Welcome to E-XFL.COM

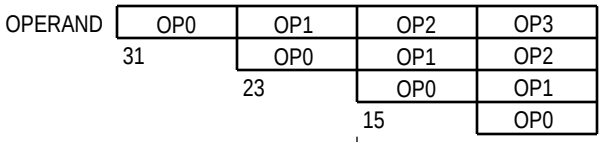
Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Active
Core Processor	CPU32
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	16MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	3.3V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68340ag16ve



Case	Transfer Case	SIZ1	SIZ0	A0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	Data Bus D15 D8	D7 D0
(a)	Byte to Byte	0	1	X	1	0	OP0	(OP0)
(b)	Byte to Word (Even)	0	1	0	0	X	OP0	(OP0)
(c)	Byte to Word (Odd)	0	1	1	0	X	(OP0)	OP0
(d)	Word to Byte (Aligned)	1	0	0	1	0	OP0	(OP1)
(e)	Word to Word (Aligned)	1	0	0	0	X	OP0	OP1
(f)	Long Word to Byte (Aligned)	0	0	0	1	0	OP0	(OP1)
(g)	Long Word to Word (Aligned)	0	0	0	0	X	OP0	OP1

NOTES:
1. Operands in parentheses are ignored by the MC68340 during read cycles.
2. A 3-byte to byte transfer does occur as the second byte transfer of a long-word to byte port transfer.

Figure 3-2. MC68340 Interface to Various Port Sizes

3.2.2 Misaligned Operands

In this architecture, the basic operand size is 16 bits. Operand misalignment refers to whether an operand is aligned on a word boundary or overlaps the word boundary, determined by address line A0. When A0 is low, the address is even and is a word and byte boundary. When A0 is high, the address is odd and is a byte boundary only. A byte operand is properly aligned at any address; a word or long-word operand is misaligned at an odd address.

At most, each bus cycle can transfer a word of data aligned on a word boundary. If the MC68340 transfers a long-word operand over a 16-bit port, the most significant operand word is transferred on the first bus cycle, and the least significant operand word is transferred on a following bus cycle.

The CPU32 restricts all operands (both data and instructions) to be aligned. That is, word and long-word operands must be located on a word or long-word boundary, respectively. The only type of transfer that can be performed to an odd address is a single-byte transfer, referred to as an odd-byte transfer. If a misaligned access is attempted, the CPU32 generates an address error exception, and enters exception processing. Refer to **Section 5 CPU32** for more information on exception processing.

3.2.3 Operand Transfer Cases

The following cases are examples of the allowable alignments of operands to ports.

3.2.3.1 BYTE OPERAND TO 8-BIT PORT, ODD OR EVEN (A0 = X). The MC68340 drives the address bus with the desired address and the SIZx pins to indicate a single-byte operand.

State 0—The MC68340 asserts RMC in S0 to identify a read-modify-write cycle. The MC68340 places a valid address on A31–A0 and valid function codes on FC3–FC0. The function codes select the address space for the operation. SIZ1/SIZ0 become valid in S0 to indicate the operand size. The MC68340 drives R/W high for the read cycle.

State 1—One-half clock later during S1, the MC68340 asserts AS indicating a valid address on the address bus. The MC68340 also asserts DS during S1.

State 2—The selected device uses R/W, SIZ1/SIZ0, A0, and DS to place information on the data bus. Either or both of the bytes (D15–D8 and D7–D0) are selected by SIZ1/SIZ0 and A0. Concurrently, the selected device may assert DSACK≈.

State 3—As long as at least one of the DSACK≈ signals is recognized by the end of S2 (meeting the asynchronous input setup time requirement), data is latched on the next falling edge of the clock, and the cycle terminates. If DSACK≈ is not recognized by the start of S3, the MC68340 inserts wait states instead of proceeding to S4 and S5. To ensure that wait states are inserted, both DSACK1 and DSACK0 must remain negated throughout the asynchronous input setup and hold times around the end of S2. If wait states are added, the MC68340 continues to sample the DSACK≈ signals on the falling edges of the clock until one is recognized.

State 4—At the end of S4, the MC68340 latches the incoming data.

State 5—The MC68340 negates AS and DS during S5. If more than one read cycle is required to read in the operand(s), S0–S5 are repeated for each read cycle. When finished reading, the MC68340 holds the address, R/W, and FC3–FC0 valid in preparation for the write portion of the cycle. The external device keeps its data and DSACK≈ signals asserted until it detects the negation of AS or DS (whichever it detects first). The device must remove the data and negate DSACK≈ within approximately one clock period after sensing the negation of AS or DS. DSACK≈ signals that remain asserted beyond this limit may be prematurely detected for the next portion of the operation.

Idle States—The MC68340 does not assert any new control signals during the idle states, but it may internally begin the modify portion of the cycle at this time. S0–S5 are omitted if no write cycle is required. If a write cycle is required, R/W remains in the read mode until S0 to prevent bus conflicts with the preceding read portion of the cycle; the data bus is not driven until S2.

State 0—The MC68340 drives R/W low for a write cycle. Depending on the write operation to be performed, the address lines may change during S0.

State 1—In S1, the MC68340 asserts AS, indicating a valid address on the address bus.

State 2—During S2, the MC68340 places the data to be written onto D15–D0.

State 3—The MC68340 asserts DS during S3, indicating stable data on the data bus. As long as at least one of the DSACK≈ signals is recognized by the end of S2 (meeting the asynchronous input setup time requirement), the cycle terminates one clock later. If DSACK≈ is not recognized by the start of S3, the MC68340 inserts wait states instead of

EXAMPLE B: A system uses error detection and correction on RAM contents. The designer may:

1. Delay DSACK≈ until data is verified and assert BERR and HALT simultaneously to indicate to the MC68340 to automatically retry the error cycle (case 5), or if data is valid, assert DSACK≈ (case 1).
2. Delay DSACK≈ until data is verified and assert BERR with or without DSACK≈ if data is in error (case 3). This initiates exception processing for software handling of the condition.
3. Return DSACK≈ prior to data verification; if data is invalid, BERR is asserted on the next clock cycle (case 4). This initiates exception processing for software handling of the condition.
4. Return DSACK≈ prior to data verification; if data is invalid, assert BERR and HALT on the next clock cycle (case 6). The memory controller can then correct the RAM prior to or during the automatic retry.

Table 3-4. DSACK≈, BERR, and HALT Assertion Results

Case Num	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	DSACK≈ BERR HALT	A NA NA	S NA X	Normal cycle terminate and continue.
2	DSACK≈ BERR HALT	A NA A/S	S NA S	Normal cycle terminate and halt; continue when HALT negated.
3	DSACK≈ BERR HALT	NA/A A NA	X S X	Terminate and take bus error exception, possibly deferred.
4	DSACK≈ BERR HALT	A NA NA	X A NA	Terminate and take bus error exception, possibly deferred.
5	DSACK≈ BERR HALT	NA/A A A/S	X S S	Terminate and retry when HALT negated.
6	DSACK≈ BERR HALT	A NA NA	X A A	Terminate and retry when HALT negated.

NOTES:
N — Number of the current even bus state (e.g., S2, S4, etc.)
A — Signal is asserted in this bus state
NA — Signal is not asserted in this state
X — Don't care
S — Signal was asserted in previous state and remains asserted in this state

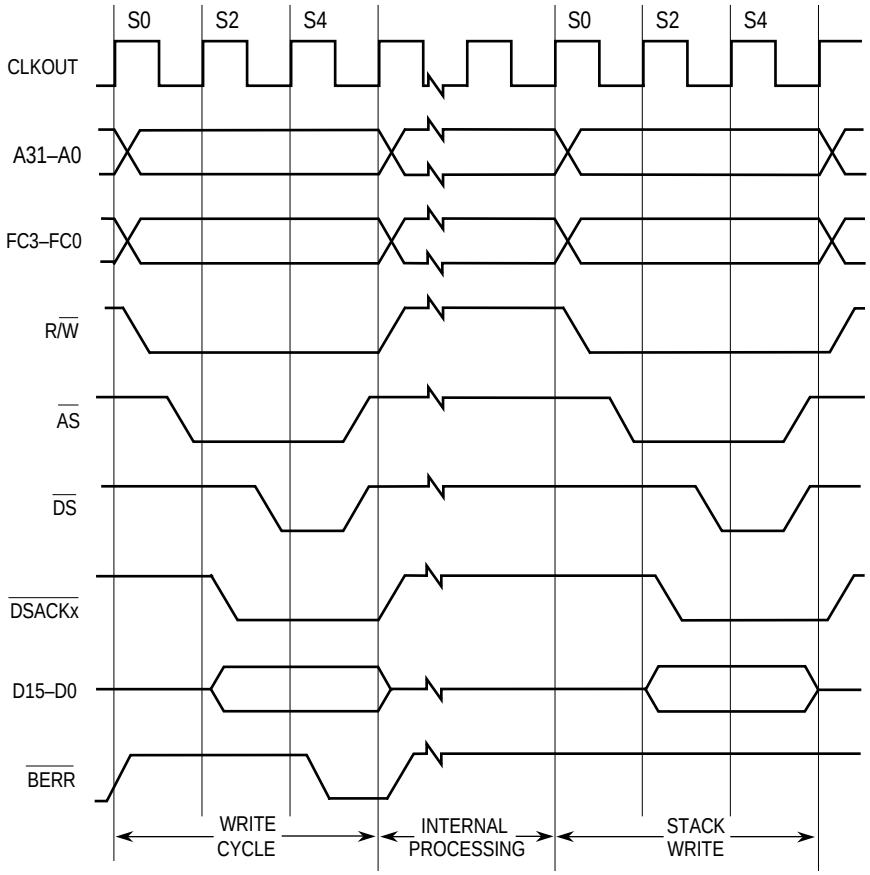


Figure 3-18. Late Bus Error with DSACK≈

3.5.2 Retry Operation

When both BERR and HALT are asserted by an external device during a bus cycle, the MC68340 enters the retry sequence shown in Figure 3-19. A delayed retry, which is similar to the delayed BERR signal described previously, can also occur (see Figure 3-20). The MC68340 terminates the bus cycle, places the control signals in their inactive state, and does not begin another bus cycle until the BERR and HALT signals are negated by external logic. After a synchronization delay, the MC68340 retries the previous cycle using the same access information (address, function code, size, etc.). BERR should be negated before S2 of the retried cycle to ensure correct operation of the retried cycle.

To use an external clock source (see Figure 4-6), the operating clock frequency can be driven directly into the EXTAL pin (the XTAL pin must be left floating for this case). This approach results in a system clock and CLKOUT that are the same as the input signal frequency, but not tightly coupled to it. To enable this mode, MODCK must be held low during reset, and V_{CCSYN} held at 0 V while the chip is in operation.

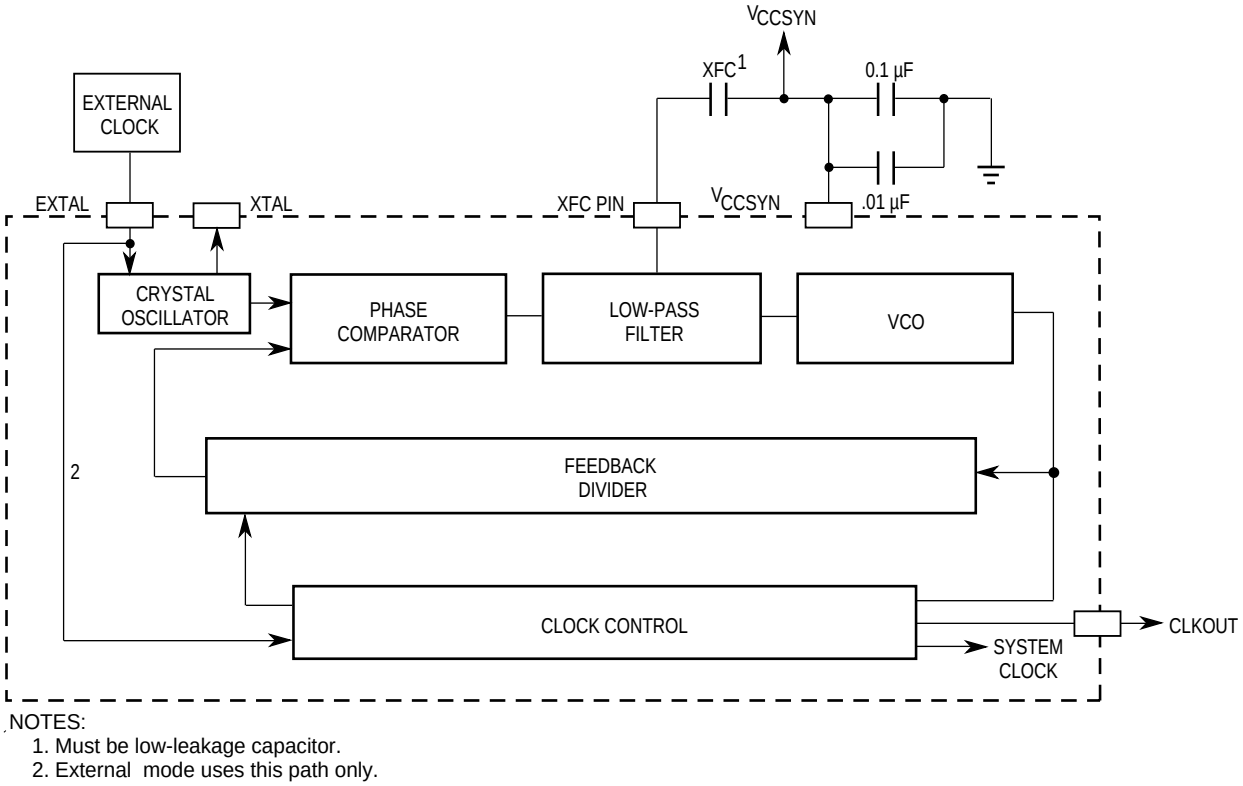


Figure 4-6. Clock Block Diagram for External Oscillator Operation

Alternatively, an external clock signal can be directly driven into EXTAL (with XTAL left floating) using the on-chip PLL. This configuration results in an internal clock and CLKOUT signal of the same frequency as the input signal, with a tight skew between the external clock and the internal clock and CLKOUT signals. To enable this mode, MODCK must be held low during reset, and V_{CCSYN} should be connected to a quiet 5-V source.

If an input signal loss for either of the clock modes utilizing the PLL occurs, chip operation can continue in limp mode with the VCO running at approximately one-half the operating speed (affected by the value of the X-bit in the SYNCR), using an internal voltage reference. The SLIMP bit in the SYNCR indicates that a loss of input signal reference has been detected. The RSTEN bit in the SYNCR controls whether an input signal loss causes a system reset or causes the device to operate in limp mode. The SLOCK bit in the SYNCR indicates when the VCO has locked onto the desired frequency or if an external clock is being used.

4.2.3.1 PHASE COMPARATOR AND FILTER. The phase comparator takes the output of the frequency divider and compares it to an external input signal reference. The result of

Besides the operation code, which specifies the function to be performed, an instruction defines the location of every operand for the function. Instructions specify an operand location in one of three ways:

- Register Specification — A register field of the instruction contains the number of the register.
- Effective Address — An effective address field of the instruction contains address mode information.
- Implicit Reference — The definition of an instruction implies the use of specific registers.

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register is an address or data register and how it is to be used. The M68000PM/AD, *M68000 Family Programmer's Reference Manual*, contains detailed register information.

Except where noted, the following notation is used in this section:

Data	Immediate data from an instruction
Destination	Destination contents
Source	Source contents
Vector	Location of exception vector
An	Any address register (A7–A0)
Ax, Ay	Address registers used in computation
Dn	Any data register (D7–D0)
Rc	Control register (VBR, SFC, DFC)
Rn	Any address or data register
Dh, Dl	Data registers, high- and low-order 32 bits of product
Dr, Dq	Data registers, division remainder, division quotient
Dx, Dy	Data registers, used in computation
Dym, Dyn	Data registers, table interpolation values
Xn	Index register
[An]	Address extension
cc	Condition code
d#	Displacement Example: d_{16} is a 16-bit displacement
⟨ea⟩	Effective address
#⟨data⟩	Immediate data; a literal integer
label	Assembly program label
list	List of registers Example: D3–D0
[...]	Bits of an operand Examples: [7] is bit 7; [31:24] are bits 31–24

5.5.4 CPU32 Stack Frames

The CPU32 generates three different stack frames: four-word frames, six-word frames, and twelve-word bus error frames.

5.5.4.1 FOUR-WORD STACK FRAME. This stack frame is created by interrupt, format error, TRAP #n, illegal instruction, A-line and F-line emulator trap, and privilege violation exceptions. Depending on the exception type, the PC value is either the address of the next instruction to be executed or the address of the instruction that caused the exception (see Figure 5-12).

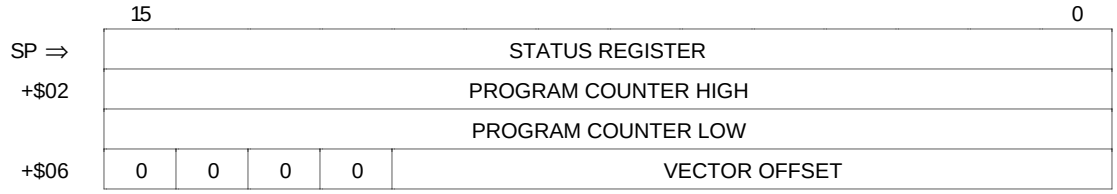


Figure 5-12. Format $0—Four-Word Stack Frame

5.5.4.2 SIX-WORD STACK FRAME. This stack frame (see Figure 5-13) is created by instruction-related traps, which include CHK, CHK2, TRAPcc, TRAPV, and divide-by-zero, and by trace exceptions. The faulted instruction PC value is the address of the instruction that caused the exception. The next PC value (the address to which RTE returns) is the address of the next instruction to be executed.

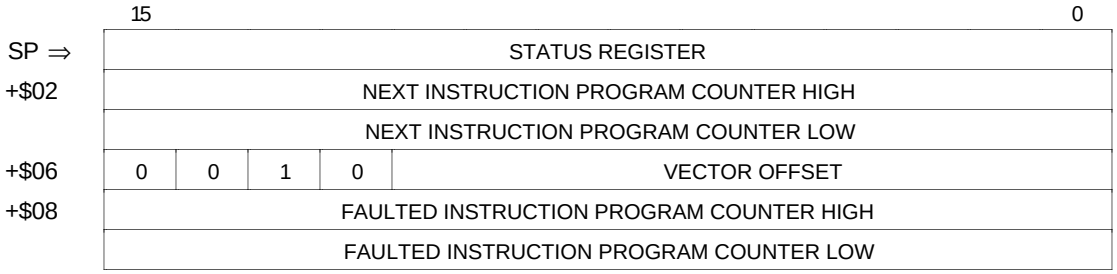


Figure 5-13. Format $2—Six-Word Stack Frame

Hardware breakpoints also utilize this format. The faulted instruction PC value is the address of the instruction executing when the breakpoint was sensed. Usually this is the address of the instruction that caused the breakpoint, but, because released writes can overlap following instructions, the faulted instruction PC may point to an instruction following the instruction that caused the breakpoint. The address to which RTE returns is the address of the next instruction to be executed.

5.5.4.3 BUS ERROR STACK FRAME. This stack frame is created when a bus cycle fault is detected. The CPU32 bus error stack frame differs significantly from the equivalent stack frames of other M68000 Family members. The only internal machine state required in the CPU32 stack frame is the bus controller state at the time of the error and a single register.

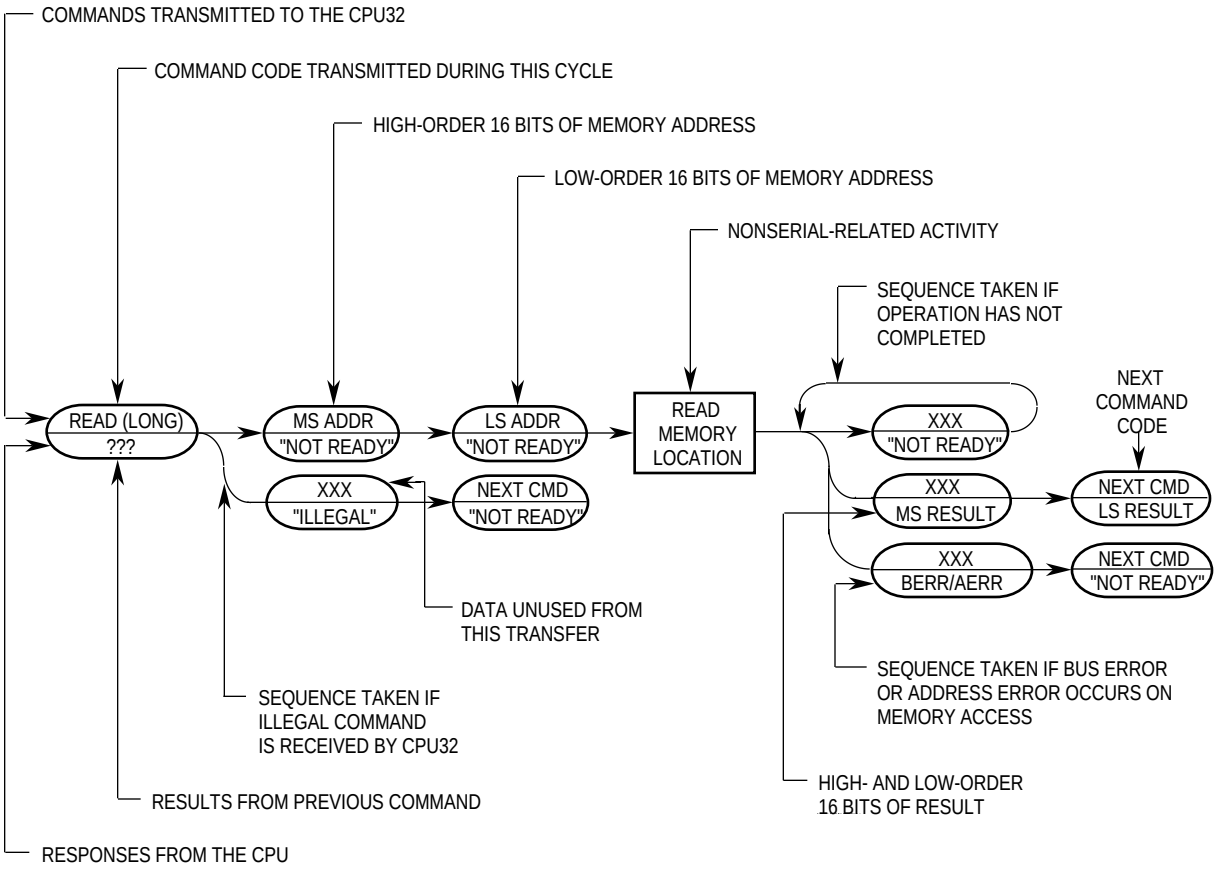


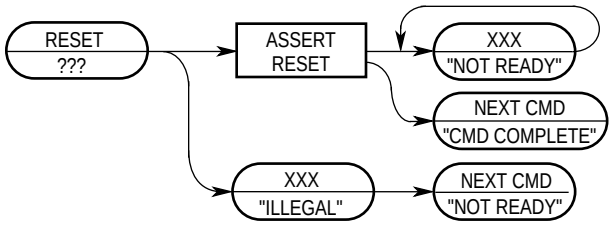
Figure 5-27. Command-Sequence Diagram

5.6.2.8.3 Command Set Summary. The BDM command set is summarized in Table 5-23. Subsequent paragraphs contain detailed descriptions of each command.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

None

Result Data:

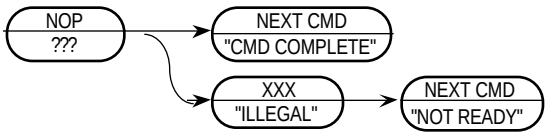
The “command complete” response ($0FFFF) is loaded into the serial shifter after negation of RESET.

5.6.2.8.15 No Operation (NOP). NOP performs no operation and may be used as a null command where required.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

None

Result Data:

The “command complete” response ($0FFFF) is returned during the next shift operation.

5.7.3.7 BINARY-CODED DECIMAL AND EXTENDED INSTRUCTIONS. The BCD and extended instruction table indicates the number of clock periods needed for the processor to perform the specified operation using the specified addressing mode. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles
ABCD	Dn, Dm	2	0	4(0/1/0)
ABCD	–(An), –(Am)	2	2	12(2/1/1)
SBCD	Dn, Dm	2	0	4(0/1/0)
SBCD	–(An), –(Am)	2	2	12(2/1/1)
ADDX	Dn, Dm	0	0	2(0/1/0)
ADDX	–(An), –(Am)	2	2	10(2/1/1)
SUBX	Dn, Dm	0	0	2(0/1/0)
SUBX	–(An), –(Am)	2	2	10(2/1/1)
CMPM	(An)+, (Am)+	1	0	8(2/1/0)

SECTION 6
DMA CONTROLLER MODULE

The direct memory access (DMA) controller module provides for high-speed transfer capability to/from an external peripheral or for memory-to-memory data transfer. The DMA module, shown in Figure 6-1, provides two channels that allow byte, word, or long-word operand transfers. These transfers can be either single or dual address and to either on- or off-chip devices. The DMA contains the following features:

- Two, Independent, Fully Programmable DMA Channels
- Single-Address Transfers with 32-Bit Address and 32-Bit Data Capability
- Dual-Address Transfers with 32-Bit Address and 16-Bit Data Capability
- Two 32-Bit Transfer Counters
- Four 32-Bit Address Pointers That Can Increment or Remain Constant
- Operand Packing and Unpacking for Dual-Address Transfers
- Supports All Bus-Termination Modes
- Provides Two-Clock-Cycle Internal Module Access
- Provides Two-Clock-Cycle External Access Using MC68340 Chip Selects
- Provides Full DMA Handshake for Burst Transfers and Cycle Steal

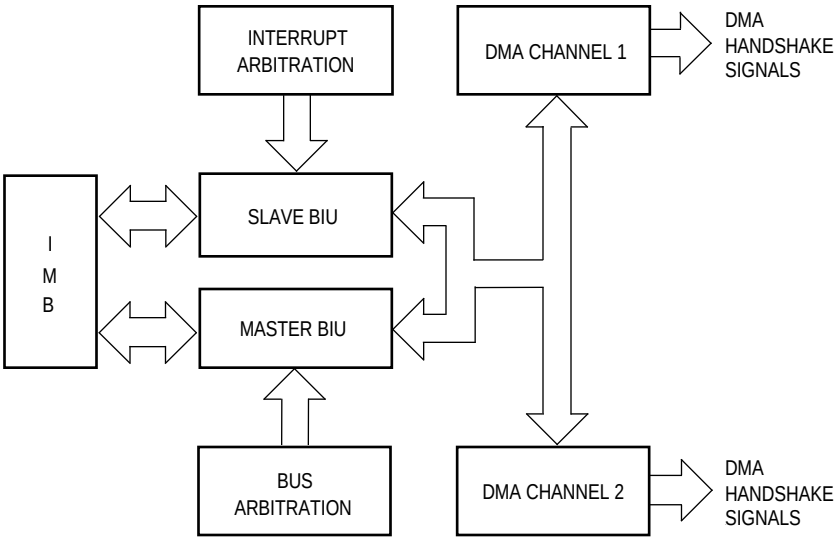
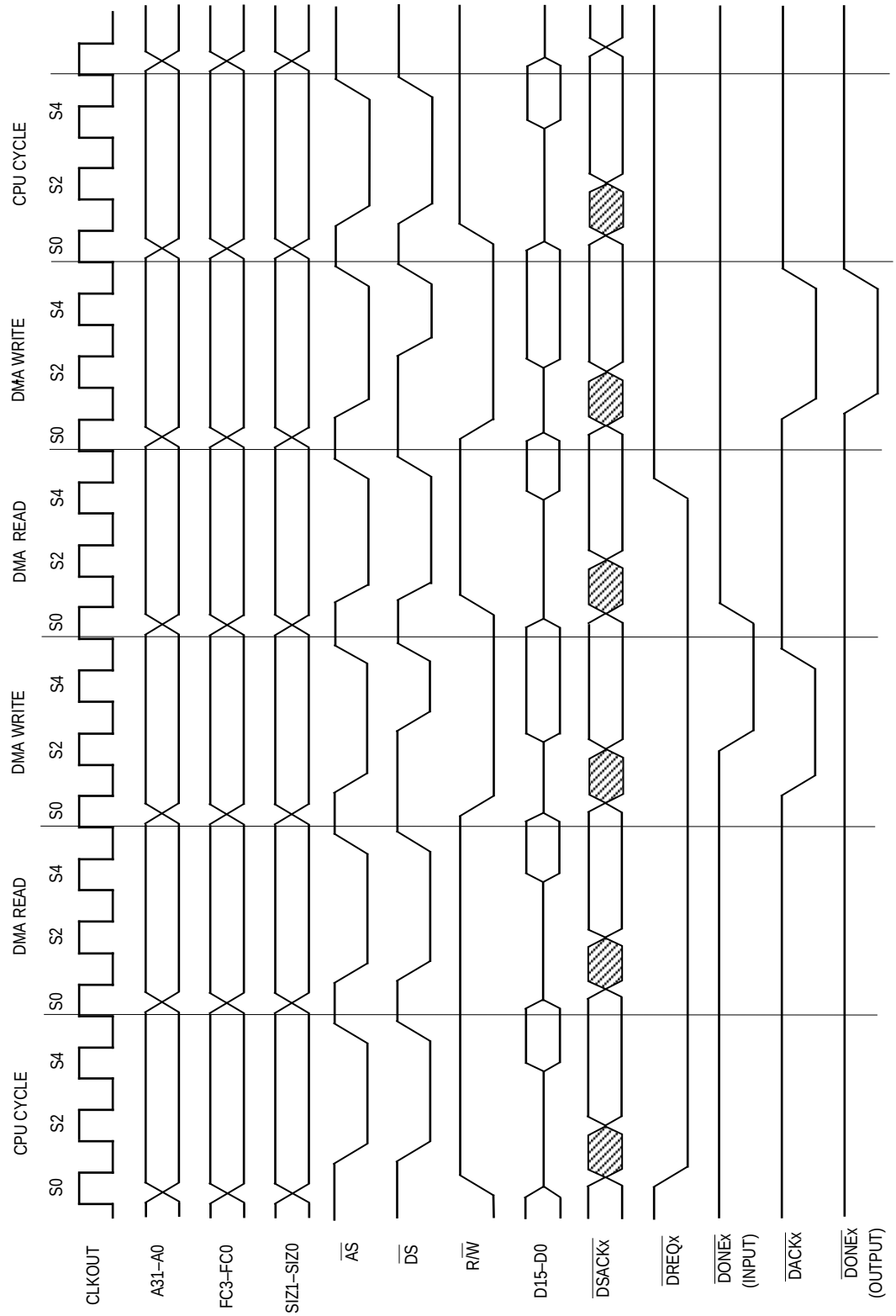


Figure 6-1. DMA Block Diagram



NOTE:
1. Timing to generate more than one DMA transfer.
2. DACKx and DONEx (DMA control signals) are asserted in the destination (write) DMA cycle.
3. DREQx must be asserted while DACKx is asserted and meet the setup and hold times for more than one DMA transfer to be recognized.
4. DONEx (input) can be asserted in either the read or write DMA bus cycle to indicate that the next DMA transfer will be the last one.

Figure 6-11. Dual-Address Write Timing (External Burst–Destination Requesting)

SE—Single-Address Enable

This bit is implemented for future MC683xx family compatibility.

1 = In single-address mode, the external data bus is driven during a DMA transfer.
0 = In single-address mode, the external data bus remains in a high-impedance state during a DMA transfer (used for intermodule DMA).

In dual-address mode, the SE bit has no effect.

Bit 11—Reserved

ISM2–ISM0—Interrupt Service Mask

These bits contain the interrupt service mask level for the channel. When the interrupt service level on the IMB is greater than the interrupt service mask level, the DMA vacates the bus and negates BR until the interrupt service level is less than or equal to the interrupt service mask level.

NOTE

When the CPU32 status register (SR) interrupt priority mask bits I2–I0 are at a higher level than the DMA ISM bits, the DMA channel will not start. The channel will begin operation when the level of the SR I2–I0 bits is less than or equal to the level of the DMA ISM bits.

SUPV—Supervisor/User

The value of this bit has no effect on registers permanently defined as supervisor-only access.

1 = The DMA channel registers defined as supervisor/user reside in supervisor data space and are only accessible from supervisor programs.
0 = The DMA channel registers defined as supervisor/user reside in user data space and are accessible from either supervisor or user programs.

MAID—Master Arbitration ID

These bits establish bus arbitration priority level among modules that have the capability of becoming bus master. For the MC68340, the MAID bits are used to arbitrate between DMA channel 1 and channel 2. If both channels are programmed with the same MAID level, channel 1 will have priority. These bits are implemented for future MC683xx Family compatibility. In the MC68340, only the SIM and the DMA can be bus masters. However, future versions of the MC683xx Family may incorporate other modules that may also be bus masters. For these devices, the MAID bits will be required. For the MAID bits, zero is the lowest priority and seven is the highest priority.

IARB — Interrupt Arbitration ID

Each module that generates interrupts has an IARB field. These bits are used to arbitrate for the bus in the case that two or more modules simultaneously generate an interrupt at the same priority level. No two modules can share the same IARB value.

The reset value of the IARB field is $0, which prevents the DMA module from arbitrating during the interrupt acknowledge cycle. The system software should initialize the IARB field to a value from $F (highest priority) to $1 (lowest priority).

NOTE

The DMA module uses only one set of IARB bits for both channels. A read or write to either MCR accesses the same IARB control bits.

6.7.2 Interrupt Register (INTR)

The INTR contains the priority level for the channel interrupt request and the 8-bit vector number of the interrupt. The register can be read or written to at any time while in supervisor mode and while the DMA module is enabled (i.e., the STP bit in the MCR is cleared).

INTR1, INTR2 $784, $7A4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	INTL			INTV							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Supervisor Only

Bits 15–11—Reserved

INTL—Interrupt Level Bits

Each module that can generate interrupts has an interrupt level field. The interrupt level field contains the priority level of the interrupt for its associated channel. The priority level encoded in these bits is sent to the CPU32 on the appropriate IRQ≈ signal. The CPU32 uses this value to determine servicing priority. See **Section 5 CPU32** for more information.

INTV—Interrupt Vector Bits

Each module that can generate interrupts has an interrupt vector field. The interrupt vector field contains the vector number of the interrupt for its associated channel. This 8-bit number indicates the offset from the base of the vector table where the address of the exception handler for the specified interrupt is located. The INTV field is reset to $0F, which indicates an uninitialized interrupt condition. See **Section 5 CPU32** for more information.

6.7.3 Channel Control Register (CCR)

The CCR controls the configuration of the DMA channel. This register is accessible in either supervisor or user space. The CCR can always be read or written to when the DMA module is enabled (i.e., the STP bit in the MCR is cleared).

This register is decremented by 1, 2, or 4 for each successful operand transfer from source to destination locations. When the BTC decrements to zero and no error has occurred, the CSR DONE bit is set. In the external request mode, the DONE≈ handshake line is also asserted when the BTC is decremented to zero.

If the operand size is byte, then the register is always decremented by 1. If the operand size is word and the starting count is even word, the register is decremented by 2. If the operand size is word and the byte count is not a multiple of 2, the CSR CONF bit is set, and a transfer does not occur. If the operand size is long word and the count is even long word, then the register is decremented by 4. If the operand size is long word and the byte count is not a multiple of 4, the CSR CONF bit is set, and a transfer does not occur. If the STR bit is set with a zero count in the BTC, the CONF bit is set, and the STR bit is cleared.

When read, this register always contains the count for the next access. If a bus error terminates the transfer, this register contains the count for the next access that would have been run had the error not occurred.

6.8 DATA PACKING

The internal DHR is a 32-bit register that can serve as a buffer register for the data being transferred during dual-address DMA cycles. No address is specified since this register can not be addressed by the programmer. The DHR allows the data to be packed and unpacked by the DMA during the dual-address transfer. For example, if the source operand size is byte and the destination operand size is word, then two-byte read cycles occur, followed by a one-word write cycle (see Figure 6-16). The two bytes of data are buffered in the DHR until the destination (write) word cycle occurs. The DHR allows for packing and unpacking of operands for the following sizes: bytes to words, bytes to long words, words to long words, words to bytes, long words to bytes, and long words to words.

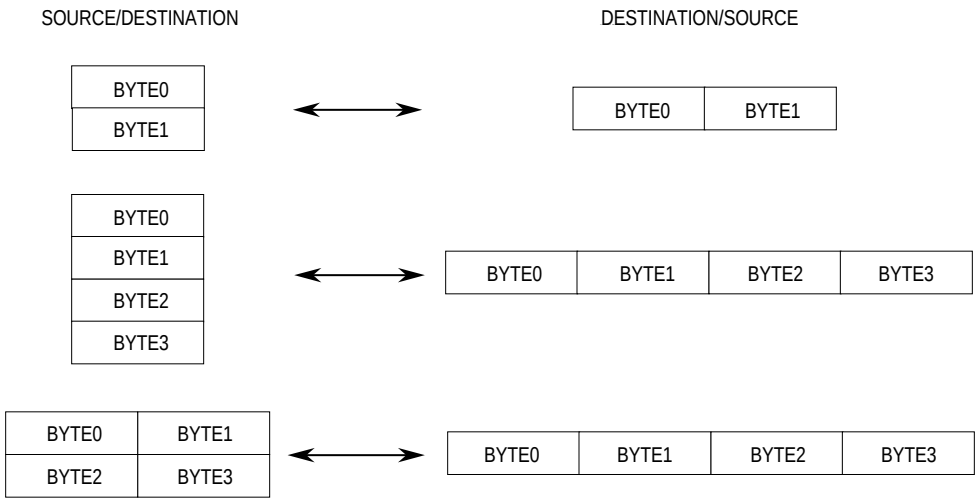


Figure 6-16. Packing and Unpacking of Operands

transmit shift register, if any, is completely sent out. If the transmitter is reset through a software command, operation ceases immediately (refer to **7.4.1.7 Command Register (CR)**). The transmitter is re-enabled through the CR to resume operation after a disable or software reset.

If clear-to-send operation is enabled, CTS≈ must be asserted for the character to be transmitted. If CTS≈ is negated in the middle of a transmission, the character in the shift register is transmitted, and TxDx remains in the 'mark' state until CTS≈ is asserted again. If the transmitter is forced to send a continuous low condition by issuing a send break command, the state of CTS≈ is ignored by the transmitter.

The transmitter can be programmed to automatically negate request-to-send (RTS≈) outputs upon completion of a message transmission. If the transmitter is programmed to operate in this mode, RTS≈ must be manually asserted before a message is transmitted. In applications in which the transmitter is disabled after transmission is complete and RTS≈ is appropriately programmed, RTS≈ is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually re-enabled by reasserting RTS≈ before the next message is to be sent.

7.3.2.2 RECEIVER. The receivers are enabled through their respective CRs located within the serial module. Functional timing information for the receiver is shown in Figure 7-6. The receiver looks for a high-to-low (mark-to-space) transition of the start bit on RxDx. When a transition is detected, the state of RxDx is sampled each 16× clock for eight clocks, starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit time clock (synchronous operation). If RxDx is sampled high, the start bit is invalid, and the search for the valid start bit begins again. If RxDx is still low, a valid start bit is assumed, and the receiver continues to sample the input at one-bit time intervals, at the theoretical center of the bit, until the proper number of data bits and parity, if any, is assembled and one stop bit is detected. Data on the RxDx input is sampled on the rising edge of the programmed clock source. The least significant bit is received first. The data is then transferred to a receiver holding register, and the RxRDY bit in the appropriate SR is set. If the character length is less than eight bits, the most significant unused bits in the receiver holding register are cleared.

After the stop bit is detected, the receiver immediately looks for the next start bit. However, if a nonzero character is received without a stop bit (framing error) and RxDx remains low for one-half of the bit period after the stop bit is sampled, the receiver operates as if a new start bit is detected. The parity error (PE), framing error (FE), overrun error (OE), and received break (RB) conditions (if any) set error and break flags in the appropriate SR at the received character boundary and are valid only when the RxRDY bit in the SR is set.

If a break condition is detected (RxDx is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver holding register, and the RB and RxRDY bits in the SR are set. The RxDx signal must return to a high condition for at least one-half bit time before a search for the next start bit begins.

1 = This bit is set when the counter output equals the value in the COM.
0 = This bit is cleared when a timeout occurs, the COM register is accessed (read or write), the timer is reset with the SWR bit, or the RESET signal is asserted on the IMB. This bit is cleared regardless of the state of the TC bit.

This bit can be used to indicate when a write to the PREL1 or PREL2 registers will not cause a problem during a counter reload at timeout. To ensure that the write to the PREL register is recognized at timeout, the latency between the read of the COM bit and the write to the PREL register must be considered.

PO7–PO0—Prescaler Output

These bits show the levels on each of the eight output taps of the prescaler. These values are updated every time that the system clock goes high and a read cycle of this byte in the SR is not in progress.

8.4.5 Counter Register (CNTR)

The CNTR reflects the value of the counter. This value can be reliably read at any time since it is updated on every rising edge of the system clock (except in the input capture/output compare mode) when a read of the register is not in progress. This read-only register can be read when the timer module is enabled (i.e. the STP bit in the MCR is cleared).

CNTR $60A, $64A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Supervisor/User

All 24 bits of the prescaler and the counter may be obtained by one long-word read at the address of the SR, since the CNTR is contiguous to it. Any changes in the prescaler value due to the two cycles necessary to perform a long-word read should be considered. If this latency presents a problem, the TGATE≈ signal may be used to disable the decrement function while the reads are occurring.

8.4.6 Preload 1 Register (PREL1)

The PREL1 stores a value that is loaded into the counter in some modes of operation. This value is loaded into the counter on the first falling edge of the counter clock after the counter is enabled. This register can be be read and written when the timer module is enabled (i.e. the STP bit in the MCR is cleared). However, a write to this register must be completed before timeout for the new value to be reliably loaded into the counter.

Caution must be exercised when accessing the COM. If it were to be accessed simultaneously by the compare logic and by a write, the old compare value may get compared to the counter value.

8.5 TIMER MODULE INITIALIZATION SEQUENCE

The following paragraphs discuss a suggested method for initializing the timer module. Since both timers are functionally equivalent, only one timer module will be referenced.

8.5.1 Timer Module Configuration

If the timer capability of the MC68340 is being used, the following steps should be followed to initialize a timer module properly. Note that this sequence must be done for each timer module used.

Control Register (CR)

- Clear the SWR bit to disable the timer.

Status Register (SR)

- Clear the TO, TG, and TG bits to reset the interrupts.

Module Configuration Register (MCR)

- Initialize the STP for normal operation.
- Select whether to respond to or ignore FREEZE (FRZx bits).
- Select the access privilege for the supervisor/user registers (SUPV bit).
- Select the interrupt arbitration level for the timer module (IARBx bits).

Interrupt Register (IR)

- Program the interrupt priority level for the timer interrupts (ILx bits).
- Program the interrupt vector number for the timer interrupts (IVx bits).

Preload Registers (PREL1 and PREL2)

- If required, initialize the preload registers for mode of operation.

Compare Register (COM)

- If desired, initialize the compare register.

The following steps begin operation:

Control Register (CR)

- Set the SWR bit to enable the timer.
- Enable the desired interrupts (IEx bits).
- Enable TGATE if required for mode of operation (TGE bit).
- Select the prescaler clock (PCLK bit).

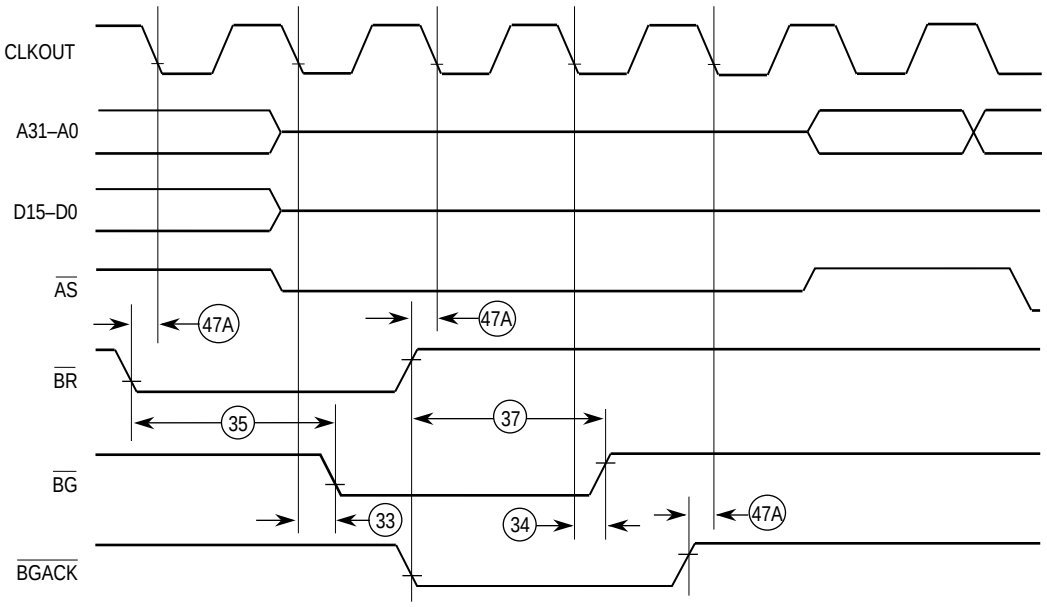


Figure 11-7. Bus Arbitration Timing—Idle Bus Case

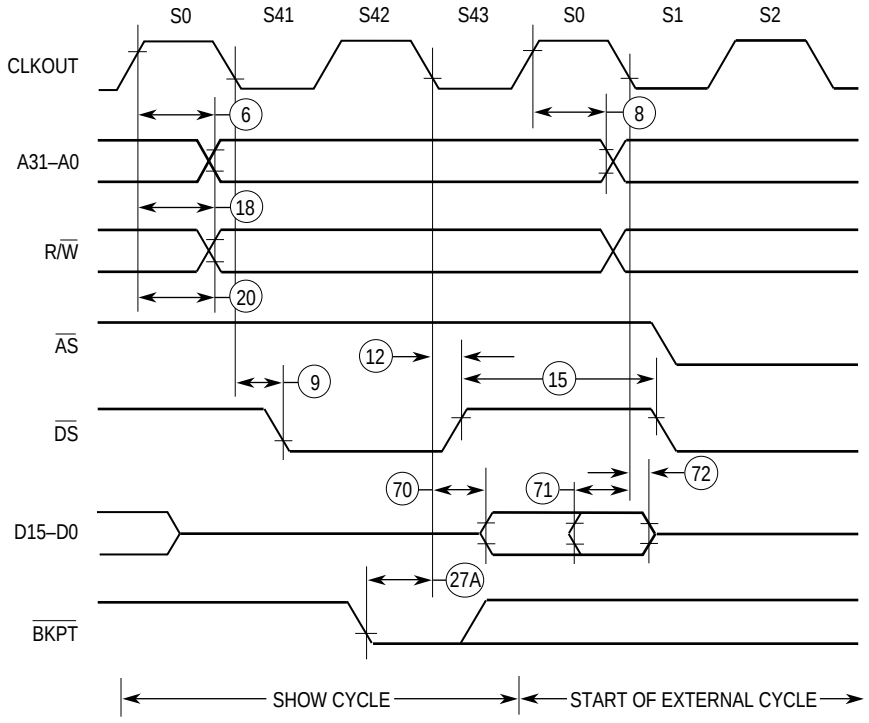


Figure 11-8. Show Cycle Timing Diagram

— E —

— F —