

SECTION 9 PORT A

Port A is the memory expansion port that can be used for either memory expansion or for memory-mapped I/O (see 2.9.1 Expansion Port (Port A)). A number of features make port A versatile and easy to use. These features provide a low-parts-count connection with fast memories, slow memories/devices, and multiple bus master systems.

The port A data bus is 24 bits wide with a separate 16-bit address bus capable of a sustained rate of one memory access per machine cycle (using no-wait-state memory). External memory is divided into three 64K-word X 24-bit spaces – X:, Y:, and P:. An internal wait-state generator can be programmed to insert up to 15 wait states if access to slower memory or I/O devices is required. A bus wait signal allows an external device to control the number of wait states inserted in a bus access operation. Bus arbitration signals allow an external device (e.g., a DMA controller or another processor) use of the bus while internal operations continue using the internal memories. Two power-reduction features are specific to port A. The first power-reduction feature is that accessing the internal memory spaces does not toggle the external memory signals, eliminating unneeded switching current. The second power-reduction feature is that, if lower memory speed is acceptable, wait states can be added to external memory accesses to significantly reduce power while accessing those memories.

9.1 PORT A INTERFACE

One or more of the digital signal processor (DSP) memory sources (X data memory, Y data memory, and program memory) can be accessed during the execution of an instruction. Each of these memory sources may be either internal or external to the DSP. Three address buses (XAB, YAB, and PAB) and four data buses (XDB, YDB, PDB, and GDB) are available for internal memory accesses during one instruction cycle, but only one address bus and one data bus (port A) are available for external memory accesses. If all memory sources are internal to the DSP, one or more of the three memory sources may be accessed in one instruction cycle (i.e., program memory access or program memory access plus an X, Y, XY, or L memory reference). However, when one or more of the memories are external to the DSP56000/DSP56001, memory references may require

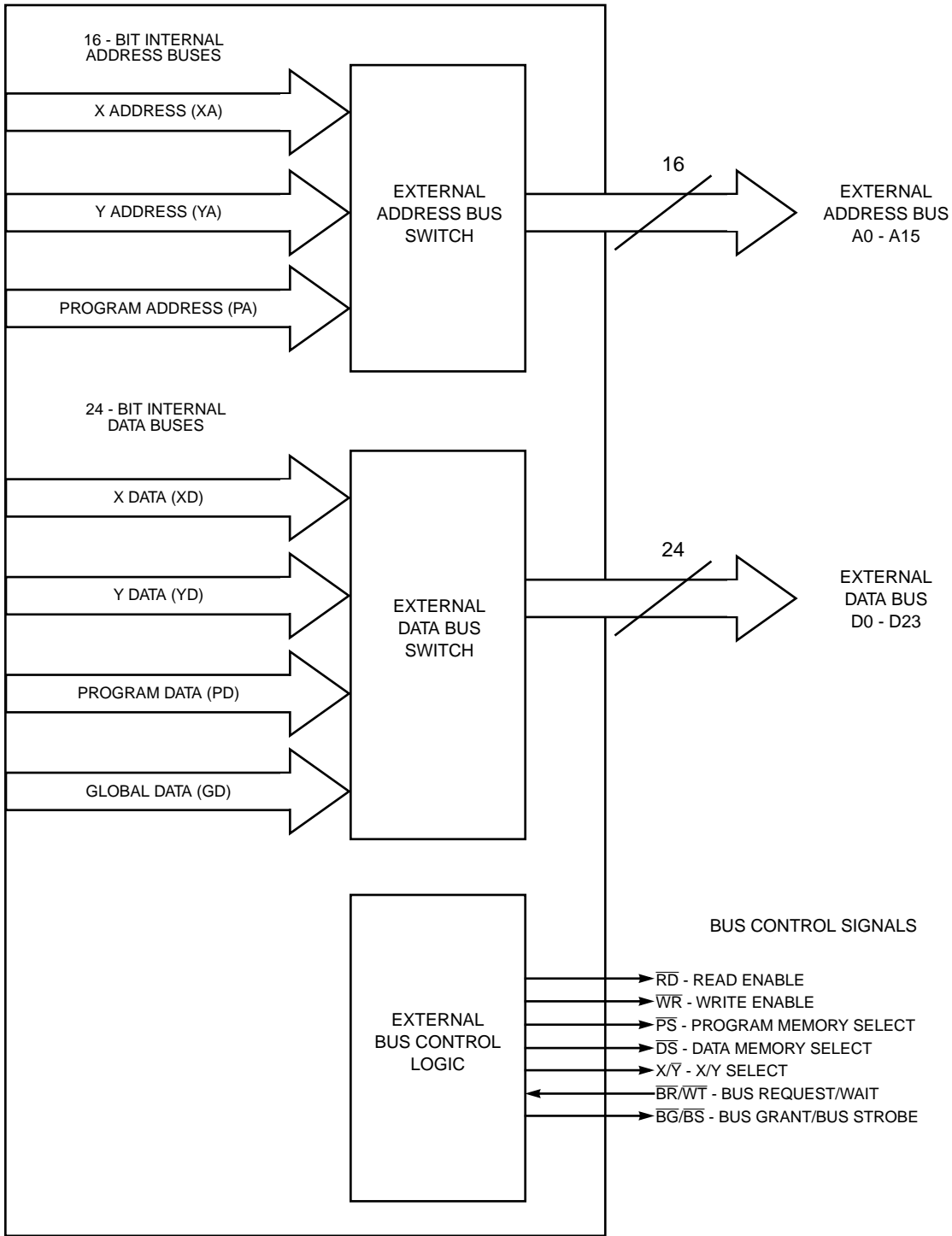


Figure 9-1 Port A Signals

additional instruction cycles because only one external memory access can occur per instruction cycle.

If more than one external access is required in one instruction cycle, the accesses will be made in the following priority: X memory, Y memory, and program memory. It takes one

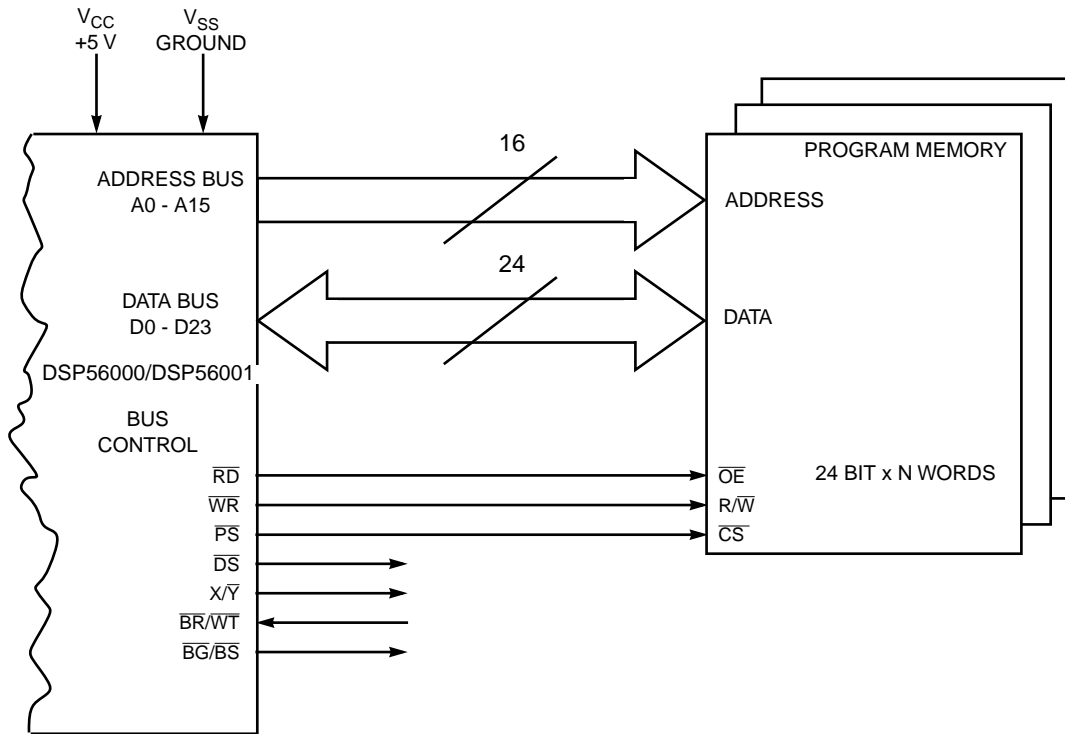


Figure 9-2 External Program Space

instruction cycle for each external memory access – i.e., one access can be executed in one instruction cycle, two accesses take two instruction cycles, etc. Since the external bus is only 24 bits wide, one XY or long external access will take two instruction cycles.

Figure 9-1 shows the port A signals divided into their three functional groups. The bus control signals can be subdivided into three additional groups: read/write control, address space selection, and bus access control. The read/write controls are self-descriptive. They can be used as decoded read and write controls, or, as seen in Figure 9-2, Figure 9-3, Figure 9-4, and Figure 9-6, the write signal can be used as the read/write control, and the read signal can be used as an output enable (or data enable) control for the memory. Decoding in this fashion simplifies connection to high-speed random-access memories (RAMs). The program memory select, data memory select, and X/Y select can be considered additional address signals, which extend the addressable memory from 64K words to 192K words

Since external logic delay is large relative to RAM timing margins, timing becomes more difficult as faster DSPs are introduced. The separate read and write strobes used by the DSP56000/DSP56001 are mutually exclusive, with a guard time between them to avoid two data buffers being enabled simultaneously. Other methods using external logic gates to generate the RAM control inputs require either faster RAM chips or external data buffers to avoid data bus buffer conflicts.

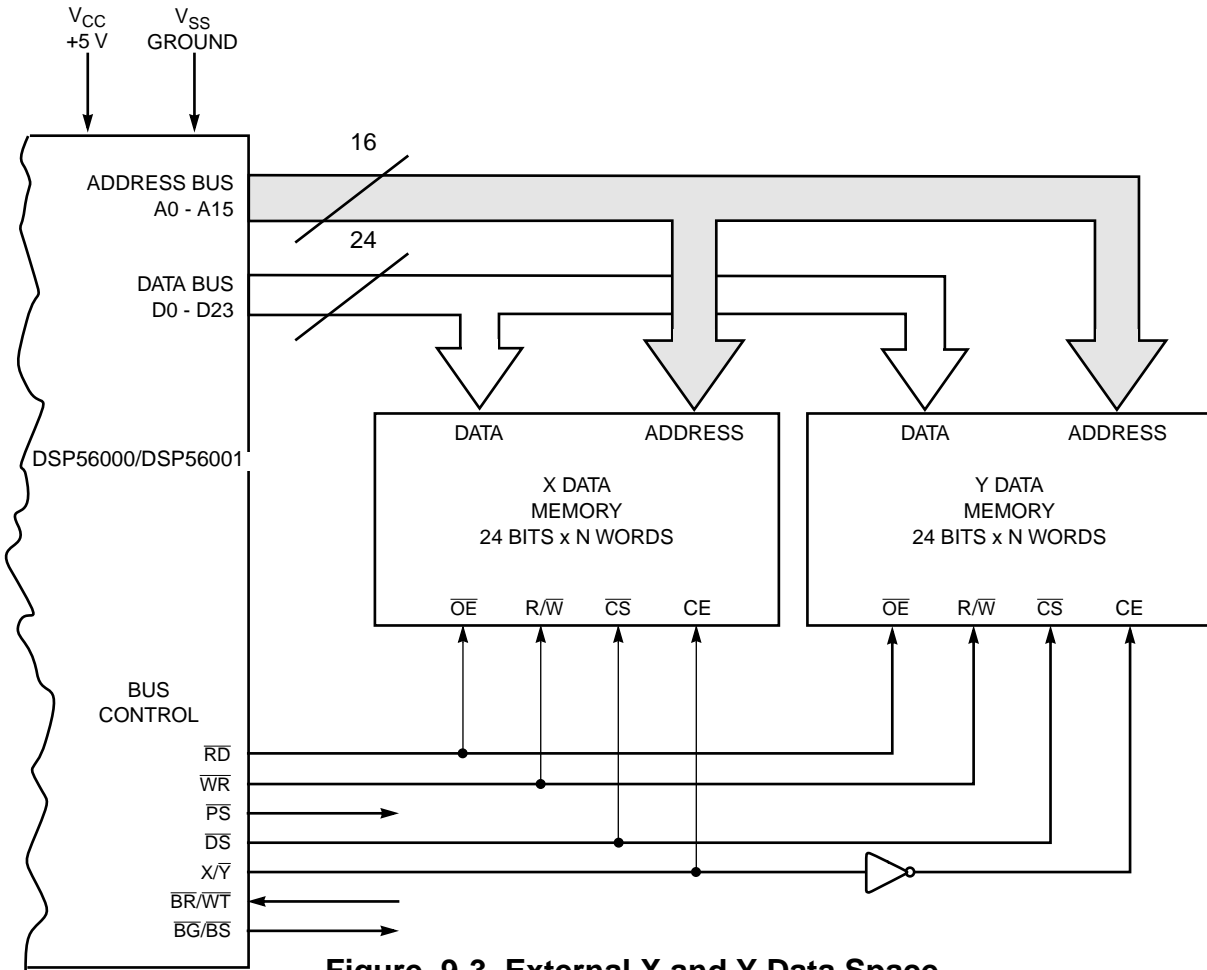


Figure 9-3 External X and Y Data Space

Additional DSP56000/DSP56001 peripherals can be memory mapped. An easy way to interface with MC6800 and MC68000 peripherals and to have an early read/write indication is to use the X/\bar{Y} output pin as an early R/\bar{W} indication. The peripheral chip select should be derived from the address lines and the data strobe so the peripheral registers appear in both X and Y data memory spaces at the same addresses. For a read operation, perform an X memory read:

MOVE X:PERIPHERAL,X0 ; X/\bar{Y} signal is high.

For a write operation, perform a Y memory write:

MOVE X0,Y:PERIPHERAL ; X/\bar{Y} signal is low.

Since the X/\bar{Y} output signal has the same timing as the address lines, it provides an early direction indication. The \bar{RD} and \bar{WR} signals are ANDed together to form a "data strobe" signal. The only restriction is that X and Y memory space must be external at the same address. Thus, the I/O short addressing mode and the MOVEP instruction cannot be used for this application. Otherwise, the hardware and software are trivial.

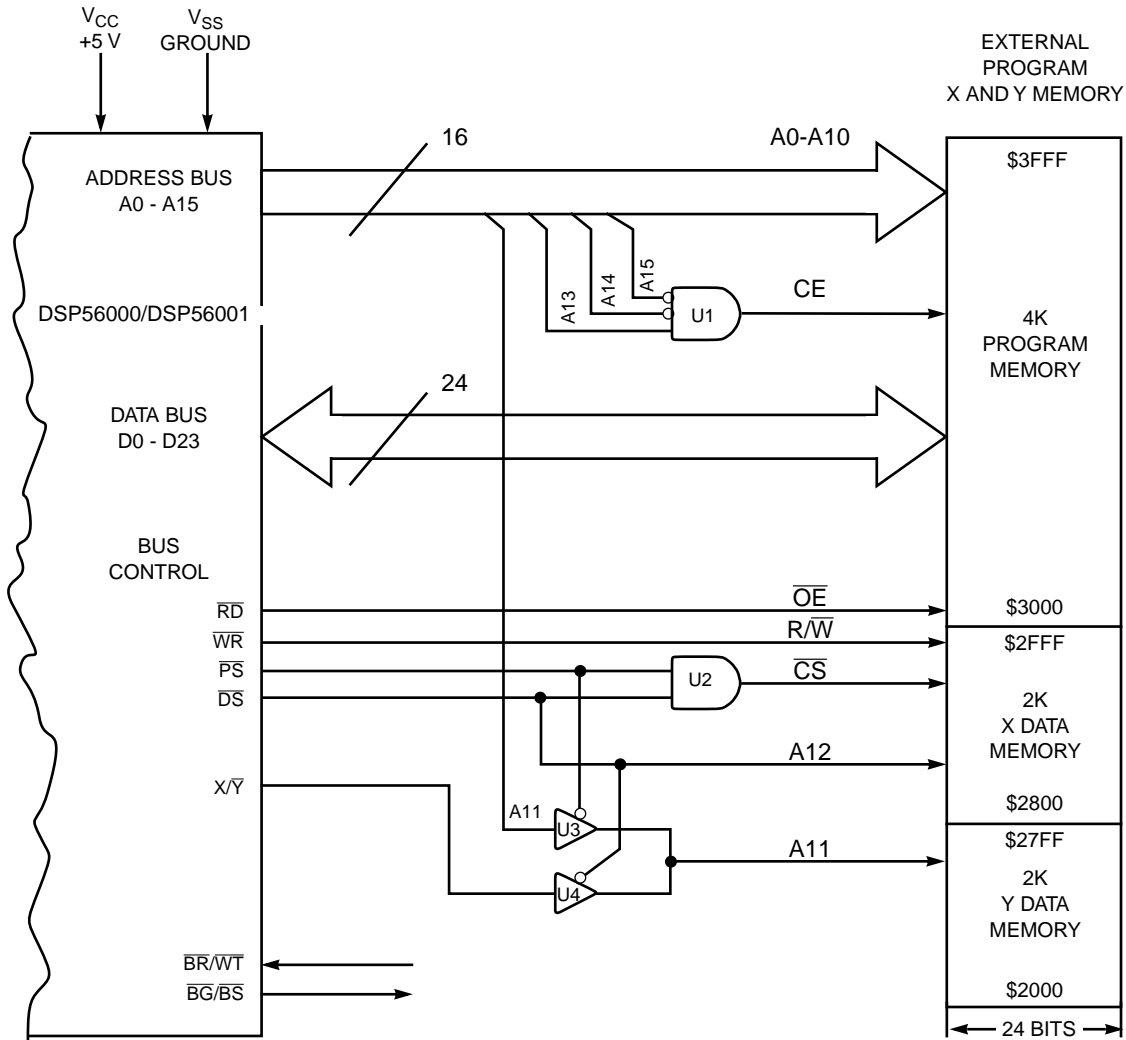
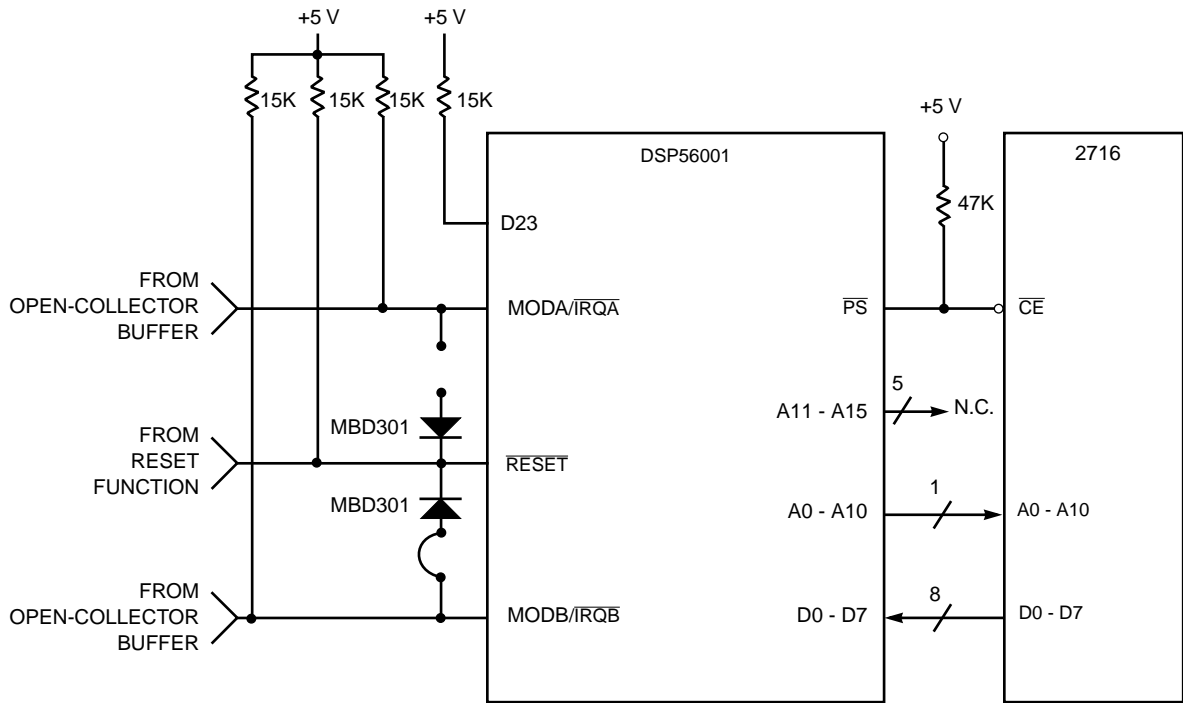


Figure 9-4 Memory Segmentation

Figure 9-2 shows an example of external program memory. A typical implementation of this circuit would use three-byte-wide static memories and would not require any additional logic. The \overline{PS} signal is used as the program-memory chip-select signal to enable the program memory at the appropriate time.

Figure 9-3 shows a similar circuit using the \overline{DS} signal to enable two data memories and using the X/\overline{Y} signal to select between them. The three external memory spaces (program, X data, and Y data) do not have to reside in separate physical memories; a single memory can be employed by using the \overline{PS} , \overline{DS} , and X/\overline{Y} signals as additional address lines to segment the memory into three spaces (see Figure 9-4). Table 9-1 shows how the \overline{PS} , \overline{DS} , and X/\overline{Y} signals are decoded. If the DSP is in the development mode, an exception fetch to any interrupt vector location will cause the X/\overline{Y} signal to go low when \overline{PS} is asserted. This procedure is useful for debugging and for allowing external circuitry to track interrupt servicing.

Special provisions have been made to allow the DSP to load a program from an inexpensive byte-wide ROM (see Figure 9-5 and the DSP56001 Advance Information Data Sheet (ADI1290) into internal program memory during a power-on reset. On powerup, the wait-state generator adds 15 wait states to all external memory accesses so that slow memory can be used. If bit 23 of external memory is a logic one, the DSP will load the contents of an external ROM into internal program memory (if bit 23 is a logic zero, it will load from the host port). The bootstrap program uses the bytes in three consecutive memory locations in the external ROM to build a single word in internal program memory. Figure 9-6 shows a system that uses internal program memory loaded from an external ROM during powerup and that splits the data memory space of a single memory bank into X: and Y: memory spaces. Although external program memory must be 24 bits, external data memory does not. Of course, this is application specific. However, many systems use 16 or fewer bits for A/D and D/A conversion, since they only need to store 16, 12, or even eight bits of data.



ADDRESS OF EXTERNAL BYTE-WIDE P MEMORY	CONTENTS LOADED TO INTERNAL PRAM AT:
P:\$C000	P:\$0000 LOW BYTE
P:\$C001	P:\$0000 MID BYTE
P:\$C002	P:\$0000 HIGH BYTE
⋮	⋮
P:\$C5FD	P:\$01FF LOW BYTE
P:\$C5FE	P:\$01FF MID BYTE
P:\$C5FF	P:\$01FF HIGH BYTE

Figure 9-5 Port A Bootstrap Circuit

The 24/56 bits of internal precision is usually sufficient for intermediate results. Recognizing this fact can save cost by reducing the number of external memory chips.

All unused inputs should have pullup resistors for two reasons: 1) floating inputs draw excessive power, and 2) a floating input can cause erroneous operation. For example, during RESET, all signals are three-stated. Without pullup resistors, the \overline{PS} and \overline{DS} signals may become active, causing two or more memory chips to try to simultaneously drive the external data bus, which can damage the memory chips. A pullup resistor in the 50K-ohm range should be sufficient.

9.2 PORT A TIMING

The external bus timing is defined by the operation of the address bus, data bus, and bus control pins. The transfer of data over the external data bus is synchronous with the clock. The timing A, B, and C relative to the edges of an external clock (see Figure 9-7 and Fig-

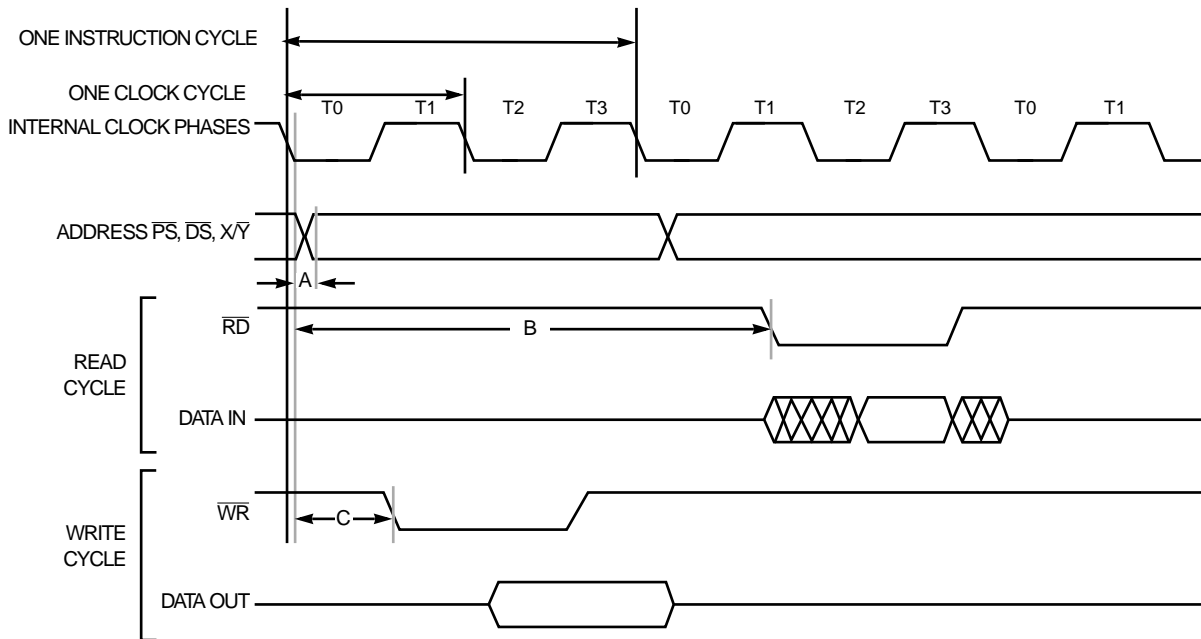


Figure 9-7 Port A Bus Operation with No Wait States

ure 9-8) are provided in the DSP56001 Advance Information Data Sheet (ADI1290). This timing is essential for designing synchronous multiprocessor systems. Figure 9-7 shows the port A timing with no wait states (wait-state control is discussed in 9.2.1 Port A Wait States). One instruction cycle equals two clock cycles or four clock phases. The clock phases, which are numbered T0 – T3, are used for timing on the DSP. Figure 9-8 shows the same timing with two wait states added to the external X: memory access. Four TW clock phases have been added because one wait state adds two T phases and is equiv-

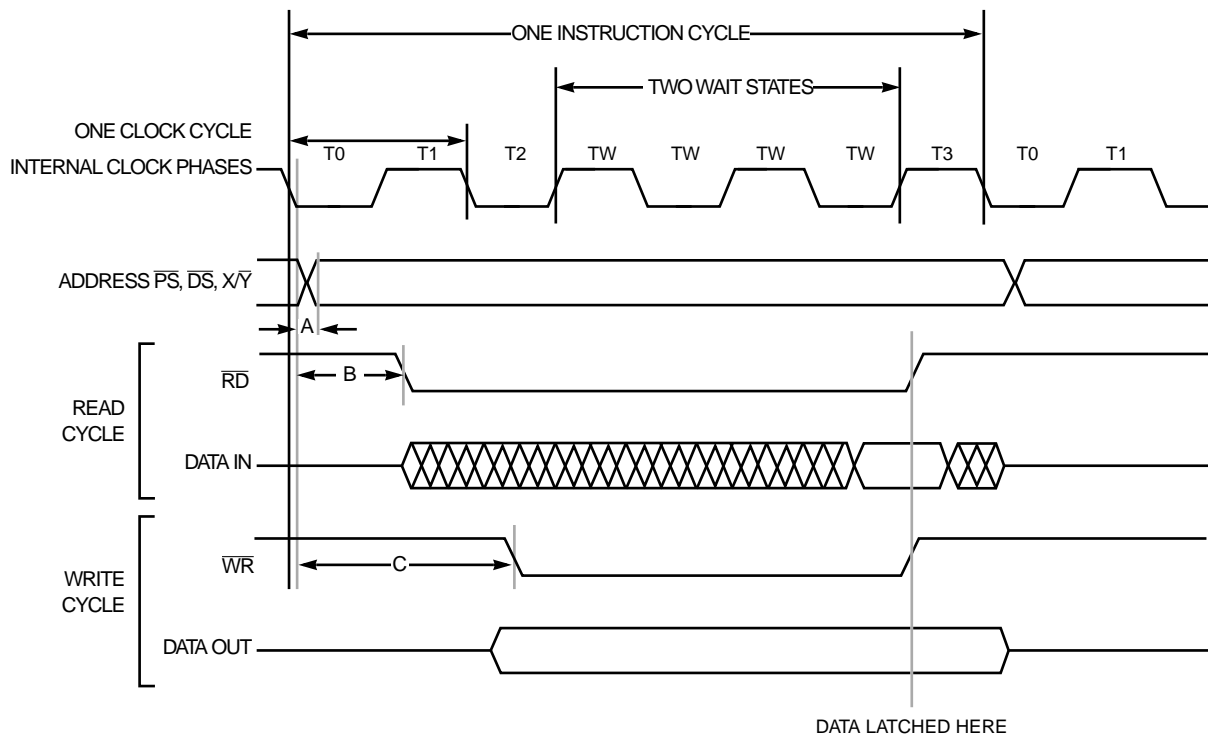


Figure 9-8 Port A Bus Operation with Two Wait States

T1 to the T2 state when one or more wait states are added to ease interfacing to the port. Each external memory access requires the following procedure:

1. The external memory address is defined by the address bus (A0–A15) and the memory reference selects (\overline{PS} , \overline{DS} , and X/\overline{Y}). These signals change in the first phase (T0) of the bus cycle. Since the memory reference select signals have the same timing as the address bus, they may be used as additional address lines. The address and memory reference signals are also used to generate chip-select signals for the appropriate memory chips. These chip-select signals change the memory chips from low-power standby mode to active mode and begin the read access time. This mode change allows slower memories to be used since the chip-select signals can be address based rather than read or write enable based. Read and write enable do not become active until after the address is valid. See the timing diagrams in the DSP56001 Advance Information Data Sheet (ADI1290) for detailed timing information.
2. When the address and memory reference signals are stable, the data transfer is enabled by read enable (\overline{RD}) or write enable (\overline{WR}). \overline{RD} or \overline{WR} is asserted to “qualify” the address and memory reference signals as stable and to perform the read or write data transfer. \overline{RD} and \overline{WR} are asserted in the second phase of the bus cycle (if there are no wait states). Read enable is typically con-

Table 9-1 Program and Data Memory Select Encoding

PS	DS	X/Y	External Memory Reference
1	1	1	No Activity
1	0	1	X Data Memory on Data Bus
1	0	0	Y Data Memory on Data Bus
0	1	1	Program Memory on Data Bus (Not an Exception)
0	1	0	External Exception Fetch: Vector or Vector +1 (Development Mode Only)
0	0	X	Reserved
1	1	0	Reserved

nected to the output enable (\overline{OE}) of the memory chips and simply controls the output buffers of the chip-selected memory. Write enable is connected to the write enable (\overline{WE}) or write strobe (\overline{WS}) of the memory chips and is the pulse that strobes data into the selected memory. For a read operation, \overline{RD} is asserted and \overline{WR} remains deasserted. Since write enable remains negated, a memory read operation is performed. The DSP data bus becomes an input, and the memory data bus becomes an output. For a write operation, \overline{WR} is asserted and \overline{RD} remains deasserted. Since read enable remains deasserted, the memory chip outputs remain in the high-impedance state even before write strobe is asserted. This state assures that the DSP and the chip-selected memory chips are not enabled onto the bus at the same time. The DSP data bus becomes an output, and the memory data bus becomes an input.

3. Wait states are inserted into the bus cycle by a wait-state counter or by asserting \overline{WT} . The wait-state counter is loaded from the bus control register. If the value loaded into the wait-state counter is zero, no wait states are inserted into the bus cycle, and \overline{RD} and \overline{WR} are asserted as shown in Figure 9-7. If a value $W \neq 0$ is loaded into the wait state counter, W wait states are inserted into the bus cycle. When wait states are inserted into an external write cycle, \overline{WR} is delayed from $T1$ to $T2$. The timing for the case of two wait states ($W=2$) is shown in Figure 9-8.
4. When \overline{RD} or \overline{WR} are deasserted at the start of $T3$ in a bus cycle, the data is latched in the destination device – i.e., when \overline{RD} is deasserted, the DSP latches the data internally; when \overline{WR} is deasserted, the external memory latches the data on the positive-going edge. The address signals remain stable until the first phase of the next external bus cycle to minimize power dissi-

pation. The memory reference signals (\overline{PS} , \overline{DS} , and X/\overline{Y}) are deasserted (held high) during periods of no bus activity, and the data signals are three-stated. For read-modify-write instructions such as BSET, the address and memory reference signals remain active for the complete composite (i.e., two I_{cyc}) instruction cycle.

Figure 9-9 shows an example of mixing different memory speeds and memory-mapped peripherals in different address spaces. The internal memory uses no wait states, X: memory uses two wait states, Y: memory uses four wait states, P: memory uses five wait states, and the analog converters use 14 wait states. Controlling five different devices at five different speeds requires only one additional logic package. Half the gates in that package are used to map the analog converters to the top 64 memory locations in Y: memory.

Adding wait states to external memory accesses can substantially reduce power requirements. Table 9-2 shows how the power was reduced during external memory and I/O operations by changing from zero to 15 wait states at four different clock speeds in a test circuit.

Table 9-2 Power Requirements for Minimum and Maximum External Memory Wait States

Clock	Current for 0 Wait States	Current for 15 Wait States
4.000 MHz	19.8 mA	8.6 mA
6.5536 MHz	31.0 mA	12.8 mA
10.245 MHz	46.8 mA	18.8 mA
20.000 MHz	91.0 mA	36.6 mA

9.2.1 Port A Wait States

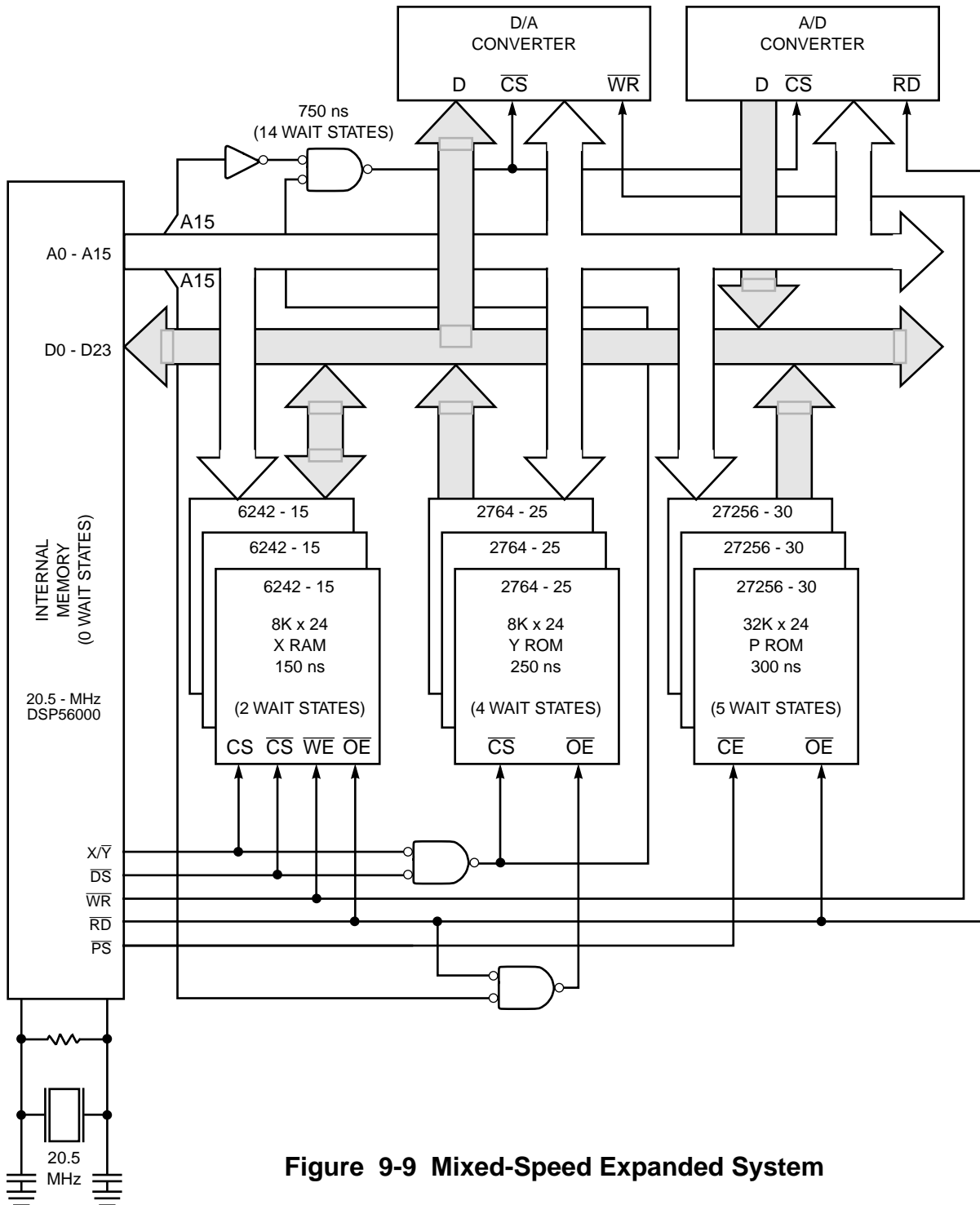
The DSP56000/DSP56001 features two methods to allow the user to accommodate slow memory by changing the port A bus timing. The first method uses the bus control register (BCR), which allows a fixed number of wait states to be inserted in a given memory access to all locations in each of the four memory spaces: X, Y, P, and I/O. The second method uses the bus strobe/wait ($\overline{BS}/\overline{WT}$) facility, which allows an external device to insert an arbitrary number of wait states when accessing either a single location or multiple locations of external memory or I/O space. Wait states are executed until the external device releases the DSP to finish the external memory cycle.

9.2.2 Bus Control Register

The expansion bus timing is controlled by the BCR (see Figure 9-10) which controls the

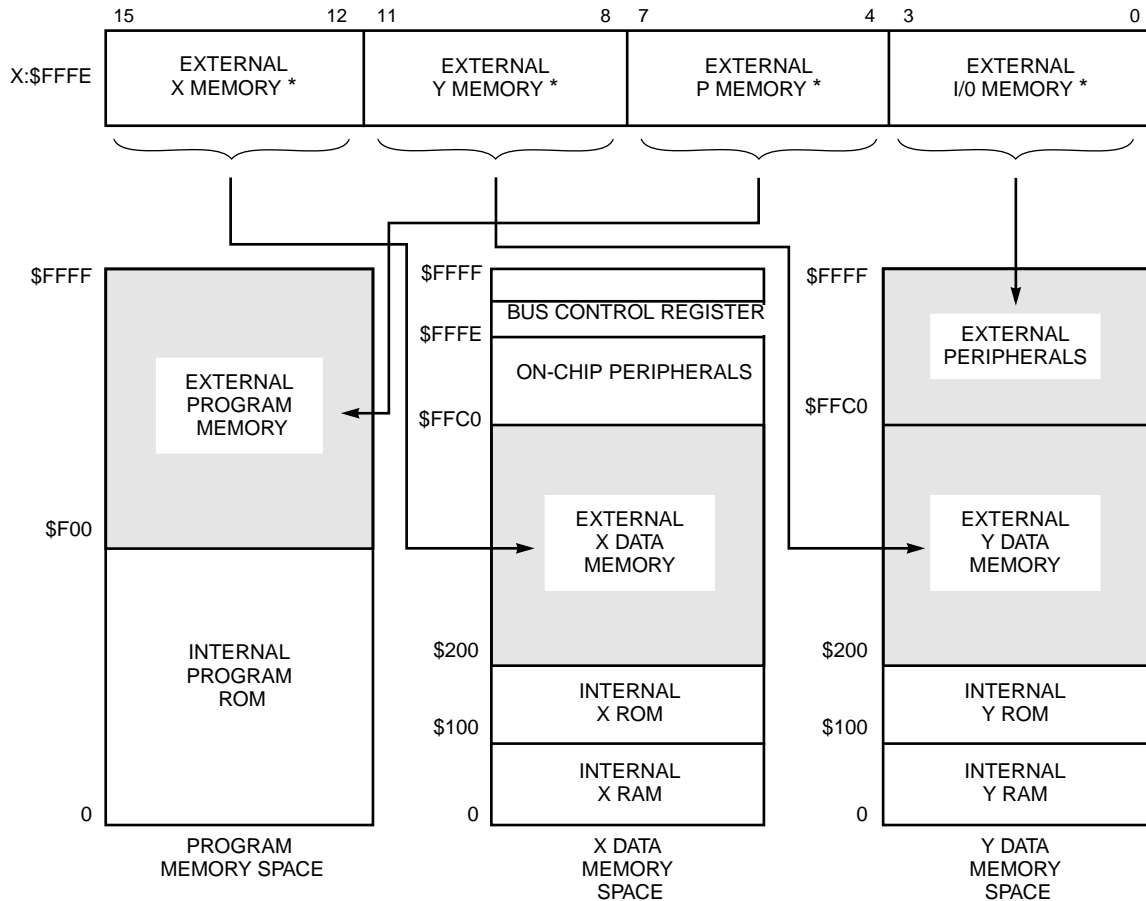
PORT A BUS CONTROL REGISTER (BCR)

	EXTERNAL X MEMORY	EXTERNAL Y MEMORY	EXTERNAL P MEMORY	EXTERNAL I/O MEMORY
X:\$FFE	0010	0100	0101	1110



timing of the bus interface signals, \overline{RD} and \overline{WR} , and the data output lines. Each of the

memory spaces, X data, Y data, program data, and I/O, has its own 4-bit BCR, which can be programmed for inserting up to 15 wait states (each wait state adds one-half instruction cycle to each memory access – i.e., 50 ns for a 20-Mhz clock). In this way, external bus timing can be tailored to match the speed requirements of the different memory spaces. On processor RESET, the BCR is preset to all ones (15 wait states).



* Zero to 15 wait states can be inserted into each external memory access.

Figure 9-10 Bus Control Register

Figure 9-10 illustrates which of the four BCR subregisters affect which external memory space. The BCR is a memory-mapped register located at X:\$FFFE. All the internal peripheral devices are memory mapped, and their control registers reside between X:\$FC00 and X:\$FFFF. Loading the BCR as shown in Figure 9-9 can be accomplished by executing a “MOVEP #\$245E, X:\$FFFE” instruction. Changing individual bits in one of the four subregisters can be accomplished by using the BSET and BCLR instructions.

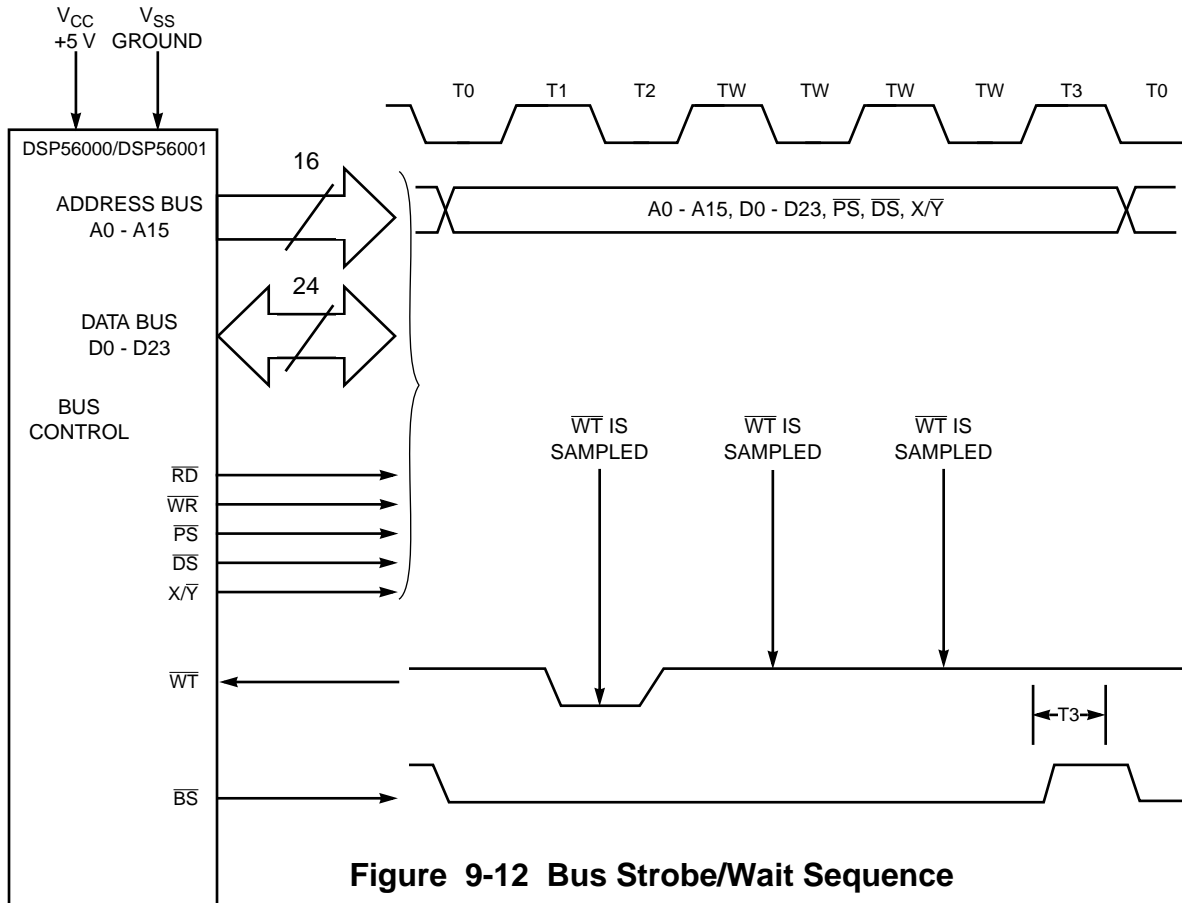
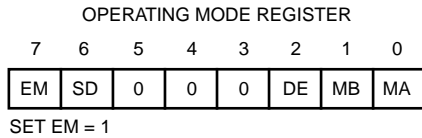


Figure 9-12 Bus Strobe/Wait Sequence

external device to request and be given control of the external memory bus (port A) while the DSP continues internal operations using internal memory spaces. This configuration allows a bus controller to arbitrate a multiple bus-master system. (A bus master can issue

Table 9-3 Wait State Control

BCR Contents	\overline{WT}	Number of Wait States Generated
0	Deasserted	0
0	Asserted	2 (minimum)
> 0	Deasserted	Equals value in BCR
> 0	Asserted	Minimum equals 2 or value in BCR. Maximum is determined by \overline{WT} .

addresses on the bus; a bus slave can respond to addresses on the bus. A single device can be both a master and a slave, but can only be one or the other at any given time.) The $\overline{BS}/\overline{BW}$ mode allows a bus arbitrator to extend the bus cycle of the DSP56000/DSP56001 to allow another bus master time to finish its bus access before allowing the DSP56000/DSP56001 access to the bus.

9.3.1 Bus Request/Bus Grant

The $\overline{BR}/\overline{BG}$ mode is selected if OMR bit 7 (see Figure 9-11) is set to zero (execute an “ANDI #7F,OMR” instruction). When \overline{BR} is asserted (see Figure 9-13), the DSP will assert \overline{BG} after the current external access cycle completes and will simultaneously three-state the port A signals (see the DSP56001 Advance Information Data Sheet (ADI1290) for exact timing of $\overline{BR}/\overline{BG}$). The bus is then available to be used by the bus master requesting the bus. When \overline{BR} is deasserted, \overline{BG} is deasserted after the current external access, and the port A signals are no longer three-stated. Reset clears bit 7 of the OMR. Information on operation of the $\overline{BR}/\overline{BG}$ pins after executing a WAIT or STOP instruction can be found in 8.4 WAIT PROCESSING STATE and 8.5 STOP PROCESSING STATE.

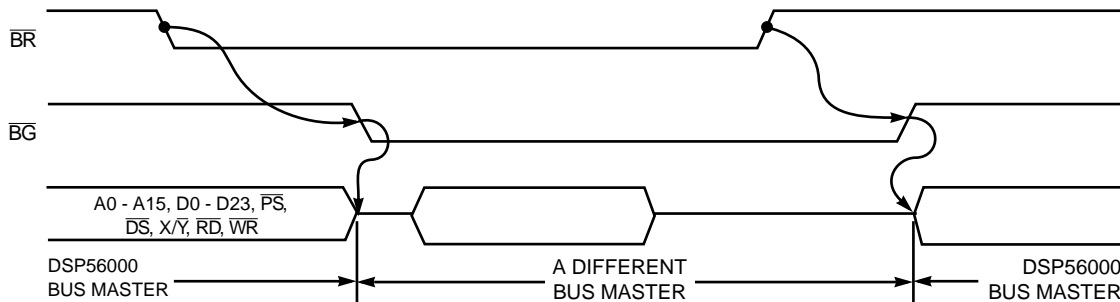


Figure 9-13 Bus Request/Bus Grant Sequence

9.3.2 Shared Memory

The bus control signals described in the previous paragraphs provide the means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) to the port A bus. Four arbitration examples will be described in the following paragraphs: 1) bus arbitration using only $\overline{BR}/\overline{BG}$ with internal control, 2) bus arbitration using only $\overline{BR}/\overline{BG}$ with external control, 3) bus arbitration using $\overline{BR}/\overline{BG}$ and $\overline{BS}/\overline{WT}$ with no overhead, and 4) signaling using semaphores.

9.3.2.1 Bus Arbitration Using Only BR/BG With Internal Control

Perhaps the simplest example of a shared memory system using a DSP56000/DSP56001 is shown in Figure 9-14. The bus arbitration is performed internal to the DSP#2 by using

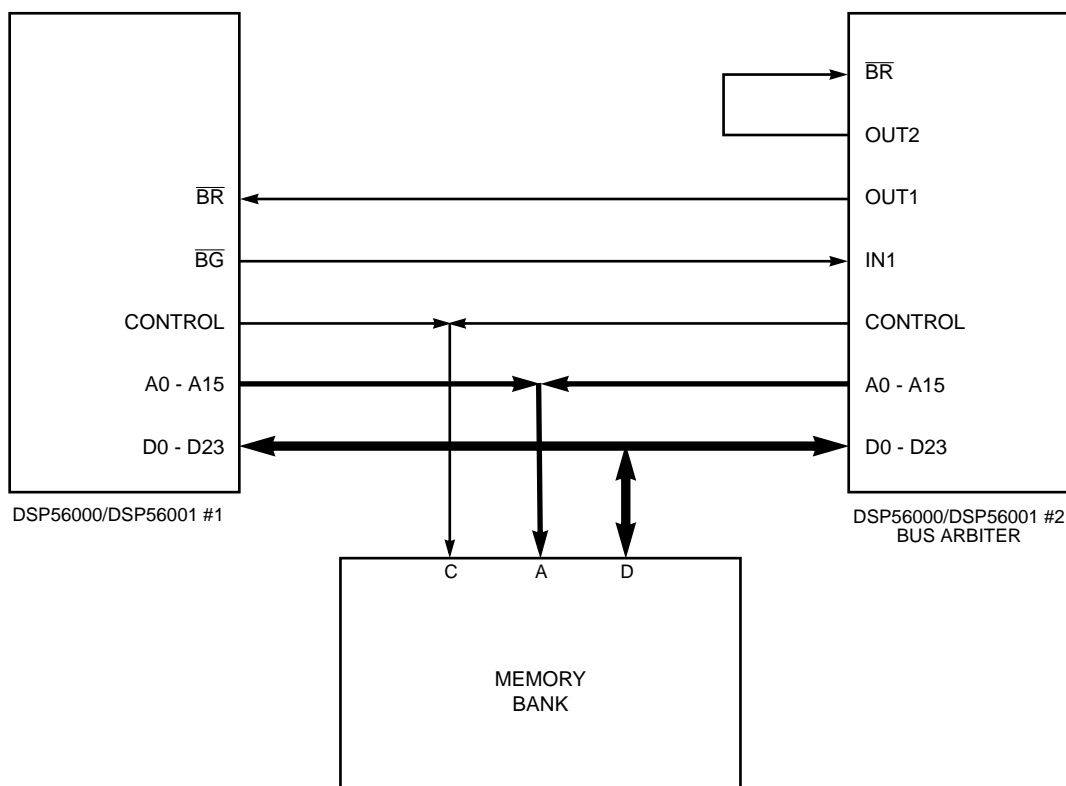


Figure 9-14 Bus Arbitration Using Only $\overline{BR}/\overline{BG}$ with Internal Control

software. DSP#2 controls all bus operations by using I/O pin OUT2 to three-state its own port A and by never accessing port A without first calling the subroutine that arbitrates the bus. When the DSP#2 needs to use external memory, it uses I/O pin OUT1 to request bus access and I/O pin IN1 to read bus grant. DSP#1 does not need any extra code for bus arbitration since the $\overline{BR}/\overline{BG}$ hardware handles its bus arbitration automatically. The protocol for bus arbitration is as follows:

At RESET: DSP#2 sets OUT2=0 ($\overline{BR}\#2=0$) and OUT1=1 ($\overline{BR}\#1=1$), which gives DSP#1 access to the bus and suspends DSP#2 bus access.

When DSP#2 wants control of the memory, the following steps are performed (see Figure 9-15):

1. DSP# 2 sets OUT1=0 ($\overline{BR}\#1=0$).
2. DSP# 2 waits for IN1=0 ($\overline{BG}\#1=0$ and DSP#1 off the bus). This takes at most $13T+4T \cdot WS+20$ ns (about 400 ns at 20 MHz) where T is $I_{cyc}/4$ and WS is the number of wait states used by DSP# 1. If DSP#1 is not using any read/modify/write instructions in its external space, the maximum becomes only $9T+2T \cdot WS+20$ ns (about 250 ns at 20 MHz).

3. DSP#2 sets $OUT2=1$ ($\overline{BR\#2}=1$ to let DSP#2 on the bus).
4. DSP#2 accesses the bus for block transfers, etc. at full speed.
5. To release the bus, DSP#2 sets $OUT2=0$ ($\overline{BR\#2}=0$) after the last external access.
6. DSP#2 then sets $OUT1=1$ ($\overline{BR\#1}=1$) to return control of the bus to DSP#1.
7. DSP#1 then acknowledges mastership by deasserting $\overline{BG\#1}$.

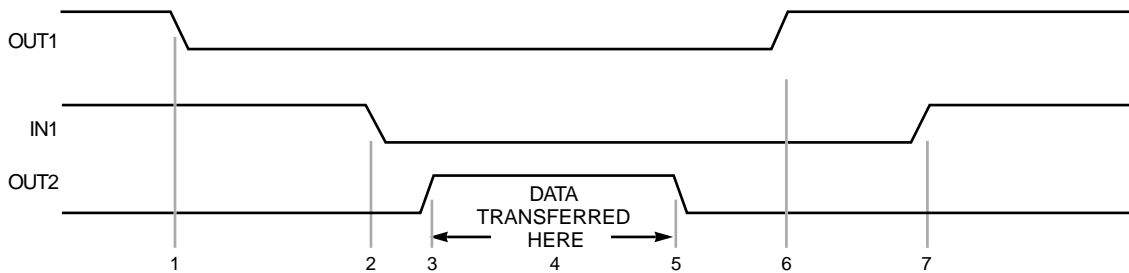


Figure 9-15 Two DSPs with External Bus Arbitration Timing

9.3.2.2 Bus Arbitration Using Only $\overline{BR}/\overline{BG}$ With External Control

Figure 9-16 can be implemented with external bus arbitration logic, which will save processing capacity on the DSPs and can make bus access much faster at a cost of additional hardware. Operation is similar to the system shown in Figure 9-14. The bus arbitration logic takes control of the external bus by deasserting an enable signal (E1, E2, and E3) to all DSPs, which will then acknowledge by granting the bus ($\overline{BG}=0$). When a DSP (DSP#1 in Figure 9-16) wants the bus, it will jump to a subroutine, which will set $PC3=1$. When the arbitration logic grants the bus to a DSP, it will issue a $BG1$ ($BG2$ for DSP#2; $BG3$ for DSP#3) to let the DSP know that it can have the bus. Arbitration logic will then enable the bus by asserting the appropriate enable ($E1=1$). When the DSP is ready to relinquish the bus, it deasserts $PC3$, and the arbiter deasserts $E1$ and $BG1$.

9.3.2.3 Bus Arbitration Using $\overline{BR}/\overline{BG}$ and $\overline{BS}/\overline{WT}$ With No Overhead

By using the circuit shown in Figure 9-17, two DSPs can share memory with hardware arbitration that requires no software on the part of the DSPs. In Figure 9-17, DSP#1 has $EM=1$ in its OMR, and DSP#2 has $EM=0$ in its OMR. The protocol for bus arbitration in Figure 9-17 is as follows:

At RESET: \overline{BG} of DSP#2 is deasserted, which three-states the buffers, giving DSP#2 control of the memory. Reset causes DSP#1 to initially be in the $\overline{BR}/\overline{BG}$ mode. DSP#1 OMR bit 7 must be set by software during initialization to change $\overline{BR}/\overline{BG}$ to $\overline{BS}/\overline{WT}$.

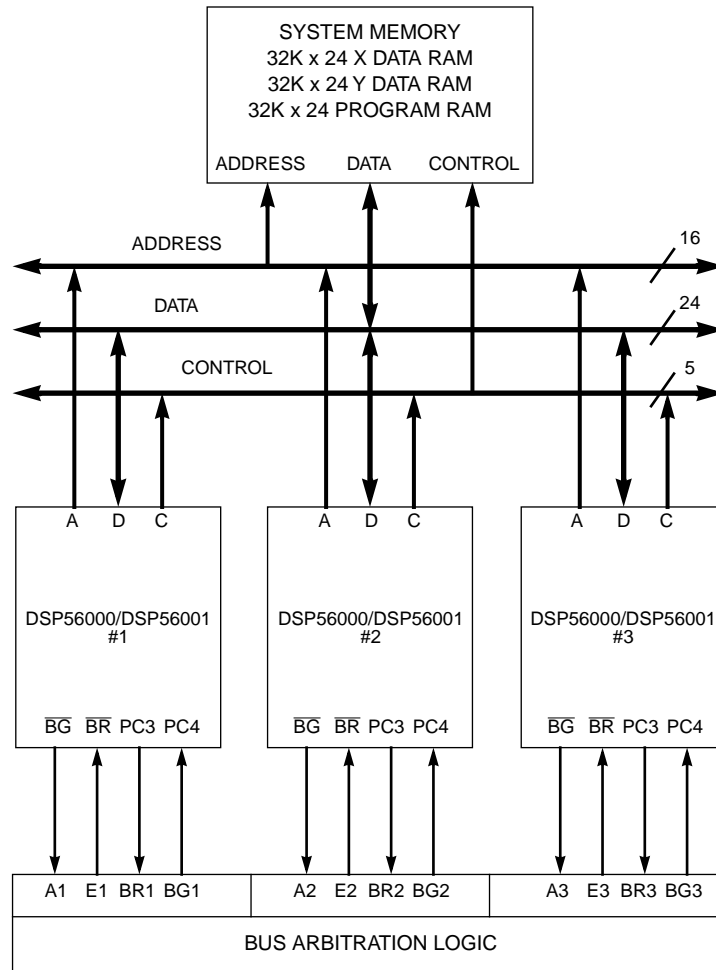


Figure 9-16 Bus Arbitration Using Only $\overline{BR}/\overline{BG}$ with External Control

When DSP#1 wants control of the memory the following steps are performed (see Figure 9-18):

1. DSP#1 makes an external access, thereby asserting \overline{BS} , which asserts \overline{WT} (causing DSP#1 to execute wait states in the current cycle) and asserts DSP#2 \overline{BR} (requesting that DSP#2 release the bus).
2. When DSP#2 finishes its present bus cycle, it three-states its bus drivers and asserts \overline{BG} . Asserting \overline{BG} enables the three-state buffers, placing the DSP#1 signals on the memory bus. Asserting \overline{BG} also deasserts \overline{WT} , which allows DSP#1 to finish its bus cycle.
3. When DSP#1's memory cycle is complete, it releases \overline{BS} , which deasserts \overline{BR} . DSP#2 then deasserts \overline{BG} , three-stating the buffers and allowing DSP#2 to access the memory bus.

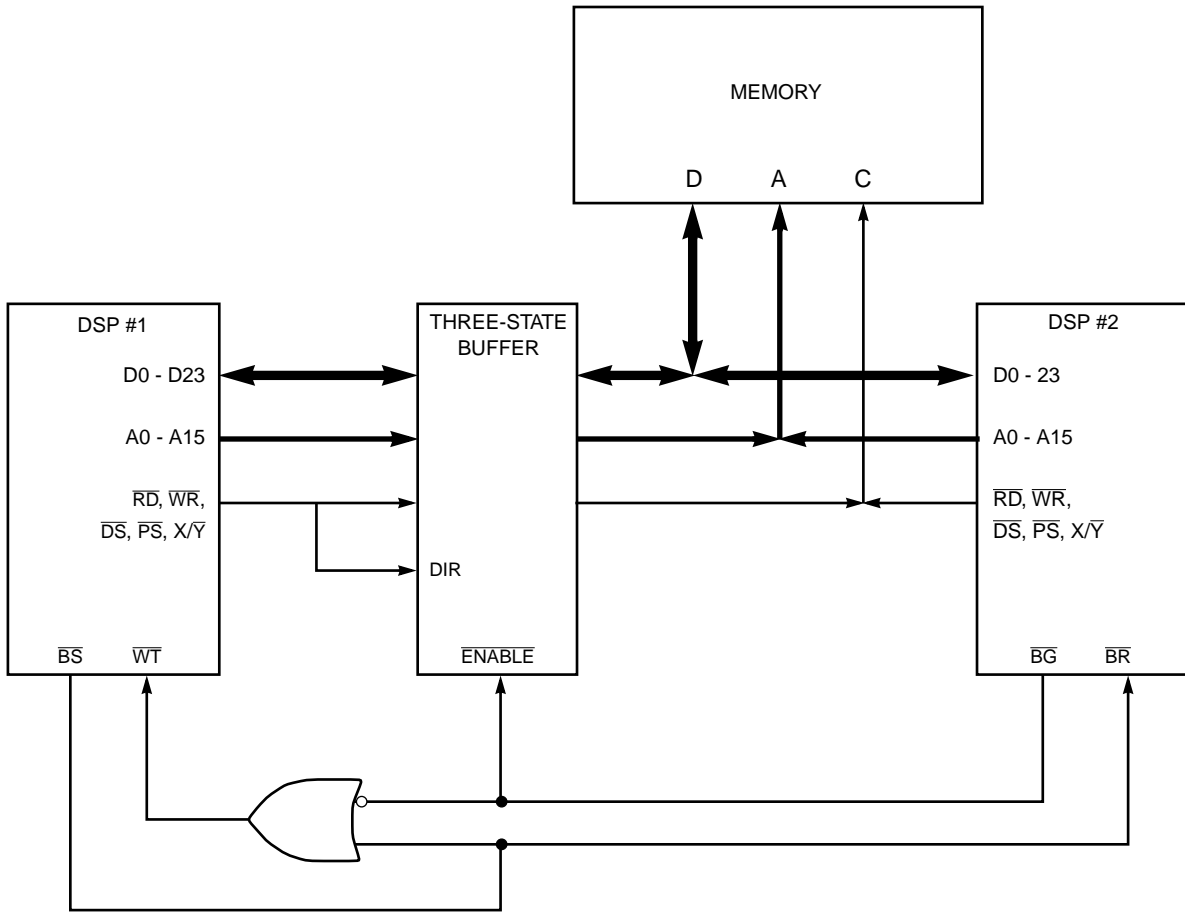


Figure 9-17 Bus Arbitration Using $\overline{BR}/\overline{BG}$ and $\overline{BS}/\overline{WT}$ with No Overhead

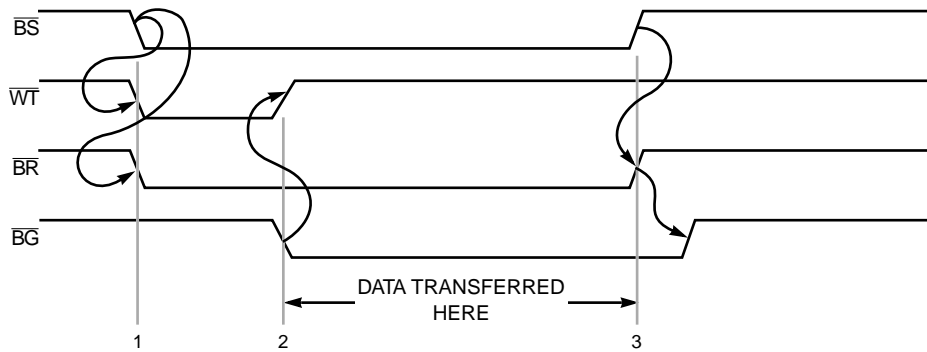


Figure 9-18 Two DSPs with External Bus Arbitration Timing

9.3.2.4 Signaling Using Semaphores

Figure 9-19 shows a more sophisticated shared memory system that uses external arbitration with both local external memory and shared memory. The four semaphores are bits in one of the words in each shared memory bank used by software to arbitrate memory use. Semaphores are commonly used to indicate that the contents of the semaphore's memory blocks are being used by one processor and are not available for use by another processor. Typically, if the semaphore is cleared, the block is not allocated to a processor; if the semaphore is set, the block is allocated to a processor.

Without semaphores, one processor may try to use data while it is being changed by another processor, which may cause errors. This problem can occur in a shared memory system when separate test and set instructions are used to "lock" a data block for use by a single processor.

The correct procedure is to test the semaphore and then set the semaphore if it was clear to lock and gain exclusive use of the data block. The problem occurs when the second processor acquires the bus and tests the semaphore after the first processor tests the semaphore but before the first processor can lock the data block. The incorrect sequence is 1) the first processor tests the semaphore and sees that the block is available; 2) the second processor then tests the bit and also sees that the block is available; 3) both processors then set the bit to lock the data; and 4) both proceed to use the data on the assumption that the data cannot be changed by another processor.

The DSP56000/DSP56001 has a group of instructions designed to prevent this problem. They perform an indivisible read-modify-write operation and do not release the bus between the read and write (specifically, \overline{DS} , \overline{PS} , and X/\overline{Y} do not change state). Not releasing the bus allows these instructions to test the semaphore and then to set, clear, or change the semaphore without the possibility of another processor testing the semaphore before it is changed. The instructions are bit test and change (BCHG), bit test and clear (BCLR), and bit test and set (BSET). The proper way to set the semaphore to gain exclusive access to a memory block is to use BSET to test the semaphore and to set it to one. After the bit is set, the result of the test operation will reveal if the semaphore was clear before it was set by BSET and if the memory block is available. If the bit was already set and the block is in use by another processor, the DSP will wait to access the memory block.

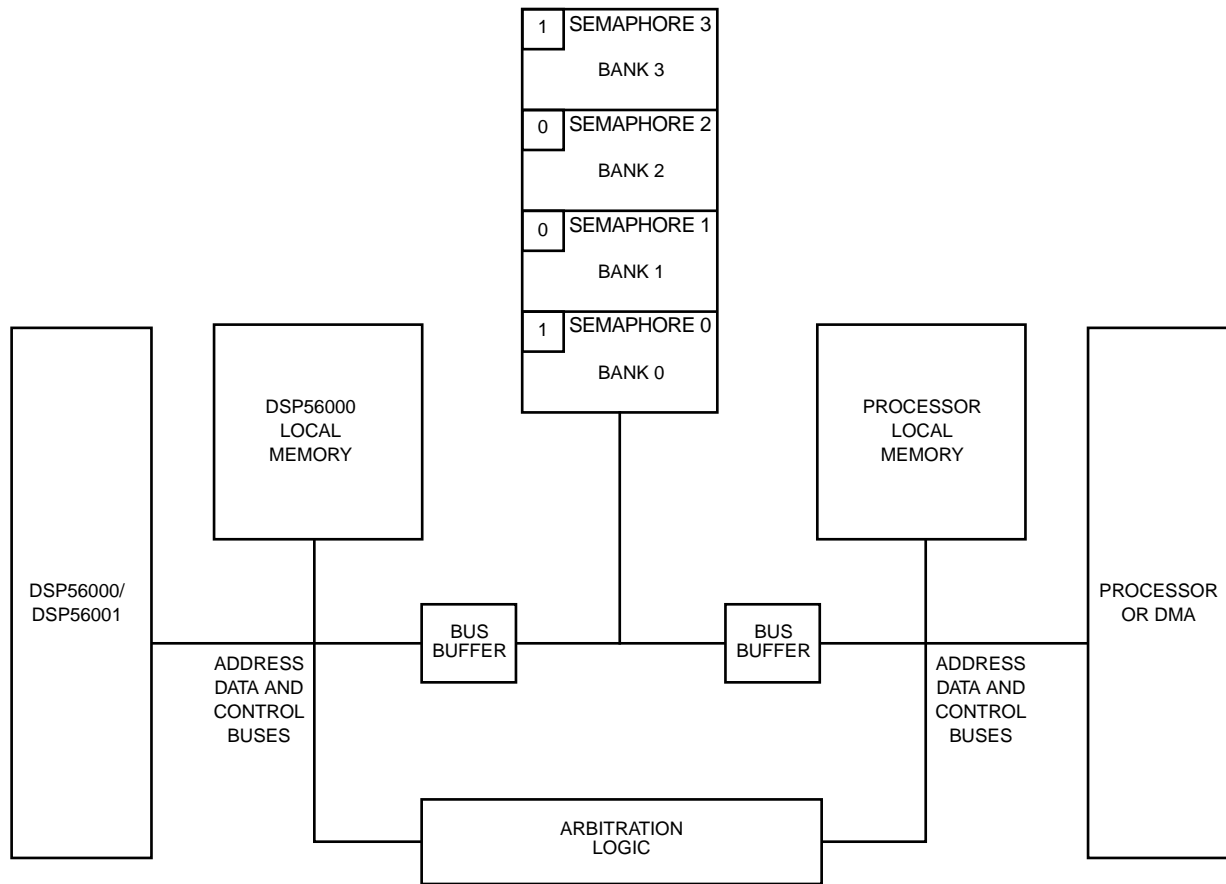


Figure 9-19 Signaling Using Semaphores

