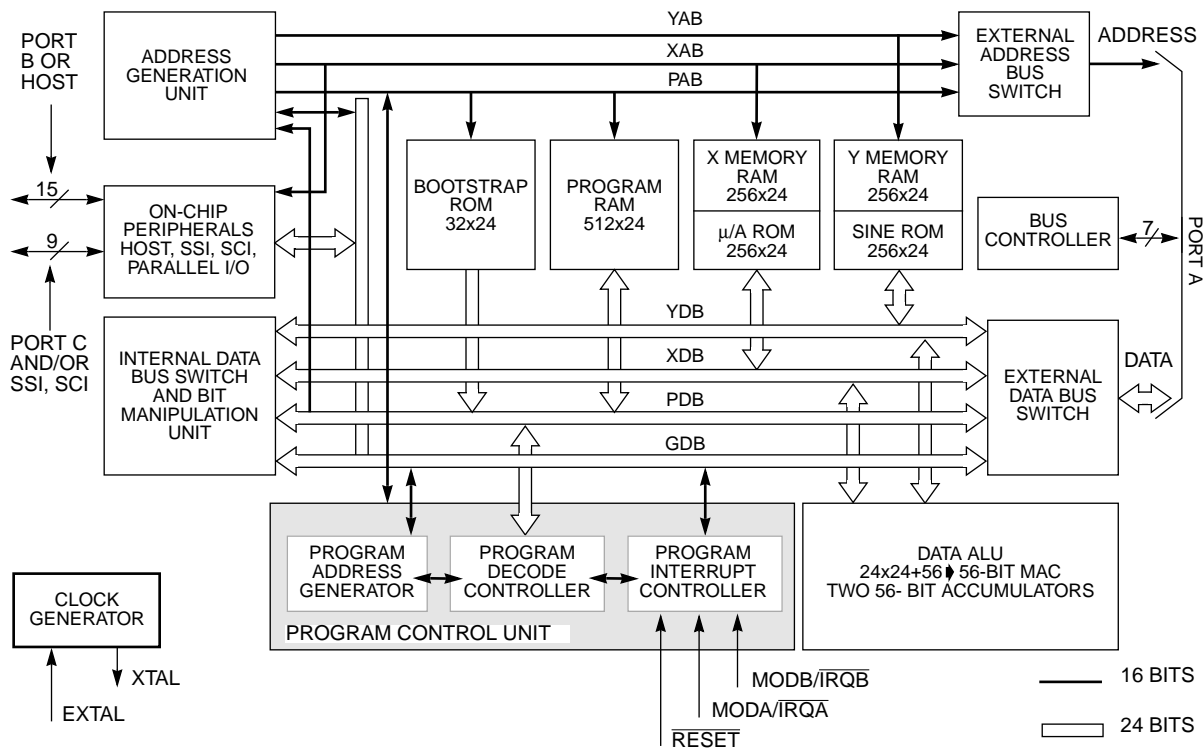


## SECTION 6 PROGRAM CONTROL UNIT

This section describes the hardware of the program control unit and concludes with a description of the programming model. The instruction pipeline description is also included since understanding the pipeline is particularly important in understanding the DSP56000/DSP56001.

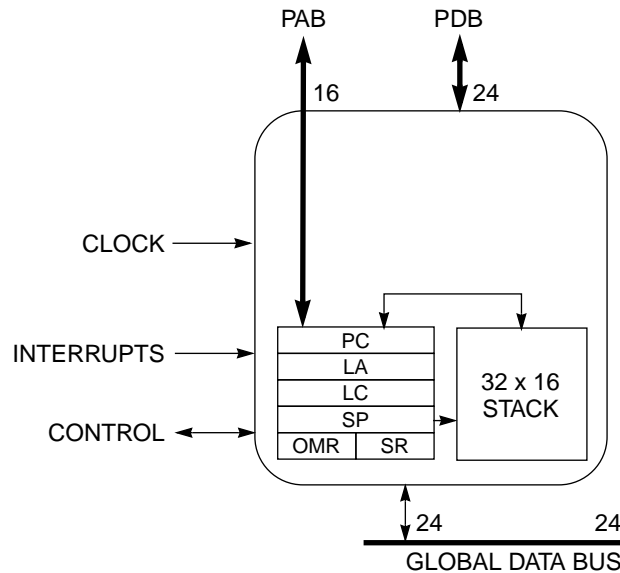
### 6.1 OVERVIEW

The program control unit (one of the three concurrent execution units in the central processor) performs program address generation (instruction prefetch), instruction decoding, hardware DO loop control, and exception processing (see Figure 6-1). The programmer views the program control unit as consisting of six registers and a hardware system stack (SS) as shown in Figure 6-2. In addition to the standard program flow-control resources, such as a program counter (PC), complete status register (SR), and SS, the program control unit features registers (loop address (LA) and loop counter (LC) dedicated to supporting the hardware DO loop instruction.



**Figure 6-1 DSP56001 Block Diagram**

The SS is a 15-level by 32-bit separate internal memory used to store the PC and SR



**Figure 6-2 DSP56000/DSP56001 Program Control Unit**

during subroutine calls and long interrupts. The SS will also store the LC and LA registers in addition to the PC and SR registers for program looping. Each location in the SS is addressable as 16-bit registers, system stack high (SSH) and system stack low (SSL), which are pointed to by the stack pointer (SP). Thus, SS management is under software control.

All registers are read/write to facilitate system debugging. Although none of the program control unit registers are 24 bits, they are read or written over 24-bit buses. When they are read, the least significant bits (LSBs) are significant, and the most significant bits (MSBs) are zeroed as appropriate. When they are written, only the appropriate LSBs are significant, and the MSBs are written as don't care. The program control unit implements a three-stage (prefetch, decode, execute) pipeline and controls the five processing states of the DSP56000/DSP56001: normal, exception, reset, wait, and stop.

## 6.2 PROGRAM CONTROL UNIT ARCHITECTURE

The program control unit consists of three hardware blocks: the program decode controller (PDC), the program address generator (PAG), and the program interrupt controller (PIC) (see Figure 6-1).

### 6.2.1 Program Decode Controller

The PDC contains the program logic array decoders, the register address bus generator, the loop state machine, the repeat state machine, the condition code generator, the inter-

rupt state machine, the instruction latch, and the backup instruction latch. The PDC decodes the 24-bit instruction loaded into the instruction latch and generates all signals necessary for pipeline control. The backup instruction latch stores a duplicate of the prefetched instruction to optimize execution of the repeat (REP) and jump (JMP) instructions.

## 6.2.2 Program Address Generator

The PAG contains the PC, the SP, the SS, the operating mode register (OMR), the SR, the LC register, and the LA register. Loops, which are frequent constructs in digital signal processing (DSP) algorithms, are supported by dedicated hardware on the DSP56000/DSP56001. Executing a DO instruction loads the LC register with the number of times the loop should be executed, loads the LA register with the address of the last instruction word in the loop (fetched during one loop pass), and asserts the loop flag in the SR. Executing the DO instruction also causes the contents of the LA, LC, and SR to be stacked prior to the execution of the DO instruction, thereby supporting nesting of DO loops. Under control of the loop state machine, the address of the first instruction in the loop is also stacked so the loop can be repeated with no overhead. While the loop flag in the SR is asserted, the loop state machine will compare the PC contents to the contents of the LA to determine if the last instruction word in the loop was fetched. If the last word was fetched, the LC contents are tested for one. If LC is not equal to one, then it is decremented, and the SS is read to update the PC with the address of the first instruction in the loop, effectively executing an automatic branch. If the LC is equal to one, then the LC, LA, and the loop flag in the SR are restored with the stack contents, while instruction fetches continue at the incremented PC value (LA + 1).

Block data moves can be accomplished using the repeat feature. The REP instruction loads the LC with the number of times the next instruction is to be repeated. Since the instruction to be repeated is only fetched once, throughput is increased by reducing external bus contention. However, REP instructions are not interruptable since they are fetched only once. A single-instruction DO loop can be used in place of an REP if interrupts must be allowed.

## 6.2.3 Program Interrupt Controller

The PIC receives all interrupt requests, arbitrates among all of them each cycle, and generates the interrupt vector address. There are four external and 16 internal interrupt sources that may generate interrupts.

The interrupts are organized in a flexible priority structure. Each interrupt has associated with it an interrupt priority level (IPL) that can be from zero to three. Levels 0 (lowest level), 1, and 2 are maskable. Level 3 is the highest IPL and is not maskable. Two interrupt mask bits in the SR reflect the current processor IPL and indicate the level needed for an interrupt source to interrupt the processor. Interrupts are inhibited for all IPLs less than the current processor priority. Level 3 interrupts can always interrupt the processor. All interrupt sources and their IPLs are listed in Table 6-1. Each interrupt source is vectored (one of 32 vectors) to a separate, fixed, two-word service routine located in the lowest 64 words of program memory. If some of this space is not used, it may be used for program storage.

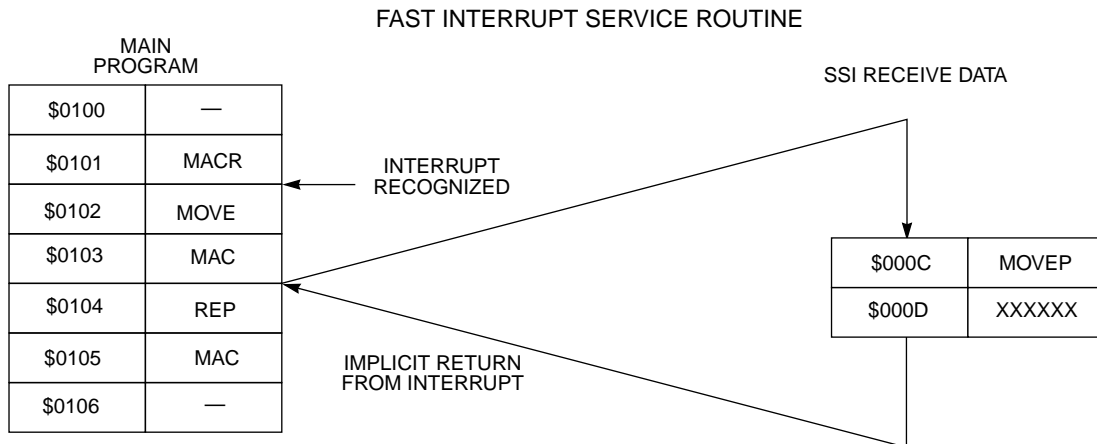
Upon entering the exception processing state, the current instruction in decode will execute normally, unless it is the first word of a two-word instruction, in which case it will be aborted and refetched at the completion of exception processing. The next two fetch addresses are supplied by the PIC. During these fetches, the PC is not updated. The PIC generates an interrupt instruction fetch address, which points to the first instruction word of a two-word fast-interrupt routine. All interrupts begin as fast interrupts (Figure 6-3 (a)). During fast interrupt servicing, the two instruction words at the interrupt vector addresses are jammed into the instruction stream without any overhead or stack usage. If one of the two words is a jump to subroutine (JSR), the fast interrupt routine becomes a long interrupt routine (see Figure 6-3 (b)). The long interrupt service is the traditional context switch in which the stack is used for saving the status and return address. Subroutines and interrupts can be nested using the 15-level stack. The stack can be extended in memory by

## Table 6-1 Interrupt Sources

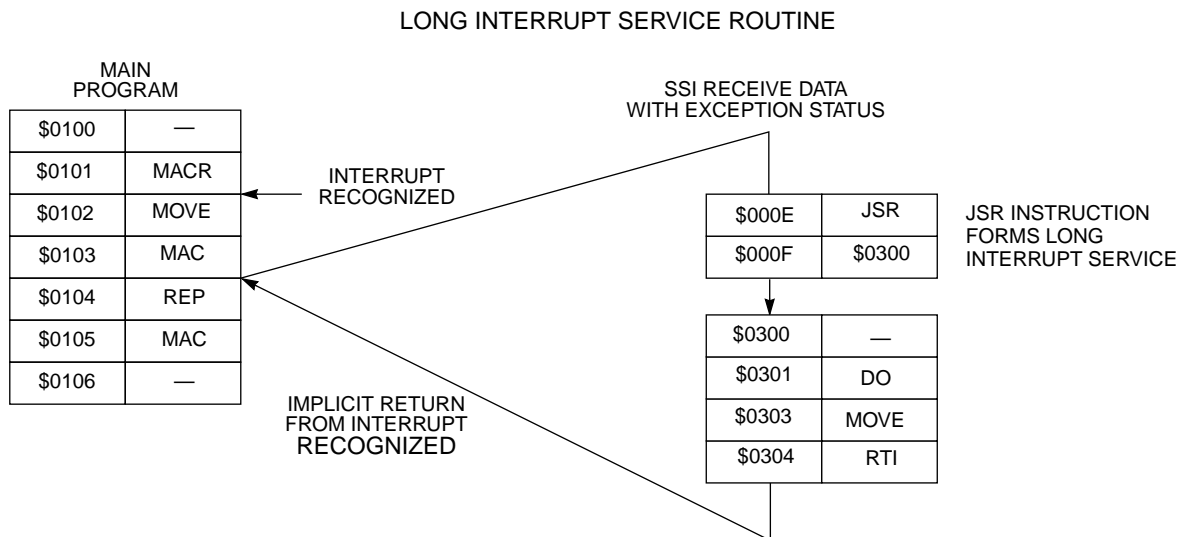
| Interrupt Starting Address | IPL | Interrupt Source                                   |
|----------------------------|-----|--|
| P:\$0000 or P:\$E000       | 3   | Hardware $\overline{\text{RESET}}$ (External)      |
| P:\$0002                   | 3   | Stack Error  |
| P:\$0004                   | 3   | Trace  |
| P:\$0006                   | 3   | SWI (Software Interrupt)                           |
| P:\$0008                   | 0-2 | $\overline{\text{IRQA}}$ (External)                |
| P:\$000A                   | 0-2 | $\overline{\text{IRQB}}$ (External)                |
| P:\$000C                   | 0-2 | SSI Receive Data                                   |
| P:\$000E                   | 0-2 | SSI Receive Data with Exception Status             |
| P:\$0010                   | 0-2 | SSI Transmit Data                                  |
| P:\$0012                   | 0-2 | SSI Transmit Data with Exception Status            |
| P:\$0014                   | 0-2 | SCI Receive Data                                   |
| P:\$0016                   | 0-2 | SCI Receive Data with Exception Status             |
| P:\$0018                   | 0-2 | SCI Transmit Data                                  |
| P:\$001A                   | 0-2 | SCI Idle Line                                      |
| P:\$001C                   | 0-2 | SCI Timer  |
| P:\$001E                   | 3   | NMI — Reserved for Hardware Development (External) |
| P:\$0020                   | 0-2 | Host Receive Data                                  |
| P:\$0022                   | 0-2 | Host Transmit Data                                 |
| P:\$0024                   | 0-2 | Host Command (Default)                             |
| P:\$0026                   | 0-2 | Available for Host Command                         |
| P:\$0028                   | 0-2 | Available for Host Command                         |
| P:\$002A                   | 0-2 | Available for Host Command                         |
| P:\$002C                   | 0-2 | Available for Host Command                         |
| P:\$002E                   | 0-2 | Available for Host Command                         |
| P:\$0030                   | 0-2 | Available for Host Command                         |
| P:\$0032                   | 0-2 | Available for Host Command                         |
| P:\$0034                   | 0-2 | Available for Host Command                         |
| P:\$0036                   | 0-2 | Available for Host Command                         |
| P:\$0038                   | 0-2 | Available for Host Command                         |
| P:\$003A                   | 0-2 | Available for Host Command                         |
| P:\$003C                   | 0-2 | Available for Host Command                         |
| P:\$003E                   | 0-2 | Illegal Instruction                                |

using software to access the SSH and SSL registers. The exception processing state is described in more detail in SECTION 8 PROCESSING STATES.

Two external interrupt request inputs, IRQA and IRQB, can be defined as either level sensitive or negative edge triggered. One other external interrupt source is available. The nonmaskable interrupt (NMI) is edge sensitive and is generated on the first transition to 10



(a) DSP56000/DSP56001 Fast Interrupt



(b) DSP56000/DSP56001 Long Interrupt

Figure 6-3 Fast and Long Interrupt Examples

V on the IRQB pin after the last time that the NMI interrupt was serviced or the chip was reset. The NMI is a priority level 3 interrupt and cannot be masked. Only RESET and Illegal Instruction have higher priority than NMI. NMI is reserved for hardware development and should not be used as a general-purpose interrupt pin. Continued use of this interrupt can cause damage to the chip (see the DSP56001 Advance Information Data Sheet (AD11290)). NMI has been provided strictly as an aid to the developer. The hardware reset address vector may point to internal (P:\$0000) or external (P:\$E000) program memory, determined by the value of the MODA and MODB pins when the RESET pin is deasserted.

The NMI, trace, and software interrupt (SWI) instructions are used for debugging and development purposes. The SWI instruction is useful for implementing breakpoints. Tracing is entered after turning on the trace flag in the SR. During tracing, a trace interrupt will be generated after each instruction is executed, thereby creating a single-step feature.

Internally, the peripheral registers are accessed through the global data bus. All on-chip peripherals use the same interrupt request interface mechanism. Each peripheral provides a single interrupt request line to the PIC and receives two lines: vector read and interrupt acknowledge. Each peripheral possesses more than one interrupt source (see Table 6-1); therefore, interrupt arbitration between internal peripherals must be handled by the peripheral according to its own predefined IPL. The PIC arbitrates between the different I/O peripherals: when one of them is selected, the peripheral supplies the correct vector address to the PIC. The host command vector in the host interface (see CHAPTER 10 PORT B) can be programmed to point to any of the 32 starting addresses, including 13 routines designated specifically as host commands and located at locations P:\$0024-P:\$003C. The default value set in the host command vector register during a reset is \$0024.

## 6.2.4 Instruction Pipeline

The program control unit implements a three-level pipelined architecture in which concurrent instruction fetch, decode, and execution occur. The fact that the pipelined operation remains essentially hidden from the user makes programming straightforward. The pipeline is illustrated in Figure 6-4. The first instruction, I1, should be interpreted as follows: multiply the contents of X0 by the contents of Y0, add the product to the contents already in accumulator A, round the result to the “nearest even,” store the result back in accumulator A, move the contents in X data memory (pointed to by R0) into X0; postincrement R0; move the contents in Y data memory (pointed to by R4) into Y1; postincrement R4. The second instruction, I2, should be interpreted as follows: clear accumulator A; move the contents in X0 into the location in X data memory pointed to by R0; postincrement R0; before the clear operation, move the contents in accumulator A into the location in Y data memory pointed to by R4; postdecrement R4. The third instruction, I3, is the same as I1, except a rounding operation is not performed. The operations of each of the execution units and all initial conditions necessary to follow the execution of the instruction sequence

### 6.3 CLOCK OSCILLATOR

The DSP56000/DSP56001 uses a four-phase clock for instruction execution; therefore, the clock runs at twice the instruction execution rate. The clock can be provided by an internal oscillator (see Figure 6-1) by connecting an external crystal between XTAL and EXTAL or by an external oscillator connected to EXTAL.

### 6.4 PROGRAMMING MODEL

The program control unit features LA and LC registers dedicated to supporting the hardware DO loop instruction in addition to the standard program flow-control resources, such as a PC, complete SR, and SS. With the exception of the PC, all registers are read/write to facilitate system debugging. Figure 6-5 shows the program control unit programming model with the six registers and SS. The following paragraphs give a detailed description of each register.

#### 6.4.1 Program Counter

This 16-bit register contains the address of the next location to be fetched from program memory space. The PC can point to instructions, data operands, or addresses of operands. References to this register are always inherent and are implied by most instructions.



EXAMPLE: PROGRAM SEGMENT

```

11 MACR X0,Y1,A X:(R0)+,X0 Y:(R4)+,Y1
12 CLR A X0,X:(R0)+ A,Y:(R4)-
13 MAC X0,Y1,A X:(R0)+,X0 Y:(R4)+,Y1
  
```

| INSTRUCTION FETCH<br>INSTRUCTION DECODE<br>INSTRUCTION EXECUTION |   | 11 | 12<br>11 | 13<br>12<br>11  | 14<br>13<br>12  | 15<br>14<br>13  |
|--|---|----|----------|---|---|---|
| PARALLEL OPERATIONS  | INITIAL CONDITIONS  |    |          |   |   |   |
| ADDRESS UPDATE (AGU)   | R0=\$0005<br>R4=\$0008  |    |          | R0=5+1<br>R4=8+1  | R0=6+1<br>R4=9-1  | R0=7+1<br>R4=8+1  |
| INSTRUCTION EXECUTION<br><br>(DATA ALU)                          | A:<br>A2=\$00<br>A1=\$000066<br>A0=\$000000<br><br>X0=\$400000<br>Y1=\$000077 |    |          | A:<br>A2=\$00<br>A1=\$0000A2<br>A0=\$000000<br><br>X0=\$000005<br>Y1=\$000008 | A:<br>A2=\$00<br>A1=\$000000<br>A0=\$000000<br><br>X0=\$000005<br>Y1=\$000008 | A:<br>A2=\$00<br>A1=\$000000<br>A0=\$000050<br><br>X0=\$000007<br>Y1=\$000008 |
| X MEMORY AT ADDRESS<br>\$0005<br>\$0006<br>\$0007                | DATA<br>\$000005<br>\$000006<br>\$000007                                      |    |          | \$000005<br>\$000006<br>\$000007  | \$000005<br>\$000005<br>\$000007  | \$000005<br>\$000005<br>\$000007  |
| Y MEMORY AT ADDRESS<br>\$0008<br>\$0009                          | DATA<br>\$000008<br>\$000009  |    |          | \$000008<br>\$000009  | \$000008<br>\$0000A2  | \$000008<br>\$0000A2  |

Figure 6-4 Three-Stage Pipeline

PROGRAM CONTROL UNIT

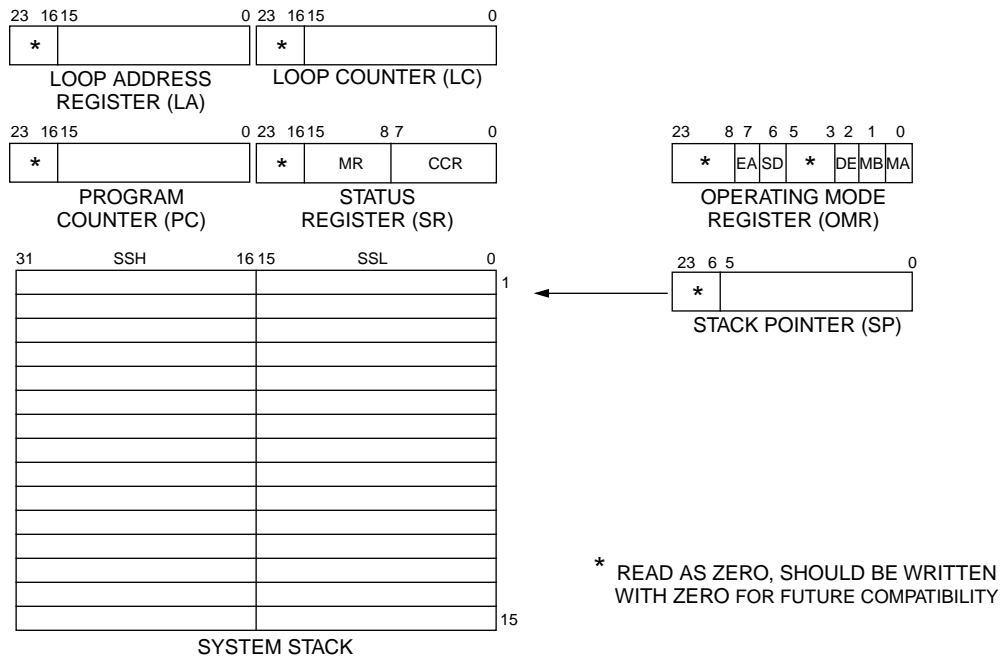


Figure 6-5 Program Control Unit Programming Model

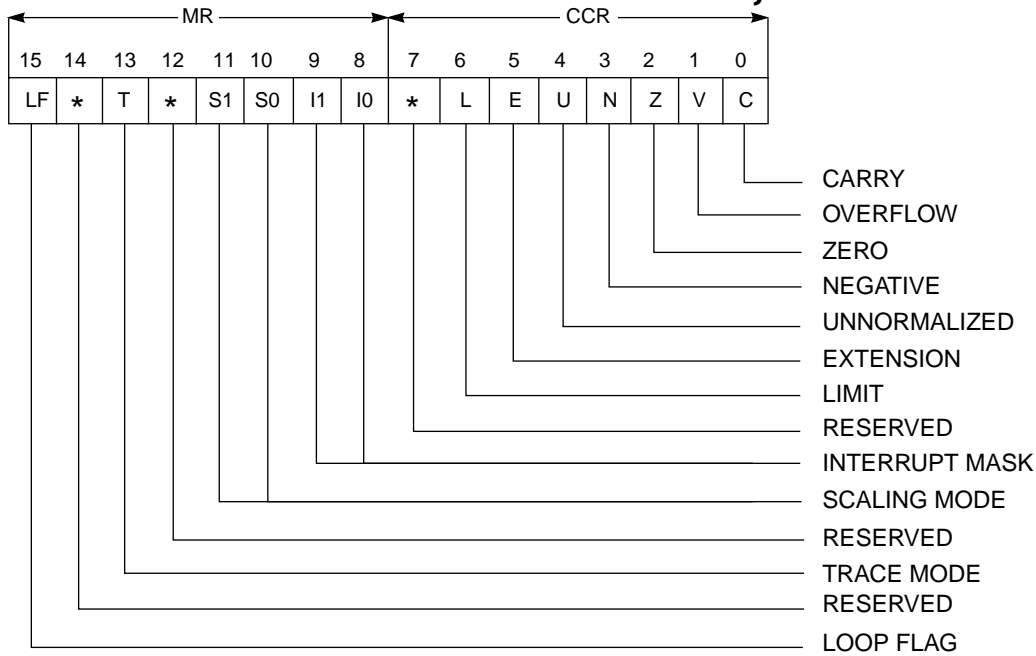
This special-purpose address register is stacked when program looping is initialized, when a JSR is performed, or when interrupts occur (except for no-overhead fast interrupts).

6.4.2 Status Register

The 16-bit SR consists of a mode register (MR) in the high-order eight bits and a condition code register (CCR) in the low-order eight bits. The SR is stacked when program looping is initialized, when a JSR is performed, or when interrupts occur, (except for no-overhead fast interrupts). The SR format is shown in Figure 6-6.

The MR is a special-purpose control register defining the current system state of the processor. The MR bits are affected by processor reset, exception processing, the DO, end current DO loop (ENDDO), return from interrupt (RTI), and SWI instructions and by instructions that directly reference the MR register =m OR immediate to control register (ORI) and AND immediate to control register (ANDI). During processor reset, the interrupt mask bits of the MR will be set; the scaling mode bits, loop flag, and trace bit will be cleared.

The CCR is a special-purpose control register that defines the current user state of the processor. The CCR bits are affected by data arithmetic logic unit (ALU) operations, parallel move operations, and by instructions that directly reference the CCR (ORI and ANDI). The CCR bits are not affected by parallel move operations unless data limiting occurs when reading the A or B accumulators. During processor reset, all CCR bits are



\* Written as don't care; read as zero

**Figure 6-6 Status Register Format**

cleared.

**6.4.2.1 Carry (Bit 0)**

The carry (C) bit is set if a carry is generated out of the MSB of the result in an addition. This bit is also set if a borrow is generated in a subtraction. The carry or borrow is generated from bit 55 of the result. The carry bit is also affected by bit manipulation, rotate, and shift instructions. Otherwise, this bit is cleared.

**6.4.2.2 Overflow (Bit 1)**

The overflow (V) bit is set if an arithmetic overflow occurs in the 56-bit result. This bit indicates that the result cannot be represented in the accumulator register; thus, the register has overflowed. Otherwise, this bit is cleared.

**6.4.2.3 Zero (Bit 2)**

The zero (Z) bit is set if the result equals zero; otherwise, this bit is cleared.

**6.4.2.4 Negative (Bit 3)**

The negative (N) bit is set if the MSB (bit 55) of the result is set; otherwise, this bit is cleared.

**6.4.2.5 Unnormalized (Bit 4)**

The unnormalized (U) bit is set if the two MSBs of the most significant product (MSP) portion of the result are identical. Otherwise, this bit is cleared. The MSP portion of the A or B accumulators, which is defined by the scaling mode and the U bit, is computed as

follows:

## Freescale Semiconductor, Inc.

| S1 | S0 | Scaling Mode | U Bit Computation                                       |
|----|----|--------------|---|
| 0  | 0  | No Scaling   | $U = \overline{(\text{Bit } 47 \oplus \text{Bit } 46)}$ |
| 0  | 1  | Scale Down   | $U = \overline{(\text{Bit } 48 \oplus \text{Bit } 47)}$ |
| 1  | 0  | Scale Up     | $U = \overline{(\text{Bit } 46 \oplus \text{Bit } 45)}$ |

### 6.4.2.6 Extension (Bit 5)

The extension (E) bit is cleared if all the bits of the integer portion of the 56-bit result are all ones or all zeros; otherwise, this bit is set. The integer portion, defined by the scaling mode and the E bit, is computed as follows:

| S1 | S0 | Scaling Mode | Integer Portion      |
|----|----|--------------|----------------------|
| 0  | 0  | No Scaling   | Bits 55,54.....48,47 |
| 0  | 1  | Scale Down   | Bits 55,54.....49,48 |
| 1  | 0  | Scale Up     | Bits 55,54.....47,46 |

If the E bit is cleared, then the low-order fraction portion contains all the significant bits; the high-order integer portion is just sign extension. In this case, the accumulator extension register can be ignored. If the E bit is set, it indicates that the accumulator extension register is in use.

### 6.4.2.7 Limit (Bit 6)

The limit (L) bit is set if the overflow bit is set. The L bit is also set if the data shifter/limiter circuits perform a limiting operation; otherwise, it is not affected. The L bit is cleared only by a processor reset or by an instruction that specifically clears it, which allows the L bit to be used as a latching overflow bit (i.e., a "sticky" bit). L is affected by data movement operations that read the A or B accumulator registers.

### 6.4.2.8 Interrupt Masks (Bits 8 and 9)

The interrupt mask bits, I1 and I0, reflect the current IPL of the processor and indicate the IPL needed for an interrupt source to interrupt the processor. The current IPL of the processor can be changed under software control. The interrupt mask bits are set during hardware reset but not during software reset.

| I1 | I0 | Exceptions Permitted | Exceptions Masked |
|----|----|----------------------|-------------------|
| 0  | 0  | IPL 0,1,2,3          | None              |
| 0  | 1  | IPL 1,2,3            | IPL 0             |
| 1  | 0  | IPL 2,3              | IPL 0,1           |
| 1  | 1  | IPL 3                | IPL 0,1,2         |

### 6.4.2.9 Scaling Mode (Bits 10 and 11)

The scaling mode bits, S1 and S0, specify the scaling to be performed in the data ALU shifter/limiter and the rounding position in the data ALU multiply-accumulator (MAC). The

scaling modes are shown in the following table:

## Freescale Semiconductor, Inc.

| S1 | S0 | Rounding Bit | Scaling Mode                              |
|----|----|--------------|---|
| 0  | 0  | 23           | No Scaling                                |
| 0  | 1  | 24           | Scale down (1-Bit Arithmetic Right Shift) |
| 1  | 0  | 22           | Scale Up (1-Bit Arithmetic Left Shift)    |
| 1  | 1  | —            | Reserved for Future Expansion             |

The shifter/limiter scaling mode affects data read from the A or B accumulator registers out to the XDB and YDB. Different scaling modes can be used with the same program code to allow dynamic scaling. One application of dynamic scaling is to facilitate block floating-point arithmetic. The scaling mode also affects the MAC rounding position to maintain proper rounding when different portions of the accumulator registers are read out to the XDB and YDB. The scaling mode bits, which are cleared at the start of a long interrupt service routine, are also cleared during a processor reset.

### 6.4.2.10 Trace Mode (Bit 13)

The trace mode (T) bit specifies the tracing function of the DSP. If the T bit is set at the beginning of any instruction execution, a trace exception will be generated after the instruction execution is completed. If the T bit is cleared, tracing is disabled and instruction execution proceeds normally. If a long interrupt is executed during a trace exception, the SR having the trace bit set will be stacked, and the trace bit in the SR is cleared (see SECTION 8 PROCESSING STATES for a complete description of a long interrupt operation). The T bit is also cleared during processor reset.

### 6.4.2.11 Reserved Status (Bits 7, 12, 14)

These bits, which are reserved for future expansion, will read as zero during DSP read operations.

### 6.4.2.12 Loop Flag (Bit 15)

The loop flag (LF) bit, set when a program loop is in progress, enables the detection of the end of a program loop. The LF is the only SR bit that is restored when terminating a program loop. Stacking and restoring the LF when initiating and exiting a program loop, respectively, allow the nesting of program loops. At the start of a long interrupt service routine, the SR (including the LF) is pushed on the SS and the

SR LF is cleared. When returning from the long interrupt with an RTI instruction, the SS is pulled and the LF is restored. During a processor reset, the LF is cleared.

### 6.4.3 Operating Mode Register

The OMR is a 24-bit register (only five bits are defined) that sets the current operating mode of the processor (i.e., the memory maps for program and data memories as well as the start-up procedure). The OMR bits are only affected by processor reset and by instructions directly referencing the OMR: ANDI, ORI, and MOVEC. During processor reset, the chip operating mode bits, MB and MA, will be loaded from the external mode select pins B and A, respectively. The data ROM enable (DE) bit will be cleared, disabling the X and Y on-chip lookup-table ROMs. The OMR format is shown in Figure 6-7. Table 6-5 summarizes the DSP56000 operating modes and Table 6-3 summarizes the DSP56001 operating modes and their respective effects on the memory map. Table 6-4 shows how the DE bit in the OMR affects the X and Y memory maps.

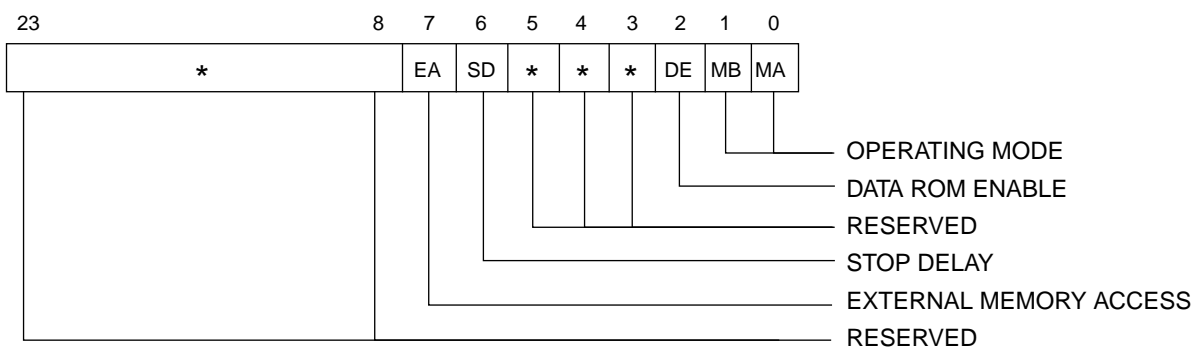


Figure 6-7 OMR Format

Table 6-5 DSP56000/DSP56001 Operating Mode Summary

| Operating Mode | MB | MA | DSP56000 Program Memory Map  |                 |                   |
|----------------|----|----|--|-----------------|-------------------|
|                |    |    | Internal ROM   | External        | Reset             |
| 0              | 0  | 0  | \$0000 — \$0EFF  | \$0200 — \$FFFF | Internal — \$0000 |
| 1              | 0  | 1  | Mode 1 is not a valid mode for the DSP56000. Attempting to put the DSP56000 in mode 1 will put it into mode 0. |                 |                   |
| 2              | 1  | 0  | \$0000 — \$0EFF  | \$0FFF — \$FFFF | External — \$E000 |
| 3              | 1  | 1  | —  | \$0000 — \$FFFF | External — \$0000 |

**Table 6-4 DSP56000/56001 DE Memory Control**

|    |                 |          |
|----|-----------------|----------|
| DE | Data Memory Map |          |
|    | Y Memory        | X Memory |

**6.4.3.1 Chip Operating Mode (Bits 0 and 1)**

The chip operating mode bits, MB and MA, indicate the bus expansion mode of the DSP56000/DSP56001. On processor reset, these bits are loaded from the external mode select pins, MODB and MODA, respectively. After the DSP leaves the reset state, MB and MA can be changed under program control. The “secure DSP56000” is an exception. The external mode select pins, MODB and MODA, are disabled on the “secure DSP56000” and are only used for interrupts as IRQA and IRQB. The operating modes are shown in the following table:

| MB | MA | Chip Operating Mode               |
|----|----|-----------------------------------|
| 0  | 0  | Single-Chip Nonexpanded           |
| 0  | 1  | Special Bootstrap (DSP56001 Only) |
| 1  | 0  | Normal Expanded                   |
| 1  | 1  | Development Expanded              |

**6.4.3.2 Data ROM Enable (Bit 2)**

The DE bit enables the two, on-chip, 256;ts24 data ROMs located at addresses \$0100–\$01FF in the X and Y memory spaces. When DE is cleared, the \$0100–\$01FF address space is part of the external X and Y data spaces, and the on-chip data ROMs are dis-

abled.

**6.4.3.3 Stop Delay (Bit 6)**

The STOP instruction causes the DSP56000/DSP56001 to indefinitely suspend processing in the middle of the STOP instruction (see **SECTION 8 PROCESSING STATES**). When exiting the stop state, if the stop delay bit is zero, a 64K clock cycle delay (i.e., 131,072 T states) is selected before continuing the stop instruction cycle. However, if the stop delay bit is one, the delay before continuing the instruction cycle is 16 T states. The long delay allows a clock stabilization period for the internal clock to begin oscillating and to stabilize. When a stable external clock is used, the shorter delay allows faster start-up of the DSP.

**6.4.3.4 External Memory Access (Bit 7)**

The external memory access mode bit selects the function of two of the port A control pins. The DSP56000/DSP56001 comes out of reset with these pins defined as bus request/bus grant (BR/BG) =m i.e., bit 7 is cleared. When bit 7 is clear, wait states are only introduced into the port A timing by using the bus control register (BCR). Additional information on the BCR can be found in CHAPTER 10 PORT B. When bit 7 is set under program control (using ANDI, ORI, or MOVEC), these pins are defined as bus strobe (BS) and wait (WT). In this mode, wait states are introduced into port A timing by using either the BCR or asserting WT. BR and BG allow the DSP56000/DSP56001 to give the external bus to an external device, thus preventing bus conflicts. BS and WT allow the DSP56000/DSP56001 to work with asynchronous devices (bus arbitrators) on port A.

**Table 6-6 DSP56001 Operating Mode Summary**

| Operating Mode | MB | MA | DSP56001 Program Memory Map   |                 |                   |
|----------------|----|----|---|-----------------|-------------------|
|                |    |    | Internal RAM  | External        | Reset             |
| 0              | 0  | 0  | \$0000 — \$01FF   | \$0200 — \$FFFF | Internal — \$0000 |
| 1              | 0  | 1  | Special bootstrap mode; after program RAM loading, mode 2 is automatically selected but PC = \$0000 |                 |                   |
| 2              | 1  | 0  | \$0000 — \$01FF   | \$0200 — \$FFFF | External — \$E000 |
| 3              | 1  | 1  | —   | \$0000 — \$FFFF | External — \$0000 |

The definition of the control pins is summarized in the following table:

| OMR Bit 7   | $\overline{BR}$ Pin (Input)     | $\overline{BG}$ Pin (Output)   |
|-------------|---------------------------------|--------------------------------|
| 0 (Default) | Bus Request ( $\overline{BR}$ ) | Bus Grant ( $\overline{BG}$ )  |
| 1           | Wait ( $\overline{WT}$ )        | Bus Strobe ( $\overline{BS}$ ) |

### 6.4.3.5 Reserved OMR Bits (Bits 3–5 and 8–23)

These OMR bits, reserved for future expansion, will read as zero during DSP read operations.

### 6.4.4 Loop Address Register

The contents of the LA register indicate the location of the last instruction word in a program loop. This register is stacked into the SSH by a DO instruction and is unstacked by end-of-loop processing or by execution of an ENDDO instruction. When the instruction at the address contained in this register is fetched, the contents of the LC register are checked. If the contents are not one, the LC is decremented, and the next instruction is taken from the address at the top of the SS; otherwise, the PC is incremented, the loop flag is restored (pulled from the SS), the SS is purged, the LA and LC registers are pulled from the SS and restored, and instruction execution continues normally. The LA register, a read/write register, is written by a DO instruction and read by the SS when stacking the register. Since the LC register can be accessed under program control, the number of times a loop has been executed can be determined.

### 6.4.5 Loop Counter Register

The LC register is a special 16-bit counter used to specify the number of times a hardware program loop is to be repeated. This register is stacked into the SSL by a DO instruction and unstacked by end-of-loop processing or by execution of an ENDDO instruction. When the end of a hardware program loop is reached, the contents of the LC register are tested for one. If the LC is one, the program loop is terminated, and the LC register is loaded with the previous LC contents stored on the SS. If LC is not one, it is decremented and the program loop is repeated. The LC can be read under program control, which allows the number of times a loop will be executed to be monitored/changed dynamically. The LC is also used in the REP instruction.

### 6.4.6 System Stack

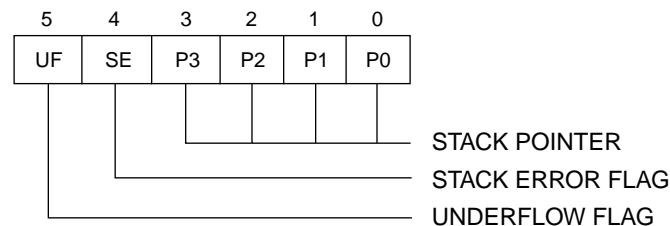
The SS is a separate 15x32-bit internal memory divided into two banks: SSH and SSL, each 16 bits wide. The SSH stores the PC contents, and the SSL stores the SR contents for subroutine calls and long interrupts. The SS will also store the LA and LC registers in addition to the PC and SR registers for program looping. The SS is in stack memory

space; its address is always inherent and implied by the current instruction.

The contents of the PC and SR register are pushed on the top location of the SS when a subroutine call or long interrupt occurs. When a return from subroutine (RTS) occurs, the contents of the top location in the SS are pulled and put in the PC; the SR is not affected. When an RTI occurs, the contents of the top location in the SS are pulled to both the PC and SR.

The SS is also used to implement no-overhead nested hardware DO loops. When the DO instruction is executed, the LA:LC are pushed on the SS, then the PC:SR are pushed on the SS. Since each SS location can be addressed as separate 16-bit registers (SSH and SSL), software stacks can be created for unlimited nesting.

Up to 15 long interrupts, seven DO loops, 15 JSRs, or combinations of these can be accommodated by the SS. When the SS limit is exceeded, a nonmaskable stack error interrupt occurs, and the PC is pushed to SS location zero, which is not implemented in hardware. The PC will be lost, and there will be no SP from the stack interrupt routine to the program that was executing when the error occurred.



**Figure 6-8 SP Register Format**

**6.4.7 Stack Pointer Register**

The 6-bit SP register indicates the location of the top of the SS and the status of the SS (underflow, empty, full, and overflow). The SP register is referenced implicitly by some instructions (DO, REP, JSR, RTI, etc.) or directly by the MOVEC instruction. The SP register format, shown in Figure 6-8, is described in the following paragraphs. The SP register is implemented as a 6-bit counter that addresses (selects) a 15-location stack with its four LSBs. The possible SP values, shown in Figure 6-9, are described in the following paragraphs

**6.4.7.1 Stack Pointer (Bits 0–3)**

The SP points to the last used location on the SS. Immediately after hardware reset,

these bits are cleared (SP=0), indicating that the SS is empty.

Data is pushed onto the SS by incrementing the SP, then writing data to the location pointed to by the SP. An item is pulled off the stack by copying it from the location pointed to by the SP and then by decrementing SP.

### 6.4.7.2 Stack Error Flag (Bit 4)

The stack error flag indicates that a stack error has occurred, and the transition of the stack error flag from zero to one causes a priority level-3 stack error exception (see Section 6.4.7.1 for additional information).

When the stack is completely full, the SP reads 001111, and any operation that pushes data onto the stack will cause a stack error exception to occur. The SR will read 010000 (or 010001 if an implied double push occurs).

Any implied pull operation with SP equal to zero will cause a stack error exception, and the SP will read 111111 (or 111110 if an implied double pull occurs). The stack error bit is set as shown in Figure 6-9.

The stack error flag is a “sticky bit” which, once set, remains set until cleared by the user. There is a sequence of instructions which can cause a stack overflow which, without the sticky bit, would not be detected because the stack pointer is decremented before the stack error interrupt is taken. The sticky bit keeps the stack error bit set until cleared by the user by writing a zero to SP bit 4. It also latches the overflow/underflow bit so that it cannot be changed by stack pointer increments or decrements as long as the stack error is set. The overflow/underflow bit remains latched until the first move to SP is executed.

**Note:** When SP is zero (stack empty), instructions that read the stack without SP post-decrement and instructions that write to the stack without SP preincrement do not cause

| UF | SE | P3  | P2 | P1 | P0 |   |
|----|----|-----|----|----|----|---|
| 1  | 1  | 1   | 1  | 1  | 0  | ← STACK UNDERFLOW CONDITION AFTER DOUBLE PULL |
| 1  | 1  | 1   | 1  | 1  | 1  | ← STACK UNDERFLOW CONDITION                   |
| 0  | 0  | 0   | 0  | 0  | 0  | ← STACK EMPTY (RESET); PULL CAUSES UNDERFLOW  |
| 0  | 0  | 0   | 0  | 0  | 1  | ← STACK LOCATION 1                            |
|    |    | ••• |    |    |    |   |
|    |    | ••• |    |    |    |   |
|    |    | ••• |    |    |    |   |
| 0  | 0  | 1   | 1  | 1  | 0  | ← STACK LOCATION 14                           |
| 0  | 0  | 1   | 1  | 1  | 1  | ← STACK LOCATION 15; PUSH CAUSES OVERFLOW     |
| 0  | 1  | 0   | 0  | 0  | 0  | ← STACK OVERFLOW CONDITION                    |
| 0  | 1  | 0   | 0  | 0  | 1  | ← STACK OVERFLOW CONDITION AFTER DOUBLE PUSH  |

Figure 6-9 SP Register Values

a stack error exception (i.e., 1) DO SSL,xxxx 2) REP SSL 3) MOVEC or move peripheral data (MOVEP) when SSL is specified as a source or destination).

### 6.4.7.3 Underflow Flag (Bit 5)

The underflow flag is set when a stack underflow occurs. The stack underflow flag is a “sticky bit” when the stack error flag is set i.e., when the stack error flag is set, the underflow flag will not change state. The combination of “underflow=1” and “stack error=0” is an illegal combination and will not occur unless it is forced by the user. If this condition is forced by the user, the hardware will correct itself based on the result of the next stack operation. Also see the description for the stack error flag (Section 6.4.7.2) for additional information.

### 6.4.7.4 Reserved Stack Pointer Registration (Bits 6–23)

Any unimplemented SP register bits are reserved for future expansion and will read as zero during DSP56000/DSP56001 read operations.

### 6.4.8 DSP56000/DSP56001 Programming Model Summary

The complete programming model for the DSP56000/DSP56001 central processor is shown in Figure 6-10. SECTION 9 PORT A, SECTION 10 PORT B, and SECTION 11 PORT C describe in detail the programming model for the peripherals and external memory control (number of wait states).



