Freescale Semiconductor - MC68340CAB25E Datasheet

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	25MHz
Co-Processors/DSP	-
RAM Controllers	DRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	144-BQFP
Supplier Device Package	144-QFP (28x28)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68340cab25e

TABLE OF CONTENTS (Concluded)

NOTE

The terms *assert* and *negate* are used throughout this section to avoid confusion when dealing with a mixture of active-low and active-high signals. The term *assert* or *assertion* indicates that a signal is active or true, independent of the level represented by a high or low voltage. The term *negate* or *negation* indicates that a signal is inactive or false.

2.2 ADDRESS BUS

The address bus signals are outputs that define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The MC68340 places the address on the bus at the beginning of a bus cycle. The address is valid while AS is asserted.

The address bus consists of the following two groups. Refer to **Section 3 Bus Operation** for information on the address bus and its relationship to bus operation.

2.2.1 Address Bus (A23–A0)

These three-state outputs (along with A31–A24) provide the address for the current bus cycle, except in the CPU address space.

2.2.2 Address Bus (A31–A24)

These pins can be programmed as the most significant eight address bits, port A parallel I/O, or interrupt acknowledge signals. These pins can be used for more than one of their multiplexed functions as long as the external demultiplexing circuit properly resolves interaction between the different functions.

A31–A24

These pins can function as the most significant eight address bits.

Port A7–A0

These eight pins can serve as a dedicated parallel I/O port. See **Section 4 System Integration Module** for more information on programming these pins.

IACK7– IACK1

The MC68340 asserts one of these pins to indicate the level of an external interrupt during an interrupt acknowledge cycle. Peripherals can use the IACK≈ signals instead of monitoring the address bus and function codes to determine that an interrupt acknowledge cycle is in progress and to obtain the current interrupt level.

2.3 DATA BUS (D15–D0)

This bidirectional, nonmultiplexed, parallel bus contains the data being transferred to or from the MC68340. A read or write operation may transfer 8 or 16 bits of data (one or two bytes) in one bus cycle. During a read cycle, the data is latched by the MC68340 on the

T≈RDYA

When used for this function, this signal reflects the complement of the status of bit 2 of the channel A status register. This signal can be used to control parallel data flow by acting as an interrupt to indicate when the transmitter contains a character.

OP6

When used for this function, this output is controlled by bit 6 in the output port data registers.

2.13.8 Receiver Ready (R≈RDYA)

This active-low output signal can be programmed as the channel A receiver ready, channel A FIFO full indicator, or a dedicated parallel output.

R≈RDYA

When used for this function, this signal reflects the complement of the status of bit 1 of the interrupt status register. This signal can be used to control parallel data flow by acting as an interrupt to indicate when the receiver contains a character.

FFULLA

When used for this function, this signal reflects the complement of the status of bit 1 of the interrupt status register. This signal can be used to control parallel data flow by acting as an interrupt to indicate when the receiver FIFO is full.

OP4

When used for this function, this output is controlled by bit 4 in the output port data registers.

2.14 TIMER SIGNALS

The following external signals are used by the timer modules. See **Section 8 Timer Modules** for additional information on these signals.

2.14.1 Timer Gate (TGATE2, TGATE1)

These active-low inputs can be programmed to enable and disable the counters and prescalers. TGATE≈ can also be programmed as a simple input.

2.14.2 Timer Input (TIN2, TIN1)

These inputs can be programmed as clocks that cause events to occur in the counters and prescalers.

2.14.3 Timer Output (TOUT2, TOUT1)

These outputs drive the various output waveforms generated by the timers.

State 0—The read cycle starts in state 0 (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 high for a read cycle. SIZ1/SIZ0 become valid, indicating the number of bytes requested for transfer.

State 1—One-half clock later, in state 1 (S1), the MC68340 asserts AS indicating a valid address on the address bus. The MC68340 also asserts DS during S1. The selected device uses R/W, SIZ1 or SIZ0, A0, and DS to place its information on the data bus. One or both of the bytes (D15–D8 and D7–D0) are selected by SIZ1/SIZ0 and A0.

State 2—As long as at least one of the DSACK≈ signals is recognized on the falling edge of S2 (meeting the asynchronous input setup time requirement), data is latched on the falling edge of S4, and the cycle terminates.

State 3—If DSACK≈ is not recognized by the start of state 3 (S3), the MC68340 inserts wait states instead of proceeding to states 4 and 5. 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.

State 4—At the falling edge of state 4 (S4), the MC68340 latches the incoming data and samples DSACK≈ to get the port size.

State 5—The MC68340 negates AS and DS during state 5 (S5). It holds the address valid during S5 to provide address hold time for memory systems. R/W, SIZ1 and SIZ0, and FC3–FC0 also remain valid throughout S5. 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 its 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 bus cycle.

4.2.5.2 PORT B. Port B pins can be independently programmed to function as chip selects, IRQ≈ and MODCK pins, or discrete I/O pins. These pins are multiplexed as shown in Figure 4-7. Selection of a pin function is accomplished by a combination of the port B pin assignment register (PPARB) and the FIRQ bit of the MCR. See Table 4-5 for port B combinations. By changing the value of the FIRQ bit and the corresponding bits in the PPARB for a particular signal, the port B pins can be configured for different pin functions. Upon reset, port B is configured as MODCK, IRQ7, IRQ6, IRQ5, IRQ3, and CS3–CS0.

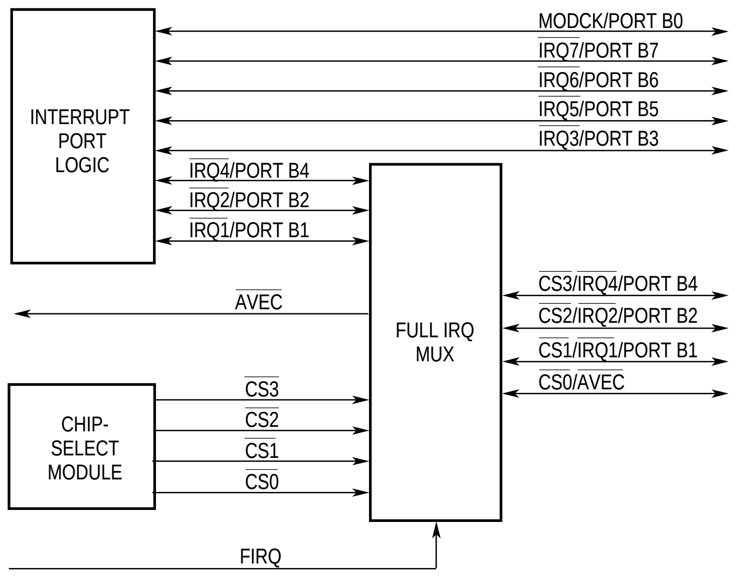


Figure 4-7. Full Interrupt Request Multiplexer

Table 4-5. Port B Pin Assignment Register

Signal	Pin Function			
	FIRQ = 0 PPARB = 0	FIRQ = 0 PPARB = 1	FIRQ = 1 PPARB = 0	FIRQ = 1 PPARB = 1
IRQ7	PORTB7	IRQ7	PORTB7	IRQ7
IRQ6	PORTB6	IRQ6	PORTB6	IRQ6
IRQ5	PORTB5	IRQ5	PORTB5	IRQ5
IRQ3	PORTB3	IRQ3	PORTB3	IRQ3
CS3	CS3	CS3	PORTB4	IRQ4
CS2	CS2	CS2	PORTB2	IRQ2
CS1	CS1	CS1	PORTB1	IRQ1
CS0	CS0	CS0	AVEC	AVEC
MODCK	PORTB0	MODCK	PORTB0	MODCK

NOTE: MODCK has no function after reset.

4.3.4.1 BASE ADDRESS REGISTERS. There are four 32-bit base address registers in the chip select function, one for each chip select signal.

Base Address 1 $044, $04C, $054, $05C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Supervisor Only

Base Address 2 $046, $04E, $056, $05E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BFC3	BFC2	BFC1	BFC0	WP	FTE	NCS	V
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	0	0

U = Unaffected by reset Supervisor Only

BA31–BA8—Base Address Bits 31–8

The base address field, the upper 24 bits of each base address register, selects the starting address for the chip select. The specified base address must be on a multiple of the selected block size. The corresponding bits, AM31–AM8, in the address mask register define the size of the block for the chip select. The base address field (and the base function code field) is compared to the address on the address bus to determine if a chip select should be generated.

BFC3–BFC0—Base Function Code Bits 3–0

The value programmed into this field causes a chip select to be asserted for a certain address space type. There are nine function code address spaces (see **Section 3 Bus Operation**) specified as either user or supervisor, program or data, CPU, and DMA. These bits should be used to allow access to one type of address space. If access to more than one type of address space is desired, the FCMx bits should be used in addition to the BFCx bits. To prevent access to CPU space, set the NCS bit.

WP—Write Protect

This bit can restrict write accesses to the address range in a base address register. An attempt to write to the range of addresses specified in a base address register that has this bit set returns BERR.

1 = Only read accesses are allowed.
0 = Either read or write accesses are allowed.

FTE—Fast-Termination Enable

This bit causes the cycle to terminate early with an internal DSACK≈, giving a fast two-clock external access. When clear, all external cycles are at least three clocks. If fast termination is enabled, the DD bits of the corresponding address mask register are overridden (see **Section 3 Bus Operation**).

1 = Fast termination cycle enabled (termination determined by PS bits).
0 = Fast termination cycle disabled (termination determined by DD and PS bits).

```
******************************************************************************
*  Data table for chip select initialization
******************************************************************************

* CS0 - EPROM - 00060000-0007ffff, 3-wait states, 16-bit term., write protect
CSAM0$       DC.L   $0001FFFD
CSBAR0$      DC.L   $00060009
* CS1 - RAM - 00000000-0000ffff, fast termination
CSAM1$       DC.L   $0000FFF0
CSBAR1$      DC.L   $00000005
* CS2 - external device - 00FFE8xx, external termination
CSAM2$       DC.L   $000000F3
CSBAR2$      DC.L   $00FFE801
* CS3 - secondary memory - 00000000-0003ffff, 3-wait states, 16-bit term.
CSAM3$       DC.L   $0003FFFD
CSBAR3$      DC.L   $00000001

******************************************************************************
          END
```

The table instruction is executed with the following bit pattern in Dx:

31 16	15															0
NOT USED	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0

Table Entry Offset ⇒ Dx [8:15] = \$A3 = 163

Interpolation Fraction ⇒ Dx [0:7] = \$80 = 128

Using this information, the table instruction calculates dependent variable Y:

$$Y = 1669 + (128\,(1679 - 1669)) / 256 = 1674$$

5.3.4.2 TABLE EXAMPLE 2: COMPRESSED TABLE. In Example 2 (see Figure 5-8), the data from Example 1 has been compressed by limiting the maximum value of the independent variable. Instead of the range $0 \leq X = 65535$, X is limited to $0 \leq X \leq 1023$. The table has been compressed to only five entries, but up to 256 levels of interpolation are allowed between entries.

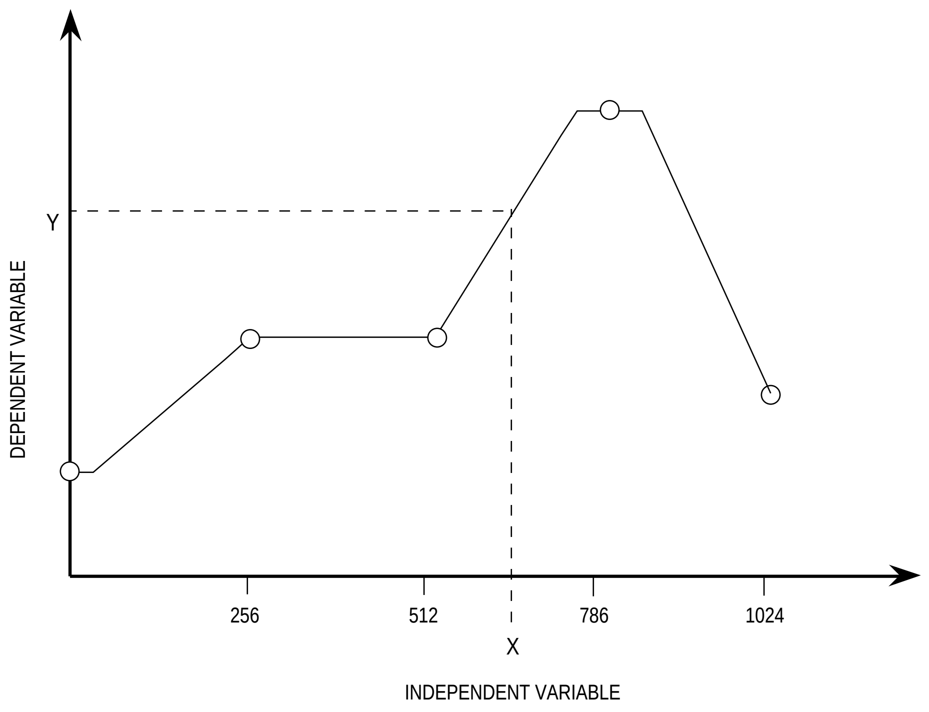


Figure 5-8. Table Example 2

NOTE

Extreme table compression with many levels of interpolation is possible only with highly linear functions. The table entries within the range of interest are listed in Table 5-14.

exceptions, will be restarted upon return from the exception handler. When a fault occurs after an operand has transferred, that transfer is not "undone". However, these memory locations are accessed a second time when the instruction is restarted. If a register used in an EA calculation is overwritten before a fault occurs, an incorrect EA is calculated upon instruction restart.

5.5.3.2.6 Type III—Correcting Faults via RTE. The preferred method of MOVEM bus fault recovery is to correct the cause of the fault and then execute an RTE instruction without altering the stack contents.

The RTE recognizes that MOVEM was in progress when a fault occurred, restores the appropriate machine state, refetches the instruction, repeats the faulted transfer, and continues the instruction.

MOVEM is the only instruction continued upon return from an exception handler. Although the instruction is refetched, the EA is not recalculated, and the mask is rescanned the same number of times as before the fault; modifying the code prior to RTE can cause unexpected results.

5.5.3.2.7 Type IV—Correcting Faults via Software. Bus error exceptions can occur during exception processing while the processor is fetching an exception vector or while it is stacking. The same stack frame and SSW are used in both cases, but each has a distinct fault address. The stacked faulted exception format/vector word identifies the type of faulted exception and the contents of the remainder of the frame. A fault address corresponding to the vector specified in the stacked format/vector word indicates that the processor could not obtain the address of the exception handler.

A bus error exception handler should execute RTE after correcting a fault. RTE restores the internal machine state, fetches the address of the original exception handler, recreates the original exception stack frame, and resumes execution at the exception handler address.

If the fault is intractable, the exception handler should rewrite the faulted exception stack frame at SP + $14 + $06 and then jump directly to the original exception handler. The stack frame can be generated from the information in the bus error frame: the pre-exception SR (SP + $0C), the format/vector word (SP + $0E), and, if the frame being written is a six-word frame, the PC of the instruction causing the exception (SP + $10). The return PC value is available at SP + $02.

A stacked fault address equal to the current SP may indicate that, although the first exception received a bus error while stacking, the bus error exception stacking successfully completed. This occurrence is extremely improbable, but the CPU32 supports recovery from it. Once the exception handler determines that the fault has been corrected, recovery can proceed as described previously. If the fault cannot be corrected, move the supervisor stack to another area of memory, copy all valid stack frames to the new stack, create a faulted exception frame on top of the stack, and resume execution at the exception handler address.

BDM operation is enabled when BKPT is asserted (low) at the rising edge of RESET. BDM remains enabled until the next system reset. A high BKPT on the trailing edge of RESET disables BDM. BKPT is relatched on each rising transition of RESET. BKPT is synchronized internally and must be held low for at least two clock cycles prior to negation of RESET.

BDM enable logic must be designed with special care. If hold time on BKPT (after the trailing edge of RESET) extends into the first bus cycle following reset, this bus cycle could be tagged with a breakpoint. Refer to **Section 3 Bus Operation** for timing information.

5.6.2.2 BDM SOURCES. When BDM is enabled, any of several sources can cause the transition from normal mode to BDM. These sources include external BKPT hardware, the BGND instruction, a double bus fault, and internal peripheral breakpoints. If BDM is not enabled when an exception condition occurs, the exception is processed normally. Table 5-19 summarizes the processing of each source for both enabled and disabled cases. As depicted in the table, the BKPT instruction never causes a transition into BDM.

Table 5-19. BDM Source Summary

Source	BDM Enabled	BDM Disabled
BKPT	Background	Breakpoint Exception
Double Bus Fault	Background	Halted
BGND Instruction	Background	Illegal Instruction
BKPT Instruction	Opcode Substitution/ Illegal Instruction	Opcode Substitution/ Illegal Instruction

5.6.2.2.1 External BKPT Signal. Once enabled, BDM is initiated whenever assertion of BKPT is acknowledged. If BDM is disabled, a breakpoint exception (vector $0C) is acknowledged. The BKPT input has the same timing relationship to the data strobe trailing edge as does read cycle data. There is no breakpoint acknowledge bus cycle when BDM is entered.

5.6.2.2.2 BGND Instruction. An illegal instruction, $4AFA, is reserved for use by development tools. The CPU32 defines $4AFA (BGND) to be a BDM entry point when BDM is enabled. If BDM is disabled, an illegal instruction trap is acknowledged. Illegal instruction traps are discussed in **5.5.2.8 Illegal or Unimplemented Instructions**.

5.6.2.2.3 Double Bus Fault. The CPU32 normally treats a double bus fault (two bus faults in succession) as a catastrophic system error and halts. When this condition occurs during initial system debug (a fault in the reset logic), further debugging is impossible until the problem is corrected. In BDM, the fault can be temporarily bypassed so that its origin can be isolated and eliminated.

5.6.2.3 ENTERING BDM. When the processor detects a BKPT or a double bus fault or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE output. FREEZE assertion is the first indication that the processor has entered BDM. Once FREEZE has been asserted, the CPU enables the serial communication hardware and awaits a command.

5.7.3.6 IMMEDIATE ARITHMETIC/LOGIC INSTRUCTIONS. The immediate arithmetic/logic instruction table indicates the number of clock periods needed for the processor to fetch the source immediate data value and to perform the specified arithmetic/logic instruction using the specified addressing mode. Footnotes indicate when to account for the appropriate fetch effective or fetch immediate EA times. 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
MOVEQ	#, Dn	0	0	2(0/1/0)
ADDQ	#, Rn	0	0	2(0/1/0)
ADDQ	#, ⟨FEA⟩	0	3	5(0/1/x)
SUBQ	#, Rn	0	0	2(0/1/0)
SUBQ	#, ⟨FEA⟩	0	3	5(0/1/x)
ADDI	#, Rn	0	0	2(0/1/0)∗
ADDI	#, ⟨FEA⟩	0	3	5(0/1/x)∗
ANDI	#, Rn	0	0	2(0/1/0)∗
ANDI	#, ⟨FEA⟩	0	3	5(0/1/x)∗
EORI	#, Rn	0	0	2(0/1/0)∗
EORI	#, ⟨FEA⟩	0	3	5(0/1/x)∗
ORI	#, Rn	0	0	2(0/1/0)∗
ORI	#, ⟨FEA⟩	0	3	5(0/1/x)∗
SUBI	#, Rn	0	0	2(0/1/0)∗
SUBI	#, ⟨FEA⟩	0	3	5(0/1/x)∗
CMPI	#, Rn	0	0	2(0/1/0)∗
CMPI	#, ⟨FEA⟩	0	3	5(0/1/x)∗

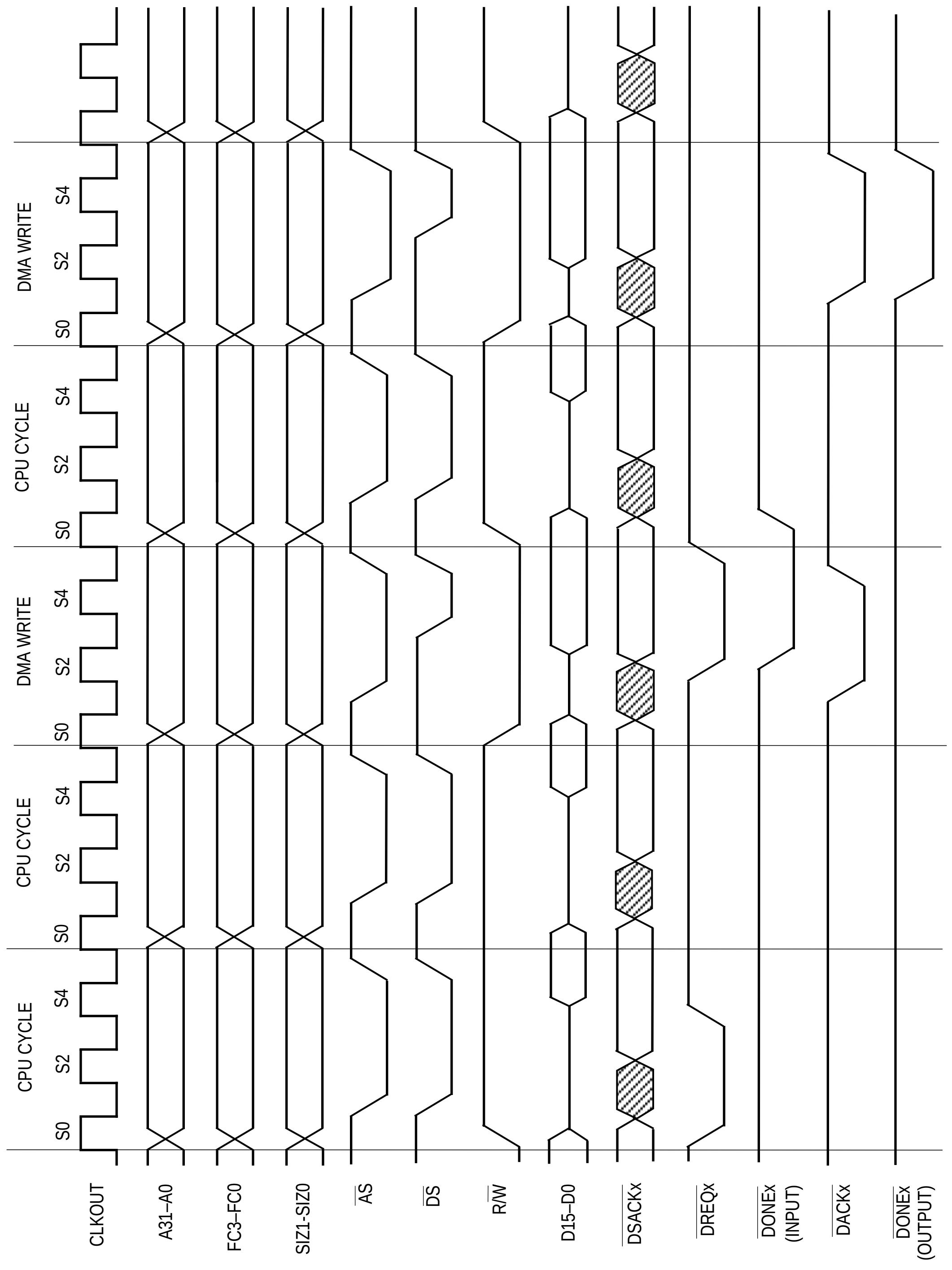
X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles.

∗ = An # fetch EA time must be added for this instruction: ⟨FEA⟩ +⟨FEA⟩ + ⟨OPER⟩

5.7.3.8 SINGLE OPERAND INSTRUCTIONS. The single operand instruction table indicates the number of clock periods needed for the processor to perform the specified operation using the specified addressing mode. 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
CLR	Dn	0	0	2(0/1/0)
CLR	⟨CEA⟩	0	2	4(0/1/x)
NEG	Dn	0	0	2(0/1/0)
NEG	⟨FEA⟩	0	3	5(0/1/x)
NEGX	Dn	0	0	2(0/1/0)
NEGX	⟨FEA⟩	0	3	5(0/1/x)
NOT	Dn	0	0	2(0/1/0)
NOT	⟨FEA⟩	0	3	5(0/1/x)
EXT	Dn	0	0	2(0/1/0)
NBCD	Dn	2	0	4(0/1/0)
NBCD	⟨FEA⟩	0	2	6(0/1/1)
Scc	Dn	2	0	4(0/1/0)
Scc	⟨CEA⟩	2	2	6(0/1/1)
TAS	Dn	4	0	6(0/1/0)
TAS	⟨CEA⟩	1	0	10(0/1/1)
TST	⟨FEA⟩	0	0	2(0/1/0)

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles.



NOTE:
1. DREQx must be active for two consecutive clocks for a DMA request to be recognized.
2. To cause another DMA transfer, DREQx is asserted after DACKx is asserted and before DACKx is negated.
3. DACKx and DONEx (DMA control signals) are asserted in the destination (write) DMA cycle.

Figure 6-8. Single-Address Write Timing (Cycle Steal)

```
* 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.
        MOVE.W      #$0742,DMAINT1(A0)

* Initialize channel status reg.
* Clear the DONE, BES, BED, CONF and BRKP bits to allow channel to startup.
        MOVE.B      #$7C,DMACSR1(A0)

* Initialize function code reg.
* DMA space, supervisor data space for source and destination.
        MOVE.B      #$DD,DMAFCR1(A0)

* Initialize source operand address
* Source address is equal to $6000.
        MOVE.L      SARADD,DMASAR1(A0)

* Initialize destination operand address
* Destination address is equal to $8000.
        MOVE.L      DARADD,DMADAR1(A0)

* Initialize the byte transfer count reg.
* The number of bytes to be transferred is $E or 7 words
        MOVE.L      NUMBYTE,DMABTC1(A0)

* Channel control reg. init. and Start DMA transfers
* No interrupts are enabled, destination (write) cycle. Increment source and
* destination addresses,source size is word, destination size is word.
* REQ is internal. 100% of bus bandwidth, dual-address transfers,
* start the DMA transfers.
        MOVE.W      #$0E8D,DMACCR1(A0)

****************************************************************************
        END
****************************************************************************
```

Example 3: Internal Request Generation, Memory Block Initialization.

```
****************************************************************************
* MC68340 basic DMA channel register initialization example code.
```

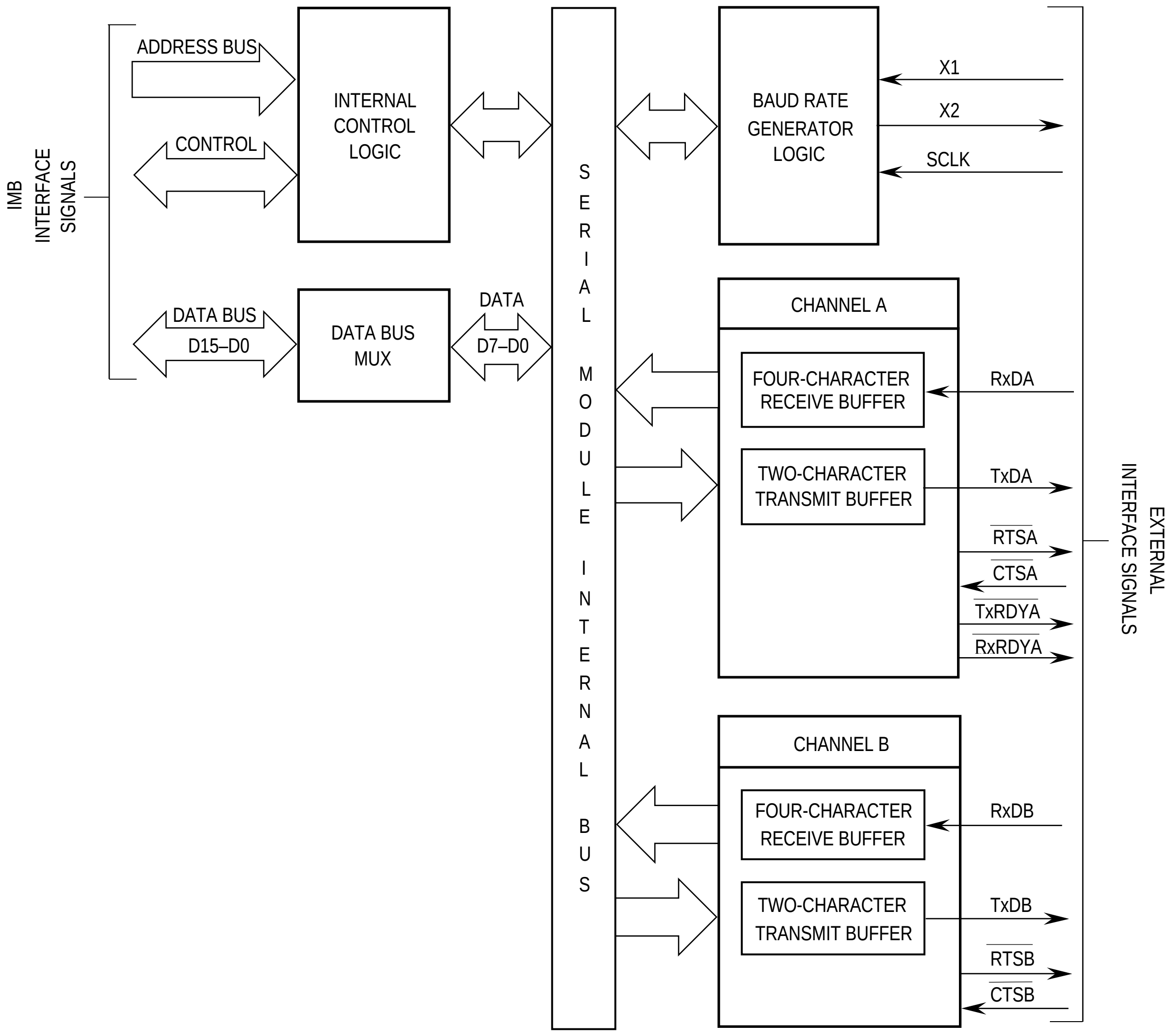

IMB INTERFACE SIGNALS
ADDRESS BUS
CONTROL
DATA BUS
D15–D0
INTERNAL CONTROL LOGIC
DATA BUS MUX
DATA
D7–D0
SERIAL MODULE INTERNAL BUS
BAUD RATE GENERATOR LOGIC
X1
X2
SCLK
CHANNEL A
FOUR-CHARACTER RECEIVE BUFFER
TWO-CHARACTER TRANSMIT BUFFER
RxDA
TxDA
RTSA
CTSA
TxRDYA
RxRDYA
CHANNEL B
FOUR-CHARACTER RECEIVE BUFFER
TWO-CHARACTER TRANSMIT BUFFER
RxDB
TxDB
RTSB
CTSB
EXTERNAL INTERFACE SIGNALS

Figure 7-2. External and Internal Interface Signals

7.2.1 Crystal Input or External Clock (X1)

This input is one of two connections to a crystal or a single connection to an external clock. A crystal or an external clock signal, at 3.6864 MHz, must be supplied when using the baud rate generator. If a crystal is used, a capacitor of approximately 10 pF should be connected from this signal to ground. If this input is not used, it must be connected to V_{CC} or GND. Refer to **Section 10 Applications** for an example of a clock driver circuit.

7.2.2 Crystal Output (X2)

This output is the additional connection to a crystal. If a crystal is used, a capacitor of approximately 5 pF should be connected from this signal to ground. If an external TTL-level clock is used on X1, the X2 output must be left open. Refer to **Section 10 Applications** for an example of a clock driver circuit.

RxRDY bit and loads the character into the receiver holding register FIFO stack provided the received A/D bit is a one (address tag). The character is discarded if the received A/D bit is a zero (data tag). If the receiver is enabled, all received characters are transferred to the CPU32 via the receiver holding register stack during read operations.

In either case, the data bits are loaded into the data portion of the stack while the A/D bit is loaded into the status portion of the stack normally used for a parity error (SR bit 5). Framing error, overrun error, and break detection operate normally. The A/D bit takes the place of the parity bit; therefore, parity is neither calculated nor checked. Messages in this mode may still contain error detection and correction information. One way to provide error detection, if 8-bit characters are not required, is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

7.3.5 Bus Operation

This section describes the operation of the IMB during read, write, and interrupt acknowledge cycles to the serial module. All serial module registers must be accessed as bytes.

7.3.5.1 READ CYCLES. The serial module is accessed by the CPU32 with no wait states. The serial module responds to byte reads. Reserved registers return logic zero during reads.

7.3.5.2 WRITE CYCLES. The serial module is accessed by the CPU32 with no wait states. The serial module responds to byte writes. Write cycles to read-only registers and reserved registers complete in a normal manner without exception processing; however, the data is ignored.

7.3.5.3 INTERRUPT ACKNOWLEDGE CYCLES. The serial module is capable of arbitrating for interrupt servicing and supplying the interrupt vector when it has successfully won arbitration. The vector number must be provided if interrupt servicing is necessary; thus, the interrupt vector register (IVR) must be initialized. If the IVR is not initialized, a spurious interrupt exception will be taken if interrupts are generated.

7.4 REGISTER DESCRIPTION AND PROGRAMMING

This section contains a detailed description of each register and its specific function as well as flowcharts of basic serial module programming.

7.4.1 Register Description

The operation of the serial module is controlled by writing control bytes into the appropriate registers. A list of serial module registers and their associated addresses are shown in Figure 7-9. The mode, status, command, and clock-select registers are duplicated for each channel to provide independent operation and control.

RC1–RC0—Receiver Commands

These bits select a single command as listed in Table 7-8.

Table 7-8. RCx Control Bits

RC1	RC0	Command
0	0	No Action Taken
0	1	Enable Receiver
1	0	Disable Receiver
1	1	Do Not Use

No Action Taken—The no action taken command causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.

Receiver Enable—The receiver enable command enables operation of the channel's receiver. If the serial module is not in multidrop mode, this command also forces the receiver into the search-for-start-bit state. If the receiver is already enabled, this command has no effect.

Receiver Disable—The receiver disable command disables the receiver immediately. Any character being received is lost. The command has no effect on the receiver status bits or any other control register. If the serial module is programmed to operate in the local loopback mode or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, this command has no effect.

Do Not Use—Do not use this bit combination because the result is indeterminate.

7.4.1.8 RECEIVER BUFFER (RB). The receiver buffer contains three receiver holding registers and a serial shift register. The channel's RxDx pin is connected to the serial shift register. The holding registers act as a FIFO. The CPU32 reads from the top of the stack while the receiver shifts and updates from the bottom of the stack when the shift register has been filled (see Figure 7-4). This register can only be read when the serial module is enabled (i.e., the STP bit in the MCR is cleared).

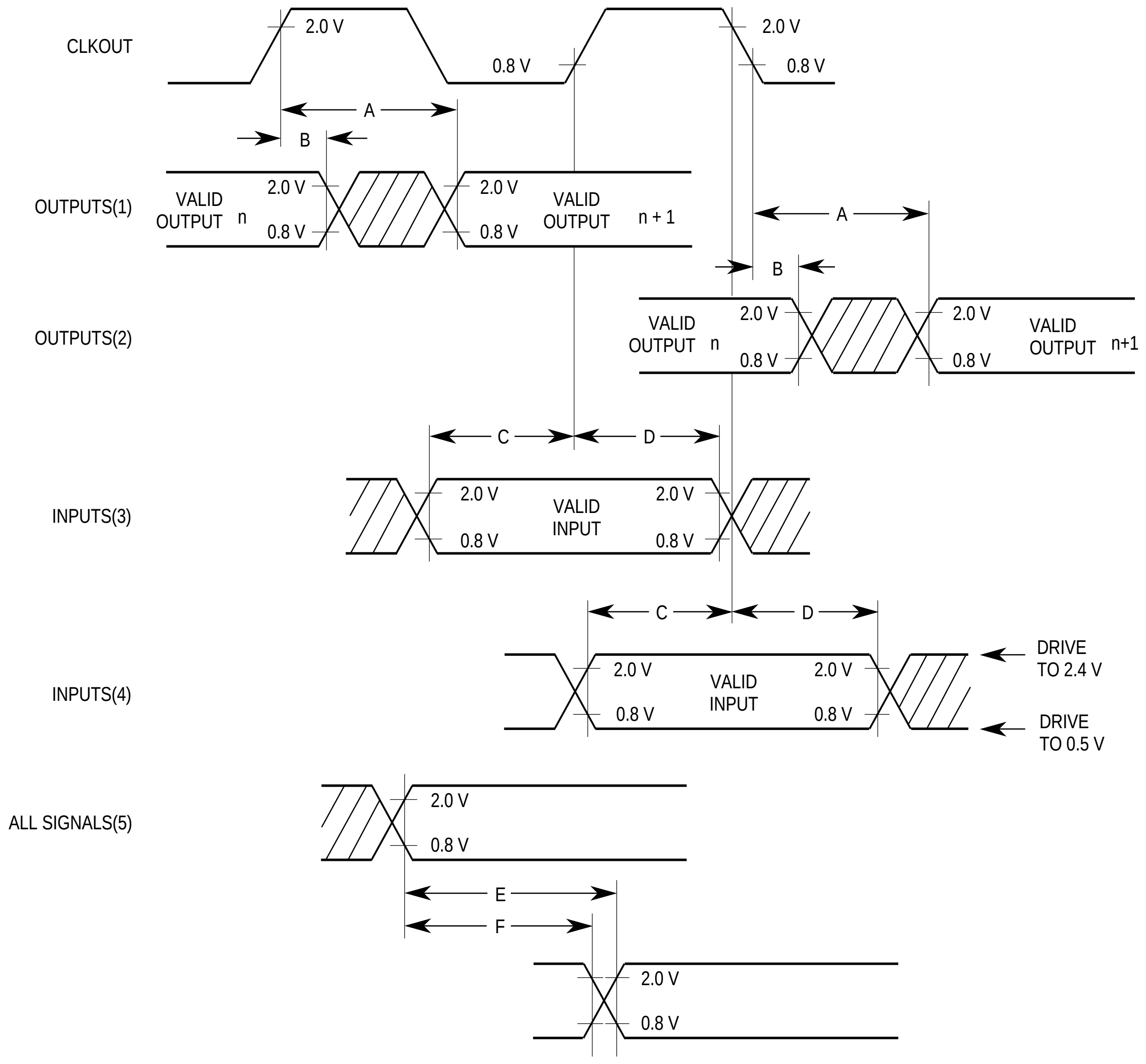
RBA, RBB $713, $71B

7	6	5	4	3	2	1	0
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
RESET:							
0	0	0	0	0	0	0	0

Read Only Supervisor/User

RB7–RB0—These bits contain the character in the receiver buffer.

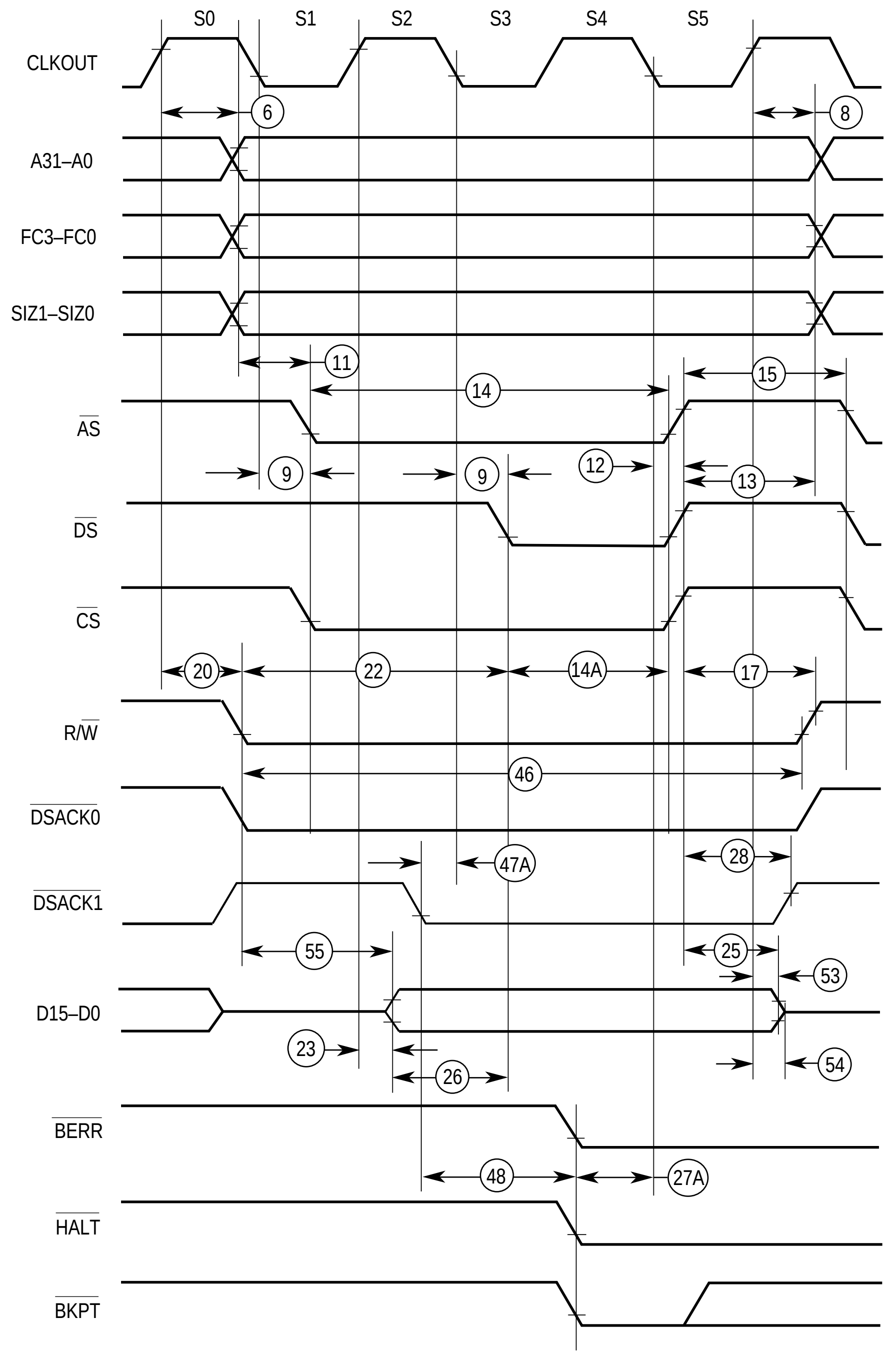
7.4.1.9 TRANSMITTER BUFFER (TB). The transmitter buffer consists of two registers, the transmitter holding register and the transmitter shift register (see Figure 7-4). The holding register accepts characters from the bus master if the TxRDY bit in the channel's SR is set. A write to the transmitter buffer clears the TxRDY bit, inhibiting any more



NOTES:
1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This output timing is applicable to all parameters specified relative to the falling edge of the clock.
3. This input timing is applicable to all parameters specified relative to the rising edge of the clock.
4. This input timing is applicable to all parameters specified relative to the falling edge of the clock.
5. This timing is applicable to all parameters specified relative to the assertion/negation of another signal.

LEGEND:
A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Signal valid to signal valid specification (maximum or minimum).
F. Signal valid to signal invalid specification (maximum or minimum).

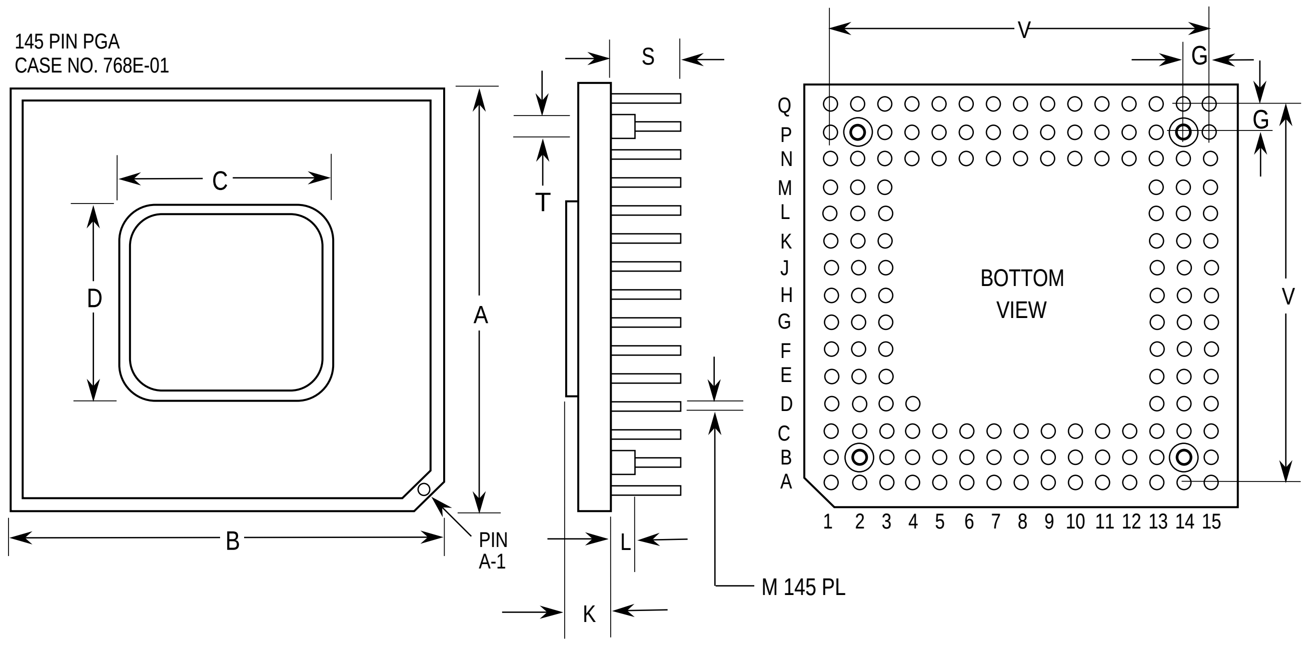
Figure 11-1. Drive Levels and Test Points for AC Specifications



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 11-3. Write Cycle Timing Diagram

12.3.2 RP Suffix



	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	39.37	39.88	1.550	1.570
B	39.37	39.88	1.550	1.570
C	22.75	22.97	0.895	0.905
D	22.75	22.97	0.895	0.905
G	2.54 BASIC		0.100 BASIC	
K	2.92	3.43	0.115	0.135
L	1.02	1.52	0.040	0.060
M	0.43	0.55	0.017	0.022
S	4.32	4.95	0.170	0.195
V	35.56 BASIC		1.400 BASIC	