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	68040
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	33MHz
Co-Processors/DSP	-
RAM Controllers	-
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	184-BCQFP
Supplier Device Package	184-CQFP (31.3x31.3)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68040fe33a

the logical addresses of the currently executing process. Portions of translation tables can be dynamically allocated as the process requires additional memory.

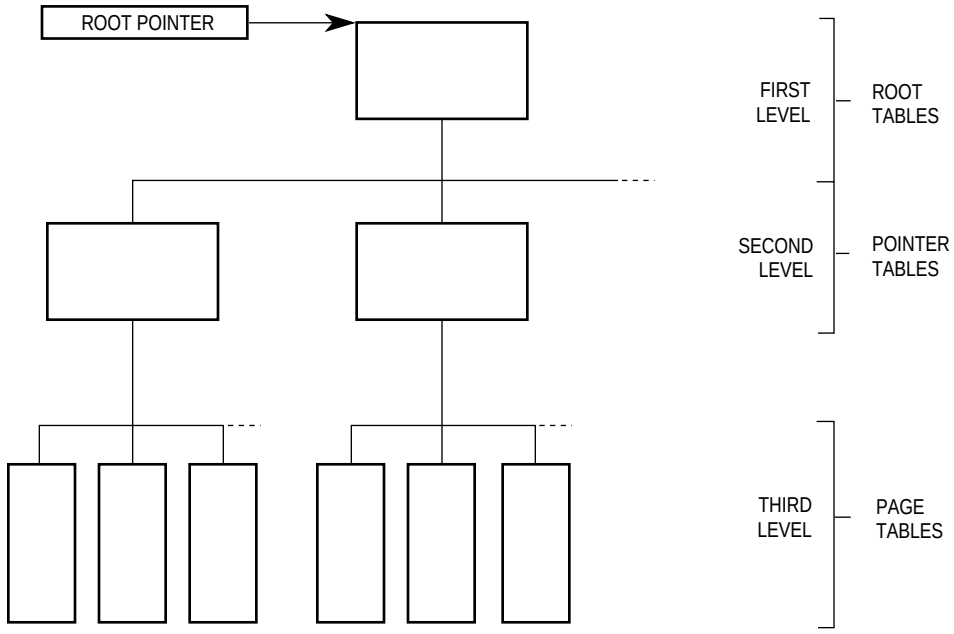
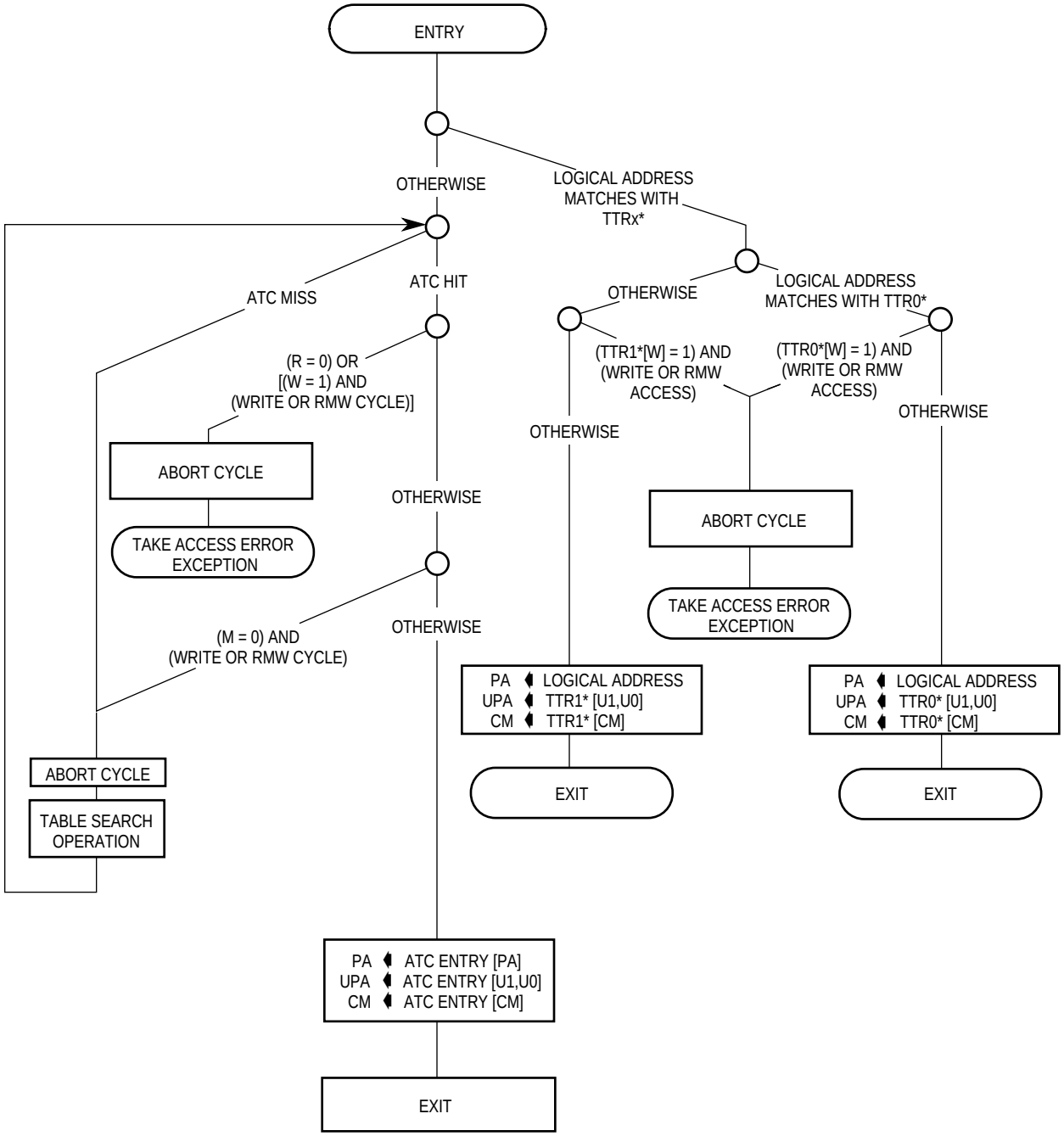


Figure 3-7. Translation Table Structure

The current privilege mode determines the use of the URP or SRP for translation of the access. The root pointer contains the base address of the translation table's root-level table. The translation table consists of tables of descriptors. The table descriptors of the root- and pointer-levels can be either resident or invalid. The page descriptors of the page-level table can be resident, indirect, or invalid. A page descriptor defines the physical address of a page frame in memory that corresponds to the logical address of a page. An indirect descriptor, which contains a pointer to the actual page descriptor, can be used when two or more logical addresses access a single page descriptor.

The table search uses logical addresses to access the translation tables. Figure 3-8 illustrates a logical address format, which is segmented into four fields: root index (RI), pointer index (PI), page index (PGI), and page offset. The first three fields extracted from the logical address index the base address for each table level. The seven bits of the logical address RI field are multiplied by 4 or shifted to the left by two bits. This sum is concatenated with the upper 23 bits of the appropriate root pointer (URP or SRP) to yield the physical address of a root-level table descriptor. Each of the 128 root-level table descriptors corresponds to a 32-Mbyte block of memory and points to the base of a pointer-level table.



* Refers to either instruction or data transparent translation register.

Figure 3-22. Address Translation Flowchart

entire cache, and can select one or both caches for the operation. For line and page operations, a physical address in an address register specifies the memory address.

4.3 CACHING MODES

Every IU access to the cache has an associated caching mode that determines how the cache handles the access. An access can be cachable in either the write-through or copyback modes, or it can be cache inhibited in nonserialized or serialized modes. The CM field corresponding to the logical address of the access normally specifies, on a page-by-page basis, one of these caching modes. The default memory access caching mode is nonserialized. When the cache is enabled and memory management is disabled, the default caching mode is write-through. The transparent translation registers and MMUs allow the defaults to be overridden. In addition, some instructions and IU operations perform data accesses that have an implicit caching mode associated with them. The following paragraphs discuss the different caching accesses and their related cache modes.

4.3.1 Cachable Accesses

If a page descriptor's CM field indicates write-through or copyback, then the access is cachable. A read access to a write-through or copyback page is read from the cache if matching data is found. Otherwise, the data is read from memory and used to update the cache. Since instruction cache accesses are always reads, the selection of write-through or copyback modes do not affected them. The following paragraphs describe the write-through and copyback modes in detail.

4.3.1.1 WRITE-THROUGH MODE. Accesses to pages specified as write-through are always written to the external address, although the cycle can be buffered, keeping memory and cache data consistent. Writes in write-through mode are handled with a no-write-allocate policy—i.e., writes that miss in a data cache are written to memory but do not cause the corresponding line in memory to be loaded into the cache. Write accesses always write through to memory and update matching cache lines. Specifying write-through mode for the shared pages maintains cache coherency for shared memory areas in a multiprocessing environment. The cache supplies data to instruction or data read accesses that hit in the appropriate cache; misses cause a new cache line to be loaded into the cache, replacing a valid cache line if there are no invalid lines.

4.3.1.2 COPYBACK MODE. Copyback pages are typically used for local data structures or stacks to minimize external bus usage and reduce write access latency. Write accesses to pages specified as copyback that hit in the data cache update the cache line and set the corresponding D-bits without an external bus access. The dirty cached data is only written to memory if 1) the line is replaced due to a miss, 2) a cache inhibited access matches the line, or 3) the CPUSH instruction explicitly pushes the line. If a write access misses in the cache, the memory unit reads the needed cache line from memory and updates the cache. When a miss causes a dirty cache line to be selected for replacement, the memory unit places the line in an internal copyback buffer. The replacement line is read into the cache, and writing the dirty cache line back to memory updates memory.

4.3.2 Cache-Inhibited Accesses

Address space regions containing targets such as I/O devices and shared data structures in multiprocessing systems can be designated cache inhibited. If a page descriptor's CM field indicates nonserialized or serialized, then the access is cache inhibited. The caching operation is identical for both cache-inhibited modes. If the CM field of a matching address indicates either nonserialized or serialized modes, the cache controller bypasses the cache and performs an external bus transfer. The data associated with the access is not cached internally, and the cache inhibited out (CIOUT) signal is asserted during the bus transfer to indicate to external memory that the access should not be cached. If the data cache line is already resident in an internal cache, then the data cache line is pushed from the cache if it is dirty or the data cache line is invalidated if it is valid.

If the CM field indicates serialized, then the sequence of read and write accesses to the page is guaranteed to match the sequence of the instruction order. Without serialization, the IU pipeline allows read accesses to occur before completion of a write-back for a previous instruction. Serialization forces operand read accesses for an instruction to occur only once by preventing the instruction from being interrupted after the operand fetch stage. Otherwise, the instruction is aborted, and the operand is accessed when the instruction is restarted. These guarantees apply only when the CM field indicates the serialized mode and the accesses are aligned. Regardless of the selected cache mode, locked accesses are implicitly serialized. The TAS, CAS, and CAS2 instructions use locked accesses for operands in memory and for updating translation table entries during table search operations.

4.3.3 Special Accesses

Several other processor operations result in accesses that have special caching characteristics besides those with an implied cache-inhibited access in the serialized mode. Exception stack accesses, exception vector fetches, and table searches that miss in the cache do not allocate cache lines in the data cache, preventing replacement of a cache line. Cache hits by these accesses are handled in the normal manner according to the caching mode specified for the accessed address.

Accesses by the MOVE16 instruction also do not allocate cache lines in the data cache for either read or write misses. Read hits on either valid or dirty cache lines are read from the cache. Write hits invalidate a matching line and perform an external access. Interacting with the cache in this manner prevents a large block move or block initialization implemented with a MOVE16 from being cached, since the data may not be needed immediately.

If the data cache is re-enabled after a locked access has hit and the data cache was disabled, the next non-locked access that results in a data cache miss will not be cached.

4.4 CACHE PROTOCOL

The cache protocol for processor and snooped accesses is described in the following paragraphs. In all cases, an external bus transfer will cause a cache line state to change

Table 5-1. Signal Index

Signal Name	Mnemonic	Function
Address Bus	A31–A0	32-bit address bus used to address any of 4-Gbytes.
Data Bus	D31–D0	32-bit data bus used to transfer up to 32 bits of data per bus transfer.
Transfer Type	TT1,TT0	Indicates the general transfer type: normal, MOVE16, alternate logical function code, and acknowledge.
Transfer Modifier	TM2–TM0	Indicates supplemental information about the access.
Transfer Line Number	TLN1,TLN0	Indicates which cache line in a set is being pushed or loaded by the current line transfer.
User-Programmable Attributes	UPA1,UPA0	User-defined signals, controlled by the corresponding user attribute bits from the address translation entry.
Read/Write	R/W	Identifies the transfer as a read or write.
Transfer Size	SIZ1,SIZ0	Indicates the data transfer size. These signals, together with A0 and A1, define the active sections of the data bus.
Bus Lock	LOCK	Indicates a bus transfer is part of a read-modify-write operation, and the sequence of transfers should not be interrupted.
Bus Lock End	LOCKE	Indicates the current transfer is the last in a locked sequence of transfers.
Cache Inhibit Out	CIOUT	Indicates the processor will not cache the current bus transfer.
Transfer Start	TS	Indicates the beginning of a bus transfer.
Transfer in Progress	TIP	Asserted for the duration of a bus transfer.
Transfer Acknowledge	TA	Asserted to acknowledge a bus transfer.
Transfer Error Acknowledge	TEA	Indicates an error condition exists for a bus transfer.
Transfer Cache Inhibit	TCI	Indicates the current bus transfer should not be cached.
Transfer Burst Inhibit	TBI	Indicates the slave cannot handle a line burst access.
Data Latch Enable[1]	DLE	Alternate clock input used to latch input data when the processor is operating in DLE mode.
Snoop Control	SC1,SC0	Indicates the snooping operation required during an alternate master access.
Memory Inhibit	MI	Inhibits memory devices from responding to an alternate master access during snooping operations.
Bus Request	BR	Asserted by the processor to request bus mastership.
Bus Grant	BG	Asserted by an arbiter to grant bus mastership to the processor.
Bus Busy	BB	Asserted by the current bus master to indicate it has assumed ownership of the bus.
Cache Disable	CDIS	Dynamically disables the internal caches to assist emulator support.
MMU Disable[2]	MDIS	Disables the translation mechanism of the MMUs.
Reset In	RSTI	Processor reset.
Reset Out	RSTO	Asserted during execution of a RESET instruction to reset external devices.
Interrupt Priority Level[3]	IPL2–IPL0	Provides an encoded interrupt level to the processor.
Interrupt Pending	IPEND	Indicates an interrupt is pending.
Autovector	AVEC	Used during an interrupt acknowledge transfer to request internal generation of the vector number.
Processor Status	PST3–PST0	Indicates internal processor status.
Bus Clock	BCLK	Clock input used to derive all bus signal timing.

The TLNx signals can be used in high-performance systems to build an external snoop filter with a duplicate set of cache tags. The TLNx signals and address bus provide a direct indication of the state of the data caches and can be used to help maintain the duplicate tag store. The TLNx pins do not indicate the correct TLN number when an instruction cache burst fill occurs.

5.3.4 User-Programmable Attributes (UPA1, UPA0)

The UPAx signals are three-state outputs. If they match the logical address, the user-programmable attribute bits in the address translation entry or the transparent translation register determine the UPAx signal level. These signals are only for normal code, data, and MOVE16 accesses. For all other accesses, including table search and cache line push accesses, which may result from a normal access, the UPAx signals are zero. If the transparent translation register and the memory management unit are disabled, the UPAx signals are also zero. When the M68040 is not the bus master, these signals are set to a high-impedance state.

5.3.5 Read/Write (R/W)

This bidirectional three-state signal defines the data transfer direction for the current bus cycle. A high level indicates a read cycle, and a low level indicates a write cycle. The bus snoop controller examines this signal when the processor is not the bus master.

5.3.6 Transfer Size (SIZ1, SIZ0)

These bidirectional three-state signals indicate the data size for the bus transfer. The bus snoop controller examines this signal when the processor is not the bus master. Refer to **Section 7 Bus Operation** for more information on the encoding of these signals.

5.3.7 Lock (LOCK)

This three-state output indicates that the current transfer is part of a sequence of locked transfers for a read-modify-write operation. The external arbiter can use LOCK to prevent an alternate bus master from gaining control of the bus and accessing the same operand between processor accesses for the locked sequence of transfers. Although LOCK indicates that the processor requests the bus be locked, the processor will give up the bus if the external arbiter negates the BG signal. When the M68040 is not the bus master, the LOCK signal is set to a high-impedance state. LOCK drives high before three-stating. Refer to **Section 7 Bus Operation** for information on locked transfers.

5.3.8 Lock End (LOCKE)

This three-state output indicates that the current transfer is the last in a sequence of locked transfers for a read-modify-write operation. The external arbiter can use LOCKE to support arbitration between unrelated locked transfer sequences while still maintaining the indivisible nature of each read-modify-write operation. When the M68040 is not the bus master, the LOCKE signal is set to a high-impedance state. LOCKE drives high before

num	cell	port	function	safe	ccell	dsval	rslt			
"0	(BC_2,	RSTO,	output2,	X),				"	&	
"1	(BC_2,	IPEND,	output2,	X),				"	&	
"2	(BC_2,	CIOUT,	output3,	X,	156,	0,	Z),	"	&	—156 = io.0
"3	(BC_2,	UPA(0),	output3,	X,	156,	0,	Z),	"	&	
"4	(BC_2,	UPA(1),	output3,	X,	156,	0,	Z),	"	&	
"5	(BC_2,	TT(0),	output3,	X,	156,	0,	Z),	"	&	
"6	(BC_4,	TT(0),	input,	X),				"	&	
"7	(BC_2,	TT(1),	output3,	X,	156,	0,	Z),	"	&	
"8	(BC_4,	TT(1),	input,	X),				"	&	
"9	(BC_2,	A(10),	output3,	X,	150,	0,	Z),	"	&	—150 = io.ab
"10	(BC_4,	A(10),	input,	X),				"	&	
"11	(BC_2,	A(11),	output3,	X,	150,	0,	Z),	"	&	
"12	(BC_4,	A(11),	input,	X),				"	&	
"13	(BC_2,	A(12),	output3,	X,	150,	0,	Z),	"	&	
"14	(BC_4,	A(12),	input,	X),				"	&	
"15	(BC_2,	A(13),	output3,	X,	150,	0,	Z),	"	&	
"16	(BC_4,	A(13),	input,	X),				"	&	
"17	(BC_2,	A(14),	output3,	X,	150,	0,	Z),	"	&	
"18	(BC_4,	A(14),	input,	X),				"	&	
"19	(BC_2,	A(15),	output3,	X,	150,	0,	Z),	"	&	
"20	(BC_4,	A(15),	input,	X),				"	&	
"21	(BC_2,	A(16),	output3,	X,	150,	0,	Z),	"	&	
"22	(BC_4,	A(16),	input,	X),				"	&	
"23	(BC_2,	A(17),	output3,	X,	150,	0,	Z),	"	&	
"24	(BC_4,	A(17),	input,	X),				"	&	
"25	(BC_2,	A(18),	output3,	X,	150,	0,	Z),	"	&	
"26	(BC_4,	A(18),	input,	X),				"	&	
"27	(BC_2,	A(19),	output3,	X,	150,	0,	Z),	"	&	
"28	(BC_4,	A(19),	input,	X),				"	&	
"29	(BC_2,	A(20),	output3,	X,	150,	0,	Z),	"	&	
"30	(BC_4,	A(20),	input,	X),				"	&	
"31	(BC_2,	A(21),	output3,	X,	150,	0,	Z),	"	&	
"32	(BC_4,	A(21),	input,	X),				"	&	
"33	(BC_2,	A(22),	output3,	X,	150,	0,	Z),	"	&	
"34	(BC_4,	A(22),	input,	X),				"	&	
"35	(BC_2,	A(23),	output3,	X,	150,	0,	Z),	"	&	
"36	(BC_4,	A(23),	input,	X),				"	&	
"37	(BC_2,	A(24),	output3,	X,	150,	0,	Z),	"	&	
"38	(BC_4,	A(24),	input,	X),				"	&	
"39	(BC_2,	A(25),	output3,	X,	150,	0,	Z),	"	&	
"40	(BC_4,	A(25),	input,	X),				"	&	
"41	(BC_2,	A(26),	output3,	X,	150,	0,	Z),	"	&	
"42	(BC_4,	A(26),	input,	X),				"	&	
"43	(BC_2,	A(27),	output3,	X,	150,	0,	Z),	"	&	
"44	(BC_4,	A(27),	input,	X),				"	&	
"45	(BC_2,	A(28),	output3,	X,	150,	0,	Z),	"	&	
"46	(BC_4,	A(28),	input,	X),				"	&	
"47	(BC_2,	A(29),	output3,	X,	150,	0,	Z),	"	&	
"48	(BC_4,	A(29),	input,	X),				"	&	
"49	(BC_2,	A(30),	output3,	X,	150,	0,	Z),	"	&	
"50	(BC_4,	A(30),	input,	X),				"	&	
"51	(BC_2,	A(31),	output3,	X,	150,	0,	Z),	"	&	
"52	(BC_4,	A(31),	input,	X),				"	&	
"53	(BC_2,	D(0),	output3,	X,	151,	0,	Z),	"	&	— 151 = io.db
"54	(BC_2,	D(1),	output3,	X,	151,	0,	Z),	"	&	
"55	(BC_2,	D(2),	output3,	X,	151,	0,	Z),	"	&	
"56	(BC_2,	D(3),	output3,	X,	151,	0,	Z),	"	&	

```
num     cell      port      function     safe     ccell    dsval    rslt

"171    (BC_4,    SC(0),    input,       X),                                  "  &
"172    (BC_4,    TBI,      input,       X),                                  "  &
"173    (BC_4,    AVEC,     input,       X),                                  "  &
"174    (BC_4,    TCI,      input,       X),                                  "  &
"175    (BC_4,    DLE,      input,       X),                                  "  &
"176    (BC_4,    PCLK,     input,       X),                                  "  &
"177    (BC_4,    BCLK,     input,       X),                                  "  &
"178    (BC_4,    IPL(0),   input,       X),                                  "  &
"179    (BC_4,    IPL(1),   input,       X),                                  "  &
"180    (BC_4,    IPL(2),   input,       X),                                  "  &
"181    (BC_4,    RSTI,     input,       X),                                  "  &
"182    (BC_4,    CDIS,     input,       X),                                  "  &
"183    (BC_4,    MDIS,     input,       X)                                   "  ;

attribute DESIGN_WARNING of MC68040: entity is
        "A non-standard clocking protocol on BCLK and PCLK must be          "  &
        "observed when entering Boundary Scan Test Mode.                    "  ;

end MC68040                                                                    ;
```

6.7 MC68040, MC68LC040, MC68EC040 JTAG ELECTRICAL CHARACTERISTICS

The following paragraphs provide information on JTAG electrical and timing specifications. This section is subject to change. For the most recent specifications, contact a Motorola sales office or complete the registration card at the beginning of this manual.

JTAG DC Electrical Specifications

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2	V_{CC}	V
Input Low Voltage	V_{IL}	GND	0.8	V
Undershoot	—	—	0.8	V
TCK Input Leakage Current @ 0.5–2.4 V	I_{in}	20	20	µA
TDO Hi-Z (Off-State) Leakage Current @ 0.5–2.4 V	I_{TST}	20	20	µA
Signal Low Input Current, V_{IL} = 0.8 V TMS, TDI, TRST	I_L	–1.1	–0.18	mA
Signal High Input Current, V_{IH} = 2.0 V TMS, TDI, TRST	I_H	–0.94	–0.16	mA
TDO Output High Voltage	V_{OH}	2.4	—	V
TDO Output Low Voltage	V_{OL}	—	0.5	V
Capacitance*, V_{in} = 0 V, f = 1 MHz	C_{in}	—	25	pF

*Capacitance is periodically sampled rather than 100% tested.

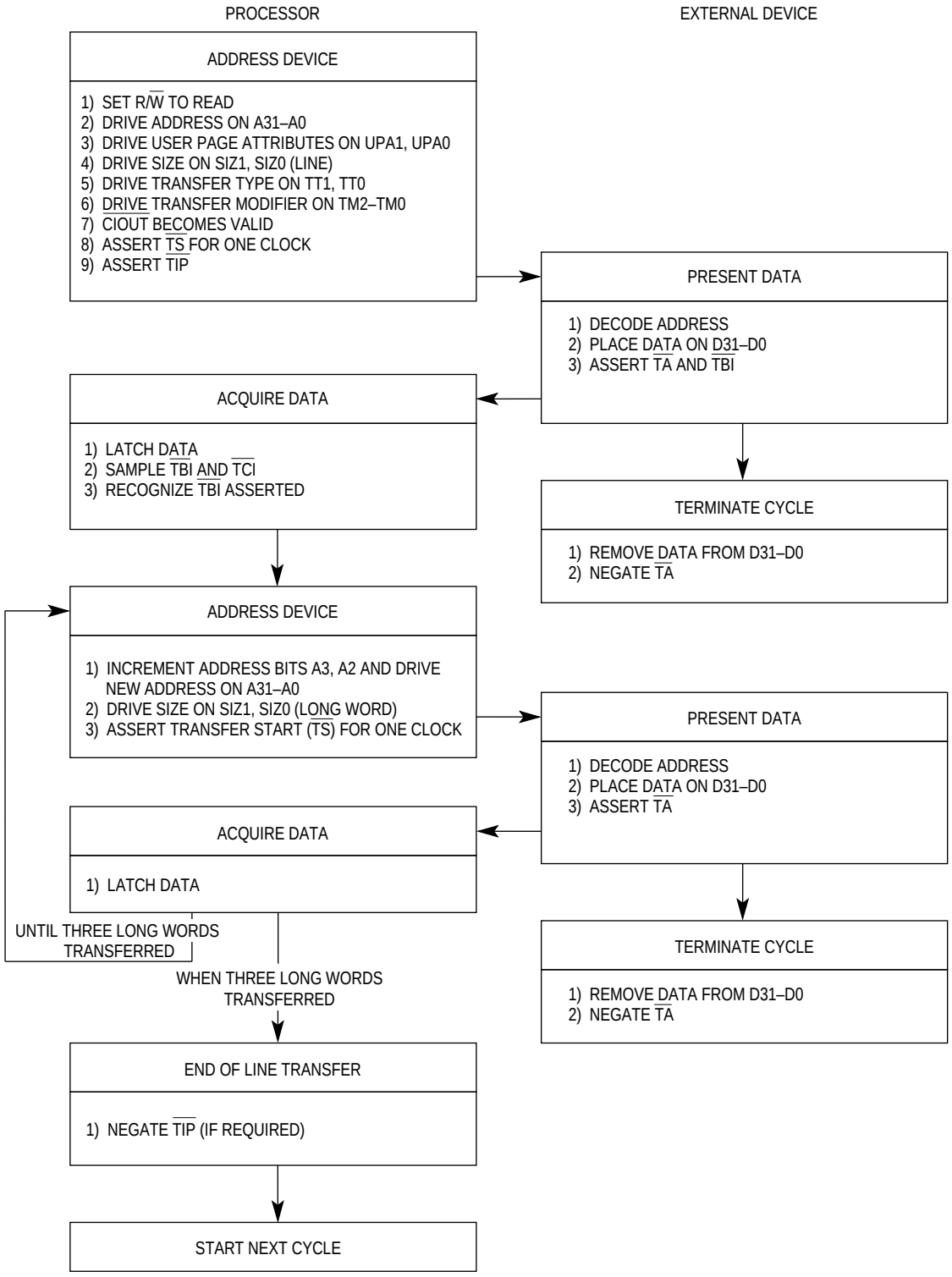


Figure 7-12. Burst-Inhibited Line Read Transfer Flowchart

generates the vector number, which is the sum of the interrupt priority level plus 24 ($18). There are seven distinct autovectors that can be used, corresponding to the seven levels of interrupts available with $\overline{IPL2}$–$\overline{IPL0}$ signals. Figure 7-23 illustrates a functional timing diagram for an autovector operation.

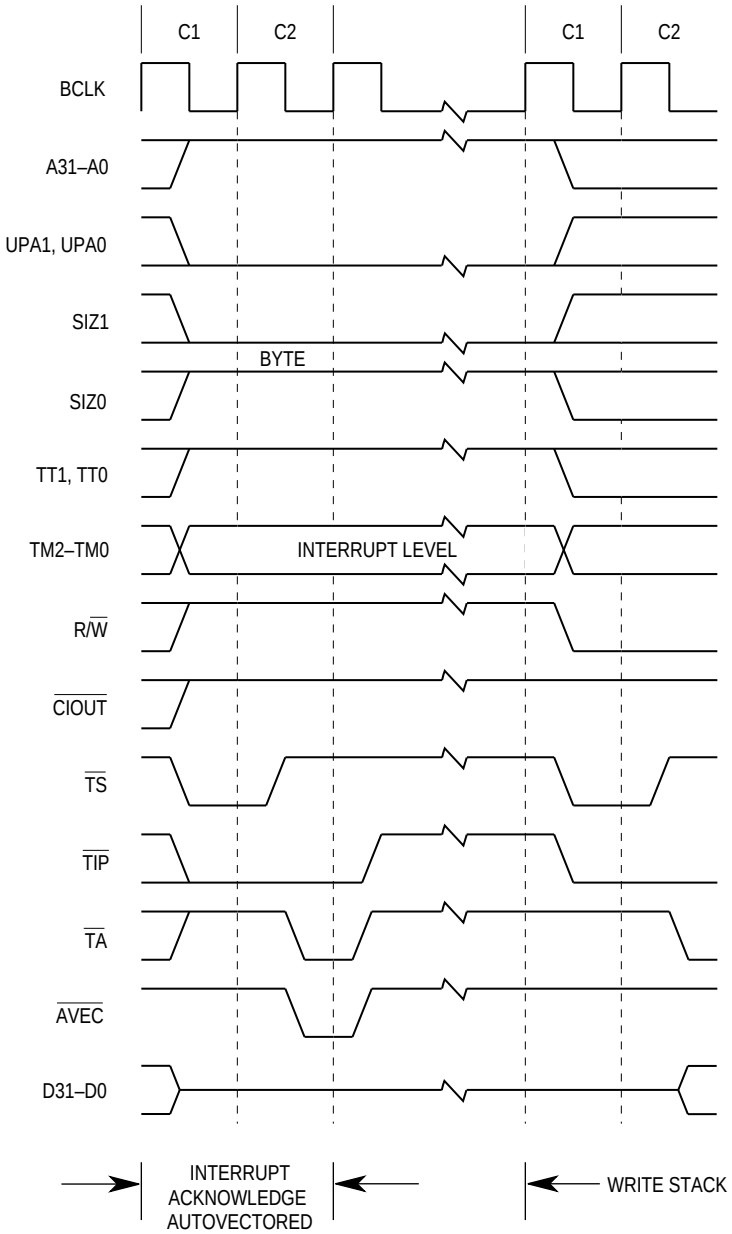


Figure 7-23. Autovector Interrupt Acknowledge Bus Cycle Timing

7.5.1.3 SPURIOUS INTERRUPT ACKNOWLEDGE BUS CYCLE. When a device does not respond to an interrupt acknowledge bus cycle with $\overline{TA}$, or $\overline{AVEC}$ and $\overline{TA}$, the external logic typically returns the transfer error acknowledge signal ($\overline{TEA}$). In this case, the M68040 automatically generates the spurious interrupt vector number 24 ($18) instead of the interrupt vector number. If $\overline{TA}$ and $\overline{TEA}$ are both asserted, the processor retries the cycle.

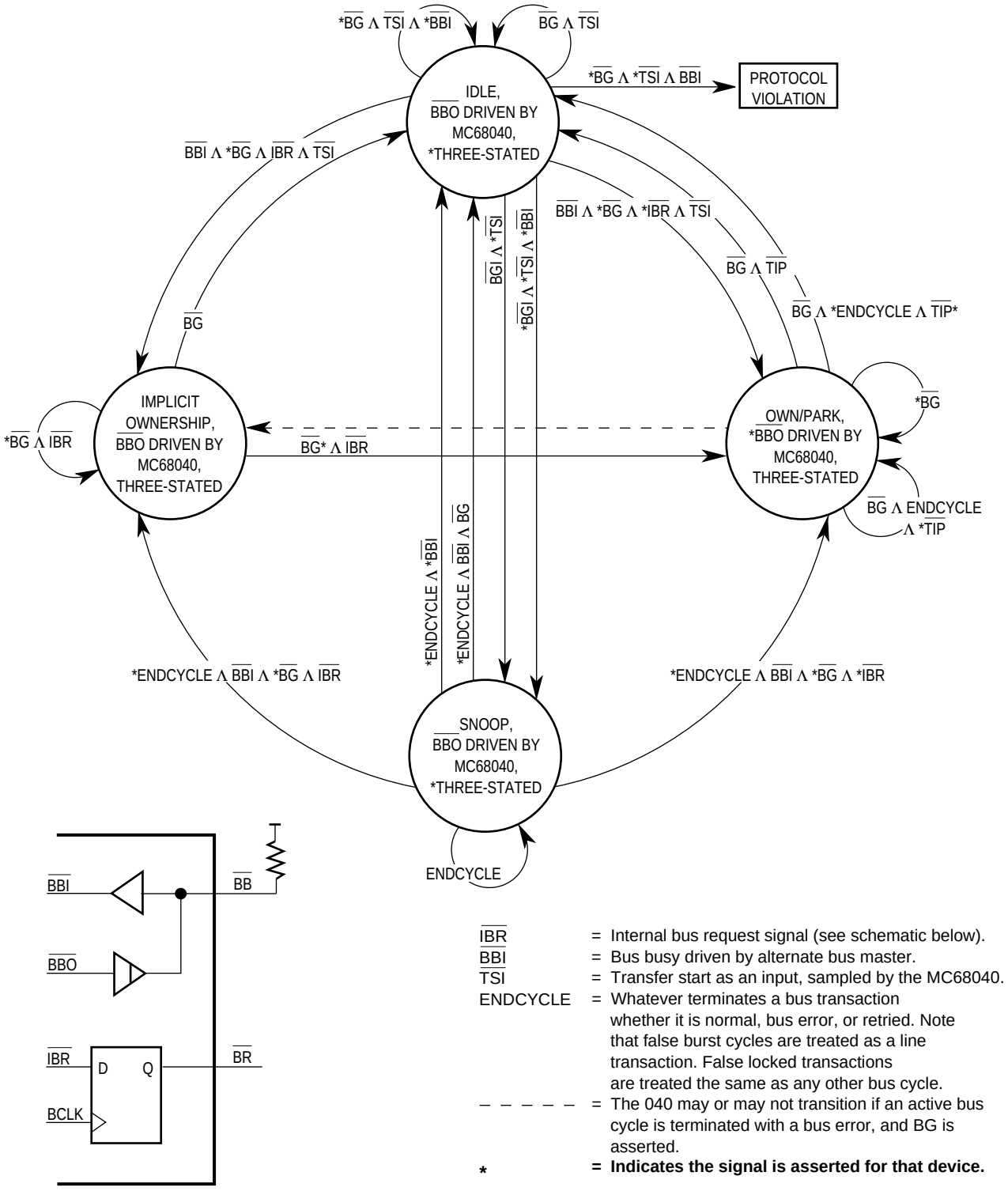


Figure 7-30. M68040 Internal Interpretation State Diagram and External Bus Arbiter Circuit

The third step is to save the current processor contents for all exceptions other than reset. The processor creates one of five exception stack frame formats on the active supervisor stack and fills it with information appropriate for the type of exception. Other information can also be stacked, depending on which exception is being processed and the state of the processor prior to the exception. If the exception is an interrupt and the M-bit of the SR is set, the processor clears the M-bit and builds a second stack frame on the interrupt stack. Figure 8-2 illustrates the general form of the exception stack frame.

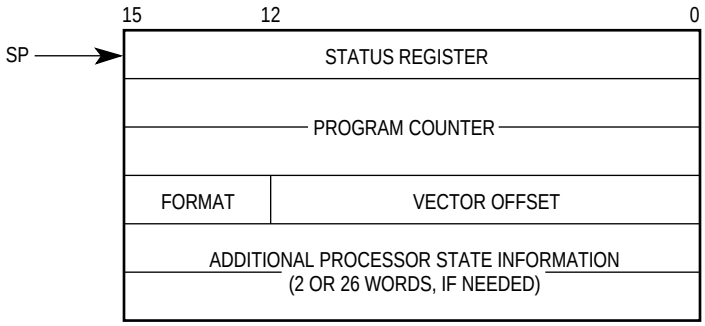


Figure 8-2. General Form of Exception Stack Frame

The last step initiates execution of the exception handler. The processor multiplies the vector number by four to determine the exception vector offset. It adds the offset to the value stored in the vector base register (VBR) to obtain the memory address of the exception vector. Next, the processor loads the program counter (PC) (and the interrupt stack pointer (ISP) for the reset exception) from the exception vector table entry. After prefetching the first four long words to fill the instruction pipe, the processor resumes normal processing at the address in the PC. When the processor executes an RTE instruction, it examines the stack frame on top of the active supervisor stack to determine if it is a valid frame and what type of context restoration it requires.

All exception vectors are located in the supervisor address space and are accessed using data references. Only the initial reset vector is fixed in the processor's memory map; once initialization is complete, there are no fixed assignments. Since the VBR provides the base address of the exception vector table, the exception vector table can be located anywhere in memory; it can even be dynamically relocated for each task that an operating system executes.

The M68040 supports a 1024-byte vector table containing 256 exception vectors (see Table 8-1). Motorola defines the first 64 vectors and reserves the other 192 vectors for user-defined interrupt vectors. External devices can use vectors reserved for internal purposes at the discretion of the system designer. External devices can also supply vector numbers for some exceptions. External devices that cannot supply vector numbers use the autovector capability, which allows the M68040 to automatically generate a vector number.

level 6 interrupt, the SR mask is automatically updated with a value of 6 before entering the handler routine so that subsequent level 6 interrupts and lower level interrupts are masked. Provided no instruction that lowers the mask value is executed, the external request can be lowered to level 3 and then raised back to level 6 and a second level 6 interrupt is not processed. However, if the M68040 is handling a level 7 interrupt (SR mask set to level 7) and the external request is lowered to level 3 and than raised back to level 7, a second level 7 interrupt is processed. The second level 7 interrupt is processed because the level 7 interrupt is transition sensitive. A level comparison also generates a level 7 interrupt if the request level and mask level are at 7 and the priority mask is then set to a lower level (with the MOVE to SR or RTE instruction, for example). The level 6 interrupt request and mask level example in Figure 8-3 is the same as for all interrupt levels except 7.

9.6 FLOATING-POINT EXCEPTIONS

There are two classes of floating-point-related exceptions: nonarithmetic floating-point exceptions and arithmetic floating-point exceptions. The latter relates to the handling of arithmetic exceptions caused by floating-point activity, and the former includes unimplemented floating-point instructions and unsupported data types not related to the handling of arithmetic exceptions. Format error and FTRAPcc exceptions may seem to be floating-point related, but are considered IU exceptions (see **Section 8 Exception Processing**). The following sections detail floating-point exceptions and how the MC68040 and M68040FPSP handle them. Table 9-9 lists the vector numbers related to floating-point exceptions.

Table 9-9. Floating-Point Exception Vectors

Vector Number	Vector Offset (Hex)	Assignment
11	02C	Floating-Point Unimplemented Instruction (also used for F-line instruction)
48	0C0	Floating-Point Branch or Set on Unordered Condition
49	0C4	Floating-Point Inexact Result
50	0C8	Floating-Point Divide by Zero
51	0CC	Floating-Point Underflow
52	0D0	Floating-Point Operand Error
53	0D4	Floating-Point Overflow
54	0D8	Floating-Point SNAN
55	0DC	Floating-Point Unimplemented Data Type

The following paragraphs detail nonarithmetic floating-point exceptions.

9.6.1 Unimplemented Floating-Point Instructions

F-line instructions are instruction word patterns with bits 15–12 that have an $F encoding, causing F-line exceptions. These instructions are termed unimplemented floating-point instructions and cause an unimplemented floating-point exception. The MC68040 recognizes some F-line instructions, such as the FMUL and CPUSH, which do not cause F-line exceptions. There are some F-line instructions that the MC68040 recognizes as valid MC68881/MC68882 floating-point instruction patterns, but as floating-point instructions that the processor cannot complete in hardware. Table 9-10 lists the floating-point instructions that are unimplemented and therefore cause an unimplemented instruction exception.

If the processor encounters an F-line instruction and the instruction patterns do not match either of the above two cases, the processor takes an F-line illegal exception. F-line illegal exceptions are discussed further in **Section 8 Exception Processing**. The processor generates an exception with vector number 11 and pushes a four-word stack frame format $0 on the system stack. An illegal instruction exception is also reported when a breakpoint acknowledge bus cycle is run and terminated with either a transfer acknowledge ($\overline{TA}$) or transfer error acknowledge ($\overline{TEA}$) signal. Since the unimplemented floating-point

The separation of calculation and execution in the <ea> calculate and execute stages allows instruction reordering during compile time to take advantage of potential instruction overlap. Figure 10-2 illustrates this overlap for an instruction requiring multiple clocks in the execute stage and with an instruction with a long lead time. The execution time for LEA (3_L + 1) indicates that the instruction can be stalled three clocks without affecting execution.

When the LEA (A) instruction precedes the ABCD (B) instruction, the execution stalls during C4–C6 (equivalent to the LEA lead time) while the instruction completes in the <ea> calculate and <ea> fetch stages. The resulting execution time for the LEA (A) and ABCD (B) sequence is eight clocks.

However, if the LEA (C) instruction follows the ABCD (B) instruction, the LEA stalls in the <ea> fetch instead, during C9–C11. The LEA then executes in a single clock in the execution stage. The resulting execution time for the LEA (C) and ABCD (B) sequence is five clocks.

LABEL	INSTRUCTION		<ea> CALCULATE	EXECUTE
P1	TRAPF		1	1
A	LEA	$24(PC),A1	4	3_L + 1
B	ABCD	D0,D1	1	3
C	LEA	$24(PC),A1	4	3_L + 1
N1	TRAPF		1	1
N2	TRAPF		1	1

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
<ea> CALCULATE	P1	A	A	A	A	B	C	C	C	C	N1	N2	
<ea> FETCH		P1	A	A	A	A	B	C	C_{L1}	C_{L2}	C_{L3}	N1	N2
EXECUTE			P1	A_L	A_L	A_L	A	B	B	B	C*	C	N1
WRITE-BACK													

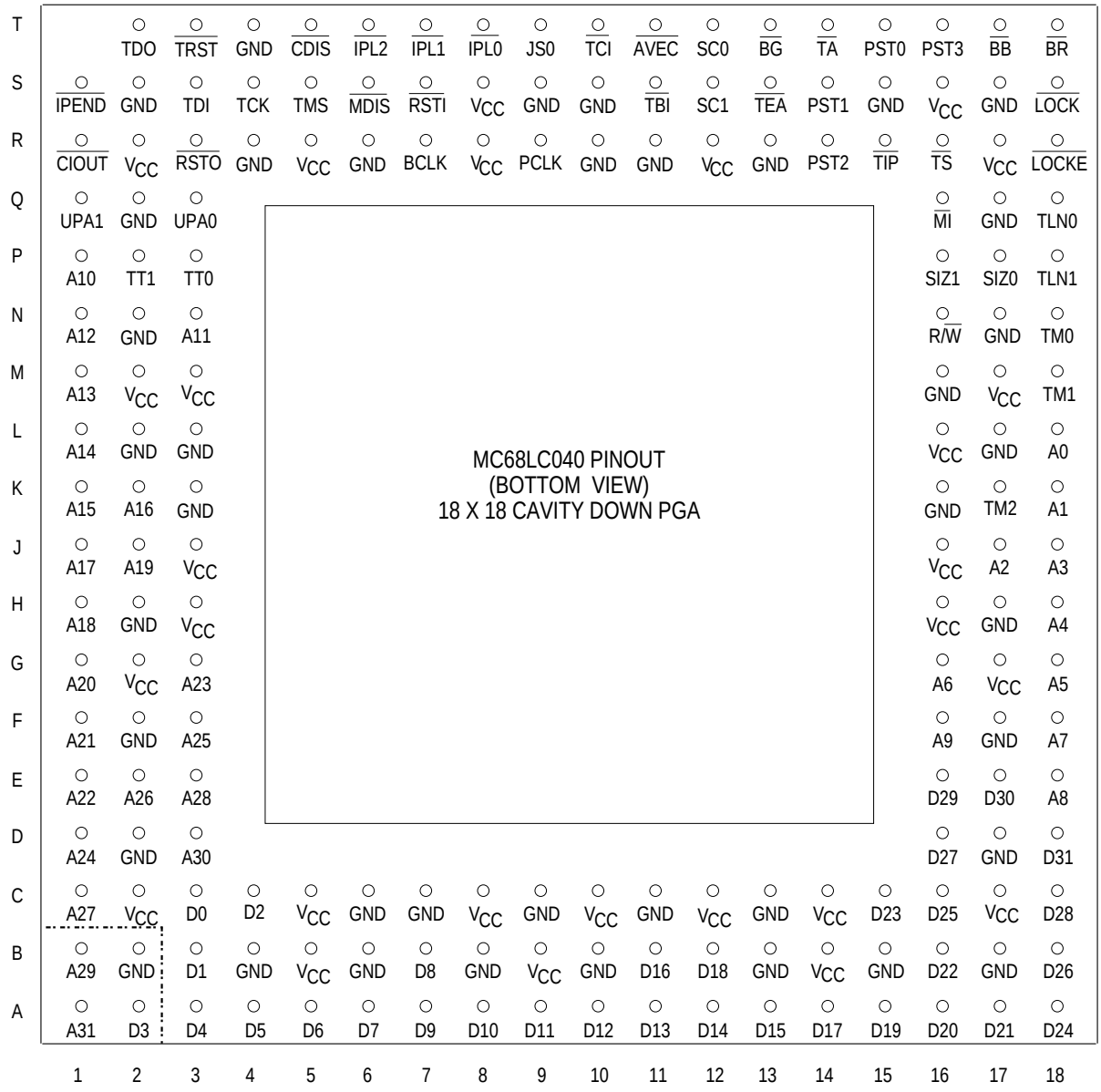
NOTE: *Possible stalls in this stage.

Figure 10-2. Instruction Overlap with Multiple Clocks

10.7.3 Timings in the Floating-Point Unit (Continued)

Instruction	Opclass	Size	Precision	Operands	Conversion	Execution	Normalization
FDIV	2	S,D	Any	— ,NAN	4	0	0
	2	X	Any	Norm,Norm	3(4)	37.5	2(3)
	2	X	Any	— ,Zero	5	0	0
	2	—	Any	— ,Inf	5	0	0
	2	X	Any	— ,NAN	5	0	0
FSQRT	0	—	Any	Norm	2(3)	103	2(3)
	0	—	Any	(Zero|Inf|NAN)	4	0	0
	2	S,D	Any	Norm	2(3)	103	2(3)
	2	S,D	Any	(Zero|Inf|NAN)	4	0	0
	2	X	Any	Norm	3(4)	103	2(3)
	2	X	Any	(Zero|Inf|NAN)	5	0	0
FMOVE,	0	—	X	(Norm|Zero|Inf)	2	0	0
FABS,	0	—	X	NAN	3	0	0
FNEG	0	—	S,D	Norm	5	0	0
	0	—	S,D	(Zero|Inf)	3	0	0
	0	—	S,D	NAN	4	0	0
	2	S	Any	(Norm|Zero|Inf)	3	0	0
	2	S	Any	NAN	4	0	0
	2	D	D,X	(Norm|Zero|Inf)	3	0	0
	2	D	D,X	NAN	4	0	0
	2	D	S	Norm	5	0	0
	2	D	S	(Zero|Inf)	4	0	0
	2	D	S	NAN	5	0	0
	2	X	X	(Norm|Zero|Inf)	4	0	0
	2	X	X	NAN	5	0	0
	2	X	S,D	Norm	6	0	0
	2	X	S,D	(Zero|Inf)	5	0	0
	2	X	S,D	NAN	6	0	0
	2	B,W	Any	(+Norm|Zero)	1.5(11)	4.5	2
	2	L	D,X	(+Norm|Zero)	1.5(11)	4.5	2
	2	L	S	(+Norm|Zero)	1.5(12.5)	4.5	2
	2	B,W	Any	—Norm	1.5(11.5)	5	2
	2	L	D,X	—Norm	1.5(11.5)	5	2
	2	L	S	—Norm	1.5(13)	5	2
FMOVE	3	S,D	Any	Any	3	0	0
	3	X	Any	Any	4	0	0
	3	B,W,L	Any	+(Norm|Zero)	3(9)	1.5	3.5
	3	B,W,L	Any	–(Norm|Zero)	3(10)	1.5	4.5

12.2.2 MC68LC040 Pin Grid Array



Pin Group	GND	V_{CC}
PLL	S9, R6, R10	R8, S8
Internal Logic	C6, C7, C9, C11, C13, K3, K16, L3, M16, R4, R11, R13, S6, S10, T4	C5, C8, C10, C12, C14, H3, H16, J3, J16, L16, M3, R5, R12
Output Drivers	B2, B4, B6, B8, B10, B13, B15, B17, D2, D17, F2, F17, H2, H17, L2, L17, N2, N17, Q2, Q17, S2, S15, S17	B5, B9, B14, C2, C17, G2, G17, M2, M17, R2, R17, S16

A.6.3 DC Electrical Specifications (V_{CC} = 5.0 Vdc ±5 %)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2	V_{CC}	V
Input Low Voltage	V_{IL}	GND	0.8	V
Undershoot	—	—	0.8	V
Input Leakage Current @ 0.5/2.4 V $\overline{AVEC}$, BCLK, $\overline{BG}$, $\overline{CDIS}$, $\overline{MDIS}$, $\overline{IPLA}$, PCLK, $\overline{RSTI}$, SCx, $\overline{TBI}$, TLNx, $\overline{TCI}$, TCK, $\overline{TEA}$	I_{in}	20	20	µA
Hi-Z (Off-State) Leakage Current @ 0.5/2.4 V An, $\overline{BB}$, $\overline{CIOUT}$, Dn, $\overline{LOCK}$, $\overline{LOCKE}$, R/$\overline{W}$, SIZx, $\overline{TA}$, TDO, $\overline{TIP}$, TMx, TLNx, $\overline{TS}$, TTx, UPAx	I_{TSI}	20	20	µA
Signal Low Input Current, V_{IL} = 0.8 V TMS, TDI, $\overline{TRST}$	I_{IL}	–1.1	–0.18	mA
Signal High Input Current, V_{IH} = 2.0 V TMS, TDI, $\overline{TRST}$	I_{IH}	–0.94	–0.16	mA
Output High Voltage, I_{OH} = 5 mA (Small Buffer Mode)	V_{OH}	2.4	—	V
Output Low Voltage, I_{OL} = 5 mA (Small Buffer Mode)	V_{OL}	—	0.5	V
Output High Voltage, I_{OH} = 55 mA (Large Buffer Mode)	V_{OH}	2.4	—	V
Output Low Voltage, I_{OL} = 55 mA (Large Buffer Mode)	V_{OL}	—	0.5	V
Capacitance*, V_{in} = 0 V, f = 1 MHz	C_{in}	—	25	pF

*Capacitance is periodically sampled rather than 100% tested.

A.6.4 Power Dissipation

Frequency	Watts
Maximum Values (V_{CC} = 5.25 V, T_A = 0°C)	
20 MHz	3.2
25 MHz	3.9
33 MHz	4.9
40 MHz	5.5
Typical Values (V_{CC} = 5 V, T_A = 25°C)*	
20 MHz	2.0
25 MHz	2.4
33 MHz	3.0
40 MHz	3.5

*This information is for system reliability purposes.

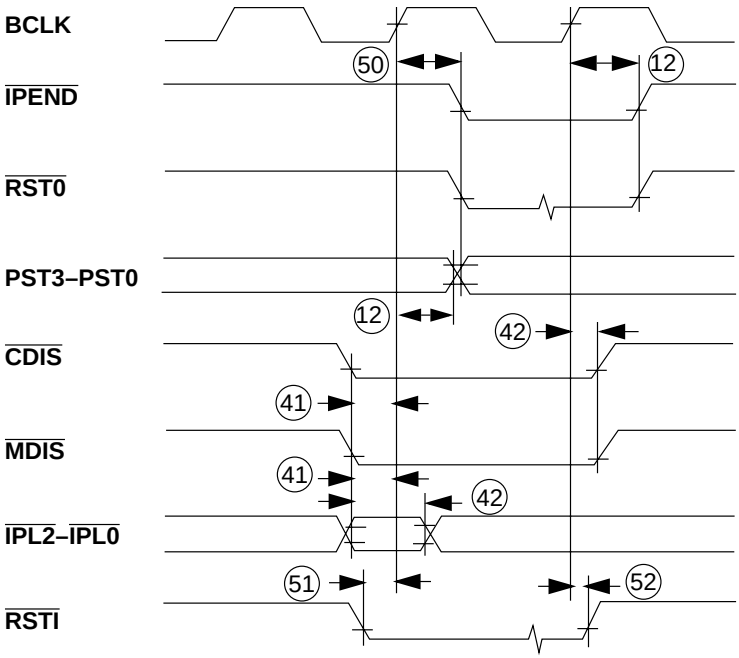


Figure A-9. Other Signal Timing

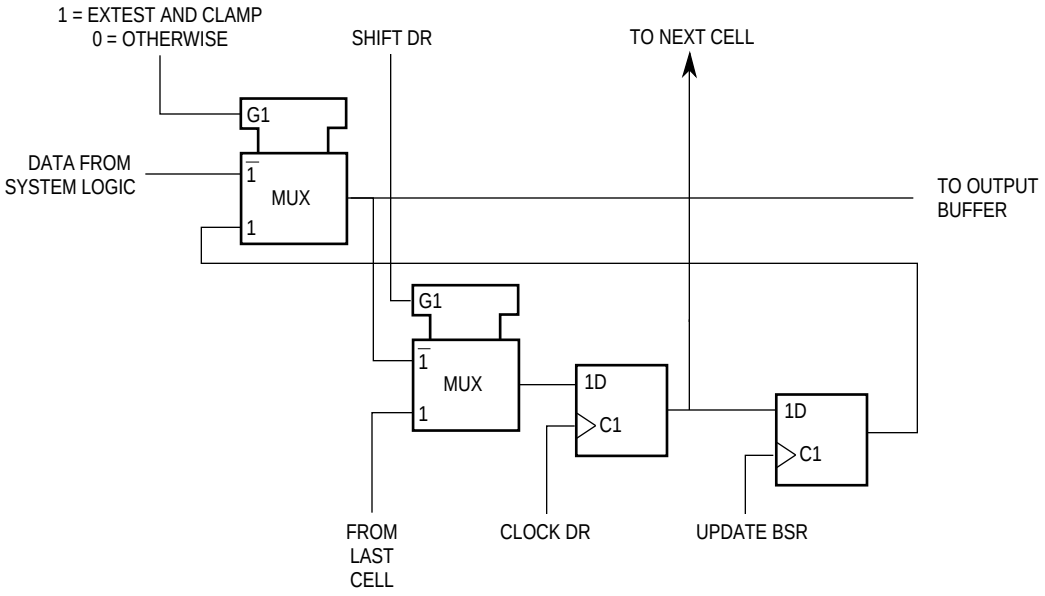


Figure C-6. Output Latch Cell (O.Latch)

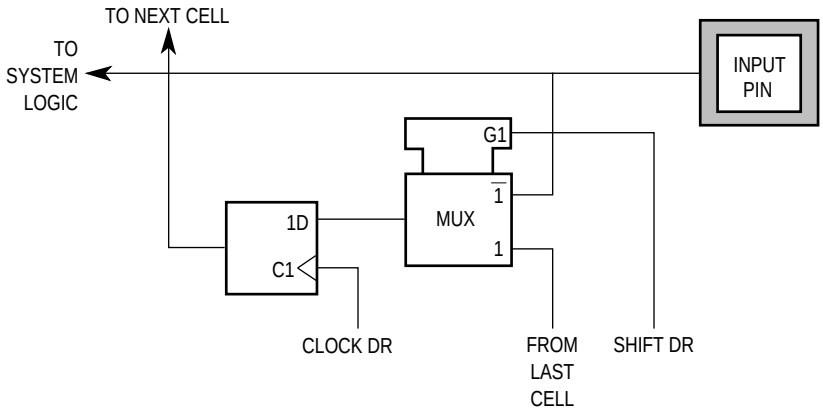


Figure C-7. Input Pin Cell (I.Pin)