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	40MHz
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/mc68040fe40v

TABLE OF CONTENTS (Continued)

Section 8
Exception Processing

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

avoided by pushing cache lines when a page descriptor is changed and ensuring that alternate bus masters indicate the appropriate snoop operation for writes to corresponding pages (i.e., mark invalid for write-through pages and sink data for copyback pages). If the access is copyback, the cache controller updates the cache line and sets the D-bit for of the appropriate long words in the cache line. An external write is not performed, and the cache line state changes to, or remains in, the dirty state.

An alternate bus master can drive the SCx signals for a write access with an encoding that indicates to the M68040 that it should sink the data, inhibit memory, and respond as a slave if the access hits in the cache. The cache operation depends on the access size and current line state. A snooped line write that hits a valid line always causes the corresponding cache line to be invalidated. For snooped writes of byte, word, or long-word size that hit a dirty line, the processor inhibits memory and responds to the alternate bus master as a slave, sinking the data. Data received from the alternate bus master is written to the appropriate long word in the cache line, and the D-bit is set for that entry. The cache controller invalidates a cache line if the snoop control pins have indicated that a matching cache line is marked invalid for a snoop write.

4.5 CACHE COHERENCY

The M68040 provides several different mechanisms to assist in maintaining cache coherency in multimaster systems. Both write-through and copyback memory update techniques are supported to maintain coherency between the data cache and memory.

Alternate bus master accesses can reference data that the M68040 caches, causing coherency problems if the accesses are not handled properly. The M68040 snoops the bus during alternate bus master transfers. If a write access hits in the cache, the M68040 can update its internal caches, or if a read access hits, it can intervene in the access to supply dirty data. Caches can be snooped even if they are disabled. The alternate bus master controls snooping through the snoop control signals, indicating which access can be snooped and the required operation for snoop hits. Table 4-1 lists the requested snoop operation for each encoding of the snoop control signals. Since the processor and the bus snooper must both access the caches, the snoop controller has priority over the processor for snoopable accesses to maintain cache coherency.

Table 4-1. Snoop Control Encoding

		Requested Snoop Operation	
SC1	**SC0**	**Alternate Bus Master Read Access**	**Alternate Bus Master Write Access**
0	0	Inhibit Snooping	Inhibit Snooping
0	1	Supply Dirty Data and Leave Dirty Data	Sink Byte/Word/Long/Long Word
1	0	Supply Dirty Data and Mark Line Invalid	Invalidate Line
1	1	Reserved (Snoop Inhibited)	Reserved (Snoop Inhibited)

The snooping protocol and caching mechanism supported by the M68040 are optimized to support multimaster systems with the M68040 as the single caching master. In systems

Table 4-4. Data-Cache Line State Transitions (Continued)

Cache Operation	Current State					
	Invalid Cases		Valid Cases		Dirty Cases	
Alternate Master Read Hit (Snoop Control = 10 — Invalidate)	I10	Not Possible	V10	No action; go to invalid state.	D10	Inhibit memory and source data; go to invalid state
Alternate Master Write Hit (Snoop Control = 10 —Invalidate)	I11	Not Possible	V11	No action; go to invalid state.	D11	No action; go to invalid state.
Alternate Master Write Hit (Snoop Control = 01 — Sink Data and Size ≠ Line)	I12	Not Possible	V12	No action; go to invalid state.	D12	Inhibit memory and sink data; set Dn bits of modified long words; remain in current state.
Alternate Master Write Hit (Snoop Control = 01 — Sink Data and Size = Line)	I13	Not Possible	V13	No action; go to invalid state.	D13	No action; go to invalid state.

5.3.2 Transfer Modifier (TM2–TM0)

These three-state outputs provide supplemental information for each transfer type. Table 5-3 lists the encoding for normal and MOVE16 transfers, and Table 5-4 lists the encoding for alternate access transfers. For interrupt acknowledge transfers, the TMx signals carry the interrupt level being acknowledged; for breakpoint acknowledge transfers and LPSTOP broadcast cycles on the MC68040V and MC68EC040V, the TMx signals are low. When the M68040 is not the bus master, the TMx signals are set to a high-impedance state.

Table 5-3. Normal and MOVE16 Access Transfer Modifier Encoding

TM2	TM1	TM0	Transfer Modifier
0	0	0	Data Cache Push Access
0	0	1	User Data Access*
0	1	0	User Code Access
0	1	1	MMU Table Search Data Access
1	0	0	MMU Table Search Code Access
1	0	1	Supervisor Data Access*
1	1	0	Supervisor Code Access
1	1	1	Reserved

* MOVE16 accesses use only these encodings.

Table 5-4. Alternate Access Transfer Modifier Encoding

TM2	TM1	TM0	Transfer Modifier
0	0	0	Logical Function Code 0
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Logical Function Code 3
1	0	0	Logical Function Code 4
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Logical Function Code 7

5.3.3 Transfer Line Number (TLN1, TLN0)

These three-state outputs indicate which line in the set of four data cache lines is being accessed for normal push and line data read accesses. TLNx signals are undefined for all other accesses to instruction space and are placed in a high-impedance state when the processor relinquishes the bus.

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),	"	&	

7.4.5 Read-Modify-Write Transfers (Locked Transfers)

The read-modify-write transfer performs a read, conditionally modifies the data in the processor, and writes the data out to memory. In the M68040, this operation can be indivisible, providing semaphore capabilities for multiprocessor systems. During the entire read-modify-write sequence, the M68040 asserts the $\overline{\text{LOCK}}$ signal to indicate that an indivisible operation is occurring and asserts the $\overline{\text{LOCKE}}$ signal for the last transfer to indicate completion of the locked sequence. The external arbiter can use the $\overline{\text{LOCK}}$ and $\overline{\text{LOCKE}}$ signals to prevent arbitration of the bus during locked processor sequences. External bus arbitrations can use $\overline{\text{LOCKE}}$ to support bus arbitration between consecutive read-modify-write cycles. A read-modify-write operation is treated as noncachable. If the access hits in the data cache, it invalidates a matching valid entry and pushes a matching dirty entry. The read-modify-write transfer begins after the line push (if required) is complete; however, $\overline{\text{LOCK}}$ may assert during the line push bus cycle.

The TAS, CAS, and CAS2 instructions are the only M68040 instructions that utilize read-modify-write transfers. Some page descriptor updates during translation table searches also use read-modify-write transfers. Refer to **Section 3 Memory Management Unit (Except MC68EC040 and MC68EC040V)** for information about table searches.

The read-modify-write transfer for the CAS and CAS2 instructions in the M68040 differs from those used by previous members of the M68000 family. If an operand does not match one of these instructions, the M68040 still executes a single write transfer to terminate the locked sequence with $\overline{\text{LOCKE}}$ asserted. For the CAS instruction, the value read from memory is written back; for the CAS2 instruction, the second operand read is written back. Figure 7-18 illustrates a functional timing diagram for a TAS instruction read-modify-write bus transfer.

Clock 1 (C1)

The read cycle starts in C1. During the first half of C1, the processor places valid values on the address bus and transfer attributes. $\overline{\text{LOCK}}$ is asserted to identify a locked read-modify-write bus cycle. For user and supervisor mode accesses, which the corresponding memory unit translates, the UPAx signals are driven with the values from the matching U1 and U0 bits. The TTx and TMx signals identify the specific access type. R/$\overline{\text{W}}$ is driven high for a read cycle. $\overline{\text{CIOUT}}$ is asserted if the access is identified as noncachable. The processor asserts $\overline{\text{TS}}$ during C1 to indicate the beginning of a bus cycle. If not already asserted from a previous bus cycle, the $\overline{\text{TIP}}$ signal is also asserted at this time to indicate that a bus cycle is active. Refer to **Section 3 Memory Management Unit (Except MC68EC040 and MC68EC040V)** for information on the M68040 and MC68LC040 memory units and **Appendix B MC68EC040** for information on the MC68EC040 memory unit.

7.5.2 Breakpoint Interrupt Acknowledge Bus Cycle

The execution of a breakpoint instruction (BKPT) generates the breakpoint interrupt acknowledge bus cycle. An acknowledged access is indicated with TT1 and TT0 = $3, address A31–A0 = $00000000, and TM2–TM0 = $0. When the external device terminates the cycle with either $\overline{TA}$ or $\overline{TEA}$, the processor takes an illegal instruction exception. Figures 7-24 and 7-25 illustrate a flowchart and functional timing diagram for a breakpoint interrupt acknowledge transfer.

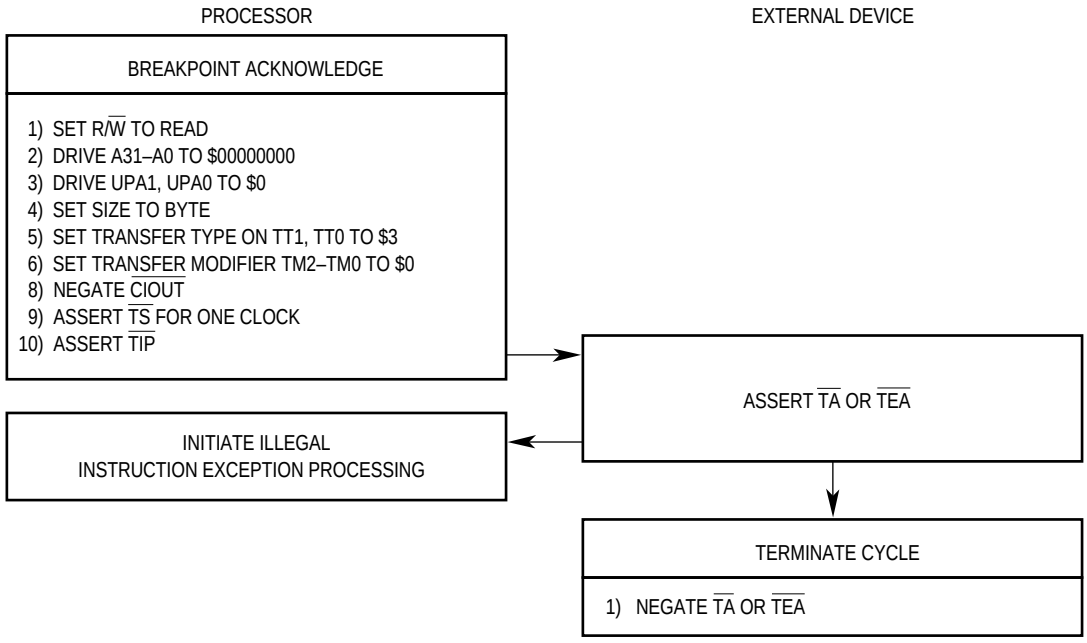


Figure 7-24. Breakpoint Interrupt Acknowledge Bus Cycle Flowchart

that caused the trap is also saved. Instruction execution resumes at the address in the exception vector after the required instruction is prefetched.

8.2.4 Illegal Instruction and Unimplemented Instruction Exceptions

An illegal instruction exception corresponds to vector number 4, and occurs when the processor attempts to execute an illegal instruction. An illegal instruction is an instruction that contains any bit pattern that does not correspond to the bit pattern of a valid M68040 instruction. An illegal instruction exception is also taken after a breakpoint acknowledge bus cycle is terminated, either by the assertion of the transfer acknowledge ($\overline{TA}$) or the transfer error acknowledge ($\overline{TEA}$) signal. An illegal instruction exception can also be a MOVEC instruction with an undefined register specification field in the first extension word.

Instruction word patterns with bits 15–12 equal to $A do not correspond to legal instructions for the M68040 and are treated as unimplemented instructions. $A word patterns are referred to as an unimplemented instruction with A-line opcodes. When the processor attempts to execute an unimplemented instruction with an A-line opcode, an exception is generated with vector number 10, permitting efficient emulation of unimplemented instructions. For instruction word patterns with bits 15–12 equal to $F refer to **Section 9 Floating-Point Unit (MC68040 Only)**.

Exception processing for illegal and unimplemented instructions is similar to that for instruction traps. When the processor has identified an illegal or unimplemented instruction, it initiates exception processing instead of attempting to execute the instruction. The processor copies the SR, enters the supervisor mode, and clears T1 and T0, disabling further tracing. The processor generates the vector number, either 4 or 10, according to the exception type. The illegal or unimplemented instruction vector offset, current PC, and copy of the SR are saved on the supervisor stack, with the saved value of the PC being the address of the illegal or unimplemented instruction. Instruction execution resumes at the address contained in the exception vector. It is the responsibility of the exception handling routine to adjust the stacked PC if the instruction is emulated in software or is to be skipped on return from the exception handler.

8.2.5 Privilege Violation Exception

To provide system security, some instructions are privileged. An attempt to execute one of the following privileged instructions while in the user mode causes a privilege violation exception:

ANDI to SR	FSAVE	MOVEC	PTEST
CINV	MOVE from SR	MOVES	RESET
CPUSH	MOVE to SR	ORI to SR	RTE
EORI to SR	MOVE USP	PFLUSH	STOP
FRESTORE			

Exception processing for privilege violations is similar to that for illegal instructions. When the processor identifies a privilege violation, it begins exception processing before

effective address in the stack frame. The saved PC value is the logical address of the instruction that follows the unimplemented floating-point instruction. The processor generates exception vector number 11 for the unimplemented F-line instruction exception vector, fetches the address of the F-line exception handler from the processor's exception vector table, pushes the format $2 stack frame on the system stack, and begins execution of the exception handler after prefetching instructions to fill the pipeline. The exception handler emulates the unimplemented floating-point instruction in software, maintaining user-object-code compatibility. Refer to **Section 8 Exception Processing** for details about exception vectors and format $2 stack frames.

The F-line exception handler checks for the format $2 stack frame to distinguish an unimplemented floating-point instruction from other F-line unimplemented instructions. When the exception handler for unimplemented floating-point instructions executes an FSAVE, a 26-word unimplemented instruction state frame is created (see Figure 9-10). At this point, an FSAVE instruction yields the information as listed in Table 9-16. Note that unless the instruction specifies a packed decimal real source, the state frame contains both operands (if required). For packed decimal real data format, the second operand is in the designated format of the destination floating-point data register.

The exception handler uses the information provided in the state frame to determine the instruction that it needs to emulate and the input operands to that instruction. Once the instruction has been emulated and the result is reached, the exception handler moves the result into the appropriate destination floating-point data register, discards the unimplemented instruction state frame, and returns to normal instruction flow using the RTE instruction. The limitation to this approach is that no floating-point arithmetic exceptions can be reported at the end of the emulated instruction.

The M68040FPSP not only emulates the instruction, but in addition, it ensures that if any floating-point arithmetic exceptional conditions arise from the emulation of the unimplemented instruction and if the corresponding floating-point arithmetic exception is enabled, the M68040FPSP manipulates the stack and restores the stack back into the FPU in the desired exceptional state. This effectively imitates the action of the MC68040 implemented instructions since the exception is not reported until the next floating-point instruction is encountered. This manipulation of the stack is rather complicated and is beyond the scope of this manual. Motorola recommends that the user utilize the M68040FPSP if a full exception-reporting model is required. Motorola does not provide any printed documentation other than what is embedded in the source code of the M68040FPSP.

9.6.2 Unsupported Floating-Point Data Types

An unsupported data type exception occurs when either operand to an implemented floating-point instruction is denormalized (for single-, double-, and extended-precision operands), unnormalized (for extended-precision operands), or either the source or destination data format is packed decimal real. These data types are unimplemented in the MC68040 and must be emulated in software.

10.6 INTEGER UNIT INSTRUCTION TIMINGS (Continued)

Addressing Mode	CMPA.L		CMPI		CMP2*	
	<ea> Calculate	Execute	<ea> Calculate	Execute	<ea> Calculate	Execute
Dn	1	1	1	1	—	—
An	1	1	—	—	—	—
(An)	1	1	1	1	13	2_L + 11
(An)+	2	1_L + 1	2	1_L + 1	0	0
–(An)	2	1_L + 1	2	1_L + 1	0	0
(d_{16},An)	2	1_L + 1	2	1_L + 1	13	2_L + 11
(d_{16},PC)	3	2_L + 1	3	2_L + 1	14	3_L + 11
(xxx).W, (xxx).L	1	1	2	1_L + 1	13	2_L + 11
#<xxx>	1	1	—	—	—	—
(d_8,An,Xn)	3	3	3	3	15	1_L + 14
(d_8,PC,Xn)	5	1_L + 4	5	2_L + 4	16	2_L + 14
(BR,Xn)	6	1_L + 5	6	2_L + 5	17	2_L + 15
(bd,BR,Xn)	7	1_L + 6	7	2_L + 6	18	2_L + 16
([bd,BR,Xn])	9	1_L + 8	9	2_L + 8	21	2_L + 19
([bd,BR,Xn],od)	10	1_L + 9	10	2_L + 9	22	2_L + 20
([bd,BR],Xn)	10	3_L + 7	10	4_L + 7	22	4_L + 18
([bd,BR],Xn,od)	11	3_L + 8	11	4_L + 8	23	4_L + 19

*Times listed are typical.

11.8 MC68040 THERMAL DEVICE CHARACTERISTICS

The need to efficiently cool microprocessors is becoming more important as they become more complex and require more power. In the past, the M68000 family has been able to provide a 0–70°C ambient temperature part for speeds less than 40 MHz. However, the MC68040, MC68LC040, and MC68EC040 with a 50 MHz processor clock has a maximum power dissipation for a particular operating mode, a maximum junction temperature, and a thermal resistance from the die junction to the case. This section provides a more accurate method of evaluating the environment, taking into consideration both the airflow and ambient temperature. This section also gives the user information to design a cooling method that meets both thermal performance requirements and constraints of the board environment. This section discusses the device characteristics and several methods for thermal management as well as an example of one method for cooling the MC68040, MC68LC040, and MC68EC040. The MC68040, MC68LC040, and MC68EC040 contain inherent characteristics that should be considered when evaluating a method for cooling the device. The following paragraphs discuss these die/package and power considerations for each of these parts.

NOTE

All references to the MC68040 also include the MC68LC040 and MC68EC040. Note that the MC68LC040 and MC68EC040 only implement the small buffer mode.

11.8.1 MC68040 Die and Package

The MC68040 is in a cavity-down alumina-ceramic 179-pin PGA that has a worst case thermal resistance from junction to case of 3°C/W. The package differs from previous M68000 family PGA packages that are cavity-up. The cavity-down design allows the die to be attached to the top surface of the package, increasing the part's ability to dissipate heat through the package surface or an attached heat sink. The system designer needs to determine the specific dimensions and design of the particular heat sink, considering both thermal performance and size requirements.

11.8.2 MC68040 Power Considerations

The MC68040 has a maximum power rating, which varies depending on the combination of output buffer mode and operating frequency. Note that this section assumes large buffers terminated to 2.5 V. The large buffer output mode dissipates more power than the small, and higher frequencies of operation dissipate more power than the lower frequencies.

The MC68040 allows a combination of either large or small buffers on the outputs of the miscellaneous control signals, data bus, and address bus/transfer attribute pins. The large buffers offer quicker output times, allowing for an easier logic design. However, they do so by driving about 10 times as much current as the small buffers. The designer should consider whether the quicker timings present enough advantage to justify the additional consideration to the individual signal terminations, the die power consumption, and the required cooling for the device. Since the MC68040 can be powered up in one of many output buffer modes upon reset, the actual maximum power consumption for an MC68040

Maximum power dissipation in the large buffer mode occurs for output:

Low: $P_{llb} = I^2R = (49.6\ mA)^2 \times 6\ \Omega = 14.8\ mW$

High: $P_{hlb} = I^2R = (50.8\ mA)^2 \times 12\ \Omega = 30.1\ mW$

Similar calculations for unterminated small buffers yield:

$$I = 5\ mA\ (by\ spec)$$

and

$$P = I^2R = (5\ mA)^2 \times 25\ \Omega$$

so

$$P_{hsb} = 0.625\ mW$$
$$P_{lsb} = 0.625\ mW$$

Assuming that the duty cycle of output j is driving a valid logic value instead of being three-stated as given by DC_j, then the following equation approximates total average power dissipation in the output buffers:

$$I_{Total} = \sum_{j=1}^{\text{Number of Outputs Used}} (I_j \times DC_j)^2 \times R_{effj}$$

I_j and Z_{oj} are calculated for every pin as illustrated above. In practice the above summation is carried out by groups of pins instead of individual pins.

Motorola has calculated the values for DC_j for typical situations. On an average clock there will be 37.8 pins high, 41.5 pins low, and 11.7 pins three-stated. The following examples demonstrate how to calculate the power dissipation that is added to small buffer power dissipation numbers, assuming a termination as illustrated in Figure 11-18.

a. For the numbers listed in this section in a large buffer design with no caching.

   $P = (\text{Number of Pins High}) \times (P_{hlb}) + (\text{Number of Pins Low}) \times (P_{llb})$

   $= 37.8\ \text{Pins} \times 30.1\ \text{mW per Pin} + 41.5\ \text{Pins} \times 14.8\ \text{mW per Pin}$

   $= 1.75\ W$

b. For a single bus master system in a large buffer design with no caching or snooping and only standard features (i.e., TLN, UPA, $\overline{BR}$, $\overline{BB}$, $\overline{LOCK}$, $\overline{LOCKE}$, $\overline{CIOUT}$, $\overline{TIP}$, $\overline{MI}$, TDO, $\overline{IPEND}$, PST not used):

   $P = (\text{Number of Pins High}) \times (P_{hlb}) + (\text{Number of Pins Low}) \times (P_{llb})$

   $= 29.8\ \text{Pins} \times 30.1\ \text{mW per Pin} + 34.5\ \text{Pins} \times 14.8\ \text{mW per Pin}$

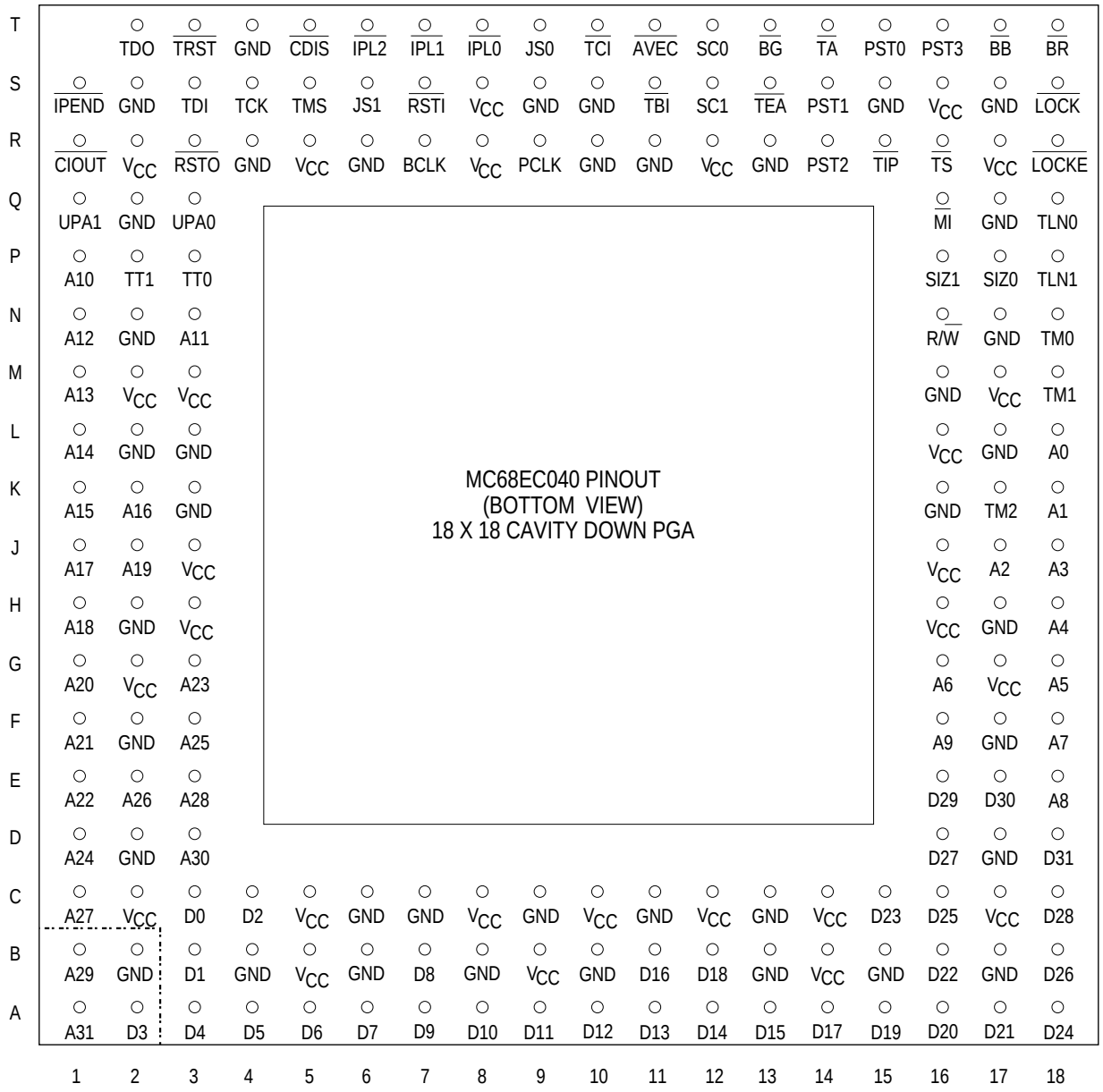
   $= 1.41\ W$

c. For the example b system with copyback caching, assuming 85% cache hit rate:

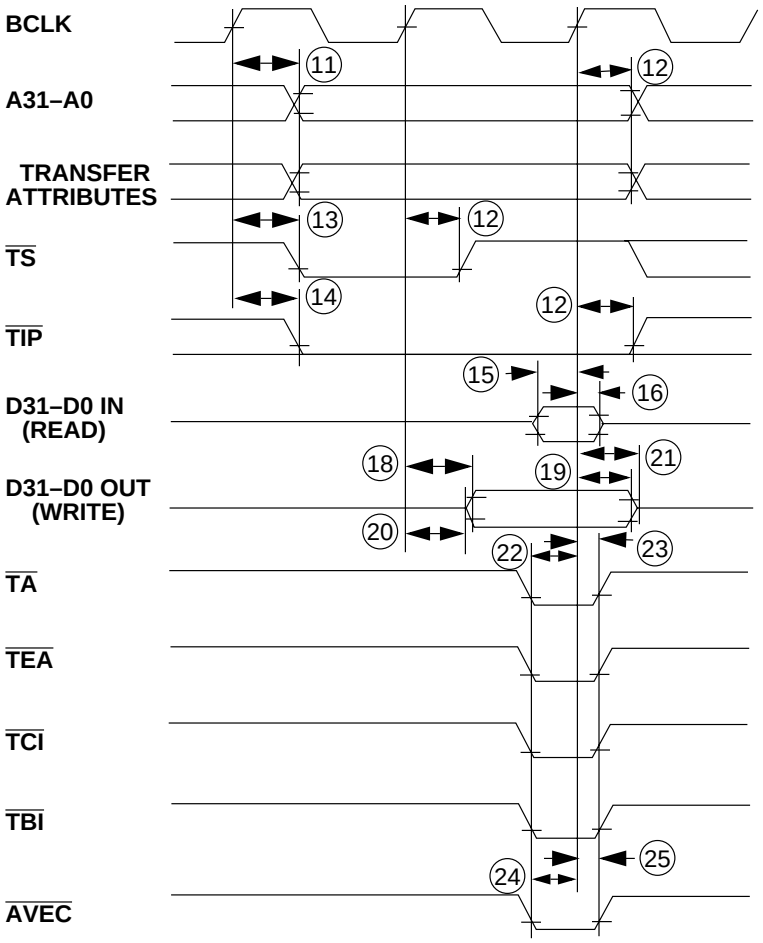
   $P = (29.8\ \text{Pins} \times 30.1\ \text{mW per Pin} + 34.5\ \text{Pins} \times 14.8\ \text{mW per Pin}) \times (1 - 0.85)$

   $= 0.21\ W$

12.2.3 MC68EC040 Pin Grid Array

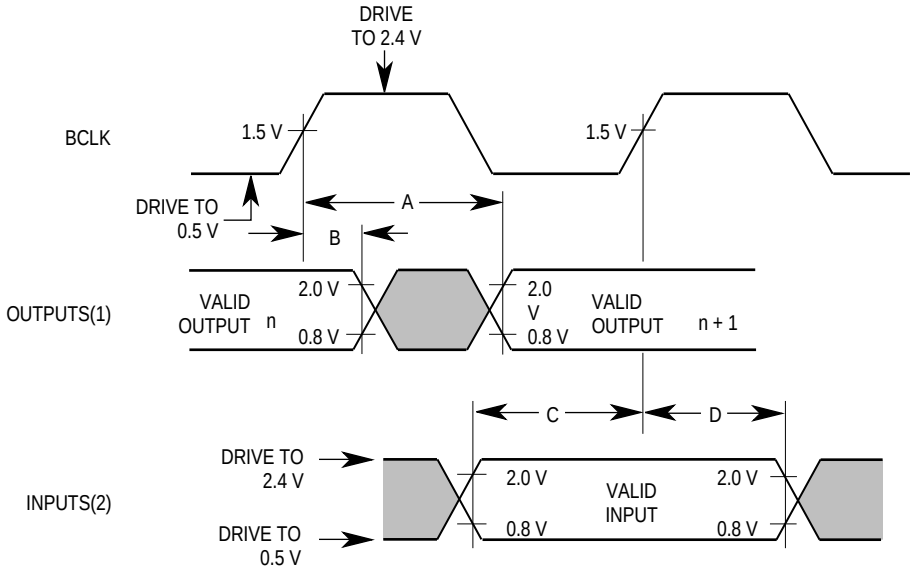


Pin Group	GND	V_{CC}
PLL	S9, R6, R10	R8, S8
Internal Logic	C6, C7, C9, C11, C13, K3, K16, L3, M16, R4, R11, R13, S6, S10, T4	C5, C8, C10, C12, C14, H3, H16, J3, J16, L16, M3, R5, R12
Output Drivers	B2, B4, B6, B8, B10, B13, B15, B17, D2, D17, F2, F17, H2, H17, L2, L17, N2, N17, Q2, Q17, S2, S15, S17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



NOTE: Transfer Attribute Signals = UPAx, SIZx, TTx, TMx, TLNx, R/$\overline{W}$, $\overline{LOCK}$, $\overline{LOCKE}$, and $\overline{CIOUT}$

Figure A-5. Read/Write Timing



NOTES:

1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This input timing is applicable to all parameters specified relative to the rising edge of the clock.

LEGEND:

A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.

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

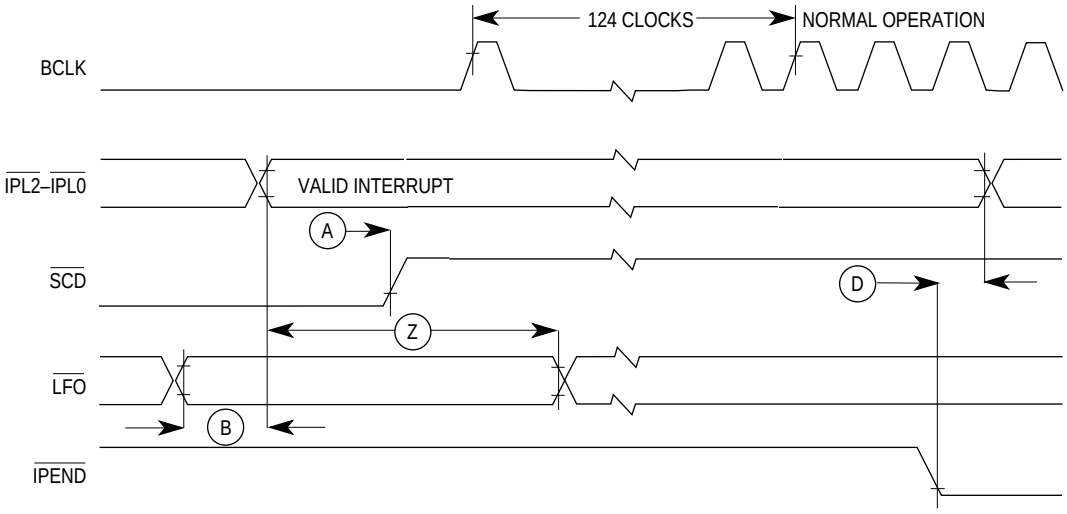


Figure C-20. Exiting LPSTOP with Interrupt

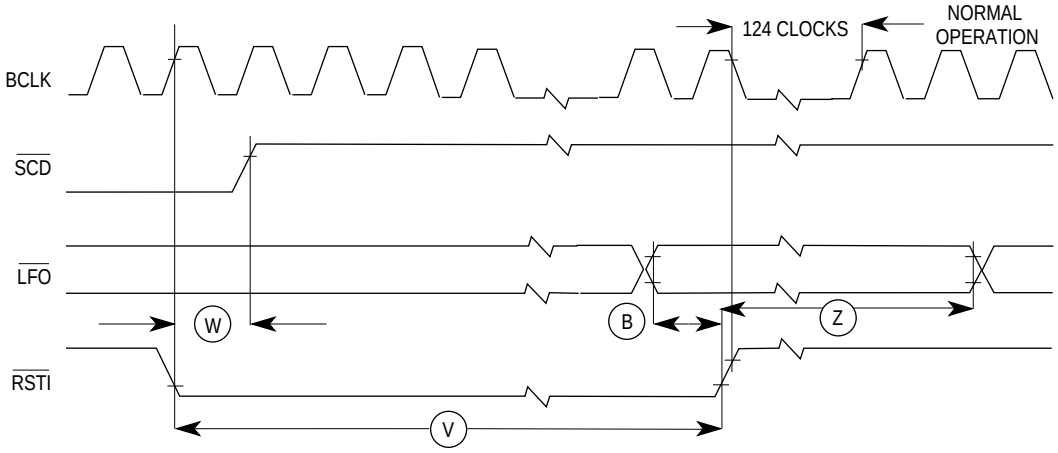


Figure C-21. Exiting of LPSTOP with RESET

MC68020, MC68030, and MC68040 Instruction Set Extensions

Instruction	Notes	Applies To		
		MC68020	MC68030	MC68040
Bcc	Supports 32-Bit Displacements			
BFxxxx	Bit Field Instructions (BCHG, BFCLR, BFEXTS, BFEXTU, BFFFO, BFINS, BFSET, BFTST)			
BKPT	New Instruction Functionally			
BRA	Supports 32-Bit Displacement			
BSR	Supports 32-Bit Displacement			
CALLM	New Instruction			
CAS, CAS2	New Instructions			
CHK	Supports 32-Bit Operands			
CHK2	New Instruction			
CINV	Cache Maintenance Instruction			
CMPI	Supports Program Counter Relative Addressing Modes			
CMP2	New Instruction			
CPUSH	Cache Maintenance Instruction			
cp	Coprocessor Instructions			
DIVS/DIVU	Supports 32-Bit and 64-Bit Operands			
EXTB	Supports 8-Bit Extend to 32-Bits			
FABS	New Instruction			
FADD	New Instruction			
FBcc	New Instruction			
FCMP	New Instruction			
FDBcc	New Instruction			
FDIV	New Instruction			
FMOVE	New Instruction			
FMOVEM	New Instruction			
FMUL	New Instruction			
FNEG	New Instruction			
FNOP	New Instruction			
FRESTORE	New Instruction			
FSGLDIV	New Instruction			
FSGLMUL	New Instruction			
FSAVE	New Instruction			
FScc	New Instruction			
FSQRT	New Instruction			
FSUB	New Instruction			
FTRAPcc	New Instruction			
FTST	New Instruction			
LINK	Supports 32-Bit Displacement			

–U–

–V–

–W–