1. INTRODUCTION

The *WE* DSP16A Digital Signal Processor is a 16-bit, high-performance, CMOS integrated circuit. This device can be programmed to perform a wide variety of signal-processing functions. This is the DSP of choice for applications requiring low power, high performance, and low cost.

This manual is a reference guide for the DSP16A device. It describes the architecture, instruction set, and interfacing requirements of the device.

1.1 FEATURES

- 25 ns, 33 ns, 55 ns instruction cycle
- 16 × 16-bit multiply/add in one instruction cycle
- Two 36-bit accumulators
- Up to 24,576 words of ROM and up to 2,048 words of RAM on-chip
- Complete set of ALU (arithmetic logic unit) operations
- Immediate, indirect, and compound addressing modes
- Instruction cache for high-speed, ROM-efficient vector operations
- Serial and parallel I/O ports with multiprocessor capability
- Low-power CMOS technology, 84-pin plastic package

1.2 DEVICE DESCRIPTION

1.2.1 Architecture

The arithmetic unit contains a 16 × 16-bit parallel multiplier that generates a full 32-bit product in one instruction cycle. The product can be accumulated with one of two 36-bit accumulators. The data in these accumulators can be directly loaded from or stored to memory in 16-bit words with automatic saturation on overflow. The ALU supports a full set of arithmetic and logic operations on either 16- or 32-bit data. A standard set of ALU conditions can be tested for conditional branches and subroutine calls. This procedure allows the processor to function as a powerful 16- or 32-bit microprocessor for logical and control applications.

Two addressing units support high-speed, register-indirect memory addressing with postmodification of the register. Four address registers can be used for either read or write addresses to the RAM without restrictions. One address register is dedicated to the ROM for table look-up. Direct and immediate addressing is supported at a cost of only one additional instruction cycle and one ROM location.

The DSP16A offers up to 24,576 words of ROM and 2,048 words of RAM. The ROM may be replaced with up to 64 Kwords of external memory or augmented with up to an additional 60 Kwords of external memory. An on-chip cache memory can be selectively used to store such repetitive operations as a filter section. The code in the cache can be repeated up to 127 times with no looping overhead. In addition, operations in the cache that require a ROM access (for example, reading fixed coefficients) execute at twice the normal rate. The cache greatly reduces the need for writing in-line repetitive code and, therefore, conserves ROM storage.

The device has both serial and parallel I/O ports. The serial I/O unit is double-buffered and easily interfaces with other DSP16A devices, commercially available codecs, and time-division multiplexed (TDM) channels with few (if any) additional components. The parallel I/O unit is capable of interfacing to a 16-bit bus containing other DSP16A devices, microprocessors, microprocessor peripherals, or other I/O devices.

The processor is implemented in low-power CMOS technology and is packaged in an 84-pin, plastic leaded chip carrier (PLCC) and an 84-pin, plastic quad flat package (PQFP).

1.2.2 Instruction Set

The DSP16A instructions fall into five possible categories: multiply/ALU, special function, control, data move, and cache. All instructions are 16 bits wide and have a C-like assembler syntax. Although some pipelining of DSP16A instructions is necessary to achieve the real-time performance required in many signal processing applications, the degree of pipelining has been reduced from previous generation DSPs to simplify programming. Latency effects have been eliminated.

1.3 APPLICATION DEVELOPMENT

Application development is aided by the use of the *WE* DSP16A Support Software Library, the *WE* DSP16A Application Library, the *WE* DSP16A Digital Signal Processor Development System, and the *WE* DSP16A Evaluation Board.

1.3.1 Support Software Library

Support software tools to help create, test, and debug DSP16A application programs are available from the *WE* DSP16A Support Software Library. The support software library consists of an integrated assembler and simulator that run on the *MS-DOS**, *VMS*†, or *UNIX*‡ Operating Systems.

The DSP16A software simulator provides access to all registers and to memory and allows program breakpointing. The simulator also provides the user interface to the *WE* DSP16A Development System. The hardware development systems have many of the breakpointing capabilities of the software simulator and can also be used in a simulator/accelerator mode to speed software simulations.

---

* *MS-DOS* is a registered trademark of Microsoft Corporation.

† *VMS* is a registered trademark of Digital Equipment Corporation.

‡ *UNIX* is a registered trademark of UNIX Systems Laboratories.

1.3.2 Development System

Application system hardware development and software testing are supported by the *WE* DSP16A Development System. Each development system provides in-circuit emulation to facilitate real-time debugging of user hardware, as well as a simulator/accelerator to speed software simulations. Up to 16 development systems can be cascaded when developing applications that involve multiple DSPs.

While connected to the development system, the user may edit, assemble, and load programs, as well as utilize the software simulator. An assembled program can be transferred from the host into the development system's program memory through an internal bus of a parallel interface to the AT&T PC 6300 (or compatible) Personal Computer.

The software simulator and the development system can be used in three different modes:

- **Simulation Mode.** In this mode, the development system is not being used; program execution is being simulated in the host computer. This mode is used for program development and testing.
- **Hardware Mode.** The program has been downloaded into the development system and is being executed by the actual DSP16A device. Hardware mode is used for real-time program testing.
- **Simulator/Accelerator Mode.** The program is executed in the development system (as in the hardware mode), but data is supplied to and from the host. The simulator/accelerator mode is used to speed algorithm development by executing the program in the hardware of the development system rather than in the host computer.

1.3.3 Evaluation Board

The *WE* DSP16A Evaluation Board is a development tool for the DSP16A Digital Signal Processor that is used with an *IBM PC/XT/AT* * (or compatible) personal computer.

Included in the evaluation board interface software package is a program called **emu16** that provides a means to download an assembled DSP16A program onto the evaluation board and then run and monitor the behavior of the program. While the program is running, internal DSP16A registers, internal RAM, and external RAM can be clearly displayed. Other features, such as breakpoints and single-stepping, are also provided.

Although the evaluation board is not intended to replace the *WE* DSP16A Development System for hardware development, the evaluation board is a convenient tool for prototyping and testing application software.

---

* *IBM PC/XT/AT* are trademarks of International Business Machines Corporation.

1.4 DOCUMENTATION

This document is a reference guide for the DSP16A device. It describes the architecture, instruction set, and interfacing requirements of the both devices. Information on DSP16A programming techniques and several complete sample application programs are provided. The remaining chapters of this manual are outlined below:

- **Chapter 2. DSP16A Architecture** – a detailed description of the DSP16A device, including separate sections describing the major elements of the architecture and how they function.
- **Chapter 3. Instruction Set** – lists the complete instruction set of both devices and provides a description of each instruction, including restrictions and normal uses. Addressing modes are also discussed in detail.
- **Chapter 4. Device Programming** – discusses topics related to programming beyond the syntax of the instruction set. Possible problems are discussed, along with solutions and advice.
- **Chapter 5. Serial I/O** – a detailed analysis of the operation of the serial I/O port. Information specific to the serial I/O unit is provided that is essential to designs that utilize this port.
- **Chapter 6. Parallel I/O** – a detailed analysis of the operation of the parallel I/O port. Information specific to the parallel I/O unit and detailed information regarding the interrupt mechanism are provided that are essential to designs utilizing the parallel port.
- **Chapter 7. Interface Guide** – provides information regarding the physical design of the devices, including pin assignments, external memory interfacing, and reset and interrupt control.
- **Appendix A. Instruction Set Encoding** – lists the hardware-level encoding of the instruction set.
- **Appendix B. Programming Examples** – presents four complete sample application programs that demonstrate proper programming techniques.

1.4.1 Other Applicable Documentation

When designing application hardware and software, it is important to have accurate information. A variety of documents exist to provide specific information on various members of the DSP16A product family. Contact your AT&T Account Manager for the latest issue of any of the following documents.

- The *AT&T Digital Signal Processor Family Description* introduces the AT&T DSP family of products and provides a brief overview of the capabilities of its members.
- *WE® DSP16A Digital Signal Processor Information Manual* (this manual) is a reference guide for the DSP16A device. It describes the architecture, instruction set, and interfacing requirements. Information on programming techniques and several complete sample application programs are provided.
- *WE® DSP16A Digital Signal Processor* Data Sheet provides up-to-date timing requirements and specifications, electrical characteristics, and a summary of the instruction set and device architecture.
- *WE® DSP16A Support Software Library User Manual* provides the information necessary to install and use the DSP16A support software. This manual is also required when working with the DSP16A Development System, as the support software provides an interface between the host computer and the development system.

- *WE® DSP16A Application Software Library* contains descriptions of general-purpose DSP16A routines that may be used in a variety of applications.
- *WE® DSP16A Development System User Manual* provides the information necessary to set up and use the DSP16A Development System.
- *WE® DSP16A Evaluation Board User Manual* provides the information necessary to set up and use the DSP16A Evaluation Board.

1.5 ASSISTANCE

Assistance is available during conception, development, and throughout the life of the product. These services include:

- Technical documentation and product samples
- Information on determining and selecting the appropriate hardware and software
- The AT&T DSP Bulletin Board provides the latest and most up-to-date information about AT&T DSP products and application assistance:

  1200/2400 baud
  7 data bits, even parity
  1 stop bit
  215-778-4444

For technical assistance or further information including ordering information and part numbers, please contact the nearest office through the following phone numbers:

Domestic (USA & Canada)

Northeast Region
Boston
508-626-2161

North Central Region
Chicago
708-250-9777

Rocky Mountain Region
Denver
303-850-7000

South Central Region
Dallas
214-869-2040

Pacific Northwest
San Francisco
408-522-5555

Southern California
Los Angeles
714-220-6223

Southwest Region
Phoenix
602-844-6529

Southeast Region
Atlanta
404-446-4700
or
Orlando
407-875-4620

International

Europe (except Spain and Portugal)
+089 950 86 0
Telfax: +089 950 86 111

Spain and Portugal
+34 1 807 1441
Fax: +34 1 807 1420

Japan
(03)-5371-2700
Telex: J32562 ATTIJ
Fax: (03)-5371-3556

Pacific Rim
65-778-8833
Telex: RS 42898 ATTM
Fax: 65-777-7495

Internal (AT&T Customers)

AT&T internal customers should contact their local AT&T Account Management Office. If the Account Management telephone number is not known, call **1-800-372-2447** and ask for the telephone number of your account representative.

CHAPTER 2. DSP16A ARCHITECTURE

CONTENTS

2. DSP16A ARCHITECTURE

The major element of the DSP16A architecture is the memory, ROM and RAM; the cache; the address arithmetic units; the data arithmetic unit; the I/O, serial and parallel; and the control unit that connects these elements in a pipeline. Figure 2-1 shows a block diagram.

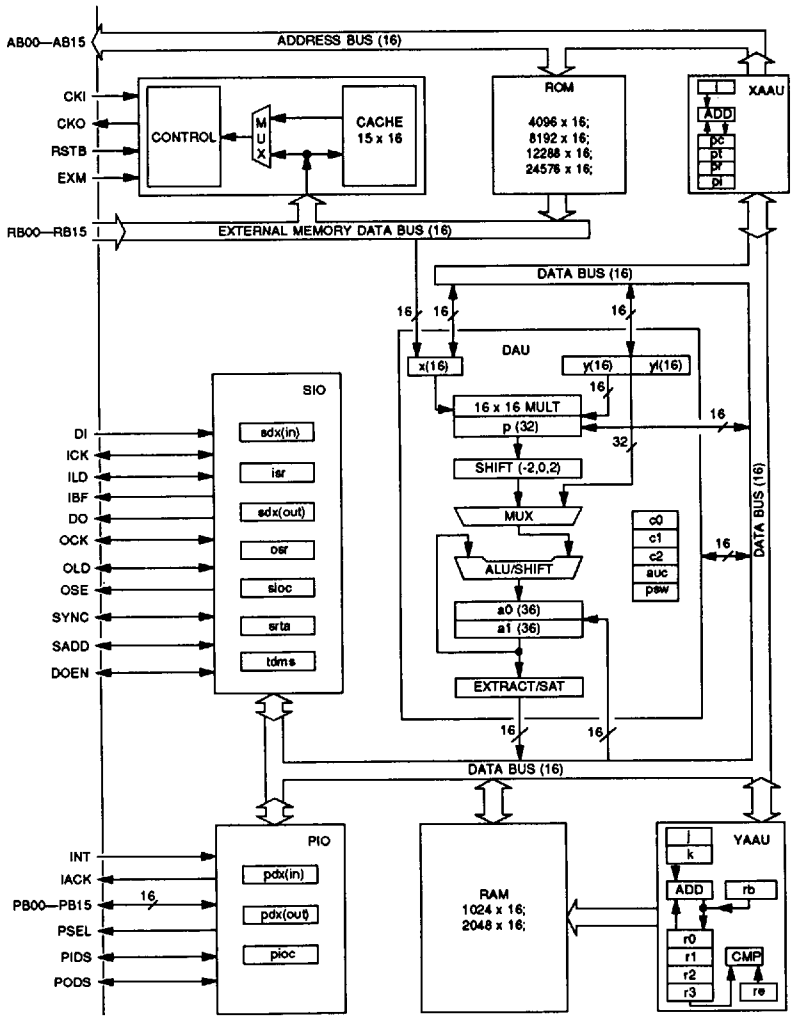


Figure 2-1. DSP16A Digital Signal Processor Block Diagram

2.1 MEMORY

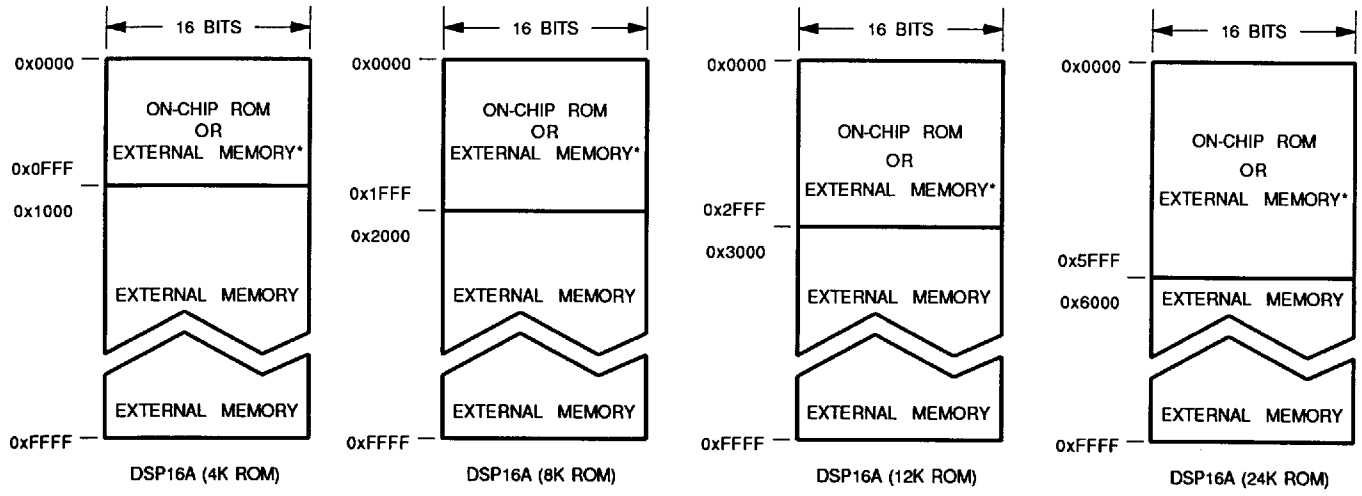
The DSP16A device provides both on-chip program memory and on-chip data memory. The program instructions and fixed operands are stored in the ROM. Instructions and immediate values are 16 bits wide and are fetched in one memory access each. Such variable operands as adaptive filter coefficients, state variables, intermediate results, and I/O data are stored in RAM. The ROM can either be replaced with up to 64 Kwords or augmented with additional off-chip program memory.

Table 2-1. DSP16A On-Chip Memory Size Options

Device	Program ROM	Data RAM	Program ROM Augmented Boundary
DSP16A	8,192 × 16	1,024 × 16	8,192
DSP16A	4,096 × 16	2,048 × 16	4,096
DSP16A	12,288 × 16	2,048 × 16	12,256
DSP16A	24,576 × 16	2,048 × 16	24,576

2.1.1 ROM

The DSP16A features a 4K × 16-bit, 8K × 16-bit, 12K × 16-bit, or 24K × 16-bit on-chip ROM. The on-chip ROM of the DSP16A can be replaced with up to 64 Kwords of external memory. The DSP16A also allows the use of both internal ROM and up to 60 Kwords, 56 Kwords, 52 Kwords, or 40 Kwords of external memory.



* When EXM signal is low.

Figure 2-2. DSP16A Program Memory Maps

2.1.2 RAM

The DSP16A provides a 2,048 × 16-bit or 1,024 × 16-bit on-chip, static RAM. RAM expansion is supported by the parallel I/O unit (see Chapter 6).

2.2 CACHE

The on-board cache memory selectively stores repetitive operations to increase the throughput and the coding efficiency of the DSP16A device. The cache can store up to 15 instructions at a time. The DSP16A device can be programmed to execute the instructions in the cache up to 127 times without having to use branching instructions. Instructions previously stored in the cache can be re-executed without reloading the cache.

Cache instructions eliminate the overhead when repeating a block of instructions. Therefore, the cache reduces the need to implement a routine that uses in-line coding in order to maximize the throughput. A routine utilizing the cache uses less ROM locations than an in-line coding of the same routine.

For two-operand multiply/arithmetic logic unit (ALU) instructions that do not require a write memory, decreasing the execution time from two instruction cycles to one instruction cycle results in an increase in throughput.

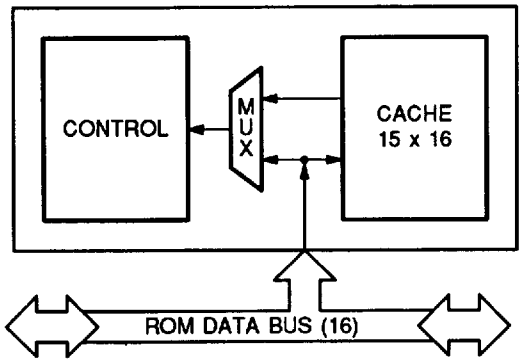


Figure 2-3. Control and Cache

2.3 CONTROL

The control block provides overall DSP16A system coordination. The instructions are decoded by hardware in the control block. The execution of the phases of an instruction is controlled by hardware throughout the DSP16A device. The hardware sequences instructions through the pipeline and controls the I/O, the processing, the memory accesses, and the timing necessary to perform each operation.

2.4 ADDRESS ARITHMETIC UNITS

Separate address arithmetic units for the on-chip ROM and RAM are provided. Each address arithmetic unit consists of static registers and an adder. Addresses are held by four of the registers in each address arithmetic unit. The remaining registers hold values that can be used to modify the address pointers. The adder is used to increment or decrement the addresses stored in the registers.

2.4.1 ROM Address Arithmetic Unit

The ROM address arithmetic unit (XAAU) consists of a 12-bit adder; a 12-bit static offset register, i; and four 16-bit static pointer registers: the program counter, pc; the program return, pr; the program interrupt, pi; and the table pointer, pt. These registers are used to address the ROM. The i register can be used to postmodify the pt register. The pt, pr, and i registers are user-accessible and can be modified under program control.

The pc register can be loaded with the address of a subroutine or branch directly from the control section. The program return address from a subroutine invoked using the call control instruction is saved in the pr register. The program return address from an interrupt is saved in the pi register. The pc register is loaded with the address in pr when returning from a subroutine or in pi when returning from an interrupt. The pc register can also be loaded from the pt register.

The pt register is normally used to point to tables of fixed data in ROM. The contents of the pt register can be modified by 1 or the value stored in the i register.

The i register contains a 12-bit, 2's complement signed number with a range of −2,048 to +2,047. The adder in the XAAU is used to postmodify the contents of the pt register. The data in the i register is sign-extended to 16 bits when transferred on the data bus. The XAAU has a 12-bit adder. Therefore, the address space can be viewed as sixteen 4 Kword pages. The adder is used to modify the 12 least significant bits (LSBs). The four most significant bits (MSBs) of the address may be changed by **goto pt** and **call pt** instructions.

Due to the XAAU 12-bit adder, pt can only be postincremented to 4095 by x = *pt++. But, since pt is a 16-bit unsigned register, it can be loaded with values to 64K.

The pi register is a 16-bit "shadow" register. Each time the pc register is modified, its new value is also loaded into the pi register. While in an interrupt service routine (ISR), this "shadowing" is disabled, and pi holds the last value of pc before the interrupt was taken. The return from interrupt instruction (**ireturn**) is simply a "goto pi" instruction.

The pi register may be read or written while in an ISR, but writing affects the return address. When not in an ISR, writing to pi has no effect on the contents of pi (the write to pi resets the pseudorandom sequence generator).

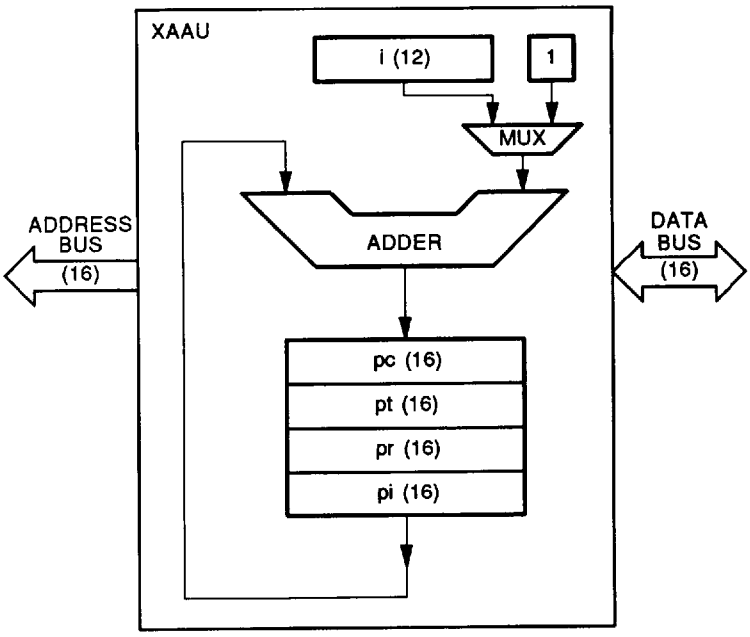
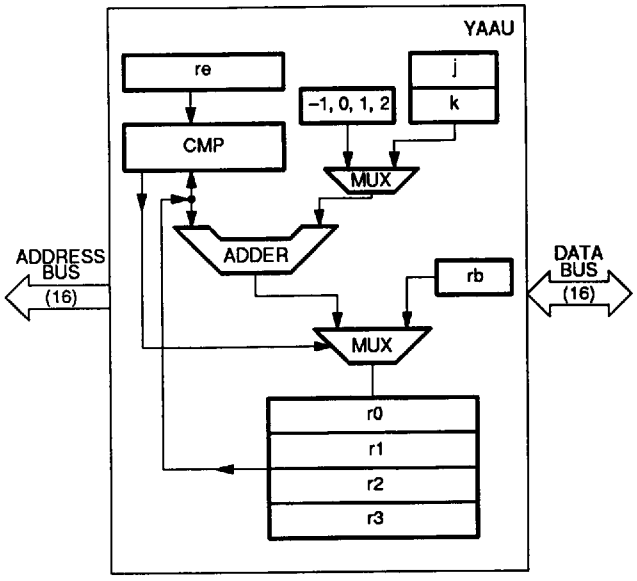


Figure 2-4. XAAU – ROM Address Arithmetic Unit



Note: All registers are 16 bits wide.

Figure 2-5. YAAU – RAM Address Arithmetic Unit

2.4.2 RAM Address Arithmetic Unit

The RAM address arithmetic unit (YAAU) consists of eight static registers and an adder. These registers are 16 bits wide. The RAM is addressed by the r0—r3 pointer registers. The j and k offset registers can be used to postmodify registers r0—r3. The rb and re registers are used when a register addressing the RAM is used in a cyclical (modulo) fashion. The eight YAAU registers are accessible to the user and can be loaded under program control.

The registers r0—r3 point to the RAM location that is the source of data to be loaded into the destination specified in the instruction or to the RAM location that is the destination of data from the source specified in the instruction. The r0—r3 pointers may be automatically postmodified by 0, +1, −1, +2, the contents of the j register, or the contents of the k register. The j and k registers contain 16-bit, 2's complement signed numbers with a range of −32,768 to +32,767 in the DSP16A. The adder in the YAAU is used to postmodify the contents of the r0—r3 registers.

Data in RAM can be addressed by using a virtual shift addressing mode. This addressing mode forms the equivalent of a cyclic shift register within the RAM.

Virtual shift addressing realizes a shift register for the FIR filter tap values without moving the data stored in RAM. The memory space allocated in RAM for the table of data to be addressed by using virtual shift addressing is defined by the addresses loaded into rb and re. The rb contains the physical address of the first location of the shift register; re points to the last entry. When a pointer is used, its value is compared with the contents of re. If they are equal and the postincrement is 1, the RAM address arithmetic unit writes the value of rb into the RAM pointer after the memory access. The DSP16A device can support virtual shift registers of up to 2,048 words in length. The virtual shift addressing mode is disabled when the value in re is 0. Register re is cleared (0) on reset.

2.5 DATA ARITHMETIC UNIT

The data arithmetic unit (DAU) is the main execution unit for signal processing algorithms. The DAU consists of a 16 × 16-bit multiplier, 36-bit ALU, and two 36-bit accumulators, a0 and a1. The DAU performs 2's complement, fixed-point arithmetic and is software configurable as a multiply/accumulate or ALU structure. The DAU multiplier and adder operate in parallel, each requiring one instruction cycle for their execution. Microprocessor-like instructions are executed by the ALU.

The multiplier executes a 16 × 16-bit multiply and stores the 32-bit product in the product register, p, in one instruction cycle. Data for the multiplier's inputs is stored in the 16-bit x register and the upper 16-bits (high half) of the 32-bit y register. The x register may be directly loaded from ROM, RAM, or the high half of an accumulator (bits 16—31 of the 36-bit word). The high half of the y register may be directly loaded from RAM or the high half of an accumulator.

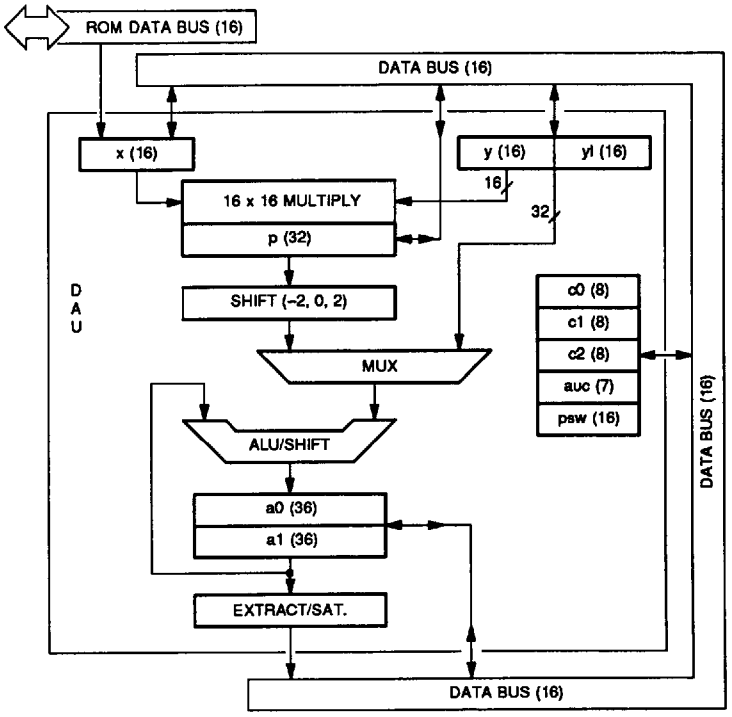


Figure 2-6. DAU – Data Arithmetic Unit

In addition to being used as an adder in the multiply/accumulate instructions, the 36-bit ALU provides the capability to implement functions and algorithms in the DSP16A device that conventionally would have been executed in a microcomputer or a microprocessor. Operands to the ALU can be data in y, p, a0, or a1. The ALU sign-extends 32-bit operands from y or p to 36 bits and produces a 36-bit output (32 bits of data and 4 guard bits) in one instruction cycle. Either accumulator can receive the 36-bit result. The ALU supports diadic functions with register y and an accumulator, including addition, subtraction, and logical AND, OR, and XOR. Monadic functions of an accumulator include rounding, negation, incrementation, and left and right shifts of 1, 4, 8, or 16 bits.

The y register is 32 bits wide. To read or write the low half of the y register (bits 0—15), yl is used in an assembly-language instruction. When y is used in an assembly-language instruction, the DSP16A device will read or write the high half (bits 16—31) of the y register. Automatic clearing of yl may be selected (according to the CLR field of the auc register) to simplify 16-bit operations. If clearing of the yl is enabled, the lower half of register y is cleared (0) with a write to the high half of the y register. Writes to yl do not change the data in the high half of y.

The p register is 32 bits wide. To read or write the low half of the p register (bits 0—15), pl is used in an assembly-language instruction. When p is used in an assembly-language instruction, the DSP16A device will read or write the high half (bits 16—31) of the p register. There is no automatic clearing of pl.

The accumulators are 36 bits wide. The contents of either the high half of the accumulator (bits 16—31) or the low half of the accumulator (bits 0—15) may be transferred to the 16-bit data bus. Automatic clearing of a0l or a1l may be selected (according to the CLR field of the auc register) to simplify 16-bit operations. If clearing of the low half of the accumulator is enabled, the lower half of the accumulator is cleared (0) with a write to the high half of the accumulator. Writes to the low half of the accumulator do not change the data in the high half of the accumulator. When the high half of an accumulator is loaded, the guard bits (35—32) are also loaded with the value of bit 31 and, thereby, sign-extended. Access to the guard bits for reading and writing is provided by the psw register.

The SAT field of the auc register allows the enabling of saturation on overflow when transferring the contents of accumulators onto the data bus. If saturation on overflow is enabled and either half of an accumulator is selected, the value transferred is saturated to 32 bits. If the selected accumulator has overflowed, then either the positive or negative (based on bit 35) saturation value is transferred.

$2^{31}-1$ is the positive saturation value.

-2^{31} is the negative saturation value.

A write of the contents of a 32-bit register to RAM requires two instructions: a write of the data in the high half of the register to RAM and a write of the data in the low half of the register to RAM. The order of the two writes to memory is left to the programmer. A read of the contents of RAM to a 32-bit register or accumulator also requires two instructions. If clearing of the low half of the destination's 32-bit register is enabled (according to the auc register, CLR field), the read of data in RAM to a 32-bit register must be done in the following order: load data to the high half of the register and load data to the low half of the register. This order is necessary because a load to the low half of a register does not change the data in the high half, while a load of the high half of a register clears the data from the low half. If clearing of the low half of the register is disabled, the two register loads may be performed in either order.

In addition to the registers already mentioned, the user has access to the arithmetic unit control register, auc; the processor status word register, psw; and the counters, c0—c2. The auc register configures some features of the data arithmetic unit. The psw register contains status information regarding the data arithmetic unit. Table 2-2 shows the contents of the psw register. The c0—c2 counters are 8 bits wide and may be used under program control to count events such as the number of times the program has executed a sequence of code.

Table 2-2. Processor Status Word (psw) Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DAU Flags				X	X	a1[V]	a1[35—32]				a0[V]	a0[35—32]			

Bit(s)	Field	Value*	Result/Description
15—12	DAU Flags	Wxxx	LMI – logical minus when set.
		xWxx	LEQ – logical equal when set.
		xxWx	LLV – logical overflow when set.
		xxxW	LMV – mathematical overflow when set.
11, 10	X	—	Reserved.
9	a1[V]	W	Accumulator 1 (a1) overflow when set.
8—5	a1[35—32]	Wxxx	Accumulator 1 (a1) bit 35.
		xWxx	Accumulator 1 (a1) bit 34.
		xxWx	Accumulator 1 (a1) bit 33.
		xxxW	Accumulator 1 (a1) bit 32.
4	a0[V]	W	Accumulator 0 (a0) overflow when set.
3—0	a0[35—32]	Wxxx	Accumulator 0 (a0) bit 35.
		xWxx	Accumulator 0 (a0) bit 34.
		xxWx	Accumulator 0 (a0) bit 33.
		xxxW	Accumulator 0 (a0) bit 32.

* W indicates that the bit may be read or written.

2.5.1 Arithmetic and Precision

Fixed-point, 2's complement arithmetic is used throughout the DSP16A device. The arithmetic bit alignment for the DSP16A device is shown in Figures 2-7, 2-8, and 2-9. The 16-bit data in the x register and in the high half of the y register can be multiplied together, and the 32-bit result is stored in the p register. The data in the y or p registers can be operated on by the ALU and the result stored in either of the 36-bit accumulators. The 32-bit data from the y or p register is sign-extended to 36 bits when operated on by the ALU.

For notational convenience, the 36-bit accumulators can be thought of as having an implied binary point to the right of bit 16. Bits 15—0 are then the fractional part, which is referred to as aNI, where aN = a0 or a1, and bits 35—16 are the integer part. The ALU operates on all 36 bits of the accumulators. The CLR field of the auc register controls automatic clearing of the low half of a0, a1, and y, making it easy to perform 16-bit integer operations in the ALU by automatically clearing the low half of the register when the high half is loaded.

The data transferred between an accumulator and memory or register must be scaled properly to reflect the bit alignment between p and a[0, 1] determined by the auc word. The user can select where the data in p is placed in the accumulator. Table 2-3 shows how two bits in the auc register, auc[1, 0], determine the bit alignment of the data in p with respect to the data in the accumulators. The connection of the data bus to the RAM, the accumulators, and the remaining registers in the DSP16A device is fixed.

Table 2-3. Arithmetic Unit Control (auc) Register

Bit	6	5	4	3	2	1	0
Field	CLR			SAT		ALIGN	

Bit(s)	Field	Value	Result/Description
6—4	CLR	1xx	Clearing yl is disabled (enabled when 0).
		x1x	Clearing a1l is disabled (enabled when 0).
		xx1	Clearing a0l is disabled (enabled when 0).
3, 2	SAT	1x	a1 saturation on overflow is disabled (enabled when 0).
		x1	a0 saturation on overflow is disabled (enabled when 0).
1, 0	ALIGN	00	Product register unshifted when used in multiply/ALU instructions aD = p, aD = aS + p, aD = aS − p.
		01	Product register shifted right 2 when used in multiply/ALU instructions aD = p ÷ 4, aD = aS + (p ÷ 4), aD = aS − (p ÷ 4).
		10	Product register shifted left 2 when used in multiply/ALU instructions aD = p × 4, aD = aS + (p × 4), aD = aS − (p × 4).
		11	Reserved.

Note: The auc register is not affected by reset.

If the auc[1, 0] bits are 00, the data in the p register is not shifted with respect to the bits in the accumulator before p[31—0] is transferred into bits 31—0 of an accumulator. The sign of p is extended by four bits, p[35—32], to provide overflow protection. The data transfer from p to an accumulator moves the overflow bits of the product into the guard bits 35—32 of the accumulator (see Figure 2-7 for the bit alignment in the DAU for auc[1, 0] = 00). This mode is most often used when both x and y operands are 16-bit integers.

If the auc[1, 0] bits are 10, the data in the p register is shifted 2 bits to the left with respect to the bits in the accumulator before p[31—0] are transferred into bits 33—2 of an accumulator. Bits 1 and 0 of the accumulator are not changed by the load of the accumulator with the data in p, since 00 is added to or copied into these accumulator bits as indicated in Figure 2-8. The sign of p is extended by two bits, p[33—32], to provide overflow protection. The data transfer from p to an accumulator moves the overflow bits of the product into guard bits 35—34 of the accumulator (see Figure 2-8 for the bit alignment in the DAU for auc[1, 0] = 10). This mode is often used in filtering applications where coefficients in the x register are in Q14 format (2 magnitude bits, 14 fractional bits), and state variables in the y register are 16-bit integers. If the p register is not shifted prior to accumulation, the accumulated result would have 4 guard bits, 18 magnitude bits, and 14 fractional bits. Since it is often desirable to have the implied binary point to the right of bit 16 (16 fractional bits), the setting auc[1, 0] = 2 automatically shifts the result 2 bit locations to the left generating an accumulated result with 4 guard bits, 16 magnitude bits, and 16 fractional bits.

If the auc[1, 0] bits are 01, the data in the p register is shifted 2 bits to the right with respect to the bits in the accumulator before p[31—2] are transferred into bits 29—0 of an accumulator. Bits p[1, 0] are not saved in the accumulator by the load of the accumulator with the data in p in this auc setting. The sign of p is extended by six bits, p[37—32], to provide overflow protection. The data transfer to an accumulator from p moves the overflow bits of the product into the guard bits 35—32 and bits 31—30 of the accumulator (see Figure 2-10 for the bit alignment in the DAU for auc[1, 0] = 01). This setting is most useful when avoiding overflow is a primary consideration, and the loss of the two LSBs of the product can be tolerated.

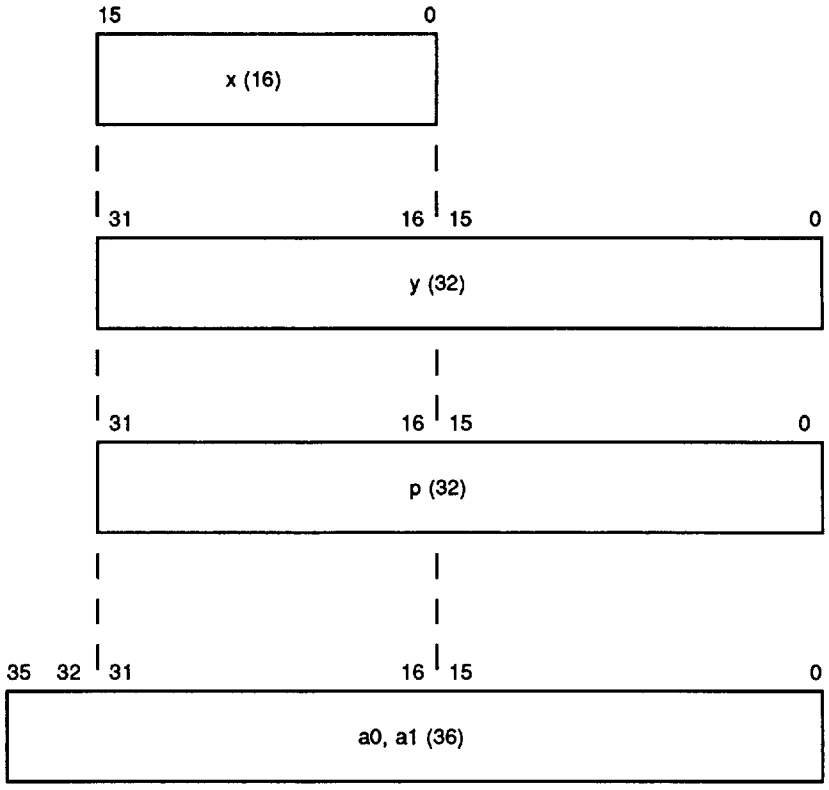


Figure 2-7. DSP16A Arithmetic Bit Alignment When auc[1, 0] = 00

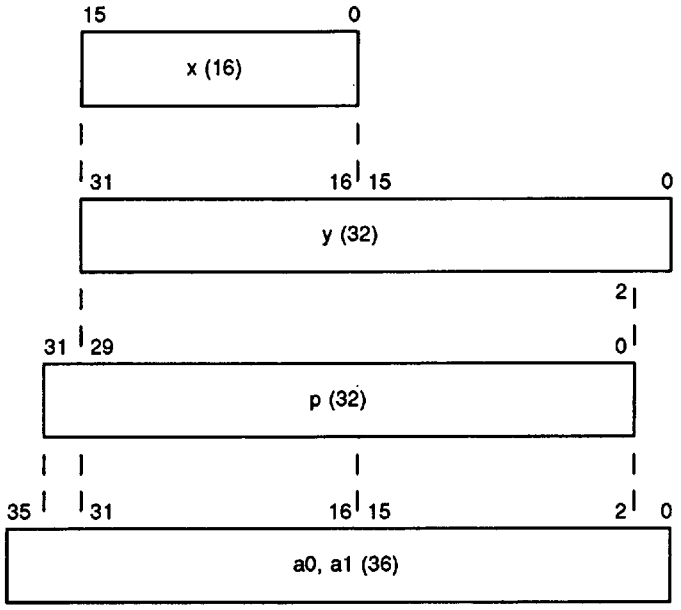


Figure 2-8. DSP16A Arithmetic Bit Alignment When auc[1, 0] = 10

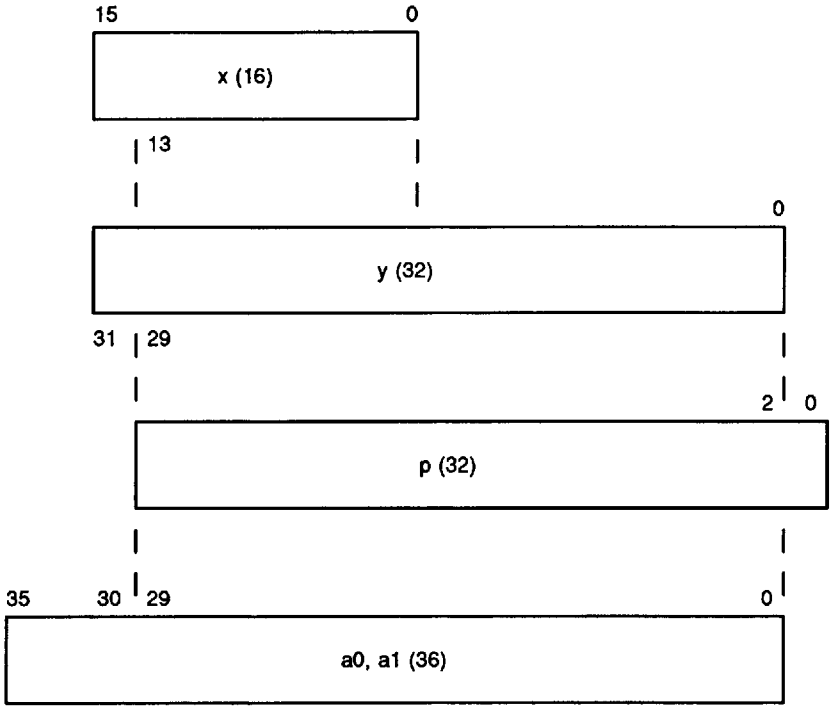


Figure 2-9. DSP16A Arithmetic Bit Alignment When auc [1, 0] = 01

2.6 SERIAL I/O

The serial I/O port (SIO) allows the DSP16A device to interface serially to other devices with few, if any, external chips (see Figure 2-10). The serial I/O port converts the serial input data stream to a parallel input data word and the parallel output data word to a serial output data stream. Serial I/O transfers are double-buffered to enable the DSP16A device to handle back-to-back serial transfers. The second serial transmission can begin before the data from the first serial transmission has been processed. The external DSP16A serial I/O control signals allow a zero chip interface to commercially available codecs and to the DSP16A, DSP32, and DSP32C devices for multiple DSP applications.

The serial I/O control register, sioc, allows the specification under program control of: the length of the serial input and output data words; the mode, active or passive, of the serial bit clocks; the mode, active or passive, of the serial load signals; bit ordering of the I/O; the active serial I/O bit clock rate; and active load generated from either ICK or OCK. The length of serial input and output data words can be 8 or 16 bits. The serial I/O bit clocks and the serial I/O load signals are transmitted by the DSP16A device to the rest of the external system in the active mode or are generated by the external system and transmitted to the DSP16A device in the passive mode. The mode of the input bit clock and load signal can be selected independent of the mode for the output bit clock and load signal. Active serial I/O bit clock rates of CKI/4, CKI/12, CKI/16, and CKI/20 can be selected. The bit order of the serial input data can be specified to be bit-reversed when the input is moved from the serial input buffer to the destination register or RAM location; the bit order of the data can be specified to be bit-reversed when data is transferred to the serial output buffer from the source register or RAM location. Bit reversal of the data is necessary for μ-law and A-law conversion and is performed in hardware rather than in software. See Chapter 5.

The tdms register specifies the time slot of the DSP in a time-division multiplexed signal, whether the DSP is operating in a single or multiple DSP16A environment and the active frame synchronization signal clock rate, f/128 or f/256.

The synchronization clock can be active or passive. In the multiple DSP environment, a DSP16A device can directly address a maximum of seven other DSP16A devices via the serial I/O port. The 16-bit srta register is loaded with two 8-bit addresses. One address specifies the receiving DSP identity; the second address identifies the destination DSP, where the serial data is accepted. See Chapter 5 for more detailed information.

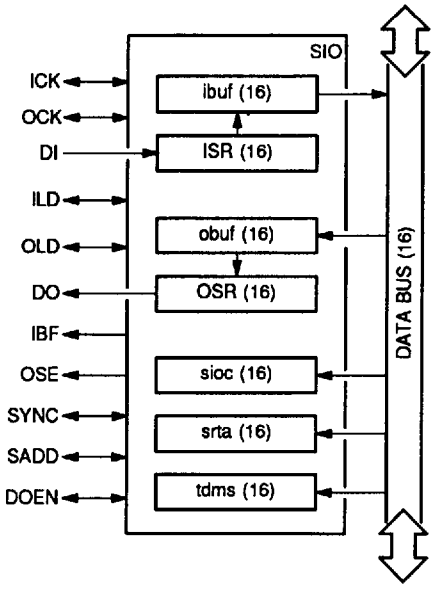


Figure 2-10. SIO – Serial Input/Output Unit

2.7 PARALLEL I/O

The DSP16A parallel I/O port (PIO) provides a 16-bit, bidirectional data link to microprocessors and other I/O devices (see Figure 2-11). The parallel I/O port supports bidirectional communication with other devices over a wide range of data transfer rates.

The parallel I/O control word, pioc, allows the specification, under program control, of the configuration of the PIO data pins; the mode, active or passive, of the parallel I/O data strobe signals; and the width of the parallel I/O data strobe signals in the active mode. The 16-bit PIO data bus can be configured to transmit and receive 8 bits simultaneously or, under program control, either to input 16 bits or output 16 bits. The parallel I/O data strobe signals are transmitted by the DSP16A device to the external system in the active mode or are generated by the external system and transmitted to the DSP16A device in the passive mode. The mode of the input data strobe signal can be selected independent of the mode for the output data strobe signal. In the active mode, the pulse width of the strobe signals can be selected. The contents of the pioc register can be read to determine the status of the DSP16A serial and parallel I/O ports. The pioc register contains a bit field with the status of the serial and parallel I/O flags. Under program control, this bit field can be read to determine the state of the I/O flags; the program cannot write this bit field in the pioc register.

The DSP16A device can directly address two devices via the PIO ports, pdx0 and pdx1. The signal level on the PIO address line, psel, is low when data is written to pdx0 and high when data is written to pdx1. A DSP16A instruction writes to either pdx0 or pdx1 to output data via the PIO port. A DSP16A instruction reads pdx0 or pdx1 to access the data input to the PIO port by external devices. The PIO data strobe signals are asserted when data is written to the PIO port by the DSP16A device or by external devices. The PIO port can be used to interface the DSP16A device with a large external RAM memory.

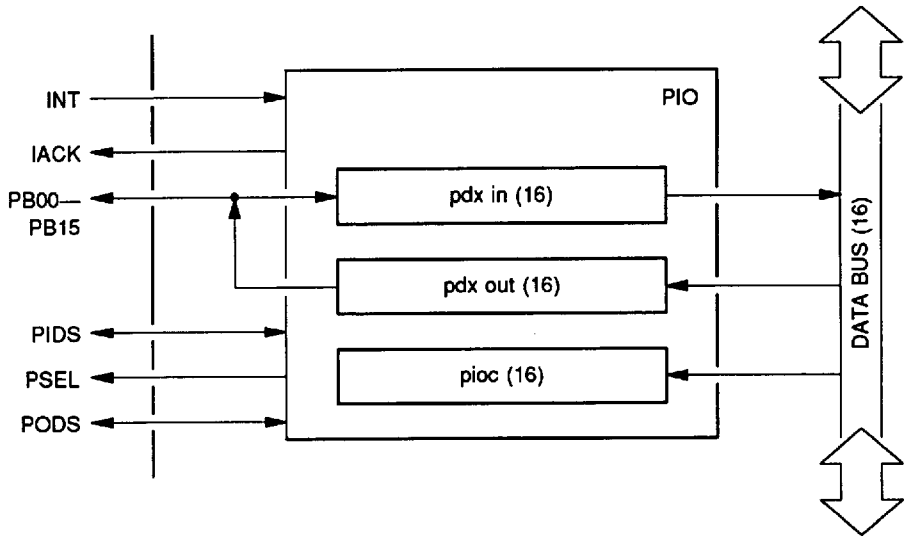


Figure 2-11. PIO – Parallel Input/Output Unit

2.8 INTERRUPTS

The DSP16A device has one external interrupt request signal and four internal interrupt request signals. Hence, it can respond to one external and four internal conditions. These conditions are:

- **INT – Interrupt by an External Device.** An external device has requested service by asserting the INT pin.
- **IBF – Input Buffer Full.** Indicates that an external device has written data into the device's serial input buffer.
- **OBE – Output Buffer Empty.** Indicates that an external device has read data from the SIO serial output buffer.
- **PIDS – Parallel Input Data Strobe.** Indicates that an external device has written data into the PIO parallel input register.
- **PODS – Parallel Output Data Strobe.** Indicates that an external device has read the data from the PIO parallel output buffer.

Each interrupt is maskable; that is, by appropriately clearing bits in the pioc register, the associated interrupt is ignored. The interrupt mask bits, pioc[5—9], are cleared (0) on reset, thereby disabling all interrupts.

An interrupt causes a **goto 1** instruction to be jammed into the instruction register. When in an interrupt service routine, the shadowing of the pc register is disabled (see Section 2.4.1). The DSP16A device also provides a software interrupt facility. The instruction **icall** causes the same sequence of events to occur as any other interrupt source, except the branch address of the interrupt service routine is 2 rather than 1. Note that the branch address for interrupts is always at location 1 or 2 (the branch is not limited to the current 4 Kword page).

Table 2-4. Parallel I/O Control (pioc) Register

Bit	15	14	13	12	11	10	9—5	4—0
Field	IBF	STROBE		PODS	PIDS	S/C	INTERRUPTS	STATUS

Bit(s)	Field	Value*	Result/Description
15	IBF	R	IBF interrupt status bit (same as bit 4).
14, 13	STROBE	00 01 10 11	Strobe width of **POD†** **PIDS†** T T 2T 2T 3T 3T 4T 4T
12	PODS	0	PODS is an input (passive mode).
		1	PODS is an output (active mode).
11	PIDS	0	PIDS is an input (passive mode).
		1	PIDS is an output (active mode).
10	S/C	0	Not S/C mode.
		1	S/C mode.
9—5	INTERRUPTS	1xxxx	IBF interrupt is enabled (disabled when 0).
		x1xxx	OBE interrupt is enabled (disabled when 0).
		xx1xx	PIDS interrupt is enabled (disabled when 0).
		xxx1x	PODS interrupt is enabled (disabled when 0).
		xxxx1	INT interrupt is enabled (disabled when 0).
4—0	STATUS	Rxxxx	IBF status bit.
		xRxxx	OBE status bit.
		xxRxx	PIDS status bit.
		xxxRx	PODS status bit.
		xxxxR	INT status bit.

* R indicates a read-only bit.
† T = 2 × tCKIHCKIH.

2.8.1 Hardware Description

See Section 5.1 for more information on serial I/O operations and interrupts. See Section 6.3 for more information on parallel I/O interrupts.

The external interrupt mechanism has two relevant signals: INT and IACK.

- **INT – External Interrupt Request.** When asserted, this input indicates to the DSP16A device that an external device is requesting service. To guarantee that this request is serviced, the INT signal must remain asserted for twice the period of the DSP16A device's CKO signal (i.e., 4 × tCKIHCKIH). Asserting INT sets the status bit (pioc bit 0) only when the INT interrupt has been enabled. Note that if INT is previously asserted and IACK is not high to clear it, enabling the INT interrupt will set the INT status bit and generate an interrupt.

- **IACK – Interrupt Acknowledge.** When asserted, this output indicates that one of the internal or the external interrupt requests has been recognized by the DSP16A device. Once the IACK signal has been asserted, it is negated upon completion of the interrupt service routine (i.e., upon execution of an ireturn instruction).

2.8.2 Software Description

After recognizing an interrupt, the DSP16A device completes execution of the current instruction and then branches to address 1. Branch and conditional branch instructions and instructions executing within the cache are not interruptible. If an interrupt request occurs during an uninterruptible instruction, the DSP16A device does not recognize the request until an interruptible instruction is encountered. See Chapter 3 for more information on the DSP16A instruction set.

The interrupt service routine entered should clear the respective status bit in the pioc register. If this is not done, the DSP16A device repeatedly recognizes the interrupt until the appropriate status bit in the pioc is cleared. (This can be useful in certain applications.)

Individual status bits are cleared in the following manner:

- **IBF – Input buffer full** (pioc bits 4 and 15) is cleared by reading sdx, the serial input buffer.
- **OBE – Output buffer empty** (pioc bit 3) is cleared by writing to sdx, the serial output buffer.
- **PIDS – Parallel input data strobe** (pioc bit 2) is cleared by reading the parallel input buffer, pdx0 or pdx1.
- **PODS – Parallel output data strobe** (pioc bit 1) is cleared by writing to the parallel output buffer, pdx0 or pxd1.
- **INT – Interrupt by an external device** (pioc bit 0) is cleared when IACK makes a high-to-low transition (initiated by an **ireturn** instruction), indicating the end of the interrupt service routine.

The interrupt status bits are also cleared on reset.

See Section 6.3 for more information on interrupts utilizing the PIO and Section 4.2.5 for techniques on handling multiple interrupts.

2.9 LOW-POWER SLEEP MODE

The DSP16A has a sleep state that reduces power when the device is waiting. This is implemented by stopping the internal processor clock when a one is written to the SLEEP bit in the CIOC register. An interrupt awakens the device, restarting the clock to the internal processor. The external INT, SIO, and PIO remain active during this sleep state to generate the interrupts that awaken the device. Four NOPs need to be placed after the write to CIOC with the SLEEP bit set in order to insure that the processor is inactive when going to sleep. The processor will stop on the second NOP after the write to CIOC and then execute the third and fourth NOPs after an interrupt awakens the processor, but before the jump to address one. Remember that cache operations cannot be interrupted, hence the SLEEP bit in the CIOC register should not be written from a cache operation.

Table 2-5. Sleep Control (cioc) Register

Bit	15—13	12	11—0
Field	RES	SLEEP	RES

Bit(s)	Field	Value	Result/Description
15—13	RES	xxx	Reserved.
12	SLEEP	0 1	Normal Operation. Processor clock stopped until interrupted.
11—0	RES	xxxx xxxx xxxx	Reserved.