9 DMA PORTS

Overview

The ADSP-218x processors include the following DMA interfaces:

- **Byte Memory Space and Byte Memory DMA (BDMA) —** The byte memory space can address up to 4M bytes. The BDMA interface supports booting from and runtime access to inexpensive 8-bit memories. The BDMA feature lets you define the number of memory locations the BDMA interface transfers to or from internal memory in the background while the ADSP-218x DSP core continues processing in the foreground.
- **Internal Direct Memory Access (IDMA) Port —**This 16-bit wide parallel port supports booting from and runtime access to host systems (for example, PC Bus Interface ASICs). The DMA feature of this port lets you transfer data to/from internal memory in the background while continuing foreground processing.

These DMA transfers are accomplished internally by "cycle stealing," in the same way as serial port autobuffering. This means that the ADSP-218x processor uses internal bus cycles to transfer the data to and from memory. The stolen cycles only occur at instruction cycle boundaries— not between cycles of a multiple-cycle instruction. See "DMA Cycle Stealing, Hold Offs, and IACK Acknowledge" on page 9-47 for additional details.

BDMA Port

Byte memory provides access to an 8-bit wide memory space through the BDMA port. The byte memory space provides access to 4 Mbytes of memory by utilizing 8 data lines as additional address lines. This gives the BDMA port an effective 22-bit address range. Figure 9-1 shows the ADSP-218x processor interface to BDMA.

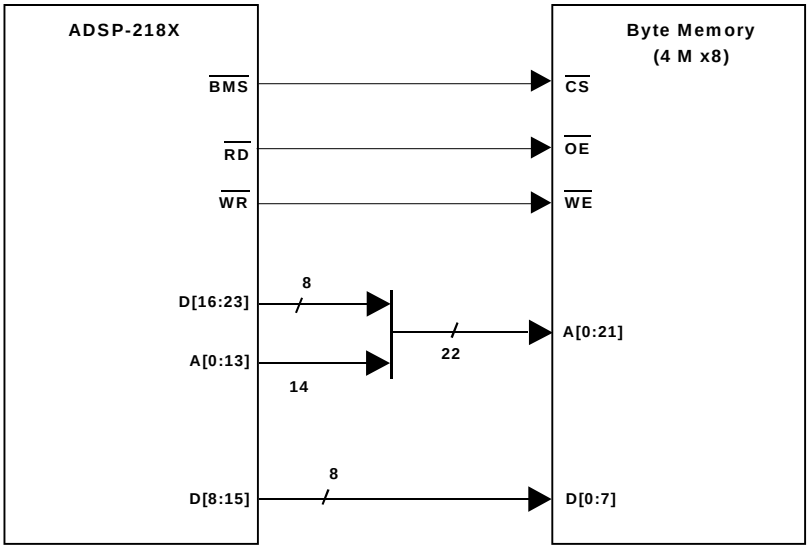


Figure 9-1. ADSP-218x Processor BDMA Port Interface

Byte memory space consists of 256 pages, each containing 16 K x 8-bit wide locations. This memory can be written and read in four different formats: 24-bit, 16-bit, 8-bit MSB alignment, and 8-bit LSB alignment.

On powerup, the ADSP-218x processor can automatically load bootstrap code from byte memory. To use byte memory for purposes other than boot loading, such as runtime access to bulk data storage, you must know the following:

- 22-bit address that the code/data starts (BMPAGE, BEAD)
- Number of words (BWCOUNT) to read or write
- Word format (BTYPE) of the data

A BDMA transfer (non-boot loading) begins when data is written to the BWCOUNT register and completes when the BWCOUNT register decrements to zero. When the register reaches zero, a BDMA interrupt is issued.

There are no restrictions to the byte address alignment for BDMA accesses. The upper 8 bits of the address are from BMPAGE and the lower 14 bits are from BEAD. As the BDMA controller accesses memory, it automatically increments BMPAGE at page boundaries. Data or code can cross BMPAGE boundaries without a problem.

The following restrictions apply to BDMA transfers:

- The BDMA Internal Address register (BIAD) lets you set the 14-bit internal starting address for the BDMA transfer.

To access the internal Program Memory and Data Memory Overlay regions for the ADSP-2187L, ADSP-2188M, and ADSP-2189M processors via BDMA, the BIAD register should be set to 0x2000-0x3fff for Program Memory Overlay regions and 0x0000-0x1fff for Data Memory Overlay regions. The BDMA Overlay bit field (bits 7:4 of the BDMA Control register) and the BTYPE field (bits 1:0 of the BDMA Control register) should also be set for the appropriate overlay regions.

- The BEAD or BIAD registers should not be accessed during BDMA transfers.

- Other external memory accesses (PM Overlay, DM Overlay, or I/O space) take precedence over BDMA port accesses. These accesses cannot occur at the same time because they also use the processor's external bus. (See "Priority Chain" on page 9-49 for more information.)
- Powerdown mode should not be entered with the BDMA port active. (See "Powerdown" in Chapter 7, "System Interface" for more information on powerdown restrictions.)

BDMA Port Functional Description

The BDMA port lets you load or store program instructions and data from or to byte memory with very low processor overhead. While the ADSP-218x processor is executing program instructions, the BDMA port reads or writes code or data from or to byte memory—stealing one ADSP-218x processor cycle per word when it needs to write to or read from internal memory. You can calculate BDMA transfer time from the following formula:

$$\left(\begin{array}{c}\text{Number}\\ \text{of PM}\\ \text{or DM}\\ \text{Words}\end{array}\right)\left[\left(\begin{array}{c}\text{Number}\\ \text{of Bytes}\\ \text{per Word}\end{array}\right)\left(\begin{array}{c}\text{Number}\\ \text{of Added}\\ \text{Wait States}\\ \text{per Byte}\end{array}\right)+\left(\begin{array}{c}1\\ \text{Cycle}\\ \text{for}\\ \text{Transfer}\end{array}\right)+\left(\begin{array}{c}1\\ \text{Cycle for}\\ \text{Internal}\\ \overline{\text{RD}}/\overline{\text{WR}}\end{array}\right)\right]+\left(\begin{array}{c}\text{Hold}\\ \text{Offs}\end{array}\right)$$

If, for example, you wanted to transfer one hundred 24-bit program memory words through the BDMA port, assuming five wait states and no hold offs, the operation would take 1900 cycles. This is shown in the following equation:

$$\left(\begin{array}{c}100\\ \text{PM}\\ \text{Words}\end{array}\right)\left[\left(\begin{array}{c}3\\ \text{Bytes}\\ \text{per Word}\end{array}\right)\left(\begin{array}{c}5\\ \text{Added}\\ \text{Wait States}\\ \text{per Byte}\end{array}+\begin{array}{c}1\\ \text{Cycle}\\ \text{for}\\ \text{Transfer}\end{array}\right)+\left(\begin{array}{c}1\\ \text{Cycle for}\\ \text{Internal}\\ \overline{\text{RD}}/\overline{\text{WR}}\end{array}\right)\right]+\left(\begin{array}{c}0\\ \text{Hold}\\ \text{Offs}\end{array}\right)$$

Hold offs for DMA transfers are defined in the section, "DMA Cycle Stealing, Hold Offs, and IACK Acknowledge" on page 9-47.

BDMA Control Registers

Memory-mapped registers are used to set up and control transfers through the BDMA port. Figures 9-2 through 9-7 show these registers.

The BDMA Internal Address register (BIAD) lets you set the 14-bit internal starting address for the BDMA transfer. To access the internal Program Memory and Data Memory Overlay regions for all the ADSP-2187, ADSP-2188, and ADSP-2189 processors via BDMA, the BIAD register should be set to 0x2000-0x3fff for Program Memory Overlay regions and 0x0000-0x1fff for Data Memory Overlay regions. The BDMA Overlay bit field (bits 7:4 of the BDMA Control register) and the BTYPE field (bits 1:0 of the BDMA Control register) should also be set for the appropriate overlay regions.

The BDMA Internal Address register (BIAD) lets you set the 14-bit internal starting address for the BDMA transfer (see Figure 9-2). To access the internal Program Memory and Data Memory Overlay regions for all the ADSP-2187, ADSP-2188, and ADSP-2189 processors via BDMA, the BIAD register should be set to 0x2000-0x3fff for Program Memory Overlay regions and 0x0000-0x1fff for Data Memory Overlay regions. The BDMA Overlay (BMOVLAY) field (bits 7:4 of the BDMA Control register) and the BTYPE field (bits 1:0 of the BDMA Control register) should also be set for the appropriate overlay regions.

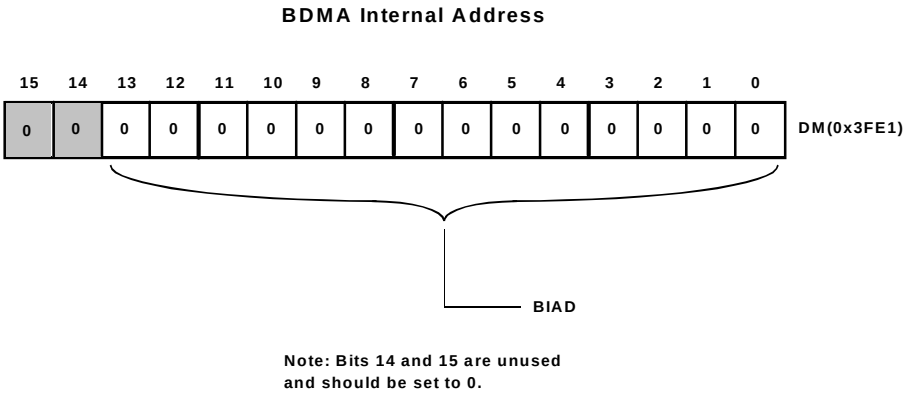


Figure 9-2. BDMA Internal Address Register

The BDMA External Address register (BEAD) lets you set the 14-bit external memory starting address for a BDMA transfer (see Figure 9-3). This register value represents the value of the address bits A[13:0], which are driven on the external address bus to your byte-wide device.

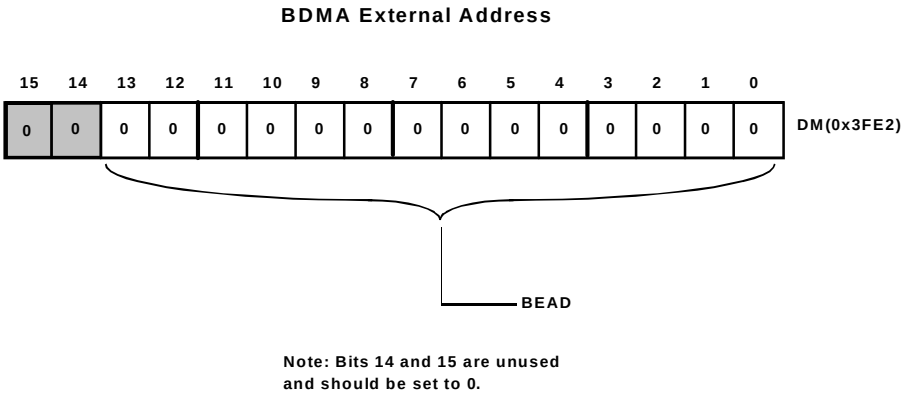


Figure 9-3. BDMA External Address Register

The BDMA port can access up to 4M bytes of information by using 22 bits of external addressing. These 22 bits are comprised of the following:

- Address bits 13:0, which correspond to the value of the BEAD register
- Address bits 21:14, which constitute the BMPAGE field (bits 15:8 of the BDMA Control register)

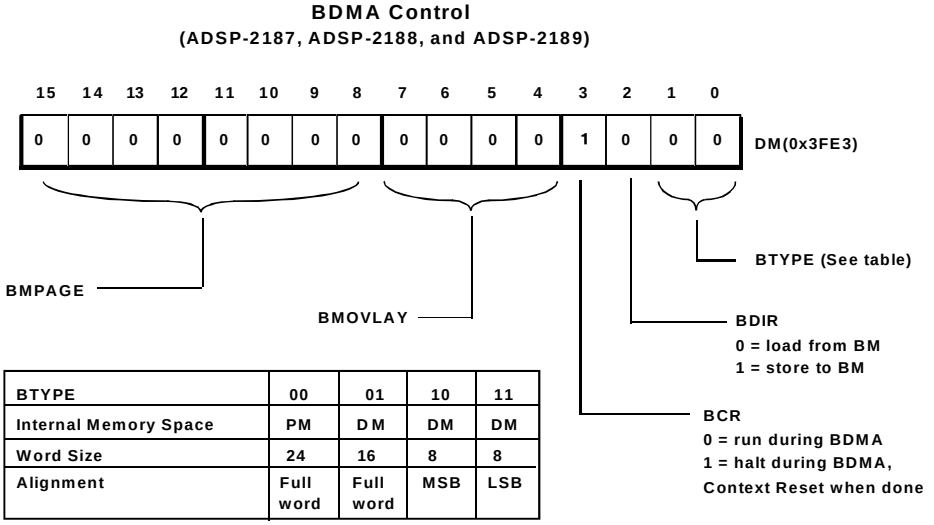
Table 9-1 lists the values driven onto the external bus for BDMA addressing.

Table 9-1. BDMA External Addresses

BDMA address	DSP Pins Used	BDMA Register Field
A21:A14	D23:D16	BDMA Control register bits 15:8 (BMPAGE)
A13:A0	A13:A0	BDMA External Address (BEAD) bits 13:8

BDMA transfers that cross BDMA page boundaries update the BMPAGE field of the BDMA Control register automatically.

BDMA Overlay bits (bits 7:4 of the BDMA Control register, shown in Figure 9-4) apply only to the ADSP-2187, ADSP-2188, and ADSP-2189 processors. These bits must be set to zero for all other ADSP-218x processors (see Figure 9-5).



BTYPE	00	01	10	11
Internal Memory Space	PM	DM	DM	DM
Word Size	24	16	8	8
Alignment	Full word	Full word	MSB	LSB

Figure 9-4. BDMA Control Register (ADSP-2187, ADSP-2188, and ADSP-2189)

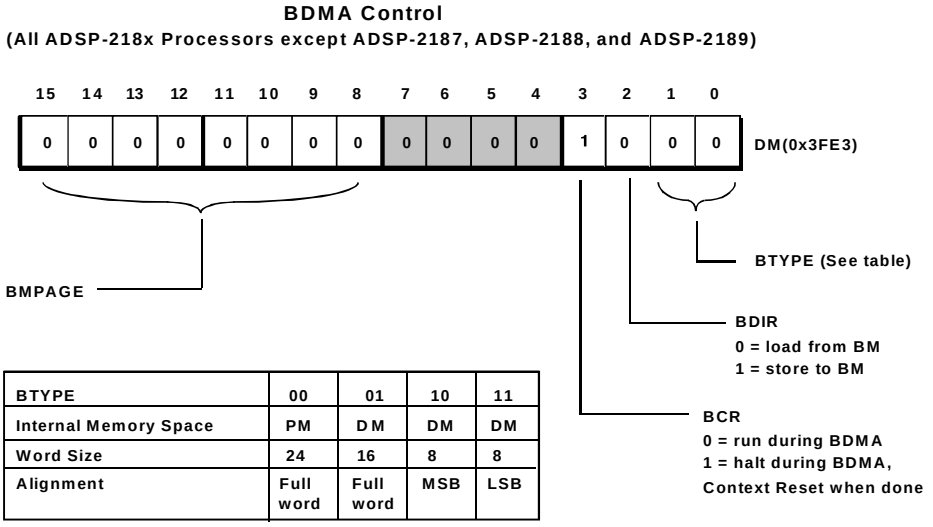


Figure 9-5. BDMA Control Register (All ADSP-218x Processors except the ADSP-2187, ADSP-2188, and ADSP-2189)

The BDMA Control register lets you set:

- BDMA Transfer Type (BTYPE)
- BDMA Direction (BDIR)
- BDMA Context Reset (BCR)
- Internal Overlay pages to be accessed by the BDMA transfer (applies to the ADSP-2187, ADSP-2188, and ADSP-2189 processors only)
- BDMA Page (BMPAGE)

BTYPE can be:

- 00 24-bit Program Memory Words
- 01 16-bit Data Memory
- 10 8-bit bytes for Data Memory, MSB alignment
- 10 8-bit bytes for Data Memory, LSB alignment

BDIR can be:

- 0 from Byte Memory
- 1 to Byte Memory

BCR can be set to:

- 0 Allows program execution during BDMA
- 1 Inhibits program execution during BDMA transfers and causes a context reset after transfer is complete

BMPAGE lets you select the starting page for BDMA transfer.

Rebooting with BDMA Context Reset (BCR=1) is similar to a Powerup Context Reset. For more details on processor states during reset and reboot, see Chapter 7, "System Interface" in this manual.

The BWCOUNT register (shown in Figure 9-6) lets you start a BDMA transfer by writing the number of words for the transfer to this register. The count automatically decrements as the transfer proceeds. When the count is zero (i.e. transfer complete), the processor issues a BDMA interrupt. When MMAP and BMODE (ADSP-2181 and ADSP-2183 processors) or Mode B (all other processors) are set to zero on boot, a value of 32 (decimal) is written to this register directing the ADSP-218x processor to load the first 32 locations of its internal program memory.

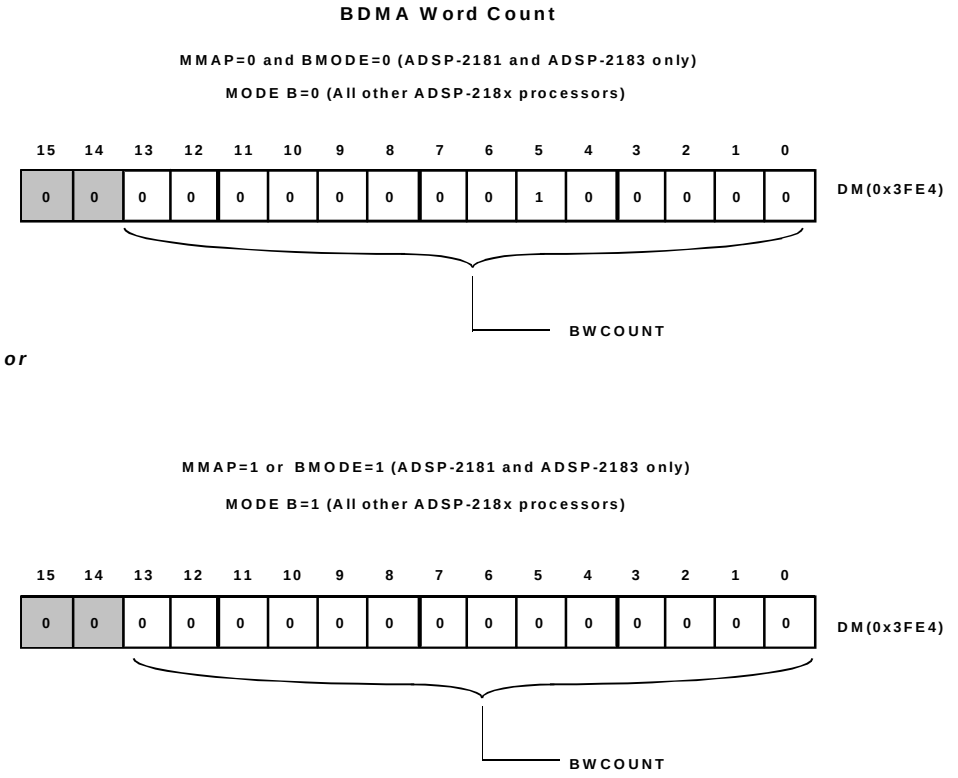


Figure 9-6. BDMA Word Count Register

Two useful control techniques using this register are:

- Poll the BWCOUNT register to determine when the DMA transfer is complete (BWCOUNT=0) instead of waiting for the BDMA interrupt. The following code example illustrates this technique:

```
Poll_BWCOUNT:

   ax0 = dm(0x3fe4);        /* read value of BDMA count
                               register */
   ar = pass ax0;           /* pass count value through
                               ALU */
   if eq jump BDMA_done;    /* if count value = 0 then BDMA is
                               complete */
   jump Poll_BWCOUNT;       /* else continue polling the BDMA
                               count register */
```

- Abort the DMA operation by writing a 1 to the BWCOUNT register and poll to determine when the transfer is complete (BWCOUNT=0) instead of waiting for the BDMA interrupt. (Note that the DMA transfer is aborted and cannot be resumed later.)

  Writing a zero to the BWCOUNT register results in 16 K words transferred.

BMWAIT consists of bits 12, 13, and 14 of the Composite Select Control register for all ADSP-218x processors except the M and N series (see Figure 9-7). For the M and N series processors, this field consists of bits 12, 13, 14, and 15 of the Composite Select Control register (see Figure 9-8). BMWAIT lets you select 0-7 wait states (each equal to a single instruction cycle) to apply to each byte memory access. BMWAIT is set to 7 after a reboot.

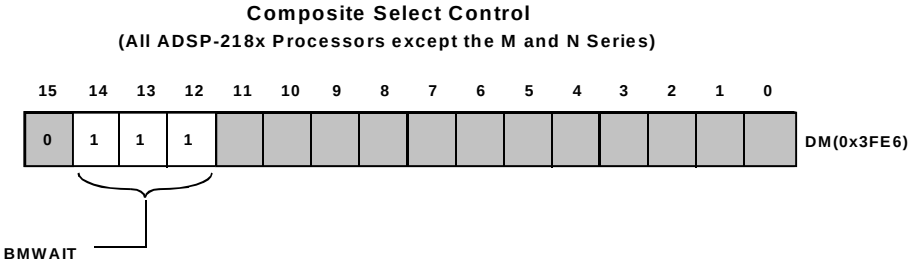


Figure 9-7. BMWAIT Field in Composite Select Control Register (All ADSP-218x Processors except the M and N Series)

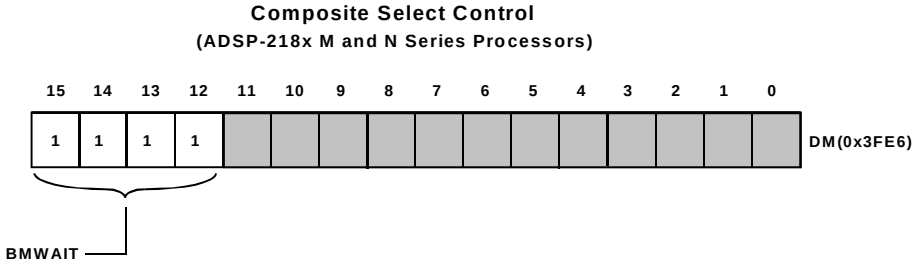


Figure 9-8. BMWAIT Field in Composite Select Control Register (ADSP-218x M and N Series)

Byte Memory Word Formats

In your byte memory ROM or RAM, data is stored by the ADSP-218x PROM Splitter according to the data format you select: 24-bit program memory words, 16-bit data memory words, 8-bit data memory bytes with MSB-alignment, or 8-bit data memory bytes with LSB-alignment. The byte order for 24-bit program memory words and 16-bit data memory words stored in byte memory is most-significant-byte in the lower address. Table 9-1 shows an example of byte memory storage for all four code/data formats.

BDMA transfers to/from internal memory are written to/stored from 16-bit wide locations. When transferring either of the data memory byte formats, the unused byte of Data Memory is zero-filled.

Table 9-2. Byte Memory Storage Formats

BTYPE	Internal Memory Address	Internal Memory Contents	Byte Memory Address (page 0x00)	Byte Memory Contents
00	PM(0x0000)	0xABCDEF	BM(0x0000) BM(0x0001) BM(0x0002)	0xAB 0xCD 0xEF
00	PM(0x0001)	0x123456	BM(0x0003) BM(0x0004) BM(0x0005)	0x12 0x34 0x56
01	DM(0x0000)	0x9876	BM(0x0006) BM(0x0007)	0x98 0x76
01	DM(0x0001)	0x3456	BM(0x0008) BM(0x0009)	0x34 0x56

Table 9-2. Byte Memory Storage Formats

BTYPE	Internal Memory Address	Internal Memory Contents	Byte Memory Address (page 0x00)	Byte Memory Contents
10	DM(0x0002)	0x9800	BM(0x000A)	0x98
10	DM(0x0003)	0x7600	BM(0x000B)	0x76
11	DM(0x0004)	0x0034	BM(0x000C)	0x34
11	DM(0x0005)	0x0056	BM(0x000D)	0x56

BDMA Booting

The ADSP-218x processor offers two methods for automatic booting after reset: BDMA booting and IDMA booting. This section describes BDMA booting. For information about IDMA booting, see "Boot Loading through the IDMA Port" on page 9-46.

When using BDMA booting, the entire on-chip program memory of an ADSP-218x processor, or any portion of it, can be loaded from an external source using a byte memory booting sequence. Table 9-3 shows how to select the post-reset booting method using the `MMAP` and `BMODE` pins on the ADSP-2181 and ADSP-2183 processors. Table 9-4 on page 9-16 shows how to select the post-reset booting method using the `Mode D`, `Mode C`, `Mode B`, and `Mode A` pins on all other processors.

The `Mode D` pin is not available on the ADSP-2184, ADSP-2184L, ADSP-2185, ADSP-2185L, ADSP-2186, and ADSP-2186L processors.

Table 9-3. Booting Methods for the ADSP-2181 and ADSP-2183 Processors

MMAP	BMODE	Booting Method
0	0	**Boot through BDMA port.** Boot sequence loads the first 32 program memory words from the byte memory space. After all 32 words are loaded, program execution begins at internal address PM(0x0000) with a BDMA interrupt pending.
0	1	**Boot through IDMA port.** Boot sequence holds off execution while the host processor loads Program Memory using writes through the IDMA port. Program execution begins when internal address PM(0x0000) is loaded.
1	–	**No Booting.** Boot sequence does *not* load memory *or* hold off execution. Program execution starts at external address PM(0x0000). The PMOVLAY register must be cleared (to zero).

Table 9-4. Booting Methods for All Processors Except the ADSP-2181 and ADSP-2183 Processors

Mode D [1]	Mode C	Mode B	Mode A	Booting Method
X	0	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Full Memory Mode.[2]
X	0	1	0	No Automatic boot operations occur. Program execution starts at external memory location 0. Chip is configured in Full Memory Mode. BDMA can still be used but the processor does not automatically use or wait for these operations.

Table 9-4. Booting Methods for All Processors Except the ADSP-2181 and ADSP-2183 Processors (Cont'd)

Mode D [1]	Mode C	Mode B	Mode A	Booting Method
0	1	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Host Mode. $\overline{\text{IACK}}$ has active pulldown. **Note:** Requires additional hardware.
0	1	0	1	IDMA feature is used to load any internal memory as desired. Program execution is held off until internal program memory location 0 is written to. Chip is configured in Host Mode. $\overline{\text{IACK}}$ has active pulldown.[2]
1	1	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Host Mode; $\overline{\text{IACK}}$ requires external pull-down. **Note:** Requires additional hardware.
1	1	0	1	IDMA feature is used to load any internal memory, as desired. Program execution is held off until internal program memory location 0 is written to. Chip is configured in Host Mode. $\overline{\text{IACK}}$ requires external pulldown.

1 `Mode D` pin is not available on the ADSP-2184, ADSP-2184L, ADSP-2185, ADSP-2185L, ADSP-2186, or ADSP-2186L processors.

2 Considered as standard operating settings. Using these configurations allows for easier design and better memory management.

The ADSP-218x processors use a BDMA boot sequence after reset when the BMODE and MMAP pins equal 0 (ADSP-2181 and ADSP-2183 processors) or the Mode B pin equals 0 (all other ADSP-218x processors).

The BDMA port is initialized for booting as follows:

- BWCOUNT is set to 32
- BDIR, BMPAGE, BEAD, BIAD, and BTYPE are set to zero
- BCR is set to 1
- BMWAIT is set to 15 for the ADSP-218x M and N series processors and 7 for all other ADSP-218x processors
- BMOVLAY is set to 0 for the ADSP-2187, ADSP-2188, and ADSP-2189 processors

These initializations configure the BDMA port to load 32 Program Memory words (96 bytes) (as specified by the BWCOUNT register) from Byte Memory Page zero (as specified in the BMPAGE field of the BDMA Control Register) and Byte Memory Address zero (as specified in the BEAD register) to internal Program Memory address zero (as specified in the BIAD register), using 24-bit Program Memory Word Format (as specified in the BTYPE field of the BDMA Control register).

When set to 1, the BDMA context reset bit (BCR) inhibits program execution during BDMA transfer and causes execution to begin at address PM(0x0000) after the transfer is completed. For the ADSP-218x M and N series processors, the number of wait states (BMWAIT bits [15:12] of the Composite Select Control register) for BDMA accesses is set to the maximum of 15. For all other ADSP-218x processors, the number of wait states (BMWAIT bits [14:12] of the Composite Select Control register) for BDMA accesses is set to the maximum value of 7. After the boot sequence is complete (32 words transferred), program execution begins at internal PM address 0x0000.

The ADSP-218x PROM Splitter utility provides a boot loader option for ADSP-218x processor-based designs. See "Development Software Features for BDMA Booting" on page 9-20 for information.

If you are developing your own boot-loading software for ADSP-218x processors, however, you should note that the BDMA Context Reset bit (BCR) is set to 1 (inhibiting program execution during BDMA transfer) and a BDMA interrupt is pending (signalling the first 32 word were sent) after the boot sequence is complete. Your program will have to process the interrupt (if you unmask the BDMA interrupt with the IMASK register) or clear the interrupt (with the IFC register).

In an alternate method, using the BDMA interrupt without context clear, a loader program could suspend program execution with the IDLE instruction while BDMA boot loading. If the loader sets the PM boot-load parameters, the loader enables only the BDMA interrupt in the IMASK register, and then executes an IDLE instruction. The IDLE instruction suspends program execution until the BDMA interrupt occurs. At that point all of program memory is loaded.

Development Software Features for BDMA Booting

The ADSP-218x PROM Splitter utility lets you create BDMA boot-loader programs for ADSP-218x processor-based designs. This provides a low overhead method for BDMA boot-loading your program. The boot loader program attaches memory loader code to the beginning of your executable program. The PROM Splitter generates loader code that initializes up to 6 pages of program memory and 4 pages of data memory, where each page is 16 K bytes in size. Typically, the code generated by the PROM Splitter is burned into an EPROM and used as the ADSP-218x's Byte Memory space.

When the `BMODE` and `MMAP` pins equal 0 (ADSP-2181 and ADSP-2183 processors) or the `Mode B` pin equals 0 (all other ADSP-218x processors), the ADSP-218x processors loads the first 32 program memory words from the Byte memory space and then begins execution. The loader routine is in those first 32 words; it continues to load from the BDMA port until your whole program is loaded.

For complete information on the PROM Splitter features, see the *Linker & Utilities Manual for ADSP-218x & ADSP-219x Family DSPs* and the software release notes.

IDMA Port

The IDMA port is a separate port on the ADSP-2181 and ADSP-2183 processors and a configured port on all other ADSP-218x processors when they are in Host Mode (`Mode C` pin equals 1). It is a 16-bit parallel slave I/O port that allows the processor's internal memory to be read or written by a host system. Figure 9-9 shows the ADSP-218x interface to the IDMA port.

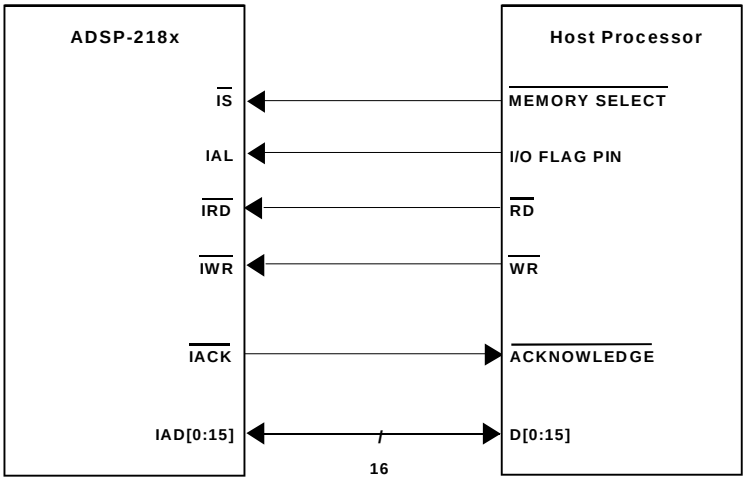


Figure 9-9. ADSP-218x Processor IDMA Port Interface

The IDMA port is a gateway to all internal memory locations on the DSP (except for the processor's memory-mapped control registers). The IDMA port is made up of 16 multiplexed data/address pins and 5 control pins. It provides transparent, direct access to the DSP's on-chip program and data RAM. IDMA port read/write access is completely asynchronous and a host can access the DSP's internal memory while the ADSP-218x processor is operating at full speed.

The IDMA port does not require any ADSP-218x processor intervention to maintain data flow. The host system can access the ADSP-218x processor's internal memory directly, without going through a set of mailbox registers. Direct access to DSP memory increases throughput for block data transfers. Through the IDMA port, internal memory accesses can be performed with an overhead of one DSP processor cycle per word.

The ADSP-218x processor supports boot loading through the IDMA port, through the BDMA port, or from an external Program Memory Overlay. The `MMAP` and `BMODE` pins (ADSP-2181 and ADSP-2183 processors) or `Mode B` pin (all other processors) select the DSP's boot mode and memory map. Setting `BMODE=1` and `MMAP=0` (ADSP-2181 and ADSP-2183 processors) or `Mode A=1` and `Mode C=1` (all other ADSP-218x processors) directs the ADSP-218x to boot through the IDMA port. For information on IDMA booting, see "Boot Loading through the IDMA Port" on page 9-46.

The IDMA port cannot be used to read or write the ADSP-218x's memory-mapped control registers. For more information, see "Modifying Control Registers for IDMA" on page 9-31.

IDMA Port Pin Summary

Table 9-5 identifies and describes the IDMA port pins.

Table 9-5. IDMA Port Pins

Pin Name(s)	Input/ Output	Function
$\overline{\text{IRD}}$	I	IDMA Port Read Strobe
$\overline{\text{IWR}}$	I	IDMA Port Write Strobe
$\overline{\text{IS}}$	I	IDMA Port Select
IAL	I	IDMA Port Address Latch Enable

Table 9-5. IDMA Port Pins (Cont'd)

Pin Name(s)	Input/ Output	Function
IAD0-15	I/O	IDMA Port Address/Data Bus
$\overline{\text{IACK}}$	O	IDMA Port Access Ready Acknowledge[1]

1 After reset, $\overline{\text{IACK}}$ is asserted (low). It stays low until an IDMA transfer is initiated. After each IDMA operation is completed, $\overline{\text{IACK}}$ is again low.

Four input signals control the IDMA port. Table 9-6 identifies and describes these signals.

Table 9-6. IDMA Port Input Signals

Input Signal	Description
IDMA Port Select ($\overline{\text{IS}}$)	This signal acts as a chip select for all IDMA operations.
IDMA Read ($\overline{\text{IRD}}$)	When both the $\overline{\text{IS}}$ and $\overline{\text{IRD}}$ signals are active (low), an IDMA read cycle begins.
IDMA Write ($\overline{\text{IWR}}$)	When both the $\overline{\text{IS}}$ and $\overline{\text{IWR}}$ signals are active (low), an IDMA write cycle begins.
IDMA Address Latch (IAL)	When the host wishes to initiate an Address Latch Sequence, this signal is asserted (active high). When the $\overline{\text{IS}}$ and IAL signals are both active, an IDMA Address Latch Sequence begins. At this point the host processor should drive the starting address of the IDMA transfer on the IAD bus.

An IDMA access ends when any one of the input signals goes inactive (high).

Asserting the IDMA Port Select ($\overline{IS}$) and address latch enable (IAL) directs the ADSP-218x processor to write the address on the IAD0-15 bus into the IDMA Control register. This register, shown in Figure 9-10, is memory-mapped at address DM(0x3FE0). Note that the latched address (IDMAA) cannot be read back by the host.

Because the IDMA Control register is a memory mapped register, the address information can be written by either the host processor or the DSP itself. This allows more flexibility in your system design.

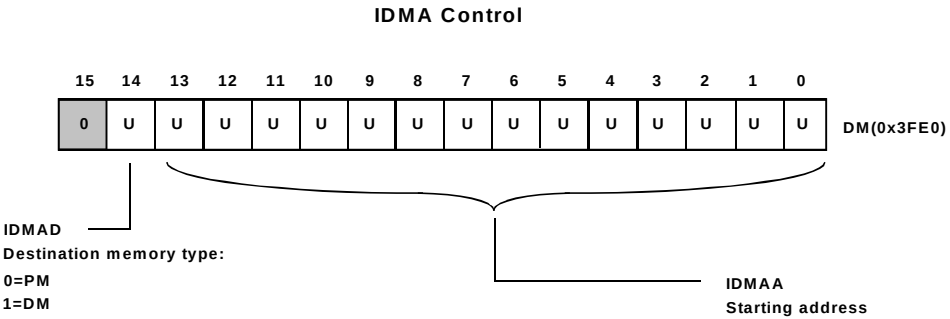


Figure 9-10. IDMA Control Register

Because the ADSP-2187, ADSP-2188, and ADSP-2189 processors have additional on-chip overlay regions for Program Memory and Data Memory, these processors contain an IDMA Overlay register that allows either the host or the DSP core to configure the specific overlay memory region to perform a DMA access. See Figure 9-13 on page 9-28. The ADSP-218x processors specified above can access this register at address 0x3FE7 in Data Memory. The host processor can access this register by performing an IDMA address latch cycle.

If bit 15 is set to 1 when performing an IDMA Address Latch Cycle, the data written on the IAD bus from the host will be written to the IDMA Overlay register. If bit 15 is set to zero when performing the Address Latch Cycle, the data written on the IAD bus from the host will be the IDMA starting address, which is written into the IDMAA bit field of the IDMA Control register.

Bits 3 through 0 of the IDMA Overlay register specify the PM Overlay page of the IDMA transfer. Program Memory Overlays are accessed via the IDMA port when the IDMA access is within the address range PM 0x2000 through PM 0x3fff. Bits 7 through 4 of the IDMA Overlay register specify the DM Overlay page of the IDMA transfer. Data Memory Overlays are accessed via the IDMA port when the IDMA access is within the address range DM 0x0000 through DM 0x1fff. These bit fields only apply to the ADSP-2187, ADSP-2188, and ADSP-2189 processors, due to their additional on-chip overlay memory regions. For all other ADSP-218x processors these bit fields do not apply and must be set to 0.

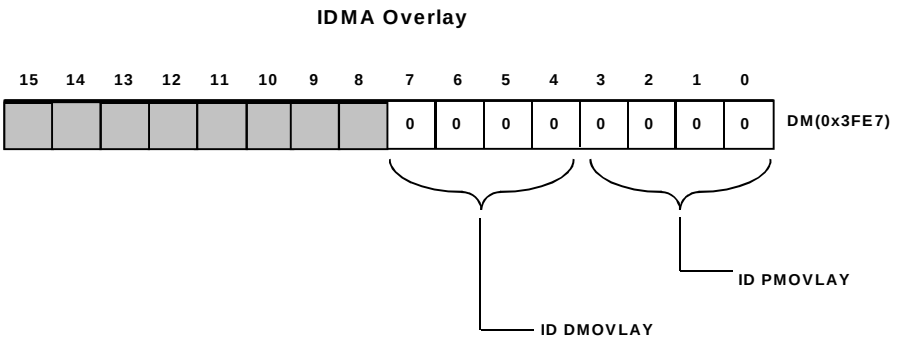
When accessing the internal Program Memory and Data Memory Overlay regions of the ADSP-2187, ADSP-2188, and ADSP-2189 processors via the IDMA port, you must specify the target overlay region in the IDMA Overlay register *prior to* writing the target address to the 14-bit IDMAA starting address (bits [13:0]) of the IDMA Control register.

Please note the following:

- Core accesses to or from DM(0x3FE7) have bit 15 always cleared
- The IDMA port cannot access off-chip overlay pages directly
- There is no interaction between the IDMA OVLAY register and the core registers, PMOVLAY and DMOVLAY

Through the IDMAA register, the DSP can also specify the starting address and data format for DMA operation. Asserting the IDMA port select ($\overline{IS}$) and address latch enable (IAL) directs the DSP to write the address onto the IAD0-14 bus into the IDMA Control register. The address value is written on the bus by the host; then, the address information is written to or latched into the IDMA Address register (IDMAA). If bit 15 is set to 0, IDMA latches the address. If bit 15 is set to 1, IDMA latches into the OVLAY register.

The IDMA Address register, shown in Figure 9-11, is memory mapped at address DM (0x3FE0). Note that the latched address (IDMAA) cannot be read back by the host.



Note: The ID DMOVLAY and ID PMOVLAY bit fields apply only to the ADSP-2187, ADSP-2188, and ADSP-2189 processors. For all other ADSP-218x processors, these bits are unused and must be set to 0.

Figure 9-11. IDMA Overlay Register

Asserting the IDMA Port Select ($\overline{\text{IS}}$) and Read strobe ($\overline{\text{IRD}}$) inputs directs the ADSP-218x to output the contents of the memory location pointed to by the IDMA Control register onto the IDMA data bus.

Asserting the IDMA Port Select ($\overline{\text{IS}}$) and Write strobe ($\overline{\text{IWR}}$) inputs directs the ADSP-218x to write the input from the IDMA data bus to the address pointed to by the IDMA register.

When reading or writing to Data Memory, the IDMA data bus pins make up a 16-bit Data Memory word. When reading or writing to Program Memory, the upper 16 bits of the 24-bit Program Memory word are sent first on the IDMA data bus pins. On the next IDMA port read or write, the lowest 8 bits of the Program Memory word are sent on bits 0-7 of the IDMA data bus. For reads, the ADSP-218x processor sets data bus lines 8-15 to 0; for writes, the ADSP-218x processor ignores bits 8-15 from the host.

The IDMA Port Access Acknowledge ($\overline{\text{IACK}}$) line identifies completion of data read/write operations. It also acts as a busy signal for the IDMA port. External devices must wait for this signal to go low before modifying the IDMA Control register or starting the next read or write operation.

DMA Port Functional Description

The IDMA port lets a host system directly access internal ADSP-218x memory locations (but *not* the memory-mapped control registers). Figure 9-11 shows a flow chart of the most general case for IDMA transfers.

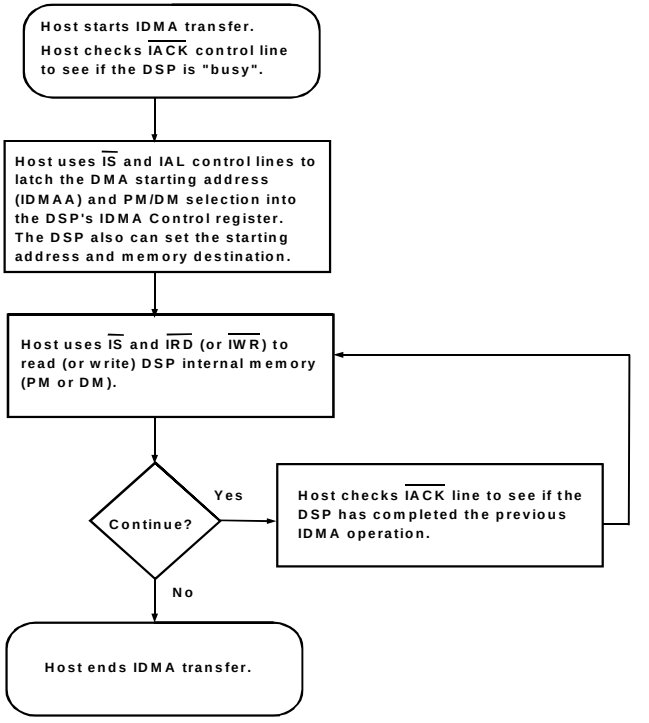


Figure 9-13. General IDMA Transfer Flow Chart

In the case shown in Figure 9-12, the host system starts an IDMA transfer by checking the state of the $\overline{\texttt{IACK}}$ line to determine port status (ready/busy). When the IDMA port is ready (when the $\overline{\texttt{IACK}}$ signal is low), the host directs the ADSP-218x (with the $\overline{\texttt{IS}}$ and IAL lines) to latch the IDMA internal memory address from the IDMA address/data bus to the IDMA Control register. Note that the latched address cannot be read back by the host.

Next, the host (using the $\overline{\texttt{IS}}$ and $\overline{\texttt{IRD}}$ or $\overline{\texttt{IS}}$ and $\overline{\texttt{IWR}}$ lines) begins reading (or writing) the DSP's internal memory until done. With each IDMA read or write operation, the ADSP-218x automatically increments the IDMA internal memory address. Note that the ADSP-218x continues program execution throughout the IDMA transfer operation, *except* during the "stolen" cycle used to do the memory access.

The IDMAA starting address field of the IDMA Control register wraps around when incrementing from addresses PM 0x3fff and DM 0x3fff. In other words, for the next IDMA access, the value for the IDMAA address field will point to address location PM 0x0000 or DM 0x0000.

The case shown in Figure 9-12 is not the only way to use the IDMA port. Some variations on this scheme include the following:

- After completing an IDMA port read/write operation, the host could change the IDMA internal memory address and start a new operation from a different starting address.
- After latching an IDMA internal memory address, the host could stop the operation and come back at a later time to proceed with the read/write operation. The IDMA starting memory address remains in the IDMA Control register until the host or DSP changes it.
- The ADSP-218x processor can also read and write the IDMA Control register as part of your program. This means that the host could just control read/write operations and let the ADSP-218x processor control the IDMA starting memory address.

- Using the IDMA short read cycle (which does not wait for the data-ready assertion of the $\overline{\text{IACK}}$ signal), you could set up a single-location data buffer for IDMA read transfers. For information on how this data buffer would work, see "Short Read Cycle" on page 9-37.
- For ADSP-218x applications with a host processor or host ASIC that does *not* use a data-ready or write-complete acknowledge, use the IDMA short read/write cycles.

There are some restrictions on IDMA operations. These hardware/software design restrictions include the following:

- If your design has both the host and ADSP-218x processors writing to the IDMA Control register, do *not* let both write to this register at the same time; the results of this are indeterminate.
- Host reads of internal Program Memory take two IDMA reads (for a 24-bit word through a 16-bit port). If an IDMA address latch cycle or an ADSP-218x processor write to the IDMA Control register occurs after the first Program Memory read cycle, the IDMA port "loses" the second half of the 24-bit Program Memory word. The next IDMA read or write uses the address selected by the new contents of the IDMA Control register. Note that writing to the IDMA Control register after the first half of a Program Memory IDMA read lets you read just 16-bit data from Program Memory.
- Host writes to internal Program Memory take two IDMA writes (for a 24-bit word through a 16-bit port). If an IDMA address latch cycle or a ADSP-218x write to the IDMA Control register occurs after a first Program Memory write cycle, the IDMA port "loses" the Program Memory word without changing the contents of memory. The next IDMA read or write accesses the address selected by the new contents of the IDMA Control register.

- Host memory accesses through the IDMA port that occur while the ADSP-218x processor is in powerdown have some restrictions. See Chapter 7, "System Interface" for information on powerdown restrictions on IDMA port transfers.

Modifying Control Registers for IDMA

The ADSP-218x's memory-mapped control registers are protected from DMA transfers to prevent accidental corruption. You may want the host processor to read and write these registers, however, in order to determine the ADSP-218x's configuration and then change it.

To read the memory-mapped control registers, you must first transfer the contents of these locations to another area of internal RAM. Listing 9-1 shows a loop that performs this task:

Listing 9-1. Loop to Transfer Memory-Mapped Control Register Contents

```
#define NUM_REG 32

.section/dm    Data_Memory;
.var           temp_array[NUM_REG];

.section/pm    Program_Code;
   i0 = temp_array;        /* i0 points to 1st location of
                              buffer */
   l0 = 0;                 /* length of zero means
                              non-circular buffer */
   i1 = 0x3fe0;            /* i1 points to 1st memory mapped
                              control register */
   l1 = 0;                 /* length of zero means
                              non-circular buffer */
   m1= 1;                  /* modify DAG registers by one
                              after each access */
   cntr = NUM_REG;         /* counter equals number of
                              elements in buffer to be
                              swapped */
```

```
    do transfer until ce;    /* loop body begins at the next
                                instruction */
    ax0 = dm(i0,m1);         /* read from buffer written to by
                                host via IDMA */
transfer:dm(i1,m1) = ax0;    /* transfer buffer values to
                                memory-mapped control
                                registers */
```

To have the host write to the memory-mapped control registers, you must first load the values to a temporary buffer (through the IDMA port) and then signal the ADSP-218x processor to transfer the contents of the temporary buffer to the memory-mapped control registers. This transfer is performed in a manner similar to that shown in Listing 9-1. You should set up some form of signalling between the host and the ADSP-218x processor: interrupts, flag I/O, or a mailbox register. This signalling provides a mechanism for the host to tell the DSP when to perform an operation and vice versa.

IDMA Timing

From the host system interface point of view, there are four IDMA port operations with critical timing parameters. These operations are:

- Latching the IDMA internal memory address
- Latching the IDMA Overlay pages (ADSP-2187, ADSP-2188, ADSP-2189 processors only)
- Reading from the IDMA port
- Writing to the IDMA port

The following sections cover the timing details for each of these operations.

Address Latch Cycle

The host writes the DMA starting address and destination memory type (DM or PM) using the IDMA address latch cycle. The address latch cycle, shown in Figure 9-14, consists of the following steps:

1. Host ensures that $\overline{\text{IACK}}$ line is low.
2. Host asserts IAL and $\overline{\text{IS}}$, directing the ADSP-218x processor to latch the IDMA starting address from the IAD15-0 address/data bus into the IDMA Control register.
3. Host drives the starting address (bits 13-0) and destination memory type (bit 14) onto the IAD15-0 bus. Bit 15 must be a 0.

The $\overline{\text{IRD}}$ and $\overline{\text{IWR}}$ remain high (inactive) throughout the latch operation.

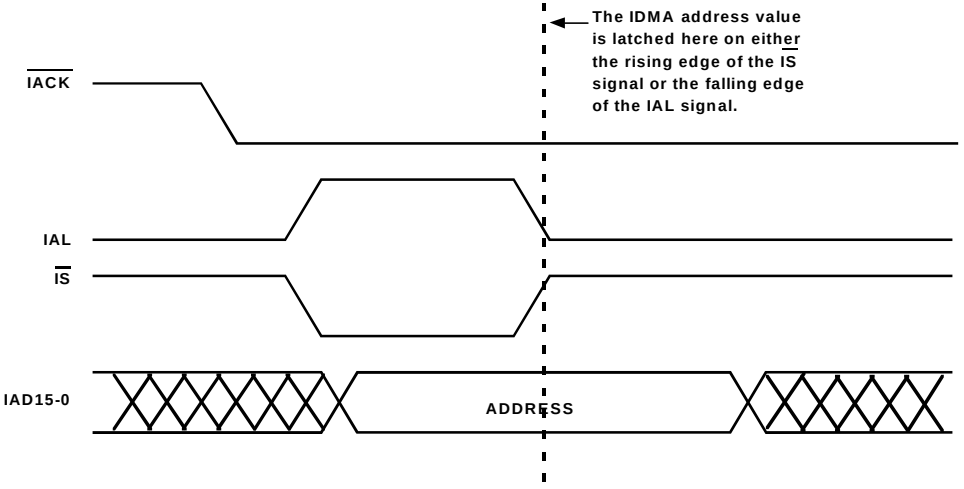


Figure 9-14. IDMA Address Latch or Overlay Latch Cycle

The IDMA starting address and destination memory type is available to the host and to the ADSP-218x processor in the IDMA Control register. For Data Memory accesses, the ADSP-218x processor increments the address automatically after each IDMA read or write transfer (16-bit word). For Program Memory accesses, the ADSP-218x processor increments the address automatically after each pair of IDMA read or write transfers (24-bit word).

Both the ADSP-218x processor and the host can specify the starting address by writing to the IDMA Control register. Do not let the ADSP-218x processor access the IDMA Control register while it is being written by the host; this operation will have an indeterminate result.

Overlay Latch Cycle

The Overlay latch cycle applies only to the ADSP-2187L and all M and N series processors. The host writes the DMA starting address and destination memory type (DM or PM) using the IDMA address latch cycle. The overlay latch cycle, shown in Figure 9-14 on page 9-33, consists of the following steps:

1. Host ensures that $\overline{\texttt{IACK}}$ line is low.
2. Host asserts IAL and $\overline{\texttt{IS}}$, directing the ADSP-218x processor to latch the IDMA Overlay pages from the IAD15-0 address/data bus into the IDMA Overlay register.
3. The host drives a 1 on bit 15, the DMOVLAY value on bits 7:4, and the PMOVLAY value on bits 3:0.

The $\overline{\texttt{IRD}}$ and $\overline{\texttt{IWR}}$ remain high (inactive) throughout the latch operation.

Long Read Cycle

An IDMA long read cycle can be performed if your host processor uses a data-ready acknowledge signal to notify the host when to latch data on the IAD bus or if the host is configured to wait for the worst case data delay. A long read cycle could be used by a Motorola 68322 processor for example, where the $\overline{\text{IACK}}$ signal of the ADSP-218x processor would be connected to the $\overline{\text{DTACK}}$ signal of the Motorola 68322.

The host reads the contents of an ADSP-218x processor internal memory location using the IDMA port long read cycle. The read cycle, shown in Figure 9-15, consists of the following steps:

1. Host ensures that $\overline{\text{IACK}}$ line is low.
2. Host asserts $\overline{\text{IRD}}$ and $\overline{\text{IS}}$ (low), causing the ADSP-218x processor to put the contents of the location pointed to by the IDMA address on the IAD15-0 address/data bus.
3. ADSP-218x processor deasserts $\overline{\text{IACK}}$ line, indicating the requested data is being fetched. When the ADSP-218x processor asserts the $\overline{\text{IACK}}$ line, the requested data is driven on the IAD address/data bus.
4. Host detects the $\overline{\text{IACK}}$ line is now low and reads the data (Read Data) from the IAD15-0 address/data bus. (Alternately, the host can just wait a fixed worst case delay, which also guarantees the $\overline{\text{IACK}}$ signal is low again. After reading the data, the host deasserts $\overline{\text{IRD}}$ and $\overline{\text{IS}}$.

The IAL is low (inactive) and $\overline{\text{IWR}}$ is high (inactive) throughout the read operation.

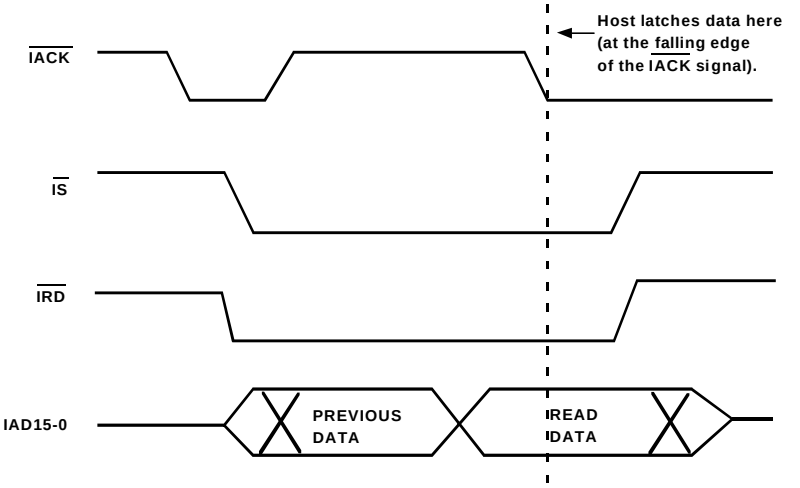


Figure 9-15. IDMA Long Read Cycle

IDMA memory accesses "steal" one processor cycle, but they may only occur on instruction cycle boundaries. The best-case response for a 16-bit Data Memory read or the first 16 bits of a Program Memory read is 2.5 processor cycles; the worst case response is 3.5 cycles. One cycle is for synchronization, one is for reading the memory internally, and one-half cycle is for $\overline{\text{IACK}}$ setup time.

A second cycle of synchronization may be required. Thus the best-case and worst-case response times are determined as follows:

Best Case: 1 cycle (sync) + 1 cycle (internal memory read) + 0.5 cycle ($\overline{\text{IACK}}$ setup) = 2.5 cycles

Worst Case: 1 cycle (sync) + 1 cycle (sync) + 1 cycle (internal memory read) + 0.5 cycle ($\overline{\text{IACK}}$ setup) = 3.5 cycles

In the case of a Program Memory operation, the second IDMA port read cycle for a given internal 24-bit word does not require an internal memory access, does not wait for an instruction cycle boundary, and takes 1.5 or 2.5 cycles.

The best- and worst-case response times given above assume no system hold offs. Hold offs for DMA transfers are defined in the section "DMA Cycle Stealing, Hold Offs, and IACK Acknowledge" on page 9-47.

If an IDMA address latch cycle or an ADSP-218x processor write to the IDMA Control register occurs after a first Program Memory read cycle (16 bits), the IDMA port will lose the second half of the Program Memory word. The ADSP-218x processor treats the next IDMA access as the first operation for the new IDMA address and destination.

Short Read Cycle

Figure 9-16 on page 9-38 shows the host reading the contents of an ADSP-218x processor's internal memory location using the IDMA short read cycle.The short read cycle can be used to allow the host to operate more quickly by not waiting for the internal access to complete. This method can be used if the host can do read accesses shorter than t_{IRDH1} and t_{IRDH2} and longer than t_{IRP1}.

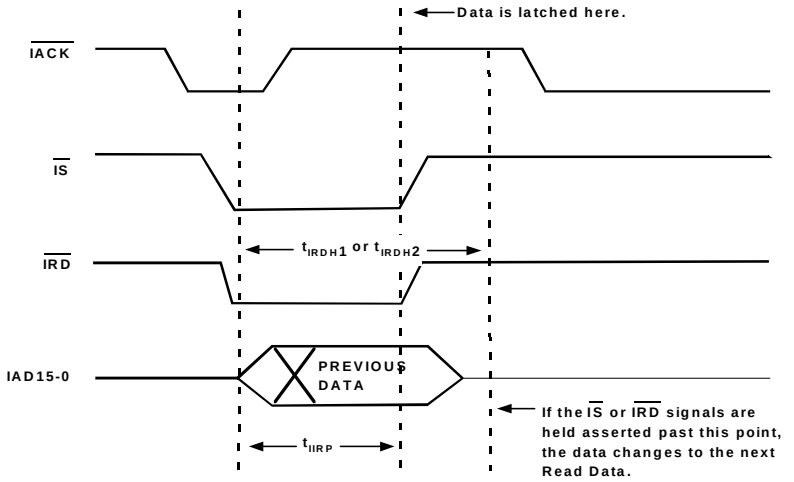


Figure 9-16. IDMA Short Read Cycle

The read cycle consists of the following steps:

1. Host ensures that $\overline{\text{IACK}}$ line is low.
2. Host asserts $\overline{\text{IRD}}$ and $\overline{\text{IS}}$ (low), directing the ADSP-218x processor to put the contents of the location pointed to by the target IDMA address on the IAD15-0 address/data bus.
3. ADSP-218x processor deasserts $\overline{\text{IACK}}$ line, indicating the requested data is being fetched.
4. The host asserts the $\overline{\text{IS}}$ and $\overline{\text{IRD}}$ signals for a minimum amount of time, adhering to the t_{IRP} timing specification. The host then latches the Previous Data and deasserts the $\overline{\text{IS}}$ and $\overline{\text{IRD}}$ signals prior to t_{IRDH1}or t_{IRDH2}. (See the subsection entitled “IDMA Read, Short Read Cycle Timing” in the “Timing Parameters” section of the appropriate ADSP-218x processor data sheet.)

The host ignores the falling edge of the $\overline{\text{IACK}}$ signal, since the data is latched by the host on the rising edge of the $\overline{\text{IS}}$ or $\overline{\text{IRD}}$ signal not on the falling edge of the $\overline{\text{IACK}}$ signal.

The host must perform an initial "dummy read," since the first short read access reads in the Read Data from the IAD bus. The next short read access reads in the first correct data word, Previous Data, on the IAD bus. The advantage of using short read accesses versus long read accesses is that short reads allow for shorter block transfer times.

IAL is low (inactive) and $\overline{\text{IWR}}$ is high (inactive) throughout the read operation.

The IDMA short read and long read cycles provide different alternatives for implementing your DMA transfers. Short reads are useful for hosts that can handle the faster timing of these accesses, while long reads allow slower hosts more time.

The IDMA short read cycle also serves as a single-location data buffer. If you are using the ADSP-218x processor in a multiprocessing environment, using this buffer is one way to avoid tying up the IAD bus (waiting for the $\overline{\text{IACK}}$ signal).

If an IDMA address latch cycle or a ADSP-218x processor write to the IDMA Control register occurs after a first Program Memory read cycle, the IDMA port will lose the second half of the Program Memory word. The ADSP-218x processor treats the next host data on the IAD address/data bus as the new contents of the IDMA Control register.

IDMA Read—Short Read Only Mode

A new IDMA read mode cycle, Short Read Only Mode, has been added for the ADSP-218x M and N series processors. Because these processors are running at an increased clock rate (up to 75 MHz maximum for the M series processors and up to 80 MHz maximum for the N series processors), this increase in clock speeds has made the timing window for a host processor more critical when performing IDMA short read accesses.

To alleviate this timing constraint, the ADSP-218x M and N series processors provide support for an enhancement to the IDMA read cycle. The Short Read Cycle in Short Read Only Mode allows a host processor to read in only the Previous Data. This allows for longer timing duration on the $\overline{\text{IRD}}$ and $\overline{\text{IS}}$ signals by removing the maximum $t_{IRDH1}$and t_{IRDH2} timing constraints. This increase in timing duration relieves the processor from complying with the tighter timing restrictions of the DSP. As shown in Figure 9-17, the $\overline{\text{IRD}}$ and $\overline{\text{IS}}$ signals can be held active indefinitely, allowing for the host processor to always latch in the Previous Data.

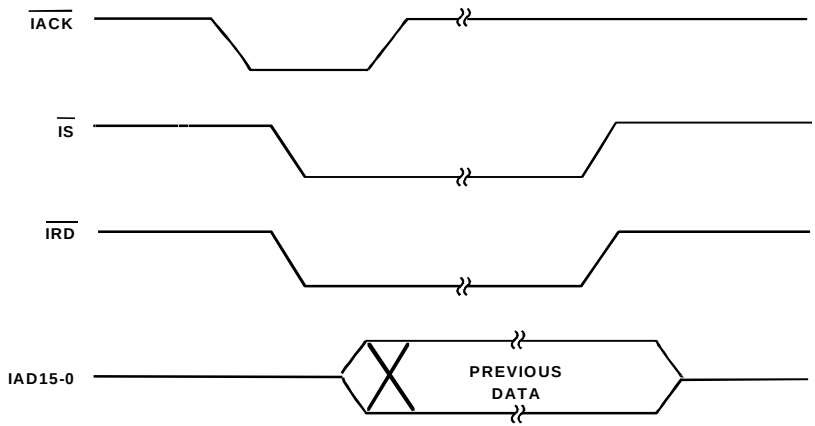


Figure 9-17. IDMA Short Read Cycle in Short Read Only Mode Timing

The Short Read Only mode can be enabled or disabled by setting or clearing bit 14 in the IDMA Overlay register. The default value for this bit (as well as the remaining bits in this control register) are shown in Figure 9-18.

This bit applies to the ADSP-218x M and N series processors only. For all other ADSP-218x processors, this bit is unused and should be set to 0. Also, all bits shown in grey are reserved and must be set to 0.

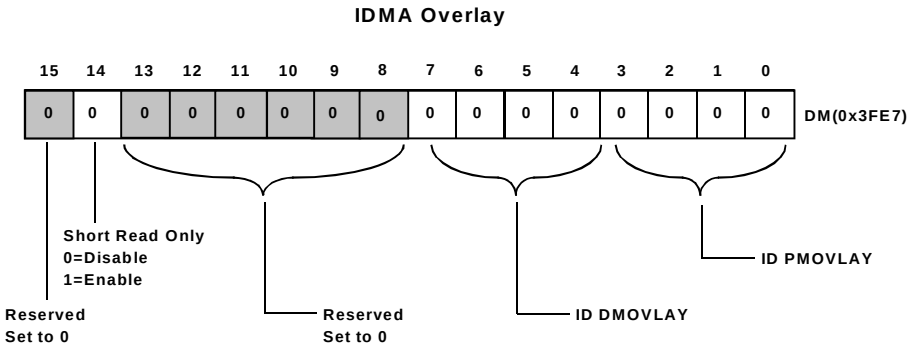


Figure 9-18. IDMA Overlay Register (Short Read Only Mode)

Long Write Cycle

As with IDMA long read cycles, an IDMA long write cycle can be performed if your host processor uses a data-ready acknowledge signal to notify the host when to stop driving data on the IAD bus. A long write cycle could be used by a Motorola 68322 processor for example, where the $\overline{\text{IACK}}$ signal of the ADSP-218x processor would be connected to the $\overline{\text{DTACK}}$ signal of the Motorola 68322.

The host writes the contents of an internal memory location using the IDMA long write cycle. The write cycle, shown in Figure 9-19, consists of the following steps:

1. Host ensures that $\overline{\text{IACK}}$ line is low.
2. Host asserts $\overline{\text{IWR}}$ and $\overline{\text{IS}}$ (low), directing the ADSP-218x processor to write the data on the IAD15-0 address/data bus to the location pointed to by the target IDMA address.
3. ADSP-218x processor deasserts the $\overline{\text{IACK}}$ line, indicating it recognizes the IDMA write operation.
4. Host drives the data on the IAD address/data bus.
5. ADSP-218x processor asserts $\overline{\text{IACK}}$ line, indicating it latched the data on the IAD15-0 address/data bus.
6. Host recognizes the $\overline{\text{IACK}}$ line is now low, stops driving the data on the IDMA address/data bus and deasserts $\overline{\text{IWR}}$ and $\overline{\text{IS}}$ (ending the IDMA long write cycle).

Note that IAL is low (inactive) and $\overline{\text{IRD}}$ is high (inactive) throughout the write operation.

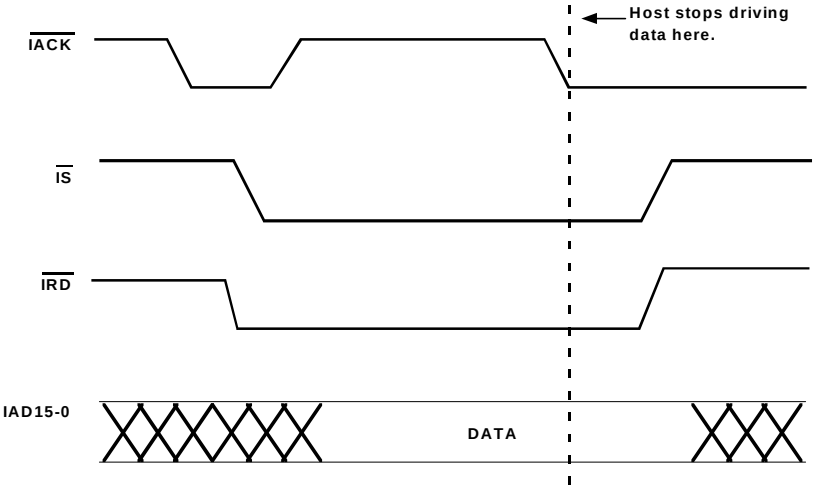


Figure 9-19. IDMA LongWrite Cycle

IDMA port writes to Program Memory require two IDMA port write cycles to write a word to ADSP-218x processor internal Program Memory. The ADSP-218x processor acknowledges the IDMA port write of the first 16 bits (MSBs of PM word) as they are written to a temporary holding latch, not waiting for an instruction cycle boundary. The ADSP-218x processor does not assert the $\overline{\text{IACK}}$ line after the second Program Memory write (or all Data Memory writes) until the internal memory write is complete and the IDMA port is ready for another transaction.

Host IDMA write accesses to internal Program Memory take two IDMA port writes (24-bit word through a 16-bit port). If an IDMA address latch cycle or a ADSP-218x processor write to the IDMA Control register occurs after a first program memory write cycle, the IDMA port "loses" the Program Memory word without changing the contents of ADSP-218x processor internal memory. The next IDMA read or write uses the address selected by the new contents of the IDMA Control register.

Short Write Cycle

An IDMA short write cycle should be performed if your host processor does not use a data-ready acknowledge signal to signify the completion of a write access.

The host writes the contents of a ADSP-218x processor internal memory location using the IDMA short write cycle. The write cycle, shown in Figure 9-20, consists of the following steps:

1. Host ensures that $\overline{\texttt{IACK}}$ line is low.
2. Host asserts $\overline{\texttt{IWR}}$ and $\overline{\texttt{IS}}$ (low), directing the ADSP-218x processor to write the data on the IAD15-0 address/data bus to the location pointed to by the target IDMA address.
3. ADSP-218x processor deasserts $\overline{\texttt{IACK}}$ line (high), indicating it recognizes the IDMA write operation.
4. Host drives the data on the IAD address/data bus.
5. Host deasserts $\overline{\texttt{IWR}}$ and $\overline{\texttt{IS}}$ *after* meeting the short write timing requirements (ending the short write cycle).
6. ADSP-218x processor detects $\overline{\texttt{IWR}}$ and $\overline{\texttt{IS}}$ have gone high, then latches the data on the IAD address/data bus.

7. Host stops driving the data on the IAD15-0 address/data bus after meeting the short write timing requirements. (See the t_{IRP} timing specification in the subsection entitled "IDMA Read, Short Write Cycle" in the "Timing Parameters" section of the appropriate data sheet.)

IAL is low (inactive) and $\overline{\text{IRD}}$ is high (inactive) throughout the write operation.

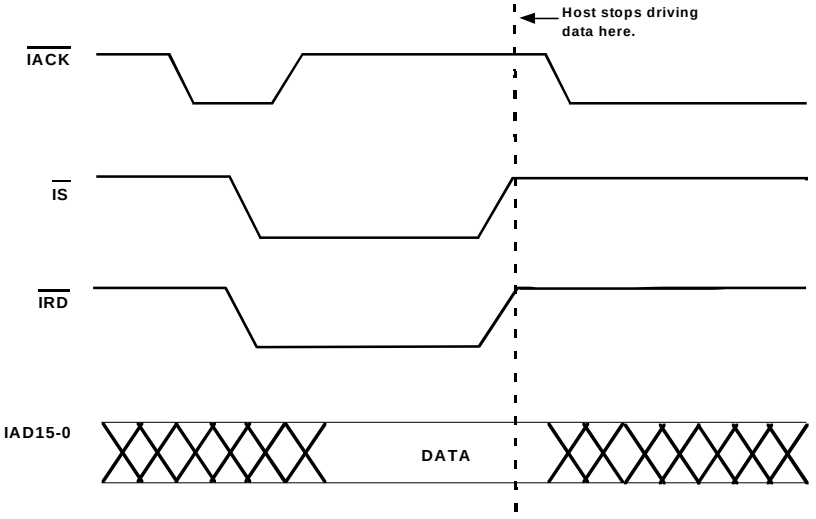


Figure 9-20. IDMA Short Write Cycle

IDMA port writes to Program Memory require two IDMA port write cycles to write a word to ADSP-218x processor internal Program Memory. The ADSP-218x processor acknowledges the IDMA port write of the first 16 bits (MSBs of PM word) as they are written to a temporary holding latch.Writes to this holding latch are not done on an instruction cycle boundary. The ADSP-218x processor does not assert the $\overline{\text{IACK}}$ line (low) after the second Program Memory write (or all Data Memory writes) until the internal memory write is complete (performed on an instruction cycle boundary) and the IDMA port is ready for another transaction.

If an IDMA address latch cycle or a ADSP-218x processor write to the IDMA Control register occur after a first Program Memory write cycle, the IDMA port will lose the first half of the Program Memory word. The next Program Memory write will be considered the first half of a Program Memory write pair.

There are two features that differentiate the IDMA port long write from the IDMA short write. The long write supports hosts (processors or ASICs) that allow a data-written acknowledge. If your host needs the ADSP-218x processor to signal that it has written the data, use the IDMA long read cycle.

The short write lets your host hold data on the bus just until it is latched and then releases the bus. If you are using the ADSP-218x processor in a multiprocessing environment, using the short write is one way to avoid tying up the IAD15-0 data bus (waiting for the $\overline{\texttt{IACK}}$ signal). Short writes are also useful for hosts that can handle the short write timing but cannot extend the accesses with the $\overline{\texttt{IACK}}$ signal (when hold offs occur).

Boot Loading through the IDMA Port

The ADSP-218x processor supports boot loading through the IDMA port. To boot through the IDMA port, use the following steps:

- Reset the processor (assert $\overline{\texttt{RESET}}$).
- Set `MMAP=0` and `BMODE=1` (ADSP-2181 and ADSP-2183 processors) or `Mode A=1` and `Mode C=1` (all other ADSP-218x processors). These pin settings select IDMA booting.
- Deassert $\overline{\texttt{RESET}}$.

- Load ADSP-218x processor internal memory through the IDMA port. Program execution is held off until you write to Program Memory address zero, PM(0x0000). The ADSP-218x processor responds to IDMA control signals (IAL, $\overline{\text{IS}}$, $\overline{\text{IWR}}$, and $\overline{\text{IRD}}$) and provides acknowledge ($\overline{\text{IACK}}$) in the same manner as during non-booting IDMA transfers.
- Write to PM(0x0000) to begin program execution.

Make certain to load all of the necessary memory locations with the proper data before writing to PM 0x0000. When configured for an IDMA boot, the DSP core executes an IDLE instruction until PM 0x0000 is written to by the host via the IDMA port. Writing to this PM location forces the DSP to begin program execution from PM 0x0000.

DMA Cycle Stealing, Hold Offs, and IACK Acknowledge

ADSP-218x processors generate the $\overline{\text{IACK}}$ signal to notify the system that it is safe to read or write through the IDMA port. After reset, $\overline{\text{IACK}}$ is asserted (low). It stays low until an IDMA transfer is initiated. After each IDMA operation is completed, the $\overline{\text{IACK}}$ signal will again be low.

In order for $\overline{\text{IACK}}$ to be asserted (low) during the IDMA operation, the IDMA port must have completed the internal memory access by either writing data to memory or reading data from memory. The IDMA port must "steal" a processor cycle to do this. In order to steal a processor cycle, the IDMA port must wait for an instruction completion boundary. Thus if $\overline{\text{IACK}}$ is not asserted, it is not safe for the host to access the IDMA port.

In most cases, there is an instruction boundary on every clock cycle (CLKOUT period) and the IDMA port can complete its transfer in a given period of time. There are, however, some situations where either the ADSP-218x processor does not complete an instruction in one clock cycle or the IDMA port cannot access memory. These situations are called *DMA hold offs.* The following describes DMA hold-off situations:

- **Bus Request** — If the ADSP-218x processor is being held in Bus Request when it attempts an external access (DM Overlay, PM Overlay, or I/O memory space), or if it is not in Go mode, processor execution stops in the middle of the cycle and no instruction boundary is encountered. Therefore, the IDMA port cannot complete its internal memory access and $\overline{\text{IACK}}$ will be held off.
- **External Access with Wait State(s)** — If the ADSP-218x processor is performing a wait-stated external access (DM Overlay, PM Overlay, or I/O memory space), then the instruction cycle will not complete until the access has completed; the IDMA port cannot steal a cycle, and $\overline{\text{IACK}}$ will be held off.
- **Multiple External Accesses** — If the ADSP-218x processor is executing a multifunction instruction where more than one of the required elements (PM instruction fetch, PM data access, or DM data access) resides externally, it will require more than one cycle to complete the instruction and $\overline{\text{IACK}}$ will be held off. Likewise, if the ADSP-218x processor is executing an instruction from external PM that initiates an I/O memory space access, $\overline{\text{IACK}}$ will be held off until the cycle completes.
- IDLE n (**clock-reducing** IDLE **instruction**) — Because this instruction slows down the effective cycle time of the ADSP-218x processor, $\overline{\text{IACK}}$ may be delayed.

- **SPORT Autobuffering to External Memory with Wait Stated Access** — When one of the processor's serial ports needs to access external memory for autobuffering and the external access takes more than one cycle, the IDMA transfer will be held off.
- **EZ-ICE Emulation** — When the EZ-ICE emulator is controlling your ADSP-218x processor target system, IDMA transfers may be held off for periods of time.

Priority Chain

The ADSP-218x processor priority chain for concurrent requests occurring at instruction cycle boundaries is as follows:

1. Completion of an external memory access
2. IDMA internal memory transfers
3. BDMA internal memory transfers
4. SPORT autobuffer operations
5. Emulator interrupt
6. Emulator instruction
7. Powerdown interrupt
8. Unmasked interrupt
9. Normal instruction execution

Using the $\overline{\text{IACK}}$ signal simplifies your system design by allowing you to ignore hold-off conditions. If you always wait for $\overline{\text{IACK}}$ to assert before accessing the IDMA port, the DMA transfers will always operate properly.

You can ignore $\overline{\text{IACK}}$, however, if you are sure that no hold-offs occur in your system or if your IDMA accesses are longer than any hold-offs. To be sure of this, you must carefully analyze all possible hold-off conditions of your system.