MC68030

Technical Summary

SECOND-GENERATION 32-BIT ENHANCED MICROPROCESSOR

The MC68030 is a 32-bit virtual memory microprocessor that integrates the functionality of an MC68020 core with the added capabilities of an on-chip paged memory management unit (MMU) and an on-chip 256-byte data cache. Additionally, the MC68030 is enhanced with multiple internal address and data buses as well as a more versatile bus controller that can support two-clock bus accesses and one-clock burst accesses to maximize performance. The rich instruction set and addressing mode capabilities of the MC68020 have been maintained, allowing a clear migration path for M68000 systems. For detailed information on the MC68030, refer to MC68030 UM/AD, *MC68030 Enhanced 32-Bit Microprocessor User's Manual.*

The main features of the MC68030 are as follows:

- Object-Code Compatible with the MC68020 and Earlier M68000 Microprocessors
- Complete 32-Bit Nonmultiplexed Address and Data Buses
- 16 32-Bit General-Purpose Data and Address Registers
- Two 32-Bit Supervisor Stack Pointers and 10 Special-Purpose Control Registers
- 256-Byte Instruction Cache and 256-Byte Data Cache Can Be Accessed Simultaneously
- Paged MMU Translates Addresses in Parallel with Instruction Execution
- Two Transparent Segments Allow Untranslated Blocks To Be Defined for Systems That Transfer Large Blocks of Data to Predefined Addresses — i.e., Graphics Applications
- Pipelined Architecture with Increased Parallelism Allows Accesses from Internal Caches to Occur in Parallel with Bus Transfers and Instruction Execution To Be Overlapped

- Enhanced Bus Controller Supports Asynchronous Bus Cycles, (three clocks minimum), Synchronous Bus Cycles, (two clocks minimum), and Burst Data Transfers (one cock minumum), all to the Physical Address Space
- Dynamic Bus Sizing Supports 8-/16-/32-Bit Memories and Peripherals
- Complete Support for Coprocessors with the M68000 Coprocessor Interface
- Internal Status Indication for Hardware Emulation Support
- 4-Gbyte Direct Addressing Range
- Implemented in Motorola's HCMOS Technology That Allows CMOS and HMOS (High-Density NMOS) Gates To Be Combined for Maximum Speed, Low Power, and Small Die Size
- Processor Speeds Beyond 20 MHz

INTRODUCTION

The MC68030 is an integrated processor that incorporates the capabilities of the MC68020 microprocessor, the memory management structure defined by the MC68851 paged memory management unit (PMMU), data cache, an instruction cache, and an improved bus controller on one VLSI device. It maintains the 32-bit registers available with the entire M68000 Family as well as the 32-bit address and data paths, rich instruction set, versatile addressing modes, and flexible coprocessor interface provided with the MC68020. In addition, the internal operations of this integrated processor are designed to operate in parallel, allowing multiple instructions to be executed concurrently. It allows instruction execution to proceed in parallel with accesses to the internal caches, the on-chip MMU, and the bus controller.

The MC68030 fully supports the nonmultiplexed asynchronous bus of the MC68020 as well as the dynamic bus sizing mechanism that allows the processor to transfer operands to or from external devices while automatically determining device port size on a cycle-by-cycle basis. In addition to the asynchronous bus, the MC68030 also supports a fast synchronous bus for off-chip caches and fast memories. Furthermore, the MC68030 bus is capable of fetching up to four long words of data in a burst mode compatible with DRAM chips that have burst capability. Burst mode can reduce (up to 50 percent) the time necessary to fetch the four long words. The four long words are used to prefill the on-chip instruction and data caches so that the hit ratio of the caches is improved and the average access time for operand fetches is minimized.

The block diagram shown in Figure 1 depicts the major sections of the MC68030 and illustrates the autonomous nature of these blocks. The bus controller consists of the address and data pads, the multiplexers required to support dynamic bus sizing, and a microbus controller that schedules the bus cycles on the basis of priority. The micromachine contains the execution unit and all related control logic. Microcode control is provided by a modified two-level store of microROM and nanoROM contained in the micromachine. Programmed logic arrays (PLAs) are used to provide instruction decode and sequencing information. The instruction pipe and other individual control sections provide the secondary decode of instructions and generate the actual control signals that result in the decoding and interpretation of nanoROM and microROM information.

The instruction and data cache blocks operate independently from the rest of the machine, storing information read by the bus controller for future use with very fast access time. Each cache resides on its own address bus and data bus, allowing simultaneous access to both. Both caches are organized as a total of 64 long-word entries (256 bytes) with a line size of four long words. The data cache uses a write-through policy with programmable write allocation for cache misses.

Finally, the MMU controls the mapping of addresses for page sizes ranging from 256 bytes to 32K bytes. Mapping information stored in descriptors resides in translation tables in memory that are automatically searched by the MC68030 on demand. Recently used descriptors are maintained in a 22-entry fully associative cache called the address translation cache (ATC), allowing address translation and other MC68030 functions to occur simultaneously. Additionally, the MC68030 contains two transparent translation registers that can be used to define a one-to-one mapping for two segments ranging in size from 16 Mbytes to 2 Gbytes each.

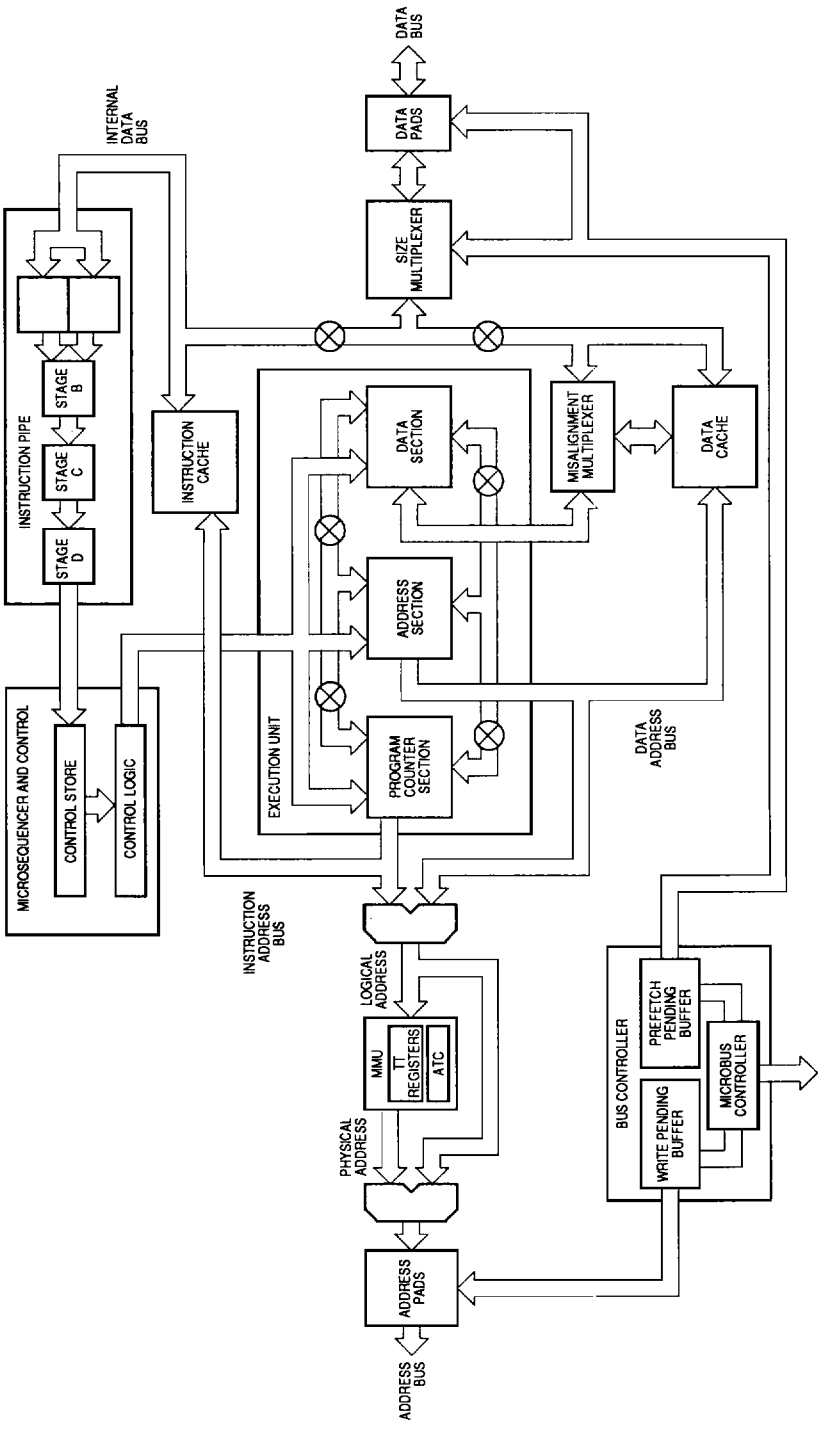


Figure 1. MC68030 Block Diagram

PROGRAMMING MODEL

As shown in the programming models (see Figures 2 and 3), the MC68030 has 16 32-bit general-purpose registers, a 32-bit program counter, two 32-bit supervisor stack pointers, a 16-bit status register, a 32-bit vector base register, two 3-bit alternate function code registers, two 32-bit cache handling (address and control) registers, two 64-bit root pointer registers used by the MMU, a 32-bit translation control register, two 32-bit transparent translation registers, and a 16-bit MMU status register. Registers D0–D7 are used as data registers for bit and bit field (1 to 32 bit), byte (8 bit), word (16 bit), long-word (32 bit), and quad-word (64 bit) operations. Registers A0–A6 and the user, interrupt, and master stack pointers are address registers that may be used as software stack pointers or base address registers. In addition, the address registers may be used for word and long-word operations. All 16 general-purpose registers (D0–D7, A0–A7) can used as index registers.

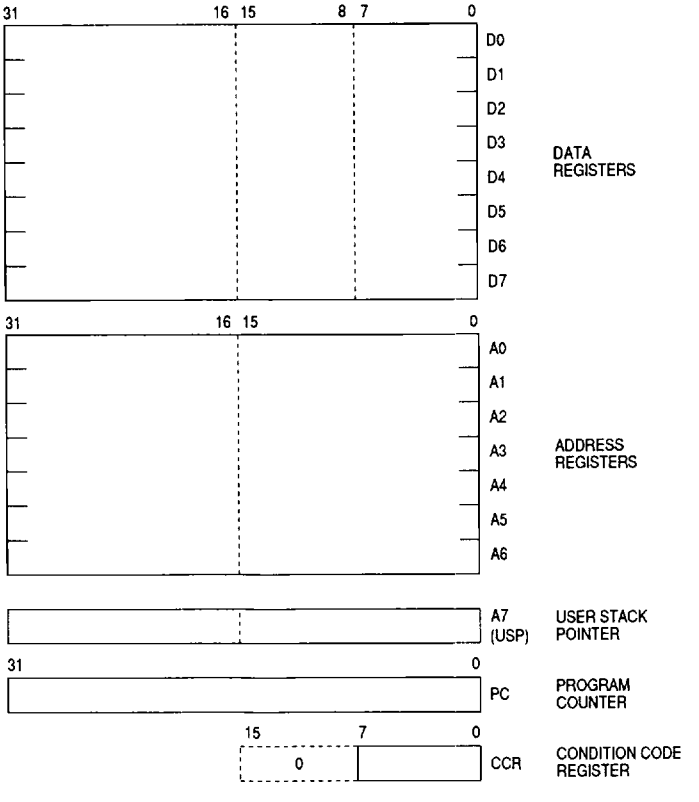


Figure 2. User Programming Model

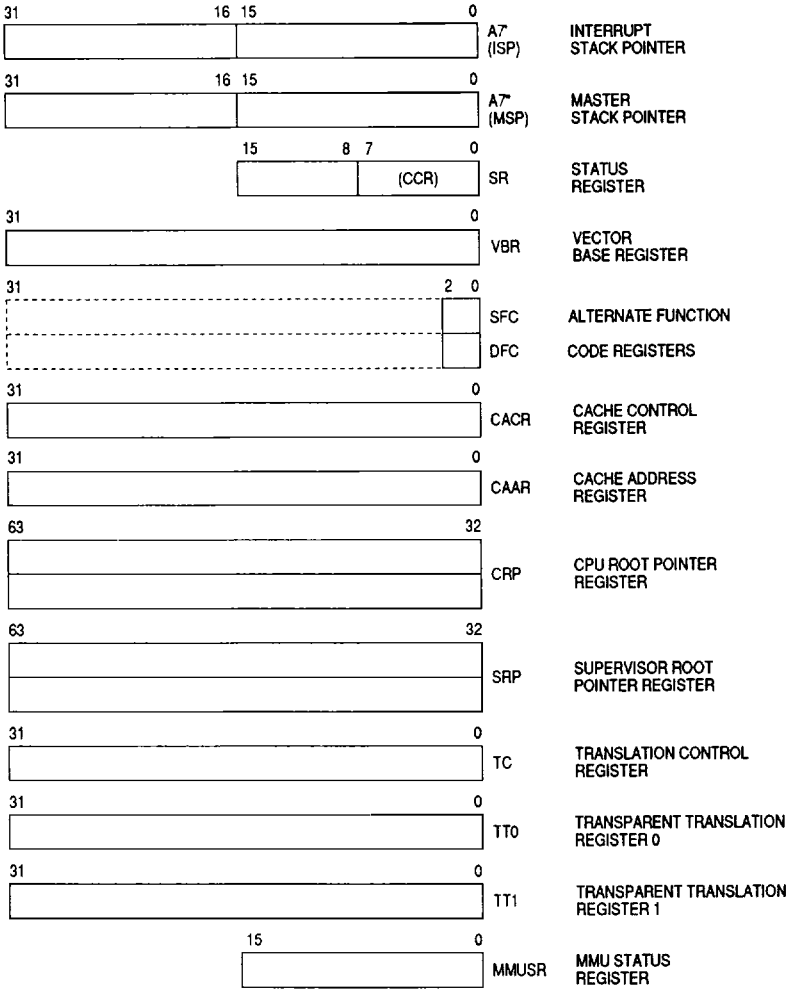


Figure 3. Supervisor Programming Model Supplement

The status register (see Figure 4) contains the interrupt priority mask (three bits) as well as the following condition codes: extend (X), negate (N), zero (Z), overflow (V), and carry (C). Additional control bits indicate that the processor is in the trace mode (T1 or T0), supervisor/user state (S), and master/interrupt state (M).

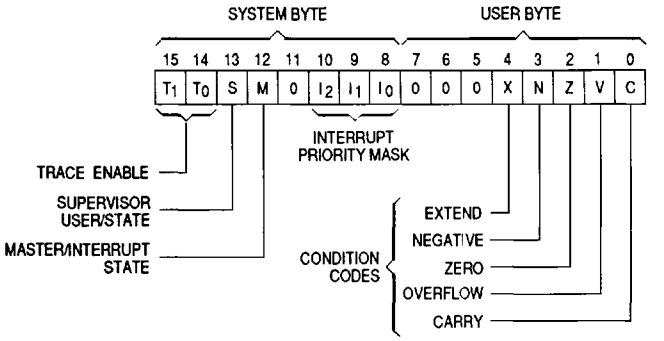


Figure 4. Status Register

All microprocessors of the M68000 Family support instruction tracing (via the T0 status bit in the MC68030) where each executed instruction is followed by a trap to a user-defined trace routine. The MC68030 also has the capability to trace only on change-of-flow instructions (branch, jump, subroutine call and return, etc.) using the T1 status bit. These features are important for software program development and debug.

Since the vector base register is used to determine the run-time location of the exception vector table in memory, it supports multiple vector tables; thus, each process or task can properly manage exceptions independent of each other.

The M68000 Family processors distinguish address spaces as supervisor/user, program/data, and CPU space. These five combinations are specified by the function code pins (FC0/FC1/FC2) during bus cycles, indicating the particular address space. Using the function codes, the memory subsystem (hardware) can distinguish between supervisor accesses and user accesses as well as program accesses, data accesses, and CPU space accesses. Additionally, the system software can configure the on-chip MMU so that supervisor/user privilege checking is performed by the address translation mechanism, and the lookup of translation descriptors can be differentiated on the basis of function code. To support the full privileges of the supervisor, the alternate function code registers allow the supervisor to specify the function code for an access by appropriately preloading the SFC/DFC registers.

The cache registers allow supervisor software manipulation of the on-chip instruction and data caches. Control and status accesses to the caches are provided by the cache control register (CACR); the cache address register (CAAR) specifies the address for those cache control functions that require an address.

All MMU registers (CRP, SRP, TC, TT0, TT1, and MMUSR) are accessible by the supervisor only. The central processor unit (CPU) root pointer contains a descriptor for the first pointer to be used in the translation table search for page descriptors pertaining to the current task. If the supervisor root pointer enable (SRE) bit of the translation control register is set, the supervisor root pointer is used as a pointer to the translation tables for all supervisor accesses. If the SRE bit is clear, this register is unused, and the CPU root pointer is used for both supervisor and user translations. The translation control register configures the table lookup mechanism to be used for all table searches as well as the page size and any initial shift of logical address required by the operating system. In addition, this register has an enable bit that enables the MMU. The transparent translation registers can be used to define two transparent windows for transferring large blocks of data with untranslated addresses. Finally, the MMU status register (MMUSR) contains status information related to a specific address translation and the results generated by the PTEST instruction. This information can be useful in locating the cause of an MMU fault.

DATA TYPES AND ADDRESSING MODES

Seven basic data types are supported by the MC68030:

1. Bits
2. Bit Fields (String of consecutive bits, 1–32 bits long)
3. BCD Digits (Packed: 2 digits/byte, Unpacked: 1 digit/byte)
4. Byte Integers (8 bits)
5. Word Integers (16 bits)
6. Long-Word Integers (32 bits)
7. Quad-Word Integers (64 bits)

In addition, operations on other data types, such as memory addresses, status word data, etc., are provided in the instruction set. The coprocessor mechanism allows direct support of floating-point data types with the MC68881/MC68882 floating-point coprocessors as well as specialized user-defined data types and functions.

The 18 addressing modes listed in Table 1 include nine basic types:

1. Register Direct
2. Register Indirect
3. Register Indirect with Index
4. Memory Indirect
5. Program Counter Indirect with Displacement
6. Program Counter Indirect with Index
7. Program Counter Memory Indirect
8. Absolute
9. Immediate

The register indirect addressing modes support postincrement, predecrement, offset, and indexing. These capabilities are particularly useful for handling advanced data structures common to sophisticated applications and high-level languages. The program counter relative mode also has index and offset capabilities; this addressing mode is generally required to support position-independent software. In addition to these addressing modes, the MC68030 provides data operand sizing and scaling; these features provide performance enhancements for the programmer.

Table 1. Addressing Modes

Addressing Modes	Syntax
Register Direct	
Data Register Direct	Dn
Address Register Direct	An
Register Indirect	
Address Register Indirect	(An)
Address Register Indirect with Postincrement	(An)+
Address Register Indirect with Predecrement	−(An)
Address Register Indirect with Displacement	(d_{16},An)
Register Indirect with Index	
Address Register Indirect with Index (8-Bit Displacement)	(d_8,An,Xn)
Address Register Indirect with Index (Base Displacement)	(bd,An,Xn)
Memory Indirect	
Memory Indirect Postindexed	([bd,An],Xn,od)
Memory Indirect Preindexed	([bd,An,Xn],od)
Program Counter Indirect with Displacement	(d_{16},PC)
Program Counter Indirect with Index	
PC Indirect with Index (8-Bit Displacement)	(d_8,PC,Xn)
PC Indirect with Index (Base Displacement)	(bd,PC,Xn)
Program Counter Memory Indirect	
PC Memory Indirect Postindexed	([bd,PC],Xn,od)
PC Memory Indirect Preindexed	([bd,PC,Xn],od)
Absolute	
Absolute Short	(xxx).W
Absolute Long	(xxx).L
Immediate	#⟨data⟩

NOTES:

Dn = Data Register, D0–D7

An = Address Register, A0–A7

d_8, d_{16} = A twos-complement or sign-extended displacement; added as part of the effective address calculation; size is 8 (d_8) or 16 (d_{16}) bits; when omitted, assemblers use a value of zero.

Xn = Address or data register used as an index register; form is Xn.SIZE*SCALE, where SIZE is .W or .L (indicates index register size) and SCALE is 1, 2, 4, or 8 (index register is multiplied by SCALE); use of SIZE and/or SCALE is optional.

bd = A twos-complement base displacement; when present, size can be 16 or 32 bits.

Table 1. Addressing Modes (Continued)

od = Outer displacement, added as part of effective address calculation after any memory indirection; use is optional with a size of 16 or 32 bits.
PC = Program Counter
⟨data⟩ = Immediate value of 8, 16, or 32 bits
( ) = Effective Address
[ ] = Use as indirect access to long-word address.

INSTRUCTION SET OVERVIEW

The MC68030 instruction set is listed in Table 2. Each instruction, with few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 18 addressing modes. The MC68030 is upward source- and object-code compatible with the M68000 Family because it supports all instructions of previous family members. Included in this set are the bit field operations, binary-coded decimal support, bounds checking, additional trap conditions, and additional multiprocessing support (CAS and CAS2 instructions) offered by the MC68020. The new instructions supported by the MC68030, a subset of the instructions introduced by the MC68851 PMMU, are used to communicate with the MMU. For detailed information on the MC68030 instruction set, refer to M68000 PM/AD, *M68000 Programmer's Reference Manual.*

Table 2. Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
tCAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
CMP2	Compare Register Against Upper and Lower Bounds
DBcc	Test Condition, Decrement and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive OR
EORI	Logical Exclusive OR Immediate
EXG	Exchange Registers
EXT, EXTB	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine

Table 2. Instruction Set (Continued)

Mnemonic	Description
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE CCR	Move Condition Code Register
MOVE SR	Move Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive OR
ORI	Logical Inclusive OR Immediate
ORI CCR	Logical Inclusive OR Immediate to Condition Codes
ORI SR	Logical Inclusive OR Immediate to Status Register
PACK	Pack BCD
PEA	Push Effective Address

Mnemonic	Description
PFLUSH	Flush Entry(ies) in the ATC
PFLUSHA	Flush All Entries in the ATC
PLOADR, PLOADW	Load Entry into the ATC
PMOVE	Move to/from MMU Registers
PMOVEFD	Move to/from MMU Registers with Flush Disable
PTESTR, PTESTW	Test a Logical Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTR	Return and Restore Codes
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

Coprocessor Instructions

cpBCC	Branch Conditionally
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGEN	Coprocessor General Instruction
cpRESTORE	Restore Internal State of Coprocessor
cpSAVE	Save Internal State of Coprocessor
cpScc	Set Conditionally
cpTRAPcc	Trap Conditionally

INSTRUCTION AND DATA CACHES

Studies have shown that typical programs spend most of their execution time in a few main routines or tight loops. This phenomenon, known as locality of reference, has an impact on program performance. The MC68010 takes limited advantage of this phenomenon with the loop mode of operation that can be used with the DBcc instruction. The MC68030 takes further advantage of cache technology to provide the system with two on-chip caches, one for instructions and one for data.

MC68030 CACHE GOALS

Similar to the MC68020, there were two primary design goals for the MC68030 microprocessor caches. The first design goal was to reduce the processor external bus activity even more than what was accomplished with the MC68020. The second design goal was to increase effective CPU throughput as larger memory sizes or slower memories increased average access time. By placing a high-speed cache between the processor and the rest of the memory system, the effective memory access time becomes:

$$t_{acc} = h * t_{cache} + (1 - h) * t_{ext}$$

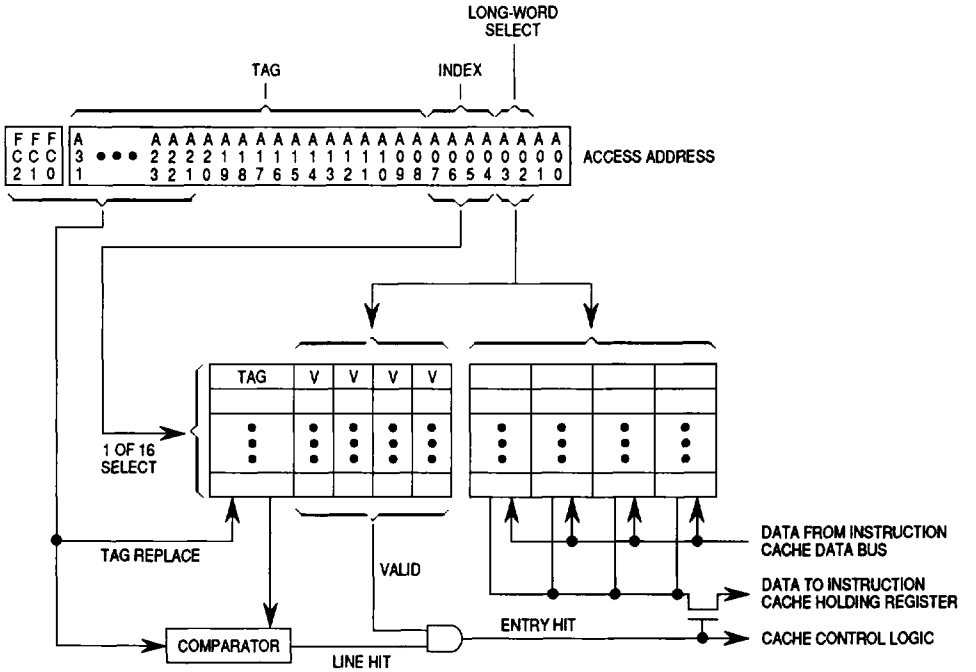
where t_{acc} is the effective system access time, t_{cache} is the cache access time, t_{ext} is the access time of the rest of the system, and h is the hit ratio or the percentage of time that the data is found in the cache. Thus, for a given system design, two MC68030 on-chip caches provide an even more substantial CPU performance increase over that obtainable with the MC68020 instruction cache. Alternately, slower and less expensive memories can be used for the same processor performance.

The throughput increase in the MC68030 is gained in three ways. First, the MC68030 caches are accessed in less time than is required for external accesses, providing improvement in the access time for items residing in the cache. Second, the burst filling of the caches allows instruction and data words to be found in the on-chip caches the first time they are accessed by the micromachine, minimizing time required to bring those items into the cache. Burst filling lowers the average access time for items found in the caches even further. Third, the autonomous nature of the caches allows instruction stream fetches, data fetches, and a third external access to occur simultaneously with instruction execution. For example, if the MC68030 requires both an instruction stream access and an external peripheral access and if the instruction is resident in the on-chip cache, the peripheral access proceeds unimpeded rather than being queued behind the instruction fetch. If a data operand is also required and is resident in the data cache, it can also be accessed without hindering either the

instruction access or the external peripheral access. The parallelism designed into the MC68030 also allows multiple instructions to execute concurrently so that several internal instructions (those that do not require any external accesses) can execute while the processor is performing an external access for a previous instruction.

INSTRUCTION CACHE

The MC68030 instruction cache is a 256-byte direct-mapped cache organized as 16 lines consisting of four long words per line. Each long word is independently accessible, yielding 64 possible entries, with address bit A1 selecting the correct word during an access. Thus, each line has a tag field composed 24 address bits, the FC2 (supervisor/user) value, four valid bits (one for each long-word entry), and the four long-word entries (see Figure 5). The instruction cache is automatically filled by the MC68030 whenever a cache miss occurs; using the burst transfer capability, up to four long words can be filled in one burst operation. The caches can not be manipulated directly by the programmer except by the use of the CACR, which provides cache clearing and cache entry clearing facilities. The caches can also be enabled/disabled by this register. Finally, the system hardware can disable the on-chip caches at any time by asserting of the $\overline{\text{CDIS}}$ signal.



CACHE SIZE = 64 (LONG WORDS)
LINE SIZE = 4 (LONG WORDS)
SET SIZE = 1

Figure 5. On-Chip Instruction Cache Organization

DATA CACHE

The organization of the data cache (see Figure 6) is similar to that of the instruction cache. However, the tag is composed of the upper 24 address bits, the four valid bits, and all three function code bits, explicitly specifying the address space associated with each line. The data cache employs a write-through policy with programmable write allocation of data writes i.e., if a cache hit occurs on a write cycle, both the data cache and the external device are updated with the new data. If a write cycle generates a cache miss, the external device is updated, and a new data cache entry can be replaced or allocated for that address, depending on the state of the write-allocate (WA) bit in the CACR.

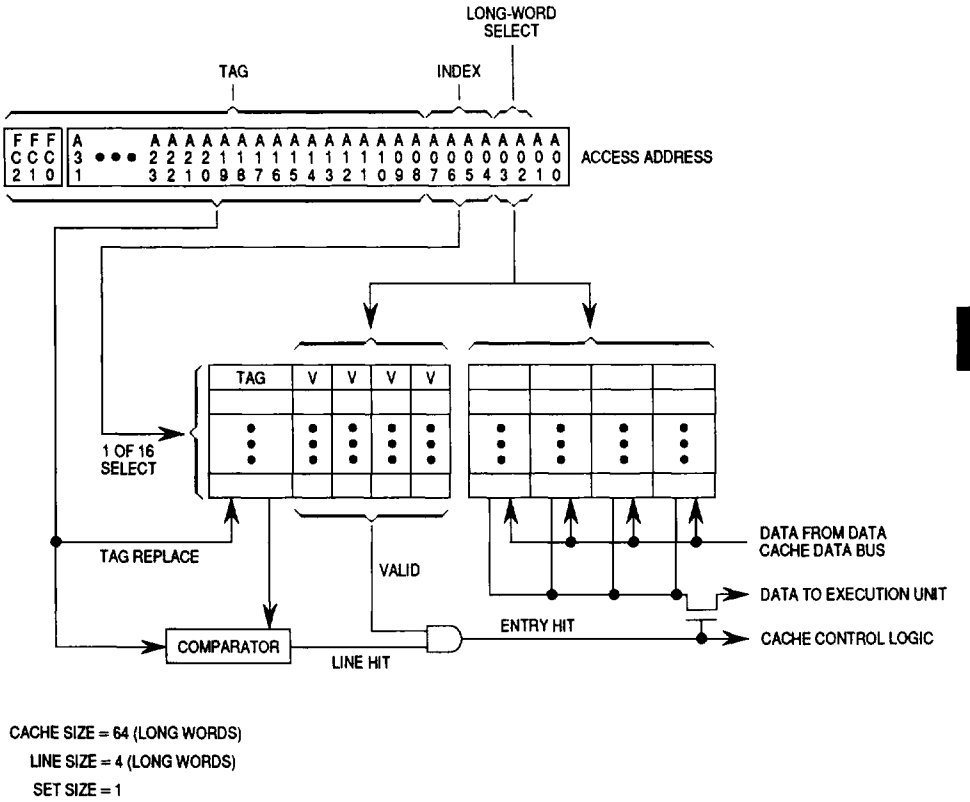


Figure 6. On-Chip Data Cache Organization

OPERAND TRANSFER MECHANISM

The MC68030 offers three different mechanisms by which data can be transferred into and out of the chip. Asynchronous bus cycles, compatible with the asynchronous bus on the MC68020, can transfer data in a minimum of three clock cycles; the amount of data transferred on each cycle is determined by the dynamic bus sizing mechanism on a cycle-by-cycle basis with the data transfer and size acknowledge ($\overline{\text{DSACKx}}$) signals. Synchronous bus cycles are terminated with the synchronous termination ($\overline{\text{STERM}}$) signal and always transfer 32-bits of data in a minimum of two clock cycles, increasing the bus bandwidth available for other bus masters, therefore increasing possible performance. Burst mode transfers can be used to fill lines of the instruction and data caches when the MC68030 asserts cache burst request ($\overline{\text{CBREQ}}$). After completing the first cycle with $\overline{\text{STERM}}$, subsequent cycles may accept data on every clock cycle where $\overline{\text{STREM}}$ is asserted until the burst is completed. Use of this mode can

further increase the available bus bandwidth in systems that use DRAMs with page, nibble, or static-column mode operation.

ASYNCHRONOUS TRANSFERS

Though the MC68030 has a full 32-bit data bus, it offers the ability to automatically and dynamically downsize its bus to 8 or 16 bits if peripheral devices are unable to accommodate the entire 32 bits. This feature allows the programmer to write code that is not bus-width specific. For example, long-word (32 bit) accesses to peripherals may be used in the code; yet, the MC68030 will transfer only the amount of data that the peripheral can manage at one time. This feature allows the peripheral to define its port size as 8, 16, or 32 bits wide, and the MC68030 will dynamically size the data transfer accordingly, using multiple bus cycles when necessary. Hence, programmers are not required to program for each device port size or know the specific port size before coding; hardware designers have the flexibility to choose hardware implementations regardless of software implementations.

The dynamic bus sizing mechanism is invoked by $\overline{\text{DSACKx}}$ and occurs on a cycle-by-cycle basis. For example, if the processor is executing an instruction that requires the reading of a long-word operand, it will attempt to read 32 bits during the first bus cycle to a long-word address boundary. If the port responds that it is 32 bits wide, the MC68030 latches all 32 bits of data and continues. If the port responds that it is 16 bits wide, the MC68030 latches the 16 valid bits of data and runs a second cycle to obtain the remaining 16 bits of data. An 8-bit port is handled similarly but has four bus read cycles. Each port is fixed in the assignment to particular sections of the data bus. However, the MC68030 has no restrictions concerning the alignment of operands in memory; long-word operands need not be aligned to long-word address boundaries. When misaligned data requires multiple bus cycles, the MC68030 automatically runs the minimum number of bus cycles. Instructions must still be aligned to word boundaries.

The timing of asynchronous bus cycles is also determined by the assertion of the $\overline{\text{DSACKx}}$ signals on a cycle-by-cycle basis. If the $\overline{\text{DSACKx}}$ signals are valid 1.5 clocks after the beginning of the bus cycle (with the appropriate setup time), the cycle terminates in the minimum amount of time (corresponding to three clock cycles total). The cycle can be lengthened by delaying $\overline{\text{DSACKx}}$ (effectively inserting wait states in one-clock increments) until the device being accessed is able to terminate the cycle. This flexibility gives the processor the ability to communicate with devices of varying speeds while operating at the fastest rate possible for each device.

The asynchronous transfer mechanism allows external errors to abort cycles upon the assertion of bus error ($\overline{BERR}$), or allows individual bus cycles to be retried with the simultaneous assertion of $\overline{BERR}$ and $\overline{HALT}$.

SYNCHRONOUS TRANSFERS

Synchronous bus cycles are terminated by asserting $\overline{STERM}$, which automatically indicates that the port size is 32 bits. Since this is asynchronous input, two-clock-cycle bus accesses can be performed if the signal is valid one clock after the beginning of the bus cycle with the appropriate setup time. However, the bus cycle may be lengthened by delaying $\overline{STERM}$ (inserting wait states in one-clock increments) until the device being accessed is able to terminate the cycle. Additionally, these cycles may be aborted upon the assertion of $\overline{BERR}$, or they may be retried with the simultaneous assertion of $\overline{BERR}$ and $\overline{HALT}$.

BURST READ CYCLES

The MC68030 provides support for burst filling of its on-chip instruction and data caches, adding to the overall system performance. The on-chip caches are organized with a line size of four long words with one tag for the four long words in a line. Since locality of reference is present to some degree in most programs, filling of all four entries when a single entry misses can be advantageous, especially if the time spent filling the additional entries is minimal. When the caches are burst filled, data can be latched by the processor in as little as one clock for each 32 bits.

Burst read cycles can be performed only when the MC68030 requests them (with the assertion of $\overline{CBREQ}$) and only when the first cycle is a synchronous cycle as previously described. If the cache burst acknowledge ($\overline{CBACK}$) input is valid at the appropriate time in the synchronous bus cycle, the processor keeps the original $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, address, function code, and size outputs asserted and latches 32 bits from the data bus at the end of each subsequent clock cycle that has $\overline{STERM}$ asserted. This procedure continues until the burst is complete (the entire block has been transferred), $\overline{BERR}$ is asserted in lieu of or after $\overline{STERM}$, the cache inhibit in ($\overline{CIIN}$) input is asserted, or the $\overline{CBACK}$ input is negated.

EXCEPTIONS

The types of exceptions and the exception processing sequence are discussed in the following paragraphs.

TYPES OF EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are interrupts, bus error ($\overline{BERR}$), and reset ($\overline{RESET}$). Interrupts are requests from peripheral devices for processor action; whereas, $\overline{BERR}$ and $\overline{RESET}$ are used for access control and processor restart. The internally generated exceptions come from instructions, address errors, tracing, or breakpoints. The TRAP, TRAPcc, TRAPVcc, cpTRAPcc, CKH, CKH2, and DIV instructions can all generate exceptions as part of instruction execution. Tracing behaves like a very high-priority, internally generated interrupt whenever it is processed. The other internally generated exeptions are caused by illegal instructions, instruction fetches from odd addresses, and privilege violations. Finally, the MMU can generate exceptions when it detects an invalid translation in the address translation cache (ATC) and an access to the corresponding address is attempted, or when it is unable to locate a valid translation for an address in the translation tables.

EXCEPTION PROCESSING SEQUENCE

Exception processing occurs in four steps. During the first step, an internal copy is made of the status register. After the copy is made, the special processor state bits in the status register are changed. The S bit is set, putting the processor into the supervisor state. Also, the T1 and T0 bits are negated, allowing the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor read that is classified as an interrupt acknowledge cycle. For coprocessor-detected exceptions, the vector number is included in the coprocessor exception primitive response. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status. The exception stack frame is created and filled on the current supervisor stack. To minimize the amount of machine state that is saved, various stack frame sizes are used to contain the processor state, depending on the type of exception and where it occurred

during instruction execution. If the exception is an interrupt and the M bit is set, the M bit is then cleared, and the short four-word exception stack frame that is saved on the master stack is also saved on the interrupt stack. If the exception is a reset, the M bit is simply cleared, and the reset vector is accessed.

The MC68030 provides the same extensions to the exeption stacking process as the MC68020. If the M bit is set, the master stack pointer (MSP) is used for all task-related exceptions. When a nontask-related exception occurs (i.e., an interrupt), the M bit is cleared, and the interrupt stack pointer (ISP) is used. This feature allows all the task's stack area to be carried within a single processor control block, and new tasks can be initiated by simply reloading the master stack pointer and setting the M bit.

The fourth and last step of exception processing is the same for all exceptions. The exception vector offset is determined by mutliplying the vector number by four. This offset is then added to the contents of the vector base register (VBR) to determine the memory address of the exception vector. The new program counter is fetched from the exception vector. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

$\overline{\text{STATUS}}$ and $\overline{\text{REFILL}}$

The MC68030 provides the $\overline{\text{STATUS}}$ and $\overline{\text{REFILL}}$ signals to identify internal microsequencer activity associated with the processing of pipelined data. Since bus cycles are independently controlled and scheduled by the bus controller, information concerning the processing state of the microsequencer is not available by monitoring bus signals by themselves. The internal activity identified by the $\overline{\text{STATUS}}$ and $\overline{\text{REFILL}}$ signals include instruction boundaries, some exception conditions, when the microsequencer has halted, and instruction pipeline refills. $\overline{\text{STATUS}}$ and $\overline{\text{REFILL}}$ track only the internal microsequencer activity and are not directly related to bus activity.

ON-CHIP MEMORY MANAGEMENT UNIT

The full addressing range of the MC68030 is 4 Gbytes (4,294,967,296 bytes); however, most MC68030 systems implement a smaller physical memory. Nonetheless, by using virtual memory techniques, the system can be made to appear to have the full 4 Gbytes of physical memory available to each user program. In a similar fashion, a virtual system provides user-program access to other devices not physically present in the system, such as tape drives, disk drives,

printers, or terminals. The MC68030 MMU provides support for a virtual system and virtual memory. In addition, it protects supervisor areas from accesses by user programs and provides write protection on a page basis. All this capability is provided as well as maximum performance because address translations occur in parallel with other processor activities.

DEMAND-PAGED IMPLEMENTATION

A typical MC68030 system with a large addressing range provides a limited amount of high-speed physical memory that can be accessed directly by the processor while maintaining an image of a much larger virtual memory on secondary storage devices such as large-capacity disk drives. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, the access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory; the suspended access is then either restarted or continued.

A paged system is one in which the physical memory is subdivided into equal-sized blocks called page frames and the logical (untranslated) address space of a task is divided into pages having the same size as the page frames. The operating system controls the allocation of pages to page frames, bringing in data on a page basis as it is needed from the secondary storage device. The MC68030 memory management scheme is called a demand implementation because a process does not need to specify in advance the required areas of its logical address space. An access to a logical address is interpreted by the system as a request for the corresponding page.

The MC68030 MMU employs the same address translation mechanism introduced by the MC68851 PMMU, with possible page sizes ranging from 256 bytes to 32K bytes.

TRANSLATION MECHANISM

Since logical-to-physical address translation is the most frequently executed operation of the MC68030 MMU, this task has been optimized and can function autonomously. The MMU initiates address translation by searching for the address translation information (a page descriptor) in the on-chip address translation cache (ATC). The ATC is a very fast fully associative cache memory that stores recently used page descriptors. If the descriptor does not reside in the ATC, then the MMU requests external bus cycles of the bus controller to search the translation tables in physical memory. After being located, the page descriptor is loaded into the ATC, and the address is correctly translated for the access if no exception conditions are encountered.

The status of the page in question is easily maintained in the translation tables. When a page must be brought in from a secondary storage device, the table entry can signal that this descriptor is invalid so that the table search results in an invalid descriptor being loaded into the ATC. In this way, the access to the page is aborted, and the processor initiates bus error exception processing for this address. The operating system can then control the allocation of a new page in physical memory and can load the page during the bus error handling routine.

ADDRESS TRANSLATION CACHE

An integral part of the translation function previously described is the cache memory that stores recently used logical-to-physical address translation information or page descriptors. This cache consists of 22 entries and is fully associative. The ATC compares the logical address and function code of the incoming access against its entries. If one of the entries matches, there is a hit, and the ATC sends the physical address to the bus controller, which then starts the external bus cycle (provided no hit occurred in the instruction or data caches for the access).

The ATC is composed of three major components: the content-addressable memory (CAM) containing the logical address and function code information to be compared against incoming logical addresses, the physical address store containing the physical address associated with a particular CAM entry, and the control section containing the entry replacement circuitry that implements the replacement algorithm (a variation of the least recently used algorithm).

TRANSLATION TABLES

The translation tables supported by the MC68030 have a tree structure, minimizing the amount of memory necessary to set up the tables for most programs since only a portion of the complete tree needs to exist at any one time. The root of a translation table tree is pointed to by one or two root pointer registers that are part of the programmer's model: the CPU and supervisor. Table entries at the higher levels of the tree (pointer tables) contain pointers to other tables. Entries at the leaf level (page tables) contain page descriptors. The mechanism for performing table searches uses portions of the logical address as indices for each level of the lookup. All addresses in the translation table entries are physical addresses.

Figure 7 illustrates the translation table structure. Several determinants of the detailed table structure are software selectable. The first level of lookup in the table normally uses the function codes as an index, but this may be suppressed if desired. In addition, up to 15 of the logical address lines can be ignored for the purposes of the table searching. The number of levels in the table indexed by the logical address can be set from one to four, and up to 15 logical address bits can be used as an index at each level. A major advantage to using this tree structure for the translation tables is the ability to deallocate large portions of the logical address space with a single entry at the higher levels of the tree. Additionally, portions of the tree itself may reside on a secondary storage device or may not exist at all until they are required by the system.

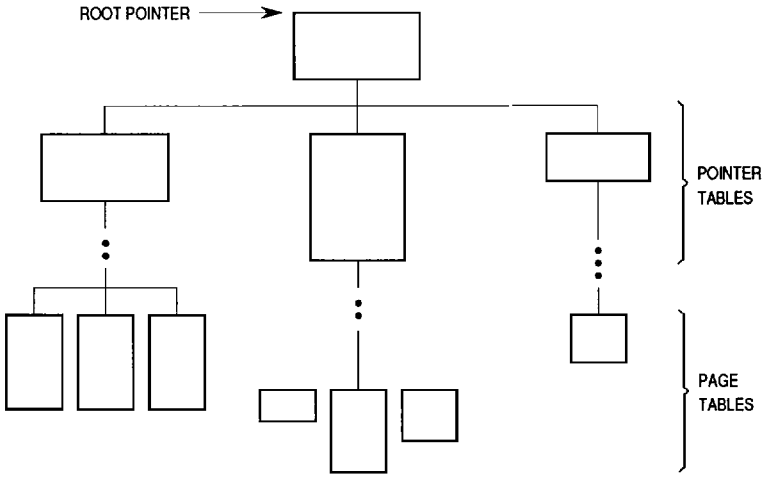


Figure 7. MMU Translation Table Structure

The entries in the translation tables contain status information pertaining to the pointers for the next level of lookup or for the pages themselves. These bits can be used to designate certain pages or blocks of pages as supervisor-only, write-protected, or noncachable. If a page is marked as noncachable, accesses within the page will not be cached by the instruction or data caches, and the cache inhibit out ($\overline{\text{CIOUT}}$) signal is asserted for those accesses. In addition, the MMU automatically maintains history information for the pointers and pages in the descriptors via the used (U) and modified (M) bits.

MMU INSTRUCTIONS

The MMU instructions supported by the MC68030, the PMOVE, PTEST, PLOAD, PFLUSH, and PFLUSHA instructions, are completely compatible with the corresponding instructions introduced by the MC68851 PMMU. Whereas the MC68851 required the coprocessor interface to execute its instructions, the MC68030 MMU instructions execute just like all other CPU instructions. All MMU instructions are privileged (can be executed by the supervisor only) and are summarized as follows:

PMOVE	Used to move data to or from MMU registers.
PTEST	Takes an address and function code and searches the ATC or the translation tables for the corresponding entry. The results of the search are available in the MMU status register (MMUSR) and are often useful in determining the cause of a fault.
PLOAD	Takes an address and function code and searches the translation tables for the corresponding page descriptor. It then loads the ATC with the appropriate information.
PFLUSH	Flushes the ATC by function code or function code and logical address.
PFLUSHA	Flushes all ATC entries.

TRANSPARENT TRANSLATION

Two transparent translation registers are provided on the MC68030 MMU to allow portions of the logical address space to be transparently mapped and accessed without corresponding entries resident in the ATC. Each register is used to define a range of logical addresses from 16 Mbytes to 2 Gbytes with a base address and a mask. All addresses within these ranges will not be mapped, and protection is provided only on a basis of read/write and function code.

COPROCESSOR INTERFACE

The coprocessor interface is a mechanism for extending the instruction set of the M68000 Family. The interface provided on the MC68030 is the same as that on the MC68020. Examples of these extensions are the addition of specialized data operands for the existing data types or, for the case of floating point, the inclusion of new data types and operations implemented by the MC68881/MC68882 floating-point coprocessors.

Coprocessors are divided into two types by their bus-utilization characteristics. A DMA coprocessor can control the bus independent of the main processor. A non-DMA coprocessor cannot control the bus. Both coprocessor types utilize the same protocol and main processor resources. Implementation of a coprocessor as a DMA or non-DMA is based primarily on coprocessor bus bandwidth requirements, performance, and cost.

The communication protocol between the main processor and the coprocessor necessary to execute a coprocessor instruction is based on a group of coprocessor interface registers (CIRs), which are defined for the M68000 Family (see Table 3) and are implemented on the coprocessor. The MC68030 hardware uses standard read and write cycles to access the registers. Thus, the coprocessor interface does not require special bus hardware; the bus interface implemented by a coprocessor for its interface register set must only satisfy the MC68030 address, data, and control signal timing to guarantee proper communication with the CPU. Since the MC68030 implements the communication protocol with all coprocessors in hardware (and microcode) and handles all operations automatically, the programmer is only concerned with the instructions and data types provided by the coprocessor as extensions to the MC68030 instruction set and data types.

Table 3. Coprocessor Interface Registers

Register	Function	R/$\overline{W}$
Response	Requests Action from CPU	R
Control	CPU Directed Control	$\overline{W}$
Save	Initiate Save of Internal State	R
Restore	Initiate Restore of Internal State	R/$\overline{W}$
Operation Word	Current Coprocessor Instruction	$\overline{W}$
Command Word	Coprocessor Specific Command	$\overline{W}$
Condition Word	Condition to be Evaluated	$\overline{W}$
Operand	32-Bit Operand	R/$\overline{W}$
Register Select	Specifies CPU Register or Mask	R
Instruction Address	Pointer to Coprocessor Instruction	R/$\overline{W}$
Operand Address	Pointer to Coprocessor Operand	R/$\overline{W}$

Since the CIRs are accessed via normal read and write cycles, coprocessors can be used as peripheral devices by other M68000 Family members that do not support the coprocessor interface. The communication protocol can be easily emulated by appropriately addressing the CIRs and by passing the required coprocessor commands and operands. In addition to the CIRs, the coprocessor contains those registers added to the MC68030 programmer's model

for specific coprocessor operations. For example, the Motorola floating-point coprocessors contain the CIRs as well as eight 80-bit floating-point data registers and three 32-bit control/status registers.

Up to eight coprocessors are supported in a single MC68030 system with a system-unique coprocessor identifier encoded in the coprocessor instruction. When accessing a coprocessor, the MC68030 executes standard bus cycles in CPU address space, as encoded by the function codes, and places the coprocessor identifier on the address bus to be used by chip-select logic to select the particular coprocessor. Since standard bus cycles are used, the coprocessor may be located according to system design requirements, whether it is located on the microprocessor local bus, on another board on the system bus, or any other place supported by the chip-select and coprocessor protocol using standard bus cycles.

COPROCESSOR PROTOCOL

Interprocessor transfers are all initiated by the main processor during coprocessor instruction execution. When processing a coprocessor instruction, the main processor transfers instruction information and data to the associated coprocessor and receives data, requests, and status information from the coprocessor. These transfers are all based on standard read and write bus cycles.

The typical coprocessor protocol for the main processor is as follows:

A. The main processor initiates the communication by writing command information to a location in the coprocessor interface.

B. The main processor reads the coprocessor response to that information.
   1. The response may indicate that the coprocessor is busy, and the main processor should requery the coprocessor, allowing the main processor and coprocessor to synchronize their concurrent operations.
   2. The response may indicate some exception condition; the main processor acknowledges the exception and begins exception processing.
   3. The response may indicate that the coprocessor needs the main processor to perform some service such as transferring data to or from the coprocessor. The coprocessor may also request that the main processor requery the coprocessor after the service is complete.
   4. The response may indicate that the main processor is not needed for further processing of the instruction. The communication is terminated, and the main processor is free to begin execution of the next instruction. At this point in the coprocessor protocol, as the main processor continues to execute the instruction stream, the main processor may operate concurrently with the coprocessor.

When the main processor encounters the next coprocessor instruction, the main processor queries the coprocessor until the coprocessor is ready; meanwhile, the main processor can service interrupts and perform a context switch to execute other tasks.

Each coprocessor instruction type has specific requirements based on this simplified protocol. The coprocessor interface may use as many extension words as required to implement a coprocessor instruction.

3 PRIMITIVE/RESPONSE

The coprocessor response register communicates service requests to the main processor. The content of the coprocessor response register is a primitive instruction to the main processor, which is read during coprocessor communication by the main processor. The main processor executes this primitive, thereby providing the services required by the coprocessor. Table 4 summarizes the coprocessor primitives accepted by the MC68030.

Table 4. Coprocessor Primitives

Primitive	Function
Processor Synchronization	Busy with Current Instruction Proceed with Next Instruction If No Trace Service Interrupts and Requery If Trace Enabled Proceed with Execution, Condition True/False
Instruction Manipulation	Transfer Operation Word Transfer Words from Instruction Stream
Exception Handling	Take Privilege Violation If S Bit Not Set Take Pre-Instruction Exception Take Mid-Instruction Exception Take Post-Instruction Exception
General Operand Transfer	Evaluate and Pass ⟨ea⟩ Evaluate ⟨ea⟩ and Transfer Data Write to Previously Evaluated ⟨ea⟩ Take Address and Transfer Data Transfer to/from Top of Stack
Register Transfer	Transfer CPU Register Transfer CPU Control Register Transfer Multiple CPU Registers Transfer Multiple Coprocessor Registers Transfer CPU SR and/or ScanPC

SIGNAL DESCRIPTION

Figure 8 illustrates the functional signal groups, and Table 5 describes the signals and their function.

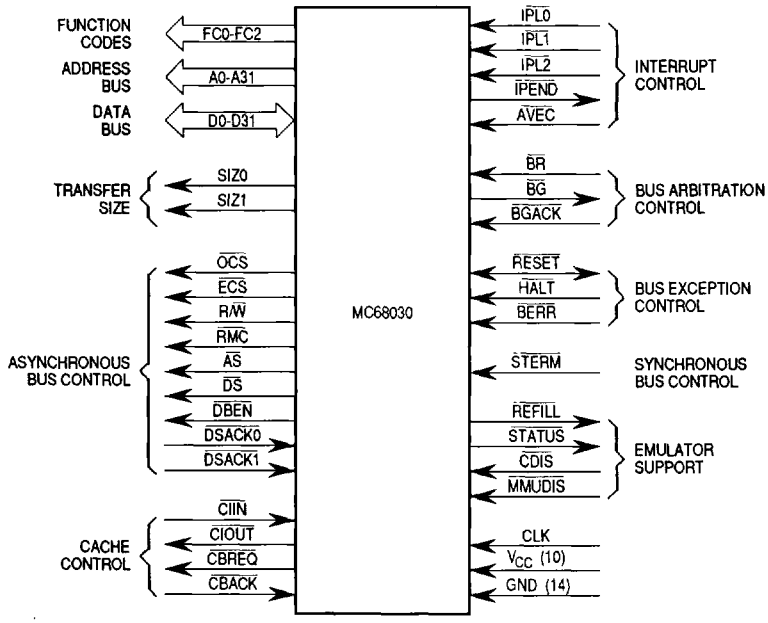


Figure 8. Functional Signal Groups

Table 5. Signal Index

Signal Name	Mnemonic	Function
Function Codes	FC0–FC2	3-bit function code used to identify the address space of each bus cycle.
Address Bus	A0–A31	32-bit address bus.
Data Bus	D0–D31	32-bit data bus used to transfer 8, 16, 24, or 32 bits of data per bus cycle.
Size	SIZ0/SIZ1	Indicates the number of bytes remaining to be transferred for this cycle. These signals, together with A0 and A1, define the active sections of the data bus.
Operand Cycle Start	$\overline{\text{OCS}}$	Identical operation to that of $\overline{\text{ECS}}$ except that $\overline{\text{OCS}}$ is asserted only during the first bus cycle of an operand transfer.
External Cycle Start	$\overline{\text{ECS}}$	Provides an indication that a bus cycle is beginning.
Read/Write	R/$\overline{\text{W}}$	Defines the bus transfer as a processor read or write.
Read-Modify-Write Cycle	$\overline{\text{RMC}}$	Provides an indicator that the current bus cycle is part of an indivisible read-modify-write operation.
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the bus.
Data Strobe	$\overline{\text{DS}}$	Indicates that valid data is to be placed on the data bus by an external device or has been placed on the data bus by the MC68030.
Data Buffer Enable	$\overline{\text{DBEN}}$	Provides an enable signal for external data buffers.
Data Transfer and Size Acknowledge	$\overline{\text{DSACK0}}$/ $\overline{\text{DSACK1}}$	Bus response signals that indicate the requested data transfer operation has been completed. In addition, these two lines indicate the size of the external bus port on a cycle-by-cycle basis and are used for asynchronous transfers.
Synchronous Termination	$\overline{\text{STERM}}$	Bus response signal that indicates a port size of 32 bits and that data may be latched on the next falling clock edge.
Cache Inhibit In	$\overline{\text{CIIN}}$	Prevents data from being loaded into the MC68030 instruction and data caches.
Cache Inhibit Out	$\overline{\text{CIOUT}}$	Reflects the CI bit in ATC entries or TTx register; indicates that external caches should ignore these accesses.
Cache Burst Request	$\overline{\text{CBREQ}}$	Indicates a burst request for the instruction or data cache.
Cache Burst Acknowledge	$\overline{\text{CBACK}}$	Indicates that the accessed device can operate in burst mode.
Interrupt Priority Level	$\overline{\text{IPL0}}$–$\overline{\text{IPL2}}$	Provides an encoded interrupt level to the processor.
Interrupt Pending	$\overline{\text{IPEND}}$	Indicates that an interrupt is pending.
Autovector	$\overline{\text{AVEC}}$	Requests an autovector during an interrupt acknowledge cycle.
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership.
Bus Grant	$\overline{\text{BG}}$	Indicates that an external device may assume bus mastership.

Table 5. Signal Index (Continued)

Signal Name	Mnemonic	Function
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership.
Reset	$\overline{\text{RESET}}$	System reset.
Halt	$\overline{\text{HALT}}$	Indicates that the processor should suspend bus activity.
Bus Error	$\overline{\text{BERR}}$	Indicates that an erroneous bus operation is being attempted.
Cache Disable	$\overline{\text{CDIS}}$	Dynamically disables the on-chip cache to assist emulator support.
MMU Disable	$\overline{\text{MMUDIS}}$	Dynamically disables the translation mechanism of the MMU.
Pipe Refill	$\overline{\text{REFILL}}$	Indicates when the MC68030 is beginning to fill pipeline.
Microsequencer Status	$\overline{\text{STATUS}}$	Indicates the state of the microsequencer.
Clock	CLK	Clock input to the processor.
Power Supply	V_{CC}	Power supply.
Ground	GND	Ground connection.

ELECTRICAL SPECIFICATIONS

MAXIMUM RATINGS — MC68030

Rating	Symbol	Value	Unit
Supply Voltage*	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.5 to +7.0	V
Operating Temperature Range			°C
Minimum Ambient Temperature	T_A	0	
Maximum Ambient Temperature	T_A		
PGA ♦ = 33 MHz*, PPGA		70	
Maximum Junction Temperature	T_J		
PGA 40 and 50 MHz*		TBD	
CQFP		TBD	
Storage Temperature Range	T_{stg}	−55 to 150	°C

*Rated clock speed of device (not operating frequency)

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Rating
Thermal Resistance — Junction to Ambient	θ_{JA}		°C/W
PGA Package		30*	
PPGA Package		TBD	
CQFP Package		TBD	
Thermal Resistance — Junction to Case	θ_{JC}		
PGA Package		15*	
PPGA Package		TBD	
CQFP Package		TBD	

*Estimated

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T_A = Ambient Temperature, °C
θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D = $P_{INT} + P_{I/O}$
P_{INT} = $I_{CC} \times V_{CC}$, Watts — Chip Internal Power
$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

The following is an approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected):

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling, and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User-derived values for thermal resistance may differ.

AC ELECTRICAL SPECIFICATIONS DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the MC68030 clock input and, possibly, relative to one or more other signals.

The measurement of the AC specifications is defined by the waveforms in Figure 9. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 9. Outputs of the MC68030 are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs to the MC68030 are specified with minimum and, as appropriate,

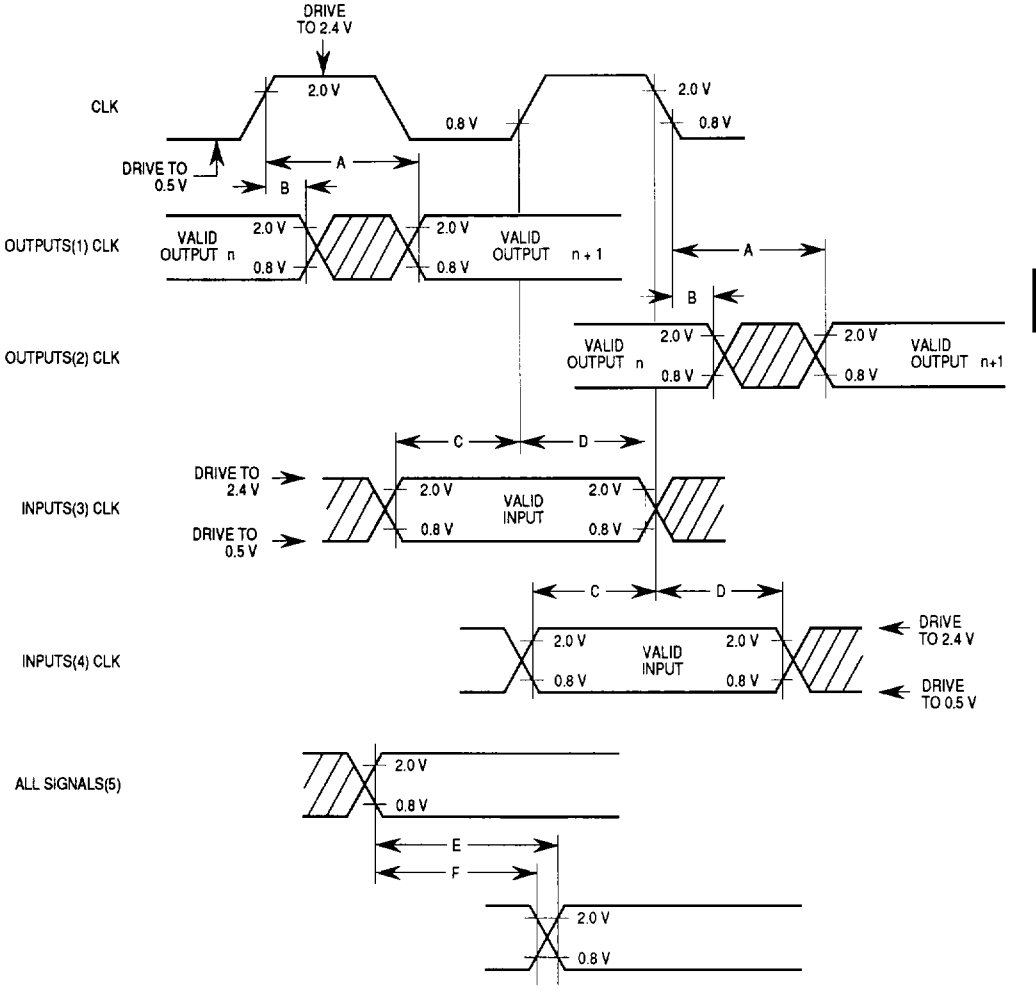
maximum setup and hold times, and are measured as shown. Finally, the measurements for signal-to-signal specifications are also shown.

Note that the testing levels used to verify conformance of the MC68030 to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.

DC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 Vdc ± 5%; GND = 0 Vdc; Temperature in defined ranges)

Characteristic		Symbol	Min	Max	Unit
Input High Voltage		V_{IH}	2.0	V_{CC}	V
Input Low Voltage		V_{IL}	GND −0.5	0.8	V
Input Leakage Current GND ≤ V_{in} ≤ V_{CC}	$\overline{BERR}$, $\overline{BR}$, $\overline{BGACK}$, CLK, $\overline{IPL0}$–$\overline{IPL2}$, $\overline{AVEC}$, $\overline{CDIS}$, $\overline{DSACK0}$, $\overline{DSACK1}$	I_{in}	−2.5	2.5	μA
	$\overline{HALT}$, $\overline{RESET}$		−20	20	
Hi-Z (Off-State) Leakage Current @ 2.4 V/0.5 V	A0–A31, $\overline{AS}$, $\overline{DBEN}$, $\overline{DS}$, D0–D31, FC0–FC2, R/$\overline{W}$, $\overline{RMC}$, SIZ0–SIZ1	I_{TSI}	−20	20	μA
Output High Voltage I_{OH} = 400 μA	A0–A31, $\overline{AS}$, $\overline{BG}$, D0–D31, $\overline{DBEN}$, $\overline{DS}$, $\overline{ECS}$, R/$\overline{W}$, $\overline{IPEND}$, $\overline{OCS}$, $\overline{RMC}$, SIZ0–SIZ1, FC0–FC2, $\overline{CBREQ}$, $\overline{CIOUT}$, $\overline{STATUS}$, $\overline{REFILL}$	V_{OH}	2.4	—	V
Output Low Voltage		V_{OL}			V
I_{OL} = 3.2 mA	A0–A31, FC0–FC2, SIZ0–SIZ1, $\overline{BG}$, D0–D31		—	0.5	
I_{OL} = 5.3 mA	$\overline{CBREQ}$, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, $\overline{RMC}$, $\overline{DBEN}$, $\overline{IPEND}$		—	0.5	
I_{OL} = 2.0 mA	$\overline{STATUS}$, $\overline{REFILL}$, $\overline{CIOUT}$, $\overline{ECS}$, $\overline{OCS}$		—	0.5	
I_{OL} = 10.7 mA	$\overline{RESET}$		—	0.5	
Power Dissipation (T_A = 0°C)		P_D	—	2.6	W
Capacitance (see Note) V_{in} = 0 V, T_A = 25°C, f = 1 MHz		C_{in}	—	20	pF
Load Capacitance	$\overline{ECS}$, $\overline{OCS}$	C_L	—	50	pF
	$\overline{CIOUT}$, $\overline{STATUS}$, $\overline{REFILL}$			70	
	All Other			130	

NOTE: Capacitance is periodically sampled rather than 100% tested.



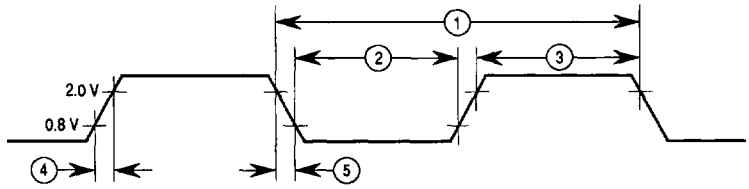
NOTES:
1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This output timing is applicable to all parameters specified relative to the falling edge of the clock.
3. This input timing is applicable to all parameters specified relative to the rising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

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

Figure 9. Drive Levels and Test Points for AC Specifications

AC ELECTRICAL SPECIFICATIONS — CLOCK INPUT (see Figure 10)

Num.	Characteristic	20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
	Frequency of Operation	12.5	20	12.5	25	20	33.33	25	40	25	50	MHz
1	Cycle Time Clock	50	80	40	80	30	50	25	40	20	40	ns
2, 3	Clock Pulse Width Measured from 1.5 V to 1.5 V	23	57	19	61	14	36	11.5	29	9.5	30.5	ns
4, 5	Clock Rise and Fall Times	—	5	—	4	—	3	—	2	—	2	ns



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 10. Clock Input Timing Diagram

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 5.0 Vdc ±5%; GND = 0 Vdc; Temperature in defined ranges)

Num.	Characteristic	20 MHz		25 MHz		33.33 MHz		40 MHz*		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
6	Clock High to Function Code, Size, $\overline{RMC}$, $\overline{IPEND}$, $\overline{CIOUT}$, Address Valid	0	25	0	20	0	14	0	14	0	14	ns
6A	Clock High to $\overline{ECS}$, $\overline{OCS}$ Asserted	0	15	0	15	0	12	0	10	0	10	ns
6B	Function Code, Size, $\overline{RMC}$, $\overline{IPEND}$, $\overline{CIOUT}$, Address Valid to Negating Edge of $\overline{ECS}$	4	—	3	—	3	—	3	—	3	—	ns
7	Clock High to Function Code, Size, $\overline{RMC}$, $\overline{CIOUT}$, Address, Data High Impedance	0	50	0	40	0	30	0	25	0	20	ns
8	Clock High to Function Code, Size, $\overline{RMC}$, $\overline{IPEND}$, $\overline{CIOUT}$, Address Invalid	0	—	0	—	0	—	0	—	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$ Asserted, $\overline{CBREQ}$ Valid	3	20	3	18	2	10	2	10	2	10	ns
9A[1]	$\overline{AS}$ to $\overline{DS}$ Assertion Skew (Read)	−10	10	−10	10	−8	8	−6	6	−6	6	ns
9B[14]	$\overline{AS}$ Asserted to $\overline{DS}$ Asserted (Write)	32	—	27	—	22	—	16	—	14	—	ns
10	$\overline{ECS}$ Width Asserted	15	—	10	—	8	—	5	—	4	—	ns
10A	$\overline{OCS}$ Width Asserted	15	—	10	—	8	—	5	—	4	—	ns
10B[7]	$\overline{ECS}$, $\overline{OCS}$ Width Negated	10	—	5	—	5	—	5	—	4	—	ns
11	Function Code, Size, $\overline{RMC}$, $\overline{CIOUT}$, Address Valid to Asserting Edge of $\overline{AS}$ Asserted (and $\overline{DS}$ Asserted, Read)	10	—	7	—	5	—	5	—	3	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CBREQ}$ Negated	0	20	0	18	0	10	0	10	0	10	ns
12A	Clock Low to $\overline{ECS}$/$\overline{OCS}$ Negated	0	20	0	18	0	15	0	12	0	11	ns
13	$\overline{AS}$, $\overline{DS}$ Negated to Function Code, Size, $\overline{RMC}$, $\overline{CIOUT}$, Address Invalid	10	—	7	—	5	—	3	—	3	—	ns
14	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted (Asynchronous Cycle)	85	—	70	—	45	—	30	—	25	—	ns
14A[11]	$\overline{DS}$ Width Asserted (Write)	38	—	30	—	23	—	18	—	13	—	ns
14B	$\overline{AS}$ (and $\overline{DS}$, Read) Width Asserted (Synchronous Cycle)	35	—	30	—	23	—	18	—	13	—	ns
15	$\overline{AS}$, $\overline{DS}$ Width Negated	38	—	30	—	23	—	18	—	13	—	ns
15A[8]	$\overline{DS}$ Negated to $\overline{AS}$ Asserted	30	—	25	—	18	—	16	—	14	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, $\overline{DBEN}$, $\overline{CBREQ}$ High Impedance	—	50	—	40	—	30	—	25	—	20	ns
17	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	10	—	7	—	5	—	3	—	3	—	ns
18	Clock High to R/$\overline{W}$ High	0	25	0	20	0	15	0	14	0	14	ns
20	Clock High to R/$\overline{W}$ Low	0	25	0	20	0	15	0	14	0	14	ns
21	R/$\overline{W}$ High to $\overline{AS}$ Asserted	10	—	7	—	5	—	5	—	3	—	ns

AC ELECTRICAL SPECIFICATIONS (Continued)

Num.	Characteristic	20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
22	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	60	—	47	—	35	—	24	—	23	—	ns
23	Clock High to Data-Out Valid	—	25	—	20	—	14	—	14	—	14	ns
24	Data-Out Valid to Negating Edge of $\overline{AS}$	8	—	5	—	3	—	3	—	3	—	ns
25[11]	$\overline{AS}$, $\overline{DS}$ Negated to Data-Out Invalid	10	—	7	—	5	—	3	—	3	—	ns
25A[9,11]	$\overline{DS}$ Negated to $\overline{DBEN}$ Negated (Write)	10	—	7	—	5	—	3	—	3	—	ns
26[11]	Data-Out Valid to Asserting Edge of $\overline{DS}$ Asserted (Write)	10	—	7	—	5	—	3	—	3	—	ns
27	Data-In Valid to Clock Low (Setup)	4	—	2	—	1	—	1	—	1	—	ns
27A	Late $\overline{BERR}$/$\overline{HALT}$ Asserted to Clock Low (Setup)	10	—	5	—	3	—	3	—	3	—	ns
28[12]	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$x, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated (Asynchronous Hold)	0	50	0	40	0	30	0	20	0	15	ns
28A[12]	Clock Low to $\overline{DSACK}$x, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated (Synchronous Hold)	12	85	8	70	6	50	6	40	6	35	ns
29[12]	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Asynchronous Hold)	0	—	0	—	0	—	0	—	0	—	ns
29A[12]	$\overline{AS}$, $\overline{DS}$ Negated to Data-In High Impedance	—	50	—	40	—	30	—	25	—	20	ns
30[12]	Clock Low to Data-In Invalid (Synchronous Hold)	12	—	8	—	6	—	6	—	6	—	ns
30A[12]	Clock Low to Data-In High Impedance (Read followed by Write)	—	75	—	60	—	45	—	30	—	25	ns
31[2]	$\overline{DSACK}$x Asserted to Data-In Valid (Asynchronous Data Setup)	—	43	—	28	—	20	—	14	—	13	ns
31A[3]	$\overline{DSACK}$x Asserted to $\overline{DSACK}$x Valid (Skew)	—	10	—	7	—	5	—	3	—	3	ns
32	$\overline{RESET}$ Input Transition Time	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	Clks
33	Clock Low to $\overline{BG}$ Asserted	0	25	0	20	0	15	0	14	0	14	ns
34	Clock Low to $\overline{BG}$ Negated	0	25	0	20	0	15	0	14	0	14	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37A[6]	$\overline{BGACK}$ Asserted to $\overline{BR}$ Negated	0	1.5	0	1.5	0	1.5	0	1.5	0	1.5	Clks
39	$\overline{BG}$ Width Negated	75	—	60	—	45	—	30	—	30	—	ns
39A	$\overline{BG}$ Width Asserted	75	—	60	—	45	—	30	—	30	—	ns
40	Clock High to $\overline{DBEN}$ Asserted (Read)	0	25	0	20	0	18	0	16	0	14	ns

AC ELECTRICAL SPECIFICATIONS (Continued)

Num.	Characteristic	20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz*		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
41	Clock Low to $\overline{DBEN}$ Negated (Read)	0	25	0	20	0	18	0	16	0	14	ns
42	Clock Low to $\overline{DBEN}$ Asserted (Write)	0	25	0	20	0	18	0	16	0	14	ns
43	Clock High to $\overline{DBEN}$ Negated (Write)	0	25	0	20	0	18	0	16	0	14	ns
44	R/$\overline{W}$ Low to $\overline{DBEN}$ Asserted (Write)	10	—	7	—	5	—	5	—	5	—	ns
45[5]	$\overline{DBEN}$ Width Asserted Asynchronous Read Asynchronous Write	 50 100	 — —	 40 80	 — —	 30 60	 — —	 22 45	 — —	 20 40	 — —	ns
45A[9]	$\overline{DBEN}$ Width Asserted Synchronous Read Synchronous Write	 10 50	 — —	 5 40	 — —	 5 30	 — —	 5 22	 — —	 5 20	 — —	ns
46	R/$\overline{W}$ Width Asserted (Asynchronous Write or Read)	125	—	100	—	75	—	50	—	40	—	ns
46A	R/$\overline{W}$ Width Asserted (Synchronous Write or Read)	75	—	60	—	45	—	30	—	25	—	ns
47A	Asynchronous Input Setup Time to Clock Low	4	—	2	—	2	—	2	—	2	—	ns
47B	Asynchronous Input Hold Time from Clock Low	12	—	8	—	6	—	6	—	6	—	ns
48[4]	$\overline{DSACKx}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	—	20	—	25	—	18	—	14	—	13	ns
53	Data-Out Hold from Clock High	3	—	3	—	2	—	2	—	2	—	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	25	—	20	—	15	—	11	—	11	—	ns
56	$\overline{RESET}$ Pulse Width (Reset Instruction)	512	—	512	—	512	—	512	—	512	—	Clks
57	$\overline{BERR}$ Negated to $\overline{HALT}$ Negated (Rerun)	0	—	0	—	0	—	0	—	0	—	ns
58[10]	$\overline{BGACK}$ Negated to Bus Driven	1	—	1	—	1	—	1	—	1	—	Clks
59[10]	$\overline{BG}$ Negated to Bus Driven	1	—	1	—	1	—	1	—	1	—	Clks
60[13]	Synchronous Input Valid to Clock High (Setup Time)	4	—	2	—	2	—	2	—	2	—	ns
61[13]	Clock High to Synchronous Input Invalid (Hold Time)	12	—	8	—	6	—	6	—	6	—	ns
62	Clock Low to $\overline{STATUS}$, $\overline{REFILL}$ Asserted	0	25	0	20	0	15	0	15	0	15	ns
63	Clock Low to $\overline{STATUS}$, $\overline{REFILL}$ Negated	0	25	0	20	0	15	0	15	0	15	ns

AC ELECTRICAL SPECIFICATIONS (Concluded)

NOTES:
Temperature must be in range described in the Maximum Ratings.

1. This number can be reduced to 5 ns if strobes have equal loads.
2. If the asynchronous setup time (#47A) requirements are satisfied, the DSACKx low to data setup time (#31) and DSACKx low to BERR low setup time (#48) can be ignored. The data must only satisfy the data-in clock low setup time (#27) for the following clock cycle, and BERR must only satisfy the late BERR low to clock low setup time (#27A) for the following clock cycle.
3. This parameter specifies the maximum allowable skew between DSACK0 to DSACK1 asserted or DSACK1 to DSACK0 asserted; specification #47A must be met by DSACK0 or DSACK1.
4. This specification applies to the first (DSACK0 or DSACK1) DSACKx signal asserted. In the absence of DSACKx, BERR is an asynchronous input using the asynchronous input setup time (#47A).
5. DBEN may stay asserted on consecutive write cycles.
6. The minimum values must be met to guarantee proper operation. If this maximum value is exceeded, BG may be reasserted.
7. This specification indicates the minimum high time for ECS and OCS in the event of an internal cache hit followed immediately by another cache hit, a cache miss, or an operand cycle.
8. This specification guarantees operation with the MC68881/MC68882, which specifies a minimum time for DS negated to AS asserted (specification #13A in the *MC68881/MC68882 User's Manual*). Without this specification, incorrect interpretation of specifications #9A and #15 would indicate that the MC68030 does not meet the MC68881/MC68882 requirements.
9. This specification allows a system designer to guarantee data hold times on the output side of data buffers that have output enable signals generated with DBEN. The timing on DBEN precludes its use for synchronous READ cycles with no wait states.
10. These specifications allow system designers to guarantee that an alternate bus master has stopped driving the bus when the MC68030 regains control of the bus after an arbitration sequence.
11. DS will not be asserted for synchronous write cycles with no wait states.
12. These hold times are specified with respect to strobes (asynchronous) and with respect to the clock (synchronous). The designer is free to use either time.
13. Synchronous inputs must meet specifications #60 and #61 with stable logic levels for ***all*** rising edges of the clock while AS is asserted. These values are specified relative to the high level of the rising clock edge. The values originally published were specified relative to the low level of the rising clock edge.
14. This specification allows system designers to qualify the CS signal of an MC68881/MC68882 with AS (allowing 7 ns for a gate delay) and still meet the CS to DS setup time requirement (specification 8B of the *MC68881/MC68882 User's Manual*).

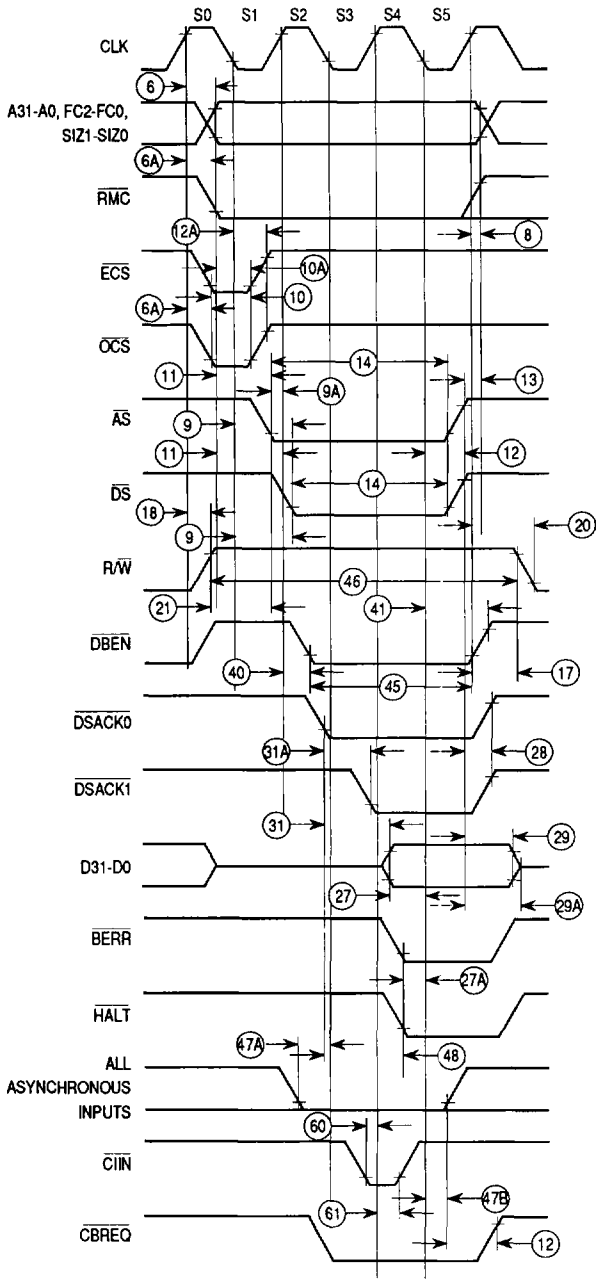


Figure 11. Asynchronous Read Cycle Timing Diagram

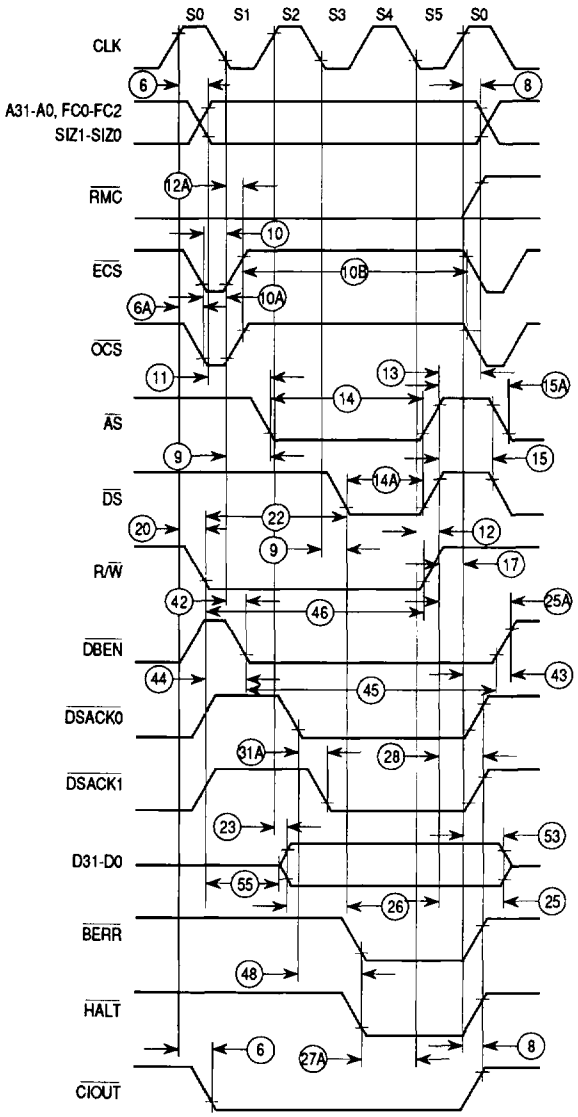


Figure 12. Asynchronous Write Cycle Timing Diagram

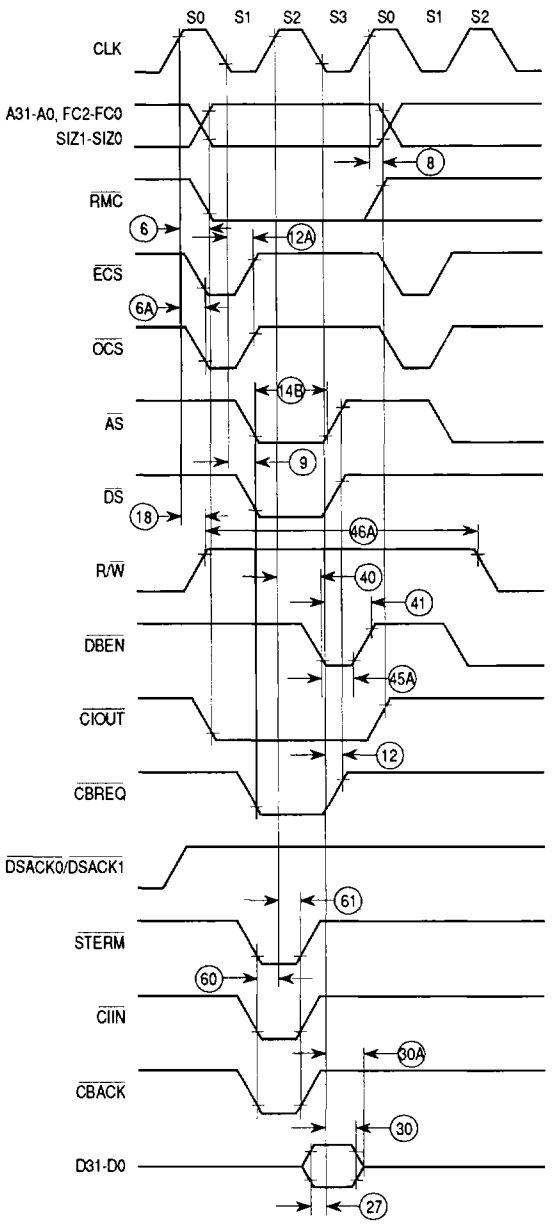


Figure 13. Synchronous Read Cycle Timing Diagram

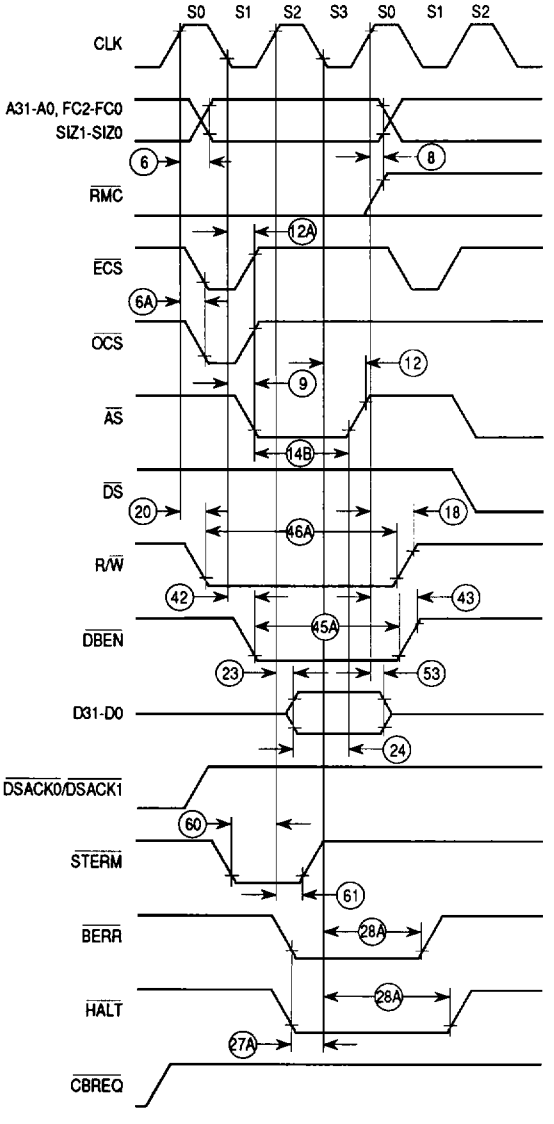
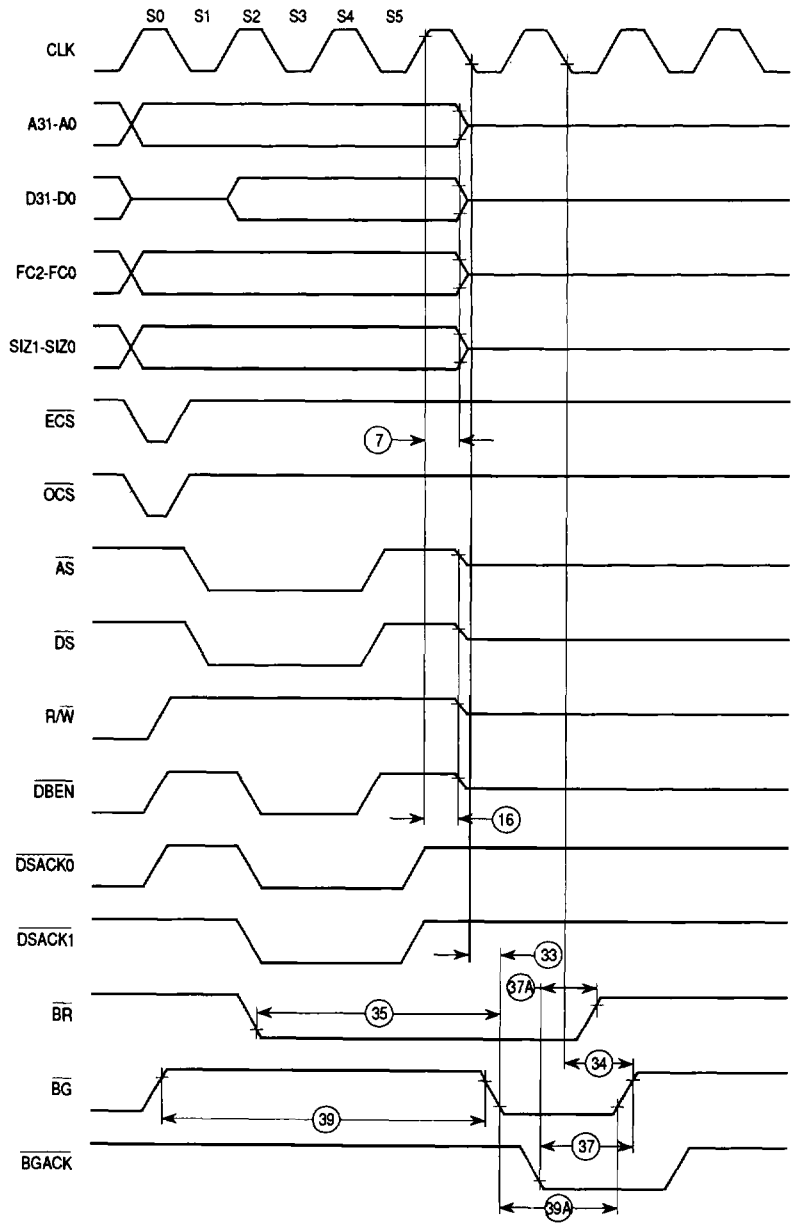


Figure 14. Synchronous Write Cycle Timing Diagram



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 15. Bus Arbitration Timing Diagram

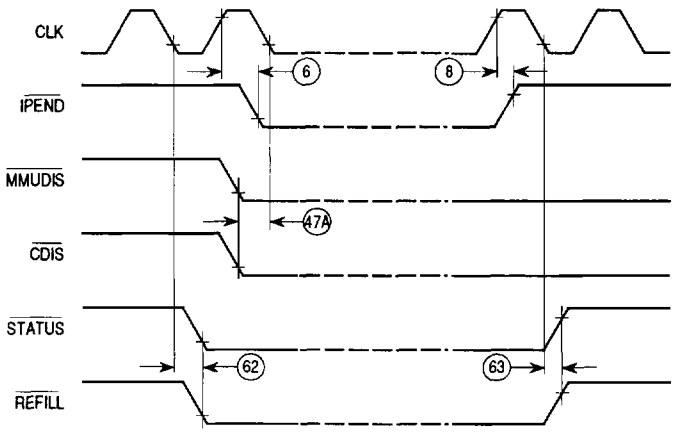
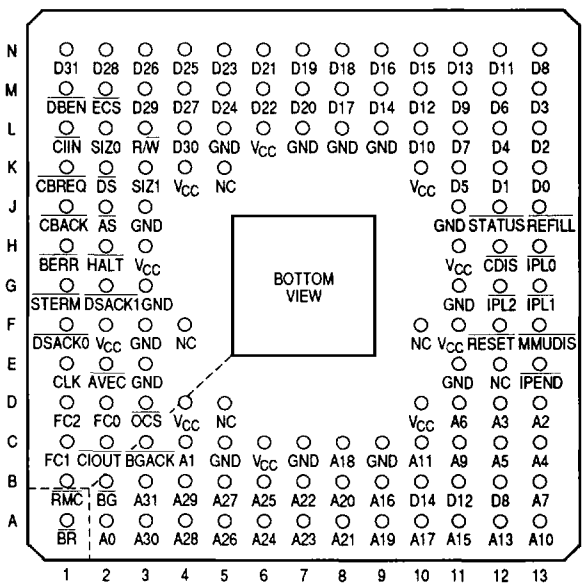


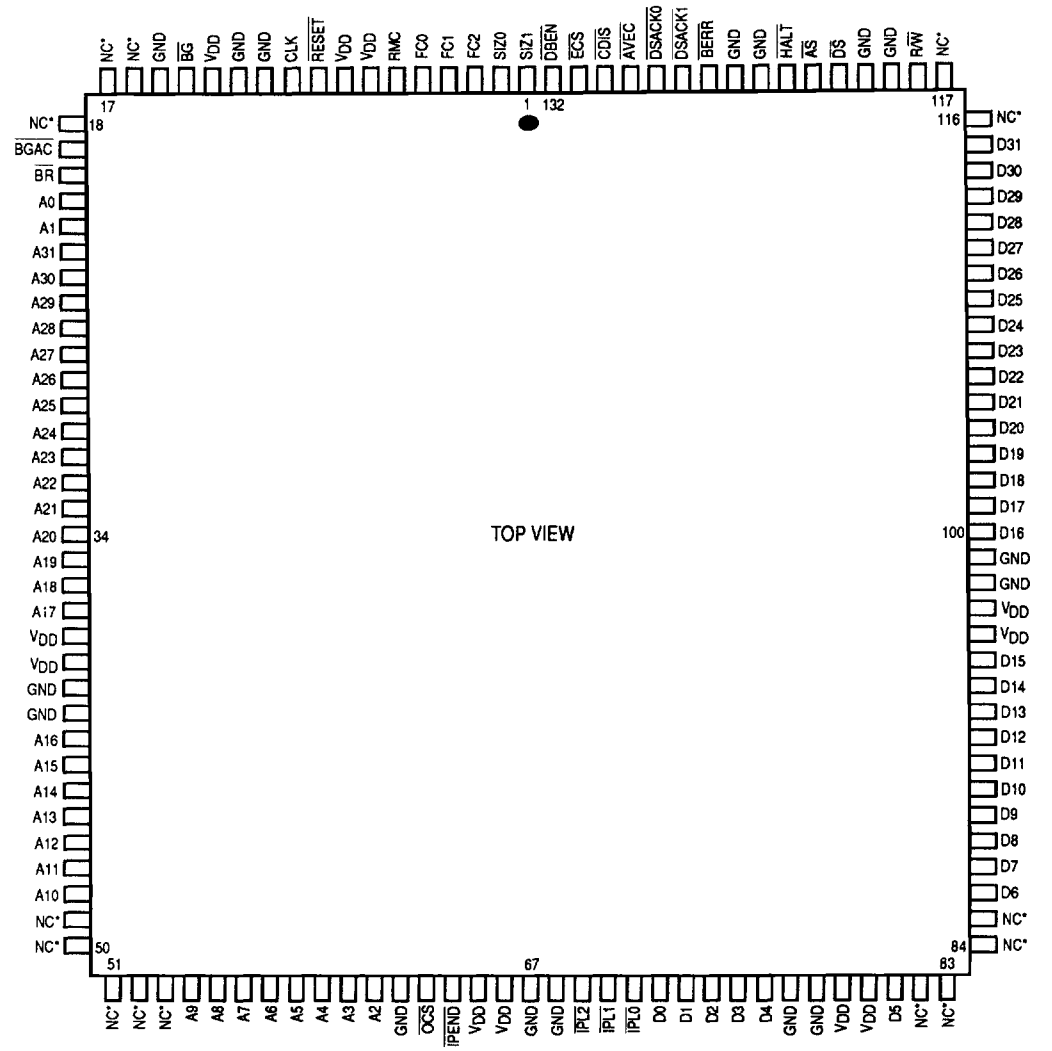
Figure 16. Other Signal Timings

PIN ASSIGNMENTS

128-LEAD PIN GRID ARRAY (RC AND RP SUFFIX)



132-LEAD CERAMIC SURFACE MOUNT (FE SUFFIX)



*NC - Do not connect to this pin.