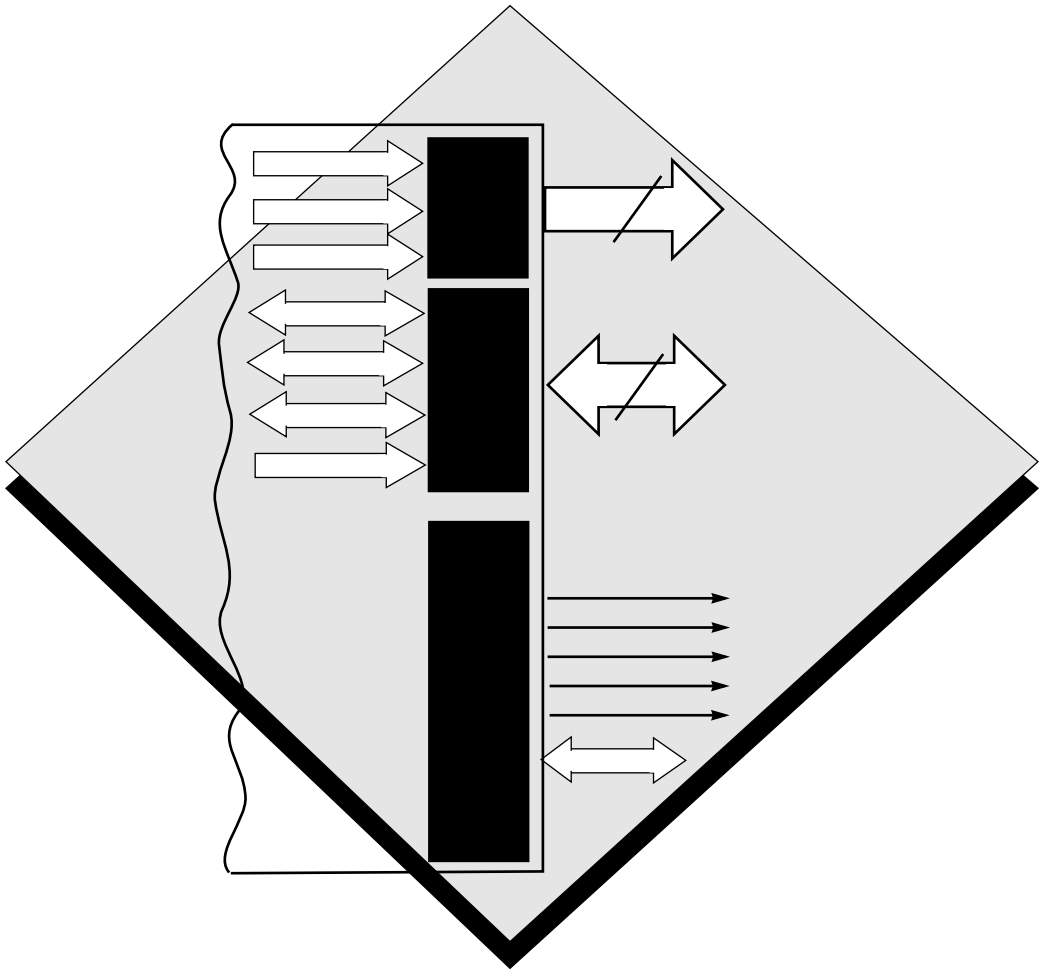
SECTION 4

EXTERNAL MEMORY INTERFACE



SECTION CONTENTS

4.1 INTRODUCTION

The External Memory Interface (often refered to as Port A) provides a versatile interface to external memory, allowing economical connection with fast memories/devices, slow memories/devices, and multiple bus master systems.

The external memory interface has two power-reduction features. It can access internal memory spaces, toggling only the external memory signals that need to change, thereby eliminating unneeded switching current. Also, if conditions allow the processor to operate at a lower memory speed, wait states can be added to the external memory access to significantly reduce power while the processor accesses those memories.

4.2 INTERFACE

The DSP56003/005 processor can access one or more of its memory sources (X data memory, Y data memory, and program memory) while it executes an instruction. The 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. The external memory interface's one address bus and one data bus 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 chip, memory references may require additional instruction cycles because only one external memory access can occur per instruction cycle.

If an instruction cycle requires more than one external access, the processor will make the accesses in the following priority: X memory, Y memory, and program memory. It takes one 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 data bus is only 24 bits wide, one XY or long external access will take two instruction cycles. The 16-bit address bus can sustain a rate of one memory access per instruction cycle (using no-wait-state memory which is discussed in Section 4.4 — Wait States).

Figure 4-1 shows the external memory interface signals divided into their three functional groups: address bus signals (A0-A15), data bus signals (D0-D15), and bus control. The bus control signals can be subdivided into three additional groups: read/write control ($\overline{RD}$ and $\overline{WR}$), address space selection (including program memory select ($\overline{PS}$), data memory select ($\overline{DS}$), external peripheral select ($\overline{EXTP}$), and X/$\overline{Y}$ select) and bus access control ($\overline{BN}$, $\overline{BR}$, $\overline{BG}$, $\overline{WT}$, $\overline{BS}$ — DSP56003 only).

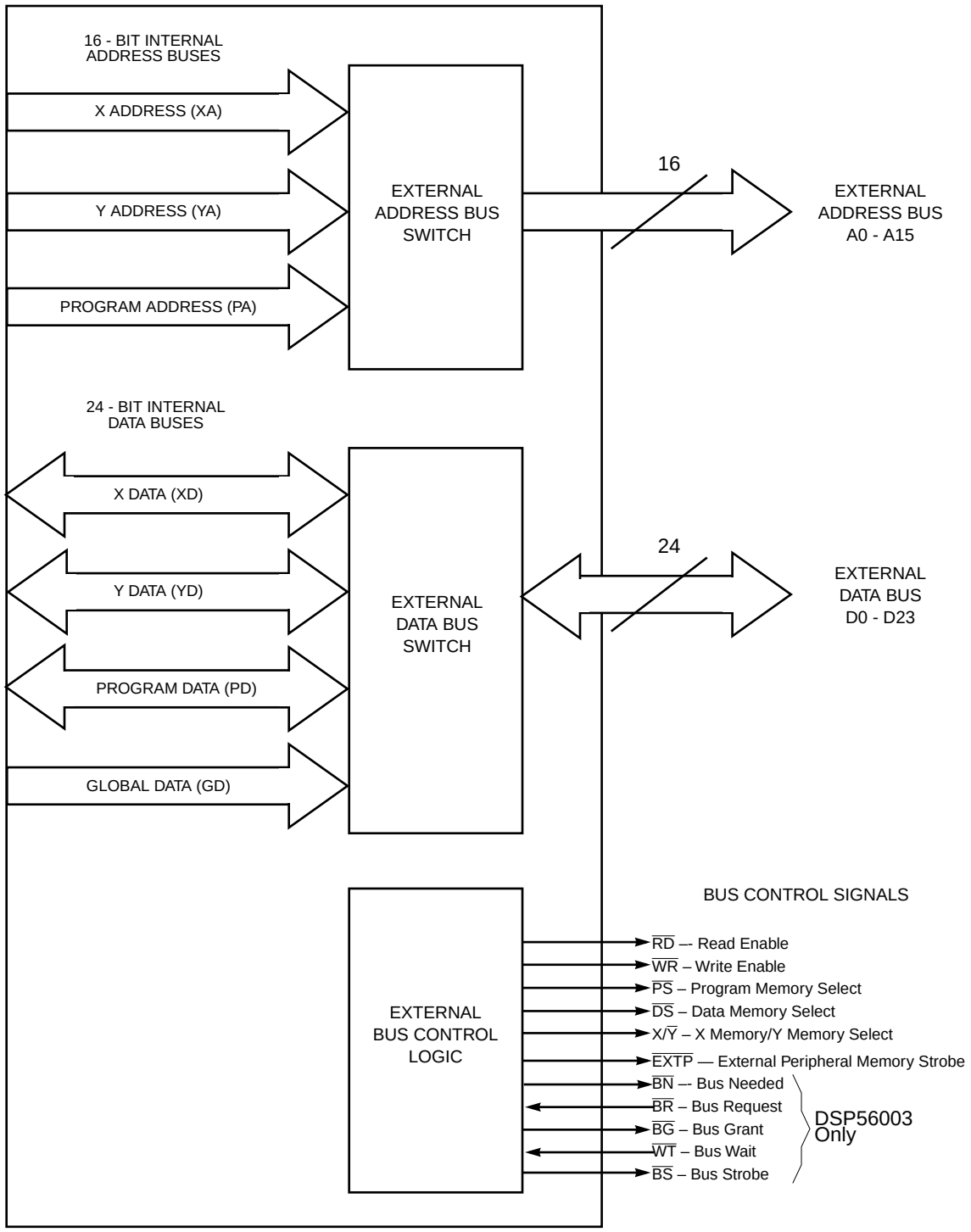


Figure 4-1 External Memory Interface Signals

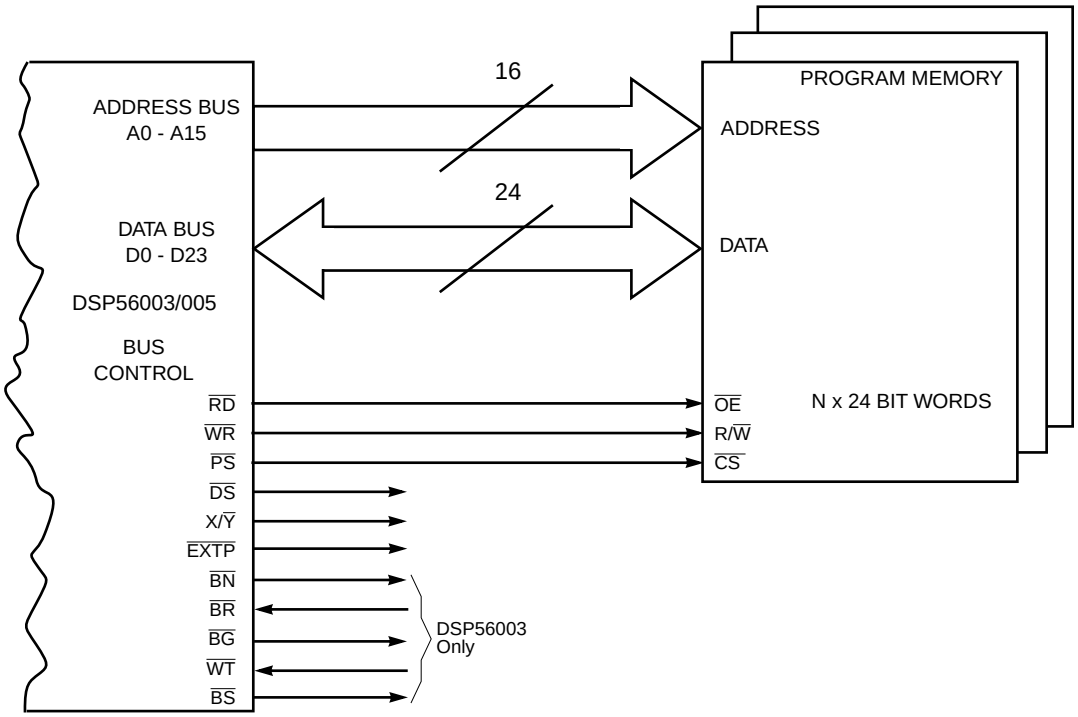


Figure 4-2 External Program Space

The read/write controls can act as decoded read and write controls, or, as seen in Figure 4-2, Figure 4-3, and Figure 4-4, 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 such a way 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 directly addressable memory from 64K words to 192K words total.

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 DSP56003/005 are mutually exclusive, with a guard time between them to avoid an instance where two data buffers are 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 4-2 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.

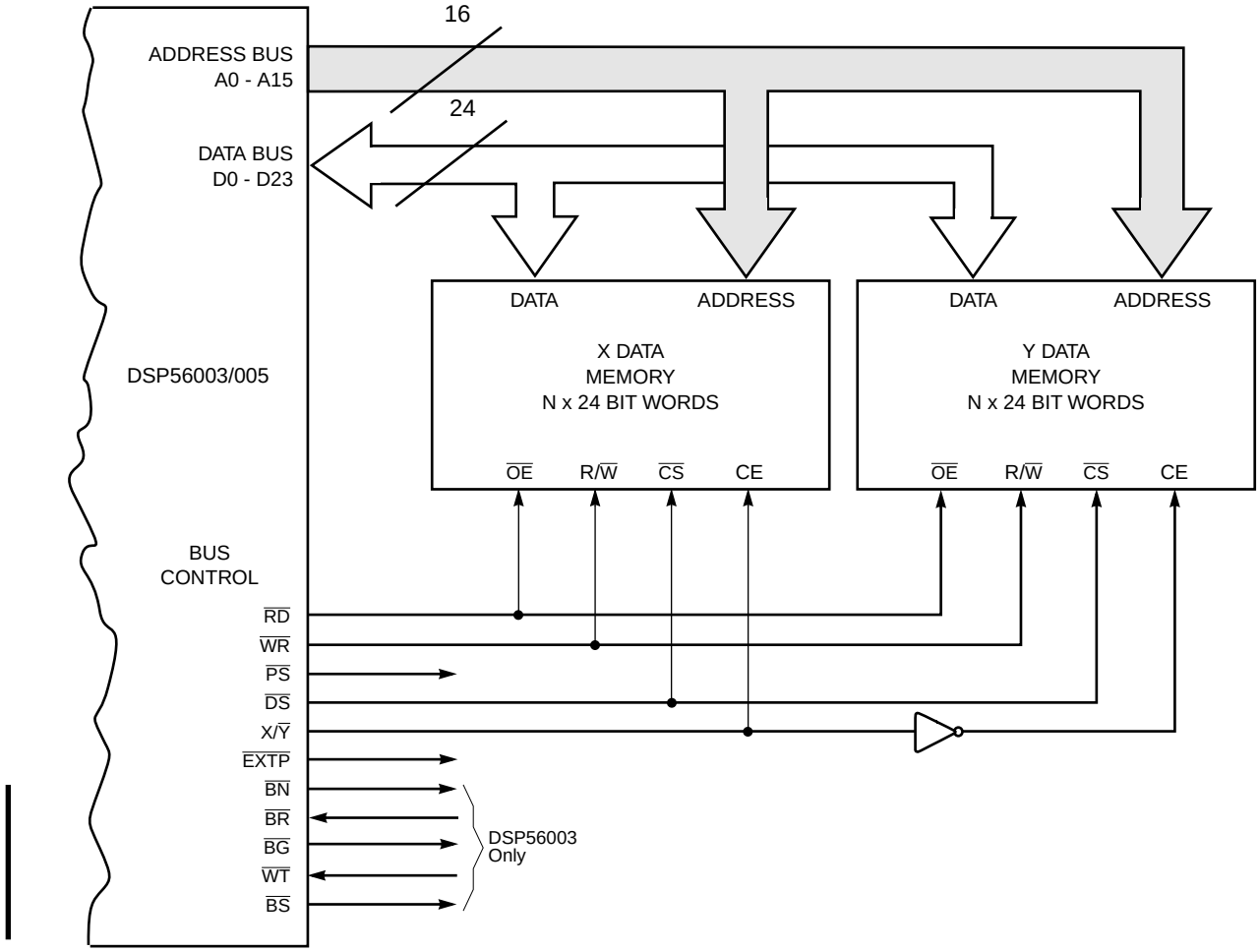


Figure 4-3 External X and Y Data Space

Figure 4-3 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 4-4). Table 4-1 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.

Table 4-1 Program and Data Memory Select Encoding

$\overline{PS}$	$\overline{DS}$	X/$\overline{Y}$	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)

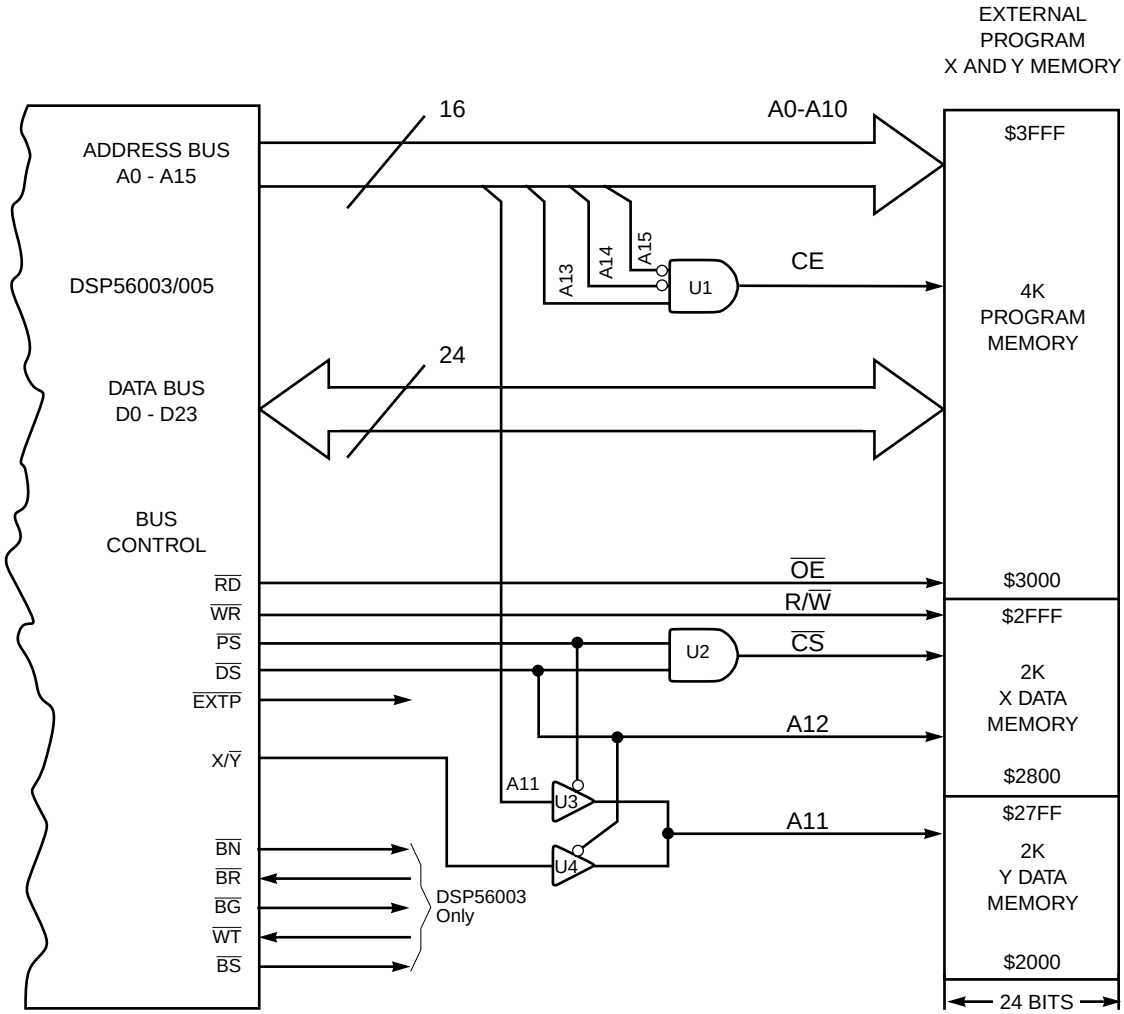


Figure 4-4 Memory Segmentation

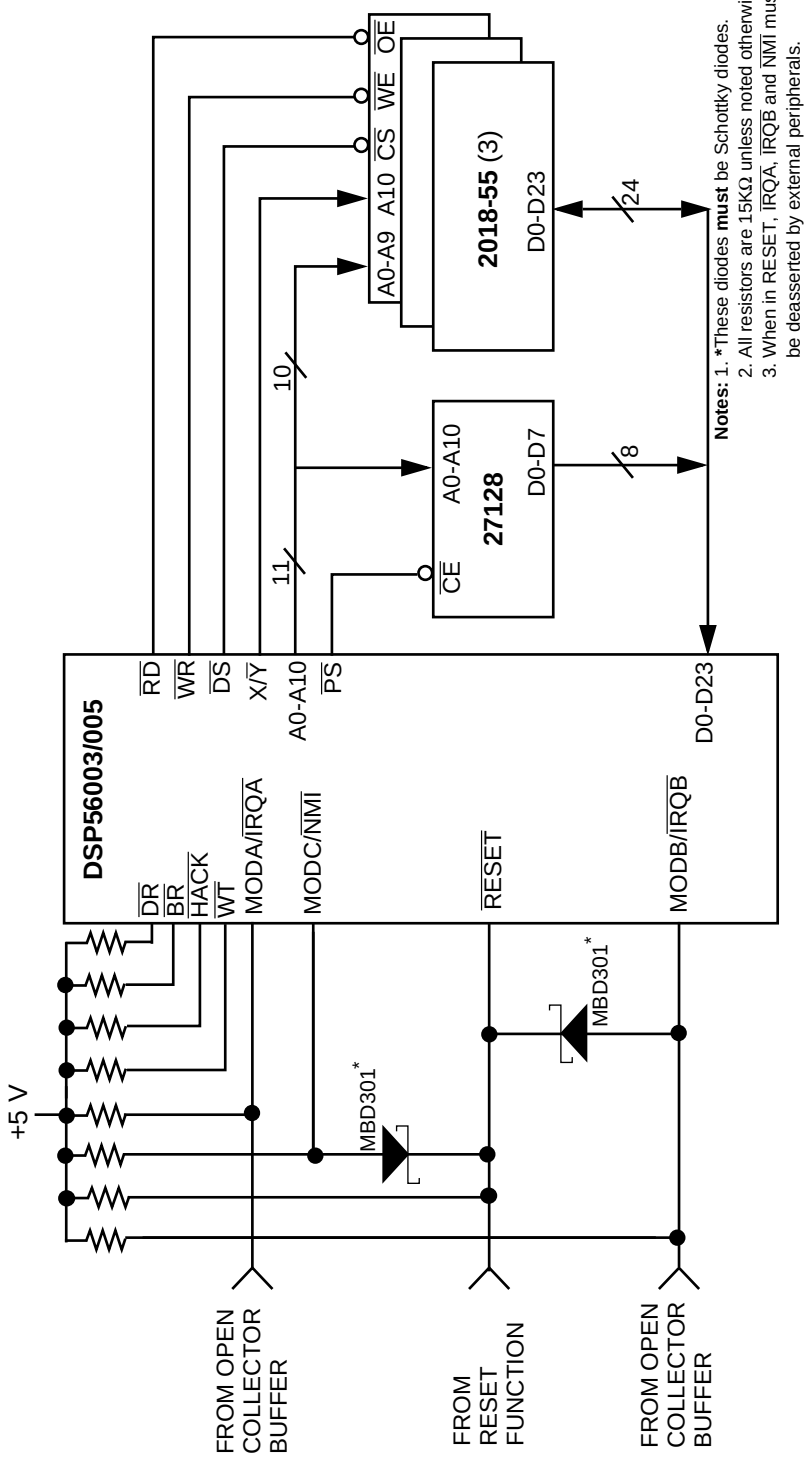


Figure 4-5 External Memory Interface Bootstrap ROM with X and Y RAM

Figure 4-5 shows a system that uses internal program memory loaded from an external ROM during power-up and 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. Many systems use 16 or fewer bits for A/D and D/A conversion and, therefore, they may only need to store 16, 12, or even eight bits of data. The 24/56 bits of internal precision is usually sufficient for intermediate results. This is a cost saving feature which can reduce the number of external memory chips.

4.3 TIMING

The external bus timing is defined by the operation of the address bus, data bus, and bus control pins. Reads or writes by the DSP to the external data bus are synchronous with the DSP clock. The timing A, B, and C relative to the edges of an external clock (see Figure 4-6 and Figure 4-7) are provided in the *DSP56003/005 Data Sheet*. This timing is essential for designing synchronous multiprocessor systems. Figure 4-6 shows the external memory interface timing with no wait states (wait-state control is discussed in Section 4.4). One instruction cycle equals two clock cycles or four clock phases.

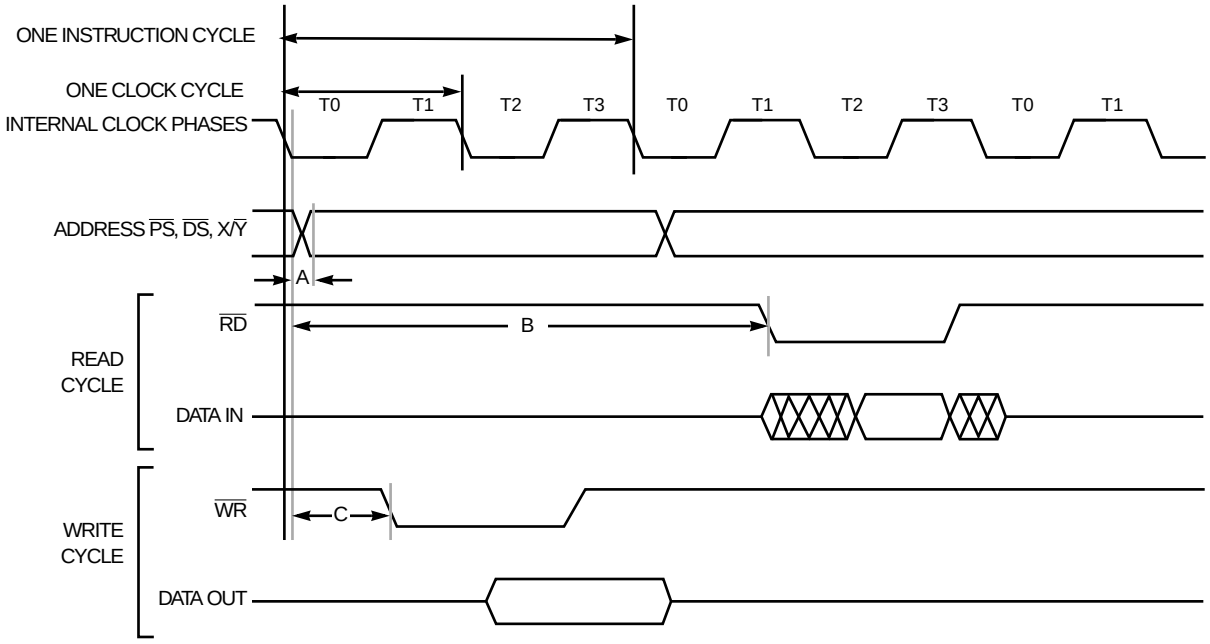


Figure 4-6 External Memory Interface Bus Operation with No Wait States

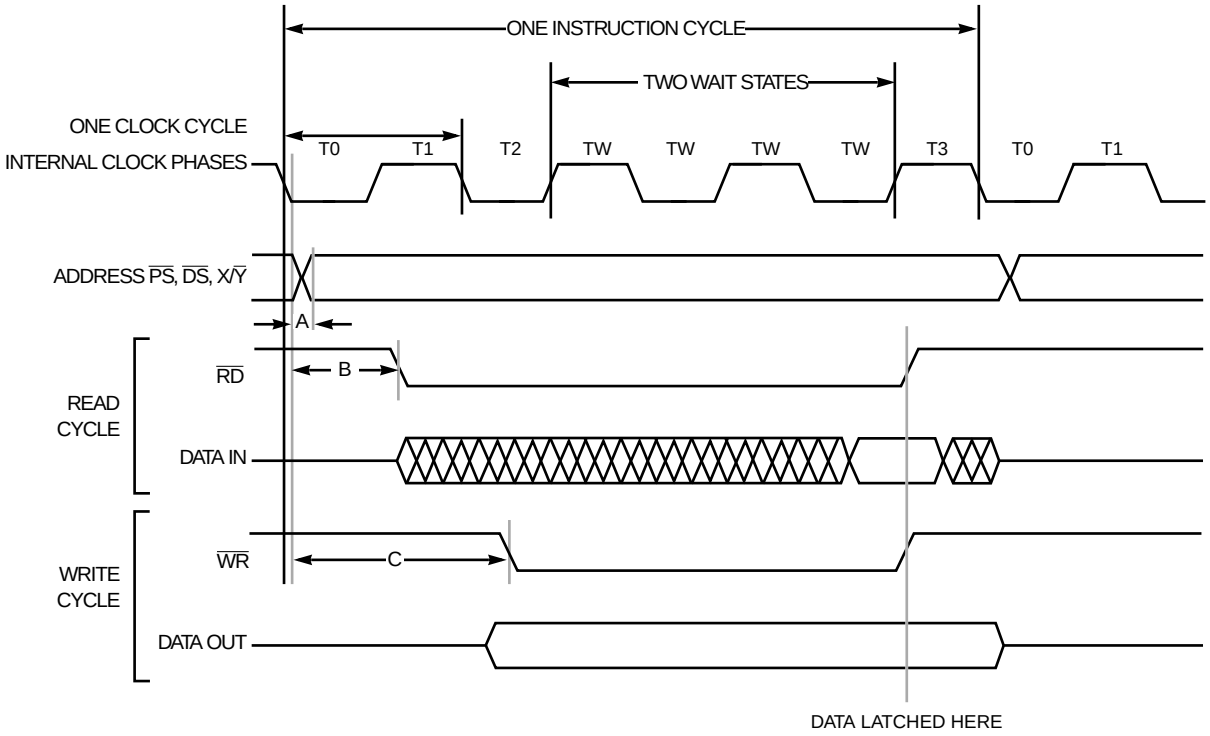


Figure 4-7 External Memory Interface Bus Operation with Two Wait States

The clock phases, which are numbered T0 – T3, are used for timing on the DSP. Figure 4-7 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 equivalent to repeating the T2 and $\overline{T2}$ clock phases. The write signal is also delayed from the 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 *DSP56003/005 Data Sheet* 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 connected 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 deasserted, 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 4-6. If a value W≠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 4-7.

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 dissipation. 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.

4.4 WAIT STATES

The DSP56003/005 features two methods to allow the user to accommodate slow memory by changing the external memory interface 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 ($\overline{BS}$) and bus wait ($\overline{WT}$) facility (DSP56003 only), 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.

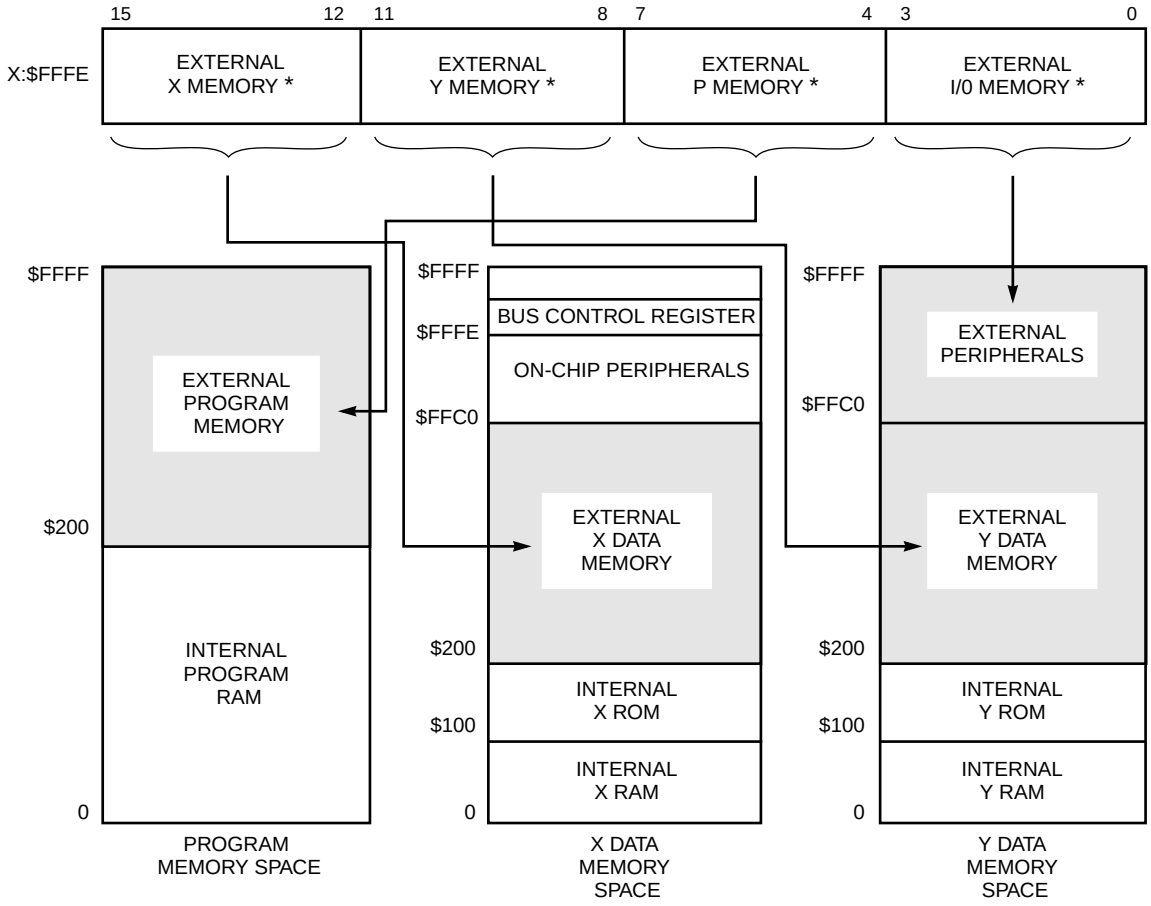
Table 4-2 Wait State Control

BCR Contents	$\overline{WT}$ (DSP56003 only)	Number of Wait States Generated
0	Deasserted	0
0	Asserted — DSP56003 only	2 (minimum)
> 0	Deasserted	Equals value in BCR
> 0	Asserted — DSP56003 only	Minimum equals 2 or value in BCR. Maximum is determined by BCR or $\overline{WT}$, whichever is larger.

4.5 BUS CONTROL REGISTER (BCR)

The BCR determines the expansion bus timing by controlling the timing of the bus interface signals, $\overline{RD}$ and $\overline{WR}$, and the data output lines. It is a memory mapped register located at X:$FFFE. Each of the memory spaces in Figure 4-8 (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., 20 ns for a 50 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).** This allows slow memory to be used for boot strapping. **The BCR needs to be set appropriately for the memory being used or the processor will insert 15 wait states between each external memory fetch and cause the DSP to run slowly.**

Figure 4-8 illustrates which of the four BCR nibbles affect which external memory space. All the internal peripheral devices are memory mapped, and their control registers reside between X:$FFC0 and X:$FFFF.



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

Figure 4-8 Bus Control Register

To load the BCR the way it is shown in Figure 4-9, execute a "MOVEP #$48AD, X:$FFFE" instruction. Or, change the individual bits in one of the four subregisters by using the BSET and BCLR instructions which are detailed in the *DSP56000 Family Manual*, SECTION 6 and APPENDIX A.

Figure 4-9 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. Consult the *DSP56003/005 Data Sheet* for specific power consumption requirements.

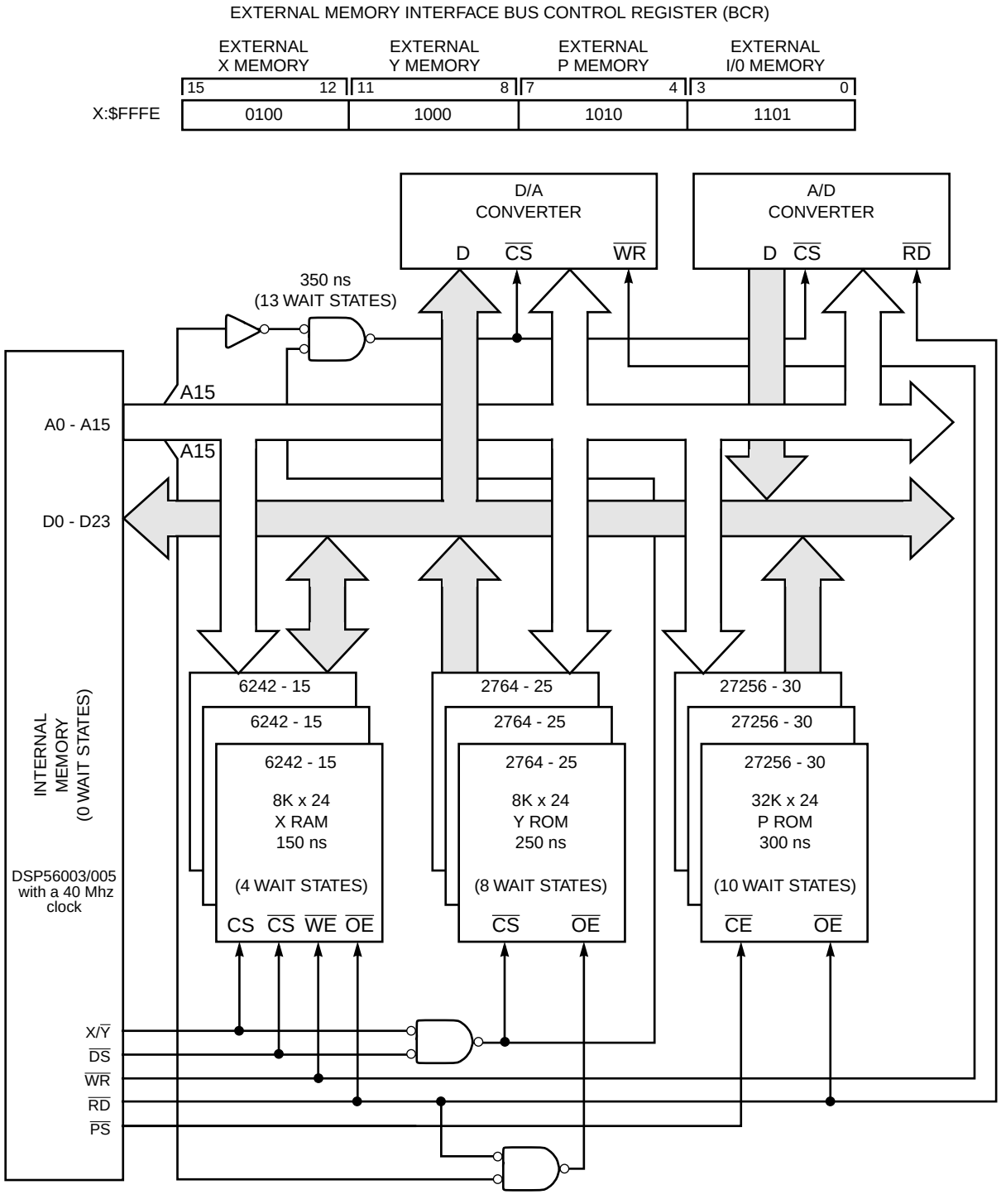


Figure 4-9 Mixed-Speed Expanded System

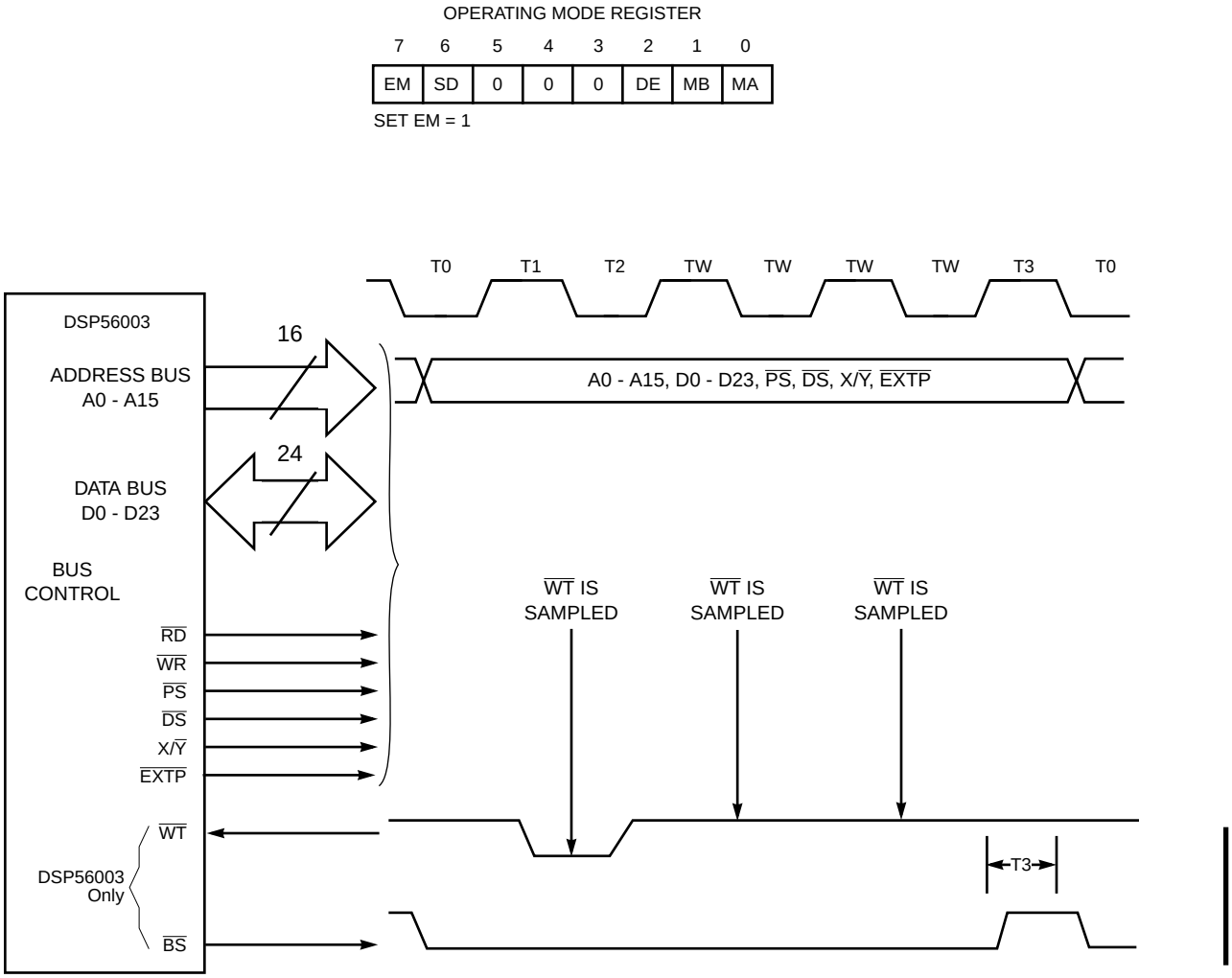


Figure 4-10 Bus Strobe/Wait Sequence — DSP56003 Only

4.6 BUS STROBE AND WAIT PINS — DSP56003 Only

The ability to insert wait states using $\overline{BS}$ and $\overline{WT}$ allows devices with differing timing requirements to reside in the same memory space, allows a bus arbiter to provide a fast multiprocessor bus access, and provides another means of halting the DSP at a known program location with a fast restart.

The timing of the $\overline{BS}$ and $\overline{WT}$ pins is illustrated in Figure 4-10. $\overline{BS}$ is asserted at the same time as the external address lines. $\overline{BS}$ can be used by external wait-state logic to establish the start of an external access. $\overline{BS}$ is deasserted in T3 of each external bus cycle, signaling that the current bus cycle will complete. Since the $\overline{WT}$ signal is internally synchronized, it can be asserted asynchronously with respect to the system clock.

The $\overline{WT}$ signal should only be asserted while $\overline{BS}$ is asserted. Asserting $\overline{WT}$ while $\overline{BS}$ is deasserted will give indeterminate results. However, for the number of inserted wait states to be deterministic, $\overline{WT}$ timing must satisfy setup and hold timing with respect to the negative-going edge of EXTAL. The setup and hold times are provided in the *DSP56003/005 Data Sheet*. The timing of $\overline{WR}$ is controlled by the BCR and is independent of $\overline{WT}$. The minimum number of wait states that can be inserted using the $\overline{WT}$ pin is two. The BCR is still operative when using $\overline{BS}$ and $\overline{WT}$ and defines the minimum number of wait states that are inserted. Table 4-2 summarizes the effect of the BCR and $\overline{WT}$ pin on the number of wait states generated.

4.7 BUS ARBITRATION AND SHARED MEMORY — DSP56003 Only

The DSP56003 has five pins that control the external memory interface. They are bus needed ($\overline{BN}$), bus request ($\overline{BR}$), bus grant ($\overline{BG}$), bus strobe ($\overline{BS}$) and bus wait ($\overline{WT}$) and they are described in Section 2 — DSP56003/005 Pin Descriptions.

The bus control signals provide the means to connect additional bus masters (which may be additional DSPs, microprocessors, direct memory access (DMA) controllers, etc.) to the external memory interface bus. They work together to arbitrate and determine what device gets access to the bus.

If an external device has requested the external bus by asserting the $\overline{BR}$ input, and the DSP has granted the bus by asserting $\overline{BG}$, the DSP will continue to process as long as it requires no external bus accesses itself. If the DSP **does** require an external access but is not the bus master, it will stop processing and remain in wait states until it regains bus ownership. The $\overline{BN}$ pin will be asserted, and an external device may use $\overline{BN}$ to help "arbitrate", or decide when to return bus ownership to the chip.

- Four examples of bus arbitration will be described later in this section:
- bus arbitration using only $\overline{BR}$ and $\overline{BG}$ with internal control
- bus arbitration using $\overline{BN}$, $\overline{BR}$, and $\overline{BG}$ with external control
- bus arbitration using $\overline{BR}$, $\overline{BG}$ and $\overline{WT}$, $\overline{BS}$ with no overhead
- signaling using semaphores.

The $\overline{BR}$ input allows an external device to request and be given control of the external bus while the DSP continues internal operations using internal memory spaces. This independent operation allows a bus controller to arbitrate a multiple bus-master system independent of operation of each DSP. (A bus master can issue 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.)

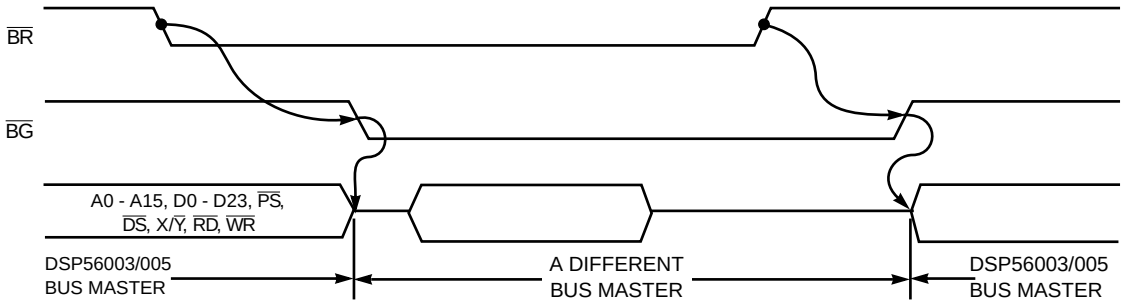


Figure 4-11 Bus Request/Bus Grant Sequence — DSP56003 Only

Before $\overline{BR}$ is asserted, all the external memory interface signals may be driven by the DSP. When $\overline{BR}$ is asserted (see Figure 4-11), the DSP will assert $\overline{BG}$ after the current external access cycle completes and will simultaneously three-state (high-impedance) the external memory interface signals (see the *DSP56003/005 Data Sheet* for exact timing of $\overline{BR}$ and $\overline{BG}$). The bus is then available to whatever external device has bus mastership. The external device will return bus mastership to the DSP by deasserting $\overline{BR}$. After the DSP completes the current cycle (an internally executed instruction with or without wait states), $\overline{BG}$ will be deasserted. When $\overline{BG}$ is deasserted, the A0-A15, $\overline{PS}$, $\overline{DS}$, X/$\overline{Y}$, $\overline{EXTP}$, and $\overline{RD}$, $\overline{WR}$ lines will be driven. However, the data lines will remain in three-state. All signals are now ready for a normal external access.

During the wait state (see SECTION 7 in the *DSP56000 Family Manual*), the $\overline{BR}$ and $\overline{BG}$ circuits remain active. However, the port is inactive - the control signals are deasserted, the data signals are inputs, and the address signals remain as the last address read or written. When $\overline{BR}$ is asserted, all signals are three-stated (high impedance). Table 4-3 shows the status of $\overline{BR}$ and $\overline{BG}$ during the wait state.

Table 4-3 BR and BG During Wait — DSP56003 Only

Signal	Before $\overline{BR}$	While $\overline{BG}$	After $\overline{BR}$	After Return to Normal State	After First

4.7.1 Bus Arbitration Using Only $\overline{BR}$ and $\overline{BG}$ With Internal Control — DSP56003 Only

Perhaps the simplest example of a shared memory system using a DSP56003 is shown in Figure 4-12. The bus arbitration is performed within the DSP#2 by using software. DSP#2 controls all bus operations by using I/O pin OUT2 to three-state its own external memory interface and by never accessing the external memory interface 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 $\overline{BR}$ and $\overline{BG}$ 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 4-13):

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).
3. DSP#2 sets OUT2=1 ($\overline{BR}$#2=1 to let DSP#2 control 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.

4.7.2 Bus Arbitration Using $\overline{BN}$, $\overline{BR}$, and $\overline{BG}$ With External Control — DSP56003 Only

The system shown in Figure 4-14 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. 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 4-14) needs the bus, it will enter the wait state with $\overline{BN}$ asserted. If DSP#1 has highest priority of the pending bus requests, the arbitration logic grants the bus to DSP#1 by asserting E1 (E2 for DSP#2; E3 for DSP#3) to let the DSP know that it can have the bus. DSP#1 will then deassert $\overline{BG}$ to tell the arbiter it has taken control of the bus. When the DSP no longer needs to make an external access it will deassert $\overline{BN}$ and the arbiter deasserts E1, after which the DSP deasserts $\overline{BG}$.

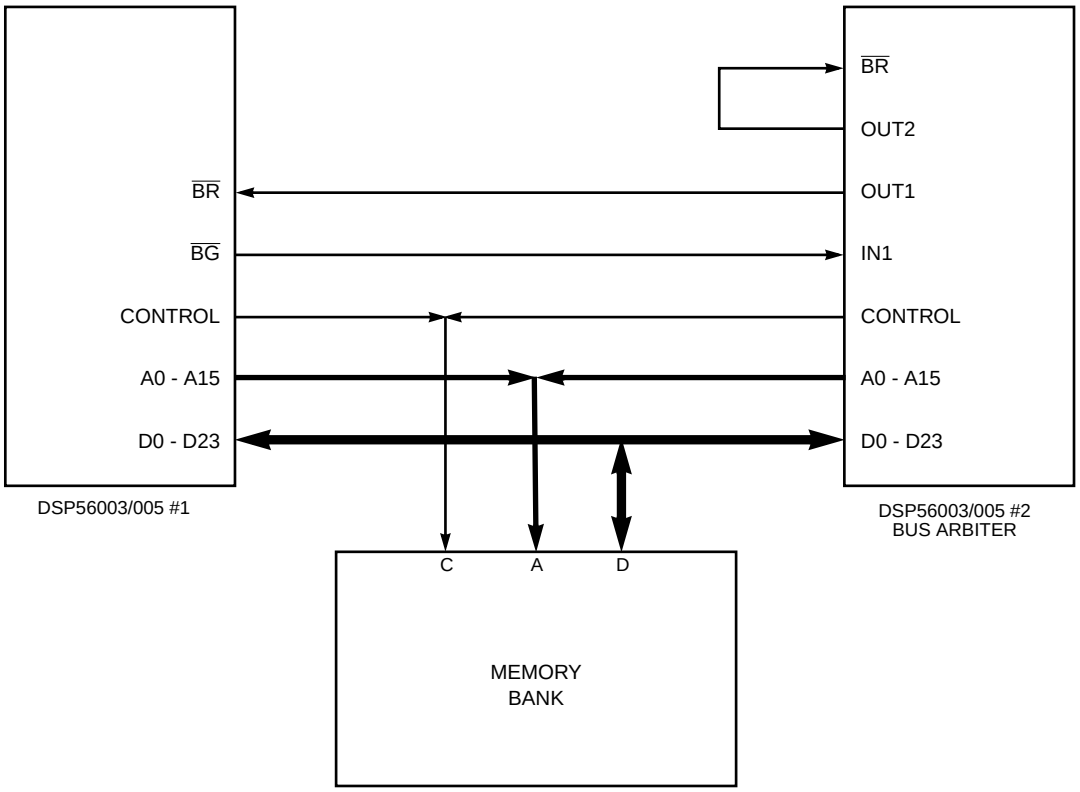


Figure 4-12 Bus Arbitration Using Only $\overline{BR}$ and $\overline{BG}$ with Internal Control — DSP56003 Only

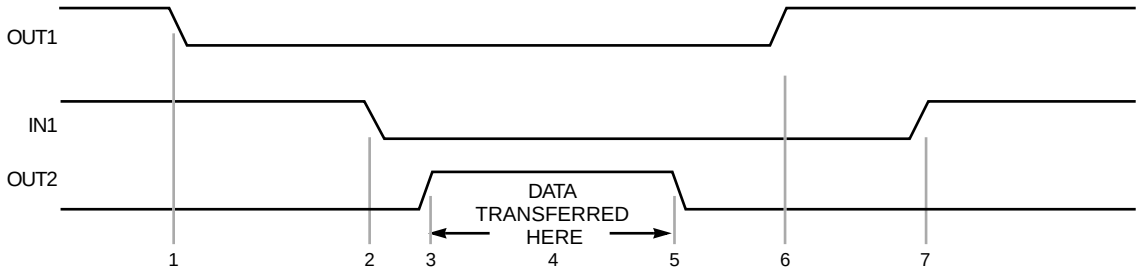


Figure 4-13 Two DSPs with External Bus Arbitration Timing

4.7.3 Arbitration Using $\overline{BR}$ and $\overline{BG}$, and $\overline{WT}$ and $\overline{BS}$ With No Overhead —

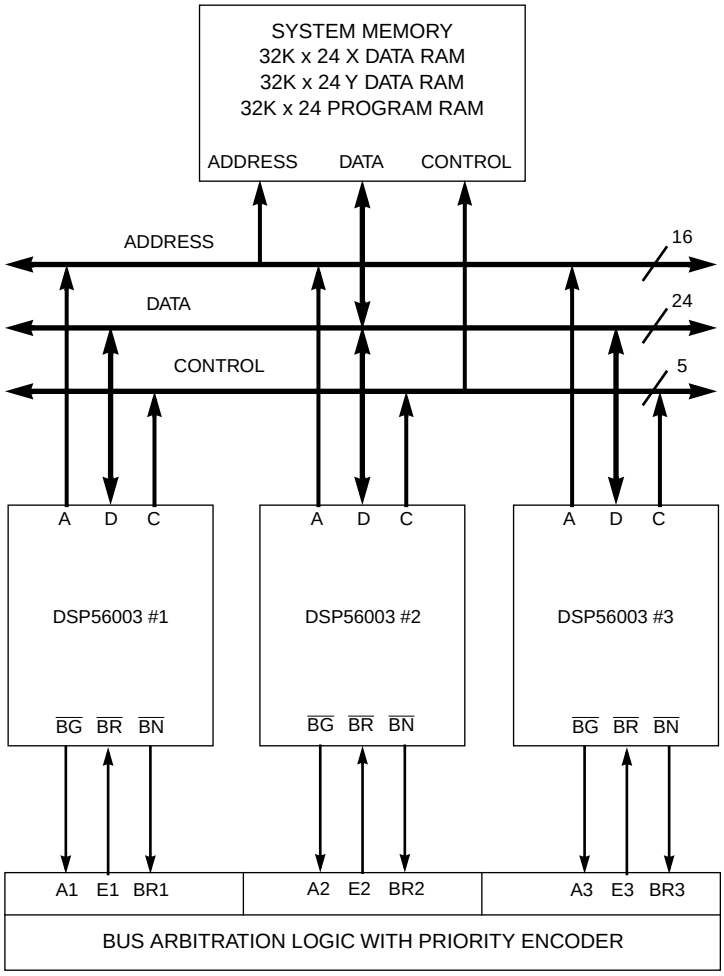


Figure 4-14 Bus Arbitration Using $\overline{BN}$, $\overline{BR}$, and $\overline{BG}$ with External Control — DSP56003 Only

DSP56003 Only

By using the circuit shown in Figure 4-15, two DSPs can share memory with hardware arbitration that requires no software on the part of the DSPs. The protocol for bus arbitration in Figure 4-15 is as follows:

At RESET assume DSP#1 is not making external accesses so that $\overline{BR}$ of DSP#2 is deasserted. Hence, $\overline{BG}$ of DSP#2 is deasserted, which three-states the buffers, giving DSP#2 control of the memory.

When DSP#1 wants control of the memory the following steps are performed (see Figure 4-16):

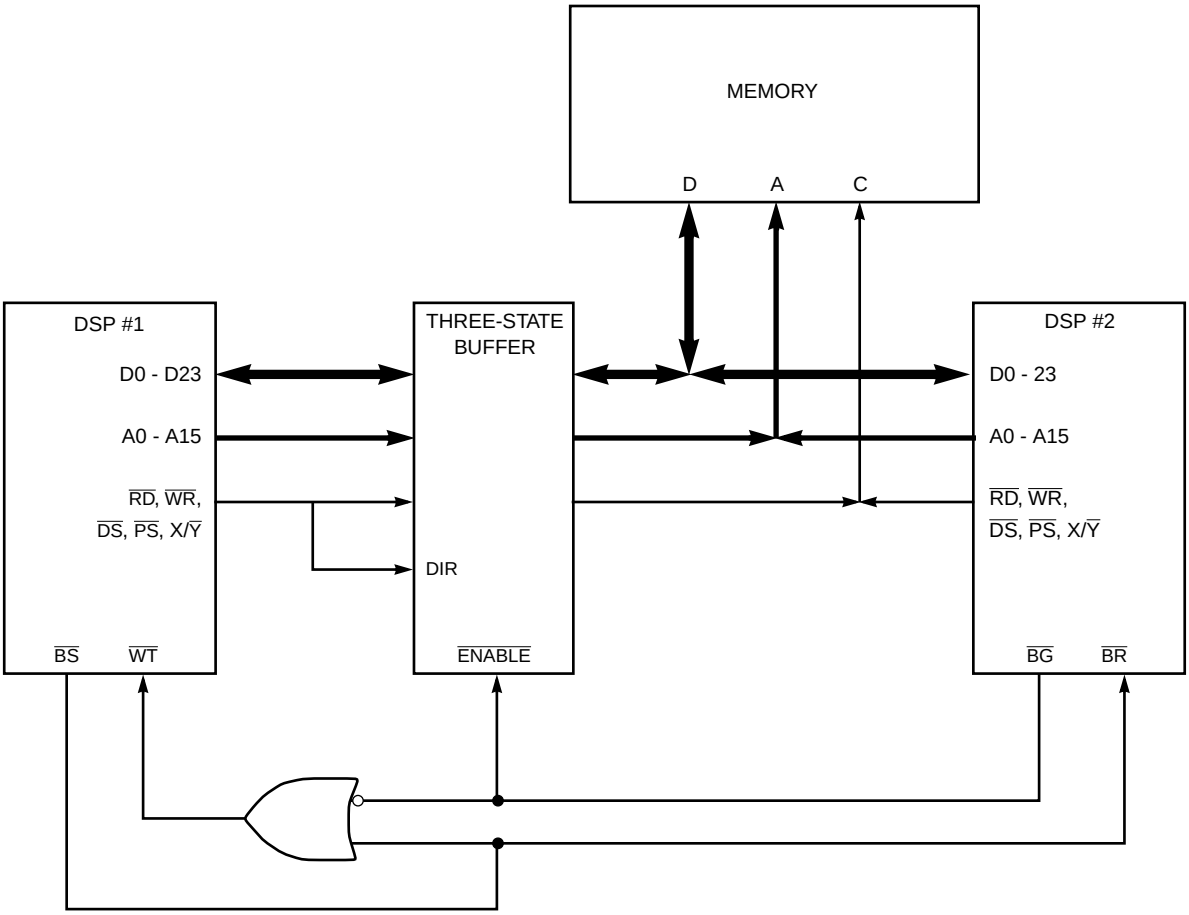


Figure 4-15 Bus Arbitration Using $\overline{BR}$ and $\overline{BG}$, and $\overline{WT}$ and $\overline{BS}$ with No Overhead — DSP56003 Only

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.

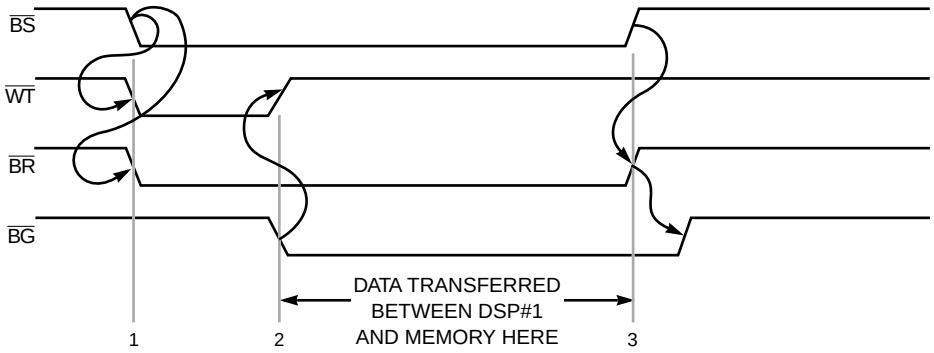


Figure 4-16 Two DSPs with External Bus Arbitration Timing — DSP56003 Only

4.7.4 Signaling Using Semaphores

Figure 4-17 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
4. both proceed to use the data on the assumption that the data cannot be changed by another processor

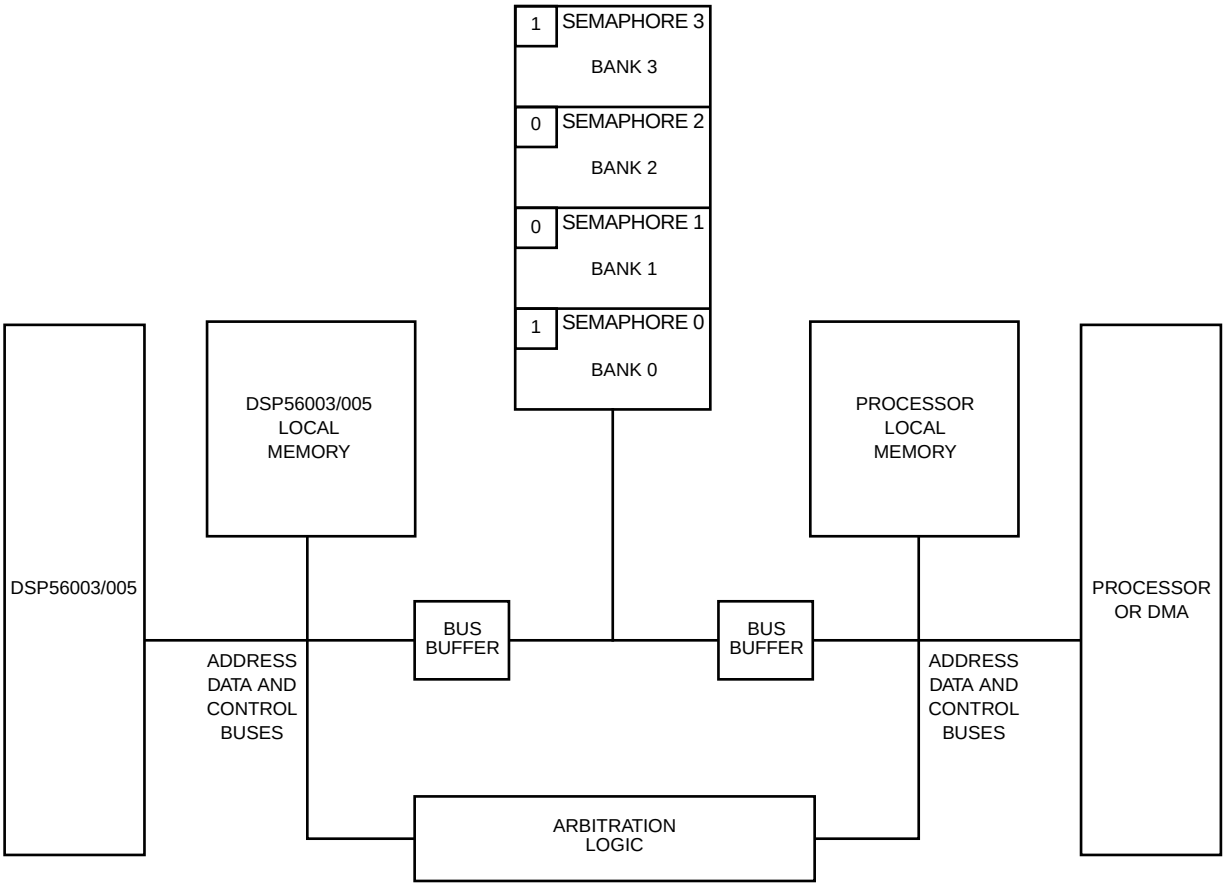


Figure 4-17 Signaling Using Semaphores

The solution is that the DSP56K processor series has a group of instructions designed specifically to prevent this problem. They perform an indivisible read-modify-write operation and do not release the bus between the read and write (specifically, A0–A15, $\overline{DS}$, $\overline{PS}$, and X/$\overline{Y}$ do not change state). **Using a read-modify-write operation 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). (They are discussed in detail in the *DSP56000 Family Manual*.) 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 must wait to access the memory block.