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	68020
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	166MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (46x46)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68020ceh16e

1.2 PROGRAMMING MODEL

The programming model of the MC68020/EC020 consists of two groups of registers, the user model and the supervisor model, that correspond to the user and supervisor privilege levels, respectively. User programs executing at the user privilege level use the registers of the user model. System software executing at the supervisor level uses the control registers of the supervisor level to perform supervisor functions.

As shown in the programming models (see Figures 1-2 and 1-3), the MC68020/EC020 has 16 32-bit general-purpose registers, a 32-bit PC two 32-bit SSPs, a 16-bit SR, a 32-bit VBR, two 3-bit alternate function code registers, and two 32-bit cache handling (address and control) registers.

The user programming model remains unchanged from earlier M68000 family microprocessors. The supervisor programming model supplements the user programming model and is used exclusively by MC68020/EC020 system programmers who utilize the supervisor privilege level to implement sensitive operating system functions. The supervisor programming model contains all the controls to access and enable the special features of the MC68020/EC020. All application software, written to run at the nonprivileged user level, migrates to the MC68020/EC020 from any M68000 platform without modification.

Registers D7–D0 are data registers used for bit and bit field (1 to 32 bits), byte (8 bit), word (16 bit), long-word (32 bit), and quad-word (64 bit) operations. Registers A6–A0 and the USP, ISP, and MSP are address registers that may be used as software stack pointers or base address registers. Register A7 (shown as A7 in Figure 1-2 and as A7′ and A7″ in Figure 1-3) is a register designation that applies to the USP in the user privilege level and to either the ISP or MSP in the supervisor privilege level. In the supervisor privilege level, the active stack pointer (interrupt or master) is called the SSP. In addition, the address registers may be used for word and long-word operations. All of the 16 general-purpose registers (D7–D0, A7–A0) may be used as index registers.

The PC contains the address of the next instruction to be executed by the MC68020/EC020. During instruction execution and exception processing, the processor automatically increments the contents of the PC or places a new value in the PC, as appropriate.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

on top of the stack was generated by an interrupt, trap, or instruction exception, the RTE instruction restores the SR and PC to the values saved on the supervisor stack. The processor then continues execution at the restored PC address and at the privilege level determined by the S-bit of the restored SR. If the frame on top of the stack was generated by a bus fault (bus error or address error exception), the RTE instruction restores the entire saved processor state from the stack.

2.2 ADDRESS SPACE TYPES

The processor specifies a target address space for every bus cycle with the FC2–FC0 signals according to the type of access required. In addition to distinguishing between supervisor/user and program/data, the processor can identify special processor cycles, such as the interrupt acknowledge cycle, and the memory management unit can control accesses and translate addresses appropriately. Table 2-1 lists the types of accesses defined for the MC68020/EC020 and the corresponding values of the FC2–FC0 signals.

Table 2-1. Address Space Encodings

FC2	FC1	FC0	Address Space
0	0	0	(Undefined, Reserved)*
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	(Undefined, Reserved)*
1	0	0	(Undefined, Reserved)*
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

* Address space 3 is reserved for user definition; 0 and 4 are reserved for future use by Motorola.

The memory locations of user program and data accesses are not predefined; neither are the locations of supervisor data space. During reset, the first two long words beginning at memory location zero in the supervisor program space are used for processor initialization. No other memory locations are explicitly defined by the MC68020/EC020.

A function code of $7 selects the CPU address space. This is a special address space that does not contain instructions or operands but is reserved for special processor functions. The processor uses accesses in this space to communicate with external devices for special purposes. For example, all M68000 processors use the CPU space for interrupt acknowledge cycles. The MC68020/EC020 also generate CPU space accesses for breakpoint acknowledge and coprocessor operations.

Supervisor programs can use the MOVES instruction to access all address spaces, including the user spaces and the CPU address space. Although the MOVES instruction can be used to generate CPU space cycles, this may interfere with proper system operation. Thus, the use of MOVES to access the CPU space should be done with caution.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

F—Freeze Cache

The F-bit is set to freeze the instruction cache. When the F-bit is set and a cache miss occurs, the entry (or line) is not replaced. When the F-bit is clear, a cache miss causes the entry (or line) to be filled. A reset operation clears the F-bit.

E—Enable Cache

The E-bit is set to enable the instruction cache. When it is clear, the instruction cache is disabled. A reset operation clears the E-bit. The supervisor normally enables the instruction cache, but it can clear the E-bit for system debugging or emulation, as required. Disabling the instruction cache does not flush the entries. If the cache is reenabled, the previously valid entries remain valid and may be used.

4.3.2 Cache Address Register (CAAR)

The format of the 32-bit CAAR is shown in Figure 4-3.

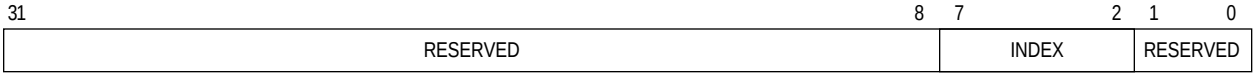


Figure 4-3. Cache Address Register

Bits 31–8, 1, and 0—Reserved

These bits are reserved for use by Motorola.

Index Field

The index field contains the address for the “clear cache entry” operations. The bits of this field, which correspond to A7–A2, specify the index and a long word of a cache line.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 5-5 lists the combinations of SIZ1, SIZ0, A1, and A0 and the corresponding pattern of the data transfer for write cycles from the internal multiplexer of the MC68020/EC020 to the external data bus.

Table 5-5. MC68020/EC020 Internal to External Data Bus Multiplexer—Write Cycles

Transfer Size	Size		Address		External Data Bus Connection			
	SIZ1	SIZ0	A1	A0	D31–D24	D23–D16	D15–D8	D7–D0
Byte	0	1	x	x	OP3	OP3	OP3	OP3
Word	1	0	x	0	OP2	OP3	OP2	OP3
	1	0	x	1	OP2	OP2	OP3	OP2
3 Bytes	1	1	0	0	OP1	OP2	OP3	OP0*
	1	1	0	1	OP1	OP1	OP2	OP3
	1	1	1	0	OP1	OP2	OP1	OP2
	1	1	1	1	OP1	OP1	OP2*	OP1
Long Word	0	0	0	0	OP0	OP1	OP2	OP3
	0	0	0	1	OP0	OP0	OP1	OP2
	0	0	1	0	OP0	OP1	OP0	OP1
	0	0	1	1	OP0	OP0	OP1*	OP0

*Due to the current implementation, this byte is output but never used.
x = Don't care
NOTE: The OP tables on the external data bus refer to a particular byte of the operand that is written on that section of the data bus.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Figure 5-7 shows a word write to an 8-bit bus port. Like the preceding example, this example requires two bus cycles. Each bus cycle transfers a single byte. SIZ1 and SIZ0 for the first cycle specify two bytes; for the second cycle, one byte. Figure 5-8 shows the associated bus transfer signal timing.

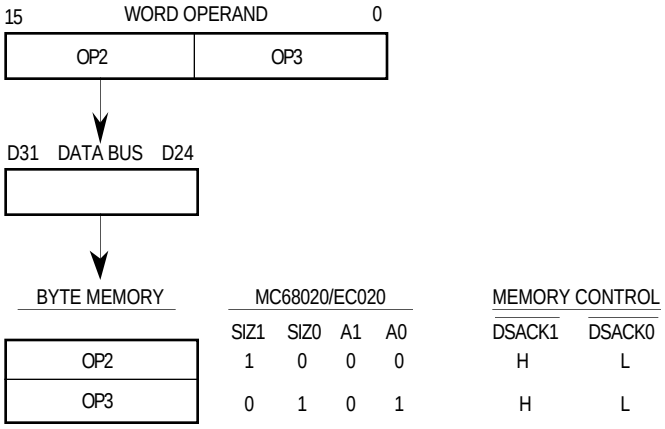
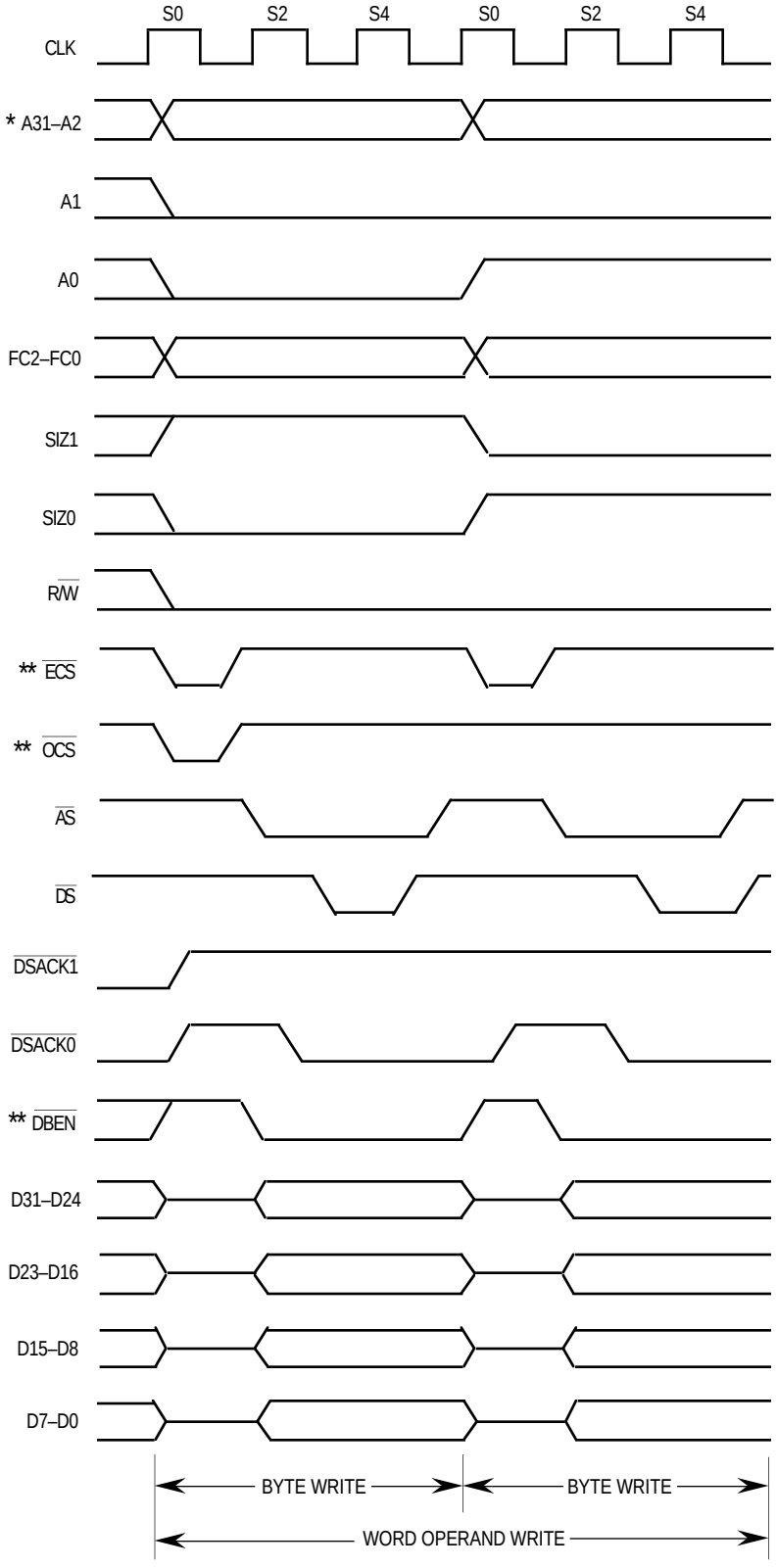


Figure 5-7. Word Operand Write to Byte Port Example

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com



* For the MC68EC020, A23–A2.
** This signal does not apply to the MC68EC020.

Figure 5-8. Word Operand Write to Byte Port Timing

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

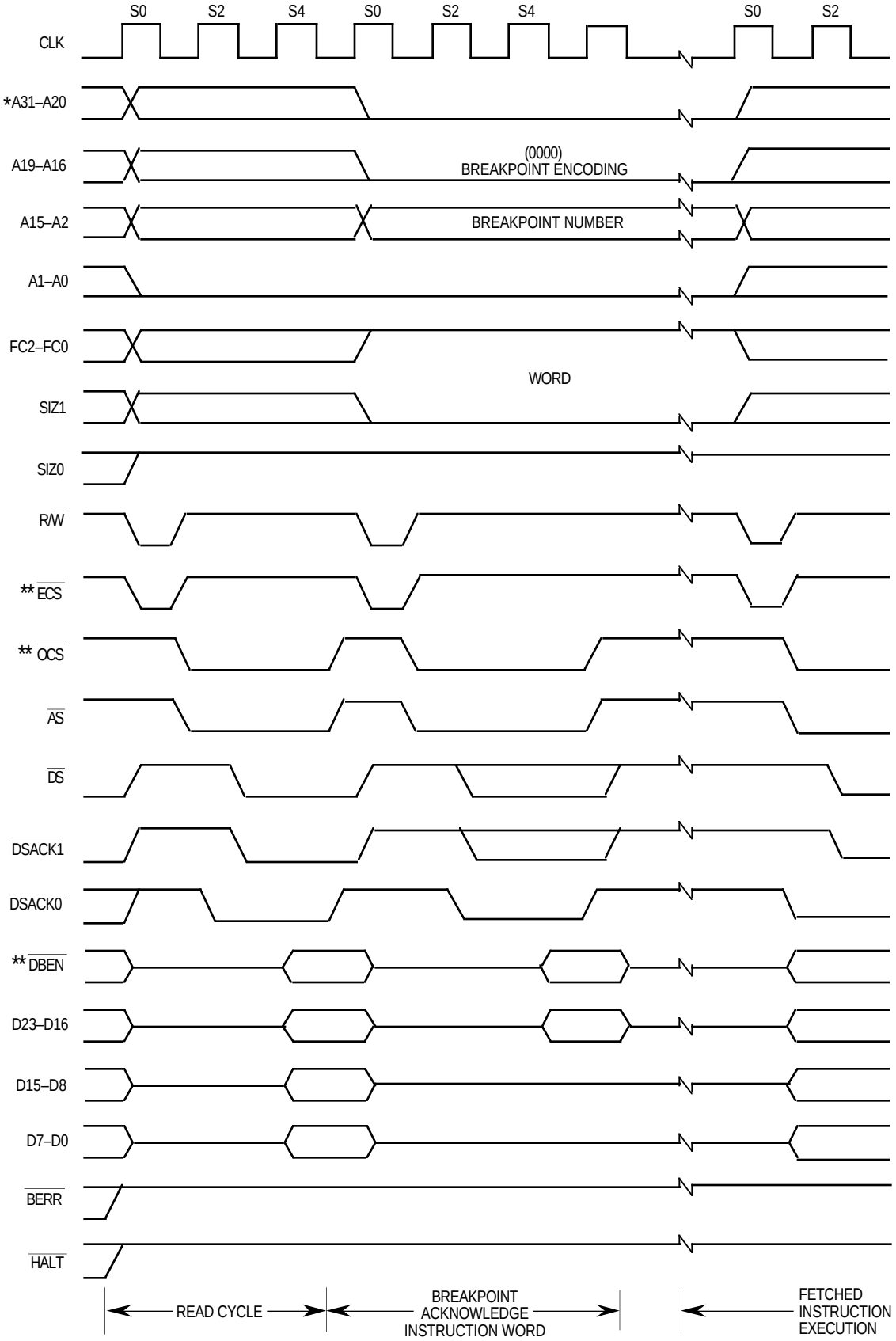


Figure 5-36. Breakpoint Acknowledge Cycle Timing

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

5.4.3 Coprocessor Communication Cycles

The MC68020/EC020 coprocessor interface provides instruction-oriented communication between the processor and as many as eight coprocessors. Coprocessor accesses use the MC68020/EC020 bus protocol except that the address bus supplies access information rather than a 32-bit address. The CPU space type field (A19–A16) for a coprocessor operation is 0010. A15–A13 contain the coprocessor identification number (CpID), and A5–A0 specify the coprocessor interface register to be accessed. The memory management unit of an MC68020/EC020 system is always identified by a CpID of zero and has an extended register select field (A7–A0) in CPU space 0001 for use by the CALLM and RTM access level checking mechanism. Refer to **Section 9 Applications Information** for more details.

5.5 BUS EXCEPTION CONTROL CYCLES

The MC68020/EC020 bus architecture requires assertion of $\overline{\text{DSACK1}}$/$\overline{\text{DSACK0}}$ from an external device to signal that a bus cycle is complete. $\overline{\text{DSACK1}}$/$\overline{\text{DSACK0}}$ or $\overline{\text{AVEC}}$ is not asserted if:

- The external device does not respond,
- No interrupt vector is provided, or
- Various other application-dependent errors occur.

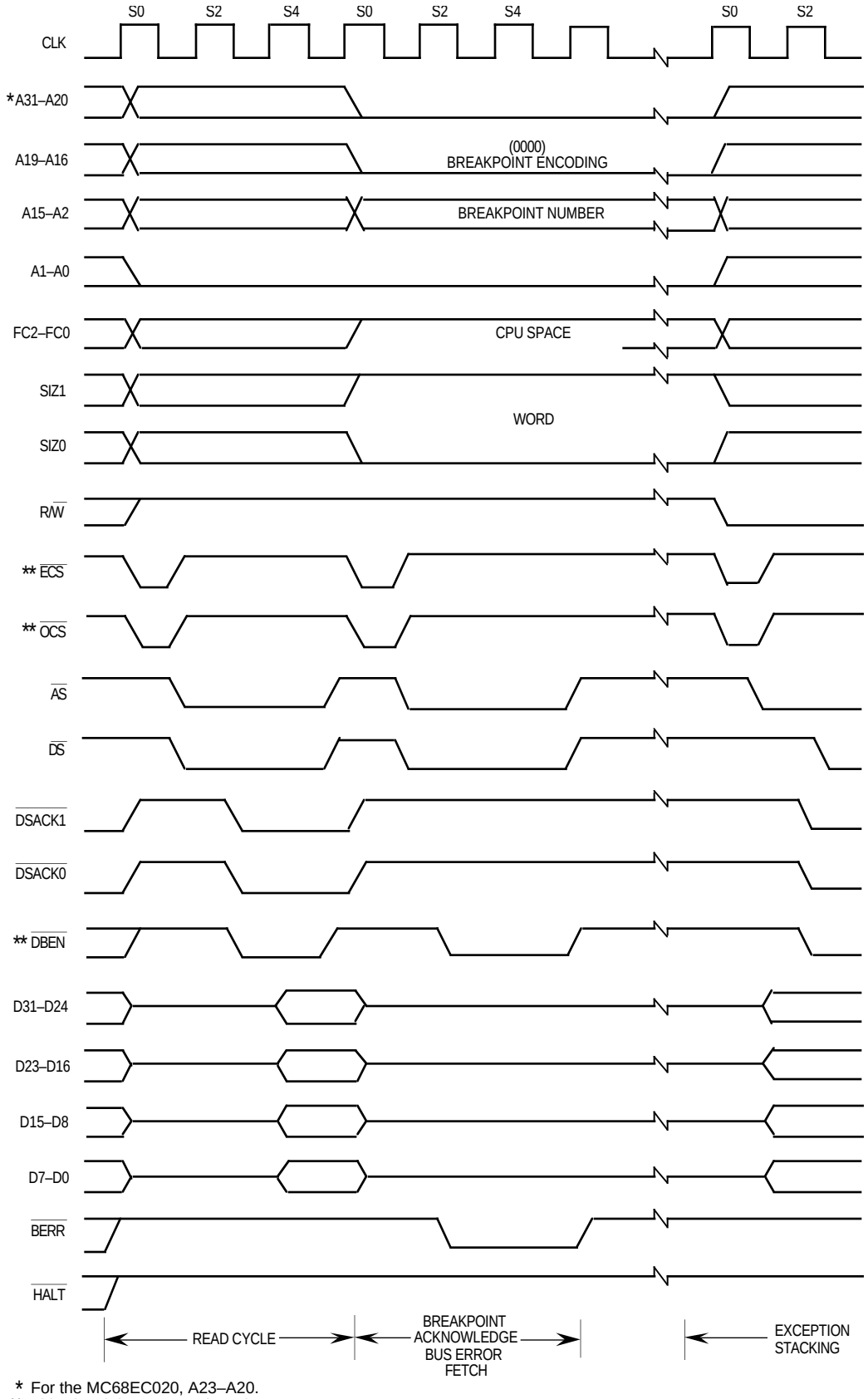
External circuitry can assert $\overline{\text{BERR}}$ when no device responds by asserting $\overline{\text{DSACK1}}$/$\overline{\text{DSACK0}}$ or $\overline{\text{AVEC}}$ within an appropriate period of time after the processor asserts $\overline{\text{AS}}$. Assertion of $\overline{\text{BERR}}$ allows the cycle to terminate and the processor to enter exception processing for the error condition.

$\overline{\text{HALT}}$ is also used for bus exception control. $\overline{\text{HALT}}$ can be asserted by an external device for debugging purposes to cause single bus cycle operation or can be asserted in combination with $\overline{\text{BERR}}$ to cause a retry of a bus cycle in error.

To properly control termination of a bus cycle for a retry or a bus error condition, $\overline{\text{DSACK1}}$/$\overline{\text{DSACK0}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ can be asserted and negated with the rising edge of the MC68020/EC020 clock. This procedure ensures that when two signals are asserted simultaneously, the required setup time (#47A) and hold time (#47B) for both of them is met for the same falling edge of the processor clock. (Refer to **Section 10 Electrical Characteristics** for timing requirements.) This or some equivalent precaution should be designed into the external circuitry that provides these signals.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com



* For the MC68EC020, A23–A20.
** This signal does not apply to the MC68EC020.

Figure 5-38. Bus Error without DSACK1/DSACK0

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

5.7.1 MC68020 Bus Arbitration

The sequence of the MC68020 bus arbitration protocol is as follows:

1. An external device asserts the $\overline{BR}$ signal.
2. The processor asserts the $\overline{BG}$ signal to indicate that the bus will become available at the end of the current bus cycle.
3. The external device asserts the $\overline{BGACK}$ signal to indicate that it has assumed bus mastership.

$\overline{BR}$ may be issued any time during a bus cycle or between cycles. $\overline{BG}$ is asserted in response to $\overline{BR}$; it is usually asserted as soon as $\overline{BR}$ has been synchronized and recognized, except when the MC68020 has made an internal decision to execute a bus cycle. Then, the assertion of $\overline{BG}$ is deferred until the bus cycle has begun. Additionally, $\overline{BG}$ is not asserted until the end of a read-modify-write operation (when $\overline{RMC}$ is negated) in response to a $\overline{BR}$ signal. When the requesting device receives $\overline{BG}$ and more than one external device can be bus master, the requesting device should begin whatever arbitration is required. The external device asserts $\overline{BGACK}$ when it assumes bus mastership, and maintains $\overline{BGACK}$ during the entire bus cycle (or cycles) for which it is bus master. The following conditions must be met for an external device to assume mastership of the bus through the normal bus arbitration procedure:

- The external device must have received $\overline{BG}$ through the arbitration process.
- $\overline{AS}$ must be negated, indicating that no bus cycle is in progress, and the external device must ensure that all appropriate processor signals have been placed in the high-impedance state (by observing specification #7 in **Section 10 Electrical Specifications**).
- The termination signal ($\overline{DSACK1}$/$\overline{DSACK0}$) for the most recent cycle must have been negated, indicating that external devices are off the bus (optional, refer to **5.7.1.3 Bus Grant Acknowledge (MC68020)**).
- $\overline{BGACK}$ must be inactive, indicating that no other bus master has claimed ownership of the bus.

Figure 5-42 is a flowchart of MC68020 bus arbitration for a single device. Figure 5-43 is a timing diagram for the same operation. This technique allows processing of bus requests during data transfer cycles.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 6-1. Exception Vector Assignments

Vector Number	Vector Offset		Assignment
	Hex	Space	
0	000	SP	Reset Initial Interrupt Stack Pointer
1	004	SP	Reset Initial Program Counter
2	008	SD	Bus Error
3	00C	SD	Address Error
4	010	SD	Illegal Instruction
5	014	SD	Zero Divide
6	018	SD	CHK, CHK2 Instruction
7	01C	SD	cpTRAPcc, TRAPcc, TRAPV Instructions
8	020	SD	Privilege Violation
9	024	SD	Trace
10	028	SD	Line 1010 Emulator
11	02C	SD	Line 1111 Emulator
12	030	SD	(Unassigned, Reserved)
13	034	SD	Coprocessor Protocol Violation
14	038	SD	Format Error
15	03C	SD	Uninitialized Interrupt
16–23	040 05C	SD SD	Unassigned, Reserved
24	060	SD	Spurious Interrupt
25	064	SD	Level 1 Interrupt Autovector
26	068	SD	Level 2 Interrupt Autovector
27	06C	SD	Level 3 Interrupt Autovector
28	070	SD	Level 4 Interrupt Autovector
29	074	SD	Level 5 Interrupt Autovector
30	078	SD	Level 6 Interrupt Autovector
31	07C	SD	Level 7 Interrupt Autovector
32–47	080 0BC	SD SD	TRAP #0–15 Instruction Vectors
48	0C0	SD	FPCP Branch or Set on Unordered Condition
49	0C4	SD	FPCP Inexact Result
50	0C8	SD	FPCP Divide by Zero
51	0CC	SD	FPCP Underflow
52	0D0	SD	FPCP Operand Error
53	0D4	SD	FPCP Overflow
54	0D8	SD	FPCP Signaling NAN
55	0DC	SD	Unassigned, Reserved
56	0E0	SD	PMMU Configuration
57	0E4	SD	PMMU Illegal Operation
58	0E8	SD	PMMU Access Level Violation
59–63	0EC 0FC	SD SD	Unassigned, Reserved
64–255	100 3FC	SD SD	User-Defined Vectors (192)

SP—Supervisor Program Space

SD—Supervisor Data Space

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

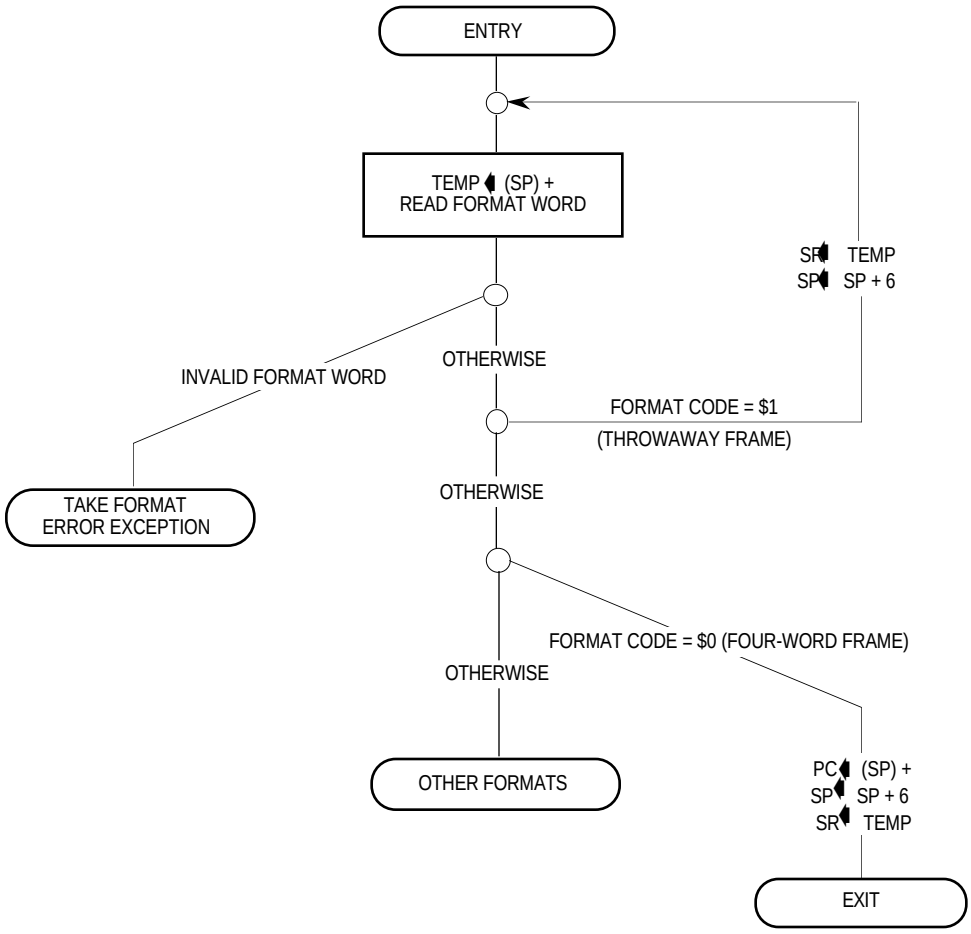


Figure 6-7. RTE Instruction for Throwaway Four-Word Frame

For the coprocessor midinstruction stack frame, the processor reads the SR, PC, instruction address, internal register values, and the evaluated effective address from the stack, restores these values to the corresponding internal registers, and increments the stack pointer by 20. The processor then reads from the response register of the coprocessor that initiated the exception to determine the next operation to be performed. Refer to **Section 7 Coprocessor Interface Description** for details of coprocessor-related exceptions.

For both the short and long bus fault stack frames, the processor first checks the format value on the stack for validity. In addition, for the long stack frame, the processor compares the version number in the stack with its own version number. The version number is located in the most significant nibble (bits 15–12) of the word at location SP + $36 in the long stack frame. This validity check is required in a multiprocessor system to ensure that the data is properly interpreted by the RTE instruction. The RTE instruction also reads from both ends of the stack frame to make sure it is accessible. If the frame is invalid or inaccessible, the processor takes a format error or a bus error exception, respectively. Otherwise, the processor reads the entire frame into the proper internal registers, deallocates the stack, and resumes normal processing. Once the processor begins to load the frame to restore its internal state, the assertion of the BERR signal

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

stack for stage B of the pipe are accepted as valid; the processor assumes that there is no prefetch pending for stage B and that software has repaired or filled the image of stage B, if necessary.

1 = Rerun faulted bus cycle or run pending prefetch
0 = Do not rerun bus cycle

Bits 11–9—Reserved by Motorola

DF—Fault/Rerun Flag

If the DF bit is set, a data fault has occurred and caused the exception. If the DF bit is set when the processor reads the stack frame, it reruns the faulted data access; otherwise, it assumes that the data input buffer value on the stack is valid for a read or that the data has been correctly written to memory for a write (or that no data fault occurred).

1 = Rerun faulted bus cycle or run pending prefetch
0 = Do not rerun bus cycle

RM—Read-Modify-Write

1 = Read-modify-write operation on data cycle
0 = Not a read-modify-write operation

RW—Read/Write

1 = Read on data cycle
0 = Write on data cycle

SIZE—Size Code

The SIZE field indicates the size of the operand access for the data cycle.

Bit 3—Reserved by Motorola

FC2–FC0—Specifies the address space for data cycle

6.2.2 Using Software to Complete the Bus Cycles

One method of completing a faulted bus cycle is to use a software handler to emulate the cycle. This is the only method for correcting address errors. The handler should emulate the faulted bus cycle in a manner that is transparent to the instruction that caused the fault. For instruction stream faults, the handler may need to run bus cycles for both the B and C stages of the instruction pipe. The RB and RC bits of the SSW identify the stages that may require a bus cycle; the FB and FC bits of the SSW indicate that a stage was invalid when an attempt was made to use its contents. Those stages must be repaired. For each faulted stage, the software handler should copy the instruction word from the proper address space as indicated by the S-bit of the copy of the SR saved on the stack to the image of the appropriate stage in the stack frame. In addition, the handler must clear the RB or RC bit associated with the stage that it has corrected. The handler should not change the FB and FC bits.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

information obtained into memory until all the bytes specified in the coprocessor format word have been transferred. Following a cpSAVE instruction, the coprocessor should be in an idle state—that is, not executing any coprocessor instructions.

The cpSAVE instruction is a privileged instruction. When the MC68020/EC020 identifies a cpSAVE instruction, it checks the S-bit in the SR to determine whether it is operating at the supervisor privilege level. If the MC68020/EC020 attempts to execute a cpSAVE instruction while at the user privilege level (S-bit in the SR is clear), it initiates privilege violation exception processing without accessing any of the CIRs (refer to **7.5.2.3 Privilege Violations**).

The MC68020/EC020 initiates format error exception processing if it reads an invalid format word (or a valid format word whose length field is not a multiple of four bytes) from the save CIR during the execution of a cpSAVE instruction (refer to **7.2.3.2.3 Invalid Format Word**). The MC68020/EC020 writes an abort mask (refer to **7.2.3.2.3 Invalid Format Word**) to the control CIR to abort the coprocessor instruction prior to beginning exception processing. Figure 7-16 does not include this case since a coprocessor usually returns either a not-ready or a valid format code in the context of the cpSAVE instruction. The coprocessor can return the invalid format word, however, if a cpSAVE is initiated while the coprocessor is executing a cpSAVE or cpRESTORE instruction and the coprocessor is unable to support the suspension of these two instructions.

7.2.3.4 COPROCESSOR CONTEXT RESTORE INSTRUCTION. The M68000 coprocessor context restore instruction category includes one instruction. The coprocessor context restore instruction, denoted by the cpRESTORE mnemonic, forces a coprocessor to terminate any current operations and to restore a former state. During execution of a cpRESTORE instruction, the coprocessor can communicate status information to the main processor by placing format codes in the restore CIR.

7.2.3.4.1 Format. Figure 7-17 shows the format of the cpRESTORE instruction.

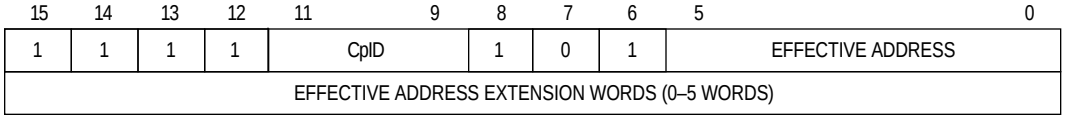


Figure 7-17. Coprocessor Context Restore Instruction Format (cpRESTORE)

The first word of the instruction, the F-line operation word, contains the CpID code in bits 11–9 and an M68000 effective addressing code in bits 5–0. The effective address encoded in the cpRESTORE instruction is the starting address in memory where the coprocessor context is stored. The effective address is that of the coprocessor format word that applies to the context to be restored to the coprocessor.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

If SP = 0 and DR = 0, the main processor writes the 16-bit SR value to the operand CIR. If SP = 0 and DR = 1, the main processor reads a 16-bit value from the operand CIR into the main processor SR.

If SP = 1 and DR = 0, the main processor writes the long-word value in the scanPC to the instruction address CIR and then writes the SR value to the operand CIR. If SP = 1 and DR = 1, the main processor reads a 16-bit value from the operand CIR into the SR and then reads a long-word value from the instruction address CIR into the scanPC.

With this primitive, a general category instruction can change the main processor program flow by placing a new value in the SR, in the scanPC, or new values in both the SR and the scanPC. By accessing the SR, the coprocessor can determine and manipulate the main processor condition codes, supervisor status, trace modes, selection of the active stack, and interrupt mask level.

The MC68020/EC020 discards any instruction words that have been prefetched beyond the current scanPC location when this primitive is issued with DR = 1 (transfer to main processor). The MC68020/EC020 then refills the instruction pipe from the scanPC address in the address space indicated by the S-bit of the SR.

If the MC68020/EC020 is operating in the trace on change of flow mode (T1, T0 in the SR = 01) when the coprocessor instruction begins to execute and if this primitive is issued with DR = 1 (from coprocessor to main processor), the MC68020/EC020 prepares to take a trace exception. The trace exception occurs when the coprocessor signals that it has completed all processing associated with the instruction. Changes in the trace modes due to the transfer of the SR to the main processor take effect on execution of the next instruction.

7.4.18 Take Preinstruction Exception Primitive

The take preinstruction exception primitive initiates exception processing using a coprocessor-supplied exception vector number and the preinstruction exception stack frame format. This primitive applies to general and conditional category instructions. Figure 7-40 shows the format of the take preinstruction exception primitive.

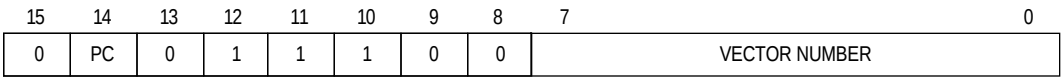


Figure 7-40. Take Preinstruction Exception Primitive Format

The take preinstruction exception primitive uses the PC bit as described in **7.4.2 Coprocessor Response Primitive General Format**. The vector number field contains the exception vector number used by the main processor to initiate exception processing.

When the main processor receives this primitive, it acknowledges the coprocessor exception request by writing an exception acknowledge mask to the control CIR (refer to **7.3.2 Control CIR**). The MC68020/EC020 then proceeds with exception processing as

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

Example 2

Using the same instruction stream, the second example demonstrates the different effects of instruction execution overlap on instruction timing when the same instructions are positioned slightly differently in 32-bit memory:

Address			
	n	MOVE #1	ADD #2
	n + 4	MOVE #3	ADD #4
	n + 8	•••	•••

The assumptions for example 2 (see Figure 8-4) are:

1. The data bus is 32 bits,
2. The first instruction is prefetched from an even-word address,
3. Memory access occurs with no wait states, and
4. The cache is disabled.

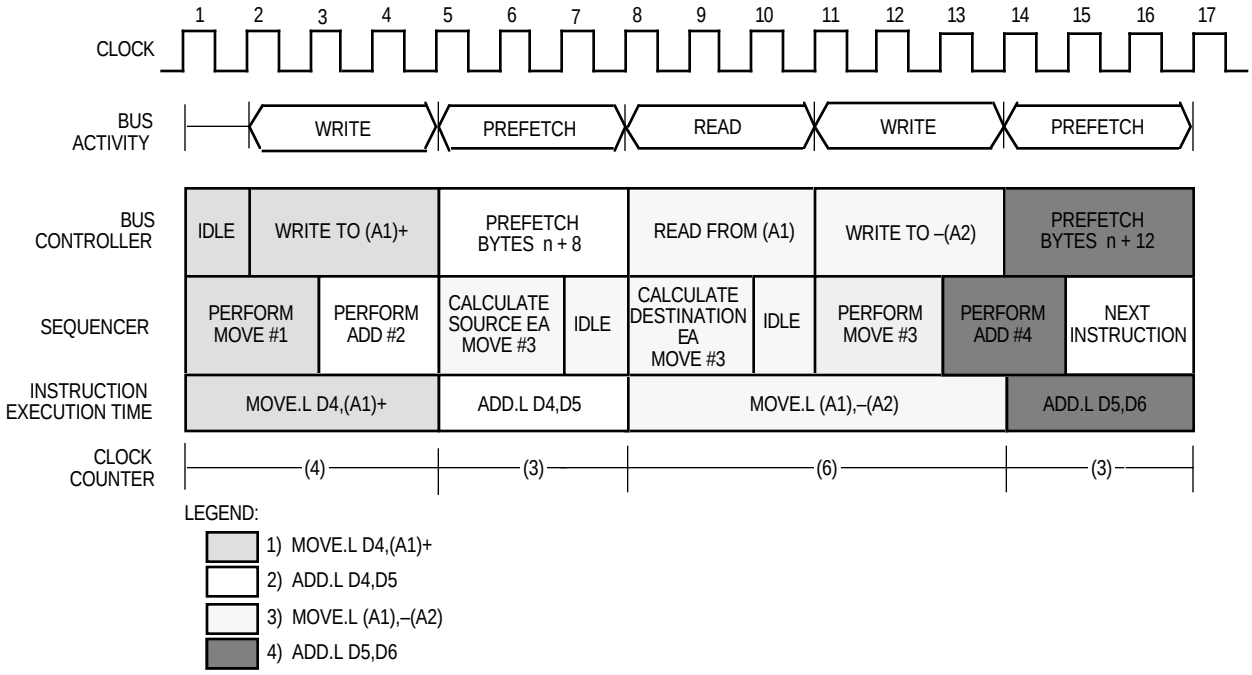


Figure 8-4. Processor Activity for Example 2

Although the total execution time of the instruction segment does not change in this example, the individual instruction times are significantly different. This example demonstrates that the effects of overlap are not only instruction-sequence dependent but are also dependent upon the alignment of the instruction stream in memory.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

8.2.7 Special-Purpose MOVE Instruction

The special-purpose MOVE instruction table indicates the number of clock periods needed for the processor to fetch, calculate, and perform the special-purpose MOVE operation on the control registers or specified effective address. The total number of clock cycles is outside the parentheses, the number of read, prefetch, and write cycles is given inside the parentheses as (r/p/w). These cycles are included in the total clock cycle number.

	Instruction		Best Case	Cache Case	Worst Case
	EXG	Ry,Rx	**0**(0/0/0)	**2**(0/0/0)	**3**(0/1/0)
	MOVEC	Cr,Rn	**3**(0/0/0)	**6**(0/0/0)	**7**(0/1/0)
	MOVEC	Rn,Cr	**9**(0/0/0)	**12**(0/0/0)	**13**(0/1/0)
	MOVE	PSW,Rn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
†	MOVE	PSW,Mem	**5**(0/0/1)	**5**(0/0/1)	**7**(0/1/1)
*	MOVE	EA,CCR	**4**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
*	MOVE	EA,SR	**8**(0/0/0)	**8**(0/0/0)	**11**(0/2/0)
‡	MOVEM	EA,RL	**8 + 4n**(n/0/0)	**8 + 4n**(n/0/0)	**9 + 4n**(n/1/0)
‡	MOVEM	RL,EA	**4 + 3n**(0/0/n)	**4 + 3n**(0/0/n)	**5 + 3n**(0/1/n)
	MOVEP.W	Dn,(d_{16},An)	**8**(0/0/2)	**11**(0/0/2)	**11**(0/1/2)
	MOVEP.L	Dn,(d_{16},An)	**14**(0/0/4)	**17**(0/0/4)	**17**(0/1/4)
	MOVEP.W	(d_{16},An),Dn	**10**(2/0/0)	**12**(2/0/0)	**12**(2/1/0)
	MOVEP.L	(d_{16},An),Dn	**16**(4/0/0)	**18**(4/0/0)	**18**(4/1/0)
‡	MOVES	EA,Rn	**7**(1/0/0)	**7**(1/0/0)	**8**(1/1/0)
‡	MOVES	Rn,EA	**5**(0/0/1)	**5**(0/0/1)	**7**(0/1/1)
	MOVE	USP	**0**(0/0/0)	**2**(0/0/0)	**3**(0/1/0)
	SWAP	Rx,Ry	**1**(0/0/0)	**4**(0/0/0)	**4**(0/1/0)

n—Number of Registers to Transfer
RL—Register List
*Add Fetch Effective Address Time
†Add Calculate Effective Address Time
‡Add Calculate Immediate Address Time

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

8.2.13 Bit Manipulation Instructions

The bit manipulation instructions table indicates the number of clock periods needed for the processor to perform the specified bit operation on the given addressing mode. Footnotes indicate when it is necessary to add another table entry to calculate the total effective execution time for the instruction. The total number of clock cycles is outside the parentheses; the number of read, prefetch, and write cycles is given inside the parentheses as (r/p/w). These cycles are included in the total clock cycle number.

Instruction			Best Case	Cache Case	Worst Case
	BTST	#<data>,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
	BTST	Dn,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
**	BTST	#<data>,Mem	**4**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
*	BTST	Dn,Mem	**4**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
	BCHG	#<data>,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
	BCHG	Dn,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
**	BCHG	#<data>,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)
*	BCHG	Dn,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)
	BCLR	#<data>,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
	BCLR	Dn,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
**	BCLR	#<data>,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)
*	BCLR	Dn,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)
	BSET	#<data>,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
	BSET	Dn,Dn	**1**(0/0/0)	**4**(0/0/0)	**5**(0/1/0)
**	BSET	#<data>,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)
*	BSET	Dn,Mem	**4**(0/0/1)	**4**(0/0/1)	**5**(0/1/1)

*Add Fetch Effective Address Time

**Add Fetch Immediate Address Time

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com