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	Obsolete
Core Processor	CPU32
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	25MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	144-BQFP
Supplier Device Package	144-QFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68340ab25e

requires only a 3.3-V power supply, reduces current consumption by 40–60% in all modes of operation (as well as reducing noise emissions).

The MC68340 has many additional methods of dynamically controlling power consumption during operation. The frequency of operation can be lowered under software control to reduce current consumption when performance is less critical. Idle internal peripheral modules can be turned off to save power (5–10% each). Running a special low power stop (LPSTOP) instruction shuts down the active circuits in the CPU and peripheral modules, halting instruction execution. Power consumption in this standby mode is reduced to about 350 μW. Processing and power consumption can be resumed by resetting the part or by generating an interrupt with the SIM40's periodic interrupt timer.

1.5 PHYSICAL

The MC68340 is available as 0–16.78 MHz and 0–25.16 MHz, 0°C to +70°C and -40°C to +85°C, and 5.0 V ±5% and 3.3 V ±0.3 supply voltages (reduced frequencies at 3.3 V). Thirty-two power and ground leads minimize ground bounce and ensure proper isolation of different sections of the chip, including the clock oscillator. A 144 pins are used for signals and power. The MC68340 is available in a gull-wing ceramic quad flat pack (CQFP) with 25.6-mil (0.001-in) lead spacing or a 15 × 15 plastic pin grid array (PPGA) with 0.1-in pin spacing.

1.6 COMPACT DISC-INTERACTIVE

The MC68340 was designed to meet the needs of many markets, including compact disc-interactive (CD-I). CD-I is an emerging standard for a publishing medium that will bring multimedia to a broad general audience—the consumer. CD-I players combine television and stereo systems as output devices, with interactive control using a TV remote-control-like device to provide a multimedia experience selected from software “titles” contained in compressed form on standard compact discs.

The highly integrated MC68340 is ideal as the central processor for CD-I players. It provides the M68000 microprocessor code compatibility and DMA functions required by the *CD-I Green Book* specification as well as many other useful on-chip functions for a very cost-effective solution. The extra demands of full-motion video CD-I systems make the best use of the MC68340 high performance. The MC68340 is CD-I compliant and has been CD-I qualified. With its low voltage operation, the MC68340V is the only practical choice for portable CD-I.

3.2.4 Bus Operation

The MC68340 bus is asynchronous, allowing external devices connected to the bus to operate at clock frequencies different from the clock for the MC68340. Bus operation uses the handshake lines (AS, DS, DSACK1/DSACK0, BERR, and HALT) to control data transfers. AS signals a valid address on the address bus, and DS is used as a condition for valid data on a write cycle. Decoding the SIZx outputs and lower address line A0 provides strobes that select the active portion of the data bus. The slave device (memory or peripheral) responds by placing the requested data on the correct portion of the data bus for a read cycle or by latching the data on a write cycle; the slave asserts the DSACK1/DSACK0 combination that corresponds to the port size to terminate the cycle. Alternatively, the SIM40 can be programmed to assert the DSACK1/DSACK0 combination internally and respond for the slave. If no slave responds or the access is invalid, external control logic may assert BERR to abort the bus cycle or BERR with HALT to retry the bus cycle.

DSACK≈ can be asserted before the data from a slave device is valid on a read cycle. The length of time that DSACK≈ may precede data must not exceed a specified value in any asynchronous system to ensure that valid data is latched into the MC68340. (See **Section 11 Electrical Characteristics** for timing parameters.) Note that no maximum time is specified from the assertion of AS to the assertion of DSACK≈. Although the MC68340 can transfer data in a minimum of three clock cycles when the cycle is terminated with DSACK≈, the MC68340 inserts wait cycles in clock-period increments until DSACK≈ is recognized. BERR and/or HALT can be asserted after DSACK≈ is asserted. BERR and or HALT must be asserted within the time specified after DSACK≈ is asserted in any asynchronous system. If this maximum delay time is violated, the MC68340 may exhibit erratic behavior.

3.2.5 Synchronous Operation with DSACK≈

Although cycles terminated with DSACK≈ are classified as asynchronous, cycles terminated with DSACK≈ can also operate synchronously in that signals are interpreted relative to clock edges. The devices that use these cycles must synchronize the response to the MC68340 clock (CLKOUT) to be synchronous. Since the devices terminate bus cycles with DSACK≈, the dynamic bus sizing capabilities of the MC68340 are available. The minimum cycle time for these cycles is also three clocks. To support systems that use the system clock to generate DSACK≈ and other asynchronous inputs, the asynchronous input setup time and the asynchronous input hold time are given. If the setup and hold times are met for the assertion or negation of a signal such as DSACK≈, the MC68340 is guaranteed to recognize that signal level on that specific falling edge of the system clock. If the assertion of DSACK≈ is recognized on a particular falling edge of the clock, valid data is latched into the MC68340 (for a read cycle) on the next falling clock edge if the data meets the data setup time. In this case, the parameter for asynchronous operation can be ignored. The timing parameters are described in **Section 11 Electrical Characteristics**.

3.3.2 Write Cycle

During a write cycle, the MC68340 transfers data to memory or a peripheral device. Figure 3-8 is a flowchart of a word write cycle.

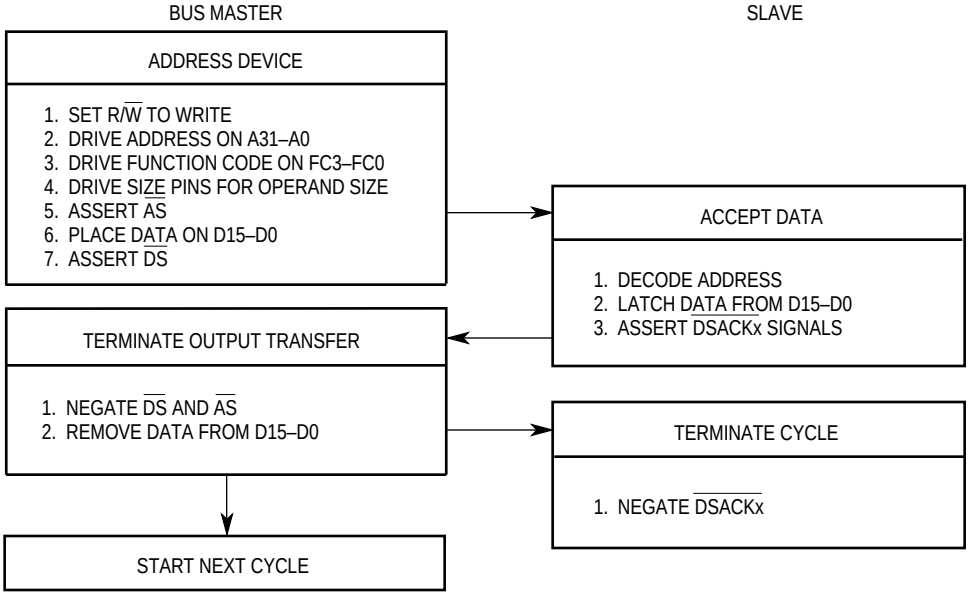


Figure 3-8. Word Write Cycle Flowchart

State 0—The write cycle starts in S0. During S0, 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 cycle. The MC68340 drives R/W low for a write cycle. SIZ1/SIZ0 become valid, indicating the number of bytes to be transferred.

State 1—One-half clock later during 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, and samples DSACK≈ at the end of S2.

State 3—The MC68340 asserts DS during S3, indicating that data is stable 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 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 DSACK≈ on the falling edges of the clock until one is recognized. The selected device uses R/W, SIZ1/SIZ0, and A0 to latch data from the appropriate byte(s) of D15–D8 and D7–D0. SIZ1/SIZ0 and A0 select the bytes of the data bus. If it has not already done so, the device asserts DSACK≈ to signal that it has successfully stored the data.

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

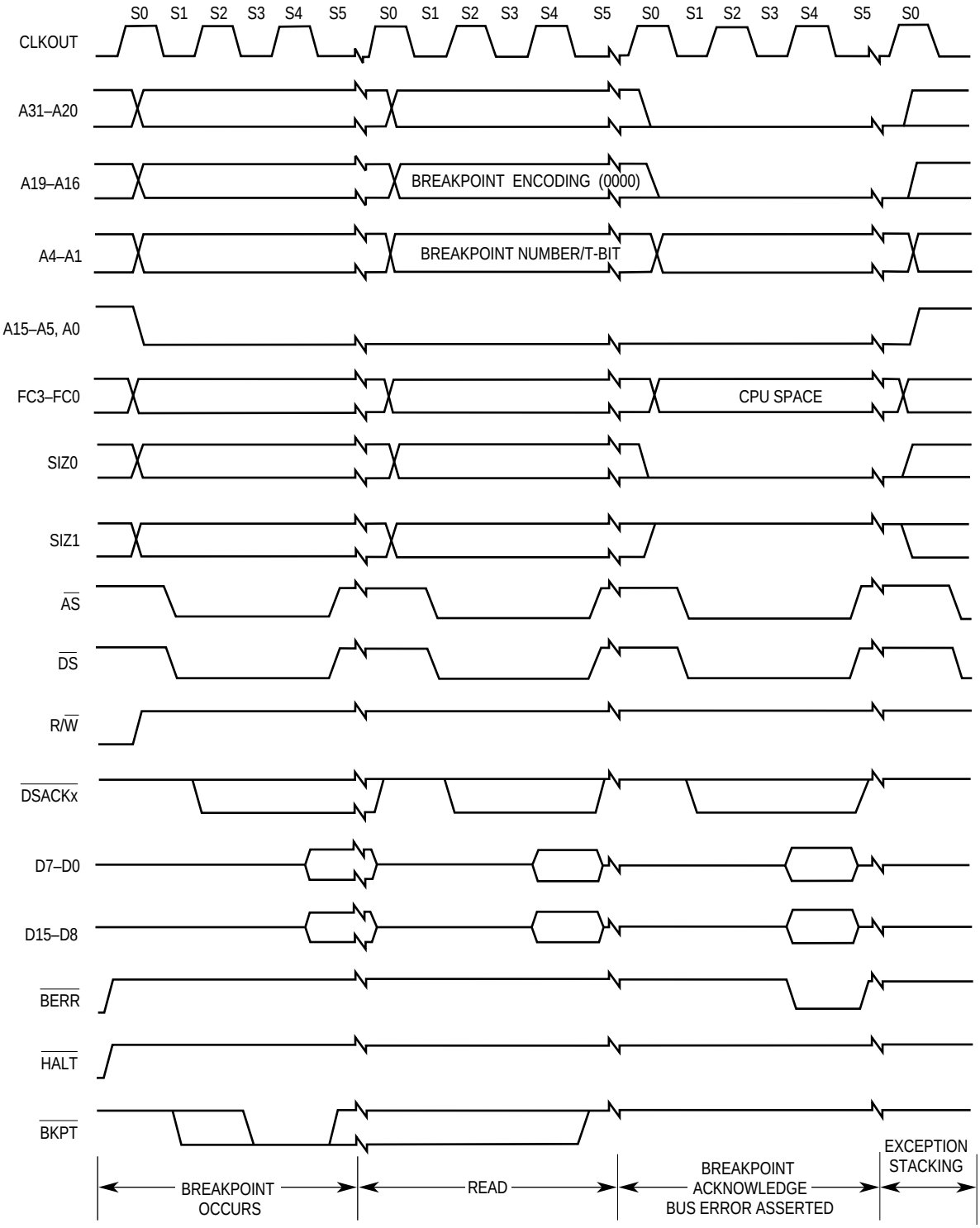


Figure 3-13. Breakpoint Acknowledge Cycle Timing (Exception Signaled)

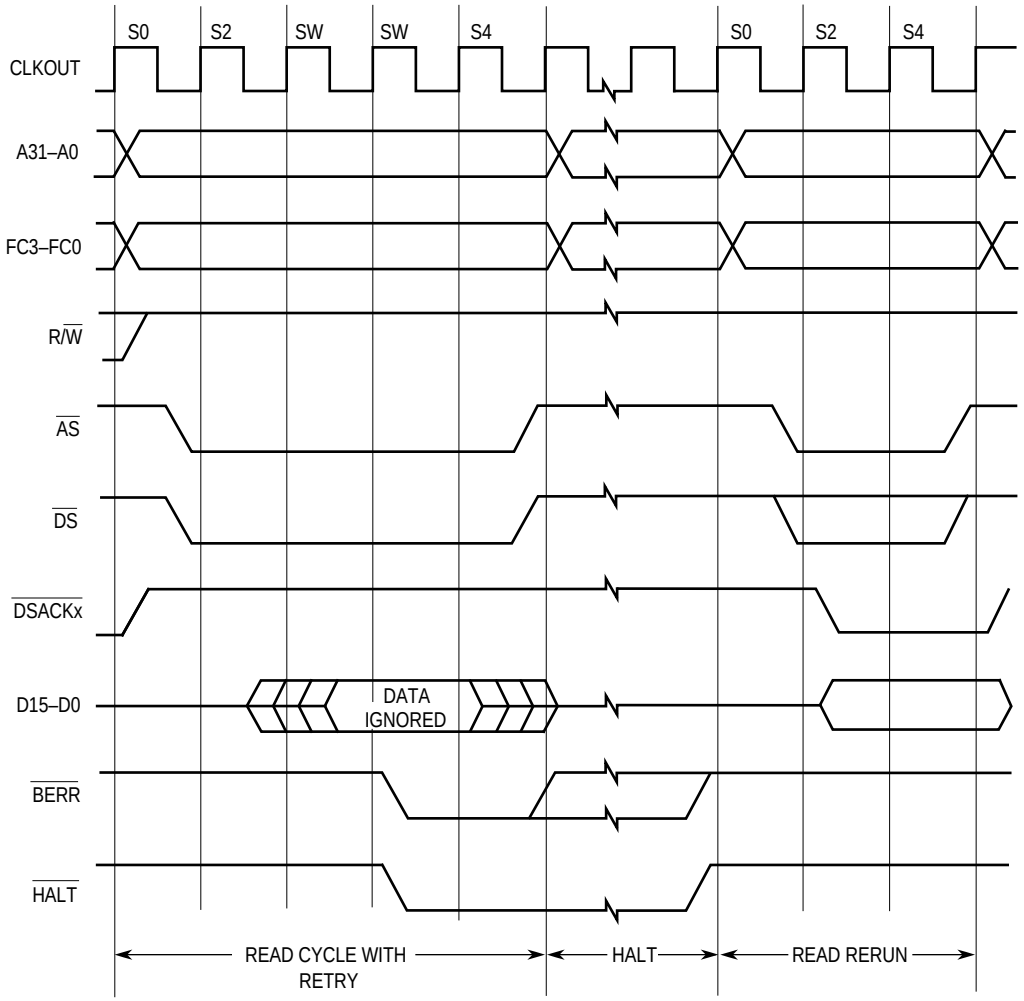


Figure 3-19. Retry Sequence

The MC68340 retries any read or write cycle of a read-modify-write operation separately; RMC remains asserted during the entire retry sequence. Asserting BR along with BERR and HALT provides a relinquish and retry operation. The MC68340 does not relinquish the bus during a read-modify-write operation. Any device that requires the MC68340 to give up the bus and retry a bus cycle during a read-modify-write cycle must assert only BERR and BR (HALT must not be included). The bus error handler software should examine the read-modify-write bit in the special status word (see **Section 5 CPU32**) and take the appropriate action to resolve this type of fault when it occurs.

Figure 3-22 is a flowchart showing bus arbitration for a single device. This technique allows processing of bus requests during data transfer cycles. Refer to Figures 3-23 and 3-24 for bus arbitration timing diagrams.

BR is negated at the time that BGACK is asserted. This type of operation applies to a system consisting of the MC68340 and one device capable of bus mastership. In a system having a number of devices capable of bus mastership, BR from each device can be wire-ORed to the MC68340. In such a system, more than one bus request could be asserted simultaneously. BG is negated a few clock cycles after the transition of BGACK. However, if bus requests are still pending after the negation of BG, the MC68340 asserts another BG within a few clock cycles after it was negated. This additional assertion of BG allows external arbitration circuitry to select the next bus master before the current bus master has finished using the bus. The following paragraphs provide additional information about the three steps in the arbitration process. Bus arbitration requests are recognized during normal processing, HALT assertion, and a CPU32 halt caused by a double bus fault.

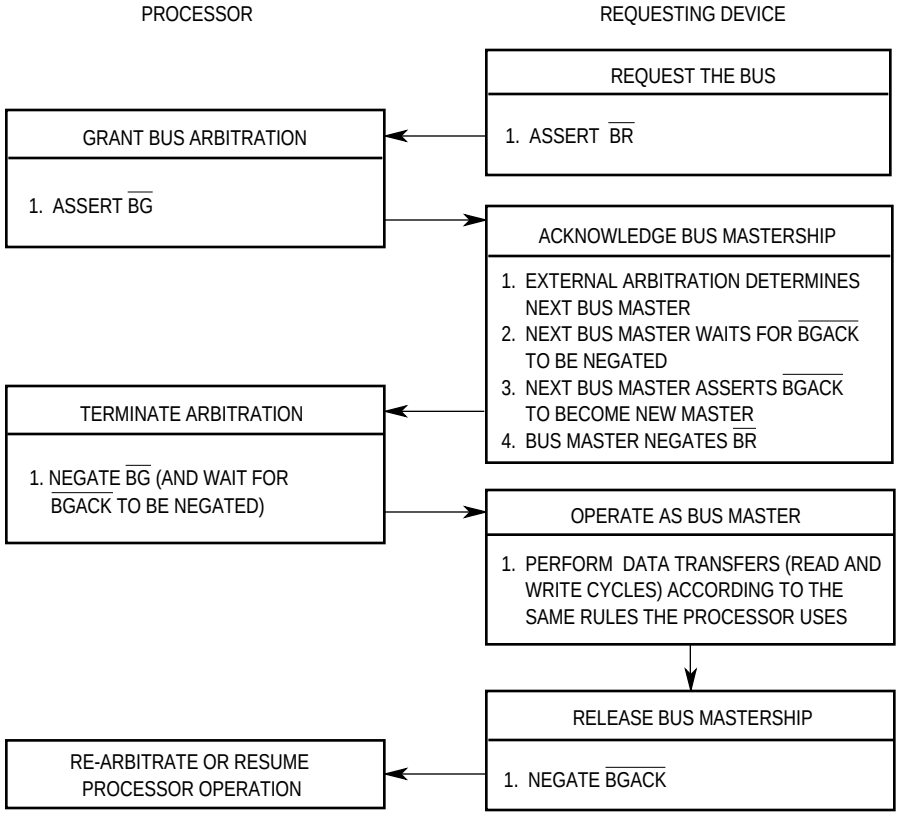


Figure 3-22. Bus Arbitration Flowchart for Single Request

4.3.5.5 PORT B PIN ASSIGNMENT REGISTER (PPARB). PPARB controls the function of each port B pin. Any set bit defines the corresponding pin to be an IRQ≈ input or CS≈ as defined in Table 4-5. Any cleared bit defines the corresponding pin to be a discrete I/O pin (or CS≈ if the FIRQ bit of the MCR is zero) controlled by the port B data and data direction registers. The MODCK signal has no function after reset. PPARB is configured to all ones at reset to provide for MODCK, IRQ7, IRQ6, IRQ5, IRQ3, and CS3–CS0. This register can be read or written at any time.

PPARB $01F

7	6	5	4	3	2	1	0
PPARB7 (IRQ7)	PPARB6 (IRQ6)	PPARB5 (IRQ5)	PPARB4 (IRQ4)	PPARB3 (IRQ3)	PPARB2 (IRQ2)	PPARB1 (IRQ1)	PPARB0 (MODCK)
RESET: 1	1	1	1	1	1	1	1

Supervisor Only

4.3.5.6 PORT B DATA DIRECTION REGISTER (DDRB). DDRB controls the direction of the pin drivers when the pins are configured as I/O. Any set bit configures the corresponding pin as an output; any cleared bit configures the corresponding pin as an input. This register affects only pins configured as discrete I/O. This register can be read or written at any time.

DDRB $01D

7	6	5	4	3	2	1	0
DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET: 0	0	0	0	0	0	0	0

Supervisor/User

4.3.5.7 PORT B DATA REGISTER (PORTB, PORTB1). This is a single register that can be accessed at two different addresses. This register affects only those pins configured as discrete I/O. A write is stored in the internal data latch, and if any port B pin is configured as an output, the value stored for that bit is driven on the pin. A read of this register returns the value stored in the register only if the pin is configured as a discrete output. Otherwise, the value read is the value of the pin. This register can be read or written at any time.

PORTB, PORTB1 $019, 01B

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
RESET: U	U	U	U	U	U	U	U

Supervisor/User

5.3.3.1 CONDITION CODE REGISTER. The CCR portion of the SR contains five bits that indicate the result of a processor operation. Table 5-3 lists the effect of each instruction on these bits. The carry bit and the multiprecision extend bit are separate in the M68000 Family to simplify programming techniques that use them. Refer to Table 5-7 as an example.

Table 5-3. Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry Z = Z Λ R∂Λ ... Λ R0
ADD, ADDI, ADDQ	*	*	*	?	?	V = Sm Λ Dm Λ R∂ V S∂ Λ D∂ Λ Rm C = Sm Λ Dm V R∂ Λ Dm V Sm Λ R∂
ADDX	*	*	?	?	?	V = Sm Λ Dm Λ R∂ V S∂ Λ D∂ Λ Rm C = Sm Λ Dm V R∂ Λ Dm V Sm Λ R∂ Z = Z Λ R∂ Λ ... Λ R0
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	—	*	*	0	0	
CHK	—	*	U	U	U	
CHK2, CMP2	—	U	?	U	?	Z = (R = LB) V (R = UB) C = (LB < UB) Λ (IR < LB) V (R > UB) V (UB < LB) Λ (R > UB) Λ (R < LB)
SUB, SUBI, SUBQ	*	*	*	?	?	V = S∂ Λ Dm Λ R∂ V Sm Λ D∂ Λ Rm C = Sm Λ D∂ V Rm Λ D∂ V Sm Λ Rm
SUBX	*	*	?	?	?	V = S∂ Λ Dm Λ R∂ V Sm Λ D∂ Λ Rm C = Sm Λ D∂ V Rm Λ D∂ V Sm Λ Rm Z = Z Λ R∂ Λ ... Λ R0
CMP, CMPI, CMPM	—	*	*	?	?	V = S∂ Λ Dm Λ R∂ V Sm Λ D∂ Λ Rm C = Sm Λ D∂ V Rm Λ D∂ V Sm Λ Rm
DIVS, DIVU	—	*	*	?	0	V = Division Overflow
MULS, MULU	—	*	*	?	0	V = Multiplication Overflow
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow Z = Z Λ R∂ Λ ... Λ R0
NEG	*	*	*	?	?	V = Dm Λ Rm C = Dm V Rm
NEGX	*	*	?	?	?	V = Dm Λ Rm C = Dm V Rm Z = Z Λ R∂ Λ ... Λ R0
ASL	*	*	*	?	?	V = Dm Λ (D∂ – 1 V ... V D∂ – r) V D∂ Λ (Dm–1 V ... + Dm – r) C = D∂ – r + 1
ASL (r = 0)	—	*	*	0	0	
LSL, ROXL	*	*	*	0	?	C = Dm – r + 1
LSR (r = 0)	—	*	*	0	0	
ROXL (r = 0)	—	*	*	0	?	C = X
ROL	—	*	*	0	?	C = Dm – r + 1
ROL (r = 0)	—	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	C = Dr – 1
ASR, LSR (r = 0)	—	*	*	0	0	
ROXR (r = 0)	—	*	*	0	?	C = X

The SSW for a released write fault contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4 3	2 0
0	0	0	TR	B1	B0	1	0	0	0	LG	SIZ	FUNC

TR, B1, and B0 are set if the corresponding exception is pending when the bus error exception is taken. Status regarding the faulted bus cycle is reflected in the LG, SIZ, and FUNC fields.

The remainder of the stack contains the PC of the next unexecuted instruction, the current SR, the address of the faulted memory location, and the contents of the data buffer that was to be written to memory. This data is written on the stack in the format depicted in Figure 5-15. When a released write fault exception handler executes, the machine will complete the faulted write and then continue executing instructions wherever the PC indicates.

5.5.3.1.2 Type II—Prefetch, Operand, RMW, and MOVEP Faults. The majority of bus error exceptions are included in this category—all instruction prefetches, all operand reads, all RMW cycles, and all operand accesses resulting from execution of MOVEP (except the last write of a MOVEP Rn,⟨ea⟩ or the last write of MOVEM, which are type I faults). The TAS, MOVEP, and MOVEM instructions account for all operand writes not considered released.

All type II faults cause an immediate exception that aborts the current instruction. Any registers that were altered as the result of an EA calculation (i.e., postincrement or predecrement) are restored prior to processing the bus cycle fault.

The SSW for faults in this category contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4 3	2 0
0	0	0	0	B1	B0	0	RM	IN	RW	LG	SIZ	FUNC

The trace pending bit is always cleared, since the instruction will be restarted upon return from the handler. Saving a pending exception on the stack causes a trace exception to be taken prior to restarting the instruction. If the exception handler does not alter the stacked SR trace bits, the trace is requeued when the instruction is started.

The breakpoint pending bits are stacked in the SSW, even though the instruction is restarted upon return from the handler. This avoids problems with bus state analyzer equipment that has been programmed to breakpoint only the first access to a specific location or to count accesses to that location. If this response is not desired, the exception handler can clear the bits before return. The RM, IN, RW, LG, FUNC, and SIZ fields all reflect the type of bus cycle that caused the fault. If the bus cycle was an RMW, the RM bit will be set, and the RW bit will show whether the fault was on a read or write.

memory access. Off-chip address comparators will not detect breakpoints on internal accesses unless show cycles are enabled. Breakpoints on prefetched instructions, which are flushed from the pipeline before execution, are not acknowledged, but operand breakpoints are always acknowledged. Acknowledged breakpoints can initiate either exception processing or BDM. See **5.5.2.6 Hardware Breakpoints** for more information.

5.6.2 Background Debug Mode

BDM is an alternate CPU32 operating mode. During BDM, normal instruction execution is suspended, and special microcode performs debugging functions under external control. Figure 5-20 is a BDM block diagram.

BDM can be initiated in several ways—by externally generated breakpoints, by internal peripheral breakpoints, by the background instruction (BGND), or by catastrophic exception conditions. While in BDM, the CPU32 ceases to fetch instructions via the parallel bus and communicates with the development system via a dedicated, high-speed, SPI-type serial command interface.

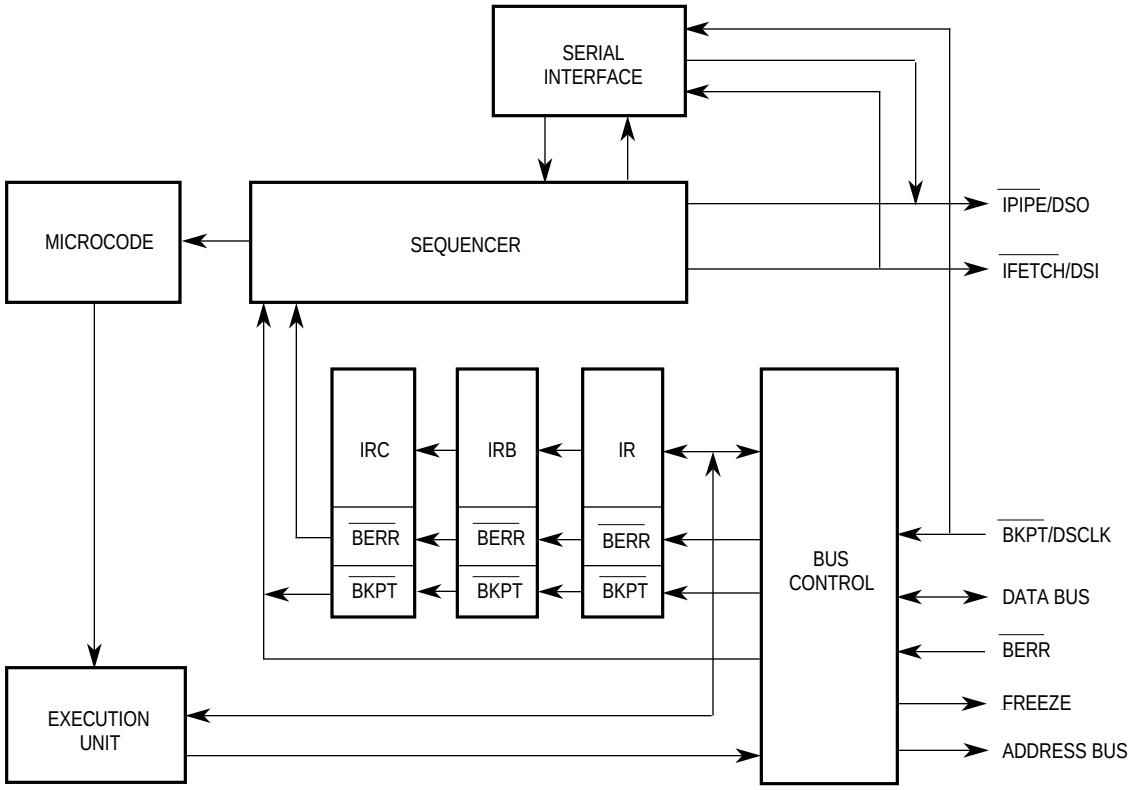


Figure 5-20. BDM Block Diagram

5.6.2.1 ENABLING BDM. Accidentally entering BDM in a nondevelopment environment could lock up the CPU32 since the serial command interface would probably not be available. For this reason, BDM is enabled during reset via the BKPT signal.

Result Data:

Always returns 32 bits of data, regardless of the size of the register being read. If the register is less than 32 bits, the result is returned zero extended.

Register Field:

The system control register is specified by the register field (see Table 5-24).

Table 5-24. Register Field for RSREG and WSREG

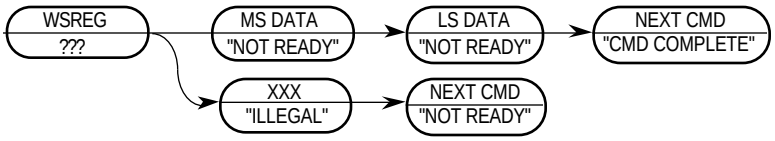
System Register	Select Code
Return Program Counter (RPC)	0000
Current Instruction Program Counter (PCC)	0001
Status Register (SR)	1011
User Stack Pointer (USP)	1100
Supervisor Stack Pointer (SSP)	1101
Source Function Code Register (SFC)	1110
Destination Function Code Register (DFC)	1111
Temporary Register A (ATEMP)	1000
Fault Address Register (FAR)	1001
Vector Base Register (VBR)	1010

5.6.2.8.7 Write System Register (WSREG). Operand data is written into the specified system control register. All registers that can be written in supervisor mode can be written in BDM. Several internal temporary registers are also accessible.

Command Format:

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

Command Sequence:



Operand Data:

The data to be written into the register is always supplied as a 32-bit long word. If the register is less than 32 bits, the least significant word is used.

Result Data:

“Command complete” status is returned when register write is complete.

Because of the stipulation that each instruction must prefetch to replace itself, the concept of negative tails has been introduced to account for these free clocks on the bus.

On a two-clock bus, it is not necessary to adjust instruction timing to account for the potential extra prefetch. The cycle times of the microsequencer and bus are matched, and no additional benefit or penalty is obtained. In the instruction execution time equations, a zero should be used instead of a negative number.

Negative tails are used to adjust for slower fetches on slower buses. Normally, increasing the length of prefetch bus cycles directly affects the cycle count and tail values found in the tables.

In the following equations, negative tail values are used to negate the effects of a slower bus. The equations are generalized, however, so that they may be used on any speed bus with any tail value.

```
NEW_TAIL = OLD_TAIL + (NEW_CLOCK – 2)
IF ((NEW_CLOCK – 4) >0) THEN
    NEW_CYCLE = OLD_CYCLE + (NEW_CLOCK -2) + (NEW_CLOCK – 4)
ELSE
    NEW_CYCLE = OLD_CYCLE + (NEW _CLOCK – 2)
```

where:

NEW_TAIL/NEW_CYCLE is the adjusted tail/cycle at the slower speed

OLD_TAIL/OLD_CYCLE is the value listed in the instruction timing tables

NEW_CLOCK is the number of clocks per cycle at the slower speed

Note that many instructions listed as having negative tails are change-of-flow instructions and that the bus speed used in the calculation is that of the new instruction stream.

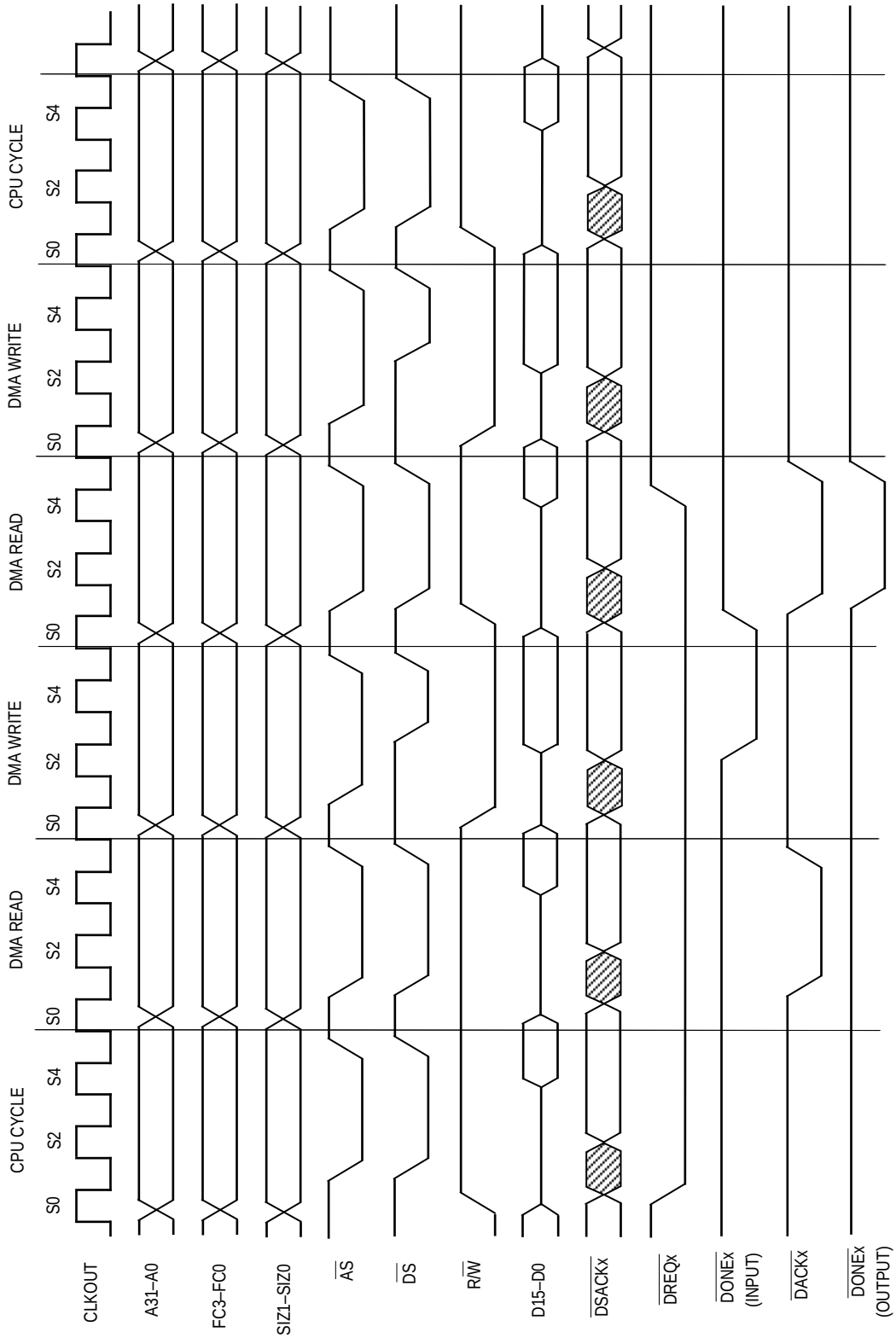
5.7.2 Instruction Stream Timing Examples

The following programming examples provide a detailed examination of timing effects. In all examples, the memory access is from external synchronous memory, the bus is idle, and the instruction pipeline is full at the start.

5.7.2.1 TIMING EXAMPLE 1—EXECUTION OVERLAP. Figure 5-33 illustrates execution overlap caused by the bus controller's completion of bus cycles while the sequencer is calculating the next EA. One clock is saved between instructions since that is the minimum time of the individual head and tail numbers.

Instructions

```
MOVE.W    A1, (A0) +
ADDQ.W    #1, (A0)
CLR.W     $30 (A1)
```



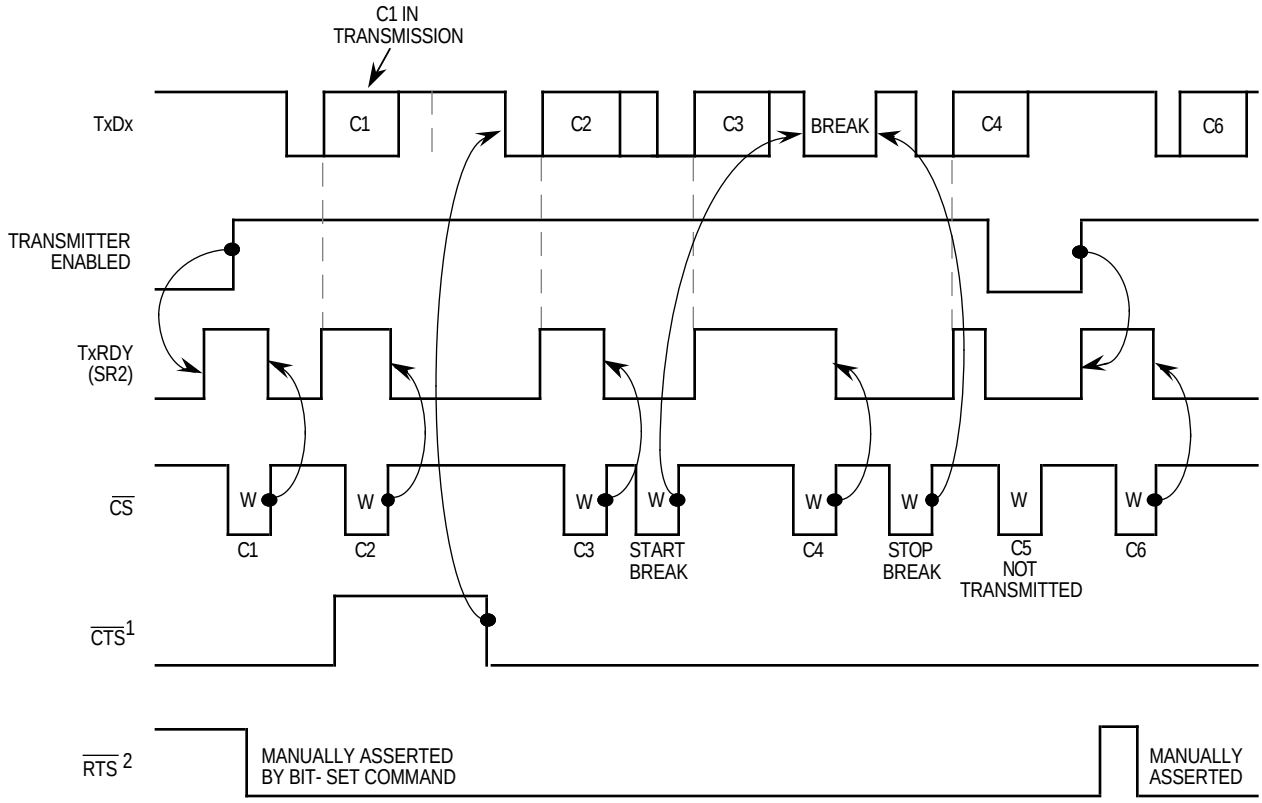
NOTE:
1. Timing to generate more than one DMA transfer.
2. DACKx and DONEx (DMA control signals) are asserted in the source (read) DMA cycle.
3. DREQx must be asserted while DACKx is asserted and meet the setup and hold times for more than one DMA transfer to be recognized.
4. DONEx (input) can be asserted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

Figure 6-9. Dual-Address Read Timing (External Burst–Source Requesting)

```
* This code is used to initialize the 68340's internal DMA channel
* registers, providing basic functions for operation.
* The code sets up channel 1 for internal request generation
* to perform a memory block initialization for 100 bytes.
*******************************************************************************
*******************************************************************************
* SIM40 equates
*******************************************************************************
MBAR        EQU   $0003FF00   Address of SIM40 Module Base Address Reg.
MODBASE     EQU   $FFFFF000   SIM40 MBAR address value

*******************************************************************************
* DMA Channel 1 equates
DMACH1      EQU   $780        Offset from MBAR for channel 1 regs
DMAMCR1     EQU   $0          MCR for channel 1

* Channel 1 register offsets from channel 1 base address
DMAINT1     EQU   $4          interrupt register channel 1
DMACCR1     EQU   $8          control register channel 1
DMACSR1     EQU   $A          status register channel 1
DMAFCR1     EQU   $B          function code register channel 1
DMASAR1     EQU   $C          source address register channel 1
DMADAR1     EQU   $10         destination address register channel 1
DMABTC1     EQU   $14         byte transfer count register channel 1
SARADD      EQU   $6000       source address
DARADD      EQU   $8000       destination address
NUMBYTE     EQU   $64         number of bytes to transfer

*******************************************************************************
*******************************************************************************
* Initialize DMA Channel 1
*******************************************************************************
        LEA     MODBASE+DMACH1,A0   Pointer to channel 1

* Initialize DMA channel 1 MCR
* Normal Operation, ignore FREEZE, dual-address mode. ISM field at 3. Make
* sure CPU32 SR I2-I0 bits are less than or equal to ISM bits for channel
* startup.Supervisor/user reg. unrestricted, MAID field at 3.
* IARB priority at 4.
        MOVE.W      #$0334,(A0)

* Clear channel control reg.
* Clear STR (start) bit to prevent the channel from starting a transfer early.
        CLR.W       DMACCR1(A0)

* Initialize interrupt reg.
* Interrupt priority at 7, interrupt vector at $42.
```

7.3.2.1 TRANSMITTER. The transmitters are enabled through their respective command registers (CR) located within the serial module. The serial module signals the CPU32 when it is ready to accept a character by setting the transmitter-ready bit (TxRDY) in the channel's status register (SR). Functional timing information for the transmitter is shown in Figure 7-5.

The transmitter converts parallel data from the CPU32 to a serial bit stream on TxDx. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Data is shifted from the transmitter output on the falling edge of the clock source.



NOTES:
1. TIMING SHOWN FOR MR2(4) = 1
2. TIMING SHOWN FOR MR2(5) = 1
3. C_N = TRANSMIT CHARACTER
4. W = WRITE

Figure 7-5. Transmitter Timing Diagram

Following transmission of the stop bits, if a new character is not available in the transmitter holding register, the TxDx output remains high ('mark' condition), and the transmitter empty bit (TxEMP) in the SR is set. Transmission resumes and the TxEMP bit is cleared when the CPU32 loads a new character into the transmitter buffer (TB). If a disable command is sent to the transmitter, it continues operating until the character in the

TxCTS—Transmitter Clear-to-Send

1 = Enables clear-to-send operation. The transmitter checks the state of the CTS≈ input each time it is ready to send a character. If CTS≈ is asserted, the character is transmitted. If CTS≈ is negated, the channel TxDx remains in the high state, and the transmission is delayed until CTS≈ is asserted. Changes in CTS≈ while a character is being transmitted do not affect transmission of that character. If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter.

0 = The CTS≈ has no effect on the transmitter.

SB3–SB0—Stop-Bit Length Control

These bits select the length of the stop bit appended to the transmitted character as listed in Table 7-10. Stop-bit lengths of nine-sixteenth to two bits, in increments of one-sixteenth bit, are programmable for character lengths of six, seven, and eight bits. For a character length of five bits, one and one-sixteenth to two bits are programmable in increments of one-sixteenth bit. In all cases, the receiver only checks for a high condition at the center of the first stop-bit position—i.e., one bit time after the last data bit or after the parity bit, if parity is enabled.

If an external 1× clock is used for the transmitter, MR2 bit 3 = 0 selects one stop bit, and MR2 bit 3 = 1 selects two stop bits for transmission.

Table 7-10. SBx Control Bits

SB3	SB2	SB1	SB0	Length 6-8 Bits	Length 5 Bits
0	0	0	0	0.563	1.063
0	0	0	1	0.625	1.125
0	0	1	0	0.688	1.188
0	0	1	1	0.750	1.250
0	1	0	0	0.813	1.313
0	1	0	1	0.875	1.375
0	1	1	0	0.938	1.438
0	1	1	1	1.000	1.500
1	0	0	0	1.563	1.563
1	0	0	1	1.625	1.625
1	0	1	0	1.688	1.688
1	0	1	1	1.750	1.750
1	1	0	0	1.813	1.813
1	1	0	1	1.875	1.875
1	1	1	0	1.938	1.938
1	1	1	1	2.000	2.000

Disabled—TOUTx is disabled and three-stated.

Toggle Mode—If the timer is disabled (SWR = 0) when this encoding is programmed, TOUTx is immediately set to zero. If the timer is enabled (SWR = 1), timeout events (counter reaches $0000) toggle TOUTx. In the input capture/output compare mode, TOUTx is immediately set to zero if the timer is disabled (SWR = 0). If the timer is enabled (SWR = 1), timer compare events toggle TOUTx. (Timer compare events occur when the counter reaches the value stored in the COM.)

Zero Mode—If the timer is disabled (SWR = 0) when this encoding is programmed, TOUTx is immediately set to zero. If the timer is enabled (SWR = 1), TOUTx will be set to zero at the next timeout. In the input capture/output compare mode, TOUTx is immediately set to zero if the timer is disabled (SWR = 0). If the timer is enabled (SWR = 1), TOUTx will be set to zero at timeouts and set to one at timer compare events. If the COM is $0000, TOUTx will be set to zero at the timeout/timer compare event.

One Mode—If the timer is disabled (SWR = 0) when this encoding is programmed, TOUTx is immediately set to one. If the timer is enabled (SWR = 1), TOUTx will be set to one at the next timeout. In the input capture/output compare mode, TOUTx is immediately set to one if the timer is disabled (SWR = 0). If the timer is enabled (SWR = 1), TOUTx will be set to one at timeouts and set to zero at timer compare events. If the COM is $0000, TOUTx will be set to one at the timeout/timer compare event.

8.4.4 Status Register (SR)

The SR contains timer status information as well as the state of the prescaler. This register is updated on the rising edge of the system clock when a read of its location is not in progress, allowing the most current information to be contained in this register. The register can be read, and the TO, TG, and TC bits can be written when the timer module is enabled (i.e., the STP bit in the MCR is cleared).

SR $608, $648

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ	TO	TG	TC	TGL	ON	OUT	COM	PO7	PO6	PO5	PO4	PO3	PO2	PO1	PO0
RESET (TGATE≈ NEGATED):															
0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1
RESET (TGATE≈ ASSERTED):															
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Supervisor/User

IRQ—Interrupt Request bit

The positioning of this bit in the most significant location in this register allows it it be conditionally tested as if it were a signed binary integer.

1 = An interrupt condition has occurred. This bit is the logical OR of the enabled TO, TG, and TC interrupt bits.

0 = The bit(s) that caused the interrupt condition has been cleared. If an IRQ≈ signal has been asserted, it is negated when this bit is cleared.

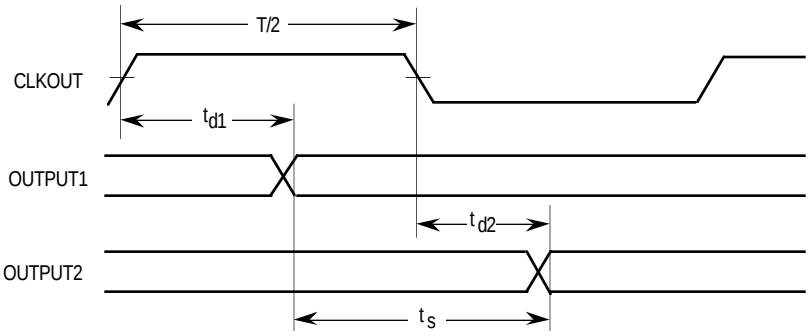


Figure 10-13. Skew between Two Outputs

The method for calculating a frequency-adjusted t_S is as follows:

$$t_S' = t_S + N\,(T_f'/2 - T_f/2) + (T_f'/2 - t_{d1})$$

where:

t_S' = the frequency-adjusted skew

t_S = the skew at full speed

N = the number of full one-half clock periods in t_S, if any

$T_f'/2$ = one-half the new clock period

$T_f/2$ = one-half the clock period at full speed

t_{d1} = the propagation time for the first output from the clock edge

The following calculation uses a 16.78-MHz port, specification 21, R/W high to AS asserted, at 8 MHz as an example:

t_S = 15 ns minimum

N = 0

$T_f'/2$ = 125/2 = 62.5 ns

$T_f/2$ = 60/2 = 30 ns

t_{d1} = 30 ns maximum

therefore:

$$t_S' = 15 + 0(62.5 - 30) + (62.5 - 30) = 47.5 \text{ ns minimum}$$

In this manner, new specifications for lower frequencies can be derived for an MC68340.

The MC68340V low voltage parts can operate up to 8.39 MHz or 16.78 MHz with a 3.3 V ±0.3 V supply. Separate part numbers are used to distinguish the operation of the parts according to the supply voltage. Refer to **Section 12 Ordering Information and Mechanical Data** for the part numbering schemes. MC68340 is used throughout this section to refer to the 16.78- or 25.16-MHz parts at 5.0 V ±5%. MC68340V is used throughout this section to refer to the 8.39- or 16.78-MHz parts at 3.3 V ±0.3 V.

NOTE

The electrical specifications in this section for the MC68340 25.16 MHz at 5.0 V ±5% and the 3.3 V ±0.3 V specifications for both the 8.39- and 16.78-MHz parts are preliminary.