III

FIRMWARE OPERATION

This section provides a description of the Central Processor Unit (CPU) firmware, which consists of 2,048 64-bit words.

3.1 FIRMWARE WORD

The firmware word is illustrated in Figure 3-1. The firmware word is divided into 15 distinct fields: (1) LS, (2) RS, (3) DI, (4) AD, (5) AF, (6) AS, (7) CK, (8) BI, (9) SM, (10) BS, (11) GP, (12) TC, (13) BR, (14) C, and (15) NA. Each of these field controls a portion of the hardware (refer to subsections 3.1.1. through 3.1.15).

3.1.1 Left Select (LS) Field

The LS field consists of bits 1 through 3 of the firmware word. This field serves a dual purpose. Along with the Select Modify (SM) field, it provides a 4-bit address to both the RAM and the RALU left select inputs. The SM field is described in subsection 3.1.9. Since the LS field is only three bits long and since four bits are required to fully address either the RAM or the RALU, one bit must be created (see Table 3-1). The two-weight bit, which is not in control store, is created from the presence of either the four-weight or one-weight bits.

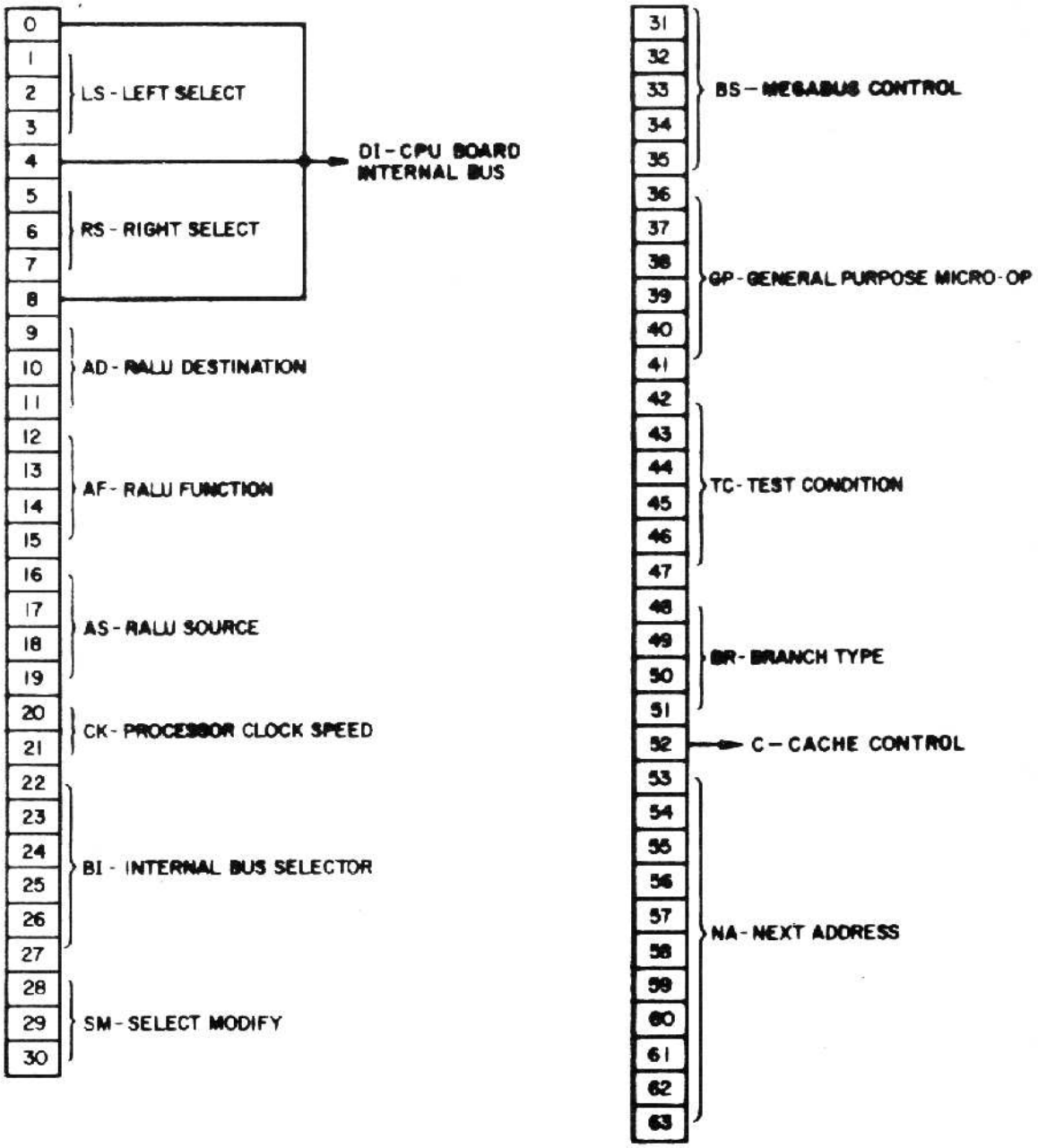


Figure 3-1 Firmware Word Format

Table 3-1 Left Select Codes

LS CODE	GENERATED ADDRESS 8	4	2	1	FIRMWARE NAME	REGISTER SELECTED
0	0	0	0	0	LSD0	D0
1	0	0	1	1	LSD3	D3
2	0	1	1	0	LSD6	D6
3	0	1	1	1	LSD7	D7
4	1	0	0	0	LSB0	B0
5	1	0	1	1	LSB3	B3
6	1	1	1	0	LSB6	B6
7	1	1	1	1	LSB7	B7

3.1.2 Right Select (RS) Field

The RS field consists of bits 5 through 7 of the firmware word. Only three bits reside in control store, thereby requiring that the two-weight bit again be created as in the LS field. The RS field provides a 4-bit address to the RALU right select inputs to transfer an operand to the right output of the reigster file. If data are to be written into the register file, RS selects the location into which the new data are to be loaded. As in the LS field, SM is utilized in determining the address delivered to the RALU right select inputs (see Table 3-2).

Table 3-2 Right Select Codes

RS CODE	GENERATED ADDRESS 8	4	2	1	FIRMWARE NAME	REGISTER SELECTED
0	0	0	0	0	RSD0	D0
1	0	0	1	1	RSD3	D3
2	0	1	1	0	RSD6	D6
3	0	1	1	1	RSD7	D7
4	1	0	0	0	RSB0	B0
5	1	0	1	1	RSB3	B3
6	1	1	1	0	RSB6	B6
7	1	1	1	1	RSB7	B7

3.1.3 Central Processor Board, Internal Bus (DI) Field

This 3-bit field (bits 0, 4, and 8) determines: (1) which portion (if any) of the RALU will be placed onto the internal bus, (2) whether the RAM will be transferred to the internal bus, and (3) whether the data currently on the internal bus will be written into the RAM (see Table 3-3).

Table 3-3 Central Processor Board Internal Bus Codes

DI CODE	FIRMWARE NAME	OPERATION PERFORMED
0	BIB*	Transfers RALU bits 0C through 0F to internal bus bits 0C through 0F.
1	DIA	Transfers RALU bits 0C through 1F to internal bus bits 0C through 1F.
2	DIC	Loads selected RAM location bits 0C through 1F with output of RALU bits 0C through 1F via the internal bus.
3	-	RFU
4	DIN	Null (neither RAM nor RALU is sent to the internal bus and the RAM is not written).
5	BIR*	Transfers RALU bits 18 through 1F to internal bus bits 18 through 1F.
6	DIW	Loads selected RAM location (bits 0C through 1F) with contents of internal bus (bits 0C through 1F).
7	DIR	Transfers RAM location contents (bits 0C through 1F) to internal bus (bits 0C through 1F).

*Refer to Table 3-10.

3.1.4 RALU Destination (AD) Field

This 3-bit field (bits 9 through 11) determines whether the RALU output will be shifted right, left, or not at all. The AD field also controls whether this value will be written into the Q register or the register file or neither, and whether the register file left output or the ALU output is made available to the internal bus (see Table 3-4).

3.1.5 RALU Function (AF) Field

This 4-bit field (bits 12 through 15) controls the type of operation that will be performed on the two operands, J and K. Fifteen different functions (see Table 3-5) may be performed (e.g., ADD, OR, AND, etc.). The most significant bit of this field controls the ALU input carry.

Table 3-4 RALU Destination Codes

AD CODE	FIRMWARE NAME	OPERATION PERFORMED
0	ADFQ	Sends the output of the ALU to the output multiplexer and writes the same value into the Q register.
1	ADFN	Sends the output of the ALU to the output multiplexer.
2	ADLR	Sends the contents of the left latch to the output multiplexer and writes the output of the ALU into the register file.
3	ADFR	Sends the output of the ALU to the output multiplexer and writes the same value into the register file.
4	ADDR	Sends the output of the ALU to the output multiplexer; performs a 32-bit shift right of the ALU output and the Q register contents, writing the shifted results to the register file and to the Q register.
5	ADSR	Sends the output of ALU to the output multiplexer; performs a 16-bit shift right of the ALU output, writing the shifted result to the register file.
6	ADDL	Sends the output of the ALU to the output multiplexer; performs a 32-bit shift left of the ALU output and the Q register contents, writing the shifted results to the register file and to the Q register.
7	ADSL	Sends the output of the ALU to the output multiplexer; performs a 16-bit shift left of the ALU output, writing the shifted result to the register file.

Table 3-5 RALU Function Codes

AF CODE	FIRMWARE NAME	OPERATION PERFORMED
0	AFADD	Adds input J and input K (J + K).
1	AFK-1	Performs a One's complement of J and adds it to K (K - J - 1).
2	AFJ-1	Performs a One's complement of K and adds it to J (J - K - 1).
3	AFIOR	ORs inputs J and K (J OR K).
4	-	Not used.
5	AFKNJ	Performs a One's complement of J and ANDs it with K (K AND $\overline{J}$).
6	AFXOR	Exclusive OR of J and K (J XOR K).
7	AFXNR	Performs a One's complement of exclusive OR of J and K.
8	AFINC	Adds input J and input K with an input carry (J + K + 1).
9	AFK-J	Subtracts J from K (K - J).
A	AFJ-K	Subtracts K from J (J - K).
B	AMIOR	ORs inputs J and K with input carry*.
C	AFAND	ANDs inputs J and K with input carry*.
D	AMKNJ	Performs a One's complement of J, then ANDs result to K with input carry*.
E	AMXOR	Exclusive OR of J and K with input carry*.
F	AMXNR	Performs a One's complement of exclusive OR of J and K with input carry*.

*An input carry during AND operations creates an output carry and overflow signal; during logic operations, in other non-arithmetic operations, it is used to control the Memory Management Unit (MMU).

3.1.6 RALU Source (AS) Field

This 4-bit field (bits 16 through 19) controls which pair of operands (internal bus, register file left output, register file right output, Q register, or Zero) will be designated as the J and K inputs to the ALU. The most significant bit of this field may alter one or both of the operands to become a 20-bit sign-extended value (see Tables 3-6 and 3-7).

Table 3-6 RALU Source Codes - AS Field

AS CODE	FIRMWARE NAME	OPERATION PERFORMED
0 8	AWLQ* ASLQ	Takes J from the register file left output and K from the Q register.
1 9	AWLR* ASLR	Takes J from the register file left output and K from the register file right output.
2 A	AWZQ* ASZQ	Takes J equal to Zero and K from the Q register.
3 B	AWZR* ASZR	Takes J equal to Zero and K from the register file right output.
4 C	AWZL* ASZL	Takes J equal to Zero and K from the register file left output.
5 D	AWIL* ASIL	Takes J from BI and K from the register file left oputput.
6 E	AWIQ* ASIQ	Takes J from BI and K from the Q register.
7 F	AWIZ* ASIZ	Takes J from BI and K equal to Zero.

*The ALU zero detector is extended to test 20 bits, and Carry Out is taken from bit 0C instead of bit 10.

Table 3-7 RALU Source Codes - AS/AF Field

AS/AF CODE	FIRMWARE NAME	OPERATION PERFORMED
1/6	AXXX*	Value in the ALU is equal to double the contents of the register file left output (sign-extended). Left Select (LS) must equal Right Select (RS).
2/0	AXLQ*	Value in the ALU is equal to the contents of the register left output (sign-extended) plus the contents of the Q register.
2/8	AXLP*	Value in the ALU is equal to the contents of the register file left output (sign-extended) plus the contents of the Q register plus 1.
3/0	AXIR*	Value in the ALU is equal to the contents of the register file left output (sign-extended) plus the contents of the register file right output.

*The ALU zero detector is extended to test 20 bits, and Carry Out is taken from bit 0C instead of bit 10.

3.1.7 Processor Clock (CK) Speed Control Field

The CK field (bits 20 and 21) permits the processor clock to operate at intervals of 160, 180, 200, or 340 nanoseconds. The duration of each firmware step is thereby determined by this field (see Table 3-8).

Table 3-8 Clock Speed Codes

CK CODE	FIRMWARE NAME	CLOCK SPEED, Nanoseconds
0	CKVL	340
1	CKHL	200
2	CKHF	180
3	CKVF	160

3.1.8 Internal Bus (BI) Selector Control Field

The BI codes are listed in Tables 3-9 through 3-11. This 6-bit field (bits 22 through 27) performs one of four functions:

1. It generates firmware constants (9-bit sign-extended).
2. It selects which register content is to be delivered to the internal bus.

3. It determines which signal(s) will be sampled by the indicator register.

4. It generates control words to communicate with external processors.

Table 3-9 Internal Bus Codes, Constants

BI CODE (BITS)						FIRMWARE NAME	BI ACTION LINES (HEX CHARACTER POSITIONS)				
22	23	24	25	26	27		0C-0F	10-13	14-17	18-1B	1C-1F
0	0	b	b	b	b	(K0yz#)	0	0	0	y	z
0	1	b	b	b	b	(KFyz#)	0	F	F	y	z
0	0	N	N	N	N	K0--	0	0	0	y	-
0	1	N	N	N	N	KF--	0	F	F	y	-
0	N	0	0	0	0	K--0	0	-	-	y	0
0	N	0	0	0	1	K--1	0	-	-	y	1
0	N	0	0	1	0	K--2	0	-	-	y	2
0	N	0	0	1	1	K--3	0	-	-	y	3
0	N	0	1	0	0	K--4	0	-	-	y	4
0	N	0	1	0	1	K--5	0	-	-	y	5
0	N	0	1	1	0	K--6	0	-	-	y	6
0	N	0	1	1	1	K--7	0	-	-	y	7
0	N	1	0	0	0	K--8	0	-	-	y	8
0	N	1	0	0	1	K--9	0	-	-	y	9
0	N	1	0	1	0	K--A	0	-	-	y	A
0	N	1	0	1	1	K--B	0	-	-	y	B
0	N	1	1	0	0	K--C	0	-	-	y	C
0	N	1	1	0	1	K--D	0	-	-	y	D
0	N	1	1	1	0	K--E	0	-	-	y	E
0	N	1	1	1	1	K--F	0	-	-	y	F

LEGEND:

bbbb = binary value of hexadecimal digit z.

y = hexadecimal digit extracted from the NA field (NA 4 through 7).

N = may be either One or Zero (don't care situation).

- = may be any hexadecimal digit (don't care situation).

Table 3-10 Internal Bus Codes Selector (Sheet 1 of 2)

BI CODE (HEX)	FIRMWARE NAME	BI RECEIVES:		
		BI(0C-0f)	BI(10-17)	BI(18-1F)
20	BIR*	0000	H(18) See Note 1	ALU (18-1F)
21	BITS	0000	Y(07), 0000000	1,CPID 0, NA(4-7)
22	BIL	0000	H(10) See Note 2	H(10-17)
23	BIN(ns)**	0000	Interrupt Buffer	
23	DSTx(ds)**	0000	Bus Data Buffer	
23	PSTx(ps)**	0000	Bus Procedure Buffer	
24	BIA	Address Bus (BA)		
25	BIP(ns)**	H(1C-1F)	Control Panel	
25	DSHx(ds)**	H(1C-1F)	Bus Data Buffer	
25	PSHx(ps)**	H(1C-1F)	Bus Procedure Buffer	
26	BIB*	AU(0C-0F)	00000000	0000, AU(0C-0F)
27	BIV	0000	00000000	10,S(LVL)
28	Not Used			
29	BITC	0000	Y(07), 0000001	0,CPID,0. NA(4-7)
2A	BIH	0000	H(18-1F)	H(10-17)
2B	BIZ	0000	Trap Status Z-Word	
2C	Not Used			
2D	BIX	0000	Hex Decode of XB(0-3)	

Table 3-10 Internal Bus Codes Selector (Sheet 2 of 2)

BI CODE (HEX)	FIRMWARE NAME	BI RECEIVERS:		
		BI(0C)-0F)	BI(10-17)	BI(10-17)
2E	BII	0000	Indicators	00000000
2F	BIS	0000	Processor Status Register (S)	

*See Table 3-3.

**LEGEND

ns = no stal.

ds = stall, awaiting Megabus data (see Table 3-12).

ps = stall, awaiting procedure from Memory (see Table 3-12).

NOTES

1. Repeat bit H(18) for all eight positions.
2. Repeat bit H(10) for all eight positions.

Table 3-11 Internal Bus Control Indicators (Sheet 1 of 2)

BI CODE	FIRMWARE NAME	OPERATION PERFORMED
30	-	None
31	RI1	Loads the indicator register from the BI(18 through 1F).
32	RI2	Sets the overflow indictor if BI(10) and BI (11) are not equal; otherwise, no action occurs.
33	RI3	Copies the ALU overflow into the overflow indicator.
34	-	Not used.
35	RI5	Copies the Megabus ACK flip-flop into the I/O indicator.
36	RI6	If the RALU output is not Zero, sets the bit indicator; otherwise, clears the bit indicator.

Table 3-11 Internal Bus Control Indicators (Sheet 2 of 2)

BI CODE	FIRMWARE NAME	OPERATION PERFORMED
37	RI7	Copies BI(10) into the bit indicator, copies the ALU overflow into the overflow indicator, and copies the ALU carry into the carry indicator.
38	RI8	Copies ALU(10) into the less than indicator. Clear the greater than indicator if either BI(10) is one or the ALU output is zero; otherwise, sets the greater than indicator.
39	RI9	Copies AU(0C) into the less than indicator. Clears the greater than indicator if either AU(0C) is one or the ALU output is zero; otherwise, sets the greater than indicator.
3A	RIA	Copies the $\overline{\text{SIGN}}$ flip-flop to the greater than indicator; copies the SIGN flip-flop to the less than indicator and copies BI(10) to the unlike signs indicator.
3B	RIB	Copies the ALU overflow into the overflow indicator and copies the ALU carry into the carry indicator.
3C	RIC	Copies the shift out from Q(1F) to the carry indicator.
3D	RID	Copies BI(1F) into the carry indicator.
3E	RIE	Copies BI(10) into the carry indicator.
3F	RIF	Copies the ALU carry into the carry indicator.

3.1.9 Select Modify (SM) Field

The SM field (bits 28 through 30) affects the LS and RS fields equally. When the SM code is Zero, the LS and RS codes directly address the RALU left and right select inputs. SM codes 1, 2, 6, and 7 cause one of four bit-groups in the F (RF) register and SEL (RSEL) register to be ANDed with the RS and LS addresses, directing the result to the left and right register file inputs. SM codes 3 and 7 create constants that are ANDed with the RS and LS addresses (see Table 3-12 and 3-13).

Table 3-12 Select Modify Codes

SM CODE	FIRMWARE NAME	OPERATION PERFORMED
0	-	The LS and RS addreses are left unmodified.
1	SMN	The LS and RS addresses are ANDed with 1, RF (01 through 03), the register number field of the instruction word.
2	SMX	The LS and RS addresses are ANDed with 1, RF (09 through 0B), the field of the address syllable.
3	SME	The LS and RS addresses are ANDed with 1, 1, 1, 0.
4	-	Not used.
5	SMD	The LS and RS addresses are ANDed with 1, 1, 0, 1.
6	SMR	The LS and RS addresses are ANDed with 1, RSEL (1 through 3), the field of the address syllable.
7	SMS	The LS and RS addresses are ANDed with RSEL (0 through 3).

3.1.10 Megabus (BS) Control Field

Refer to Table 3-14 for Megabus control codes. This 5-bit field (bits 31 through 35) performs three functions:

1. Initiates Megabus cycles.

2. Controls the loading or incrementing of either the memory address register (Y) or the program counter register (P).

3. When data is requested from memory or an I/O controller and the data has not yet arrived, the BS field stalls the clock (inhibiting advancement beyond the current firmware step) until the data request is satisfied.

Table 3-13 Register Selection

SM PATTERN	LEFT/RIGHT ADDRESS SELECTION								SM CODE
	0	3	6	7	8	B	E	F	
1,1,1,1	D0 M0	D3 M3	D6 M6	D7 M7	B0 A0	B3 A3	B6 A6	B7 A7	0
1,F(1-3)	D0 M0	- -	D#' M#'	D# M#	B0 A0	- -	- -	B#	1(SMN)
1,F(9-B)	D0 M0	DX	- -	DX7	B0 A0	- -	- -	BX	2(SMX)
1,1,1,0	DO M0	D2 M2	D6 M6	D6 M6	B0 A0	B2 A2	B6 A6	B6 A6	3(SME)
1,1,0,1	D0 M0	D1 M1	D4 M4	D5 M5	B0 A0	B1 A1	B4 A4	B5 A5	5(SMD)
1,SEL(1-3)	D0 M0	- -	DB' MB'	DB MB	B0 A0	Bb	- -	BB	6(SMR)
SEL(0-3)	D0 M0	- -	- -	- -	- -	- -	- -	RSEL RAMSEL	7(SMS)
	0	1	2	3	4	5	6	7	
SM PATTERN	LEFT/RIGHT SELECT CODES								SM CODE

DB', MB', D#', OR M#' = The most significant half of an addressed pair.

DB, MB, D#, OR M# = An addressed register.

BB or B# = Selected Base Register (1 of 7).

Bb = Selected Base Register (B1, B2, or B3).

BX = Selected Base Register (B0 through B7).

RSEL = Selected Register (1 of 16).

DX7 = Used with CIP.

DX = Selected Index Register (D0, D1, D2, or D3).

Table 3-14 Megabus Control Codes (Sheet 1 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
00	PMUS	Loads the Address Bus (BA) from the contents of the Program Counter (P) register.
01	PURG	Loads the BA from the contents of the P register; discards Procedures 1 and 2 (P1/P2); sets the MT flip-flop.
02	YMUX	Loads the BA from the contents of the Y register.
03	YINC	Loads the BA from the contents of the Y register; increments the Y register by 1.
04	YLOD	Loads the BA from the contents of the P register; loads the Y register from the internal bus.
05	PINC (See Note 1)	Loads the BA from the contents of the P register; discards the P1 and P2 procedures; increments the P and CTR registers by 1; sets the MT flip-flop.
06	PAGJ	Loads the BA from the output of the memory management unit (bits 03 through 14) and the Y register (bits 15 through 22); loads the Y register from the internal bus.
07	YGJW	Loads the Memory Management Unit (MMU) from the contents of the Y register through the address bus (request check for write permission from the MMU).
08	YBAY	Loads the BA from the contents of the Y register; loads the Y register from the internal bus.
09	PLOD	Loads the BA from the contents of the P register; discards the P2 and P2 procedures; loads the P register from the internal bus; sets the MT flip-flop.
0A	YREL	Loads the BA from the contents of the Y register; loads the Y register from the internal bus ORed with Central Processor Channel (CPID) number.
0B	YOLD	Loads the BA from the contents of the Y register; loads the Y register (bits 07 through 22) from the internal bus.

Table 3-14 Megabus Control Codes (Sheet 2 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
0C	DSTY (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the Data Buffer (BD); loads the Y register from the internal bus.
	DSHY	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; loads the Y register with the contents of the internal bus.
0D	DSTU (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the BD; increments the Y register by 1.
	DSHU	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; increments the Y register by 1.
0E	DSHP	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; loads the P register from the internal bus; sets the MT flip-flop and discards the contents of procedure buffers P1 and P2.
0F	DSTL (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the BD.
	DSHL	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD.

Table 3-14 Megabus Control Codes (Sheet 3 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
10	DRCL	Loads the BA from the contents of the Y register; initiates a memory read cycle; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction.
11	DRCB	Loads the BA from the contents of the Y register; initiates a memory read cycle bypassing cache.
	DRCI	Loads the BA from the contents of the Y register; initiating a memory read cycle.
12	IORC	Loads the BA from the contents of the Y register; initiates an I/O read cycle.
14	DWHW	Loads the BA from the contents of the Y register; increments the Y register by 1; writes one byte from the internal bus to memory.
	DWWU	Loads the BA from the contents of the Y register; increments the Y register by 1; writes one word from the internal bus to memory.
15	DWHL	Loads the BA from the contents of the Y register; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction, writes one word from the internal bus to memory.
	DWWL	Loads the BA from the contents of the Y register; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction; writes one word from the internal bus to memory.
16	IOWU	Loads the BA from the contents of the Y register; writes one word to an I/O Device; increments the Y register by 1.
17	IOWH	Loads the BA from the contents of the Y register; writes one byte to an I/O Device.
	IOWW	Loads the BA from the contents of the Y register; writes one word to an I/O Device.
19	PRCI (See Note 4)	Loads the BA from the contents of the P register; initiates a memory read cycle for two words of procedure.

Table 3-14 Megabus Control Codes (Sheet 4 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
1A	PRCP	Loads the BA from the contents of the Y register; initiates a memory read cycle for two words of procedure; loads the P register from the internal bus.
1B		Not used.
1C	PSTY (See Notes 1, 2, and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP buffer is full); transfers the next procedure word to the internal bus from the Procedure Buffer (BP); loads the Y register from the internal bus; increments the P and CTR registers by 1.
	PSHY (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necesssary, until the BP is full); transfers the next procedure word to the internal bus from the H register (bits 1C through 1F) and the BP; loads the Y register from the internal bus; increments the P and CTR registers by 1.
1D	PSTL (See Notes 1, 2 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the internal bus from the BP; increments the P and CTR registers by 1.
	PSTI (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the I register via the internal bus from the BP; increments the P and CTR registers by 1.
	PSHL (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the internal bus from the H register (bits 1C through 1F) and the BP; increments the P and CTR registers by 1.

Table 3-14 Megabus Control Codes (Sheet 5 of 5)

NOTES

1. If the XB register is clocked, the control store word (bit 54) is transferred to the CTR register rather than incrementing the CTR register by 1.

2. Internal bus bits 0C through 0F are Zero.

3. Loads only internal bus (bits 10 through 1F); BI field available to control loading of indicator register.

4. The read request is ignored unless procedure buffers P1 and P2 are empty.

3.1.11 General Purpose (GP) Micro-Op Field

Refer to Table 3-15 for GP micro-op decodes. This 6-bit field (bits 36 through 41) generates a total of 64 micro-ops and is divided into five groups:

1. 16 micro-ops (GP40 through GP7C) affect the SIGN, ZERO, and SHIN control flip-flops, as well as the XB register.

2. 16 micro-ops (GP80 through GPBC) affect the F, SEL, and H registers, as well as the MISC control flip-flop.

3. 16 micro-ops (GP00 through GP3C) combine the actions of groups 1 and 2.

4. 8 micro-ops (GPC0 through GPDC) affect the following:

   a. BOOT and WRAP control flip-flops.

   b. S, M, and LINK registers.

   c. Megabus RINT function and Memory Management Unit (MMU) control functions.

5. 8 micro-ops (GPE0 through GPFC) are used for control panel functions.

Table 3-15 General Purpose Micro-Op Codes (Sheet 1 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
00	GP00	None.
01	GP04	Loads the H register from the internal bus; loads the SIGN flip-flop from internal bus bit 10.
02	GP08	Loads the SIGN flip-flop from internal bus bit 10; loads the ZERO flip-flop from the AUZERO function; loads the MISC flip-flop from internal bus bit 1F.
03	GP0C	Loads F (bits 8 through B) and SEL registers from the internal bus (bits 18 through 1F); sets the SIGN flip-flop and the FIRST flip-flop.
04	GP10	Clears the SHIN1 and SHIN2 flip-flops; shifts the XB register right by one position (ALU bit 1F to XB register bit 0).
05	GP14	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); set the ZERO flip-flop.
06	GP18	Loads the F and SEL registers from the internal bus; clears the XB register; sets the FIRST flip-flop.
07	GP1C	Sets the SHIN1 and SHIN2 flip-flops; loads the SEL register from internal bus bits 1C through 1F; sets the FIRST flip-flop.
08	GP20	Loads the ZERO flip-flop from the AUZERO function; loads the F and SEL registers from internal bus; sets the FIRST flip-flop; clears the MISC, SHIN1, and SHIN2 flip-flops; clears the XB register.
09	GP24	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); clears the SHIN2 and MISC flip-flops; sets the SHIN1 flip-flop.
0A	GP28	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); clears the ZERO flip-flop.

Table 3-15 General Purpose Micro-Op Codes (Sheet 2 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
0B	GP2C	Loads the F (bits 8 through B) and SEL registers from the internal bus (bits 18 through 1F); loads the SIGN flip-flop with internal bus bit 1F; if the internal bus (bits 10 through 15) equal/unequal 0, sets/clears the MISC flip-flop; sets the FIRST flip-flop.
0C	GP30	Loads the SIGN flip-flop from internal bus bit 10; sets the MISC flip-flop.
0D	GP34	Loads the SIGN flip-flop from internal bus bit 1F; sets the MISC flip-flop.
0E	GP38	Loads the SIGN flip-flop from internal bus bit 1F; loads the ZERO flip-flop with the state of the panel QLT flip-flop.
0F	GP3C	RFU.
10	GP40	Loads the SHIN2 flip-flop from the $\overline{\text{SIGN}}$ flip-flop.
11	GP44	Loads the SHIN2 flip-flop from the $\overline{\text{SIGN}}$ flip-flop; loads the SIGN flip-flop with internal bus bit 10.
12	GP48	Loads the SHIN1 flip-flop from the BIT indicator.
13	GP4C	Sets the SIGN flip-flop.
14	GP50	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0).
15	GP54	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); loads the SIGN flip-flop with internal bus bit 10.
16	GP58	Clears the ZERO flip-flop.
17	GP5C	Sets the ZERO flip-flop.
18	GP60	Loads the ZERO flip-flop from the AUZERO function.
19	GP64	Loads the SIGN flip-flop from internal bus bit 10.

Table 3-15 General Purpose Micro-Op Codes (Sheet 3 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
1A	GP68	Loads the SIGN flip-flop from internal bus bit 0C.
1B	GP6C	Loads the SIGN flip-flop from internal bus bit 1F.
1C	GP70	Clears the SHIN1 and SHIN2 flip-flops.
1D	GP74	Clears the SHIN2 flip-flop and sets the SHIN1 flip-flop.
1E	GP78	Sets the SHIN2 flip-flop and clears the SHIN1 flip-flop.
1F	GP7C	Sets the SHIN2 and SHIN1 flip-flops.
20	GP80	Loads the F and SEL registers from internal bus bits 10 through 1F; sets the FIRST flip-flop.
21	GP84	RFU.
22	GP88	Loads F register bits 8 through B and SEL register from internal bus bits 18 through 1F, respectively; sets the FIRST flip-flop.
23	GP8C	Clears the MISC flip-flop.
24	GP90	Sets the MISC flip-flop.
25	GP94	Loads the MISC flip-flop from the carry function.
26	GP98	Loads the MISC flip-flop from the Megabus ACK flip-flop.
27	GP9C	Loads the MISC flip-flop from the Protection Violation function.
28	GPA0	RFU.
29	GPA4	RFU.
2A	GPA8	Loads the SEL register from internal bus bits 1C through 1F; sets the FIRST flip-flop.
2B	GPAC	Loads the H register from the internal bus; loads the SEL register with internal bus bits 1C through 1F; sets the FIRST flip-flop.

Table 3-15 General Purpose Micro-Op Codes (Sheet 4 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
2C	GPB0	Loads the H register from the internal bus.
2D	GPB4	RFU.
2E	GPB8	RFU.
2F	GPBC	RFU.
30	GPC0	Broadcasts RINT (Retry Interrupts).
31	GPC4	Clears BOOT flip-flop.
32	GPC8	Loads the WRAP flip-flop from the result of an exclusive OR operation between the CARRY function and the output of the $\overline{\text{SIGN}}$ flip-flop; clears the FIRST flip-flop.
33	GPCC	Loads the RING number in the S register from internal bus bits 11 and 12.
34	GPD0	Loads the interrupt priority level in the S register from internal bus bits 1A through 1F; clears INTBSY if F register bit 5 is zero.
35	GPD4	Loads the LINK register from internal bus bits 17 through 1E.
36	GPD8	Loads M collector register from Y register bit 15, F register bit A, SEL register bits 0 and 2, $\overline{\text{ZERO}}$ flip-flop state, and the $\overline{\text{AUZERO}}$ function.
37	GPDC	With the RALU function code bits (see Table 3-1), AF0 and AF3 both Zero, loads the Memory Management Unit (MMU) with the validity bit and base address field of a segment descriptor. With AF0 Zero and AF3 One, loads the MMU with the access rights and segment size field of a segment descriptor. With AF0 One and AF3 Zero, sets the MMU NO-CHECK flip-flop. With AF0 and AF3 both One, makes the MMU check the validity of range, address, and access rights.

Table 3-15 General Purpose Micro-Op Codes (Sheet 5 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
38	GPE0	Loads panel display H4 from internal bus bits 1C through 1F.
39	GPE4	Loads panel displays H5, H6, H7, and H8 from from internal bus bits 10 through 1F.
3A	GPE8	Clears QLT indicator if the panel is Load mode.
3B	GPEC	Sets MPLOAD mode; sets panel QLT indicator if panel is locked, memory is volatile, and the ZERO flip-flop is Off.
3C	GPF0	BI receives control word from panel (see Table 3-9, BIP code).
3D	GPF4	Loads the panel TRAFFIC indicator from the ZERO function; sets Run mode if panel is in Ready mode.
3E	GPF8	Clears panel Load mode; sets TRAFFIC indicator and Run mode if panel is in Ready mode.
3F	GPFC	Clears panel CHANGE control flip-flop.

3.1.12 Test Condition (TC) Field

This 6-bit field (bits 42 through 47) samples one of 64 signals upon which the firmware may branch (see Tables 3-16 through 3-20).

Table 3-16 Test Condition Codes (Sheet 1 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
0	TC00	False.
1	TC01	Scientific Store Operation.
2	TC02	SIP presence.
3	TC03	CIP presence.
4	TC04	Operand size is 8 bits.
5	TC05	Operand size is 16 bits.
6	TC06	Operand size is greater than 16 bits.

Table 3-16 Test Condition Codes (Sheet 2 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
7	TC07	Operand size is 64 bits.
8	TC10*	Branch operation condition; if F register (bits 0 through 7) is equal to 0, POWER OK is tested.
9	TC11	Control panel $\overline{\text{EXECUTE}}$ mode; copy control panel LOAD mode into MYBOOT flip-flop.
A	TC12	LAF.
B	TC13	Control panel $\overline{\text{LOAD}}$ mode.
C	TC14	Inclusive OR of MISC or ZERO flip-flop.
D	TC15	SHIN1 flip-flop.
E	TC16	SHIN2 flip-flop.
F	TC17	In previous cycle, the address bus was loaded with the contents of the Y register.
10	TC20**	If # (F register bits 1 through 3) is not equal to Zero, test the complement of the specified bit of the internal bus (19 through 1F); if # is equal to Zero, test complement of the I/O indicator.
11	TC21	Carry function.
12	TC22	WCS absence.
13	TC23	Internal bus bit 10.
14	TC24	SHIN function.
15	TC25	Internal bus bit 10 is not equal to internal bus bit 11.
16	TC26	Inclusive OR of SHIN and AUZERO functions.
17	TC27	AUZERO function.
18	TC30	Cache absence.
19	TC31	Read-Modify-Write Flip-flop (RMWF) (initialize MT logic).
1A	TC32	Control panel LOCK function.

Table 3-16 Test Condition Codes (Sheet 3 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
1B	TC33	M register bit 0 (J).
1C	TC34	Indicator (I) register bit 0 (carry).
1D	TC35	Megabus Acknowledge flip-flop.
1E	TC36	Ring number code greater than 1.
1F	TC37	Parity error indicator.
20	TC40	SEL register value not equal to Zero.
21	TC41	RFU.
22	TC42	F register # field equals 7.
23	TC43	XB register (bit 0).
24	TC44	F register (bit 4).
25	TC45	F register (bit 5).
26	TC46	F register (bit 6).
27	TC47	F register (bit 7).
28	TC50	F register (bit 8).
29	TC51	SEL register register = 0 (decrements the SEL register by 1).
2A	TC52	F register (bit 9).
2B	TC53	F register (bit B).
2C	TC54	SEL register (bit 0).
2D	TC55	SEL register (bit 1).
2E	TC56	SEL register (bits 1, 2, 3, = 7).
2F	TC57	SEL register (bit 3).
30	TC60	WDT/RTC request (SNAPIT).
31	TC61	YELLOW flag (clear SNAPER and RBYELF flip-flops).
32	TC62	Register address syllable.

Table 3-16 Test Condition Codes (Sheet 4 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
33	TC63	ZERO flip-flop.
34	TC64	SIGN flip-flop.
35	TC65	MISC flip-flop.
36	TC66	SEL register bit 2.
37	TC67	Inclusive OR of GJBARF function and FFWRAP flip-flop.
38	TC70	Interrupt not busy.
39	TC71	Interrupt busy or External Trap; clears SIGN flip-flop.
3A	TC72	Internal bus bit 18.
3B	TC73	Overflow function; if AS field equals AWXX, test AU(0C).
3C	TC74	RFU.
3D	TC75	RFU.
3E	TC76	Q register shift right; if not shift right, test Data Descriptor Length flip-flop.
3F	TC77	Internal bus bit 1F.

*Three additional multiplexers are used to create the input for TC10 (see Tables 3-17, 3-18, and 3-19).

**An additional multiplexer is used to create the input for TC20 (see Table 3-20).

Table 3-17 Test Condition 10 (Group A)

FUNCTION DECODE			
NUMZ	F04	F05	FUNCTION TESTED
0	0	0	Refer to Group B decode
0	0	1	Refer to Group B decode
0	1	0	Refer to Group B decode
0	1	1	Not used
1	0	0	Refer to Group C decode
1	0	1	Refer to Group B decode
1	1	0	Refer to Group C decode
1	1	1	Logical One

Table 3-18 Test Condition 10 (Group B)

FUNCTION DECODE			FUNCTION TESTED
NUMZ	F06	F07	
0	0	0	FFSIGN (SIGN flip-flop)
0	0	1	FFZERO (ZERO flip-flop)
0	1	0	$\overline{\text{FFSIGN}}$ AND $\overline{\text{FFZERO}}$
0	1	1	FFMISC (MISC flip-flop)
1	0	0	RIOVFF (Overflow Indicator)
1	0	1	RIBITF (Bit Indicator)
1	1	0	RICRYF (Carry Indicator)
1	1	1	RIACKF (I/O Indicator)

Table 3-19 Test Condition 10 (Group C)

FUNCTION DECODE			FUNCTION TESTED
F04	F06	F07	
0	0	0	MYPROK (Power OK)
0	0	1	Not used
0	1	0	RILESF (Less Than Indicator)
0	1	1	RIGTRF (Greater Than Indicator
1	0	0	RILESF and RISNEF are different
1	0	1	RILESF and RIGTRF are both off
1	1	0	RIGTRF and RISNEF are different
1	1	1	RISNEF (Signs Unlike Indicator)

Table 3-20 Test Condition 20 (Group M)

FUNCTION DECODE			FUNCTION TESTED
F01	F02	F03	
0	0	0	Acknowledge Indicator ($\overline{\text{RIACKF}}$)
0	0	1	Internal Bus ($\overline{\text{Bit 19}}$)
0	1	0	Internal Bus ($\overline{\text{Bit 1A}}$)
0	1	1	Internal Bus ($\overline{\text{Bit 1B}}$)
1	0	0	Internal Bus ($\overline{\text{Bit 1C}}$)
1	0	1	Internal Bus ($\overline{\text{Bit 1D}}$)
1	1	0	Internal Bus ($\overline{\text{Bit 1E}}$)
1	1	1	Internal Bus ($\overline{\text{Bit 1F}}$)

3.1.13 Branch Type (BR) Field

This 4-bit field (bits 48 through 51) selects the mechanism that is to produce the address of the next firmware step (see Table 3-21).

Table 3-21 Branch Type Codes (Sheet 1 of 2)

BR CODE	FIRMWARE NAME	OPERATION PERFORMED
0	X0T	If the test condition is false, use the NA field. If the test condition is true, OR the NA field with 3.
1	XLT	If the test condition is false, use the NA field. If the test condition is true, use the LINK register.
2	XAT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation ouptuts for the XA branch.
3	XBT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XB branch.
4	XRT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation ouputs for the XR branch.
5	XWT	If the test condition is false, use the NA FIELD. If the test condition is true, use the next address generation outputs for the XW branch.
6	XET	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XE branch.
7	XFT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XF branch.

Table 3-21 Branch Type Codes (Sheet 2 of 2)

BR CODE	FIRMWARE NAME	OPERATION PERFORMED
8	X0F	If the test condition is true, use the NA field. If the test condition is false, OR the NA field with 3.
	X-F	Use the NA field (test should never be false).
9	XLF	If the test condition is true, use the NA field. If the test condition is false, use the LINK register.
A	XAF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XA branch.
B	XBF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XB branch.
C	XRF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XR branch.
D	XWF	If the test conditon is true, use the NA field. If the test condition is false, use the next address generation outputs for the XW branch.
E	XEF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XE branch.
F	XFF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XF branch.

3.1.14 C Field

This 1-bit field (bit 52) controls the (optional) cache memory usage.

3.1.15 Next Address (NA) Field

This 11-bit field (bits 53 through 63) defines the address of the next firmware step except as altered by the TC and BR fields (see Table 3-22).

Table 3-22 Next Address Code

NA CODE (HEX)	FIRMWARE NAME	INTERPRETATION
000 through 3FF	N	Specifies the address of the next firmware step unless altered by the Branching (BR) field. The middle digit of the NA field (bits 57 through 60 of the firmware word) becomes the 16 weight digit of 9-bit sign-extended firmware generated constants. Bits 57 through 60 are also used to create function codes for communication with external processors (refer to Table 3-10, internal bus codes 21 and 29).

3.2 TRAPS

The trap firmware is supported by four groups of main memory locations, which are identified as follows:

- Dedicated memory locations
- Trap save area
- Interrupt save area
- Trap handler procedure.

3.2.1 Dedicated Memory Locations

The first dedicated memory locations associcated with traps are those which contain Next Available Trap Save Area Pointers (NATSAP) to the next available trap save area. The second group of locations are hexadecimal 0024 through 007F. These locations contain pointers (trap vectors) to the trap handler procedures. The third group of locations, hexadecimal 0080 through 00FF, contain pointers (interrupt vectors) to the interrupt save areas. The dedicated memory locations are illustrated in Figures 2-7 and 2-8.

3.2.2 Trap Save Area

Each trap save area consists of eight sequential entries, which are used to store the context associated with a particular trap. The layout of each trap save area is shown below.

Entry 1	TSAL*
Entry 2	I Register
Entry 3	Data Register 3
Entry 4	Instruction
Entry 5	Z Field
Entry 6	A Field*
Entry 7	P Field*
Entry 8	Base Register 3*

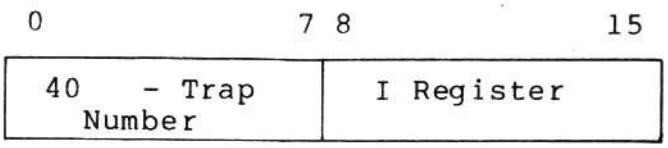
*Occupies two words in LAF

3.2.2.1 Entry 1

Entry 1 contains the Trap Save Area Link (TSAL). TSAL is an address pointer to the first location in the next trap save area. The first location (TSAL) in this trap save area points to the first location in the next one and so on, thereby forming a linked list of trap save areas. The eldest trap save area in the linked list is indicated by a null (Zero pointer) in entry 1.

3.2.2.2 Entry 2

Entry 2 stores the Trap Vector number and the current contents of the indicator register when a trap occurs. The format of entry 2 is:



3.2.2.3 Entry 3

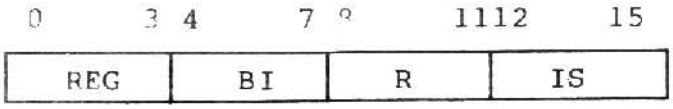
Entry 3 stores the current contents of Data Register 3 (D3) when a trap occurs.

3.2.2.4 Entry 4

Entry 4 stores the first word of the instruction that was being processed when the trap occurred.

3.2.2.5 Entry 5

Entry 5 is defined as the Z field and is used to store miscellaneous information when a trap occurs. The format of the Z field is:



3.2.2.5.1 REG FIELD

The REG field indicates whether or not the A field content is valid. If this digit equals Zero, the A field contains valid information; otherwise, the A field is unspecified.

3.2.2.5.2 BI Field

Two conditions must exist before this field is valid:

1. The REG digit must be zero.
2. The trapped instruction must be a bit, byte, or I/O instruction.

    a. Bit Instruction: The BI field contains the four low-order bits of the selected index register, or 0000 if no indexing was used.

    b. Byte Instructions: The BI field contains X000, where X equals the low-order bit of the selected index register, or 0000 if no indexing was used.

    c. I/O Instructions: If the A field (entry 6) contains the channel number, the most significant bit (4) of the BI field contains the low-order bit of the function code.

3.2.2.5.3 R Field

This field indicates the processor ring number when a trap occurs.

3.2.2.5.4 IS Field

The IS field indicates the size (in words) of the trapped instruction. However, if the trap occurs before the entire instruction was fetched, the IS field indicates the number of words that were fetched from the beginning of the Instruction Fetch firmware. For example, this condition may ocur when an I/O instruction is trapped, or if a nonrecoverable memory error occurs while fetching part of a multiword instruction.

3.2.2.6 Entry 6

Entry 6 is defined as the A field. This field is valid only when bit 0 of the Z field equals Zero. The A field contains the effective address generated by the trapped instruction address syllable, the address of the trapped instruction, or the address of the following instruction, depending on the particular trap activated.

3.2.2.7 Entry 7

Entry 7 is defined as the P field. This field contains the return address from the trap handler procedure. In most cases, this return address is the address of the location or instruction following the trapped instruction. However, when a multiword instruction is trapped, the return address may be within the multiword instruction. Consequently, modification of the P field may be required before returning from the trap (RTT). Subtracting the IS field of Entry 5 from Entry 7 always produces a pointer to the start of the current instruction.

3.2.2.8 Entry 8

Entry 8 stores the contents of Base Register 3 (B3) when a trap occurs. After thus saving its conents, and before entry into the Trap Handler procedure, B3 is loaded with a pointer to the A field of the trap save area.

3.2.3 Interrupt Save Area

There is a linked list of zero or more trap save areas attached to each interrupt save area. One interrupt save area is assigned to each of the 64 possible interrupt levels. Entry (-1) of each interrupt save area is a pointer to the string of trap save areas that are currently in use by traps in this level. If there are no trap save areas assigned to the interrupt save area, the interrupt save area contains a null (Zero address).

3.2.4 Trap Handler Procedures

The trap handler procedures are software programs which process traps. Return from the trap handler procedure is accomplished using the Return from Trap (RTT) generic instruction.

3.2.5 Trap Functionality

A trap can occur from many places in the firmware as a result of any number of causes. For example, a trap will be entered on reference to an unavailable resource (e.g., during an instruction or data fetch), or when an error condition (e.g., program error, protection violation, etc.), occurs, or when an unusual result (e.g., arithmetic overflow) is encountered.

When a trap does occur, the Next Available Trap Save Area Pointer (NATSAP) will be read from memory to obtain a new Trap Save Area (TSA). A context save will take place to store the trap vector number, indicators, data register 3, the instruction, the Z field, the A field, the P field, and base register 3 in sequential memory locations in the new TSA. The interrupt vector links the Interupt Save Area (ISA) of the current process to the new TSA. The trap vector pointer is used and the resultant trap handler procedure is entered to process the trap.

A return from the trap routine will take place when the generic instruction Return from Trap (RTT) is invoked. The TSA is unlinked from the current process and is returned to the pool of available trap save areas.

3.3 INTERRUPTS

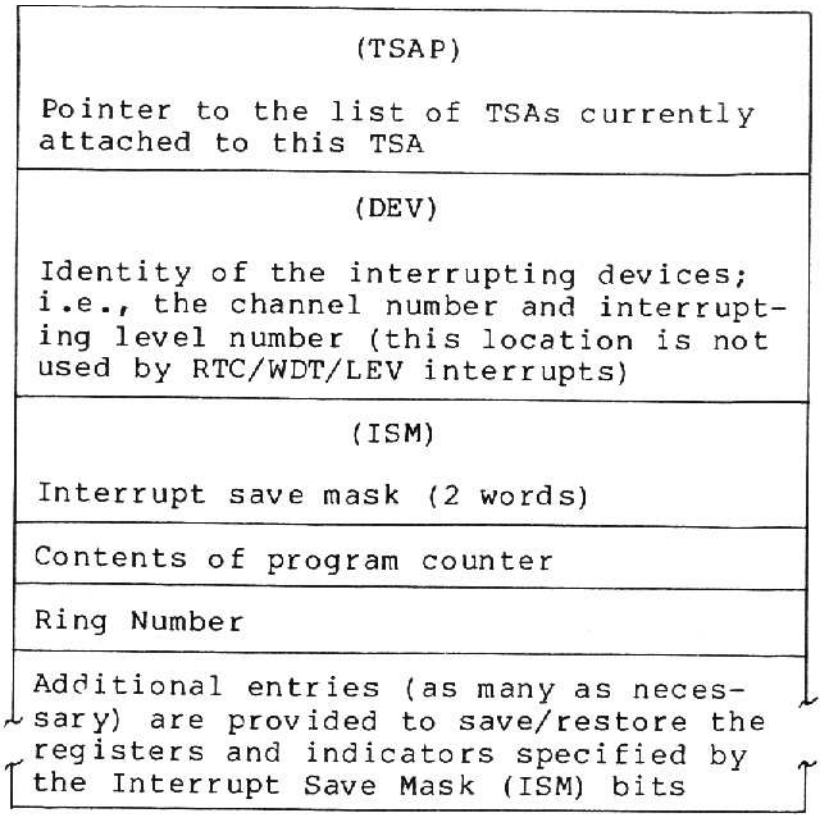
Every program in the CPU executes at a priority level, which is defined by the LVL field of the S register. This interrupt priority level has a range of 0 to 63, inclusive. Level 0 is defined as having the highest priority; level 63 has the lowest priority.

The program currently being executed can be interrupted by an event having a higher priority. The interrupt is serviced at the end of the instruction currently being executed. The level assignments for various events are shown in Table 3-23.

Table 3-23 Level Assignements for Interrupt Events

EVENT CAUSING INTERRUPT	LEVEL ASSIGNMENT	COMMENTS
Incipient power failure	0	Highest priority
Watchdog timer runout	1	-
Use of last TSA	2	-
Real-time clock	0 through 62	Level is contained in main memory location 0016 (Hex)
Device requiring service	1 through 62	Level is assigned to device by software
LEV instruction	0 through 63	Level specified by instruction

Each level has a corresponding interrupt vector, which points to an Interrupt Save Area (ISA). When the program currently being executed is interrupted, its context is stored in the assigned interrupt save area. The context of the interrupting process is then retrieved from the ISA that is assigned to its level and the process begins executing at its assigned level. The layout of each ISA is:



3.3.1 Interrupt Vectors

The interupt vectors (see Figures 2-7 and 2-8) consist of 64 address pointers. These address pointers are contained in dedicated memory locations 0080 through 00FF hexadecimal. The interrupt vector for level 0 is contained in the first entry; the interrupt vector for level 1 is in the second entry and so on up to the interrupt vector for level 63. Each interrupt vector points to one of the 64 interrupt save areas described above.

3.3.2 Activity Flags

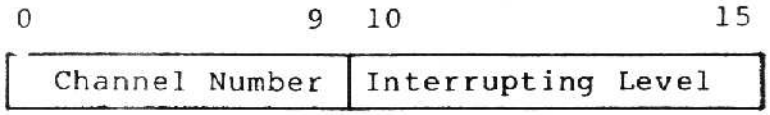
The activity flags (see Figure 2-7 or 2-8) associated with interrupts consist of 64 bits, which are stored in four dedicated memory locations: 0020, 021, 0022, and 0023 hexadecimal. The activity flag for level 0 is contained in bit 0 of location 20; the activity flag for level 15 is in bit 15 of location 20; the activity flag for level 16 is in bit 0 of location 21 and so on up to level 63, which is in bit 15 of location 23.

These activity flags indicate which processor levels are currently active (i.e., ready for execution or being executed). If a bit (activity flag) is On, an activity for the corresponding level will begin executing as soon as it is found to be the highest priority level active. If a bit is Off, the corresponding level is inactive. Activity flags are set by external interrupt requests and set/cleared by the LEV instruction.

NOTE

The activity flag for level 63 is always considered to be On.

When an external interrupt occurs, the identity of the interrupting channel is stored in the DEV entry of its interrupt save area. The format of location DEV is:



NOTES

1. If the interrupt vector for the interrupting level is null, the interrupt is acknowledged but not scheduled. However, the activity flags are scanned as described in subsection 3.3.3.

2. If the interrupting level equals the current level (or if different, and both levels have the same interrupt save area), the context save/restore process, described above, is bypassed.

3. Note 2 applies for all level changes, not just External Interrupts.

The Interrupt Save Mask (ISM) controls storing and loading of the processor registers during the save/restore process. The ISM format is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	D1	D2	D3	D4	D5	D6	D7	I	B1	B2	B3	B4	B5	B6	B7
MMU	NATSAP			RFU		SIP	CIP	RFU						T	M

M = M registers 2 through 7

T = stack address register, T

CIP = the indicator register associated with the commercial instruction processor

SIP = the three scientific accumulators and the indicator register associated with the scientific instruction processor

NATSAP = the location where NATSAP will be obtained as shown in the chart below:

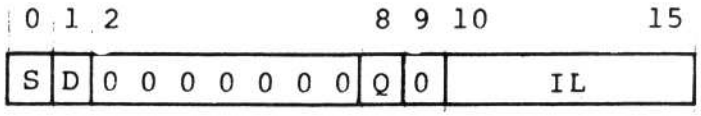
ISM BITS			MEMORY
17	18	19	LOCATION
0	0	0	0010
0	0	1	000E
0	1	0	000C
0	1	1	000A
1	0	0	0008
1	0	1	0006
1	1	0	0004
1	1	1	0002

MMU = the 16/31 segment descriptors of the memory management unit

The mask bits are scanned from right to left. If the mask bit is On, the corresponding processor register is either loaded from or stored in the ISA. The registers selected by the mask bits are loaded from/saved in consecutive memory locations, starting at the location immediately after that used to save/restore the S register. The number of memory locations utilized for the storage of processor registers is a function of the number of mask bits set.

3.3.3 LEV Instruction

The LEV instruction is one of the central processor single operand instructions and is used to set/clear activity flags. The LEV operand contains nine significant bits as shown below:



S = Suspend current level
D = Defer interrupt
Q = Quick level change (inhibit)
IL = Interrupt level number

The operations performed by the S, D, Q, and IL bits are listed in Table 3-24.

Table 3-24 LEV Operand Decode

OPERAND BITS S	D	Q	OPERATION PERFORMED
0	0	0	1. Sets the activity flag for level IL. 2. Scans activity flags to determine the highest priority active. 3. Saves the context of the current level*. 4. Loads the context for the highest priority active level*.
0	1	0	1. Sets the activity flag for level IL.
1	X	0	1. Sets the activity flag. 2. Clears the activity flag for the current level. 3. Scans the activity flags to determine the highest priority active interrupt level. 4. Saves the context of the current level*. 5. Loads the context for the highest priority active level*.
0	X	1	1. Sets the activity flag for level IL. 2. Sets the current level equal to IL without a context change. 3. Sets the IV of the new level equal to that of of the old level.
1	X	1	1. SEts the activity flag for level IL. 2. Clears the activity flag for the current level. 3. Sets the current level equal to IL without a context change. 4. Sets the IV of the new level equal to that of the old level.

*If interrupt save areas are the same, this operation is bypassed.

NOTE

An IL field equal to 63 is effectively a No-Op.