT RDDT
BITS 2, 9, 10, 11, 12
NEXT RAR
B0 B1 P0 P1 P2 P3 a0 a1 R a3 a4 a5
RESULT OF TEST CONDITIONS

The next RAR shown above contains the next address generated as a result of the test branch as specified in bits 0 through 13 of the current firmware word assuming no intervening hardware interrupt. Should a hardware interrupt occur, this address is placed on the top of the Push/Pop stack and the generated hardware vector replaces it as the contents of the next RAR.

The Push operation (special control field) can be used along with Test Branch operations. Hardware interrupts should be inhibited (RDDT bit 34) in this case. The current RAR+1 will be placed on top of the Push/Pop stack.

The Test Branch instruction is categorized into nine groups for the purpose of discussion only. Each group contains related branch conditions that can be tested with the Test Branch instruction.

The following groups define each test branch category and the test conditions associated with each group. Refer to Tables 4-2 and 4-3 for a combined list of Test Branch instruction sets.

- Group 1--Single Bit F-Register Test Branch

  The single bit F-register test branch group pertains to branch operations that can be performed by testing any one of 16 F-register bits. The format for the single bit F-register test branch is shown below.

| F-Register Bit | Test Condition Code RDDT03 Through RDDT08 | |
|---|---|---|
| FR00 | TBFR00 | (00) |
| FR01 | TBFR01 | (10) |
| FR02 | TBFR02 | (20) |
| FR03 | TBFR03 | (30) |
| FR04 | TBFR04 | (01) |
| FR05 | TBFR05 | (11) |
| FR06 | TBFR06 | (21) |
| FR07 | TBFR07 | (31) |
| FR08 | TBFR08 | (02) |
| FR09 | TBFR09 | (12) |
| FR10 | TBFR10 | (22) |
| FR11 | TBFR11 | (32) |
| FR12 | TBFR12 | (03) |
| FR13 | TBFR13 | (13) |
| FR14 | TBFR14 | (23) |
| FR15 | TBFR15 | (33) |

  The F-register is a 16-bit instruction register that is loaded from the internal bus (BI). All bits are testable, and the low-order four bits (FR11 through FR15) also constitute the low-order four bits of a 5-bit counter called the F-counter (FNCT). These five bits can be loaded, incremented, or decremented independently when specified by the destination field (RDDT28 through RDDT30) of the firmware word. The four low-order bits of the F-counter are also decoded such that a 16-bit mask can be placed onto the internal bus (BI00 through BI15) under firmware control.

- Group 2--Grouped F-Register Test Branch

  The grouped F-register test branch group pertains to branch operations that can be performed by testing specified combinations of F-register bits. The format for the grouped F-register test branch is shown below.

| F-Register Bit | Test Condition Code<br>RDDT03 Through RDDT08 | |
|---|---|---|
| FR12 through FR15 = 0000 | TFBR3Z | (0C) |
| FR01 through FR03 = 000 | TFBFIT3Z | (1C) |
| FR01 through FR03 = 111 | TFBFIT37 | (2C) |
| FR11 through FR15 = 0 (F-Counter) | TFBFTZ | (37) |

- Group 3--Control Flip-Flop Test Branch

  The control flip-flop test branch group consists of four individual control flip-flops that can be set, reset, or loaded under firmware control. The format of the control flip-flop test branch is shown below.

| Control Flip-Flop | Test Condition Code<br>RDDT03 Through RDDT08 | |
|---|---|---|
| Control Flip-Flop 1 | TBCF1 | (0F) |
| Control Flip-Flop 2 | TBCF2 | (1F) |
| Control Flip-Flop 3 | TBCF3 | (2F) |
| Control Flip-Flop 4 | TBCF4 | (3F) |

- Group 4--MMU Flip-Flop Test Branch

  The MMU flip-flop test branch group consists of two Memory Management Unit (MMU) error conditions that are testable by firmware. There are two other MMU flip-flops (MMU flip-flops 1 and 2) that are available but not testable by firmware.

  MMU flip-flops 1 and 3 indicate a nonexistent resource violation occurred or that the segment descriptor is undefined. When set, these flip-flops cause an immediate hardware trap (TV15). MMU flip-flop 3 is testable with the Test Branch instruction.

  MMU flip-flops 2 and 4 indicate an access violation and cause an immediate hardware trap (TV14). MMU flip-flop 4 is testable with the Test Branch instruction.

  The format for the MMU Flip-Flop Test Branch is shown below.

| MMU Flip-Flop | Test Condition Code<br>RDDT03 Through RDDT08 | |
|---|---|---|
| MMU Flip-Flop 3 | 101110 | (2E) |
| MMU Flip-Flop 4 | 111110 | (3E) |

- Group 5--M-, I-, Q-, and S-Register Test Branch

  The M-, I-, Q-, and S-register test branch group consists of selectable hardware register bits that are testable by firmware.

  There are two I-register tests that can be performed by the Test Branch instruction. The first test is with I-register bit 2 which is the carry indicator bit. This bit is set when the logical capacity of a register is exceeded. The second test is with I-register bit 3 which is the bit test indicator. This bit reflects the state of the last bit tested.

  A Q-register test is performed by the test branch which tests Q-register bit 15. The Q-register, a 16-bit register, handles operand shifts and holds secondary operands for the ALU.

  An S-register test is performed by the test branch which tests the most significant bit of the current ring number of the MMU to determine whether the system is in Privilege mode. The current ring number is a duplicate of S-register bits 1 and 2 loaded under control of the LDMURG micro-op.

  The format for the above mentioned register test branches are shown below.

| | Test Condition Code | |
|---|---|---|
| Register Bit | RDDT03 Through RDDT08 | |
| IR2 (C-Bit) | TBIRGC | (2D) |
| IR3 (B-Bit) | TBIRGB | (3D) |
| QR15 | TBQ15 | (27) |
| SR01 | TBSRG1 | (3C) |

- Group 6--Temporary Flip-Flop Test Branch

  The temporary flip-flop test branch group can test 10 temporary flip-flops (TF00 through TF09) and two M-register temporary flip-flops which are loaded during each firmware cycle. Since these conditions exist for just one firmware cycle, it is required that hardware interrupts be inhibited between the microcycle being tested and the microcycle performing the test branch. Each flip-flop represents a hardware condition described in the chart below.

There are two M-register tests that can be performed by the Test Branch instruction. The first test branches on the state of M-register bit 0 at the time of the test branch operation. The second test branches on M-register bit X, where X equals the value of F-register bits 1 through 3 at the time of the test branch operation. If the value of X equals zero, the test branch is inhibited.

| Temporary<br>Flip-Flop | Representation | Test Condition Code<br>RDDT03 Through RDDT08 | |
|---|---|---|---|
| TF0 | BI00 | TBBI00 | (28) |
| TF1 | BI00-BI15 = 0 | TBBI16Z | (09) |
| TF2 | CARRY(16) = 1 | TBCY16 | (19) |
| TF3 | OVERFLOW(16) | TBOV16 | (2B) |
| TF4 | BI0D=BI15 = 0 | TBBI20Z | (0B) |
| TF5 | CARRY(20) = 1 | TBCY20 | (1B) |
| TF6 | OVERFLOW(20) | TBOV20 | (3B) |
| TF7 | BI15 | TBBI15 | (38) |
| TF8 | BI00-BI11 = 0 | TBBI12Z | (08) |
| TF9 | BI0D-BI0A = 0 | TBBIDTA2 | (0A) |
| MRO | | TBMRG0 | (0D) |
| MRX | | TBMRGX | (1D) |

- Group 7--External Conditions Test Branch

The external conditions test branch group consists of conditions testable by firmware. Eight of the ten conditions (MTESTA00 through MTESTA03 and MTESTB00 through MTESTB03) can also be tested with the Major Branch instruction. The format for the external conditions test branch is shown below.

| External<br>Condition | Test Code Condition<br>RDDT03 Through RDDT08 | |
|---|---|---|
| MTESTA00 | TBMTA0 | (04) |
| MTESTA01 | TBMTA1 | (14) |
| MTESTA02 | TBMTA2 | (24) |
| MTESTA03 | TBMTA3 | (34) |
| MTESTB00 | TBMTB0 | (05) |
| MTESTB01 | TBMTB1 | (15) |
| MTESTB02 | TBMTB2 | (25) |
| MTESTB03 | TBMTB3 | (35) |
| STESTA00 | TBSTA0 | (26) |
| STESTB00 | TBSTB0 | (36) |

- Group 8--Software Conditions Test Branch

The software conditions test branch group tests external conditions that are set and reset via software and is testable by firmware. The format for the software conditions test branch is shown below.

| Software Condition | Test Condition Code RDDT03 Through RDDT08 | |
|---|---|---|
| RTC | TBRTC | (0E) |
| WDT | TBWDT | (1E) |
| TICOS | TBMTCK | (07) |
| INT PENDING | TBINTP | (06) |
| PIOVFTP | TBOVTP | (18) |

- Group 9--Options Test Branch

  The options test branch group can test any one of five options that may be present with the system. The format for the options test branch is shown below.

| Option | Test Condition Code RDDT03 Through RDDT08 | |
|---|---|---|
| MMUE | TBMMU | (2A) |
| CIPE | TBCIP | (17) |
| SIPE | TBSIP | (16) |
| LAFE | TBLAF | (1A) |
| MEMVAL | TBOP4 | (3A) |

Table 4-2. Test Branch Test Condition Codes

| Code Number | Mnemonic | Nomenclature | Code Number | Mnemonic | Nomenclature |
|---|---|---|---|---|---|
| 00 | TBFR00 | F-register bit 00 | 20 | TBFR02 | F-register bit 02 |
| 01 | TBFR04 | F-register bit 04 | 21 | TBFR06 | F-register bit 06 |
| 02 | TBFR08 | F-register bit 08 | 22 | TBFR10 | F-register bit 10 |
| 03 | TBFR12 | F-register bit 12 | 23 | TBFR14 | F-register bit 14 |
| 04 | TBMTA0[a] | MTESTA00[a] | 24 | TBMTA2 | MTESTA02[a] |
| 05 | TBMTB0 | MTESTB00[a] | 25 | TBMTB2 | MTESTB02[a] |
| 06 | TBINTP | Interrupt pending | 26 | TBSTA0 | STESTA00[a] |
| 07 | TBMTCK | Missed TICOS | 27 | TBQ15 | Q-register bit 15 |
| 08 | TBBI12Z | BI00-BI11 = 0 (TF8) | 28 | TBBI00 | BI00 (TF0) |
| 09 | TBBI16Z | BI00-BI15 = 0 (TF1) | 29 | - | BI00-BI15 = 0 (Test only) |
| 0A | TBBIDTAZ | BIOD-BIOA = 0 (TF9) | 2A | TBMMU | Option 0-MMUE |
| 0B | TBBI20Z | BIOD-BI15 = 0 (TF4) | 2B | TBOV16 | Overflow 16 (TF3) |
| 0C | TBFR3Z | F-register bits 12-15 = 0 | 2C | TBFIT37 | F-register bits 1-3 = 7 |
| 0D | TBMRG0 | M-register bit 0 | 2D | TBIRGC | I-register bit 2 (C-bit) |
| 0E | TBRTC | Real-time clock on | 2E | TBMMU3 | MMU flip-flop 3 |
| 0F | TBCF1 | Control flip-flop 1 | 2F | TBCF3 | Control flip-flop 3 |
| 10 | TBFR01 | F-register bit 01 | 30 | TBFR03 | F-register bit 03 |
| 11 | TBF05 | F-register bit 05 | 31 | TBFR07 | F-register bit 07 |
| 12 | TBFR09 | F-register bit 09 | 32 | TBFR11 | F-register bit 11 |
| 13 | TBFR13 | F-register bit 13 | 33 | TBFR15 | F-register bit 15 |
| 14 | TBMTA1 | MTESTA01[a] | 34 | TBMTA3 | MTESTA03[a] |
| 15 | TBMTB1 | MTESTB01[a] | 35 | TBMTB3 | MTESTB03[a] |
| 16 | TBSIP | Option 2 - SIPE | 36 | TBSTB0 | STESTB00[a] |
| 17 | TBCIP | Option 1 - CIPE | 37 | TBFTZ | F-counter = 0 (FR11-FR15) |
| 18 | TBOVTP | ALU overflow trap pending | 38 | TBBI15 | BI15 (TF7) |
| 19 | TBCY16 | Carry 16 = 1 (TF2) | 39 | - | Carry 16 = 1 (Test only) |
| 1A | TBLAF | Option 3 - LAFE | 3A | TBOP4 | Option 4 - Memory Valid |
| 1B | TBCY20 | Carry 20 = 1 (TF5) | 3B | TBOV20 | Overflow 20 (TF6) |
| 1C | TBFIT3Z | F-register bits 1-3 = 0 | 3C | TBSRG1 | S-register bit 1 (Privilege Mode) |
| 1D | TBMRGX | M-register bit X | 3D | TBIRGB | I-register bit (B-bit) |
| 1E | TBWDT | Watchdog timer on | 3E | TBMMU4 | MMU flip-flop 4 |
| 1F | TBCF2 | Control flip-flop 2 | 3F | TBCF4 | Control flip-flop 4 |

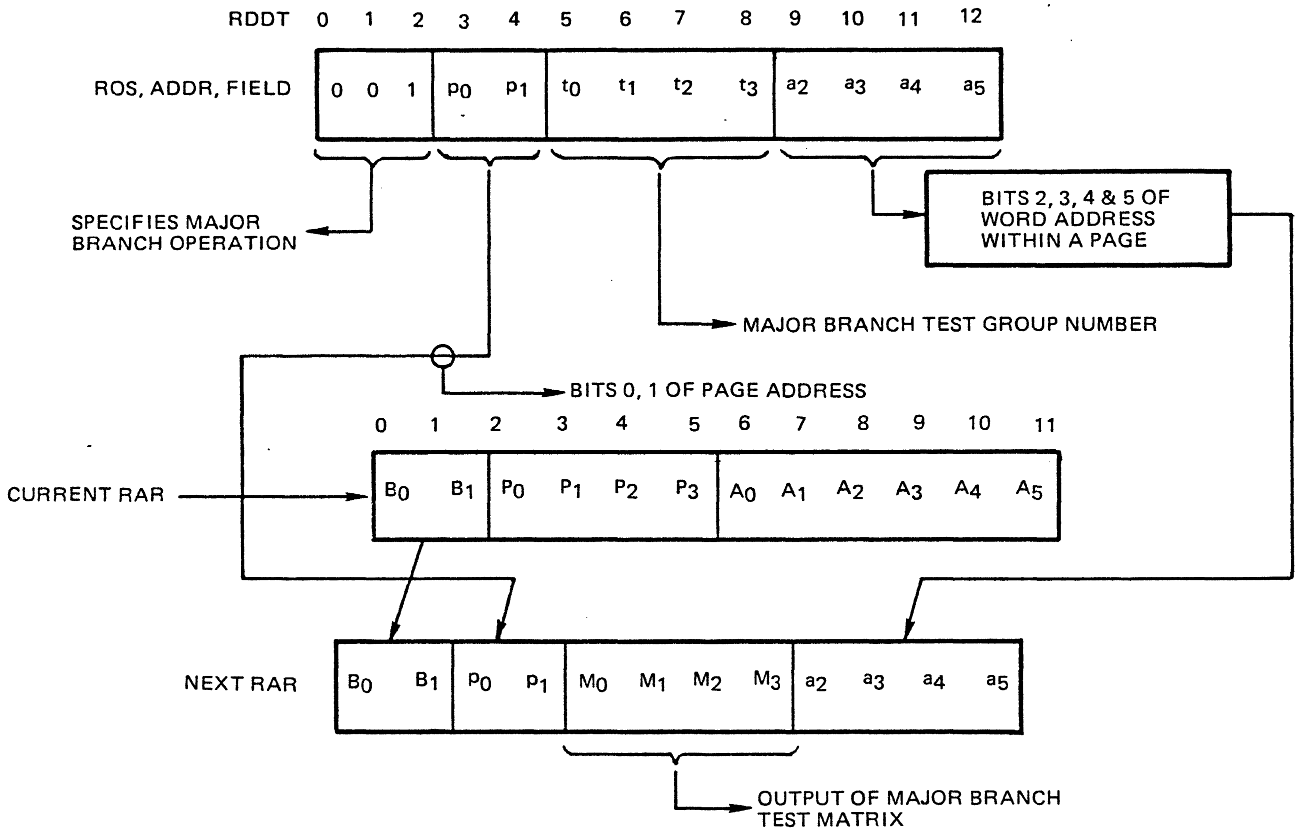
[a] Refer to Table 4-3 for Level 6 equivalent nomenclature.

Table 4-3. LSI-6/Level 6 Nomenclature

| Code Number | Mnemonic | LSI-6 Nomenclature | Level 6 Nomenclature |
|---|---|---|---|
| 04 | TBMTA0 | MTESTA00 | PROCEED-IN |
| 14 | TBMTA1 | MTESTA01 | BUSY-IN |
| 24 | TbMTA2 | MTESTA02 | BYTE-IN |
| 34 | TBMTA3 | MTESTA03 | OBC-IN |
| 05 | TBMTB0 | MTESTB00 | MIBGPO |
| 15 | TBMTB1 | MTESTB01 | MIBGP1 |
| 25 | TMBTB2 | MTESTB02 | MIBGP2 |
| 35 | TBMTB3 | MTESTB03 | MIBGP3 |
| 26 | TBSTA0 | STESTA00 | MPLOCK |
| 36 | TBSTB0 | STESTB00 | TESTBR1 |

### 4.1.3 Major Branch

The major branch is a 16-way branch using the result of 15 test groups specified as part of the ROS Address Field $(t_{\phi}-t_3)$. All major branches are restricted to branching within the current bank (1024 locations). That is, the next ROS address generated as a result of the major branch test will always be one of 16 locations (depending on output of Major Branch matrix) 16 locations apart, but within the bank (1024 locations) currently being addressed by the ROS Address Register (RAR).


RDDT 0 1 2 3 4 5 6 7 8 9 10 11 12
ROS, ADDR, FIELD
0 0 1 $p_0$ $p_1$ $t_0$ $t_1$ $t_2$ $t_3$ $a_2$ $a_3$ $a_4$ $a_5$
SPECIFIES MAJOR BRANCH OPERATION
BITS 2, 3, 4 & 5 OF WORD ADDRESS WITHIN A PAGE
MAJOR BRANCH TEST GROUP NUMBER
BITS 0, 1 OF PAGE ADDRESS
0 1 2 3 4 5 6 7 8 9 10 11
CURRENT RAR
$B_0$ $B_1$ $P_0$ $P_1$ $P_2$ $P_3$ $A_0$ $A_1$ $A_2$ $A_3$ $A_4$ $A_5$
NEXT RAR
$B_0$ $B_1$ $P_0$ $P_1$ $M_0$ $M_1$ $M_2$ $M_3$ $a_2$ $a_3$ $a_4$ $a_5$
OUTPUT OF MAJOR BRANCH TEST MATRIX

The next RAR shown above receives the next address generated as a result of the major branch as specified in bits 0 through 12 of the current firmware word assuming no intervening hardware interrupt. Should a hardware interrupt occur, this newly generated address will be placed on top of the Push/Pop stack and the generated hardware interrupt vector replaces it as the contents of the next RAR.

The Push operation (special control field) can be used along with Major Branch operations. Hardware interrupts should be inhibited (RDDT bit 34) in this case. The current RAR+1 will be placed on top of the Push/Pop stack.

The major branch is divided into two test groupings. Major branch test groups 0 through 7 are for general use where major branch test groups 8 through E are specifically dedicated to the implementation of Honeywell Level 6 functionality. RDDT bits 5 through 8 ($t_\phi$ through $t_3$) equal to a hexadecimal F is not used.

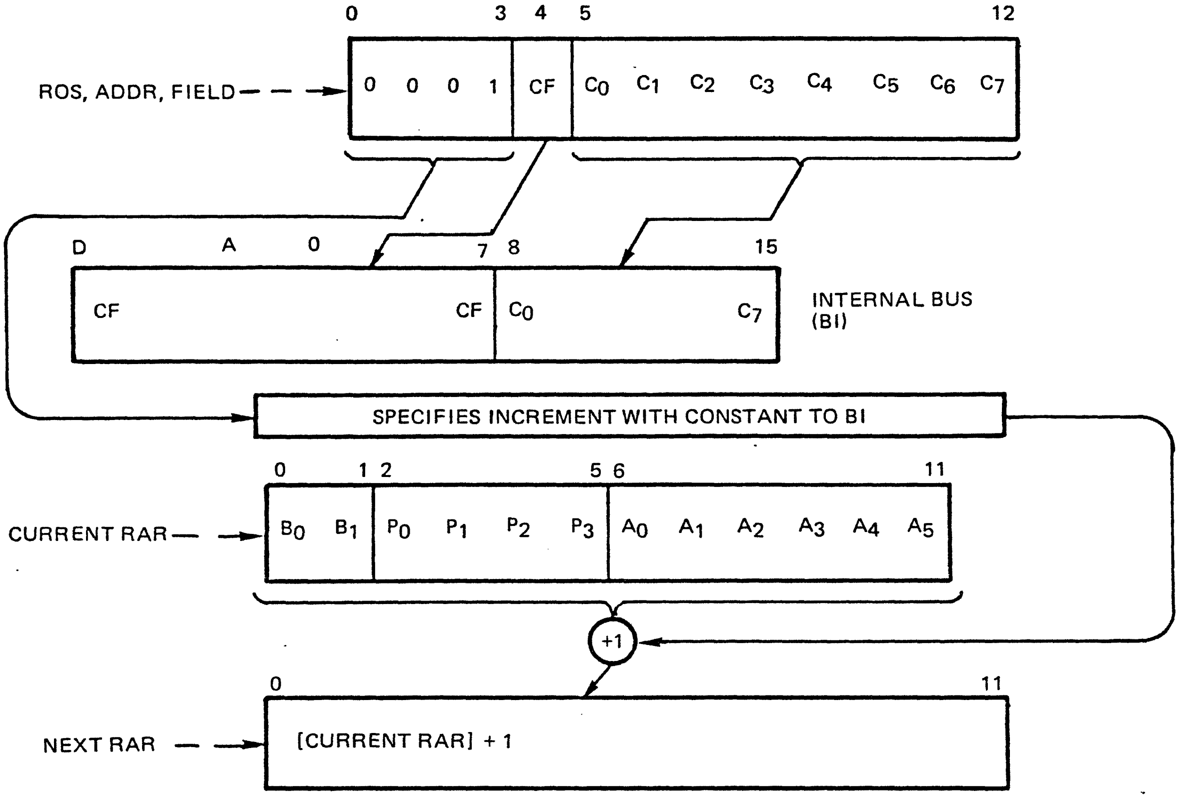
Table 4-4 lists the 16 major branch test condition codes. For detailed information of the major branch test condition codes, refer to Appendix A.

Table 4-4. Major Branch Test Condition Codes

| Code Number | Mnemonic | Description |
|---|---|---|
| 00 | MBFR0 | FR0 (F-register bits 0-3) |
| 01 | MBFR1 | FR1 (F-register bits 4-7) |
| 02 | MBFR2 | FR2 (F-register bits 8-11) |
| 03 | MBFR3 | FR3 (F-register bits 12-15) |
| 04 | MBTESTA | MTESTA00-MTESTA03 |
| 05 | MBTESTB | MTESTB00-MTESTB03 |
| 06 | MBSWIP | Software INT Prinet |
| 07 | MBMULT | 0, 0, Q14, FT=0 |
| 08 | MBFMT | Instruction format |
| 09 | MBOP | D0/S0 instruction group |
| 0A | MBINX | D0/S0 index group |
| 0B | MBGEN | Generic instruction |
| 0C | MBADRS | Address syllable |
| 0D | MBCIP | Data description address syllable |
| 0E | MBBRCH | 0, P Branch, F9TEZ, FR15 |
| 0F | ------ | Not used |

### 4.1.4 Increment With Constant to BI

The increment with constant to BI operation as specified in the ROS address field (bits 0 through 3 of the 13-bit field) of the current firmware word causes the current value of the ROS address register, incremented by One, to be placed into the ROS address register for the next microcycle. In addition to this next address generation, the remaining nine bits (RDDT bits 4 through 12) are used to generate an 8-bit constant plus a filler to the BI during the current microcycle (see format below).

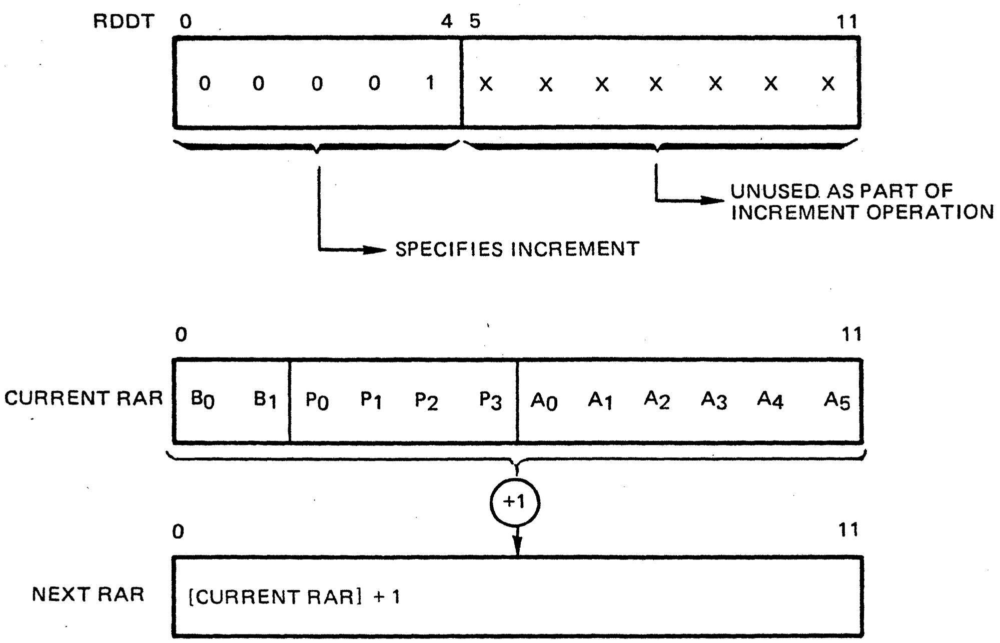


The next RAR shown above receives the current RAR contents plus One as specified by the increment with constant to BI operation in RDDT bits 0 through 3 of the current firmware word assuming no intervening hardware interrupt. Should a hardware interrupt occur, this newly generated next address will be placed on the top of the Push/Pop stack and the generated hardware interrupt vector will replace it as the contents of the next RAR.

The Push operation (special control field) can be used along with Increment operations. Interrupts should be inhibited (RDDT bit 34) when using a Push operation. The current RAR+1 will be placed on top of the Push/Pop stack, in addition to becoming the next value of RAR.

### 4.1.5 Increment

The Increment operation as specified in the ROS address field initiates the same operation as that in subsection 4.1.4 with the exception that no constant is generated to the BI.

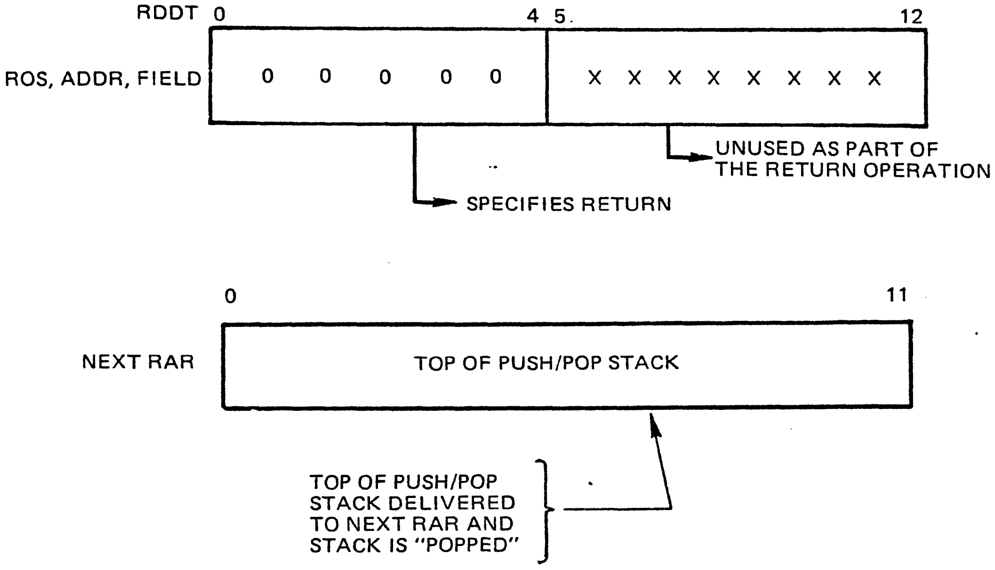

RDDT
0
4
5
11
0 0 0 0 1
X X X X X X X
UNUSED AS PART OF INCREMENT OPERATION
SPECIFIES INCREMENT
0
11
CURRENT RAR
$B_0$ $B_1$ $P_0$ $P_1$ $P_2$ $P_3$ $A_0$ $A_1$ $A_2$ $A_3$ $A_4$ $A_5$
+1
0
11
NEXT RAR
[CURRENT RAR] + 1

The next RAR shown above receives the current RAR contents plus One assuming no intervening hardware interrupt. Should a hardware interrupt occur, the newly generated next address (RAR+1) will be placed on top of the Push/Pop stack and the generated hardware vector replaces it as the contents of the next RAR.

The Push operation (special control field) can be used with the Increment operation. Hardware interrupts should be inhibited (RDDT bit 34) when using the Push operation. When the Push operation is used in conjunction with the Increment operation, the next RAR and the top of the Push/Pop stack receives the same value for the next microcycle ROS address (e.g., the contents of the current RAR incremented by One [RAR+1]).

### 4.1.6 Return (Pop Stack)

The Return or Pop Stack operation causes the contents of the top of the Push/Pop stack to be used as the ROS address for the next microcycle (i.e., next RAR contents).


RDDT 0
4 5
12
ROS, ADDR, FIELD
0 0 0 0 0
X X X X X X X X
UNUSED AS PART OF THE RETURN OPERATION
SPECIFIES RETURN
0
11
NEXT RAR
TOP OF PUSH/POP STACK
TOP OF PUSH/POP STACK DELIVERED TO NEXT RAR AND STACK IS "POPPED"

The next RAR shown above receives the current contents of the top of the Push/Pop stack and the stack is "popped" as specified by the Return (Pop stack) operation in RDDT bits 0 through 4 of the current firmware word, assuming no intervening hardware interrupt. Should a hardware interrupt occur, the Return or Pop operation will effectively be bypassed, or canceled. This is of no consequence, since any programmed interrupt sequence (such as Memory Refresh, Data Transfer, etc.) has a Return as its last steps in its microprogramming sequence.

The Push operation (special control field) must not be used with the Return operation, as the results are unspecified.

## 4.2 REGISTER FILE ADDRESS GENERATION

The LSI-6 has a 32-location register file (see Figure 3-3). Location 00 through 07 and 10 through 17 are 16-bit registers and are primarily used for storage of data, whereas locations 08 through 0F and 18 through 1F are 20-bit registers and are primarily used for storage of main memory addresses.

Fourteen of the first sixteen locations are designated as data and base address registers for use by Honeywell Level 6 software. They are R1 through R7 (locations 01 through 07) and B1 through B7 (locations 09 through 0F). These registers normally will be addressed implicitly using designated bit fields of the F-register (software instruction register) under control of the register file address field (RDDT bits 13 through 17). The facility explicitly addressing these locations exists, using a constant obtained from RDDT bits 9 through 12 of the current firmware word under control of the register file address field.

Note that since RDDT bits 9 through 12 are part of the ROS address field of the current firmware word, it is recommended that this type of explicit addressing be restricted to increment or return (Pop stack) type of next ROS address generation as designated by the ROS address field. It is during these two operations that RDDT bits 9 through 12 are normally not used. However, the hardware implementation does not restrict the microprogrammer from using RDDT bits 9 through 12 as an address constant during any of the next ROS address generations that may be specified in the ROS address field.

Other Level 6 software registers such as the S, RDBR and T (locations 10, 1E, and 1F, respectively), although implicitly designated in certain Level 6 instructions, are explicitly addressed by the register file address field.

All other registers in the register file can be considered as working registers, although some are dedicated to the operation of the Level 6 functionality. These registers are, in general, explicitly addressed by the register file address field of the current firmware word.

Table 4-5 depicts the register file address selection under control of the register file address field (RDDT bits 13 through 17).

Table 4-5. Register File Address Selection (Sheet 1 of 2)

| RDDT (13-17) | RFAR 01234[a] | Location(s) | Selected Register(s) | Firmware Micro-Op[b] |
|---|---|---|---|---|
| 00 | 0yyyy | 00-0F | R0-R7,B0-B7 | DK |
| 01 | 00000 | 00 | R0 | DO |
| 02 | $00f_1 f_2 f_3$ | 00-07 | R0-R7 | DX |
| 03 | $00f_1 f_2 0$ | 00,02,04,06 | R0,R2,R4,R6 | DXM1 |
| 04 | $00f_9 f_{10} f_{11}$ | 00-07 | R0-R7 | DM |
| 05 | $0000f_{10} f_{11}$ | 00-03 | R0-R3 | DMM4 |
| 06 | $00f_{13} f_{14} f_{15}$ | 00-07 | R0-R7 | DN |
| 07 | $000f_{14} f_{15}$ | 00-03 | R0-R3 | DNM4 |
| 08 | 0yyyy | 00-0F | R0-R7,B0-B7 | BK |
| 09 | 01000 | 08 | B0 | BO |

Table 4-5. Register File Address Selection (Sheet 2 of 2)

| RDDT<br>(13-17) | RFAR[a]<br>01234 | Location(s) | Selected<br>Register(s) | Firmware<br>Micro-Op[b] |
|---|---|---|---|---|
| 0A | 01$f_1$ $f_2$ $f_3$ | 08-0F | B0-B7 | BX |
| 0B | 01$f_1$ $f_2$ 0 | 08,0A,0C,0E | B0,B2,B4,B6 | BXM1 |
| 0C | 01$f_9$ $f_{10}$ $f_{11}$ | 08-0F | B0-B7 | BM |
| 0D | 010$f_{10}$ $f_{11}$ | 08-0B | B0-B3 | BMM4 |
| 0E | 01$f_{13}$ $f_{14}$ $f_{15}$ | 08-0F | B0-B7 | BN |
| 0F | 010$f_{14}$ $f_{15}$ | 08-0B | B0-B3 | BNM4 |
| 10 | 10000 | 10 | S | S |
| 11 | 10001 | 11 | DW1 | DW1 |
| 12 | 10010 | 12 | DW2 | DW2 |
| 13 | 10011 | 13 | DW3 | DW3 |
| 14 | 10100 | 14 | DW4 | DW4 |
| 15 | 10101 | 15 | DW5 | DW5 |
| 16 | 10110 | 16 | DW6 | DW6 |
| 17 | 10111 | 17 | L6IO | L6IO |
| 18 | 11000 | 18 | H | H |
| 19 | 11001 | 19 | AW1 | AW1 |
| 1A | 11010 | 1A | AW2 | AW2 |
| 1B | 11011 | 1B | AW3 | AW3 |
| 1C | 11100 | 1C | AW4 | AW4 |
| 1D | 11101 | 1D | AW5 | AW5 |
| 1E | 11110 | 1E | RBDR | RDBR |
| 1F | 11111 | 1F | T | T |

[a] RFAR = Register File Address Register
1 yyyy = RDDT Bits 9 through 12
2 fx = F-Register Bit (x)

[b] See Firmware Directory in Section 5

## 4.3 ALU CONTROL FIELD

The ALU control field (RDDT 18 through 22) not only controls the arithmetic and logical functions of the ALU, but also selects the inputs to its A- and B-ports. Table 4-6 represents the functions of the ALU control field.

Table 4-6. ALU Control (RDDT 18 Through 22)

| RDDT (18-22) | Micro-Op/ Subcommand | RDDT (18-22) | Micro-Op/ Subcommand |
|---|---|---|---|
| 00 | SPANDG | 10 | ZERO |
| 01 | GNOT | 11 | ZMG |
| 02 | G | 12 | SPPG |
| 03 | SPORG | 13 | SPMG |
| 04 | SPANDQNOT | 14 | MINUS1 |
| 05 | QNOT | 15 | ZMQ |
| 06 | SPNOT | 16 | SPM1 |
| 07 | SP | 17 | SPP1 |
| 08 | SPANDQ8 | 18 | SPPQ8 |
| 09 | Q8 | 19 | SPMQ8 |
| 0A | SPXORQ8 | 1A | SPPQ9 |
| 0B | SPORQ8 | 1B | SPMQ9 |
| 0C | SPANDQ | 1C | SPPQ |
| 0D | Q | 1D | SPMQ |
| 0E | SPXORQ | 1E | SPMQM1 |
| 0F | SPORQ | 1F | SPPQP1 |

SP indicates register file (x)
G indicates G-register
Q indicates Q-register
Q8 indicates sign propagation from Q08
Q9 indicates sign propagation from Q09
Z indicates input to A-port is Zero
ZERO indicates output equals Zero
MINUS1 indicates output equals all Ones
AND, OR, NOT, XOR, P and M are all operations

## 4.4 BUS CONTROL FIELD (RDDT 23-25)

The LSI-6 processor bus control field (RDDT bits 23 through 25) is summarized below:

| RDDT<br>(23-25) | Micro-Op | Remarks |
|---|---|---|
| 0 | NOP | Drivers not enabled |
| 1 | DIN | Data into LSI-6 (to internal bus) |
| 2 | DOUT | Data out of LSI-6 (internal bus) |
| 3 | DOUTM | Data out of LSI-6 (MMU) |
| 4 | MMR | Memory address out (read word) |
| 5 | MWB1 | Memory address out (write byte 1) |
| 6 | MWB0 | Memory address out (write byte 0) |
| 7 | MMW | Memory address out (write word) |

## 4.5 REGISTER MODIFICATION FIELD

The register modification field (RDDT26 through RDDT30) is composed of three simultaneous operations.

1. RDDT bit 26 - Register file load or SPW

2. RDDT bit 27 - Q-register load or LDQ

3. RDDT (28-30) - Destination field used to alter contents of FCNT, P- and G-registers.

### 4.5.1 Register File Load Field

The register file load field (RDDT bit 26) is a 1-bit field also known as Scratch Pad Write (SPW). When this bit is Zero, the register file contents cannot be altered (written). When this field is a One, its operation can be modified by micro-ops from the special control field as mentioned in register file description.

### 4.5.2 Q-Register Load Field

The Q-register load field (RDDT bit 27) is a 1-bit field also known as Load Q (LDQ). When this bit is Zero, the contents of the Q-register cannot be altered. When this bit is a One, the modification of the Q-register may be dependent upon the special control field (RDDT bits 35 through 47) which contains all the shift modification micro-ops for the Q-register. If none of the shift micro-ops are activated for the current firmware word, the default will be to load Q with the contents of the internal bus BI(00-15) -> Q(00-15).

### 4.5.3 Destination Field

This 3-bit field (RDDT bits 28 through 30) is used to alter the contents of the FCNT, G- and P-registers as shown below.

| RDDT (28-30) | Micro-Op | Function |
|---|---|---|
| 0 | ---- | No Operation |
| 1 | LDFT | BI (11-15) --> FCNT (11-15) |
| 2 | FTP1 | FCNT + 1 --> FCNT |
| 3 | FTM1 | FCNT - 1 --> FCNT |
| 4 | LDG | BI(0D-0A,00-15) --> G(0D-0A,00-15) |
| 5 | LDP | BI(0D-0A,00-15) --> P(0D-0A,00-15) |
| 6 | PP1 | P + 1 --> P |
| 7 | PM1 | P - 1 --> P |

## 4.6 MMU CONTROL FIELD

This 3-bit field (RDDT bits 31 through 33) is used to determine access rights selection and checking desired within the MMU.

| RDDT (31-33) | Micro-Op | Function |
|---|---|---|
| 0 | ---- | No operation |
| 1 | ERS | Execute ring selection |
| 2 | RRS | Read ring selection |
| 3 | WRS | Write ring selection |
| 4 | IA | Indirect address |
| 5 | FM | Firmware mode (firmware-generated address) |
| 6 | LRA | Limited read access |
| 7 | LWA | Limited write access |

## 4.7 HARDWARE INTERRUPT INHIBIT FIELD

A hardware interrupt forces a branch to a fixed firmware address. This address is determined by a priority net (see Table 4-7) which has various error signals and interrupt requests (memory refresh and data transfer requests) as inputs. Hardware interrupt also causes the next firmware generated address to be pushed onto the Push/Pop stack. If the next address had been generated from the stack via a Return micro-op, "popping" of the stack would be inhibited. Hardware interrupts must be inhibited whenever a Push operation is performed.

When the hardware interrupt inhibit field (RDDT bit 34) is a One, memory refresh and data request hardware interrupts are inhibited (prevented from intervening between current and next firmware-generated address). Hardware error inputs to the Prinet are not under control of RDDT bit 34 and can intervene between any given microcycles. When bit 34 is a Zero, all hardware interrupts are allowed.

Table 4-7. Hardware Interrupt Prinet

| Next RAR Value | Function | Remarks |
|---|---|---|
| 000 | System clear | Unaffected by hardware interrupt inhibit field (RDDT bit 34) |
| 002 | MEMPRES (active low) | |
| | MMU (nonexistent resource) | |
| 003 | MMU (access violation) | |
| 004 | Memory parity error | |
| 005 | Memory refresh request | Under control of hardware interrupt inhibit field (RDDT bit 34) |
| 006 | Data request DTR0 | |
| 007 | Data request DTR1 | |
| 008 | Data request DTR2 | |
| 009 | Data request DTR3 | |
| 00A | Data request DTR4 | |

## 4.8 SPECIAL CONTROL FIELD

The special control field (RDDT bits 35 through 47) is used to modify as well as supplement certain of the other fields in the current firmware word. It provides up to three simultaneous micro-operations during a given microcycle. The special control field is divided into four subfields as shown in the following format.

| 35 37 | 38 41 | 42 44 | 45 47 |
|---|---|---|---|
| A | B | C | D |

The A-field defines the use of the B-, C-, and D-fields as follows:

When A-Field = 0 (general use)

B-Field controls loading of F-, P-, I-, and M-registers, modification to register file loading (SPWs), modifications to Process Bus Data-Out operations, and special sourcing to the BI.

C-field controls the set, reset, and load of control flip-flops 2 and 4 in addition to providing the Push micro-op.

D-field controls the set, reset, and load of control flip-flops 1 and 3 in addition to RTC/WDT control.

When A-field = 1 (MMU control)

B-field controls loading of certain MMU registers plus sourcing of the BI from MMU register file.

C-field (same as when A-field = 0)

D-field (same as when A-field = 0)

When A-field = 2 (indicator register control)

B-field controls set, reset, and loading of individual indicator register bits.

C-field - internal results are inhibited.

D-field - internal results are inhibited.

When A-Field = 3 (shift control)

B-field designates register (S) to be shifted and provide carry (C) and overflow (OV) indicator controls for "Shift by Op-Code" micro-ops.

C-field designates the direction (left or right) and the type of shift (open, closed, or arithmetic) when B-field designates a firmware-controlled shift (not "Shift by Op-Code"). When "Shift by Op-code" micro-op is called in B-field, the C-field should contain a value of Zero (NOP).

D-field designates the filler to be used when B-field designates a firmware-controlled shift (not "Shift by Op-Code").

When "Shift by Op-Code" micro-op is called in the B-field, the D-field should contain a value of Zero (NOP).

When A-Field = 4 (constant to BI)

RDDT bits 38 through 47 will be used to generate a constant to source the BI.

When A-Field = 5, 6, or 7

Bits 38 through 47 (10 bits) are available for use external to the chip.

### 4.8.1 General Use (A-Field = 0)

| B-Field | Micro-Op | Function |
|---|---|---|
| 0 | - | No operation |
| 1 | BIFP | P (0D-15)-->BI (0D-15) |
| 2 | LDM | BI (08-15) -->Mx (00-07); x defined by RFAR bits (2-4) |
| 3 | BIFM | Mx (00-07)-->BI (08-15), Ones BI --> (00-07); x defined by RFAR bits (2-4) |
| 4 | IWDTA | Inhibit write to register file bits (0D-0A) when used with SPW. |
| 5 | SPFBIX | Swap bytes of BI to register file. BI (00-07)-->RF (08-15); BI (08-15)-->RF (00-07). |
| 6 | LDFR | BI (00-11)-->F-register (00-11) |
| 7 | SPXDTA | BI (12-15)-->register file (0D-0A) when used with SPW. Register file (00-15) remain unchanged. |
| 8 | RSTCK | Reset timer (TICOS) interrupt flip-flop. |
| 9 | BIFMK | FR3 (F-register bits 12-15) generates a 16-bit mask to BI (00-15), BI(00)-->(0D-0A). |
| A | LDI | BI(08-15)-->I-register (00-07) |
| B | BIFI | I-Register (00-07)-->BI(08-15), Zeros--BI(0D-07) |
| C | SPFAU | ALU (0D-15)-->Register file (0D-15) used with SPW |

| B-Field | Micro-Op | Function |
|---|---|---|
| D | TWNOUT | BI (08-15)-->DABS (08-15)<br>BI (08-15)-->DABS (00-07), Zeros --> DABS (0D-0A) when used with DOUT |
| E | BIFSP | Register file -->A-port -->BI (To ensure the contents of the register file to BI, the ALU command must contain the scratch-pad term in the ALU micro-op; i.e., SP,SPPG,SPMG etc. Refer to subsection 4.3 for a complete list of ALU micro-ops). ALU (0D-15)--> Register file (0D-15) when used with SPW |
| F | SWPOUT | BI (0D-0A)-->DABS (12-15); Zeros-->DABS 0D-11 when used with DOUT |

When the A-field equals Zero and the B-field is not 5, 7, D, or F, then the C- and D-fields are specified as shown below. When the B-field equals 5, 7, D, or F, the C and D-fields are inhibited internally.

| C-Field | Micro-Op | Function |
|---|---|---|
| 0 | - | No operation |
| 1 | RCF4 | Reset control flip-flop 4 |
| 2 | SCF4 | Set control flip-flop 4 |
| 3 | LCF4 | BI (00) -> CF4 |
| 4 | PUSH* | (RAR) +1 pushed into top of Push/Pop stack (refer to subsection 4.1.6) |
| 5 | RCF2 | Reset control flip-flop 2 |
| 6 | SCF2 | Set control flip-flop 2 |
| 7 | LCF2 | Q-Register (00) -> CF2 |
| D-Field | | |
| 0 | - | No operation |
| 1 | RCF1 | Reset control flip-flop 1 |
| 2 | SCF1 | Set control flip-flop 1 |
| 3 | LCF1 | BI (15) -> CF1 |

*A Push micro-op is only used when the inhibit hardware interrupt bit (RDDT34) is a One.

| D-Field | Micro-Op | Function |
|---|---|---|
| 4 | RTC | Set/reset RTC/WDR dependent on software instruction in F-register |
| 5 | RCF3 | Reset control flip-flop 3 |
| 6 | SCF3 | Set control flip-flop 3 |
| 7 | RESOVT | Reset "Trap on Overflow" flip-flop |

### 4.8.2 MMU Control (A-Field = 1)

| B-Field | Micro-Op | Function |
|---|---|---|
| 0 | CME | Clear MMU error flip-flops |
| 1 | LDMAD | Load MMU register file address register from BI. If BI (0D-0A) = 0, then 0 -> MMUAR(0), and BI (00-03) -> MMUAR (1-4). If BI (0D-0A) ≠ 0, then 1 -> MMUAR (0), and BI (0D-0A) -> MMUAR (1-4). |
| 2 | LDEF | Load effective ring from current ring |
| 3 | LDMURG | Load current and effective rings from BI (01,02). Any time S-register ring is updated this micro-op must be used. |
| 4 | WMBS | Write base word in register file BI (00-15) - MMURF (00-15) |
| 5 | BIFMMUH | MMURF (00-15) - BI (00-15) |
| 6 | WMSZ | Write Size Word in register file BI (00-15) - MMURF (16-31), [MMUAR]+1 - [MMUAR] |
| 7 | BIFMMUL | MMURF (16-31) - BI (00-15) [MMUAR]+1 - [MMUAR]- |
| 8 through F | - | No operation |

The C- and D-fields are identical to those of subsection 4.8.1.

### 4.8.3 Indicator Register Control (A-Field = 2)

| B-Field | Micro-Op | Function |
|---|---|---|
| 0 | - | No operation |
| 1 | RIOV | Reset overflow indicator (I0) |
| 2 | SIOV | Set overflow indicator (I0) |
| 3 | IBFAL00 | BI00 - Bit indicator (I3) |
| 4 | LIOVCA | Load overflow (I0), and load carry (I2) for 16-bit operations only |
| 5 | RICA | Reset carry indicator (I2) |
| 6 | SICA | Set carry indicator (I2) |
| 7 | IBFALZN | If BI (00-15) ≠ 0, set bit indicator; otherwise reset bit indicator (I3) (i.e., a One bit on BI causes Indicator bit to be set) |
| 8 | IGLU16 | Indicator bits G (I5), L (I6), and U (I7) are loaded for 16-bit ALU operations |
| 9 | RIG | Reset "greater than" (G) indicator (reset I5) |
| A | SIG | Set "greater than" indicator (set I5 or G-bit) |
| B | RII | Reset input/output bit - Reset I-bit (I4) |
| C | IGLU20 | Indicator bits G (I5), L (I6), and U (I7) are loaded as the result of a 20-bit ALU operation. |
| D | RIL | Reset "less than" indicator - reset (I6) |
| E | SIL | Set "less than" indicator - set (I6) |
| F | SII | Set input/output indicator - set (I4) |

When A-field = 2; C- and D-field internal results are inhibited.

### 4.8.4 Shift Control (A-Field = 3)

| B-Field | Micro-Op | Function |
|---|---|---|
| 0* | SHOPRI | Reset overflow indicator (I0) for SAL or DAL instructions. Reset carry indicator (I2) for SOL, SOR, SAR, DOL, DOR and DAR instructions. |
| 1* | (None) | Shift Q-register under F-register control |
| 2* | SHOPRF | Shift register file only by op-code - single word shift and set indicators by op-code:<br>Set I0 (OV), if SAL and sign changes; set I2 (C), if SOL and last bit out of ALU00 is equal to One.<br>Set I2 (C), if SOR or SAR and last bit out of ALU15 equals One.<br>NOTE<br>This command is used in conjunction with SPW micro-op active. |
| 3* | SHOP | Shift register file and Q-register concantenated. Double-word shift and set indicators by op-code:<br>Set I0 (OV), if DAL and sign changes, set I2 (C), if DOL and last bit shifted out of ALU00 equals One.<br>Set I2 (C), if DOR or DAR and last bit out of Q15 equals One.<br>NOTE<br>This micro-op command is used with both SPW and LDQ micro-ops active. |
| 4* | (None) | No operation |
| 5 | ** | Q-register shift under firmware control |
| 6 | ** | Register file shift under firmware control |
| 7 | ** | F-register, Q-register shift under firmware control |

*For B-field codes of 0 through 4, the C- and D-fields must be Zero as unspecified results occur.

**Refer to C- and D-field tables below for micro-ops when B-field has a value of 5, 6 or 7.

B-field = 5 (LDQ micro-op must be active)

| C-Field | Micro-Op | Function |
|---|---|---|
| 0 | SQOL | Shift Q-register open left |
| 1 | SQCL | Shift Q-register closed left |
| 2 | SQAL | Shift Q-register arithmetic left |
| 3 | (None) | Duplicate of SQCL |
| 4 | SQOR | Shift Q-register open right |
| 5 | SQCR | Shift Q-register closed right |
| 6 | SQAR | Shift Q-register arithmetic right |
| 7 | (None) | Duplicate of SQCR |

B-field = 6 (SPW micro-op must be active)

| C-Field | Micro-Op | Function |
|---|---|---|
| 0 | SRFOL | Shift register file open left |
| 1 | SRFCL | Shift register file closed left |
| 2 | SRFAL | Shift register file arithmetic left |
| 3 | (None) | Duplicate of SRFCL |
| 4 | SRFOR | Shift register file open right |
| 5 | SRFCR | Shift register file closed right |
| 6 | SRFAR | Shift register file arithmetic right |
| 7 | (None) | Duplicate of SRFCR |

B-field = 7 (LDQ and SPW micro-ops are both active)

| C-Field | Micro-Op | Function |
|---|---|---|
| 0 | SRFQOL | Shift register file, Q-register concatenated open left |
| 1 | SRFQCL | Shift register file, Q-register concatenated closed left |
| 2 | SRFQAL | Shift register file, Q-register concatenated arithmetic left |
| 3 | (None) | Duplicate of SRFQCL |
| 4 | SRFQOR | Shift register file, Q-register concatenated open right |
| 5 | SRFQCR | Shift register file, Q-register concatenated closed right |
| 6 | SRFQAR | Shift register file, Q-register concatenated arithmetic right |
| 7 | (None) | Duplicate of SRFQCR |

When the B-field = 5, 6 or 7, and the C-field = 0, 2 or 4, refer to the following table.

| D-Field | Micro-Op | Function |
|---|---|---|
| 0 | SFO | Shift filler equal Zero |
| 1 | SFCF1 | Shift filler equal CF1 |
| 2 | SFCF4N | Shift filler equal CF4 negation |
| 3 | SFCF4 | Shift filler equal CF4 assertion |
| 4 | SFC16 | Shift filler equal carry out of 16-bit ALU |
| 5 | SFUOA | Shift filler equal ALU (0A) 17th bit |
| 6 | SFU00 | Shift filler equal ALU (00) 16th bit |
| 7 | SFQ00 | Shift filler equal Q-register (00) |

Summary Notes: (Shift Control or A = 3)

1. When using B-field micro-ops SHOPRI, SHOPRF and SHOP (B-field = 0, 2, and 3, respectively), the C- and D-fields should be allowed to default to Zero (no C- or D-field micro-ops should be called).

2. For B-field values of 1 and 4 there are no micro-ops specified, and the C- and D-fields are also unspecified (C- and D-field should be allowed to default to Zeros).

3. For B-field value of 5 (Q-register shifts under firmware control), LDQ must be activated (RDDT 27 = 1) and the C-field micro-op specifies the type and direction of the shift. Filler micro-ops (D-field) should be called where necessary (see Note 6 below).

4. For B-field value of 6 (register file shifts under firmware control), SPW micro-op must be activated (RDDT 26 = 1) and the C-field micro-op specifies the type and direction of the shift. Filler micro-ops (D-field) should be called where necessary (see Note 6 below).

5. For B-field value of 7 (register file concatenated with Q-register to provide a 32-bit shift under firmware control), SPW and LDQ must be activated (RDDT 26, 27 = 1,1), and the C-field micro-op specifies the type and direction of the shift. Filler micro-ops (D-field) should be called where necessary (see Note 6 below).

6. The D-field micro-ops should be used to designate the filler for B-field values of 5, 6, and 7 when C-field values are 0, 2 or, 4.

7. For B-field values of 8 through F, the C- and D-field operations are inhibited internally.

8. All register file shifts are accomplished by shifting the output of the ALU into the register file.

### 4.8.5 Constant to BI (A-Field = 4)

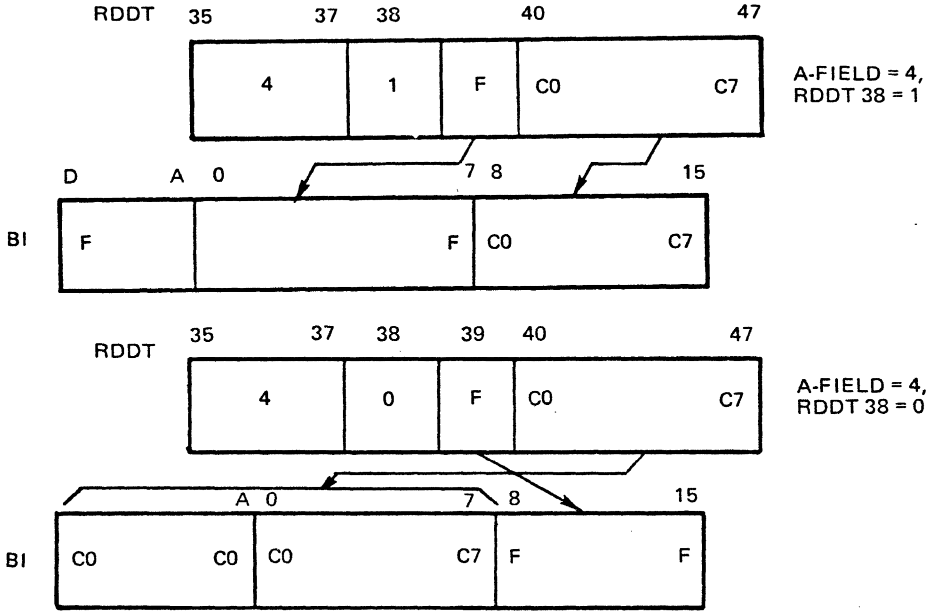
RDDT bit 38 determines if the byte constant will be sourced to BI (00 through 07) or BI (08 through 15).

RDDT bit 39 determines whether the filler byte will be Ones or Zeros.

RDDT bits 40 through 47 contain the byte constant.

NOTE

Whichever bit is selected to source BI (00) will also source BI (0D-0A).



## 4.8.6 User Codes (A-Field = 5, 6 or 7)

When the A-field contains a value of 5, 6 or 7, RDDT bits 38 through 47 become available for control operations external to the chip.

# *Section 5*
# *MICRO-OPS*

This section provides the user with a list of micro-ops that are used with the LSI-6 microprocessor. Tables 5-1 through 5-8 list the micro-ops according to the micro-op function performed, with a brief description of each micro-op. Table 5-9 lists the micro-ops in alphanumeric order, a reference to the micro-op group and subgroup if any.

Table 5-1. Branch Micro-Ops (Sheet 1 of 3)

| Micro-Op Name | | |
|---|---|---|
| Group | Subgroup | Description |
| TB<br>Two-way branching to test branching address based on test results | TBFR00 | F-register bit 00 |
| | TBFR04 | F-register bit 04 |
| | TBFR08 | F-register bit 08 |
| | TBFR12 | F-register bit 12 |
| | TBPROC[a] | Proceed - in |
| | TBMBG0[a] | Major branch in bit 0 |
| | TBINTP | Interrupt pending |
| | TBMTCK | Missed TICKOS |
| | TBBI12Z | BI00-11 = 0 temporary flip-flop |
| | TBBI16Z | BI00-15 = 0 temporary flip-flop |
| | TBBIDTAZ | BI0D-0A = 0 temporary flip-flop |

Table 5-1. Branch Micro-Ops (Sheet 2 of 3)

| Micro-Op Name | | |
|---|---|---|
| Group | Subgroup | Description |
| TB (continued) | TBBI10Z | BIOD-0A, 00-15 = 0 temporary flip-flop |
| | TBFR3Z | F-register (12-15) = 0 |
| | TBMRG0 | M-register (Bit 00) |
| | TBRTC | Real-time clock on |
| | TBCF1 | Control file flip-flop 1 |
| | TBFR01 | F-register bit 01 |
| | TBFR05 | F-register bit 05 |
| | TBFR09 | F-register bit 09 |
| | TBFR13 | F-register bit 13 |
| | TBBUSY[a] | Busy - in |
| | TBMBG1[a] | Major branch in bit 01 |
| | TBSIP | Option 2 SIPE |
| | TBCIP | Option 1 CIPE |
| | TBOVTP | ALU overflow trap pending |
| | TBCY16 | CARRY16 = 1 temporary flip-flop |
| | TBLAF | Option 3 LAFE |
| | TBCY20 | CARRY20 = 1 temporary flip-flop |
| | TBF1T3Z | F-register bits 01-03 = 0 |
| | TBMRGX | M-register bit selected by F-register bits 01-03 |
| | TBWDT | Watchdog timer on |
| | TBCF2 | Control flip-flop 2 |
| | TBFR02 | F-register bit 02 |
| | TBFR06 | F-register bit 06 |
| | TBFR10 | F-register bit 10 |
| | TBFR14 | F-register bit 14 |
| | TBBYTX[a] | BYTEX - in |
| | TBMBG2[a] | Major branch in bit 02 |
| | TBMPLK[a] | MPO lock |
| | TBQ15 | Q-register (bit 15) |
| | TBBI00 | Temporary flip-flop BI00 |
| | TBBIZERO | BI00-15 = 0 same cycle (chip test only) |
| | TBMMU | Option 0 MMUE |
| | TBOV16 | Overflow 16 temporary flip-flop |
| | TBF1T37 | F-register bits 01-03 = 7 |
| | TBIRGC | I-register (C-bit) |
| | TBMMU3 | MMU flip-flop 3<br>Validity - size check error |
| | TBCF3 | Control flip-flop 3 |
| | TBFR03 | F-register bit 03 |
| | TBFR07 | F-register bit 07 |
| | TBFR11 | F-register bit 11 |
| | TBFR15 | F-register bit 15 |
| | TBOBC[a] | OBC - in |

Table 5-1. Branch Micro-Ops (Sheet 3 of 3)

| Micro-Op Name | | |
|---|---|---|
| Group | Subgroup | Description |
| TB<br>(continued) | TBMBG3[a] | Major branch in bit 03 |
| | TBSTB[a] | TSTBR1 (monitor test bit) |
| | TBFTZ | FCNT = 0 |
| | TBBI15 | BI15 temporary flip-flop |
| | TBCARRY | 16-bit ALU = Carry same cycle (chip test only) |
| | TBOP4 | Option 4 memory valid |
| | TBOV20 | OVERFLOW20 temporary flip-flop |
| | TBSRG1 | S-register bit 01 (PRVLG mode) |
| | TBIRGB | I-register (B-bit) |
| | TBMMU4 | MMU flip-flop 4 ring check error |
| | TBCF4 | Control flip-flop 4 |
| MB<br>Sixteen-way branching based on test results | MBFR0 | FR0 (F-register bits 00-03) |
| | MBFR1 | FR1 (F-register bits 04-07) |
| | MBFR2 | FR2 (F-register bits 08-11) |
| | MBFR3 | FR3 (F-register bits 12-15) |
| | MBPROC | Proceed, Busy, Bytex, OBC |
| | MBMGT | MIBGP0 - 3 |
| | MBSWIP | Software interrupt Prinet |
| | MBMULT | 0, 0, Q14, FT = 0 |
| | MBFMT | Instruction format |
| | MBPO | D0/S0 instruction group |
| | MBINX | D0/S0 index group |
| | MBGEN | Generic instruction |
| | MBADRS | Address syllable |
| | MBCIP | Data descriptor address syllable |
| | MBBRCH | 0, P branch, F9TEZ, FR15 |
| POP<br>Go to address at the top of the Push/Pop stack | DKCNST | RDDT bits 09 through 12 to register file address. |
| INCK<br>Go to next address and generate a constant | BICNST | Constant to BI. |
| INC<br>Go to current address plus One | DKCNST | RDDT bits 09 through 12 to register file address. |

[a] Refer to Table 4-3 for LSI-6 equivalent nomenclature.

Table 5-2. Register File Address Micro-Ops

| Micro-Op Name | Description |
|---|---|
| DK | Register File address = 0; RDDT 9,10,11,12 |
| DO | Register File address = 0,0,0,0,0 |
| DX | Register File address = 0,0; FR1,2,3 |
| DXM1 | Register File address = 0,0; FR1,2,0 |
| DM | Register File address = 0,0; FR9,10,11 |
| DMM4 | Register File address = 0,0,0; FR10,11 |
| DN | Register File address = 0,0; FR13,14,15 |
| DNM4 | Register File address = 0,0,0; FR14,15 |
| BK | Register File address = 0; RDDT 9,10,11,12 |
| BO | Register File address = 0,1,0,0,0 |
| BX | Register File address = 0,1; FR1,2,3 |
| BXM1 | Register File address = 0,1; FR1,2,0 |
| BM | Register File address = 0,1; FR9,10,11 |
| BMM4 | Register File address = 0,1,0; FR10,11 |
| BN | Register File address = 0,1; FR13,14,15 |
| BNM4 | Register File address = 0,1,0; FR14,15 |
| S | Status register |
| DW1 | Data working location 1 |
| DW2 | Data working location 2 |
| DW3 | Data working location 3 |
| DW4 | Data working location 4 |
| DW5 | Data working location 5 |
| DW6 | Data working location 6 |
| L6IO | Data working location 7 |
| H | Program control history register |
| AW1 | Address working location 1 |
| AW2 | Address working location 2 |
| AW3 | Address working location 3 |
| AW4 | Address working location 4 |
| AW5 | Address working location 5 |
| RDBR | Remote descriptor base register |
| T | Stack address register |

Table 5-3. ALU Micro-Ops

| Micro-Op Name | Description |
|---|---|
| Logic Functions | |
| SPANDG | SP ANDed with G-register |
| GNOT | Negation of G-register (Ones complement) |
| G | G-register |
| SPORG | SP ORed with G-register |
| SPANDQNOT | SP ANDed with Q-register negation |
| QNOT | Negation of Q-register |
| SPNOT | SP negation |
| SP | SP |
| SPANDQ8 | Ripple QR08 to all higher bits (SP ANDed with Q-register bits 08-15) |
| Q8 | Ripple QR08 to all higher bits (Q-register 08-15) |
| SPXORQ8 | Ripple QR08 to all higher bits (SP XORed with Q-register bits 08-15) |
| SPORQ8 | Ripple QR08 to all higher bits (SP ORed with Q-register bits 08-15) |
| SPANDQ | SP ANDed with Q-register |
| Q | Q-register |
| SPXORQ | SP XORed with Q-register |
| SPROQ | SP ORed with Q-register |
| Arithmetic Functions | |
| ZERO | ALU output equals Zero |
| ZMG | Zero minus G-register |
| SPPG | SP plus G-register |
| SPMG | SP minus G-register |
| MINUS1 | ALU output equals all Ones |
| ZMQ | Zero minus Q-register |
| SPM1 | SP minus One |
| SPP1 | SP plus One |
| SPPQ8 | Ripple QR08 to all higher bits (SP plus Q 08-15) |
| SPMQ8 | Ripple QR08 to all higher bits (SP minus Q 08-15) |
| SPPQ9 | Ripple QR09 to all higher bits (SP plus Q 09-15) |
| SPMQ9 | Ripple QR09 to all higher bits (SP minus Q 09-15) |
| SPPQ | SP plus Q-register |
| SPMQ | SP minus Q-register |
| SPMQM1 | SP minus Q-register minus One |
| SPPQP1 | SP plus Q-register plus One |

Table 5-4. Bus Control Micro-Ops

| Micro-Op Name | Description |
|---|---|
| DIN | Data into LSI-6 |
| DOUT | Data out LSI-6 internal bus |
| DOUTM | Data out from MMU |
| MMR | Memory address, read word |
| MWB1 | Memory address, write byte 1 |
| MWB0 | Memory address, write byte 0 |
| MMW | Memory address, write word |

Table 5-5. Register Modification Micro-Ops

| Micro-Op Name | Description |
|---|---|
| SPW | Control write into register file |
| LDQ | BI to Q-register if no other source |
| LDP | BI to program counter |
| PP1 | P plus One to P-register |
| PM1 | P minus One to P-register |
| LDG | BI to G-register |
| LDFT | BI11-15 to FCNT11-15 |
| FTP1 | FCNT plus One to FCNT |
| FTM1 | FCNT minus One to FCNT |

Table 5-6. MMU Micro-Ops

| Micro-Op Name | Description |
|---|---|
| NOP | No operation |
| RRS | Read ring selection |
| WRS | Write ring selection |
| IA | Indirect address |
| FM | Firmware mode |
| LRA | Limited read access |
| LWA | Limited write access |
| ERS | Execute ring selection |

Table 5-7. Hardware Interrupt Micro-Op

| Micro-Op Name | Description |
|---|---|
| DHI | Disable Hardware Interrupt |

Table 5-8. Special Control Micro-Ops (Sheet 1 of 3)

| Micro-Op Name | | |
|---|---|---|
| Group | Subgroup | Description |
| General | BIFP | P-register to BI |
| | LDM | BI08-15 to M-register addressed by SPAR |
| RDDT 35-37 = 000 | BIFM | M-register to BI08-15, one to others addressed by SPAR |
| | IWDTA | INH SPW into bits 0D-0A |
| | SPFBIX | SWAP, BI to register file |
| | LDFR | BI to FR00-FR11 |
| | SPXDTA | BI12-15 to SPM 0D-0A |
| | RSTCK | Reset TICKOS |
| | BIFMK | FR3 generates a mask to BI |
| | LDI | BI08-15 to I-register |
| | BIFI | I-register to BI08-15, 0 to other |
| | SPFAU | SPM input from ALU |
| | TWNOUT | BI08-15 to PB00-07,08-15 |
| | BIFSP | SPM to BI, ALU to SPM |
| | SWPOUT | 0->PB(0D-0A,0-11) BI(0D-0A) to PB(12-15) |
| | RCF4 | Reset CF4 |
| | SCF4 | Set CF4 |
| | LCF4 | Load BI00 into CF4 |
| | PUSH | Push RAR plus One to stack |
| | RCF2 | Reset CF2 |
| | SCF2 | Set CF2 |
| | LCF2 | Load QR00 into CF2 |
| | RCF1 | Reset CF1 |
| | SCF1 | Set CF1 |
| | LCF1 | Load BI15 into CF1 |
| | RTC | Set, reset real-time clock and watch-dog timer depending on OPCD |
| | RCF3 | Reset CF3 |
| | SCF3 | Set CF3 |
| | RESOVT | Reset TRAP ON OVERFLOW |

Table 5-8. Special Control Micro-Ops (Sheet 2 of 3)

| Micro-Op Name | | |
|---|---|---|
| Group | Subgroup | Description |
| MMU | CME | Clear memory errors |
| | LDMAD | Load MMU address from BI |
| RDDT 35-37 = 001 | LDEF | MMU current ring to effective ring |
| | LDMURG | Load MMU backup ring and effective ring and SRG1 flip-flop |
| | WMBS | Write MMU base |
| | BIFMMUH | MMU00-15 BI00-15 |
| | WMSZ | Write MMU size 00 |
| | BIFMMUL | MMU16-31 BI00-15 |
| | RCF4 | Reset CF4 |
| | SCF4 | Set CF4 |
| | LCF4 | Load BI00 into CF4 |
| | PUSH | Push RAR plus One to stack |
| | RCF2 | Reset CF2 |
| | SCF2 | Set CF2 |
| | LCF2 | Load QR00 into CF2 |
| | RCF1 | Reset CF1 |
| | SCF1 | Set CF1 |
| | LCF1 | Load BI15 into CF1 |
| | RTC | Set, reset real-time clock and watchdog timer depending on OPCD |
| | RCF3 | Reset CF3 |
| | SCF3 | Set CF3 |
| | RESOVT | Reset TRAP ON OVERFLOW |
| Indicator | RIOV | Reset OV bit |
| Register | SIOV | Set OV bit |
| | IBFAL00 | Load B-bit from ALU bit $\overline{00}$ |
| RDDT 35-37 = 010 | LIOVCA | Load OV, C-bits |
| | RICA | Reset C-bit |
| | SICA | Set C-bit |
| | IBFALZN | Load B-bit from ALU = 0 |
| | IGLU16 | G-, L-, U-bits from 16-bit ALU |
| | RIG | Reset G-bit |
| | SIG | Set G-bit |
| | RII | Reset I-bit |
| | IGLU20 | G-, L-, U-bits from 20-bit ALU |
| | RIL | Reset L-bit |
| | SIL | Set L-bit |
| | SII | Set I-bit |

Table 5-8. Special Control Micro-Ops (Sheet 3 of 3)

| Micro-Op Name | | Description |
|---|---|---|
| Group | Subgroup | |
| Shift | SHOPRI | Reset IOV,ICA depending on OPCE |
| | SHOPRF | Shift register file only |
| RDDT 35-37 = 011 | SHOP | Shift according to the op-code in F-register, and Q-register will be changed, IOV |
| | SQOL | Shift Q-register open left |
| | SQCL | Shift Q-register close left |
| | SQAL | Shift Q-register arithmetic left |
| | SQOR | Shift Q-register open right |
| | SQCR | Shift Q-register close right |
| | SQAR | Shift Q-register arithmetic right |
| | SRFOL | Shift register file open left |
| | SRFCL | Shift register file close left |
| | SRFAL | Shift register file arithmetic left |
| | SRFOR | Shift register file open right |
| | SRFCR | Shift register file close right |
| | SRFAR | Shift register file arithmetic right |
| | SRFQOL | Shift register file Q-register open left |
| | SRFQCL | Shift register file Q-register closed left |
| | SRFQAL | Shift register file Q-register arithmetic left |
| | SRFQOR | Shift register file Q-register open right |
| | SRFQCR | Shift register file Q-register closed right |
| | SRFQAR | Shift register file Q-register arithmetic right |
| Constant | SF0 | Shift filler equals Zero |
| Generation | SFCF1 | Shift filler equals CF1 |
| | SFCF4N | Shift filler equals CF4 negative |
| RDDT 35-37 = 100 | SFCF4 | Shift filler equals CF4 |
| | SFC16 | Shift filler equals carry out of 16-bit ALU |
| | SFBI0A | Shift filler equals ALU0A |
| | SFU00 | Shift filler equals ALU00 |
| | SFQ00 | Shift filler equals QR00 |
| External Use | | RDDT40 to BIOD-0A |
| | | RDDT40-47 to BI00-07 |
| RDDT 35-37 = 101 | RDH | RDDT39 to BI08-15 |
| RDDT 35-37 = 110 | RDL | RDDT39 to BIOD-0A,00-07 |
| RDDT 35-37 = 111 | RDVL | RDDT40-47 to BI08-15 |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 1 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| AW1 | Register file address |
| AW2 | Register file address |
| AW3 | Register file address |
| AW4 | Register file address |
| AW5 | Register file address |
| BICNST | Branch/INCK, Pop |
| BIFI | Special control/general |
| BIFM | Special control/general |
| BIFMK | Special control/general |
| BIFMMUH | Special control/MMU |
| BIFMMUL | Special control/MMU |
| BIFP | Special control/general |
| BIFSP | Special control/general |
| BK | Register file address |
| BM | Register file address |
| BMM4 | Register file address |
| BN | Register file address |
| BNM4 | Register file address |
| BX | Register file address |
| BXM1 | Register file address |
| BO | Register file address |
| CME | Special control/MMU |
| DHI | Hardware interrupt |
| DIN | Bus control |
| DK | Register file address |
| DKCNST | Branch/INC,INCK,Pop |
| DM | Register file address |
| DMM4 | Register file address |
| DN | Register file address |
| DNM4 | Register file address |
| DOUT | Bus control |
| DOUTM | Bus control |
| DW1 | Register file address |
| DW2 | Register file address |
| DW3 | Register file address |
| DW4 | Register file address |
| DW5 | Register file address |
| DW6 | Register file address |
| DX | Register file address |
| DXM1 | Register file address |
| DO | Register file address |
| ERS | MMU |
| FM | MMU |
| FTM1 | Register modification |
| FTP1 | Register modification |
| G | ALU |
| GNOT | ALU |
| H | Register file address |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 2 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| IA | MMU |
| IBFALZN | Special Control/I-register |
| IBFAL00 | Special Control/I-register |
| IGLU16 | Special Control/I-register |
| IGLU20 | Special Control/I-register |
| INC | Special branch/INC |
| INCK | Special branch/INCK |
| JMP | Special branch/Jump |
| LCF1 | Special control/general,MMU |
| LCF2 | Special control/general,MMU |
| LCF4 | Special control/general,MMU |
| LDEF | Special control/MMU |
| LDFR | Special control/general |
| LDFT | Register modification |
| LDG | Register modification |
| LDI | Special control/general |
| LDM | Special control/general |
| LDMAD | Special control/MMU |
| LDMURG | Special control/MMU |
| LDP | Register modification |
| LDQ | Register modification |
| LIOVCA | Special control/I-register |
| LRA | MMU |
| LWA | MMU |
| L6IO | Register file address |
| MB | Branch/major branch |
| MBADRS | Branch/major branch |
| MBBRCH | Branch/major branch |
| MBCIP | Branch/major branch |
| MBFMT | Branch/major branch |
| MBFR0 | Branch/major branch |
| MBFR1 | Branch/major branch |
| MBFR2 | Branch/major branch |
| MBFR3 | Branch/major branch |
| MBGEN | Branch/major branch |
| MBINX | Branch/major branch |
| MBMGT | Branch/major branch |
| MBMULT | Branch/major branch |
| MBOP | Branch/major branch |
| MBSWIP | Branch/major branch |
| MINUS1 | ALU |
| MMR | Bus control |
| MMW | Bus control |
| MWB0 | Bus control |
| MWB1 | Bus control |
| NOP | MMU |
| PM1 | Register modification |
| POP | Branch/Pop |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 3 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| PP1 | Register modification |
| PUSH | Special control/general,MMU |
| Q | ALU |
| GNOT | ALU |
| Q8 | ALU |
| RCF1 | Special control/general,MMU |
| RCF2 | Special control/general,MMU |
| RCF3 | Special control/general,MMU |
| RCF4 | Special control/general,MMU |
| RDBF | Register file address |
| RDH | Special control/external use |
| RDL | Special control/external use |
| RDVL | Special control/external use |
| RESOVT | Special control/external,MMU |
| RICA | Special control/I-register |
| RIG | Special control/I-register |
| RII | Special control/I-register |
| RIL | Speical control/I-register |
| RIOV | Special control/I-register |
| RRS | MMU |
| RSTCK | Special control/general |
| RTC | Special control/general,MMU |
| S | Register file address |
| SCF1 | Special control/general,MMU |
| SCF2 | Special control/general,MMU |
| SCF3 | Special control/general,MMU |
| SCF4 | Special control/general,MMU |
| SFBIOA | Special control/constant general |
| SFCF1 | Special control/constant general |
| SFCF4 | Special control/constant general |
| SFCF4N | Special control/constant general |
| SFC16 | Special control/constant general |
| SFQ00 | Special control/constant general |
| SFU00 | Special control/constant general |
| SF0 | Special control/constant general |
| SHOP | Special control/shift |
| SHOPRF | Special control/shift |
| SHOPRI | Special control/shift |
| SICA | Special control/I-register |
| SIG | Special control/I-register |
| SII | Special control/I-register |
| SIL | Specail control/I-register |
| SIOV | Special control/I-register |
| SP | ALU |
| SPANDG | ALU |
| SPANDQ | ALU |
| SPANDQNOT | ALU |
| SPANDQ8 | ALU |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 4 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| SPFAU | Special control/general |
| SPFB1X | Special control/general |
| SPIWDTA | Special control/general |
| SPMG | ALU |
| SPMQ | ALU |
| SPMQM1 | ALU |
| SPMQ8 | ALU |
| SPMQ9 | ALU |
| SPM1 | ALU |
| SPNOT | ALU |
| SPORG | ALU |
| SPORQ | ALU |
| SPORQ8 | ALU |
| SPPG | ALU |
| SPPQ | ALU |
| SPPQP1 | ALU |
| SPPQ8 | ALU |
| SPPQ9 | ALU |
| SPP1 | ALU |
| SPW | Register modification |
| SPXDTA | Special control/general |
| SPXORQ | ALU |
| SPXORQ8 | ALU |
| SQAL | Special control/shift |
| SQAR | Special control/shift |
| SQCL | Special control/shift |
| SQCR | Special control/shift |
| SQOL | Special control/shift |
| SQOR | Special control/shift |
| SRFAL | Special control/shift |
| SRFAR | Special control/shift |
| SRFCL | Special control/shift |
| SRFCR | Special control/shift |
| SRFOL | Special control/shift |
| SRFOR | Special control/shift |
| SRFQAL | Special control/shift |
| SRFQAR | Special control/shift |
| SRFQCL | Special control/shift |
| SRFQCR | Special control/shift |
| SRFQOL | Special control/shift |
| SRFQOR | Special control/shift |
| SWPOUT | Special control/general |
| T | Register file address |
| TB | Branch/test branch |
| TBBIDTAZ | Branch/test branch |
| TBBIZERO | Branch/test branch |
| TBBI00 | Branch/test branch |
| TBBI12Z | Branch/test branch |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 5 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| TBBI15 | Branch/test branch |
| TBBI16Z | Branch/test branch |
| TBBI20Z | Branch/test branch |
| TBBUSY | Branch/test branch |
| TBBYTX | Branch/test branch |
| TBCARRY | Branch/test branch |
| TBCF1 | Branch/test branch |
| TBCF2 | Branch/test branch |
| TBCF3 | Branch/test branch |
| TBCF4 | Branch/test branch |
| TBCIP | Branch/test branch |
| TBCY16 | Branch/test branch |
| TBCY20 | Branch/test branch |
| TBFR00 | Branch/test branch |
| TBFR01 | Branch/test branch |
| TBFR02 | Branch/test branch |
| TBFR03 | Branch/test branch |
| TBFR04 | Branch/test branch |
| TBFR05 | Branch/test branch |
| TBFR06 | Branch/test branch |
| TBFR07 | Branch/test branch |
| TBFR08 | Branch/test branch |
| TBFR09 | Branch/test branch |
| TBFR10 | Branch/test branch |
| TBFR11 | Branch/test branch |
| TBFR12 | Branch/test branch |
| TBFR13 | Branch/test branch |
| TBFR14 | Branch/test branch |
| TBFR15 | Branch/test branch |
| TBFR3Z | Branch/test branch |
| TBFTZ | Branch/test branch |
| TBF1T3Z | Branch/test branch |
| TBF1T37 | Branch/test branch |
| TBINTP | Branch/test branch |
| TBIRGB | Branch/test branch |
| TBIRGC | Branch/test branch |
| TBLAF | Branch/test branch |
| TBMBG0 | Branch/test branch |
| TBMBG1 | Branch/test branch |
| TBMBG2 | Branch/test branch |
| TBMBG3 | Branch/test branch |
| TBMMU | Branch/test branch |
| TBMMU3 | Branch/test branch |
| TBMMU4 | Branch/test branch |
| TBMPLK | Branch/test branch |
| TBMRGX | Branch/test branch |
| TBMRG0 | Branch/test branch |
| TBMTB | Branch/test branch |

Table 5-9. Micro-Ops Listed Alphanumerically (Sheet 6 of 6)

| Micro-Op | Micro-Op Reference |
|---|---|
| TBMTCK | Branch/test branch |
| TBOBC | Branch/test branch |
| TBOP4 | Branch/test branch |
| TBOVTP | Branch/test branch |
| TBOV16 | Branch/test branch |
| TBOV20 | Branch/test branch |
| TBPROC | Branch/test branch |
| TBQ15 | Branch/test branch |
| TBRTC | Branch/test branch |
| TBSIP | Branch/test branch |
| TBSRG1 | Branch/test branch |
| TBWDT | Branch/test branch |
| TWNOUT | Special control/general |
| WMBS | Special control/MMU |
| WMSZ | Special control, MMU |
| WRS | MMU |
| ZERO | ALU |
| ZMG | ALU |
| ZMQ | ALU |

# *Appendix A*
# *MAJOR BRANCH CODING*

This appendix provides detailed information for the Major Branch instruction. The Major Branch instruction is a six-way branch which has 15 test groups, 0 through E. The first eight groups are for general use and the remaining seven groups are Honeywell Level 6 specific.

## A.1 MAJOR BRANCH TEST GROUPS 0 THROUGH 7

Major branch test groups 0 through 7 are enabled with RDDT bits 05 through 08. The eight test groups are further defined in Table A-1.

Table A-1. Major Branch Test Groups 0 through 7

| Test Group (RDDT05-08) | Test Condition Output | | | | Function |
|---|---|---|---|---|---|
| | M0 | M1 | M2 | M3 | |
| 0 | FR00 | FR01 | FR02 | FR03 | Not applicable |
| 1 | FR04 | FR05 | FR06 | FR07 | |
| 2 | FR08 | FR09 | FR10 | FR11 | |
| 3 | FR12 | FR13 | FR14 | FR15 | |
| 4 | MTESTA0 | MTESTA1 | MTESTA2 | MTESTA3 | |
| 5 | MTESTB0 | MTESTB1 | MTESTB2 | MTESTB3 | |
| | M0 through M3 = 0 | | | | Firmware trap (previous instruction had software violations) |
| | M0 through M3 = 1 | | | | POWON (Trailing edge detected for power fail) |
| 6 | M0 through M3 = 2 | | | | External Interrupt INTR0 |
| | M0 through M3 = 3 | | | | External Interrupt INTR1 |
| Software | M0 through M3 = 4 | | | | External Interrupt INTR2 |
| Interrupt | M0 through M3 = 5 | | | | TICOS (leading and trailing edge detected) |
| Prinet | M0 through M3 = 6 | | | | Not used |
| | M0 through M3 = 7 | | | | No interrupt pending (resume instruction sequence) |
| 7 | 0 | 0 | Q14 | FCNT=0 | Not applicable |

## A.2 MAJOR BRANCH TEST GROUPS 8 THROUGH 15

The major branch test groups 8 through 15 are defined in subsections A.2.1 through A.2.7.

### A.2.1 Format Group (8)

The format group is enabled when RDDT bits 05 through 08 equal a hexadecimal 8. The format groups are further defined in Table A-2.

Table A-2. Format Groups (Sheet 1 of 2)

| Input | | | | | Output | |
|---|---|---|---|---|---|---|
| F-Register (0) | F-Register (1-3) | F-Register (4-7) | F-Register (8) | F-Register (10-11) | M-Register (0-3) | Instruction Type |
| 0 | 0 | 0 | 0 | 0-3 | A | Generic |
| 0 | 0 | 0 | 1 | 0-3 | 0 | Illegal |
| 0 | 0 | 1 | X | 0-3 | 0 | Illegal |
| 0 | 0 | 2-B | X | 0-3 | 1 | BI |
| 0 | 0 | C-E | X | 0-3 | 0 | Illegal |
| 0 | 0 | F | X | 0-3 | 1 | BI |
| 0 | 1-7 | 0 | 0 | 0-2 | 4 | SWS (Shift) |
| 0 | 1-7 | 0 | 0 | 3 | 6 | DWS (Shift) |
| 0 | 1-7 | 0 | 1 | 0-3 | 6 | DWS (Shift) |
| 0 | 1-7 | 1-2 | X | 0-3 | 0 | Illegal |
| 0 | 1-7 | 3 | X | 0-3 | B | CIP Branch |
| 0 | 1-7 | 4-6 | X | 0-3 | 7 | SIP Branch |
| 0 | 1-7 | 7 | 0 | 0-3 | 2 | BDC |
| 0 | 1-7 | 7 | 1 | 0-3 | 3 | BINC |
| 0 | 1-7 | 8-B | X | 0-3 | 1 | BR |
| 0 | 1-7 | C-F | X | 0-3 | 5 | SI |
| 1 | 0 | 0 | 0 | 0-3 | 8 | IO |
| 1 | 0 | 0 | 1 | 0-3 | 0 | Illegal |
| 1 | 0 | 1 | X | 0-3 | 8 | IO |
| 1 | 0 | 2 | X | 0-3 | D | Read SO |
| 1 | 0 | 3 | 0 | 0-3 | 0 | Illegal |
| 1 | 0 | 3 | 1 | 0-3 | C | Write SO |
| 1 | 0 | 4 | X | 0-3 | E | SO |
| 1 | 0 | 5 | X | 0-3 | 0 | Illegal |
| 1 | 0 | 6 | 0 | 0-3 | D | Read SO |
| 1 | 0 | 6 | 1 | 0-3 | 0 | Illegal |
| 1 | 0 | 7 | X | 0-3 | C | Write SO |
| 1 | 0 | 8-A | X | 0-3 | D | Read SO |
| 1 | 0 | B | 0 | 0-3 | D | Read SO |
| 1 | 0 | B | 1 | 0-3 | C | Write SO |
| 1 | 0 | C | 0 | 0-3 | C | Write SO |
| 1 | 0 | C | 1 | 0-3 | E | SO |
| 1 | 0 | D | 0 | 0-3 | E | SO |
| 1 | 0 | D | 1 | 0-3 | 9 | B-register group |
| 1 | 0 | E | X | 0-3 | D | Read SO |
| 1 | 0 | F | 0 | 0-3 | C | Write SO |
| 1 | 0 | F | 1 | 0-3 | D | Read SO |
| 1 | 1-7 | 0 | X | 0-3 | D | Read DO |
| 1 | 1-7 | 1 | 0 | 0-3 | 0 | Illegal |
| 1 | 1-7 | 1 | 1 | 0-3 | D | Read DO |

Table A-2. Format Groups (Sheet 2 of 2)

| Input | | | | | Output | |
|---|---|---|---|---|---|---|
| F-Register (0) | F-Register (1-3) | F-Register (4-7) | F-Register (8) | F-Register (10-11) | M-Register (0-3) | Instruction Type |
| 1 | 1-7 | 2 | X | 0-3 | D | Read DO |
| 1 | 1-7 | 3 | 0 | 0-3 | D | Read DO |
| 1 | 1-7 | 3 | 1 | 0-3 | C | Write DO |
| 1 | 1-7 | 4-6 | X | 0-3 | D | Read DO |
| 1 | 1-7 | 7 | X | 0-3 | C | Write DO |
| 1 | 1-7 | 8 | 0 | 0-3 | D | Read DO |
| 1 | 1-7 | 8 | 1 | 0-3 | E | SIP other than branch |
| 1 | 1-7 | 9 | 0 | 0-3 | D | Read DO |
| 1 | 1-7 | 9 | 1 | 0-3 | E | SIP other than branch |
| 1 | 1-7 | A | X | 0-3 | D | Read DO |
| 1 | 1-7 | B | 0 | 0-3 | D | Read DO |
| 1 | 1-7 | B | 1 | 0-3 | C | Write DO |
| 1 | 1-7 | C | 0 | 0-3 | E | SIP other than branch |
| 1 | 1-7 | C | 1 | 0-3 | 9 | B-register group |
| 1 | 1-7 | D | 0 | 0-3 | E | SIP other than branch |
| 1 | 1-7 | D | 1 | 0-3 | 9 | B-register group |
| 1 | 1-7 | E | 0 | 0-3 | D | Read DO |
| 1 | 1-7 | E | 1 | 0-3 | 9 | B-register group |
| 1 | 1-7 | F | X | 0-3 | C | Write DO |

### A.2.2 Op-Code Group (9)

The op-code group is enabled when RDCT bits 05 through 08 equal a hexadecimal 9. The op-code group is further defined in Table A-3.

Table A-3. Op-Code Group (Sheet 1 of 2)

| Input | | | Output | Instruction | |
|---|---|---|---|---|---|
| F-Register (1-3) | F-Register (4-7) | F-Register (8) | M-Register (0-3) | Type | Mnemonic |
| 0 | 0 | 0 | 5 | IO | IO |
| 0 | 0 | 1 | 0 | Illegal | |
| 0 | 1 | 0 | 5 | IO | IOH |
| 0 | 1 | 1 | 5 | IO | IOLD |
| 0 | 2 | 0 | E | SO | NEG |
| 0 | 2 | 1 | 6 | SO | LB |
| 0 | 3 | 0 | 0 | Illegal | |
| 0 | 3 | 1 | D | SO | JMP |
| 0 | 4 | 0 | E | SO | AID |
| 0 | 4 | 1 | F | SO | SID |
| 0 | 5 | X | 0 | Illegal | |
| 0 | 6 | 0 | E | SO | CPL |
| 0 | 6 | 1 | 0 | Illegal | |
| 0 | 7 | 0 | 7 | SO | CL |
| 0 | 7 | 1 | F | SO | CLH |
| 0 | 8 | 0 | 6 | SO | LBF |
| 0 | 8 | 1 | F | SO | DEC |
| 0 | 9 | 0 | 6 | SO | LBT |
| 0 | 9 | 1 | F | SO | CMZ |
| 0 | A | 0 | 6 | SO | LBS |
| 0 | A | 1 | F | SO | INC |
| 0 | B | 0 | 6 | SO | LBC |
| 0 | B | 1 | F | SO | ENT |
| 0 | C | 0 | E | SO | STS |
| 0 | C | 1 | F | SO | LDI |
| 0 | D | 0 | E | SO | SDI |
| 0 | D | 1 | 4 | SO | CMN |
| 0 | E | 0 | E | SO | LEV |
| 0 | E | 1 | F | SO | CAD |
| 0 | F | 0 | E | SO | SAVE |
| 0 | F | 1 | F | SO | RSTR |

Table A-3. Op-Code Group (Sheet 2 of 2)

| Input | | | Output | Instruction | |
|---|---|---|---|---|---|
| F-Register (1-3) | F-Register (4-7) | F-Register (8) | M-Register (0-3) | Type | Mnemonic |
| 1-7 | 0 | 0 | 2 | DO | MTM |
| 1-7 | 0 | 1 | 3 | DO | LDH |
| 1-7 | 1 | 0 | 0 | Illegal | |
| 1-7 | 1 | 1 | 3 | DO | CMH |
| 1-7 | 2 | 0 | 2 | DO | SUB |
| 1-7 | 2 | 1 | 3 | DO | LLH |
| 1-7 | 3 | 0 | 2 | DO | DIV |
| 1-7 | 3 | 1 | F | DO | LNJ |
| 1-7 | 4 | 0 | 2 | DO | OR |
| 1-7 | 4 | 1 | 3 | DO | ORH |
| 1-7 | 5 | 0 | 2 | DO | AND |
| 1-7 | 5 | 1 | 3 | DO | ANH |
| 1-7 | 6 | 0 | 2 | DO | XOR |
| 1-7 | 6 | 1 | 3 | DO | XOH |
| 1-7 | 7 | 0 | 2 | DO | STM |
| 1-7 | 7 | 1 | F | DO | STH |
| 1-7 | 8 | 0 | 9 | DO | LDR |
| 1-7 | 8 | 1 | 1 | SIP | (May be redundant) |
| 1-7 | 9 | 0 | A | DO | CMR |
| 1-7 | 9 | 1 | 1 | SIP | (May be redundant) |
| 1-7 | A | 0 | C | DO | ADD |
| 1-7 | A | 1 | 3 | DO | SRM |
| 1-7 | B | 0 | 2 | DO | MUL |
| 1-7 | B | 1 | E | DO | LAB |
| 1-7 | C | 0 | 1 | SIP | (May be redundant) |
| 1-7 | C | 1 | 4 | DO | LDB |
| 1-7 | D | 0 | 1 | SIP | (May be redundant) |
| 1-7 | D | 1 | 4 | DO | CMB |
| 1-7 | E | 0 | 8 | DO | SWR |
| 1-7 | E | 1 | 4 | DO | SWB |
| 1-7 | F | 0 | B | DO | STR |
| 1-7 | F | 1 | 4 | DO | STB |

### A.2.3 Index Group (A)

The index group is enabled when RDDT bits 05 through 08 equal a hexadecimal A. The op-code group is further defined in Table A-4.

Table A-4. Index Group (Sheet 1 of 2)

| Input | | | | Output | |
|---|---|---|---|---|---|
| F-Register (1-3) | F-Register (4-7) | F-Register (8) | LAF[a] | M-Register (0-3) | Instruction Type/Mnemonic |
| 0 | 0 | 0 | X | 2 | Word |
| 0 | 0 | 1 | X | 0 | Illegal |
| 0 | 1 | X | X | 6 | Byte |
| 0 | 2 | 0 | X | 2 | Word |
| 0 | 2 | 1 | X | 4 | Bit |
| 0 | 3 | 0 | X | 0 | Illegal |
| 0 | 3 | 1 | X | 2 | Word |
| 0 | 4 | X | X | 3 | Double Word |
| 0 | 5 | X | X | 0 | Illegal |
| 0 | 6 | 0 | X | 2 | Word |
| 0 | 6 | 1 | X | 0 | Illegal |
| 0 | 7 | 0 | X | 2 | Word |
| 0 | 7 | 1 | X | 6 | Byte |
| 0 | 8 | 0 | X | 4 | Bit |
| 0 | 8 | 1 | X | 2 | Word |
| 0 | 9 | 0 | X | 4 | Bit |
| 0 | 9 | 1 | X | 2 | Word |
| 0 | A | 0 | X | 4 | Bit |
| 0 | A | 1 | X | 2 | Word |
| 0 | B | 0 | X | 4 | Bit |
| 0 | B | 1 | X | 2 | Word |
| 0 | C | 0 | X | 2 | Word |
| 0 | C | 1 | X | 3 | Double Word |
| 0 | D | 0 | X | 3 | Double Word |
| 0 | D | 1 | 0 | 2 | Word |
| 0 | D | 1 | 1 | 3 | Double Word |
| 0 | E | X | X | 2 | Word |
| 0 | F | X | X | 2 | Word |
| 1-7 | 0 | 0 | X | 2 | Word |
| 1-7 | 0 | 1 | X | 6 | Byte |
| 1-7 | 1 | 0 | X | 0 | Illegal |
| 1-7 | 1 | 1 | X | 6 | Byte |
| 1-7 | 2 | 0 | X | 2 | Word |
| 1-7 | 2 | 1 | X | 6 | Byte |
| 1-7 | 3 | X | X | 2 | Word |
| 1-7 | 4 | 0 | X | 2 | Word |
| 1-7 | 4 | 1 | X | 6 | Byte |
| 1-7 | 5 | 0 | X | 2 | Word |
| 1-7 | 5 | 1 | X | 6 | Byte |
| 1-7 | 6 | 0 | X | 2 | Word |

Table A-4. Index Group (Sheet 2 of 2)

| Input | | | | Output | |
|---|---|---|---|---|---|
| F-Register (1-3) | F-Register (4-7) | F-Register (8) | LAF[a] | M-Register (0-3) | Instruction Type/Mnemonic |
| 1-7 | 6 | 1 | X | 6 | Byte |
| 1-7 | 7 | 0 | X | 2 | Word |
| 1-7 | 7 | 1 | X | 6 | Byte |
| 1-7 | 8 | 0 | X | 2 | Word |
| 1-3 | 8 | 1 | X | 1 | Scientific (2 or 4 words) |
| 4 | 8 | 1 | X | 3 | Double Word |
| 5-7 | 8 | 1 | X | 1 | Scientific (2 or 4 words) |
| 1-7 | 9 | 0 | X | 2 | Word |
| 1-3 | 9 | 1 | X | 1 | Scientific (2 or 4 words) |
| 4 | 9 | 1 | X | 3 | Double Word |
| 5-7 | 9 | 1 | X | 1 | Scientific (2 or 4 words) |
| 1-7 | A | X | X | 2 | Word |
| 1-7 | B | X | X | 2 | Word |
| 1-3 | C | 0 | X | 1 | Scientific (2 or 4 words) |
| 4 | C | 0 | X | 5 | Quad Word |
| 5-7 | C | 0 | X | 1 | Scientific (2 or 4 words) |
| 1-7 | C | 1 | 0 | 2 | Word |
| 1-7 | C | 1 | 1 | 3 | Double Word |
| 1-3 | D | 0 | X | 1 | Scientific (2 or 4 words) |
| 4 | D | 0 | X | 5 | Quad Word |
| 5-7 | D | 0 | X | 1 | Scientific (2 or 4 words) |
| 1-7 | D | 1 | 0 | 2 | Word |
| 1-7 | D | 1 | 1 | 3 | Double Word |
| 1-7 | E | 0 | X | 2 | Word |
| 1-7 | E | 1 | 0 | 2 | Word |
| 1-7 | E | 1 | 1 | 3 | Double Word |
| 1-7 | F | 0 | X | 2 | Word |
| 1-7 | F | 1 | 0 | 2 | Word |
| 1-7 | F | 1 | 1 | 3 | Double Word |

[a] A One indicates Long Address Format (LAF).

### A.2.4 Generic Group (B)

The generic group is enabled when RDDT bits 05 through 08 equal a hexadecimal B. The generic group is further defined in Table A-5.

Table A-5. Generic Group

| Input | | Output | |
|---|---|---|---|
| F-Register (9-11) | F-Register (12-15) | M-Register (0-3) | Instruction Mmemonic/Class |
| 0 | 0 | 1 | Halt |
| 0 | 1 | 1 | MCL |
| 0 | 2 | 1 | BRK |
| 0 | 3 | 1 | RTT |
| 0 | 4 | 1 | RTCN |
| 0 | 5 | 1 | RTCF |
| 0 | 6 | 1 | WDTN |
| 0 | 7 | 1 | WDTF |
| 0 | 8 | 1 | MMM |
| 0 | 9 | 1 | Generic (Unassigned) |
| 0 | A | 1 | ASD |
| 0 | B | 1 | VLD |
| 0 | C | 1 | LRDB |
| 0 | D | 1 | SRDB |
| 0 | E | 1 | Generic (Unassigned) |
| 0 | F | 1 | Generic (Unassigned) |
| 1 | 0 | 2 | STAX (LDT, STT, ACQ, RLQ) |
| 1 | 1 | 3 | RSC |
| 1 | 2-F | 0 | Illegal |
| 2 | 0-F | 4 | CIP |
| 3 | 0-F | 5 | CIP (Illegal) |
| 4 | 0-F | 0 | Illegal |
| 5 | 0-F | 0 | Illegal |
| 6 | 0 | 6 | DQA |
| 6 | 1 | 7 | QOT |
| 6 | 2 | 8 | DQH |
| 6 | 3 | 9 | QOH |
| 6 | 4-F | 0 | Illegal |
| 7 | 0-F | 0 | Illegal |

## A.2.5 Address Syllable Group (C)

The address syllable group is enabled when RDDT bits 05 through 08 equal a hexadecimal C. The address syllable group is further defined in Table A-6.

Table A-6. Address Syllable Group

| Input | | | Output | |
|---|---|---|---|---|
| F-Register (9-11) | F-Register (12) | F-Register (13-15)=10 | M-Register (0-3) | Comments Address Type |
| 0 | 0 | 0 | 1 | IMA |
| 0 | 0 | 1-7 | 3 | BN |
| 0 | 1 | 0 | 2 | IMA |
| 0 | 1 | 1-7 | 4 | BN |
| 1 | 0 | 0 | 5 | IMA+R1 |
| 1 | 0 | 1-7 | 6 | BN&R1 |
| 1 | 1 | 0 | 5 | IMA[a]+R1 |
| 1 | 1 | 1-7 | 7 | BN[a]+R1 |
| 2 | 0 | 0 | 5 | IMA+R2 |
| 2 | 0 | 1-7 | 6 | BN+R2 |
| 2 | 1 | 0 | 5 | IMA[a]+R2 |
| 2 | 1 | 1-7 | 7 | BN[a]+R2 |
| 3 | 0 | 0 | 5 | IMA+R3 |
| 3 | 0 | 1-7 | 6 | BN+R3 |
| 3 | 1 | 0 | 5 | IMA[a]+R3 |
| 3 | 1 | 1-7 | 7 | BN[a]+R3 |
| 4 | 0 | 0 | 8 | P+DSP |
| 4 | 0 | 1-7 | 8 | BN+DSP |
| 4 | 1 | 0 | 8 | (P+DSP) |
| 4 | 1 | 1-7 | 8 | (BN+DSP) |
| 5 | 0 | 0 | 0 | Reserved |
| 5 | 0 | 1-7 | B | BN,RN(RAS) |
| 5 | 1 | 0 | 0 | Reserved |
| 5 | 1 | 1-3 | C | BN+R1 |
| 5 | 1 | 4 | 0 | Reserved |
| 5 | 1 | 5-7 | D | B(N-4)+R1 |
| 6 | 0 | 0 | 0 | Reserved |
| 6 | 0 | 1-7 | E | BN(Push) |
| 6 | 1 | 0 | 0 | Reserved |
| 6 | 1 | 1-3 | C | BN+R2 |
| 6 | 1 | 4 | 0 | Reserved |
| 6 | 1 | 5-7 | D | B(N-4)+R2 |
| 7 | 0 | 0 | 9 | IMO |
| 7 | 0 | 1-7 | F | BN[a](Pop) |
| 7 | 1 | 0 | A | IV+DSP |
| 7 | 1 | 1-3 | C | BN+R3 |
| 7 | 1 | 4 | 0 | Reserved |
| 7 | 1 | 5-7 | D | B(N-4)+R3 |

[a] Indirect Address

### A.2.6 CIP Data Descriptor Address Syllable Group (D)

The CIP data descriptor address syllable group is enabled when RDDT bits 05 through 08 equal a hexadecimal D. The CIP data descriptor address syllable group is further defined in Table A-7.

Table A-7. CIP Data Descriptor Address Syllable Group

| Input | | | Output | |
|---|---|---|---|---|
| F-Register (9-11) | F-Register (12) | F-Register (13-15)=N | M-Register (0-3) | Comments |
| 0 | 0 | 0 | 1 | Remote |
| 0 | 0 | 1-7 | 6 | D+DSP |
| 0 | 1 | 0 | 2 | P+DSP |
| 0 | 1 | 1-7 | 7 | (B+DSP)[a] |
| 1 | 0 | 0 | 1 | Remote |
| 1 | 0 | 1-7 | 8 | BN+DISP+R1 |
| 1 | 1 | 0 | 4 | P+DISP+R1 |
| 1 | 1 | 1-7 | 9 | (BN+DISP)[a] +R1 |
| 2 | 0 | 0 | 1 | Remote |
| 2 | 0 | 1-7 | 8 | BN+DISP+R2 |
| 2 | 1 | 0 | 4 | P+DISP+R2 |
| 2 | 1 | 1-7 | 9 | (BN+DISP)[a] +R2 |
| 3 | 0 | 0 | 1 | Remote |
| 3 | 0 | 1-7 | 8 | BN+DISP+R3 |
| 3 | 1 | 0 | 4 | P+DISP+R3 |
| 3 | 1 | 1-7 | 9 | (BN+DISP)[a] +R3 |
| 4 | 0 | 0 | 1 | Remote |
| 4 | 0 | 1-7 | 8 | BN+DISP+RS |
| 4 | 1 | 0 | 3 | (P+DSP)[a] |
| 4 | 1 | 1-7 | 9 | (BN+DISP)[a] +R4 |
| 5 | 0 | 0 | 1 | Remote |
| 5 | 0 | 1-7 | 8 | BN+DISP+R5 |
| 5 | 1 | 0 | 0 | RFU(Illegal) |
| 5 | 1 | 1-7 | 9 | (BN+DISP)[a] +R5 |
| 6 | 0 | 0 | 1 | Remote |
| 6 | 0 | 1-7 | 8 | BN+DISP+R6 |
| 6 | 1 | 0 | 0 | RFU(Illegal) |
| 6 | 1 | 1-7 | 9 | (BN+DISP)[a] +R6 |
| 7 | 0 | 0 | 1 | Remote |
| 7 | 0 | 1-7 | 8 | BN+DISP+R7 |
| 7 | 1 | 0 | 5 | IMO |
| 7 | 1 | 1-7 | 9 | (BN+DISP)[a] +R7 |

[a] Indirect Address

## A.2.7 PBRANCH Group (E)

The PBRANCH group is enabled when RDDT bits 05 through 08 equal a hexadecimal E. The result of the test conditions (M0 through M3) is shown below:

| RDDT 05-08 | M 0 | M1 | M2 | M3 |
|---|---|---|---|---|
| E | 0 | PBRANCH | F9TEZ | FR15 |

The PBRANCH indicator equals One if FR08 equals Zero and the condition tested is true, or if FR08 equals One and the condition tested is false. The PBRANCH conditions are listed in Table A-8.

Table A-8. PBRANCH Group

| Selection Bits | | | |
|---|---|---|---|
| F-Register (1-3) | F-Register (4-7) | Condition Tested | Remarks |
| 0 | 0-1 | 0 | Not used in L6 |
| 0 | 2 | I(L) | Less than indicator |
| 0 | 3 | I(G) | Greater than indicator |
| 0 | 4 | I(OV) | Overflow indicator |
| 0 | 5 | I(B) | Bit test indicator |
| 0 | 6 | I(C) | Carry indicator |
| 0 | 7 | I(I) | Input/output indicator |
| 0 | 8 | I(U) + I(L) | Arithmetic less than |
| 0 | 9 | $\overline{I(G) + I(L)}$ | Equal |
| 0 | A | I(U) + I(G) | Arithmetic greater than |
| 0 | B | I(U) | Unlike signs indicator |
| 0 | C-E | 0 | Not used in L6 |
| 0 | F | 0 | NOP/Unconditional Branch |
| 1-7 | 0-6 | SEL M-register bit | See Note 1 |
| 1-7 | 7 | TF2 } See | Carry 16=1 |
| 1-7 | 8 | TF0 } Note | BI00 = 1 |
| 1-7 | 9 | TF1 } 2 | BI00-15 = 0 |
| 1-7 | A | $\overline{(TF0+TF1)}$ | BI00-15 > 0 |
| 1-7 | B | $\overline{TF7}$ | BI15 = 0 |
| 1-7 | C-F | 0 | Not used in L6 |

NOTES

1. M-register selected by register file address bits. The bit in the M-register field is selected by F-register bits 01-03.

2. Temporary flip-flops store information from the previous microcycle.

# *Appendix B* *TIMING*

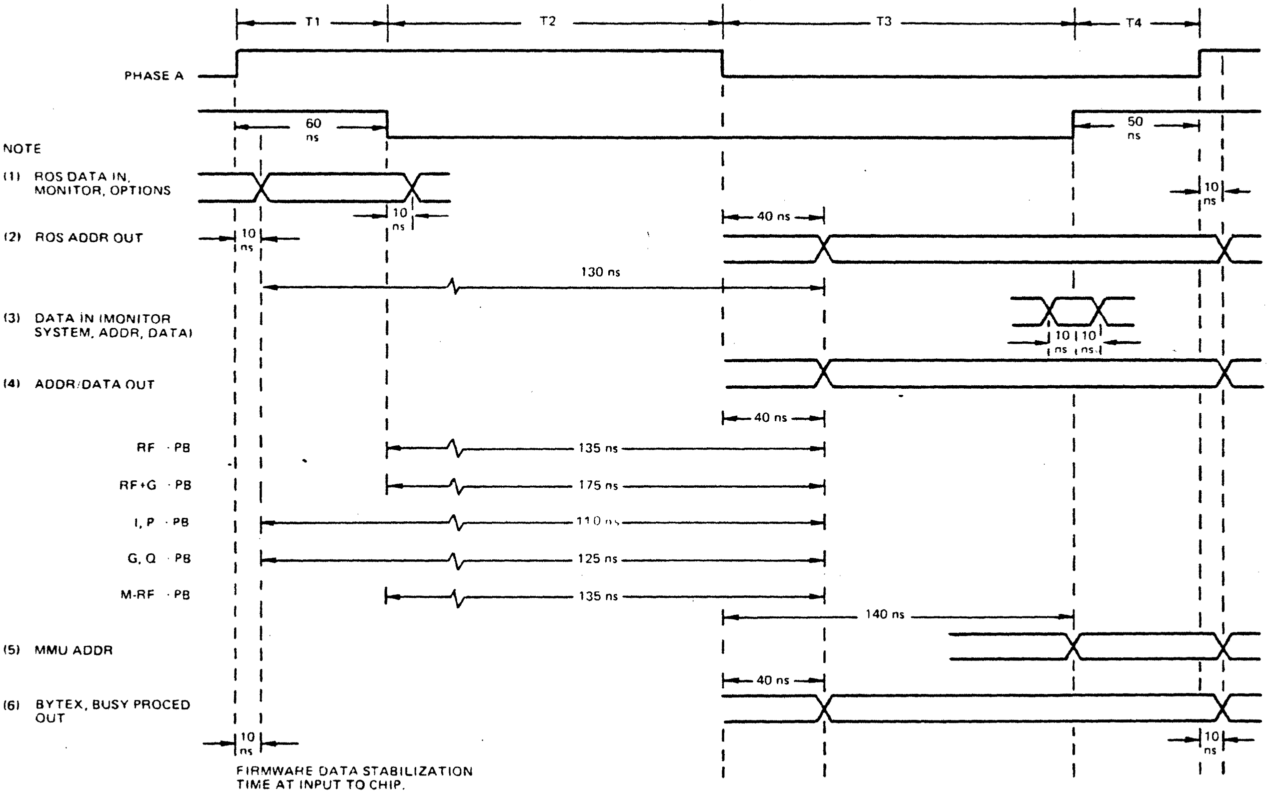
This appendix provides a detailed timing description of the LSI-6 microprocessor chip. Figure B-1 shows the timing requirements for interfacing with the external system. Figure B-2 indicates which clock signal transition controls the latching, setting, resetting, or loading of all internal registers, flip-flops, and latches. Figure B-3 lists events that occur during the four time slots (T1 through T4) that are generated internally from Phase A and Phase B clock signals.

NOTE

> The stated chip time delays are preliminary, based upon computer simulation. Final timing numbers will be specified after the LSI-6 has been fabricated and characterized.

Figure B-3 also includes the minimum slot times required by internal functional blocks. Some conflicts exist with these times but they are resolved when specific operations are considered. For example, a register file Read operation requires that T1 plus T2 have a minimum time of 110 nanoseconds. However, if an MMU address is being developed, T1 plus T2 will have a minimum time of 185 nanoseconds. This allows variable frequency clocking (depending upon the instruction being executed) to allow a user to maximize performance.

Figure B-4 illustrates three possible types of operation that can occur internally and the timing relationship associated with each operation. The times that are presented in the figure are typical times and can vary depending on the type of transfer.



NOTES

1. ROS ADDRESS IN: ROS DATA 00-48

   MONITOR BITS – OPTIONS:

   SIPE LAFE
   CIPE MEMVAL
   MMUE

2. ROS ADDR OUT: ROS ADDRESS 00-11

3. DATA IN:

   (A) MONITOR BITS · SYSTEM STATUS:

| | |
|---|---|
| TESTBR1[a] | OBC |
| BYTEX(IN)[a] | MEMPAR |
| PROCED(IN)[a] | MEMPRES |
| BUSY(IN)[a] | MPLOCK[a] |
| DET0-4 | POWON |
| INTR0-2 | MEM RFRH |
| MIBGP0-3 | |

   (B) ADDRESS/DATA:

   BUS DATA 00-15

4. ADDR/DATA OUT: BUS ADDRESS 00-15
   BUS DATA 00-15

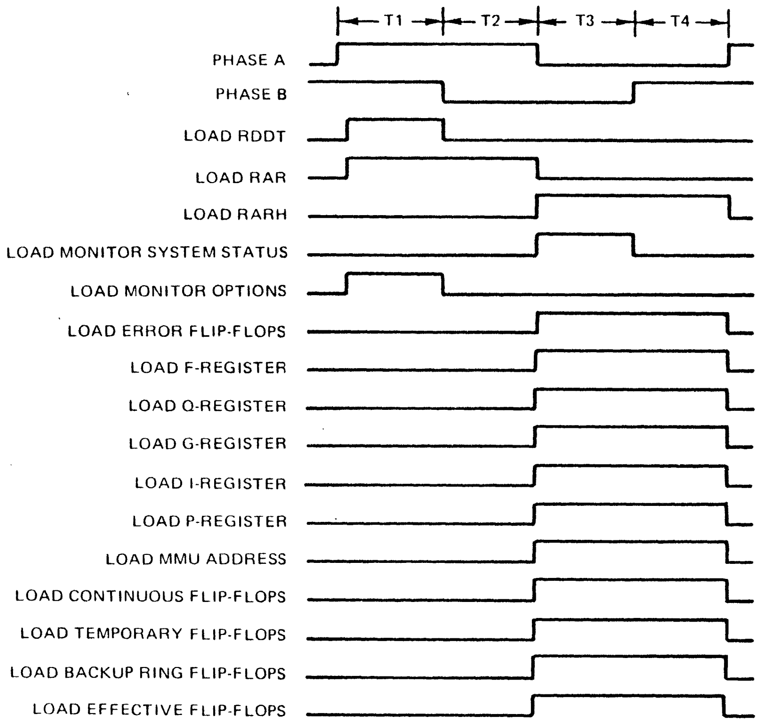
   TIMING IS SHOWN FOR THE VARIOUS SOURCES OF ADDRESS/DATA INFORMATION INTERNALLY. INFORMATION IS AVAILABLE AT THE OUTPUT OF THE CHIP AFTER THE INDICATED DELAY, OR 40 NANOSECONDS AFTER THE FALLING EDGE OF PHASE A, WHICHEVER IS GREATER.

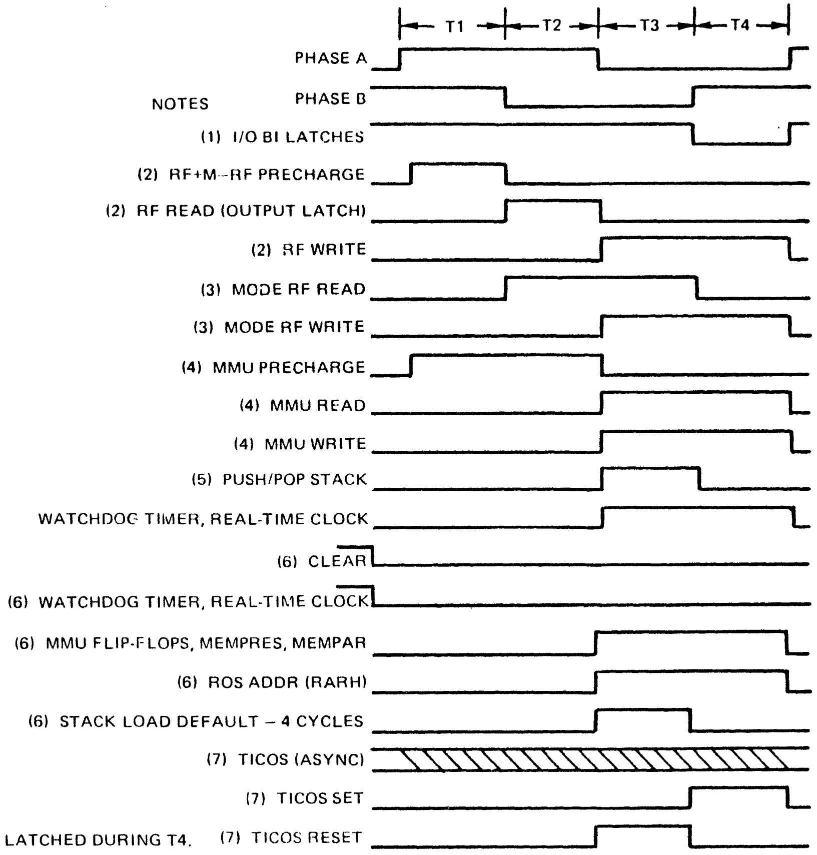
5. MMU ADDR: BUS ADDRESS 00-15
   MEMKIL

6. BYTEX(OUT)
   BUSY(OUT)
   PROCED(OUT)

[a]REFER TO TABLE 4-3 FOR LSI-6 NOMENCLATURE.

Figure B-1. External System Timing Requirements





NOTES

1. INTERNAL OR EXTERNAL SOURCES CAN DRIVE THE INTERNAL BUS (BI) DURING T3, BI IS LATCHED DURING T4.

2. REGISTER FILE (RF): DURING PRECHARGE, ALL REGISTER FILE OUTPUT IS LOW; OUTPUT IS ENABLED DURING PHASE A, BUT VALID DATA DOES NOT APPEAR UNTIL AFTER T2 ENABLES THE FILE ADDRESS. OUTPUT IS LATCHED AT THE END OF T2. NO PRECHARGE OCCURS DURING WRITE OPERATIONS.

3. OPERATION OF THE MODE RF (M-RF) IS THE SAME AS THE RF.

4. MMU-RF: DURING PRECHARGE, THE DESCRIPTOR ARRAY OUTPUT IS LOW, EXCEPT FOR BIT 0 WHICH IS HIGH. THIS PREVENTS ANY MMU ERROR CONDITIONS FROM BEING DETECTED FALSELY DURING PRECHARGE. FILE ACCESS DOES NOT BEGIN UNTIL T3.

5. STACK: DATA INTO THE STACK IS LOADED DURING PHASE A AND HELD DURING PHASE A, IN LATCHES; DATA IS TRANSFERRED TO/FROM STACK DURING T3.

6. CLEAR FUNCTIONALITY:

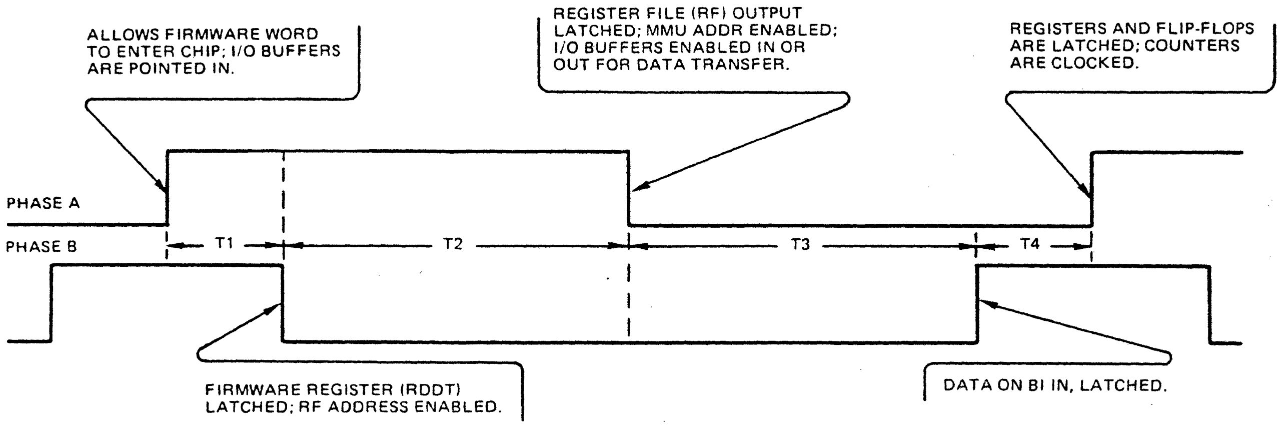
   MMU FLIP-FLOPS, MEMPRES, AND MEMPAR ARE RESET ON FIRST PHASE A FALLING TRANSITION.

   ROS ADDR IS FORCED TO ALL ZEROS ON FIRST $\overline{\phi}$A FALLING EDGE.

   STACK IS "POPPED" FOUR TIMES TO FILL IT WITH DEFAULT CODE (0001); FOUR T3 TIMES ARE REQUIRED.

7. TICOS IS AN ASYNCHRONOUS SIGNAL HAVING A TRANSITION EVERY 8.33 MILLISECONDS. IT IS STROBED INTO AN EDGE TRIGGERED FLIP-FLOP AT T3; THE FLIP-FLOP CAN BE EDGE TRIGGERED RESET AT T2 UNDER FIRMWARE CONTROL.

Figure B-2. Internal Timing Requirements


ALLOWS FIRMWARE WORD TO ENTER CHIP; I/O BUFFERS ARE POINTED IN.
REGISTER FILE (RF) OUTPUT LATCHED; MMU ADDR ENABLED; I/O BUFFERS ENABLED IN OR OUT FOR DATA TRANSFER.
REGISTERS AND FLIP-FLOPS ARE LATCHED; COUNTERS ARE CLOCKED.
PHASE A
PHASE B
T1
T2
T3
T4
FIRMWARE REGISTER (RDDT) LATCHED; RF ADDRESS ENABLED.
DATA ON BI IN, LATCHED.

CLOCK INPUTS

PHASE A – CONTROLS DIRECTION OF THE DATA FLOW INTO AND OUT OF THE CHIP; THAT IS, TRISTATE ENABLE/DISABLE. THE LEADING AND TRAILING EDGES LATCH REGISTERS AND FLIP-FLOPS.

PHASE B – USED AS A STROBE PULSE FOR LATCHING I/O DATA.

INTERNAL TIMESLOTS

T1 $\Phi A \bullet \Phi B$.THE FIRMWARE WORD IS LOADED INTO THE CHIP; THE I/O BUFFERS ARE POINTED IN; THE RF, M-RF, AND MMU RF ARE BEING PRECHARGED; A MINIMUM OF 60 NANOSECONDS IS REQUIRED FOR FIRMWARE REGISTER SETUP TIME AND FILE PRECHARGE TIME.

T2 $\Phi A \bullet \overline{\Phi B}$. RF/ALU/DATA OPERATIONS OCCUR; THE NEXT FIRMWARE ADDRESS IS GENERATED; THE I/O BUFFERS ARE POINTED IN; THE MMU IS BEING PRECHARGED; THE RF AND M-RF ACCESS OCCURS A MINIMUM OF 60 NANOSECONDS.

T3 $\overline{\Phi A \bullet \Phi B}$. MMU OPERATION OCCURS; THE I/O DRIVERS ARE POINTED IN OR OUT; THE NEXT FIRMWARE ADDRESS IS SENT OUT; THE PUSH/POP OF THE FIRMWARE STACK OCCURS.

T4 $\overline{\Phi A} \bullet \Phi B$. DATA ON THE INTERNAL BUS IS LATCHED; A MINIMUM OF 50 NANOSECONDS FOR RF SETUP.

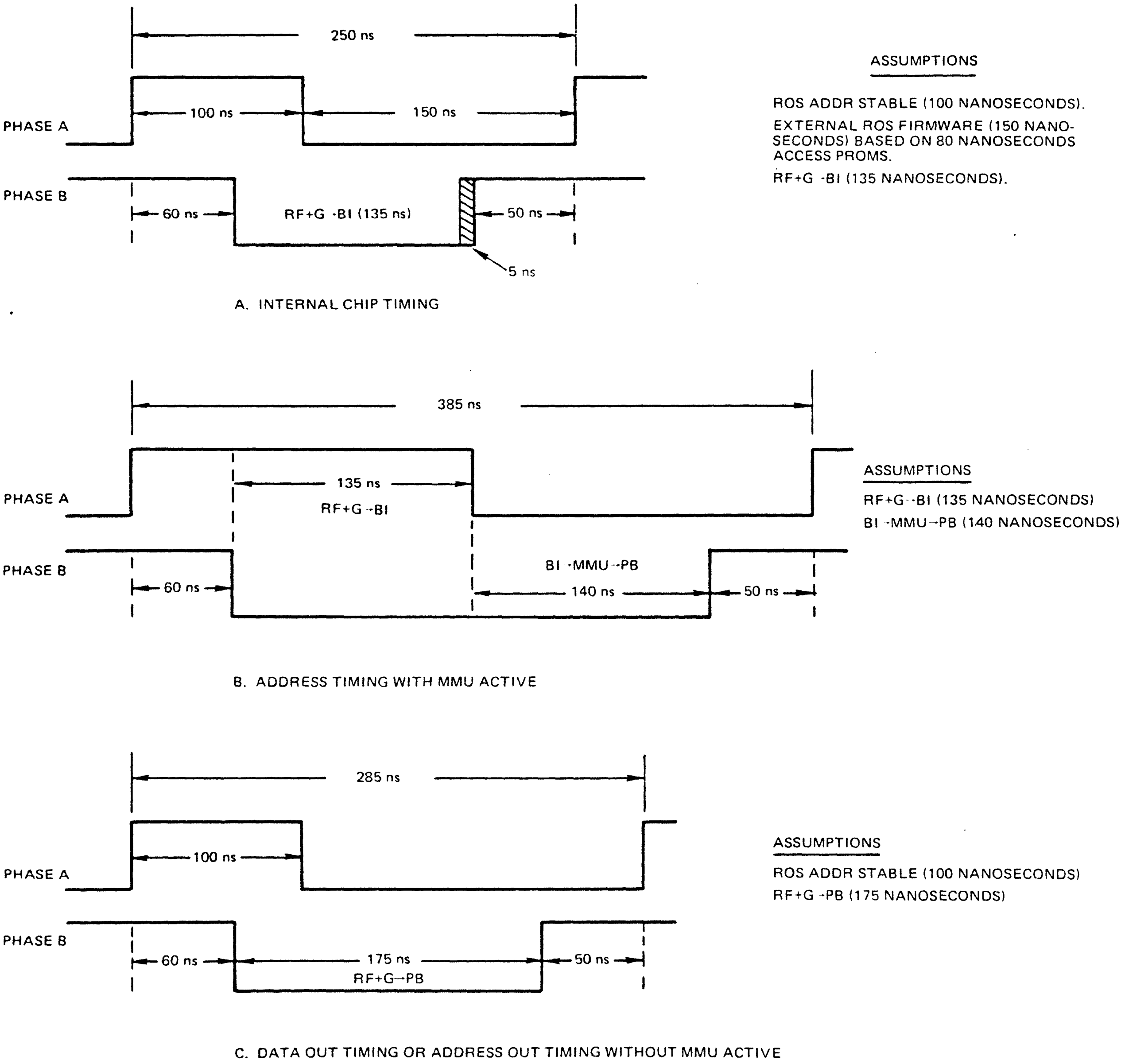
T1 + T2 EQUALS A MINIMUM OF 195 NANOSECONDS FOR MMU PRECHARGE TIME IF DESCRIPTOR ARRAY ADDRESS IS INDEXED; 155 NANOSECONDS MINIMUM TIME IF ADDRESS IS NOT INDEXED.

Figure B-3. Time Slot Events with Phase A and Phase B

Figure B-4A illustrates the internal chip timing assuming the following conditions: ROS address stable, external ROS firmware based on 80-nanosecond access PROMS, and RF+G --> MMU --> PB.

Figure B-4B illustrates address timing with MMU active assuming the following conditions: RF+G --> BI and BI --> MMU --> PB.

Figure B-4C illustrates data out timing or address out timing with MMU not active assuming the following conditions: ROS address stabilized and RF+G --> PB.



ASSUMPTIONS

ROS ADDR STABLE (100 NANOSECONDS).
EXTERNAL ROS FIRMWARE (150 NANOSECONDS) BASED ON 80 NANOSECONDS ACCESS PROMS.
RF+G -BI (135 NANOSECONDS).

A. INTERNAL CHIP TIMING

ASSUMPTIONS

RF+G -BI (135 NANOSECONDS)
BI -MMU -PB (140 NANOSECONDS)

B. ADDRESS TIMING WITH MMU ACTIVE

ASSUMPTIONS

ROS ADDR STABLE (100 NANOSECONDS)
RF+G -PB (175 NANOSECONDS)

C. DATA OUT TIMING OR ADDRESS OUT TIMING WITHOUT MMU ACTIVE

Figure B-4. Optimum Microcycle Timing of Phase A and Phase B

# *Appendix C*
# *PIN IDENTIFICATION AND PLACEMENT*

The LSI-6 microprocessor chip incorporates a 68-pin leadless chip. The chip is mounted on a 68-pin Dual Inline Package (DIP) socket which is then soldered on a Printed Wire Assembly (PWA). Figure C-1 illustrates the pin identification and placement as viewed from the component side of the PWA. Table C-1 provides the interface signal name associated with each pin.

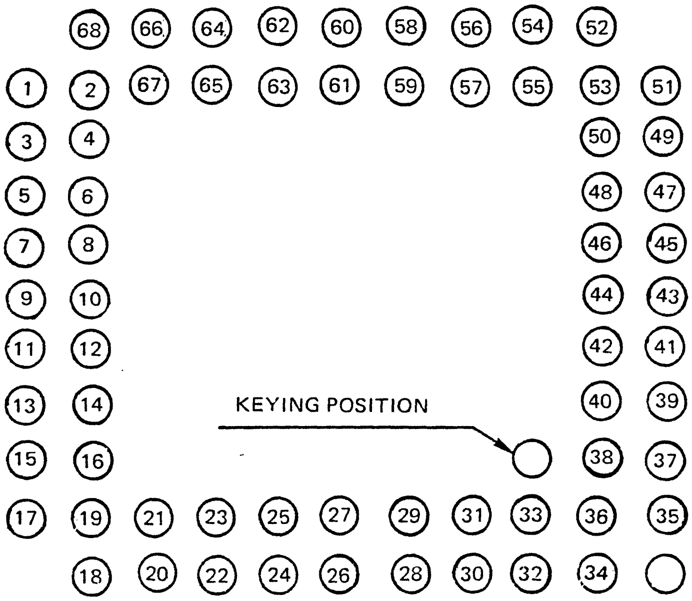


Figure C-1. Pin Identification and Placement

Table C-1. Interface Signal Name to Pin Number Correlation

| Pin No. | Interface Signal Name | Pin No. | Interface Signal Name |
|---|---|---|---|
| 1 | VBB (-5V) | 35 | DATRQ1-33 |
| 2 | No connection | 36 | MEMPAR-P0 |
| 3 | No connection | 37 | MREFSH-P4 |
| 4 | INTRQ2-28 | 38 | DATRQ0-32 |
| 5 | POWRON+22 | 39 | DATRQ2-27 |
| 6 | DABS15+47 | 40 | RSAD11+12 |
| 7 | DABS14+46 | 41 | RSAD10+11 |
| 8 | DABS13+45 | 42 | RSAD9+10 |
| 9 | DABS12+44 | 43 | RSAD8+9 |
| 10 | DABS11+43 | 44 | RSAD7+8 |
| 11 | DABS10+42 | 45 | RSAD6+7 |
| 12 | DABS09+41 | 46 | RSAD5+6 |
| 13 | DABS08+40 | 47 | RSAD4+5 |
| 14 | VCC (+5V) | 48 | RSAD3+4 |
| 15 | VSS (GND) | 49 | RSAD2+3 |
| 16 | PHASE B (ØB) | 50 | RSAD1+2 |
| 17 | PHASE A (ØA) | 51 | RSAD0+1 |
| 18 | DABS07+39 | 52 | VSS (GND) |
| 19 | DABS06+38 | 53 | VCC (+5V) |
| 20 | DABS05+37 | 54 | BYTEXX-00[a] |
| 21 | DABS04+36 | 55 | BUSYXX-P1[a] |
| 22 | DABS03+35 | 56 | ONBDCN-0A[a] |
| 23 | DABS02+25 | 57 | MPLOCK[a] |
| 24 | DABS01+24 | 58 | PROCED-P2[a] |
| 25 | DABS00+23 | 59 | MIBGP0+13[a] |
| 26 | DABS0A+21 | 60 | MIBGP1+14[a] |
| 27 | DABS0B+20 | 61 | MIBGP2+15[a] |
| 28 | DABS0C+19 | 62 | MIBGP3+16[a] |
| 29 | DABS0D+18 | 63 | TSTBR1+17[a] |
| 30 | MEMPRES-P3 | 64 | No connection |
| 31 | CLEARX-0A | 65 | TICKOS+0A |
| 32 | DATRQ4-34 | 66 | INTRQ0-30 |
| 33 | DATRQ3-26 | 67 | INTRQ1-29 |
| 34 | MEMKIL+31 | 68 | No connection |

[a] Refer to Table 2-9 for LSI-6 equivalent nomenclature.

# *Appendix D*
# *LEVEL 6 INSTRUCTION SET*

## SECTION 1

## INSTRUCTION TYPES

**Type/Source Format** | **Memory Format**

Generic (GE)
[label] op

| 0 | | | | | | | 7 | 8–15 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | f |
| additional words as needed | | | | | | | | |

f — Function.

Branch on Indicators (BI)
[label] op branchloc

| 0 | 1–3 | 4–8 | 9–15 |
|---|---|---|---|
| 0 | 0 0 0 | op | d |

d — Displacement, calculated by the assembler as follows:
If "<branchloc" is coded, d = 0 and the next word(s) contain a pointer to branchloc
If "branchloc" is coded, d = 1 and the next word contains the displacement (DSP) to branchloc. If ">branchloc" is coded, d = –64 through +63, which is the displacement to branchloc; ">branchloc" must not tag this instruction or the next word.

Branch on Registers (BR)
[label] op r#,branchloc

| 0 | 1 3 | 4 8 | 9 15 |
|---|---|---|---|
| 0 | r# | op | d |

d — See "Branch on Indicators," above.

Shift Short (SHS) and
Shift Long (SHL).
[label] op r#,s

| 0 | 1 3 | 4 7 | 8 15 |
|---|---|---|---|
| 0 | r# | 0 0 0 0 | t and s |

t — Type and direction of shift. Requires bits 8 — 11 for SHS and bits 8 — 10 for SHL.

s — Shift distance. Requires bits 12 — 15 for SHS and bits 11 — 15 for SHL. (If s = 0, the shift distance is obtained from R1.)

Short Value Immediate (SI)
[label] op r#,v

| 0 | 1 3 | 4 7 | 8 15 |
|---|---|---|---|
| 0 | r# | op | v |

v — Value between −128 and +127.

Input/Output (I/O)
See Systems Handbook, CC71

Single Operand (SO
[label] op as[,maskword]

| 0 | 1 3 | 4 8 | 9 15 |
|---|---|---|---|
| 1 | 0 0 0 | op | as |

as —See "Address Syllable."

maskword — Used only in SAVE, RSTR, and (optionally) bit instructions. If maskword is all zeros, the mask is obtained from R1.

Double Operand (DO)
[label] op r#,as[,maskword]

| 0 | 1 3 | 4 8 | 9 15 |
|---|---|---|---|
| 1 | r# | op | as |

as — See "Address Syllable."

maskword — Used only in SRM instruction. If maskword is all zeros, the mask is obtained from R1.

| | |
|---|---|
| DSP | 16-bit displacement (−32,768 through +32,767) |
| op | Operation code |
| r# | Register number |
| [] | Optional |

# SECTION 2

## BASIC INSTRUCTIONS

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| ADD | DO | Add to R Reg | R# ← R# + [EA]<br>I(ov), I(c) affected |
| ADV | SI | Add Value to R Reg | R# ← R# + V<br>I(ov), I(c) affected |
| ACQ | GE | Acquire Stack Space | See Note |
| AID | SO | Add Double Word Integer | R6, R7, ← R6, R7 + [EA]<br>I(ov), I(c) affected |
| AND | DO | AND With R Reg | R# ← R#∧[EA] |
| ANH | DO | AND Halfword With R Reg | R# ← R#∧[EA] sign extended |
| ASD | GE | Activate Segment Descriptor | See Note |
| B | BI | Branch | P ← EA |
| BAG | BI | Br Alg Greater Than | If I(g) ⊕ I(u) = 1, then P ← EA |
| BAGE | BI | Br Alg Greater Than or Equal | If I(l) ⊕ I(u) = 0, then P ← EA |
| BAL | BI | Br Alg Less Than | If I(l) ⊕ I(u) = 1, then P ← EA |
| BALE | BI | Br Alg Less Than or Equal | If I(g) ⊕ I(u) = 0, then P ← EA |
| BBF | BI | Br on Bit Test Indicator False | If I(b) = 0, then P ← EA |
| BBT | BI | Br on Bit Test Indicator True | If I(b) = 1, then P ← EA |
| BCF | BI | Br on Carry False | If I(c) = 0, then P ← EA |
| BCT | BI | Br on Carry True | If I(c) = 1, then P ← EA |
| BDEC | BR | Br After Decrementing R Reg | R# ← R# + FFFF;<br>if R# ≠ FFFF, then P ← EA |
| BE | BI | Br Equal | If I(l) ∨ I(g) = 0, then P ← EA |
| BEVN | BR | Br if R Reg Even | If R#(15) = 0, then P ← EA |
| BEZ | BR | Br if R Reg Equal to Zero | If R# = 0, then P ← EA |
| BG | BI | Br Greater Than | If I(g) = 1, then P ← EA |
| BGE | BI | Br Greater Than or Equal | If I(l) = 0, then P ← EA |
| BGEZ | BR | Br if R Reg Greater Than or Equal to Zero | If R#(0) = 0, then P ← EA |
| BGZ | BR | Br if R Reg Greater Than Zero | If R#(1:15) ≠ 0<br>and R#(0) = 0, then P ← EA |

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| BINC | BR | Br After Incrementing R Reg | R# ← R# + 0001;<br>If R# ≠ 0, then P ← EA |
| BIOF | BI | Br on I/O Indicator False | If I(i) = 0, then P ← EA |
| BIOT | BI | Br on I/O Indicator True | If I(i) = 1, then P ← EA |
| BL | BI | Br Less Than | If I(l) = 1, then P ← EA |
| BLE | BI | Br Less Than or Equal | If I(g) = 0, then P ← EA |
| BLEZ | BR | Br if R Reg Less Than or Equal to Zero | If R#(0) = 1 or P# = 0,<br>then P ← EA |
| BLZ | BR | BR if R Reg Less Than Zero | If R#(0) = 1, then P ← EA |
| BNE | BI | Br Not Equal | If I(l) ∨ I(g) = 1,<br>then P ← EA |
| BNEZ | BR | Br if R Reg Not Equal to Zero | If R# ≠ 0, then P ← EA |
| BNOV | BI | Br on Not Overflow | !f I(ov) = 0, then P ← EA |
| BODD | BR | Br if R Reg Odd | If R#(15) = 1, then P ← EA |
| BOV | BI | Br on Overflow | If I(ov) = 1, then P ← EA |
| BRK | GE | Breakpoint | Generate trap via Trap Vector #2. |
| BSE | BI | Br Signs Equal | If I(u) = 0, then P ← EA |
| BSU | BI | Br Signs Unlike | If I(u) = 1, then P ← EA |
| CAD | SO | Carry Add | [EA] ← [EA] + I(c)<br>I(ov), I(c) affected |
| CL | SO | Clear Memory | [EA] ← 0000 |
| CLH | SO | Clear Memory Halfword | [EA] ← 00 |
| CMB | DO | Compare B Reg | B# :: [EA]<br>I(g), I(l) affected<br>I(u) affected but undefined |
| CMH | DO | Compare R Reg With Halfword | R# :: [EA] sign extended<br>I(g), I(l), I(u) affected |
| CMN | SO | Compare With Null | [EA] :: Null Address<br>I(g), I(l) affected<br>I(u) affected but undefined |
| CMR | DO | Compare R Reg | R# :: [EA]<br>I(g), I(l), (u) affected |
| CMV | SI | Compare R Reg With Value | R# :: V sign extended<br>I(g), I(l), I(u) affected |
| CMZ | SO | Compare With Zero | [EA] :: 0000<br>I(g), I(l), I(u) affected |
| CNFG | GE | Configure | See Note |
| CPL | SO | Complement | [EA] ← [EA] ⊕ FFFF |

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| DAL | SHL | Dbl Shift Arith Left | See Note |
| DAR | SHL | Dbl Shift Arith Right | See Note |
| DCL | SHS | Dbl Shift Closed Left | See Note |
| DCR | SHS | Dbl Shift Closed Right | See Note |
| DEC | SO | Decrement | [EA] ← [EA] + FFFF;<br>I(b) ← [EA] (0)<br>I(ov), I(c), I(b) affected |
| DIV | DO | Divide R Reg | R# ← R# / [EA]<br>except if r# = R7,<br>then R7 ← R6, R7 / [EA]<br>and R6 ← remainder<br>I(ov), I(c) affected |
| DOL | SHL | Dbl Shift Open Left | See "Shift Instructions" |
| DOR | SHL | Dbl Shift Open Right | See "Shift Instructions" |
| DOA | GE | Dequeue by Address | See Note |
| DQH | GE | Dequeue from Head | See Note |
| ENT | SO | Enter | P ← EA;S(p) ← 0 |
| HLT | GE | Halt | See Note |
| INC | SO | Increment | I(b) ← [EA] (0);<br>[EA] ← [EA] + 0001<br>I(ov), I(c), I(b) affected |
| IO | I/O | Input/Output (Word) | See Note |
| IOH | I/O | Input/Output Halfword | See Note |
| IOLD | I/O | Input/Output Load | See Note |
| JMP | SO | Jump | P ← EA |
| LAB | DO | Load EA Into B Reg | B# ← EA |
| LB | SO | Load Bit | I(b) ← [EA] i (if indexed)<br>I(b) ← ∨ [EA] m (if masked)<br>I(b) affected |
| LBC | SO | Load Bit and Complement | $\left.\begin{array}{l}\mathrm{I(b)} \leftarrow [\mathrm{EA}]\ \mathrm{i;} \\ [\mathrm{EA}]\ \mathrm{i} \leftarrow \overline{[\mathrm{EA}]\ \mathrm{i}}\end{array}\right\}$ (if indexed)<br>$\left.\begin{array}{l}\mathrm{I(b)} \leftarrow \vee [\mathrm{EA}]\ \mathrm{m;} \\ [\mathrm{EA}]\ \mathrm{m} \leftarrow \overline{[\mathrm{EA}]\ \mathrm{m}}\end{array}\right\}$ (if masked)<br>I(b) affected |
| LBF | SO | Load Bit and Set False | $\left.\begin{array}{l}\mathrm{I(b)} \leftarrow [\mathrm{EA}]\ \mathrm{i} \\ [\mathrm{EA}]\ \mathrm{i} \leftarrow 0\end{array}\right\}$ (if indexed)<br>$\left.\begin{array}{l}\mathrm{I(b)} \leftarrow \vee [\mathrm{EA}]\ \mathrm{m;} \\ [\mathrm{EA}]\mathrm{m} \leftarrow 0\end{array}\right\}$ (if masked)<br>I(b) affected |
| LBS | SO | Load Bit and Swap | I(b) ⟷ [EA] i (if indexed)<br>$\left.\begin{array}{l}\mathrm{Temp} \leftarrow \mathrm{I(b);} \\ \mathrm{I(b)} \leftarrow \vee [\mathrm{EA}]\ \mathrm{m;}\end{array}\right\}$ (if masked)<br>[EA] m ← Temp<br>i(b) affected |

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| LBT | SO | Load Bit and Set True | I(b) ← [EA] i;<br>[EA] i ← 1 } (if indexed)<br>I(b) ← V [EA] m;<br>[EA] m ← 1<br>I(b) affected } (if masked) |
| LDB | DO | Load B Reg | B# ← [EA] |
| LDH | DO | Load Halfword Into R Reg | R# ← [EA] sign extended |
| LDI | SO | Load Doubleword Integer | R6, R7 ← [EA] |
| LDR | DO | Load R Reg | R# ← [EA] |
| LDT | GE | Load T Register | T ← B# |
| LDV | SI | Load Value Into R Reg | R# ← V sign extended |
| LEV | SO | Level Change | See Note |
| LLH | DO | Load Logical Halfword | R#(8:15) ← [EA];<br>R#(0:7) ← 00 |
| LNJ | DO | Load B Reg and Jump | B# ← NSIA;P ← EA |
| LRDB | GE | Load Remote Descriptor Base Register | RDBR ← B3 |
| MCL | GE | Monitor Call Via Trap | Generate trap via Trap Vector #1 |
| MLV | SI | Multiply R Reg by Value | R# ← R# * V<br>except if r# = 7,<br>then R6, R7 ← R7 *V<br>I(ov) affected |
| MMM | GE | Memory to Memory Move | See Note |
| MTM | DO | Modify Test M Reg | See Note |
| MUL | DO | Multiply R Reg | R# ← R# * [EA]<br>except if r# = R7,<br>then R6, R7 ← R7 * [EA]<br>I(ov) affected |
| NEG | SO | Negate | [EA] ← 0000 [EA]<br>I(ov), I(c) affected |
| NOP | BI | No Operation | None |
| OR | DO | OR With R Reg | R# ← R# V[EA] |
| ORH | DO | OR Halfword With R Reg | R# ← R# V[EA] sign extended |
| QOH | GE | Queue on Head | See Note |
| QOT | GE | Queue on Tail | See Note |
| RLQ | GE | Relinquish Stack Space | See Note |
| RSTR | SO | Restore Context | See Note |

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| RTCF | GE | Real Time Clock Off | See Note |
| RTCN | GE | Real Time Clock On | See Note |
| RTT | GE | Return From Trap | See Note |
| SAL | SHS | Sgl Shift Arith Left | See "Shift Instructions" |
| SAR | SHS | Sgl Shift Arith Right | See "Shift Instructions" |
| SAVE | SO | Save Context | See Note |
| SCL | SHS | Sgl Shift Closed Left | See "Shift Instructions" |
| SCR | SHS | Sgl Shift Closed Right | See "Shift Instructions" |
| SDI | SO | Store Doubleword Integer | [EA] ← R6, R7 |
| SID | SO | Subtract Doubleword Integer | R6, R7 ← R6, R7 + [EA] + 1; I(ov), I(c) affected |
| SOL | SHS | Sgl Shift Open Left | See "Shift Instructions" |
| SOR | SHS | Sgl Shift Open Right | See "Shift Instructions" |
| SRDB | GE | Store Remote Descriptor Base Register | B3 ← RDBR |
| SRM | DO | Store Reg Masked | [EA] ← (R#∧m)∨ ([EA]∧$\overline{m}$) |
| STB | DO | Store B Reg | [EA] ← B# |
| STH | DO | Store Halfword From R Reg | [EA] ← R# (8:15) |
| STM | DO | Store M Reg | [EA] (8:15) ← M1<br>[EA] (0:7) ← FF<br>If r# ≠ M1, generate trap via Trap Vector #5 (Model 33 only) |
| STR | DO | Store R Reg | [EA] ← R# |
| STS | SO | Store S Reg | [EA] ← S |
| STT | GE | Store T Reg | B ← T |
| SUB | DO | Subtract From R Reg | R# ← R# ← [EA]<br>I(ov), I(c) affected |
| SWB | DO | Swap B Reg | B# ⟷ [EA] |
| SWR | DO | Swap R Reg | R# ⟷ [EA] |
| VLD | GE | Validate | See Note |
| WDTF | GE | Watchdog Timer Off | See Note |
| WDTN | GE | Watchdog Timer On | See Note |
| XOH | DO | Exclusive OR Halfword With R Reg | R# ← R# ⊕ [EA] sign extended |
| XOR | DO | Exclusive OR With R Reg | R# ← R# ⊕ [EA] |

| | |
|---|---|
| B# | Value store specified B register |
| EA | Effective address |
| [EA] | Value of operand (bit, halfword, word, doubleword) pointed to by the effective address |
| [EA] (0) | Value of bit 0 of operand pointed to by the effective address |
| [EA] (0:7) | Value of bits 0 through 7 of operand pointed to by the effective address |
| [EA] (8:15) | Value of bits 8 through 15 of operand pointed to by the effective address |
| [EA] i | Value of single bit obtained through an indexed address |
| $\overline{[EA]\ i}$ | Complement value of single bit obtained through an indexed address |
| [EA] m | Value of each masked bit in the word pointed to be the effective address |
| $\overline{[EA]\ m}$ | Complement value of each masked bit in the word pointed to by the effective address |
| V[EA] m | Single bit value obtained by an inclusive OR operation on the logical product of (1) the designated mask and (2) the word pointed to be the effective address |
| I(b) | "Bit test" indicator bit of I register |
| I(c) | "Carry" indicator bit of I register |
| I(g) | "Greater than" indicator bit of I register |
| I(i) | "Input/output" indicator bit of I register |
| I(l) | "Less than" indicator bit of I register |
| I(ov) | "Overflow" indicator bit of I register |
| I(u) | "Unlike signs" indicator bit of I register |
| m | Value of mask operand |
| $\overline{m}$ | Complement of mask operand value |
| M# | Specified M register |
| NSIA | Next sequential instruction address |
| P | Program counter (P register) |
| r# | Register number |
| R# | Specified R register |
| R#(0) | Bit 0 of specified R register |
| R#(15) | Bit 15 of specified R register |
| R#(0:7) | Bits 0 through 7 of specified R register |
| R#(1:15) | Bits 1 through 15 of specified R register |
| R#(8:15) | Bits 8 through 15 of specified R register |
| S | S register |
| S(p) | "Privilege state" bits of S register |
| Temp | Temporary storage location for a single bit; the value stored in this temporary storage location |
| T | Stack address register |
| V | Value in bits 8 through 15 of this instruction |

|  |  |
|---|---|
| ⊕ | Exclusive OR |
| V | Inclusive OR |
| Λ | Logical AND |
| / | Division operator |
| * | Multiplication operator |
| ; | Separator for nonsimultaneous operations |
| :: | Is compared with |
| ← | Is replaced by |
| ⟷ | Is exchanged with |

NOTE: Refer to Systems Handbook, CC71

# SECTION 3

## SCIENTIFIC INSTRUCTIONS (ALPHABETICAL)

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| SAD | DO | Scientific Add | [SA#] ← [SA#] + [EA] |
| SBE | BI | Branch if Equal | If [SI (L) V SI (G) = 0, then [P] ← EA |
| SBEU | BI | Branch on Exponent Underflow | If [SI (EUF)] = 1, then [P] ← EA |
| SBEZ | BR | Branch on SA = 0 | If [SA# (f)] = 0, then [P] ← EA |
| SBG | BI | Branch on Greater Than | If [SI (G)] = 1, then [P] ← EA |
| SBGE | BI | Branch on Greater Than or Equal to | If (SI (L)] = 0, then [P] ← EA |
| SBGEZ | BR | Branch on SA ≥ 0 | If [SA# (s)] = 0, or if [SA# (f) = 0, then [P] ← EA |
| SBGZ | BR | Branch on SA > 0 | If [SA# (f)] ≠0 and [SA# (s)] = 0, then [P] ← EA |
| SBL | BI | Branch on Less Than | If [SI (L)] = 1, then [P] ← EA |
| SBLE | BI | Branch on Less Than or Equal to | If [SI (G)] : 0, then [P] ← EA |
| SBLEZ | BR | Branch on SA ≤ 0 | If [SA# (f)] = 0 or if [SA# (s)] = 1, then [P] ← EA |
| SBLZ | BR | Branch on SA < 0 | If [SA# (s)] = 1 and [SA# (f)] ≠ 0 then [P] ← EA |
| SBNE | BI | Branch on Not Equal | If [SI (L) V SI (G)] = 1, then [P] ← EA |
| SBNEU | BI | Branch on No Exponent Underflow | If [SI (EUF)] = 0, then [P] ← EA |
| SBNEZ | BR | Branch on SA ≠ 0 | If [SA# (f)] ≠ 0, then [P] ← EA |
| SBNPE | BI | Branch on No Precision Error | If [SI (PE)] = 0, then [P] ← EA |

| Mnemonic | Type | Description | Operation |
|---|---|---|---|
| SBNSE | BI | Branch on No Significance Error | If [SI (SE)] = 0, then [P] ← EA |
| SBPE | BI | Branch on Precision Error | If [SI (PE)] = 1, then [P] ← EA |
| SBSE | BI | Branch on Significance Error | If [SI (SE)] = 1, then [P] ← EA |
| SCM | DO | Scientific Compare | SI (G), SI (L) ← [SA#] :: [EA] |
| SCZD | SO | Scientific Compare to Zero — 2 words | SI (G), SI (L) ← [EA] :: Zero |
| SCZQ | SO | Scientific Compare to Zero — 4 words | SI (G), SI (L) ← [EA] :: Zero |
| SDV | DO | Scientific Divide | [SA#] ← [SA#] / [EA] |
| SLD | DO | Scientific Load | [SA#] ← [EA] |
| SML | DO | Scientific Multiply | [SA#] ← [SA#] * [EA] |
| SNGD | SO | Scientific Negate — 2 words | [EA (s)] ← $\overline{[\text{EA (s)}]}$ |
| SNGQ | SO | Scientific Negate | [EA (s)] ← [EA (s)] |
| SSB | SO | Scientific Subtract | [SA#] ← [SA#] − [EA] |
| SST | DO | Scientific Store | [EA] ← [SA#] |
| SSW | DO | Scientific Swap | [SA#] ⟷ [EA] |

where f = Fraction (mantissa) excluding sign
SA = Scientific Accumulator

NOTE:

These instructions are software simulated except on Models 43, 47, 53, and 57 with the SIP option.

# SECTION 4

## COMMERCIAL INSTRUCTIONS (ALPHABETICAL)

| Mnemonic[1] | Type | Description | Operation |
|---|---|---|---|
| ACM | A | Alphanumeric Compare | [DD1] :: [DD2] → CI(G,L) |
| ALR | A | Alphanumeric Move | [DD1] → [DD2] |
| AME | E | Alphanumeric Move and Edit | [DD1] edited → [DD2]; [DD3] specifies Micro-ops |
| CBD | N | Convert Binary to Decimal | [DD1] converted → [DD2] |
| CBE | B | Branch if Equal | If CI (G and L) = 0, then [P] ← EA |
| CBG | B | Branch if Greater | If CI (G) = 1, then [P] ← EA |
| CBGE | B | Branch if Greater Than or Equal | If CI (L) = 0, then [P] ← EA |
| CBL | B | Branch if Less | If CI (L) = 1, then [P] ← EA |
| CBLE | B | Branch if Less Than or Equal | If CI (G) = 0, then [P] ← EA |
| CBNE | B | Branch if Not Equal | If CI (G or L) = 1, then [P] ← EA |
| CBNOV | B | Branch if No Overflow | If CI (OV) = 0, then [P] ← EA |
| CBNSF | B | Branch if No Sign Fault | If CI (SF) = 0, then [P] ← EA |
| CBNTR | B | Branch on No Truncation | If CI (TR) = 0, then [P] ← EA |
| CBOV | B | Branch on Overflow | If CI (OV) = 1, then [P] ← EA |
| CBSF | B | Branch on Sign Fault | If CI (SF) = 1, then [P] ← EA |
| CBTR | B | Branch on Truncation | If CI (TR) = 1, then [P] ← EA |
| CDB | N | Convert Decimal to Binary | [DD1] converted → [DD2] |
| CSNCB | B | Synchronize and Branch | Prevents CP from going to next instruction until previous commercial instruction is completed; then performs unconditional branch |
| CSYNC | B | Synchronize | Prevents CP from going to next instruction until previous commercial instruction is completed |
| DAD | N | Decimal Add | [DD2] + [DD1] → [DD2] |
| DCM | N | Decimal Compare | [DD1] :: [DD2] → IND |
| DDV | N | Decimal Divide | [DD2]/[DD1] → [DD3]; R → [DD2] |
| DLS | E | Decimal Left Shift | Shift [DD1] left "d" positions[2] |
| DMC | N | Decimal Move and Convert | [DD1] converted → [DD2] |
| DME | E | Decimal Move and Edit | [DD1] Edited → [DD2]; [DD3] specifies Micro-ops |
| DML | N | Decimal Multiply | [DD2] * [DD1] → [DD2] |
| DRS | E | Decimal Right Shift | Shift [DD1] right "d" positions[2] |

| Mnemonic[1] | Type | Description | Operation |
|---|---|---|---|
| DSB | N | Decimal Subtract | [DD2] – [DD1] → [DD2] |
| DSH | N | Decimal Shift | Shift [DD1] left "d" positions<br>Shift [DD1] right "d" positions |
| MAT | A | Alphanumeric Move and Translate | [DD1] translate → [DD2]; [DD3] specifies 256-byte Translate Table |
| SRCH[3] | A | Alphanumeric Search | [DD3] is searched using [DD1] and [DD2] for some purposes |
| VRFY[3] | A | Alphanumeric Verify | [DD3] is verified using [DD1] and [DD2] for some purposes |

where:

DD1, 2 and 3 = Data Description 1, 2, or 3
N = Numeric Type
A = Alphanumeric Type
E = Edit Type
B = Branch Type

[1] These instructions are software simulated except on Models 47 and 57.

[2] DLS and DRS are software instructions that generate the DSH code and the appropriate value of the shift control word. Refer to footnote 1 on page 5-2, and to the *Level 6 Assembler Manual,* Order No. CB07, for a complete description.

[3] Requires Model 47 or Model 57.

## SECTION 5

## INSTRUCTIONS
## (NUMERICAL)

| H1 | H2 | H3 | H4 | Mnemonic |
|---|---|---|---|---|
| Generic (GE) | | | | |
| 0 | 0 | 0 | 0 | HLT |
| 0 | 0 | 0 | 1 | MCL |
| 0 | 0 | 0 | 2 | BRK |
| 0 | 0 | 0 | 3 | RTT |
| 0 | 0 | 0 | 4 | RTCN |
| 0 | 0 | 0 | 5 | RTCF |
| 0 | 0 | 0 | 6 | WDTN |
| 0 | 0 | 0 | 7 | WDTF |
| 0 | 0 | 0 | 8 | MMM |
| 0 | 0 | 0 | A | ASD |
| 0 | 0 | 0 | B | VLD |
| 0 | 0 | 0 | C | LRDB |
| 0 | 0 | 0 | D | SRDB |
| 0 | 0 | 1 | 0 | LDT* |
| 0 | 0 | 1 | 0 | STT* |
| 0 | 0 | 1 | 0 | ACQ* |
| 0 | 0 | 1 | 0 | RLQ* |
| 0 | 0 | 1 | 1 | CNFG |
| 0 | 0 | 6 | 0 | DQA |
| 0 | 0 | 6 | 1 | QOT |
| 0 | 0 | 6 | 2 | DQH |
| 0 | 0 | 6 | 3 | QOH |

*Function determined by second word of instruction. See Systems Handbook, CC71

| H1 | H2 | H3 | H4 | Mnemonic |
|---|---|---|---|---|
| Branch on Indicators (BI) | | | | |
| 0 | 2 | 0+x | – | BL |
| 0 | 2 | 8+x | – | BGE |
| 0 | 3 | 0+x | – | BG |
| 0 | 3 | 8+x | – | BLE |
| 0 | 4 | 0+x | – | BOV |
| 0 | 4 | 8+x | – | BNOV |
| 0 | 5 | 0+x | – | BBT |
| 0 | 5 | 8+x | – | BBF |
| 0 | 6 | 0+x | – | BCT |
| 0 | 6 | 8+x | – | BCF |
| 0 | 7 | 0+x | – | BIOT |
| 0 | 7 | 8+x | – | BIOF |
| 0 | 8 | 0+x | – | BAL |
| 0 | 8 | 8+x | – | BAGE |
| 0 | 9 | 0+x | – | BE |
| 0 | 9 | 8+x | – | BNE |
| 0 | A | 0+x | – | BAG |
| 0 | A | 8+x | – | BALE |
| 0 | B | 0+x | – | BSU |
| 0 | B | 8+x | – | BSE |
| 0 | F | 0+x | – | NOP |
| 0 | F | 8+x | – | B |

**Commercial Branches**

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 0+x | – | CBOV |
| 1 | 3 | 8+x | – | CBNOV |
| 2 | 3 | 0+x | – | CBTR |
| 2 | 3 | 8+x | – | CBNTR |
| 3 | 3 | 0+x | – | CBSF |
| 3 | 3 | 8+x | – | CBNSF |
| 4 | 3 | 0+x | – | CSYNC |
| 4 | 3 | 8+x | – | CSNCB |
| 5 | 3 | 0+x | – | CBNE |
| 5 | 3 | 8+x | – | CBE |
| 6 | 3 | 0+x | – | CBG |
| 6 | 3 | 8+x | – | CBLE |
| 7 | 3 | 0+x | – | CBL |
| 7 | 3 | 8+x | – | CBGE |

**Scientific Branches**

| | | | | |
|---|---|---|---|---|
| 0+r | 4 | 0+x | – | SBLZ |
| 4 | 4 | 0+x | – | SBL |
| 5 | 4 | 0+x | – | SBPE |
| 6 | 4 | 0+x | – | SBSE |
| 7 | 4 | 0+x | – | SBEU |
| 0+r | 4 | 8+x | – | SBGEZ |
| 4 | 4 | 8+x | – | SBGE |
| 5 | 4 | 8+x | – | SBNPE |
| 6 | 4 | 8+x | – | SBNSE |
| 7 | 4 | 8+x | – | SBNEU |
| 0+r | 5 | 0+x | – | SBEZ |
| 4 | 5 | 0+x | – | SBE |
| 0+r | 5 | 8+x | – | SBNEZ |
| 4 | 5 | 8+x | – | SBNE |
| 0+r | 6 | 0+x | – | SBGZ |
| 4 | 6 | 0+x | – | SBG |
| 0+r | 6 | 8+x | – | SBLEZ |
| 4 | 6 | 8+x | – | SBLE |

**Branch on Registers (BR)**

| H1 | H2 | H3 | H4 | Mnemonic |
|---|---|---|---|---|
| 0+r | 7 | 0+x | – | BDEC |
| 0+r | 7 | 8+x | – | BINC |
| 0+r | 8 | 0+x | – | BLZ |
| 0+r | 8 | 8+x | – | BGEZ |
| 0+r | 9 | 0+x | – | BEZ |
| 0+r | 9 | 8+x | – | BNEZ |
| 0+r | A | 0+x | – | BGZ |
| 0+r | A | 8+x | – | BLEZ |
| 0+r | B | 0+x | – | BEVN |
| 0+r | B | 8+x | – | BODD |

**Short Value Immediate (SI)**

| | | | | |
|---|---|---|---|---|
| 0+r | C | – | – | LDV |
| 0+r | D | – | – | CMV |
| 0+r | E | – | – | ADV |
| 0+r | F | – | – | MLV |

**Input/Output (I/O)**

| | | | | |
|---|---|---|---|---|
| 8 | 0 | 0+x | – | IO |
| 8 | 1 | 0+x | – | IOH |
| 8 | 1 | 8+x | – | IOLD |

**Single Operand (SO)**

| | | | | |
|---|---|---|---|---|
| 8 | 2 | 0+x | – | NEG |
| 8 | 2 | 8+x | – | LB |
| 8 | 3 | 8+x | – | JMP |
| 8 | 4 | 0+x | – | AID |
| 8 | 4 | 8+x | – | SID |
| 8 | 6 | 0+x | – | CPL |
| 8 | 7 | 0+x | – | CL |
| 8 | 7 | 8+x | – | CLH |
| 8 | 8 | 0+x | – | LBF |
| 8 | 8 | 8+x | – | DEC |
| 8 | 9 | 0+x | – | LBT |
| 8 | 9 | 8+x | – | CMZ |
| 8 | A | 0+x | – | LBS |
| 8 | A | 8+x | – | INC |
| 8 | B | 0+x | – | LBC |
| 8 | B | 8+x | – | ENT |
| 8 | C | 0+x | – | STS |
| 8 | C | 8+x | – | LDI |
| 8 | D | 0+x | – | SDI |
| 8 | D | 8+x | – | CMN |
| 8 | E | 0+x | – | LEV |
| 8 | E | 8+x | – | CAD |
| 8 | F | 0+x | – | SAVE |
| 8 | F | 8+x | – | RSTR |

**Double Operand (DO)**

| H1 | H2 | H3 | H4 | Mnemonic |
|---|---|---|---|---|
| 8+r | 0 | 0+x | – | MTM |
| 8+r | 0 | 8+x | – | LDH |
| 8+r | 1 | 8+x | – | CMH |
| 8+r | 2 | 0+x | – | SUB |
| 8+r | 2 | 8+x | – | LLH |
| 8+r | 3 | 0+x | – | DIV |
| 8+r | 3 | 8+x | – | LNJ |
| 8+r | 4 | 0+x | – | OR |
| 8+r | 4 | 8+x | – | ORH |
| 8+r | 5 | 0+x | – | AND |
| 8+r | 5 | 8+x | – | ANH |
| 8+r | 6 | 0+x | – | XOR |
| 8+r | 6 | 8+x | – | XOH |
| 8+r | 7 | 0+x | – | STM |
| 8+r | 7 | 8+x | – | STH |
| 8+r | 8 | 0+x | – | LDR |
| 8+r | 9 | 0+x | – | CMR |
| 8+r | A | 0+x | – | ADD |
| 8+r | A | 8+x | – | SRM |
| 8+r | B | 0+x | – | MUL |
| 8+r | B | 8+x | – | LAB |
| 8+r | C | 8+x | – | LDB |
| 8+r | D | 8+x | – | CMB |
| 8+r | E | 0+x | – | SWR |
| 8+r | E | 8+x | – | SWB |
| 8+r | F | 0+x | – | STR |
| 8+r | F | 8+x | – | STB |

**Scientific**

| | | | | |
|---|---|---|---|---|
| C | 8 | 8+x | – | SCZD |
| C+r | 8 | 8+x | – | SCM |
| 8+r | 8 | 8+x | – | SLD |
| C | 9 | 8+x | – | SNGD |
| C+r | 9 | 8+x | – | SSB |
| 8+r | 9 | 8+x | – | SAD |
| C | C | 0+x | – | SCZQ |
| C+r | C | 0+x | – | SDV |
| 8+r | C | 0+x | – | SML |
| C | D | 0+x | – | SNGQ |
| C+r | D | 0+x | – | SSW |
| 8+r | D | 0+x | – | SST |

| H1 | H2 | H3 | H4 | | Mnemonic |
|---|---|---|---|---|---|
| Commercial | | | | | |
| 0 | 0 | 2 | 0 | — | VRFY |
| 0 | 0 | 2 | 1 | — | ALR |
| 0 | 0 | 2 | 2 | — | ACM |
| 0 | 0 | 2 | 3 | — | MAT |
| 0 | 0 | 2 | 4 | — | AME |
| 0 | 0 | 2 | 5 | — | DMC |
| 0 | 0 | 2 | 6 | — | DME |
| 0 | 0 | 2 | 7 | — | CBD |
| 0 | 0 | 2 | 8 | — | SRCH |
| 0 | 0 | 2 | 9 | — | DML |
| 0 | 0 | 2 | A | — | CDB |
| 0 | 0 | 2 | B | — | DDV |
| 0 | 0 | 2 | C | — | DAD |
| 0 | 0 | 2 | D | — | DSB |
| 0 | 0 | 2 | E | — | DSH[1] |
| 0 | 0 | 2 | F | — | DCM |

where:

r = Register number contained in bits 1 through 3 or 2 through 3

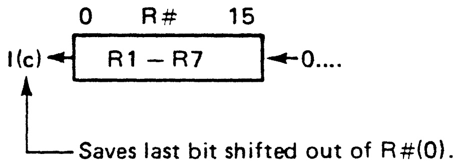
x = Value of bits 9 through 11

[1] For the DSH instruction, shift control direction is specified by the internal shift control word 2 (SCW2) bit 0. If SCW2 bit 0 = 0, shift is left; if = 1, shift is right. For the instruction format in which direction is specified, refer to the *Level 6 Assembler Manual,* Order No. CB07. Two additional software instructions are available: DLS (Decimal Left Shift) and DRS (Decimal Right Shift) and are also discussed in the same manual. When these instructions are used, the internal code of the DSH instruction (002E) is generated along with the appropriate value in the SCW2.

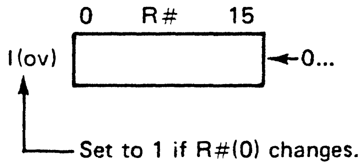
## SECTION 6

## SHIFT INSTRUCTIONS

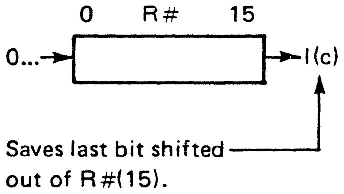
SOL
r00#



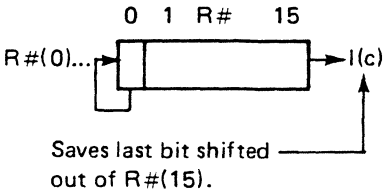
SAL
r02#



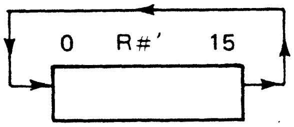
SOR
r04#



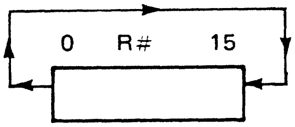
SAR
r06#



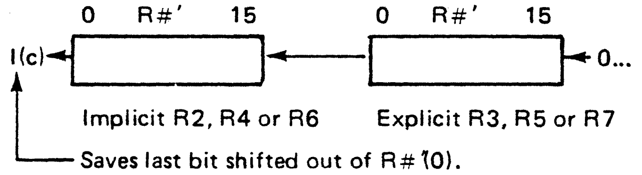
SCR
r05 #



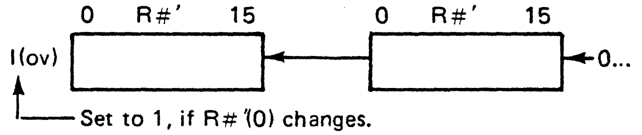
SCL
r01 #



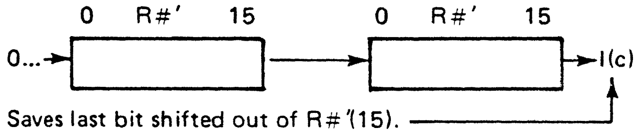
DOL
r08#



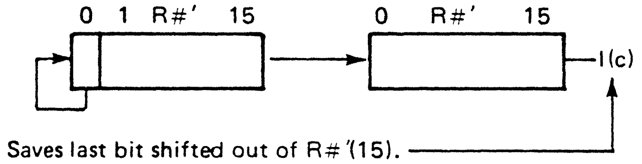
DAL
r0A#



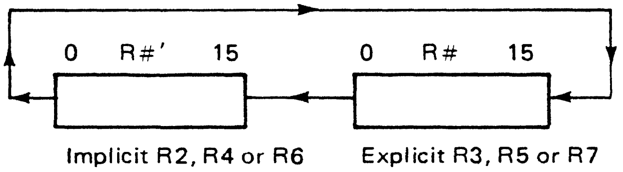
DOR
r0C#



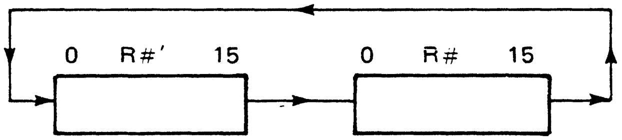
DAR
r0E#



DCL
r03#



DCR
r07#

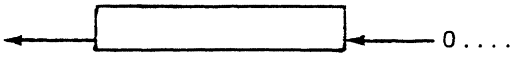


### SHIFT INSTRUCTIONS

| | |
|---|---|
| I(c) | c bit of 1 reg. contains most recent bit discarded |
| I(ov) | ov bit of 1 reg. set if sign changed, else cleared |
| r | R1 thru R7, or, R3, R5 or R7 |
| # | shift distance 1-15 or 1-31, 0 is special case |

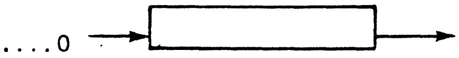
**Commercial Shift Instructions**

(DSH)
DLS



Set ov if a non-zero character
is shifted out.

(DSH)
DRS



Round least significant digit if rounding
is specified.

# SECTION 7

# ADDRESS SYLLABLE

### Address Syllable Definitions

| | |
|---|---|
| Bk | B register number 1, 2, or 3 |
| Bn | B register number n |
| DSP | 16-bit displacement (–32,768 through +32,767) that follows the word containing the address syllable – for P+DSP and *(P+DSP), DSP is added to the address of the word containing DSP |
| IMA | Immediate address |
| IMO | Immediate operand |
| IV | Interrupt vector |
| P | Location of word containing displacement |
| Rn | R register number n |
| * | Indirect operator |
| ↑ | Increment symbol; incrementation occurs after the effective address is obtained but before execution of the instruction |
| ↓ | Decrement symbol; decrementation occurs before the effective address is obtained |

### Commercial Address Syllable

| 9 11 | 12 | 13 15 |
|---|---|---|
| m | @ | n |

m – Address modifier

@ – Direct/indirect address indicator (when m = 0 through 4); otherwise, secondary address modifier

n – Register number (when 1 through 7)

| | n = 0 | | n = 1-7 | |
|---|---|---|---|---|
| m | @ = 0 | @ = 1 | @ = 0 | @ = 1 |
| 0 | – | P+DSP | Bn+DSP | *[Bn+DSP] |
| 1 | – | P+DSP+R1 | Bn+DSP+R1 | *[Bn+DSP]+R1 |
| 2 | Use Remote Data Descriptors | P+DSP+R2 | Bn+DSP+R2 | *[Bn+DSP]+R2 |
| 3 | | P+DSP+R3 | Bn+DSP+R3 | *[Bn+DSP]+R3 |
| 4 | | *[P+DSP] | Bn+DSP+R4 | *[Bn+DSP]+R4 |
| 5 | | RFU | Bn+DSP+R5 | *[Bn+DSP]+R5 |
| 6 | | RFU | Bn+DSP+R6 | *[Bn+DSP]+R6 |
| 7 | – | IMO | Bn+DSP+R7 | *[Bn+DSP]+R7 |

**Non-Commercial Instructions**

| 9 11 | 12 | 13 15 |
|---|---|---|
| m | @ | n |

m — Address modifier
@ — Direct/indirect address indicator (when m = 0 through 4)
Otherwise, secondary address modifier
n — Register number (when 1 through 7)

In the table below, the underlined items indicate the nature of the address obtained by the corresponding values of the address syllable. Indented below each item is the format of the related source language.

| m | n = 0 | | n = 1 – 7 | | | |
|---|---|---|---|---|---|---|
| | @ = 0 | @ = 1 | @ = 0 | @ = 1 | | |
| 0 | IMA<br>&lt;label | *IMA<br>*&lt;label | Bn<br>$Bn | *Bn<br>*$Bn | | |
| 1 | IMA+R1<br>&lt;label.$R1 | *IMA+R1<br>*&lt;label.$R1 | Bn+R1<br>$Bn.$R1 | *Bn+R1<br>*$Bn.$R1 | | |
| 2 | IMA+R2<br>&lt;label.$R2 | *IMA+R2<br>*&lt;label.$R2 | Bn+R2<br>$Bn.$R2 | * Bn+R2<br>*$Bn.$R2 | | |
| 3 | IMA+R3<br>&lt;label.$R3 | *IMA+R3<br>*&lt;label.$R3 | Bn+R3<br>$Bn.$R3 | *Bn+R3<br>*$Bn.$R3 | | |
| 4 | P+DSP<br>label | *(P+DSP)<br>*label | Bn+DSP<br>$Bn.value | *[Bn+DSP]<br>$Bn.value | | |
| | | | | n=1,2,3 | n=4 | n=5,6,7 |
| 5 | reserved | reserved | Register<br>=$Bn or =$Rn | Bn+↓R1<br>$Bk.–$R1 | reserved | B[n–4]+R1↑<br>$Bk.+$R1 |
| 6 | reserved | reserved | ↓Bn "Push"<br>–$Bn | Bn+↓R2<br>$Bk.–$R2 | | B[n–4]+R2↑<br>$Bk.+$R2 |
| 7 | IMO<br>=value<br>or<br>=label | IV+DSP | Bn↑ "pop"<br>+$Bn | Bn+↓R3<br>$Bk.–$R3 | | B[n--4]+R3↑<br>$Bk.+$R3 |

# SECTION 8

## HARDWARE-DEDICATED MEMORY

| SAF Memory Location | Hardware Nomenclature | LAF Memory Location** | Contents |
|---|---|---|---|
| 0000 | RHU/RSU | 0000 | – Entry to Power Failure Restart Routine |
| 000A | NATSAP3 | | Next available |
| 000C | NATSAP2 | | TSA pointers |
| 000E | NATSAP1 | | |
| 0010 | NATSAP0 | 0010 | |
| | RHU | | |
| 0014 | RTCI | 0014 | – RTC Initial Value |
| 0015 | RTCC | 0015 | – RTC Current Value |
| 0016 | RTCL | 0016 | – RTC Level |
| 0017 | WDTC | 0017 | – WDT Current Value |
| | RHU | | |
| 001F | MERC | 001F | – Memory Error Count |
| 0020 | 0 15 | 0020 | – Interrupt Level Activity Flags |
| 0021 | 16 IAF 31 | 0021 | |
| 0022 | 32 47 | 0022 | |
| 0023 | 48 63 | 0023 | |
| | RHU | | Associated Event |
| 0052 | TV =46 | 0024 | – RFU |
| | • • • | | |

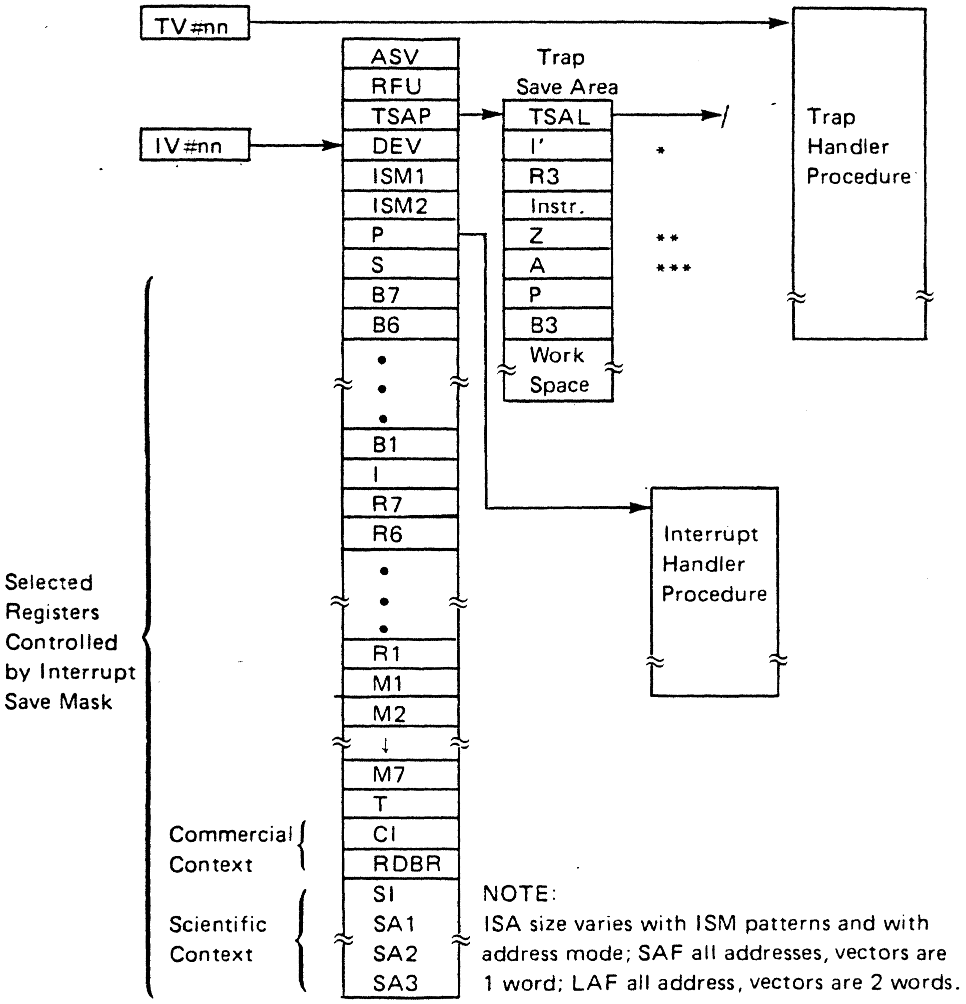
| | | | |
|---|---|---|---|
| 0061 | TV=31 | 0042 – | SIP QLT FAULT |
| 0062 | TV=30 | 0044 – | CIP QLT FAULT |
| 0063 | TV=29 | 0046 – | Commercial Overflow* |
| 0064 | TV=28 | 0048 – | Commercial Truncation* |
| 0065 | TV=27 | 004A – | Commercial Illegal Character |
| 0066 | TV=26 | 004C – | Commercial Illegal Specification |
| 0067 | TV=25 | 004E – | Commercial Divide by Zero |
| 0068 | TV=24 | 0050 – | Memory or Megabus Error seen by SIP or CIP |
| 0069 | TV=23 | 0052 – | Unavailable Resource referenced by SIP or CIP |
| 006A | TV=22 | 0054 – | Scientific Precision Error* |
| 006B | TV=21 | 0056 – | Scientific Significance Error* |
| 006C | TV=20 | 0058 – | Scientific Program Error |
| 006D | TV=19 | 005A – | Scientific Exponent Underflow* |
| 006E | TV=18 | 005C – | RFU |
| 006F | TV=17 | 005E – | Uncorrectable Memory or Megabus Error |
| 0070 | TV=16 | 0060 – | Program Error |
| 0071 | TV=15 | 0062 – | Unavailable Resource |
| 0072 | TV=14 | 0064 – | Protection Violation |
| 0073 | TV=13 | 0066 – | Privilege Violation |
| 0074 | TV=12 | 0068 – | Remote, remote data descriptor |
| 0075 | TV=11 | 006A – | RFU |
| 0076 | TV=10 | 006C – | Stack Overflow |
| 0077 | TV=9 | 006E – | Stack Underflow |
| 0078 | TV=8 | 0070 – | Scientific Exponent Overflow |
| 0079 | TV=7 | 0072 – | Scientific Divide by Zero |
| 007A | TV=6 | 0074 – | Integer Register Overflow* |
| 007B | TV=5 | 0076 – | Unimplemented Instruction |
| 007C | TV=4 | 0078 – | RSU |
| 007D | TV=3 | 007A – | Uninstalled Scientific Instruction |
| 007E | TV=2 | 007C – | Trace*/Breakpoint Trap |
| 007F | TV=1 | 007E – | Monitor Call Trap |
| 0080 | IV=0 | 0080 – | Interrupt Vector, Level 0 |
| 0081 | IV=1 | 0082 – | Interrupt Vector, Level 1 |
| | ⋮ | | |
| 00BE | IV=62 | 00FC – | Interrupt Vector, Level 62 |
| 00BF | IV=63 | 00FE – | Interrupt Vector, Level 63 |
| 00C0-00FF | RHU | NONE | |

*Maskable Trap Conditions

**All LAF addresses and vectors are contained in two memory words.

# SECTION 9

## TRAP VECTOR AND INTERRUPT VECTOR LINKAGE



NOTE:
ISA size varies with ISM patterns and with address mode; SAF all addresses, vectors are 1 word; LAF all address, vectors are 2 words.

*I' Format

| TRAP# | I |
|---|---|

Trap# = $40_{16}$ – TV#
I = Copy of I Register

**Z Format

| 0 | 1 3 | 4 7 | 8 9 | 10 11 | 12 15 |
|---|---|---|---|---|---|
| REG | 000 | BI | R | NTO | IS |

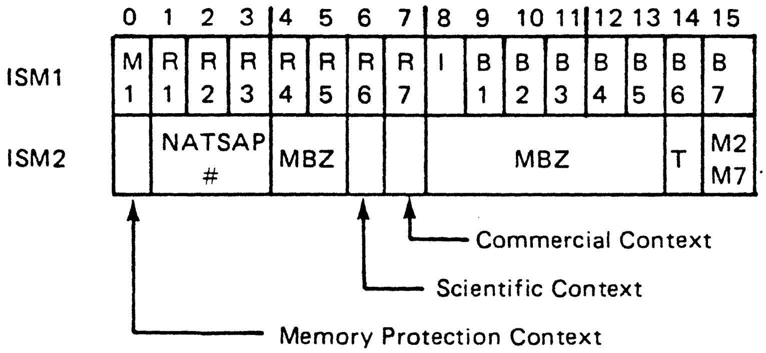
REG – 'A' word validity
1 = A word is invalid
0 = A word is valid

BI – bit/byte index field
For bit instruction, BI = 4 low-order bits of index register
For byte instruction, BI = X000 where X is the low-order bit of the index register

R – Ring Number

IS – Instruction Size

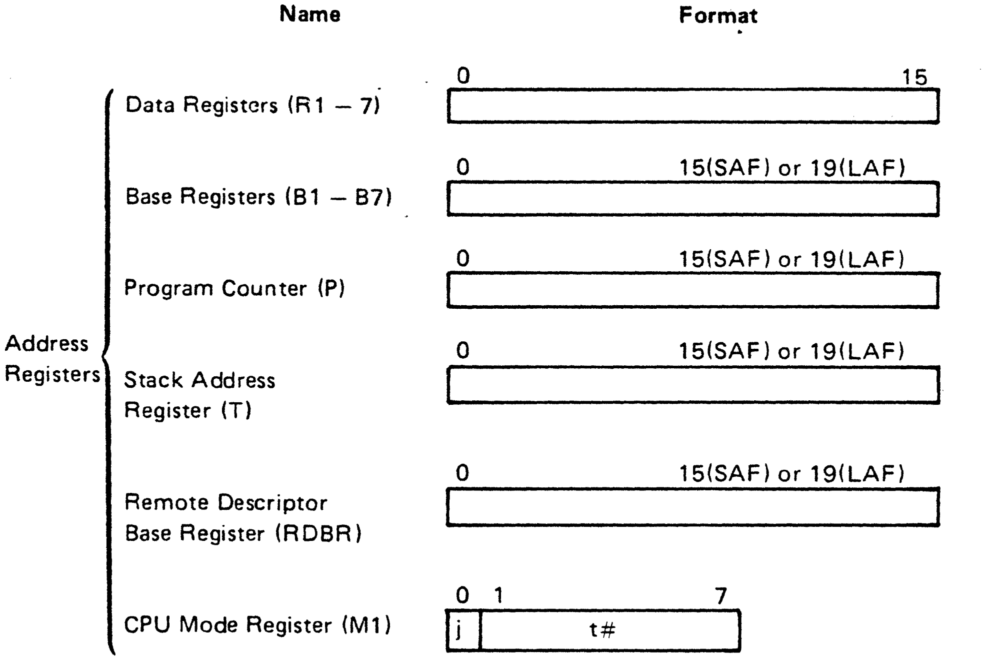
***A Format – address associated with Trap, see Systems Handbook, CC71



|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISM1 | M1 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | I | B1 | B2 | B3 | B4 | B5 | B6 | B7 |
| ISM2 |  | NATSAP # |  |  | MBZ |  |  |  | MBZ |  |  |  |  |  | T | M2<br>M7 |

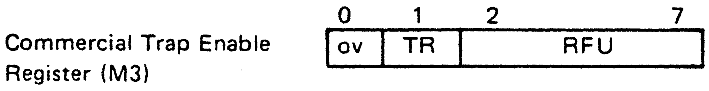
# SECTION 10

# REGISTERS



j – Trace trap enable for jumps and branches
0 = trace trap disabled; 1 = enabled

t# – Overflow trap enable controls for registers
R1 – R7, respectively
0 = trap disabled; 1 = enabled



ov – Overflow Trap Enable
TR – Truncation Trap Enable

SIP Operating Mode
Register (M4)

| 0 | 1 | 2 3 | 4 5 | 6 7 |
|---|---|---|---|---|
| R/T | RFU | SA #1 | SA #2 | SA #3 |

R/T – Round/Truncate Mode
0 = Truncate; 1 = Round

SA#– Scientific accumulator number

|  | Memory data length (words) | Accumulator data length (words) |
|---|---|---|
| 00 | 2 | 2 |
| 01 | 2 | 4 |
| 10 | 4 | 2 |
| 11 | 4 | 4 |

SIP Trap Enable
Register M5

| 0 | 1 | 2 | 3 | 4 7 |
|---|---|---|---|---|
| EUM | RFU | SEM | PEM | RFU |

EUM – Exponent Underflow Trap Enable
SEM – Significance Error Trap Enable
PEM – Precision Error Trap Enable

System Status/Security
Register (S)

| 0 | 1 2 | 3 5 | 6 9 | 10 15 |
|---|---|---|---|---|
| Q | R | 0 0 0 | id# | level |

R – Ring Number (privilege state)
IX = privilege rings; OX = user rings

id# – Processor identity (channel number)

level – Interrupt priority level – 0 (high)
through 63 (low)

Q – QLT Fault

CPU Indicator Register (I)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ov | 0 | c | b | i | g | l | u |

ov — "Overflow" indicator

c — "Carry" indicator

b — "Bit test" indicator

i — "Input/output" indicator

g — "Greater than" indicator

l — "Less than" indicator

u — "Unlike signs" indicator

Commercial Indicator Register (CI)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ov | TR | SF | RFU | | G | L | RFU |

ov — Overflow occurred during decimal instruction

TR — Alphanumeric result is truncated

SF — Sign fault (negative operand is stored in unsigned field)

G — Greater than

L — Less than

SIP Indicator Register (SI)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| EU | RFU | SE | PE | RFU | SG | SL | RFU |

EU — Exponent underflow

SE — Significance error

PE — Precision error

SG — Greater than

SL — Less than