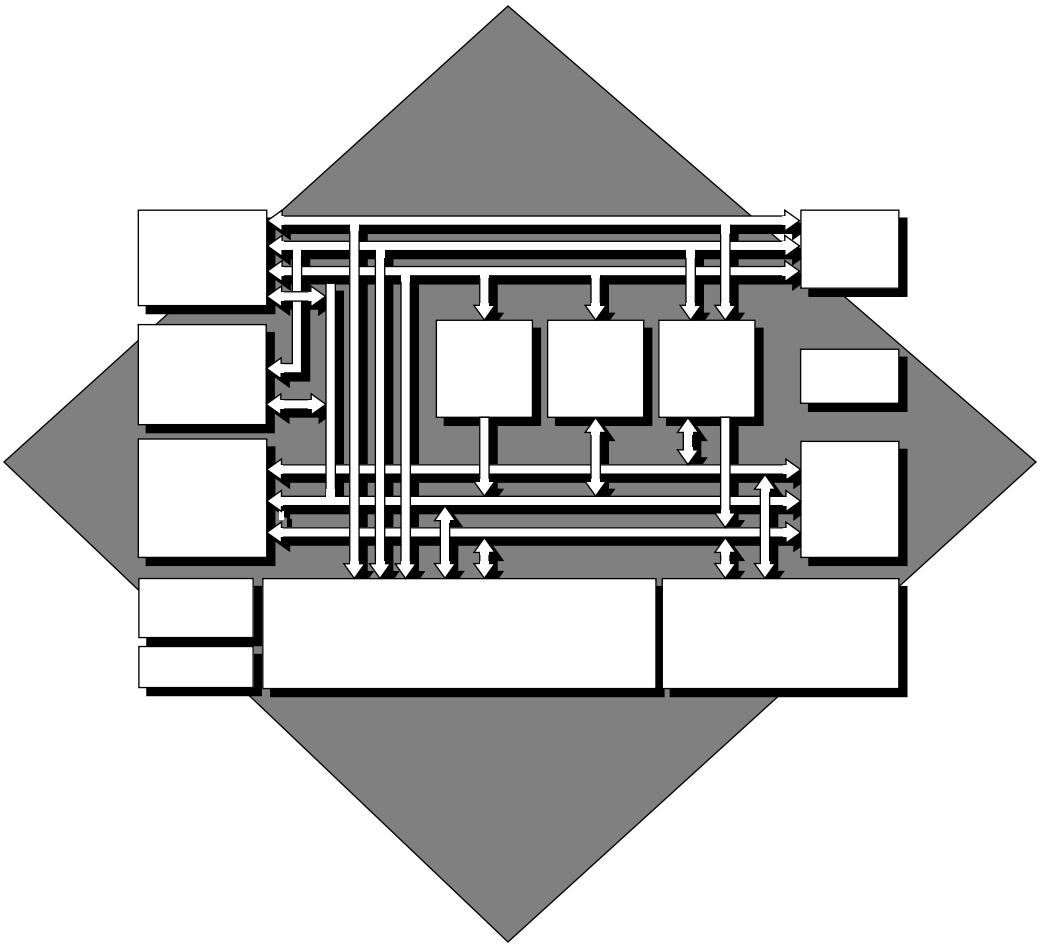
SECTION 1

DSP56166 OVERVIEW



SECTION CONTENTS

1.1 INTRODUCTION

This manual is intended to be used with the DSP56100 Family Manual (see Figure 1-1). The DSP56100 Family Manual provides a description of the components of the DSP5616 core that are common to all DSP56100 family processors and includes a detailed description of the basic DSP56100 family instruction set. The DSP56166 User's Manual provides a brief overview of the core processor and a detailed descriptions of the memory and peripherals that are specific to the DSP56166.

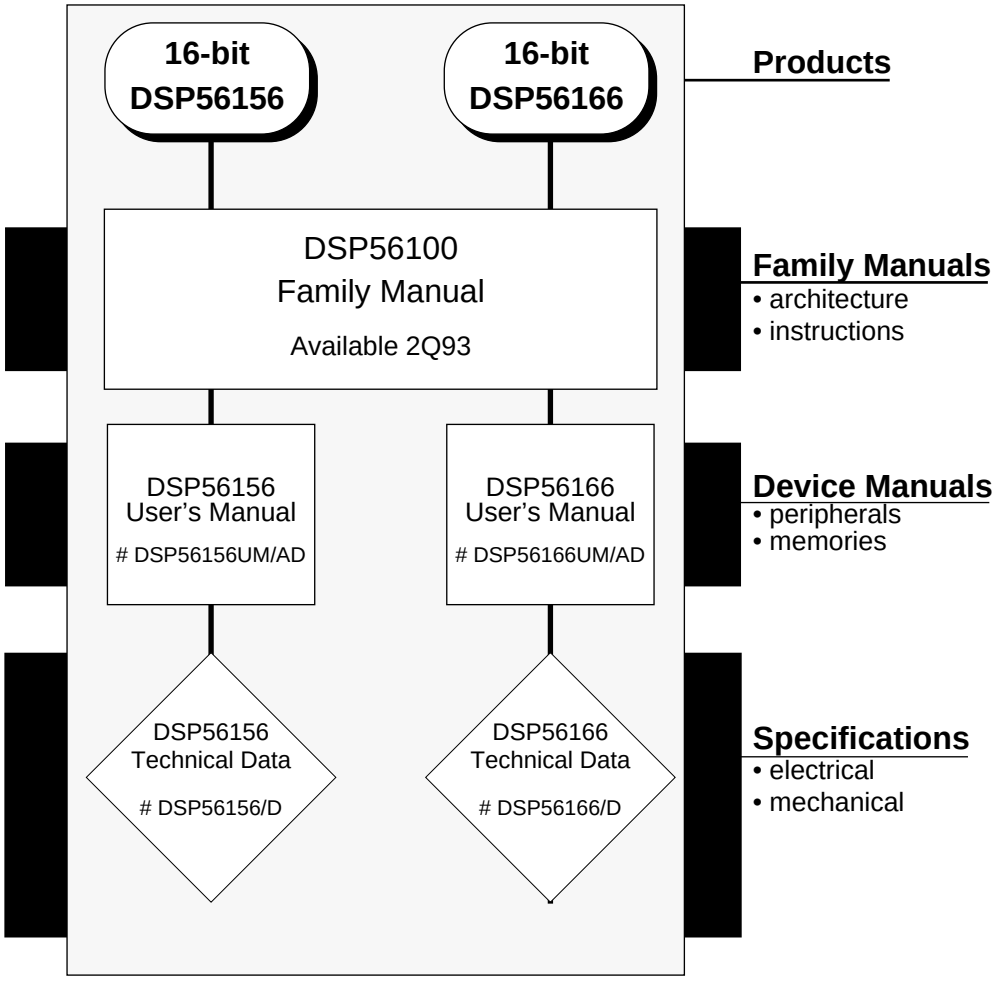


Figure 1-1 DSP56100 Family Product Literature

A general block diagram of the DSP56166 is shown in Figure 1-2. It is available as a RAM based or ROM based part (see Section 3.3 for information on the ROM based part). The DSP56166 is optimized for applications such as medium to low bit rate speech encoding but can also be used in many other types of applications.

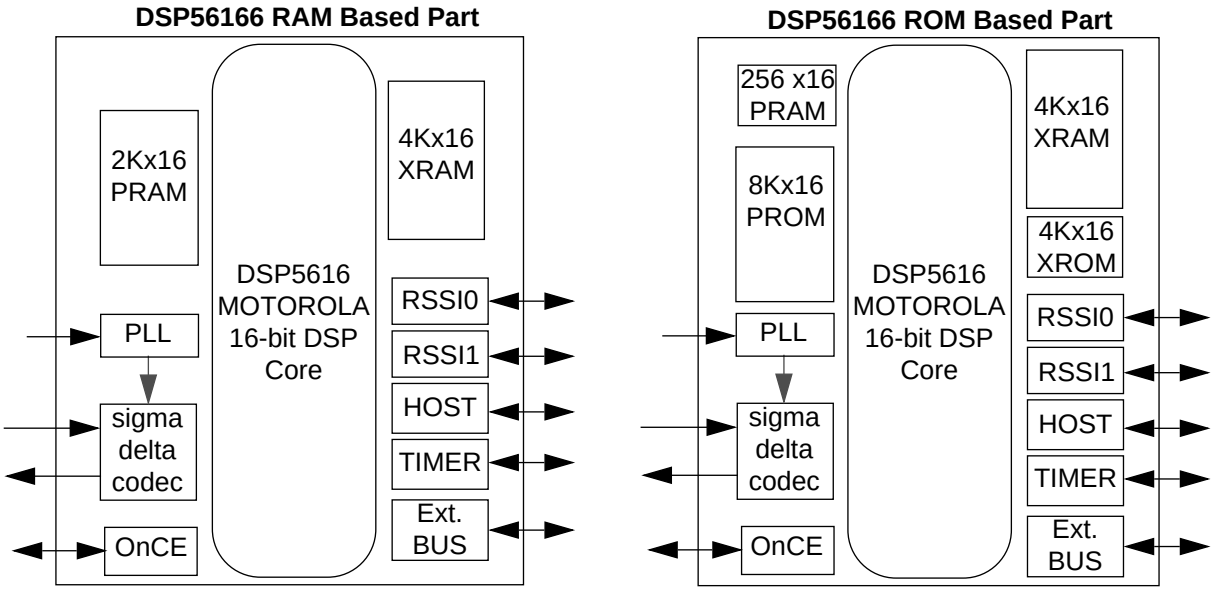


Figure 1-2 DSP56166 RAM and ROM Based Functional Block Diagram

Table 1-1 is a list of the DSP56166 primary features. The core features are common to any product using the DSP5616 Core. Figure 1-3 provides a more detailed block diagram of the DSP56166 RAM based part.

1.2 DSP5616 CORE BLOCK DIAGRAM DESCRIPTION

The heart of the DSP56166 architecture is a 16-bit multiple-bus core processor called the DSP5616 which was designed specifically for real-time digital signal processing (DSP).The overall core architecture is presented here and can be seen in Figure 1-4. For a detailed description of the core processor, see the **DSP56100 Family User's Manual**.

1.2.1 Data Buses

Data movement on the chip occurs over three bidirectional 16-bit buses: the X Data Bus (XDB), the Program Data Bus (PDB), and the Global Data Bus (GDB). Data transfer between the Data ALU and the X Data Memory occurs over the XDB when one memory access is performed or over the XDB and the GDB when two simultaneous memory reads are performed. All other data transfers occur over the Global Data Bus. Instruction word pre-fetches take place in parallel over the PDB. The bus structure supports general register to register, register to memory, memory to register, and memory to memory data movement and can transfer up to three 16-bit words in the same instruction cycle.

Table 1-1 DSP56166 Feature List

DSP5616 Core Features	DSP56166 On-chip Resources
• Up to 30 Million Instructions per Second (MIPS) at 60 MHz.– 33.3 ns instruction cycle • Single-cycle 16 x 16-bit parallel multiply-accumulate • 2 x 40-bit accumulators with extension byte • Fractional and integer arithmetic with support for multiprecision arithmetic • Highly parallel instruction set with unique DSP addressing modes • Nested hardware DO loops including infinite loops • Two instruction LMS adaptive filter loop • Fast auto-return interrupts • Three external interrupt request pins • Three 16-bit internal data buses and three 16-bit internal address buses • Programmable access time on the external bus • On-chip peripheral registers memory mapped in data memory space • Off-chip peripheral space with programmable access time memory mapped in data memory space • Low power wait and stop modes • On-Chip Emulation (OnCE) for unobtrusive, processor speed independent debugging • Operating frequency down to DC • 5V single power supply • Low power (HCMOS)	DSP56166 RAM Based Part: 4K x 16 on-chip data RAM 2K x16 on-chip program RAM One bootstrap ROM Bootstrap loading from external byte wide PROM, Host Interface, or Reduced Synchronous Serial Interface 0 (RSSI0) DSP56166 ROM Based Part: 4K x 16 on-chip data RAM 4K x16 on-chip data ROM 256 x 16 on-chip program RAM 8Kx16 on-chip program ROM DSP56166 RAM based and ROM Based Part: One external 16-bit address bus One external 16-bit data bus On-chip ΣΔ voice band codec (A/D-D/A) – Internal voltage reference (2/5 of positive power supply) – No off-chip components required 25 general purpose I/O pins On-chip, programmable PLL Byte-wide Host Interface with DMA support Two independent reduced synchronous serial interfaces One 16-bit timer 112 pin quad flat pack packaging

1.2.2 Address Buses

Addresses are specified for internal X Data Memory on two unidirectional 16-bit buses — X Address Bus One (XAB1) and X Address Bus Two (XAB2). Program memory addresses are specified on the PAB. External memory spaces are addressed via a single 16-bit, unidirectional address bus driven by a three input multiplexer that can select the XAB1, XAB2, or PAB. One instruction cycle is needed for each external memory access. There is no speed penalty if only one external memory space is accessed in an instruction and if no wait states are inserted in the external bus cycle. If two or three external memory spaces are accessed in a single instruction, there will be a one or two instruction cycle execution delay, respectively, or more if wait states are inserted on the external bus. A bus arbitrator controls external accesses, making it transparent to the user. See the DSP56100 Family User's Manual for additional information.

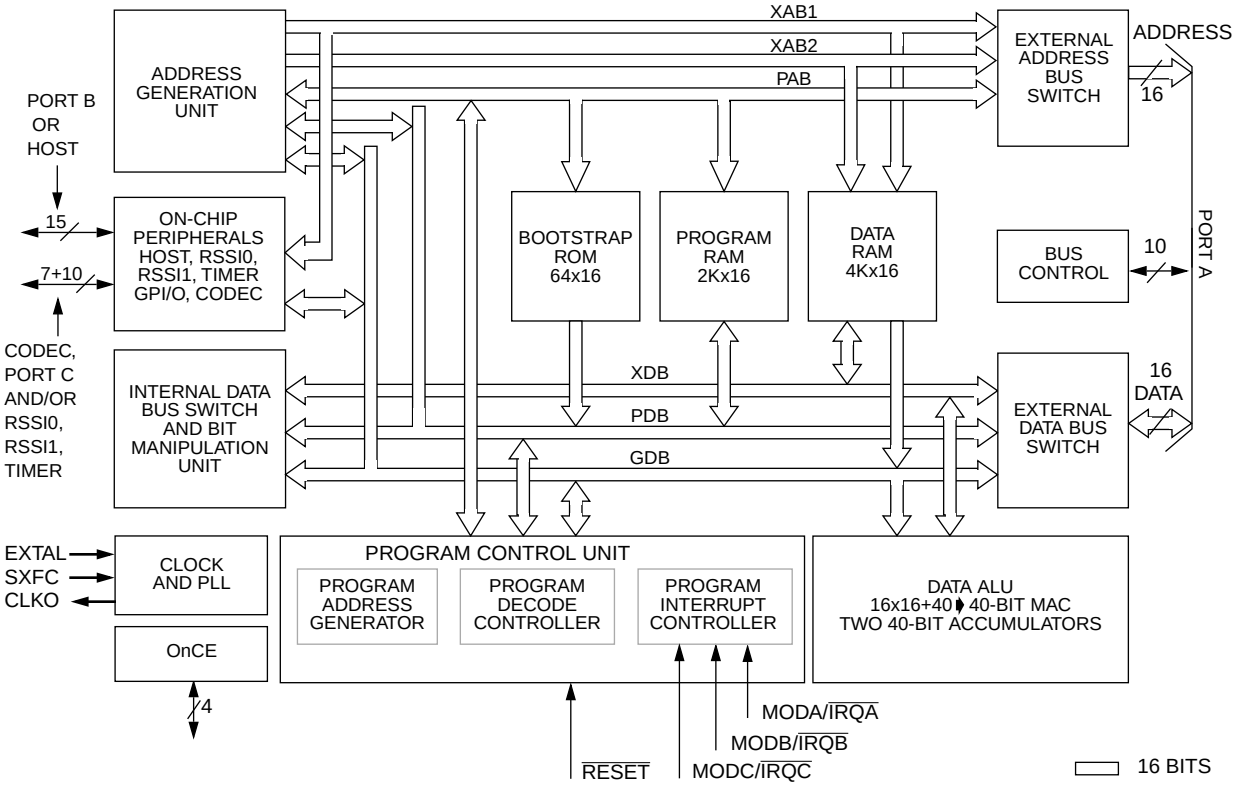


Figure 1-3 DSP56166 RAM Based Part Block Diagram

1.2.3 Data ALU

The Data ALU performs all arithmetic and logical operations on data operands and consists of:

- four 16-bit input registers,
- two 32-bit accumulator registers,
- two 8-bit accumulator extension registers,
- an accumulator shifter,
- an output shifter,
- one data bus shifter/limiter,
- one data bus limiter,
- and a parallel, single cycle, non-pipelined Multiply-Accumulator (MAC) unit.

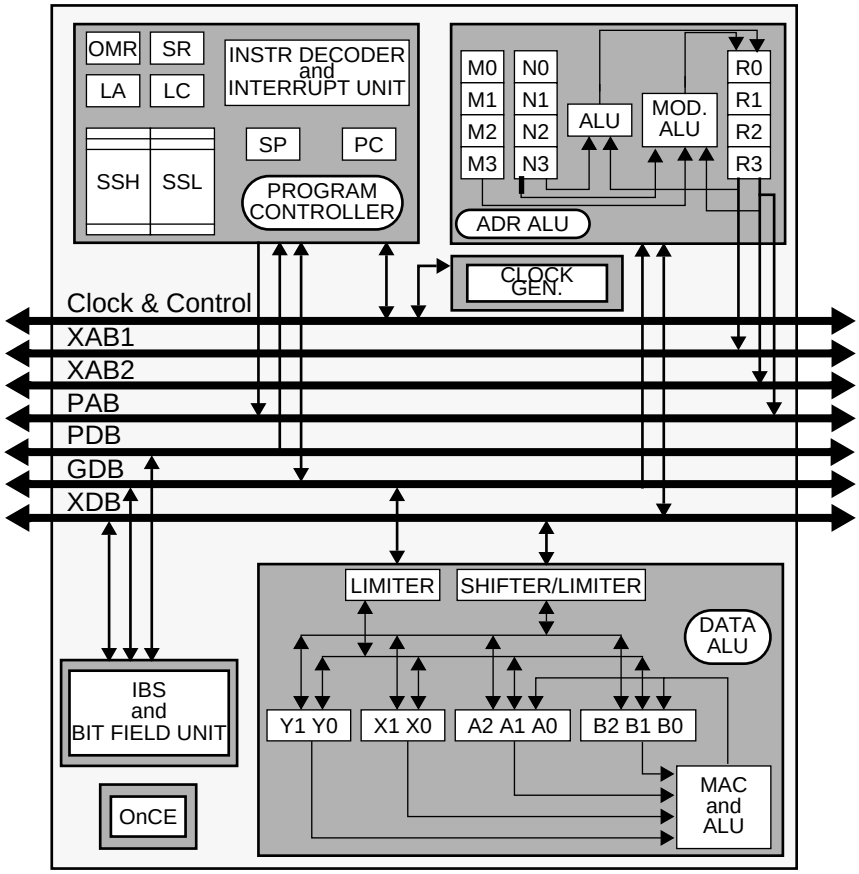


Figure 1-4 DSP5616 Core Block Diagram

Data ALU registers may be read or written via the XDB and GDB as 16-bit operands (see Figure 1-5). The Data ALU is capable of multiplication, multiply-accumulate with positive or negative accumulation, addition, subtraction, shifting, and logical operations in one instruction cycle. Data ALU arithmetic operations generally use fractional two's complement arithmetic. Some signed/unsigned and integer operations are also available. Data ALU source operands may be 16, 32, or 40 bits and may originate from input registers and/or accumulators. Data ALU results are always stored in one of the accumulators. The upper 16-bits of an accumulator can be used as a multiplier input. Arithmetic operations always have a 40-bit result and logical operations are performed on 16-bit operands yielding 16-bit results in one of the two accumulators.

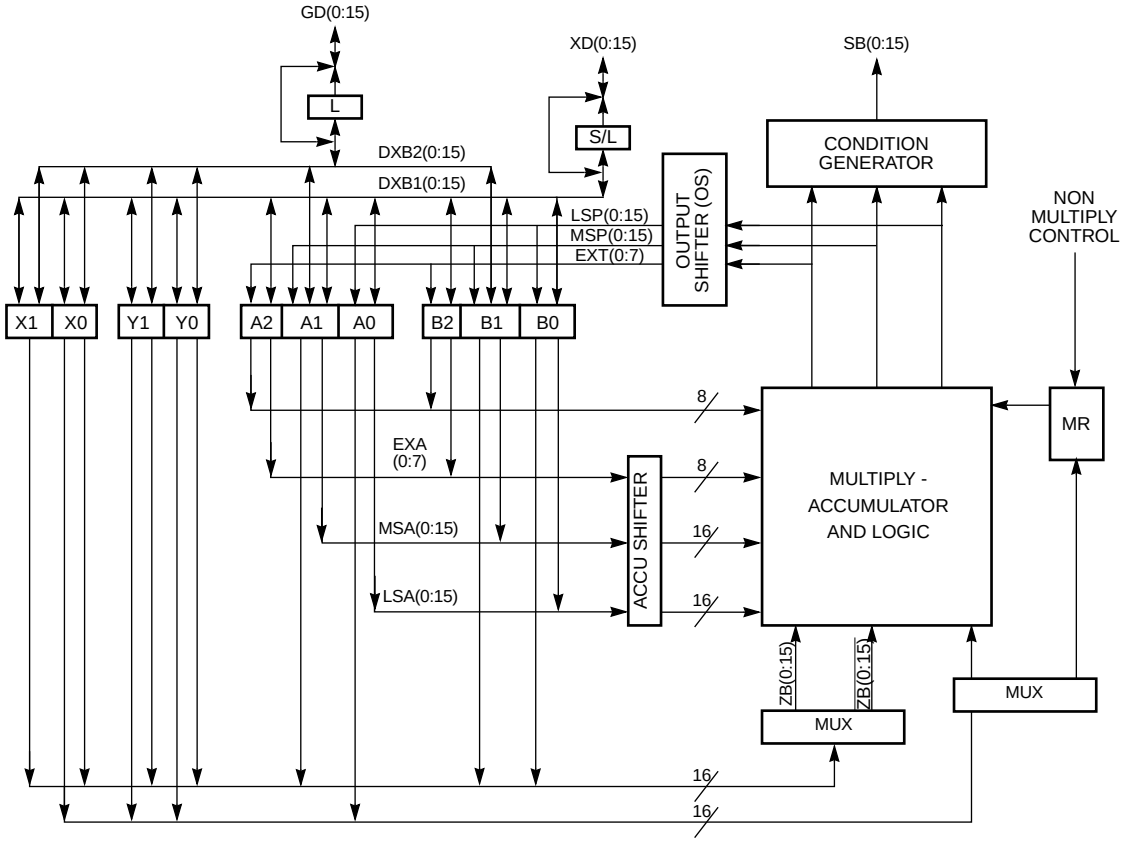


Figure 1-5 Data ALU Architecture Block Diagram

The DSP56166 supports the two's complement representation of binary numbers. Unsigned numbers are only supported by the multiply and multiply-accumulate instruction. For fractional arithmetic, the 31-bit product is added to the 40-bit contents of either the A or B accumulator. The 40-bit sum is stored back in the same accumulator. This multiply/accumulate is a single cycle operation (no pipeline). Integer operations always generate a 16-bit result located in the accumulator MSP (A1 or B1). Full precision integer operations are possible using the IMPY or IMAC instructions. Saturation arithmetic is provided to selectively limit overflow when reading a data ALU accumulator register.

The DSP56166 implements two types of rounding: convergent rounding and two's complement rounding. The type of rounding is selected by the status register rounding bit (R bit).

The logic unit in the MAC array performs the logical operations AND, OR, EOR, and NOT on data ALU registers. The logic unit is 16 bits wide and operates on data in the MSP portion of the accumulator. The LSP and EXT portions of the accumulator are not affected. See the DSP56100 Family User's Manual for additional information.

1.2.4 Address Generation Unit (AGU)

The AGU performs all address storage and effective address calculations necessary to address data operands in memory (see Figure 1-6). This unit operates in parallel with other chip resources to minimize address generation overhead. The AGU can implement three types of arithmetic: linear, modulo, and reverse carry. The Address ALU contains four address registers (R0-R3), four offset registers (N0-N3), and four modifier registers (M0-M3). The address registers are 16-bit registers which may contain addresses or data. Each address register may be output to the PAB and XAB1. R3 may be output to XAB2 when R0, R1, or R2 are output to XAB1. The modifier and offset registers are 16-bit registers which are normally used to control updating of the address registers. Any register can also be used as a general purpose register for storage of 16-bit data.

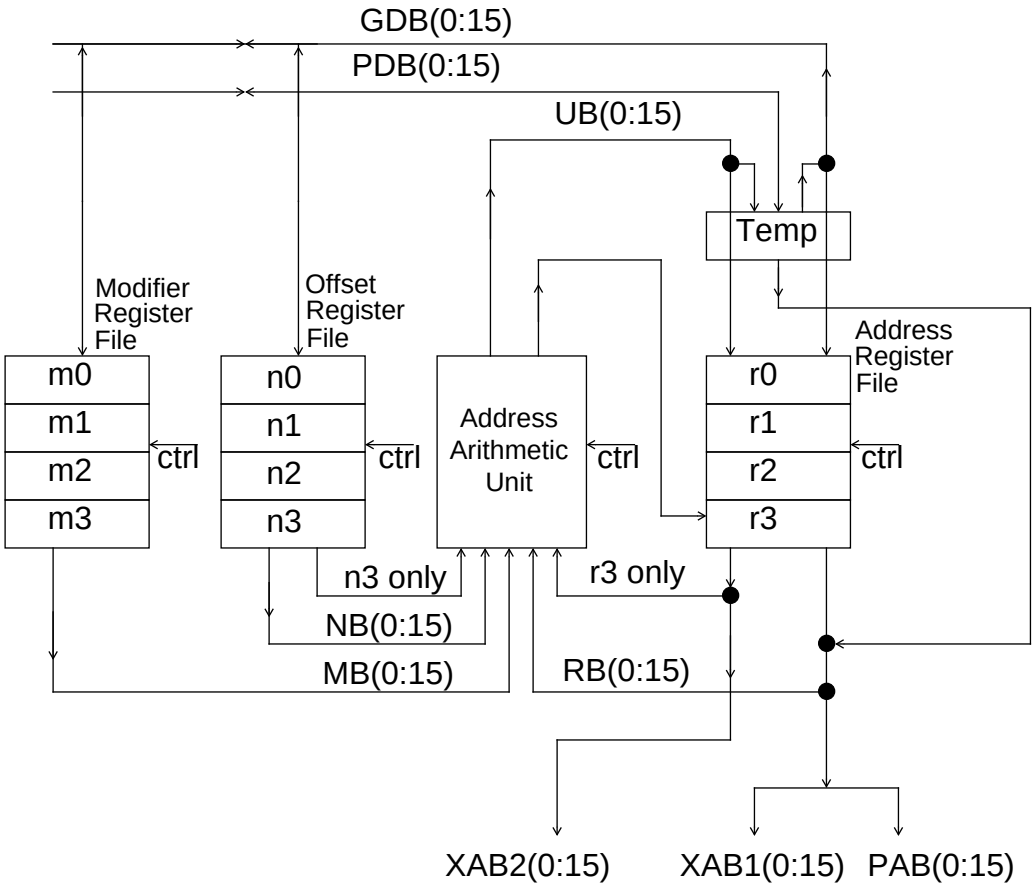


Figure 1-6 AGU Block Diagram

AGU registers may be read or written by the GDB as 16-bit operands. The AGU can generate two 16-bit addresses every instruction cycle: one for either the XAB1 or PAB and one for XAB2. The ALU can directly address 65536 locations on the XAB1 and 65536 locations on the XAB2. See the DSP56100 Family User's Manual for additional information.

1.2.5 Program Control Unit (PCU)

The PCU performs instruction fetch, instruction decoding, hardware REP, DO loop control, and exception processing. Two interrupt priority registers (IPR and IPR2) are used to program the priority level of the interrupts. They have control bits for the three external interrupt pins and each of the on-chip peripherals (see Figure 1-7, Table 1-4, Table 1-5 and Table 1-6).

There are 63 interrupts available and one reserved on the DSP56166. Table 1-2 shows each of these interrupts with their respective starting address and Interrupt Priority Level (IPL). The four level three interrupts are not maskable and if two or more are simultaneously issued, their priority is (1) Hardware Reset, (2) Illegal Instruction, (3) Stack Error, and (4) the SWI instruction. The reserved interrupt is not available for use.

The PCU contains five directly addressable registers in addition to the program counter (PC). These are the loop address (LA), loop counter (LC), status register (SR), operating mode register (OMR), and stack pointer (SP). The PC also contains a 15 level, 32-bit wide system stack memory. The 16-bit PC can address 65,536 locations in program memory space. See the DSP56100 Family User's Manual for additional information.

1.2.5.1 Interrupt Priority Structure

Four levels of interrupt priority are provided. Interrupt priority levels (IPLs) numbered 0, 1, and 2, are maskable with level 0 as the lowest level. Level 3 (the highest level), is nonmaskable. The only level 3 interrupts are Reset, Illegal Instruction, Stack Error and SWI. The interrupt mask bits (I1, I0) in the status register reflect the current processor priority level and indicate the interrupt priority level needed for an interrupt source to interrupt the processor (see Table 1-3). Interrupts are inhibited for all priority levels less than the current processor priority level. However, level 3 interrupts are not maskable and therefore can always interrupt the processor.

Table 1-2 Interrupt Sources

Interrupt Starting Address	IPL	Interrupt Source
$0000	3	Hardware RESET
$0002	3	Illegal Instruction
$0004	3	Stack Error
$0006	3	Reserved
$0008	3	SWI
$000A	0-2	IRQA
$000C	0-2	IRQB
$000E	0-2	IRQC
$0010	0-2	RSSI0 Receive Data with Exception Status
$0012	0-2	RSSI0 Receive Data
$0014	0-2	RSSI0Transmit Data with Exception Status
$0016	0-2	RSSI0 Transmit Data
$0018	0-2	RSSI1 Receive Data with Exception Status
$001A	0-2	RSSI1 Receive Data
$001C	0-2	RSSI1 Transmit Data with Exception Status
$001E	0-2	RSSI1 Transmit Data
$0020	0-2	Timer Overflow
$0022	0-2	Timer Compare
$0024	0-2	Host DMA Receive Data
$0026	0-2	Host DMA Transmit Data
$0028	0-2	Host Receive Data
$002A	0-2	Host Transmit Data
$002C	0-2	Host Command (default)
$002E	0-2	Codec Receive/Transmit
$0030	0-2	Available for Host Command
$0032	0-2	Available for Host Command
$0034	0-2	Available for Host Command
.	.	.
.	.	.
.	.	.
$007E	0-2	Available for Host Command

Table 1-3 Status Register (SR) Interrupt Mask Bits

I1	I0	Exceptions Permitted	Exceptions Masked
0	0	IPL 0,1,2,3	None
0	1	IPL 1,2,3	IPL0
1	0	IPL 2,3	IPL 0,1
1	1	IPL 3	IPL 0,1,2

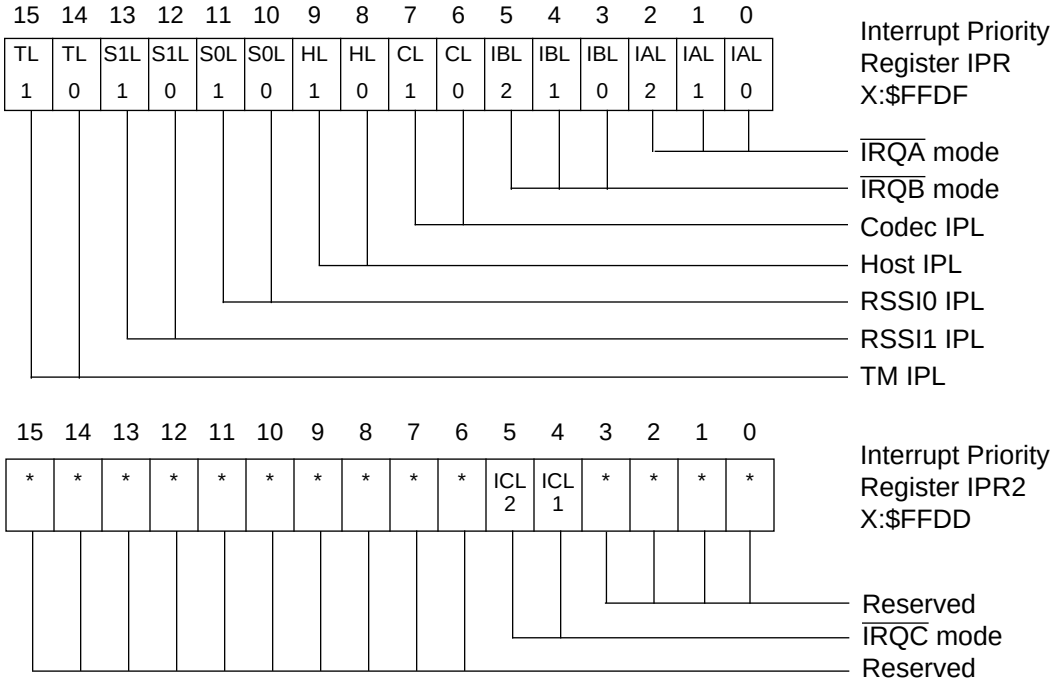


Figure 1-7 Interrupt Priority Register IPR and IPR2

Table 1-4 Interrupt Priority Level Bits

xxL1	xxL0	Enabled	IPL
0	0	No	-
0	1	Yes	0
1	0	Yes	1
1	1	Yes	2

Table 1-5 External Interrupt Trigger Mode Bits

IxL2	Trigger Mode
0	Level
1	Negative Edge

Table 1-6 Interrupt Priority Level Bits for IRQC

ICL2	ICL1	Enabled	IPL
0	0	Not enabled	-
0	1	Level	1
1	0	Not Enabled	-
1	1	Neg. Edge	1

1.2.5.2 Interrupt Priority Levels (IPL)

The interrupt priority level for each on-chip peripheral device and for each external interrupt source ($\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$) can be programmed under software control. Each on-chip or external peripheral device can be programmed to one of the three maskable priority levels (IPL 0, 1, or 2). Interrupt priority levels are set by writing to the Interrupt Priority Registers IPR and IPR2 shown in Figure 1-7. These read/write registers specify the interrupt priority level for each of the interrupting devices (Codec, Host, RSSIs, Timer, $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$, $\overline{\text{IRQC}}$). In addition, this register specifies the trigger mode of both external interrupt sources $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$ and it is used to enable or disable the individual external interrupts. This register is cleared on $\overline{\text{RESET}}$. Table 1-4 defines the interrupt priority level bits. Table 1-5 defines the external interrupt trigger mode bits for $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$. Table 1-6 defines the interrupt priority level for the third external interrupt $\overline{\text{IRQC}}$.

1.2.5.3 Exception Priorities within an IPL

If more than one exception is pending when an instruction is executed, the interrupt with the highest priority level is serviced first. When multiple interrupt requests with the same IPL are pending, a second fixed priority structure within that IPL determines which interrupt is serviced. The fixed priority of interrupts within an IPL and the interrupt enable bits for all interrupts are shown in Table 1-7. The interrupt enable bits for the Host, RSSIs, and TM are located in the control registers associated with their respective on-chip peripherals.

1.3 MEMORY ORGANIZATION

Two independent memory space configurations (X data and P program), are described in section 3. These memory spaces are configured by control bits in the Operating Mode Register, OMR. MA and MB control the program memory map and EX controls the data memory map.

Table 1-7 Exception Priorities within an IPL

Priority	Exception	Enabled by	IP Reg. Bit No.	Control Register Address
Level 3 (Non-maskable)				
Highest	Hardware $\overline{\text{RESET}}$	—	—	—
	Illegal Instruction Interrupt	—	—	—
	Stack Error	—	—	—
Lowest	SWI	—	—	—
Level 0, 1, 2 (Maskable)				
Highest	$\overline{\text{IRQA}}$ (External Interrupt)	$\overline{\text{IRQA}}$ mode bits	0, 1	X:$FFDF
	$\overline{\text{IRQB}}$ (External Interrupt)	$\overline{\text{IRQB}}$ mode bits	3, 4	X:$FFDF
	$\overline{\text{IRQC}}$ (External Interrupt)	$\overline{\text{IRQC}}$ mode bits	IPR2 4,5	X:$FFDD
	Host Command Interrupt	HCIE	2	X:$FFC4
	Host/DMA RX Data Interrupt	HRIE	0	X:$FFC4
	Host/DMA TX Data Interrupt	HTIE	1	X:$FFC4
	RSSI0 RX Data with Exception Status	RIE	15	X:$FFD1
	RSSI0 RX Data	RIE	15	X:$FFD1
	RSSI0 TX Data with Exception Status	TIE	14	X:$FFD1
	RSSI0 TX Data	TIE	14	X:$FFD1
	RSSI1 RX Data with Exception Status	RIE	15	X:$FFD9
	RSSI1 RX Data	RIE	15	X:$FFD9
	RSSI1 TX Data with Exception Status	TIE	14	X:$FFD9
	RSSI1 TX Data	TIE	14	X:$FFD9
	Timer Overflow Interrupt	OIE	9	X:$FFEC
Lowest	Timer Compare Interrupt	CIE	10	X:$FFEC

1.4 EXTERNAL BUS, I/Os and ON-CHIP PERIPHERALS

Five on-chip peripherals are provided on the DSP56166: an 8-bit parallel Host MPU/DMA Interface, a 16-bit timer, two Reduced Synchronous Serial Interfaces (RSSI1 and RSSI0), and a sigma-delta codec (see Figure 1-2). The DSP56166 provides 16 pins for an external

address bus and 16 pins for an external data bus. These pins are grouped to form the Port A bus interface. The DSP56166 also provides 25 programmable I/O pins. These pins may be used as general purpose I/O pins or allocated to an on-chip peripheral. They are separate from the DSP56166 codec, address buses, and data buses and are grouped as two I/O ports (B and C).

Port B is a 15-pin I/O interface that may be used as a general purpose I/O or as a Host MPU/DMA Interface. The Host MPU/DMA Interface provides a dedicated 8-bit parallel port to a host microprocessor or DMA controller and can provide debugging facilities via host exceptions.

Port C is a 10-pin I/O interface that may be used as a general purpose I/O or as a Timer and two identical Reduced Synchronous Serial Interfaces.

The sigma-delta codec has seven dedicated pins which are not available as general purpose I/O. Both analog to digital (A/D) and digital to analog (D/A) converters are provided on-chip. The final decimation, antialiasing and compensation filters for the A/D and interpolation, reconstruction, and compensation filters for the D/A converter are implemented in software on the DSP providing the user considerable flexibility in the codec filter characteristics.

1.4.1 Memory Expansion Port (Port A)

The DSP56166 expansion port is designed to synchronously interface over a common 16-bit data bus with a wide variety of memory and peripheral devices such as high speed static RAMs, slower memory devices, DSPs, and MPUs in master/slave configurations. This capability is possible because the external bus cycle time is programmable. The expansion bus timing is controlled by a two Bus Control Registers (BCR & BCR2). The bus control registers control the timing of the bus interface signals and data lines. Each of the two memory spaces, X data and Program data, has its own 5 control bits in the BCR which can be programmed for up to 31 wait states (one wait state is equal to a clock period, or equivalently, one-half of an instruction cycle). The access time on the external peripheral memory space can be controlled with 5 control bits in BCR2. In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces or external peripherals.

1.4.2 General Purpose I/O (Port B, Port C)

Each Port B and C pin may be programmed as a general purpose I/O pin or as a dedicated on-chip peripheral pin under software control. A 10-bit Port Control Register, PCC, is associated with Port C and allows each port pin to be programmed individually for one of these two functions. The port control register associated with Port B, PBC, contains only

one bit which programs all 15 pins. Also associated with each general purpose port is a data direction register which programs each pin as an input or output, and a data register for data I/O.

1.4.3 RSSI0 and RSSI1

The DSP56166 provides two identical Reduced Synchronous Serial Interfaces (RSSI's). They are extremely flexible, full-duplex serial interfaces which allow the DSP56166 to communicate with a variety of serial devices. These interfaces include one or more industry standard codecs, other DSPs, microprocessors, and peripherals. The RSSIx interface consists of a transmitter and a receiver section and a transmit/receive RSSI clock generator. Each of the following characteristics of the RSSI can be independently defined: the number of bits per word, the protocol or mode, and the clock.The transmit and receive sections are synchronous. Three modes of operation are available: Normal, Network, and Gated. The Normal Mode is typically used to interface with devices on a regular or periodic basis. In this mode, the RSSI functions with one data word of I/O per frame. The Network Mode provides time slots in addition to a bit clock and a frame synchronization pulse. The RSSI functions with 2 to 8 words of I/O per frame in the Network Mode. This mode is typically used in star or ring Time Division Multiplex (TDM) networks with other DSP56166s and/or codecs. The clock can be programmed to be continuous or gated. The RSSI supports a subset of the Motorola SPI interface. The RSSI requires three to four pins depending on the operating mode selected.

1.4.4 Timer

The Timer is a general purpose 16-bit timer/event counter with internal or external clocking which can be used to interrupt the DSP or to signal an external device at periodic intervals, after counting internal events or after counting external events. A Timer Input pin (TIN) can be used as an event counter input and a Timer Output pin (TOUT) can be used for timer pulse or timer clock generation.

The timer includes three 16-bit registers: the Timer Count Register (TCTR), the Timer Preload Register (TPR), and the Timer Compare Register (TCPR). An additional Timer Control Register (TCR) controls the timer operations.

A decrement register, programmed by the control register, is not available to the user. All other registers are memory mapped read/write registers.

1.4.5 Host Interface (HI)

The HI is a byte-wide parallel slave port which may be connected directly to the data bus of a host processor. The host processor may be any of a number of popular microcomputers or microprocessors, another DSP, or DMA hardware. The HI is composed of an 8-bit bidirectional data bus and 7 control lines to control data transfers. The HI appears as

a memory mapped peripheral, occupying 8 bytes in the host processor's address space and three words in the DSP processor's address space. Separate transmit and receive data registers are double-buffered to allow the DSP56166 and host processor to efficiently transfer data at high speed. Host processor communication with the HI registers is accomplished using standard host processor instructions and addressing modes. Host processors may use byte move instructions to communicate with the HI registers. The host registers are addressed so that 8-bit MC6801-type host processors can use 16-bit load (LDD) and store (STD) instructions for data transfers. The 16-bit MC68000/10 host processor can address the HI using the special MOVEP instruction for word (16-bit) or long word (32-bit) transfers. The 32-bit MC68020 host processor can use its dynamic bus sizing feature to address the HI using standard MOVE word (16-bit), long word (32-bit) or quad word (64-bit) instructions.

One of the most innovative features of the HI is the Host Command feature. With this feature, the host processor can issue vectored exception requests to the DSP56166. The host may select any one of 63 DSP56166 exception routines to be executed by writing a Vector Address Register in the HI. This flexibility allows the host programmer to execute up to 63 functions preprogrammed in the DSP56166.

1.5 OnCE

OnCE provides hardware/software emulation and debug on the DSP56166 and a means of interacting with the DSP56166 and any memory mapped peripherals non-intrusively so that a user may examine registers, memory, or peripherals. To achieve this, special circuits and dedicated pins on the DSP are used to avoid sacrificing any user accessible on-chip resource. A key feature of the special OnCE pins is to allow the user to insert the DSP56166 into his target system yet retain debug control, especially in the case of devices specified without an external bus. The need for a costly cable which brings out all processor pins on a traditional emulator system is eliminated.

1.6 PROGRAMMING MODEL

The DSP56166 is based on the DSP5616 core. The programmer can view the DSP5616 core architecture as three execution units operating in parallel. The three execution units are the Data ALU, Address Generation Unit, and Program Control Unit. The programming model appears like that of a conventional MPU. The programming model is shown in Figure 1-8 and is described in the following paragraphs.

1.6.1 Data ALU

The data ALU features four 16-bit input/output data registers which can be concatenated to handle 32-bit data, two 40-bit accumulators, automatic scaling, and saturation arithmetic.

1.6.1.1 Data ALU Input Registers (X1, X0, Y1, Y0)

X1, X0, Y1, and Y0 are 16-bit latches which serve as input pipeline registers for the data ALU. Each register may be read or written by the XDB. X0, X1, and Y0 may be written over the GDB. They may be treated as four independent 16-bit registers or as two 32-bit registers called X and Y which are developed by the concatenation of X1:X0 and Y1:Y0 respectively. X1 is the most significant word in X and Y1 is the most significant word in Y.

These Data ALU input registers are used as source operands for most data ALU operations and allow new operands to be loaded for the next instruction while the register contents are being used by the current instruction.

1.6.1.2 Data ALU Accumulator Registers (A2, A1, A0, B2, B1, B0)

A1, A0, B1, and B0 are 16-bit latches that serve as data ALU accumulator registers. A2 and B2 are 8-bit latches that serve as accumulator extension registers. Each register may be read or written by the XDB as a word operand. A1 and B1 may be written by the GDB. When A2 or B2 is read, the register contents occupy the low-order portion (bits 7-0) of the word; the high-order portion (bits 16-8) is sign-extended. When A2 or B2 is written, the register receives the low-order portion of the word; the high-order portion is not used. Automatic sign extension of the 40-bit accumulators is provided when the A or B register is written with a smaller size operand. If the A or B register is written with a 16-bit value, then the least significant 16 bits are set to zero.

It is also possible to saturate the accumulator on a 32-bit value automatically after every accumulation. Overflow protection is performed after the contents of the accumulator have been shifted according to the scaling mode defined in the status register. When limiting occurs, the L bit flag in the status register is set and latched.

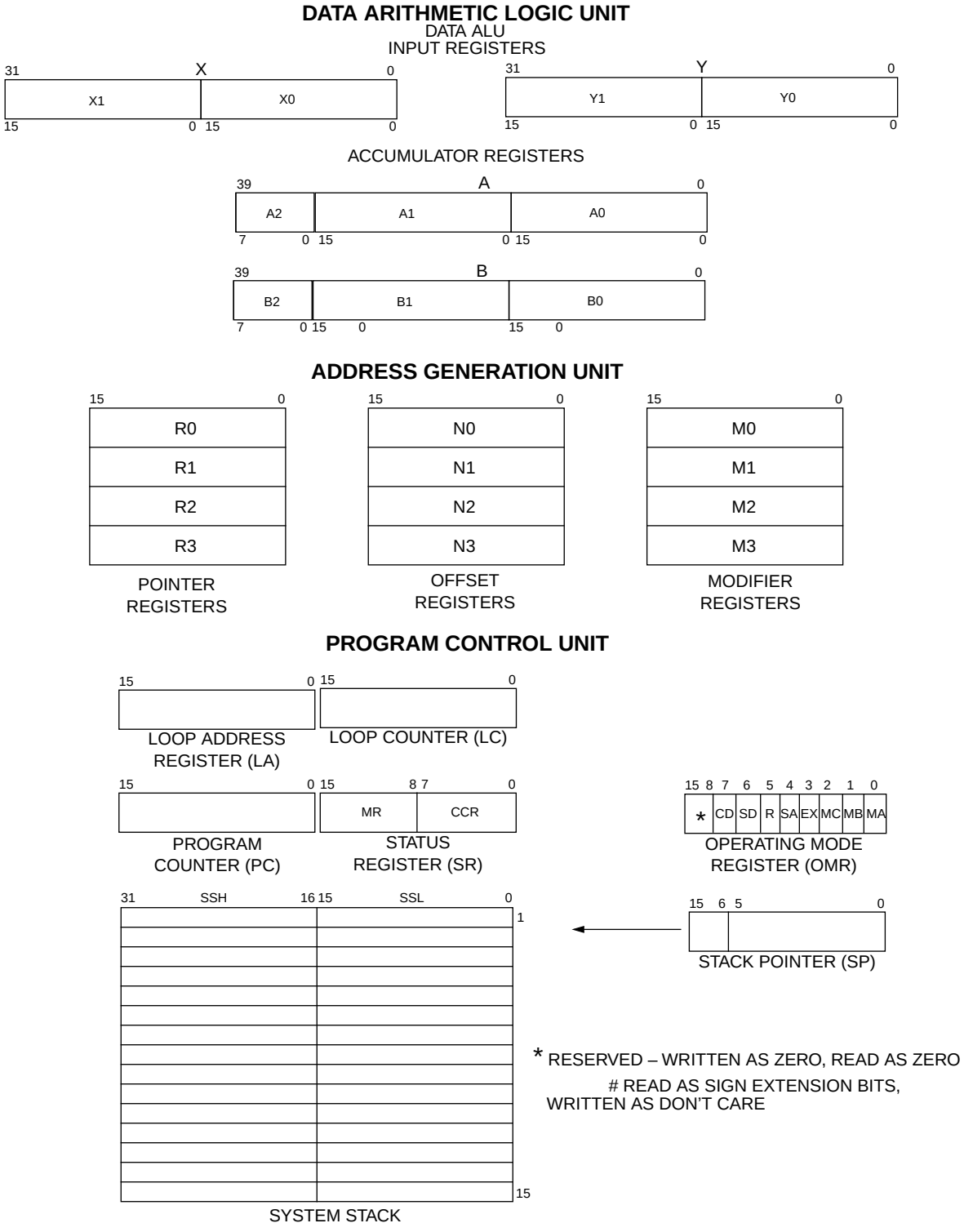


Figure 1-8 Programing Model

1.6.2 Address Generation Unit

The programmer's model for the address generation unit consists of three banks of register files — pointer register files, offset register files, and modifier register files. These provide all the registers necessary to generate address register indirect effective addresses.

1.6.2.1 Address Register File (R0-R3)

The address register file consists of four, sixteen-bit registers. The file contains the address registers R0-R3 which usually contain addresses used as pointers to memory. Each register may be read or written by the Global Data Bus. Each address register may be used as an input to the modulo arithmetic unit for a register update calculation. Each register may be written by the GDB or by the output of the modulo arithmetic unit.

1.6.2.2 Offset Register File (N0-N3)

The offset register file consists of four, sixteen-bit registers. The file contains the offset registers N0-N3 and usually contains offset values used to update address pointers. Each offset register may be read or written by the Global Data Bus. Each offset register is used as an input to the modulo arithmetic unit when the same number address register is read and used as an input to the modulo arithmetic unit.

1.6.2.3 Modifier Register File (M0-M3)

The modifier register file consists of four, 16-bit registers. The file contains the modifier registers M0-M3 and usually specifies the type of arithmetic used to modify an address register during address register update calculations — linear, modulo, or reverse carry. Each modifier register may be used as an input to the modulo arithmetic unit or written by the Global Data Bus. Each modifier register is read when the same number address register is read and used as an input to the modulo arithmetic unit. Each modifier register is preset to $FFFF during a processor reset. Note that when R3 is used for the second read of a dual read operation, only linear arithmetic is available on this address register.

1.6.3 Program Control Unit

The program control unit features loop address and loop counter registers which are dedicated to supporting the hardware DO loop instruction in addition to the standard program flow control resources such as a program counter, status register, and system stack. With the exception of the program counter, all registers are read/write to facilitate system debug.

1.6.3.1 Program Counter (PC)

This 16-bit register contains the address of the next location to be fetched from program memory space. The PC may point to instructions, data operands, or addresses of oper-

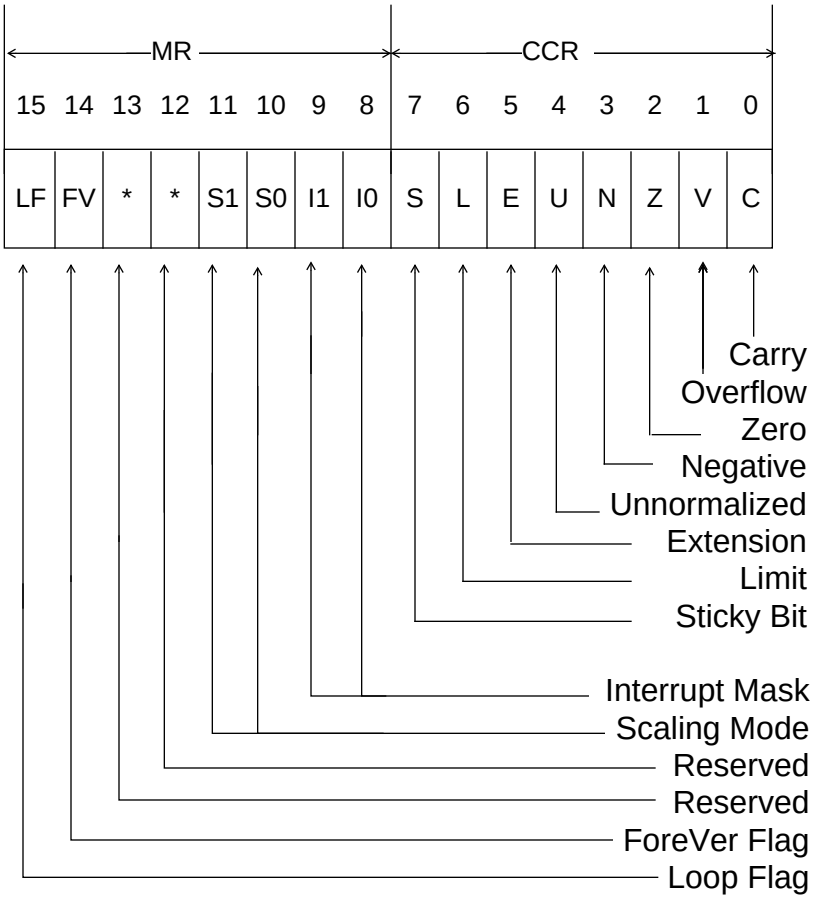


Figure 1-9 Status Register Format

ands. References to this register are always inherent and are implied by most instructions. This special purpose address register is stacked when program looping is initiated, when a branch or a jump to subroutine is performed, and when interrupts occur (except for fast interrupts).

1.6.3.2 Status Register (SR)

The SR is a 16-bit register consisting of an 8-bit Mode Register (MR) and an 8-bit Condition Code Register (CCR) — see Figure 1-9. The MR register is the high-order 8 bits of the SR; the CCR register is the low-order 8 bits.

The MR bits are only affected by processor reset, exception processing, the DO, ENDDO, RTI, and SWI instructions and by instructions which directly reference the MR register (e.g., MOVE, ANDI, ORI). During processor reset, the interrupt mask bits of the mode register will be set, the scaling mode bits, loop flag, and the forever flag will be cleared. The

CCR is a special purpose control register which defines the current user state of the processor at any given time. The CCR bits are affected by data ALU operations, one address ALU operation (CHKAAU), bit field manipulation instructions, parallel move operations, and by instructions which directly reference the CCR register. The CCR bits are not affected by data transfers over XDB except when data limiting occurs while reading the A or B accumulators or by the conditions described for the Sticky Bit, Bit 7. During processor reset, all CCR bits are cleared. The SR register is stacked when program looping is initialized, when a jump or branch to subroutine (JScc, BScc, JSR, BSR) is performed, and when long interrupts occur.

1.6.3.3 Loop Counter (LC)

The LC is a special 16-bit counter used to specify the number of times to repeat a hardware program loop. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of a BRKcc or an ENDDO instruction. When the end of a hardware program loop is reached, the contents of the loop counter 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 stack. If the LC is not one, it is decremented by one and the program loop is repeated. The LC may be read under program control. This allows the number of times a loop has been executed to be determined during execution. Note that if LC=0 during execution of the DO instruction, the loop will not be executed and the program will continue with the instruction immediately after the loop end of expression. LC is also used by the REP instruction.

1.6.3.4 Loop Address Register (LA)

The LA indicates the location of the last instruction word in a program DO loop. This register is stacked by a DO instruction and unstacked by end of loop processing or by execution of an ENDDO or BRKcc instruction. When the instruction word at the address contained in this register is fetched, the content of LC is checked. If it is not one, the LC is decremented and the next instruction is taken from the address at the top of the system stack. Otherwise; the PC is incremented, the loop flag is restored (pulled from stack), the stack pointer is decremented by two, the LA and LC registers are pulled from the stack and restored, and instruction execution continues normally. The LA register is a read/write register written into by a DO instruction.

1.6.3.5 System Stack (SS)

The SS is a separate internal RAM, 15 locations “deep”, and divided into two 16-bit wide banks: High (SSH) and Low (SSL). SSH stores the PC and LA contents; SSL stores the LC and SR contents. The PC and SR registers are pushed on the stack for subroutine calls and long interrupts. These registers are pulled from the stack for subroutine returns

using the RTS instruction (PC only) and for interrupt returns that use the RTI instruction. The system stack is also used for storing the address of the beginning instruction of a hardware program loop as well as the SR, LA, and LC register contents just prior to the start of the loop. This allows nesting of DO loops.

Up to 15 long interrupts, 7 DO loops, or 15 JSRs or combinations of these can be accommodated by the stack. Care must be taken when approaching the stack limit. When the Stack limit is exceeded, the data to be stacked will be lost and a non-maskable Stack Error interrupt will occur. The stack error interrupt occurs after the stack limits have been exceeded.

1.6.3.6 Stack Pointer (SP)

The SP is a 6-bit register that indicates the location of the top of the SS and the stack status (underflow, empty, full, and overflow conditions – see Table 1-8). The SP is referenced implicitly by some instructions (DO, REP, JSR, RTI, etc.) or directly by the MOVEC instruction. Note that the stack pointer register is implemented as a 6-bit counter which addresses (selects) a fifteen location stack with its four least significant bits.

Table 1-8 Stack Pointer Values

UF	SE	P3	P2	P1	P0	CAUSE
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 (stack full). Push causes overflow.
0	1	0	0	0	0	← Stack Overflow condition.
0	1	0	0	0	1	← Stack Overflow condition after double push.

1.6.3.7 Operating Mode Register (OMR)

The OMR is a 16-bit register which defines the current processor operating mode. The OMR bits are only affected by processor reset and by instructions which directly reference the OMR.

The operating mode register format is shown in Figure 1-10.

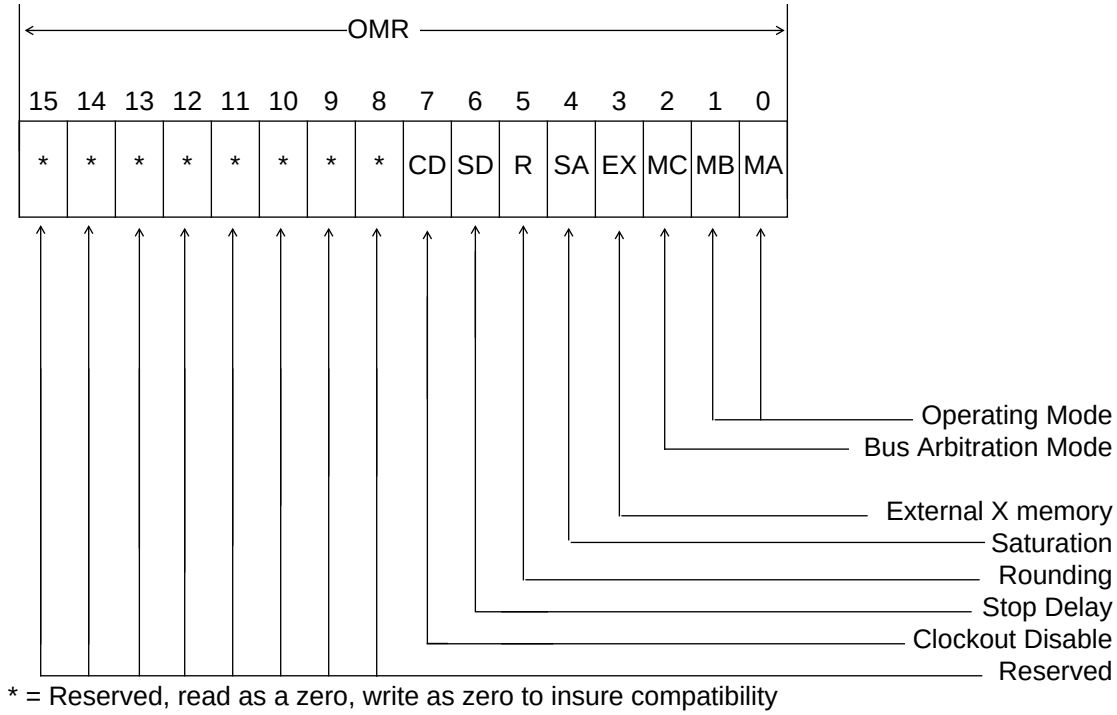


Figure 1-10 Operating Mode Register (OMR)

During processor reset, the chip operating mode bits will be loaded from the external Mode Select pins. Table 1-9 summarizes the operating modes for the 56166 RAM based part and Table 1-10 does the same for the ROM based part.

Table 1-9 Operating Mode Summary — DSP56166 RAM Based Part

Operating Mode	MB	MA	Description
Special Bootstrap 1	0	0	Bootstrap from an external byte-wide memory located at P:$C000.
Special Bootstrap 2	0	1	Bootstrap from the host port or RSSI0
Normal Expanded	1	0	Internal PRAM enabled; external reset at P:$E000
Development Expanded	1	1	Internal program RAM disabled; external reset at P:$0000.

Table 1-10 Operating Mode Summary — DSP56166 ROM Based Part

MB	MA	Chip Operating Mode	Reset Vector	Program Memory Configuration
0	0	Single Chip	Internal Pmem P:$0	Internal Pmem Enabled
0	1	Single Chip	Internal Pmem P:$100	Internal Pmem Enabled
1	0	Normal Expanded	External Pmem P:$E000	Internal Pmem Enabled
1	1	Development	External Pmem P:$0	Internal Pmem Disabled

*: the address will be the first location of the internal PROM.

The Mode C bit (MC) selects the initial bus operating mode.When set, the external bus is programmed in the master mode ($\overline{BR}$ output and $\overline{BG}$ input) and when cleared, the bus is programmed in the slave mode ($\overline{BR}$ input and $\overline{BG}$ output).

The EX bit, when set, will force all accesses to the X data memory to be external accesses, preventing access to on-chip data memory and on-chip peripheral space.This allows a continuous memory map when using 64K of external data memory. When the EX bit is set, the memory mapped I/O registers, both on and off-chip, can only be accessed during fast interrupt. During normal operation and during a long interrupt, the EX bit must be explicitly cleared by the user's code before the on-chip data memory and the memory mapped I/O registers can be read or written. The on-chip peripherals, however, will still be active and interrupts will still be recognized. Special care should be taken with this bit in a nested interrupt environment since it is not in the Status Register (SR) and thus is not stacked with each new interrupt. Note that OnCE cannot read the content of on-chip X memories, registers and peripherals when the EX bit is set, but it can still write to them.

The Saturation bit (SA), when set, selects automatic saturation on 32 bits for the results from the MAC array back to the accumulator. This saturation is done by a special saturation circuit inside the MAC unit. The purpose of this bit is to provide a saturation mode for 16-bit algorithms which do not recognize or cannot take advantage of the extension accumulator. The saturation logic operates by checking three bits of the 40-bit result: two bits of the extension byte (exp[7] and exp[0]) and one bit on the MSP (msp[15]). The result obtained in the accumulator when SA =1 is shown in Table 1-11.

The Rounding bit (R)selects between convergent rounding and two's complement rounding. When set, two's complement rounding (always round up) is used.

The Stop Delay bit (SD) is used to select the delay that the DSP needs to exit the STOP mode.

Table 1-11 Actions of the Saturation Mode (SA=1)

exp[7]	exp[0]	msp[15]	result in accumulator
0	0	0	unchanged
0	0	1	$00 7FFF FFFF
0	1	0	$00 7FFF FFFF
0	1	1	$00 7FFF FFFF
1	0	0	$FF 8000 0000
1	0	1	$FF 8000 0000
1	1	0	$FF 8000 0000
1	1	1	unchanged

When the Clock out Disable bit (CD) is cleared in the OMR, the CLKO pin provides a clock to external circuitry. Setting the CD bit will disable this signal one instruction cycle after the bit has been set.

Note: When a bit of the OMR is changed by an instruction, a delay of one instruction cycle is necessary before the new mode comes into effect.

1.7 INSTRUCTION SET SUMMARY

As indicated by the programming model, the DSP architecture can be viewed as three functional units operating in parallel (Data ALU, AGU, and PCU). The goal of the instruction set is to keep each of these units busy each instruction cycle. This achieves maximum throughput and minimum use of program memory.

This section introduces the DSP instruction set and instruction format. The complete range of instruction capabilities combined with the flexible addressing modes provide a very powerful assembly language for digital signal processing algorithms. The instruction set has also been designed to allow efficient coding for future high-level DSP language compilers. Execution time is enhanced by the hardware looping capabilities.

1.7.1 Instruction Groups

The instruction set is divided into the following groups:

- Arithmetic
- Logical
- Bit Field Manipulation
- Loop
- Move
- Program Control

Each instruction group is described in the following sections. Detailed information on each instruction is given in Appendix A of the DSP56100 Family User's Manual.

1.7.1.1 Arithmetic Instructions

The arithmetic instructions perform all of the arithmetic operations within the Data ALU. They may affect all of the condition code register bits. Arithmetic instructions are register-based (register direct addressing modes used for operands) so that the Data ALU operation indicated by the instruction does not use the XDB or the GDB. Optional data transfers may be specified with most arithmetic instructions. This allows for parallel data movement over the XDB and over the GDB during a Data ALU operation. This allows new data to be prefetched for use in subsequent instructions and allows results calculated by previous instructions to be stored. These instructions execute in one instruction cycle. The following are the arithmetic instructions.

ABS	Absolute Value
ADC	Add Long with Carry*
ADD	Add †
ASL	Arithmetic Shift Left
ASL4	4-Bit Arithmetic Shift Left*
ASR	Arithmetic Shift Right
ASR4	4-Bit Arithmetic Shift Right*
ASR16	16-Bit Arithmetic Shift Right*
CHKAAU	Update the V, Z, N flags according to the address calculation result*
CLR	Clear an Accumulator
CLR24	Clear 24 MSBs of an Accumulator
CMP	Compare
CMPM	Compare Magnitude
DEC	Decrement Accumulator
DEC24	Decrement upper 24 bits of Accumulator
DIV	Divide Iteration*
DMAC	Double (Multi) precision oriented MAC*
EXT	Sign Extend Accumulator from bit 31*
IMAC	Integer Multiply-Accumulate*
IMPY	Integer Multiply*
INC	Increment Accumulator
INC24	Increment 24 MSBs of Accumulator
MAC	Signed Multiply-Accumulate †
MACR	Signed Multiply-Accumulate and Round †
MPY	Signed Multiply †
MPYR	Signed Multiply and Round †
MPY(su,uu)	Mixed Mode Multiply*

MAC(su,uu)	Mixed Mode Multiply-Accumulate*
NEG	Negate
NEGC	Negate with Borrow*
NORM	Normalize*
RND	Round
SBC	Subtract Long with Carry
SUB	Subtract †
SUBL	Shift Left and Subtract
SWAP	Swap MSP and LSP of an Accumulator*
Tcc	Transfer Conditionally*
TFR	Transfer Data ALU Register (Accumulator as destination) †
TFR2	Transfer Accumulator (32-bit Data Alu register as destination)*
TST	Test an accumulator
TST2	Test an ALU data register*
ZERO	Zero Extend Accumulator from bit 31*

*These instructions do not allow parallel data moves.
† These instructions allow a dual read parallel move.

1.7.1.2 Logical Instructions

The logical instructions perform all of the logical operations within the Data ALU. They may affect all of the condition code register bits. Logical instructions are register-based as are the arithmetic instructions above. Optional data transfers may be specified with most logical instructions. With the exceptions of ANDI and ORI instructions, the destination of all logical instructions is A1 or B1. These instructions execute in one instruction cycle. The following are the logical instructions.

AND	Logical AND
ANDI	AND Immediate Program Controller Register*
EOR	Logical Exclusive OR
LSL	Logical Shift Left
LSR	Logical Shift Right
NOT	Logical Complement
OR	Logical Inclusive OR
ORI	OR Immediate Program Controller Register*
ROL	Rotate Left
ROR	Rotate Right

*These instructions do not allow parallel data moves.

1.7.1.3 Bit Field Manipulation Instructions

This group tests the state of any set of bits within a byte in a memory location or a register and then sets, clears, or inverts bits in this byte. Bit fields which can be tested include the upper byte and the lower byte in a 16-bit value. The carry bit of the condition code register will contain the result of the bit test for each instruction. These instructions are read-modify-write and require two instruction cycles. Parallel data moves are not allowed with any of these instructions. The following are the bit field manipulation instructions.

BFTSTL	Bit Field Test Low
BFTSTH	Bit Field Test High
BFCLR	Bit Field Test and Clear
BFSET	Bit Field Test and Set
BFCHG	Bit Field Test and Change

1.7.1.4 Loop Instructions

The loop instructions control hardware looping by initiating a program loop and setting up looping parameters, or by “cleaning” up the system stack when terminating a loop. Initialization includes saving registers used by a program loop (LA and LC) on the system stack so that program loops can be nested. The address of the first instruction in a program loop is also saved to allow no-overhead looping. The end address of the DO loop is specified as PC relative. Parallel data moves are not allowed with any of these instructions. The following are the loop instructions.

DO	Start Hardware Loop
DO FOREVER	Hardware Loop Forever
ENDDO	Disable Current Loop and Unstack Parameters
BRKcc	Conditional Exit from Hardware Loop

1.7.1.5 Move Instructions

The move instructions perform data movement over the XDB and over the GDB. Move instructions do not affect the condition code register except for the limit bit, L, if limiting is performed or the Sticky Bit, S, when reading a Data ALU accumulator register. AGU instructions are also included among the following move instructions. These instructions do not allow optional data transfers.

LEA	Load Effective Address
MOVE	Move Data with or without Register Transfer – TFR(3)
MOVE(C)	Move Control Register
MOVE(I)	Move Immediate Short
MOVE(M)	Move Program Memory

MOVE(P)	Move Peripheral Data
MOVE(S)	Move Absolute Short

1.7.1.6 Program Control Instructions

The program control instructions include branches, jumps, conditional branches, jumps and other instructions which affect the PC and system stack. Program control instructions may affect the condition code register bits as specified in the instruction. Parallel data moves are not allowed with any of these instructions. The following are the program control instructions.

Bcc	Branch Conditionally (PC relative)
BSR	Branch to Subroutine (PC relative)
BRA	Branch (PC relative)
BScc	Branch to Subroutine Conditionally (PC relative)
DEBUG	Enter Debug Mode
DEBUGcc	Enter Debug Mode Conditionally
Jcc	Jump Conditionally
JMP	Jump
JSR	Jump to Subroutine
JScc	Jump to Subroutine Conditionally
NOP	No Operation
REP	Repeat Next Instruction
REPcc	Repeat Next Instruction Conditionally
RESET	Reset Peripheral Devices
RTI	Return from Interrupt
RTS	Return from Subroutine
STOP	Stop Processing (low power stand-by)
SWI	Software Interrupt
WAIT	Wait for Interrupt (low power stand-by)

1.7.2 Instruction Formats

Instructions are one or two words in length. The instruction and its length are specified by the first word of the instruction. The next word may contain information about the instruction itself or about an operand for the instruction. The assembly language source code for a typical one word instruction is shown below. The source code is organized into four columns.

Opcode	Operands	X Bus Data	G Bus Data
MAC	X0,Y0,A	X:(R0)+,X0	X:(R3)+,Y0

The Opcode column indicates the Data ALU, AGU, or PCU operation to be performed. The Operands column specifies the operands to be used by the opcode. The X Bus Data and G Bus Data columns specify optional data transfers over the X Bus, the G bus, and the addressing modes to be used. The Opcode column must always be included in the source code.

The DSP offers parallel processing using the Data ALU, AGU, and PCU. For the instruction word above, the DSP will perform the designated ALU operation (Data ALU), up to two data transfers specified with address register updates (AGU), and will also decode the next instruction and fetch an instruction from program memory (PCU), all in one instruction cycle. When an instruction is more than one word in length, an additional instruction execution cycle is required. Most instructions involving the Data ALU are register-based (all operands are in Data ALU registers) and allow the programmer to keep each parallel processing unit busy. An instruction which is memory-oriented (such as a bit field manipulation instruction) or that causes a control flow change (such as a branch/jump) prevents the use of parallel processing resources during its execution. See the DSP56100 Family User's Manual for additional information.

1.7.3 Addressing Modes

The addressing modes are grouped into three categories — register direct, address register indirect, and special. These addressing modes are summarized in Table 1-12. All address calculations are performed in the address generation unit to minimize execution time. Addressing modes specify whether the operand(s) is(are) in a register, memory, or encoded in the instruction as immediate data.

The register direct addressing mode can be subclassified according to the specific register addressed. The data registers include X1, X0, Y1, Y0, X, Y, A2, A1, A0, B2, B1, B0, A, and B. The control registers include SR, OMR, SP, SSH, SSL, LA, LC, CCR, and MR.

Address register indirect modes use an address register, Rn, to point to locations in memory. The content of Rn is the effective address (ea) except in the indexed by offset mode where the ea is Rn+Nn, or in the indexed by short displacement where the ea is Rn+a short immediate constant. Address register indirect modes use a modifier register, Mn, to specify the type of arithmetic to be used to update Rn. If a mode using an offset is specified, an offset register, Nn, is also used for the update. The Nn and Mn registers are assigned to the Rn with the same n. Thus, the assigned register sets are R0;N0;M0, R1;N1;M1, R2;N2;M2, and R3;N3;M3.

Table 1-12 DSP5616 Addressing Modes

Addressing Mode	Uses Mn Modifier	Operand Reference						
		S	C	D	A	P	X	XX
Register Direct								
Data or Control Register	No	X	X	X				
Address Register Rn	No				X			
Address Modifier Register Mn	No				X			
Address Offset Register Nn	No				X			
Address Register Indirect								
No Update	No					X	X	
Postincrement by 1	Yes*					X	X	X
Postdecrement by 1	Yes					X	X	
Postincrement by Offset Nn	Yes*					X	X	X
Indexed by Offset Nn	Yes						X	
Predecrement by 1	Yes						X	
PC Relative								
Long Displacement	No		X			X		
Short Displacement	No		X			X		
Address Register	No		X		X	X		
Special								
Upper word of accumulator	No						X	
Immediate Data	No					X		
Immediate Short Data	No					X		
Absolute Address	No					X	X	
Absolute Short Address	No					X	X	
Short Jump Address	No					X		
I/O Short Address	No						X	
Implicit	No	X	X			X		
Indexed by short displacement	No						X	

Where:

S = System Stack Reference
P = Program Memory Reference
C =Program Controller Register Reference
X = X Memory Reference
D = Data ALU Register Reference
XX = Double X Memory Read
A = Address ALU Register Reference

***Note:** M3 is not used for updating R3 in the second read in the X memory

This structure is unique and extremely powerful in general, and particularly powerful in setting up DSP oriented data structures. Two sets of address registers can be used by the

instruction set: one set for the first memory operation and one set for a second memory operation. Note that M3 is not used for updating R3 in the second read in the X memory.

The special addressing modes include immediate and absolute modes as well as implied references to the PC, system stack, and program memory. In addition, it is possible to use the upper word (MSP) of an accumulator as an address.

1.7.4 Address Arithmetic

The DSP56166 Address Generation Unit supports linear, modulo, and bit-reversed address arithmetic for all address register indirect modes. The address modifiers, Mn, determine the type of arithmetic used to update addresses. Address modifiers allow the creation of data structures in memory for FIFOs (queues), delay lines, circular buffers, stacks, and bit-reversed FFT buffers. Data is manipulated by updating address registers (pointers) rather than moving large blocks of data. The contents of the address modifier register, Mn, defines the type of address arithmetic to be performed for addressing mode calculations, and for the case of modulo arithmetic, the content of Mn also specifies the modulus. Each address register Rn has its own modifier register Mn associated with it.

1.7.4.1 Linear Modifier

Address modification is performed using normal 16-bit (modulo 65,536) two's complement linear arithmetic. A 16-bit offset Nn, or immediate data (+1, -1, or a displacement value) may be used in the address calculations. The range of values may be considered as signed (Nn from -32,768 to +32,767) or unsigned (Nn from 0 to +65,536).

1.7.4.2 Reverse Carry Modifier

The address modification is performed by propagating the carry in the reverse direction, i.e., from the MSB to the LSB. If the (Rn)+Nn addressing mode is used with this address modifier, and Nn contains the value 2^{K-1} (a power of two), then postincrementing by Nn is equivalent to bit-reversing the K LSBs of Rn, incrementing Rn by 1, and bit-reversing the K LSBs of Rn again. This address modification is useful for 2^K point FFT addressing. The range of values for Nn is 0 to +32,767 which allows bit-reversed addressing for FFTs up to 65,536 points.

As an example, consider a 1024 point FFT with real and imaginary data stored in memory. Then Nn would contain the value 512 and postincrementing by +N would generate the address sequence 0, 512, 256, 768, 128, 640, … This is the scrambled FFT data order for sequential frequency points from 0 to 2π. For proper operation, the reverse carry modifier restricts the base address of the bit reversed data buffer to an integer multiple of 2^K, such as 1024, 2048, 3072, etc. The use of addressing modes other than postincrement by Nn is possible but may not provide a useful result.

1.7.4.3 Modulo Modifier

The modulo arithmetic unit can update one address register, Rn, during one instruction cycle and is capable of performing linear, reverse carry, and modulo arithmetic. The contents of the selected modifier register specifies the type of arithmetic required in an address register update calculation. There is no modulo capability for an R3 update for dual reads.