Rochester Electronics

Datasheet

Rochester Electronics Manufactured Components

Rochester branded components are manufactured using either die/wafers purchased from the original suppliers or Rochester wafers recreated from the original IP. All recreations are done with the approval of the OCM.

Parts are tested using original factory test programs or Rochester developed test solutions to guarantee product meets or exceed the OCM data sheet.

Quality Overview

- ISO-9001
- AS9120 certification
- Qualified Manufacturers List (QML) MIL-PRF-35835
    - Class Q Military
    - Class V Space Level
- Qualified Suppliers List of Distributors (QSLD)

    - Rochester is a critical supplier to DLA and meets all industry and DLA standards.

Rochester Electronics, LLC is committed to supplying products that satisfy customer expectations for quality and are equal to those originally supplied by industry manufacturers.

The original manufacturer's datasheet accompanying this document reflects the performance and specifications of the Rochester manufactured version of this device. Rochester Electronics guarantees the performance of its semiconductor products to the original OEM specifications. 'Typical' values are for reference purposes only. Certain minimum or maximum ratings may be based on product characterization, design, simulation, or sample testing.

Order this document
by MC68020/D

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC68020

Technical Summary

32-Bit Virtual Memory Microprocessor

The MC68020 is the first full 32-bit implementation of the M68000 Family of microprocessors from Motorola. Using Motorola's advanced HCMOS technology, the MC68020 is implemented with 32-bit registers and data paths, 32-bit addresses, a rich instruction set, and versatile addressing modes.

This document contains information on a new product. Specifications and information herein are subject to change without notice.

MOTOROLA

Rev. 4
(Replaces BR243/D Rev. 3)

The main features of the MC68020 are as follows:

- Object-Code Compatible with Earlier M68000 Microprocessors
- Addressing Mode Extensions for Enhanced Support of High-Level Languages
- New Bit Field Data Type Accelerates Bit-Oriented Applications — i.e., Video Graphics
- Fast On-Chip Instruction Cache Speeds Instructions and Improves Bus Bandwidth
- Coprocessor Interface to Companion 32-Bit Peripherals — the MC68881 and MC68882 Floating-Point Coprocessors and the MC68851 Paged Memory Management Unit
- Pipelined Architecture with High Degree of Internal Parallelism Allowing Multiple Instructions To Be Executed Concurrently
- High-Performance Asynchronous Bus Is Nonmultiplexed and Full 32-Bits
- Dynamic Bus Sizing Efficiently Supports 8-/16-/32-Bit Memories and Peripherals
- Full Support of Virtual Memory and Virtual Machine
- 16 32-Bit General-Purpose Data and Address Registers
- Two 32-Bit Supervisor Stack Pointers and Five Special-Purpose Control Registers
- 18 Addressing Modes and Seven Data Types
- 4-Gbyte Direct Addressing Range
- Selection of Processor Speeds: 16.67, 20, 25, and 33.33 MHz

INTRODUCTION

The MC68020 is a high-performance 32-bit microprocessor. It is the first microprocessor to evolve from a 16-bit machine to a full 32-bit machine that provides 32-bit address and data buses as well as 32-bit internal structures. Many techniques were utilized to improve performance and maintain compatibility with other processors of the M68000 Family. Among the improvements are new addressing modes that better support high-level language structures, an expanded instruction set providing 32-bit operations for the limited cases not supported by the MC68000 and the MC68010, and several new instructions to support new data types. For special-purpose applications, when a general-purpose processor is not adequate, a coprocessor interface is provided.

The MC68020 is a high-performance microprocessor implemented in HCMOS, which allows CMOS and HMOS (high-density NMOS) gates to be combined on the same device. CMOS structures are used when speed and low power are required, and HMOS structures are used when minimum silicon area is desired. This technology enables the MC68020 to be very fast, consume less power (less than 1.5 W), and have a reasonably small die size.

Figure 1 is a block diagram of the MC68020. The processor is divided into two main sections: the bus controller and the micromachine. This division reflects the autonomy with which the sections operate.

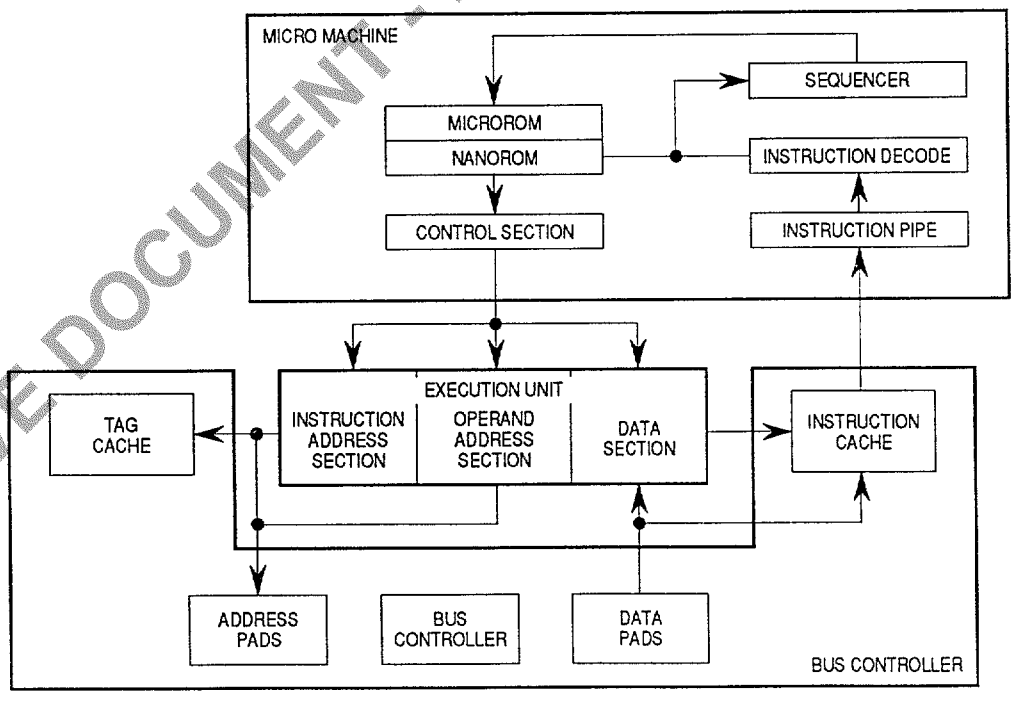


Figure 1. Block Diagram

The bus controller consists of the address and data pads and multiplexers required to support dynamic bus sizing, a macro bus controller to schedule the bus cycles on the basis of priority with two state machines (one to control the bus cycles for operand accesses and the other to control the bus cycles for instruction accesses), and the instruction cache with its associated control.

The micromachine consists of an execution unit, nanoROM and microROM storage, an instruction decoder, an instruction pipe, and associated control sections. The execution unit consists of an address section, an operand address section, and a data section. Microcode control is provided by a modified two-level store of microROM and nanoROM. Programmed logical 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 decode and interpret nanoROM and microROM information.

PROGRAMMING MODEL

As shown in the programming models (see Figures 2 and 3), the MC68020 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, and two 32-bit cache handling (address and control) registers. Registers D0–D7 are used as data registers for bit and bit field (1 to 32 bits), byte (8 bit), word (16 bit), long-word (32 bit), and quad-word (64 bit) operations. Registers A0–A6 and the user, interrupt, and master stack pointers are address registers that can be used as software stack pointers or base address registers. In addition, the address registers can be used for word and long-word operations. All 16 registers (D0–D7, A0–A7) can be used as index registers.

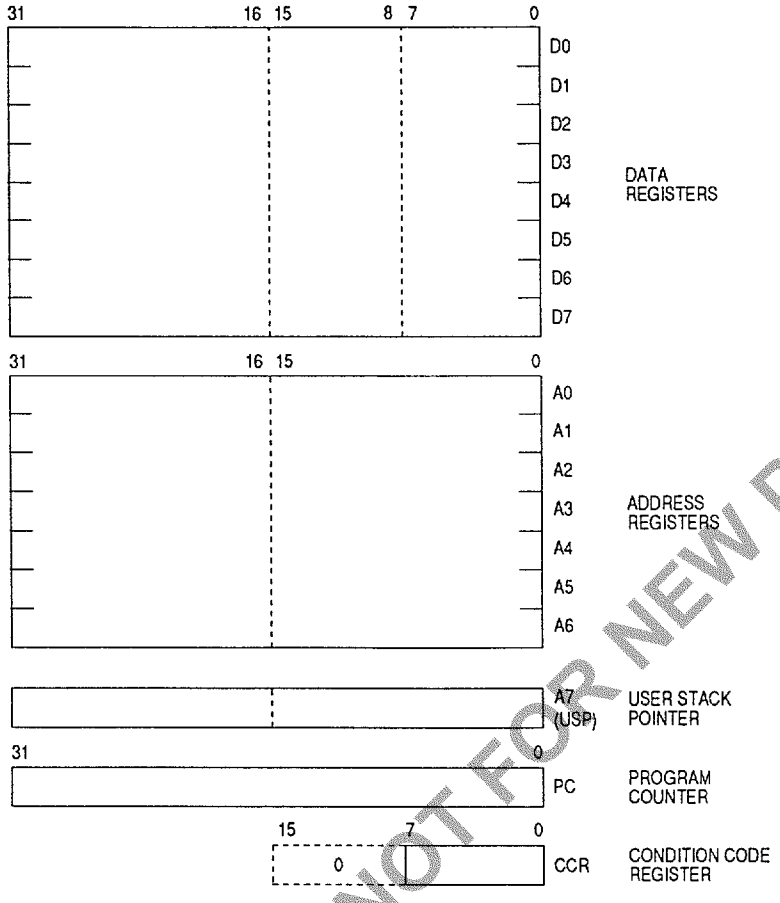


Figure 2. User Programming Model

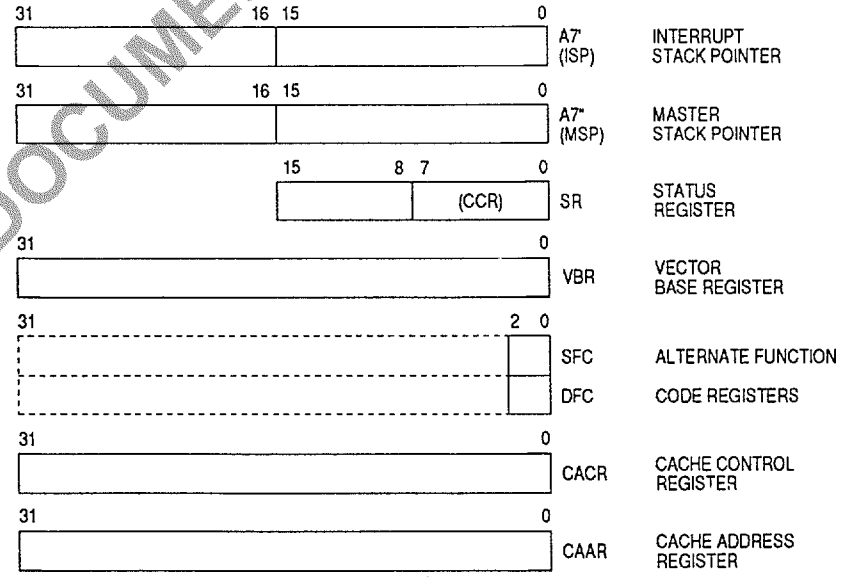


Figure 3. Supervisor Programming Model Supplement

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), or master/interrupt state (M).

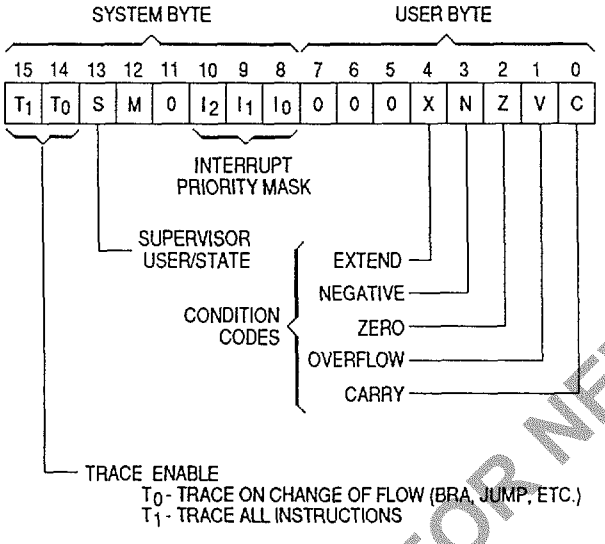


Figure 4. Status Register

All microprocessors of the M68000 Family support instruction tracing (via the T0 status bit in the MC68020) where each executed instruction is followed by a trap to a user-defined trace routine. The MC68020 adds the capability to trace only 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 and program/data. These four combinations are specified by the function code pins, FC0/FC1/FC2, during access of the particular address space. Using the function codes, the memory subsystem can distinguish between authorized access (supervisor mode is privileged access) and unauthorized access (user mode may not have access to supervisor program or data areas). To support the full privileges of the supervisor, the alternate function code registers allow the supervisor to specify an access to user program or data areas by appropriately preloading the SFC/DFC registers.

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

DATA TYPES AND ADDRESSING MODES

Seven basic data types are supported by the MC68020:

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 MC68881 floating-point coprocessor allows direct support of floating-point data types 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. Programmers find these capabilities 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; programmers find this addressing mode is required to support position-independent software. In addition to these addressing modes, the MC68020 provides data operand sizing and scaling; these features provide performance enhancements to 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.

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 MC68020 instruction set is listed in Table 2. Special emphasis has been given to the instruction set's support of structured high-level languages and sophisticated operating systems. Each instruction, with few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 18 addressing modes. Many instruction extensions have been made on the MC68020 to take advantage of the full 32-bit operation; whereas, on the earlier M68000 Family members, only 8- and 16-bit values were used. The MC68020 is upward source- and object-code compatible with the M68000 Family because it supports all instructions of previous family members. Additional instructions are provided by the MC68020 to support its advanced features. For detailed information on the MC68020 instruction set, refer to M68000 PM/AD, *M68000 Programmer's Reference Manual.*

Table 2. Instruction Set Summary

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
CALLM	Call Module
CAS	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
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 Multiple

Mnemonic	Description
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
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
RTM	Return from Module
RTR	Return and Restore Codes
RTS	Return from Subroutine

Mnemonic	Description
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

Mnemonic	Description
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

BIT FIELD OPERATIONS

The MC68020 supports variable-length bit field operations up to 32 bits. A bit field may start in any bit position and span any address boundary for the full length of the bit field, up to the 32-bit maximum. The bit field insert (BFINS) inserts a value into a field. Bit field extract unsigned (BFEXTU) and bit field extract signed (BFEXTS) extract an unsigned or signed value from the field. BFFFO finds the first bit in a bit field that is set. To complement the M68000 bit manipulation instruction, there are bit field change, clear, set, and test instructions (BFCHG, BFCLR, BFSET, BFTST). Using the on-chip barrel shifter, the bit and bit field instructions are very fast and particularly useful in applications using packed bits and bit fields found in graphics and communications.

BINARY-CODED DECIMAL (BCD) SUPPORT

The M68000 Family supports BCD operations including add, subtract, and negation. The MC68020 adds the PACK and UNPACK operations for BCD conversions to and from binary form as well as other conversions — e.g., ASCII and EBCDIC. The PACK instruction reduces two bytes of data into a single byte; UNPACK reverses the operation.

BOUNDS CHECKING

Previous M68000 Family members offer variable bounds checking only on the upper limit of the bound. The underlying assumption is that the lower bound is zero. Bounds checking is expanded on the MC68020 by providing two new instructions, CHK2 and CMP2. These instructions allow checking and comparing of both the upper and lower bounds. These instructions may be either signed or unsigned. The CMP2 instruction sets the condition codes upon completion; the CHK2 instruction, in addition to setting the condition codes, takes a system trap if either boundary condition is exceeded.

SYSTEM TRAPS

Three additions have been made to the system trap capabilities of the MC68020. The current trap on overflow (TRAPV) instruction has been expanded to a TRAPcc format where any condition code is allowed to be the trapping condition. The TRAPcc instruction is expanded to optionally provide one or two additional words following the trap instruction so user-specified information may be presented to the trap handler. These additional words can be used when needed to provide simple error codes or debug information for interactive run-time debugging or post-mortem program dumps. Compilers can provide direction to run-time execution routines for handling specific conditions.

The breakpoint instruction (BKPT) is used to support the program breakpoint function for debug monitors and real-time in-circuit or hardware emulators, and the operation will be dependent on the actual system implementation. Execution of this instruction causes the MC68020 to run a breakpoint acknowledge bus cycle with a 3-bit breakpoint identifier placed on address lines A2–A4. This 3-bit identifier permits up to eight breakpoints to be easily differentiated. The normal response to the MC68020 is an operation word (typically an instruction originally replaced by the debugger with the breakpoint instruction) placed on the data lines by external debugger hardware, and the breakpoint acknowledge cycle is properly terminated. The MC68020 then executes this operation word in place of the breakpoint instruction. The debugger hardware can count the number of executions of each breakpoint and halt execution after a predetermined number of cycles.

MULTIPROCESSING

To further support multiprocessing with the MC68020, a compare and swap instruction (CAS) has been added. This instruction uses the read-modify-write cycle to compare two operands and to swap a third operand, pending the results of the compare. A variant of this instruction, CAS2, performs similarly, comparing dual operand pairs and updating two operands.

These multiprocessing operations are useful when using common memory to share or pass data between multiple processing elements. The read-modify-write cycle is an indivisible operand that allows reading and updating a "lock" operand used to control access to the common memory elements. The CAS2 instruction is more powerful since dual operands allow the "lock" to be checked and two values (i.e., both pointers in a doubly linked list) to be updated, according to the lock's status, in a single operation.

MODULE SUPPORT

The MC68020 includes support for modules with the call module (CALLM) and return from module (RTM) instructions. The CALLM instruction references a module descriptor. This descriptor contains control information for entry into the associated module. The CALLM instruction creates a module stack frame and stores the module state in that frame. The RTM instruction recovers the previous module state from the stack frame and returns to the calling module.

The module interface also provides a mechanism for finer resolution of access control by external hardware. Although the MC68020 does not interpret the access control information, it does communicate with external hardware when the access control is to be changed and relies on the external hardware to verify that the changes are legal.

When used as subroutine calls and returns with proper descriptor formats, CALLM and RTM cause the MC68020 to verify legitimate access to modules.

VIRTUAL MEMORY/MACHINE CONCEPTS

The full addressing range of the MC68020 is 4 Gbytes (4,294,967,296); however, most MC68020 systems implement a smaller physical memory. By using virtual memory techniques, the system can be made to appear to have a full 4 Gbytes of physical memory available to each user program. These techniques have been used for many years in large mainframe computers and minicomputers. With the MC68020 (as with the MC68010), virtual memory can be fully supported in microprocessor-based systems.

In a virtual memory system, a user program can be written as though it has a large amount of memory available to it when only a smaller amount of memory is physically present in the system. In a similar fashion, a system provides user-program access to other devices not physically present in the system, such as tape drives, disk drives, printers, or terminals. With proper software emulation, a physical system can be made to appear to a user program as any other M68000 computer system, and the program can be given full access to all of the resources of that emulated system. Such an emulated system is called a virtual machine.

VIRTUAL MEMORY

The basic mechanism for supporting virtual memory is to provide 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 (referred to as a page fault), 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.

The MC68020 uses instruction continuation to support virtual memory. To use instruction continuation, the MC68020 stores its internal state on the supervisor stack when a bus cycle is terminated with a bus error signal. It then loads the program counter with the address of the virtual memory bus error handler from the exception vector table (entry number two) and resumes program execution at that new address. When the bus error exception handler routine has completed execution, an RTE instruction is executed, which reloads the MC68020 with the internal state stored on the stack, reruns the faulted bus cycle (when required), and continues the suspended instruction.

Instruction continuation is crucial to the support of virtual I/O devices in memory-mapped I/O systems. Since the registers of a virtual device may be simulated in the memory map, an access to such a register will cause a fault, and the function of the register can be emulated by software.

VIRTUAL MACHINE

A typical use for a virtual machine system is the development of software, such as an operating system, for a machine under development and not yet available for programming use. In such a system, a governing operating system emulates

the hardware of the prototype system and allows the new operating system to be executed and debugged as though it were running on the new hardware. Since the new operating system is controlled by the governing operating system, it is executed at a lower privilege level than the governing operating system. Thus, any attempts by the new operating system to use virtual resources that are not physically present (and should be emulated) are trapped to the governing operating system and handled by its software. In the MC68020, a virtual machine is fully supported by running the new operating system in the user mode. The governing operating system executes in the supervisor mode, and any attempt by the new operating system to access supervisor resources or execute privileged instructions will cause a trap to the governing operating system.

OPERAND TRANSFER MECHANISM

Though the MC68020 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 MC68020 will transfer only the amount of data that the peripheral can manage. This feature allows the peripheral to define its port size as 8, 16, or 32 bits wide, and the MC68020 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 flexibility to choose implementations independent of software implementations.

Dynamic bus sizing is accomplished through the use of the data transfer and size acknowledge ($\overline{\text{DSACK}}$) signals and occurs on a cycle-by-cycle basis. For example, if the processor is executing an instruction that requires reading 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 MC68020 latches all 32 bits of data and continues. If the port responds that it is 16 bits wide, the MC68020 latches the 16 valid data bits of data and runs another cycle to obtain the other 16 bits of data. An 8-bit port is handled similarly but four bus read cycles are required. Each port is fixed in assignment to particular sections of the data bus.

Justification of data on the bus is handled automatically by dynamic bus sizing. When reading 16-bit data from a 32-bit port, the data may appear on the top or bottom half of the bus, depending on the address of the data. The MC68020

determines which portion of the bus is needed to support the transfer and dynamically adjusts to read or write data on those data lines.

The MC68020 always transfers the maximum amount of data on all bus cycles — i.e., it always assumes the port is 32 bits wide when beginning the bus cycle. In addition, the MC68020 has no restrictions concerning alignment of operands in memory; long-word operands need not be aligned on long-word address boundaries. When misaligned data requires multiple bus cycles, the MC68020 automatically runs the minimum bus cycles.

COPROCESSOR INTERFACE

The coprocessor interface is a mechanism for extending the instruction set of the M68000 Family. 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.

The programmer's model for the M68000 Family of microprocessors is based on sequential, nonconcurrent instruction execution — i.e., each instruction is completely executed prior to the beginning of the next instruction. Hence, instructions do not operate concurrently in the programmer's model. Most microprocessors implement the sequential model, which greatly simplifies the programmer responsibilities since sequencing control is automatic and discrete.

The M68000 coprocessor interface is designed to extend the programmer's model and provide full support for the sequential, nonconcurrent instruction execution model. Hence, instruction execution by the coprocessor is assumed not to overlap with instruction execution of the main processor. The M68000 coprocessor interface does allow concurrent operation when concurrency can be properly accommodated. For example, a Motorola floating-point coprocessor allows the MC68020 to proceed executing instructions while the coprocessor continues a floating-point operation, up to the point that the MC68020 sends another request to the coprocessor. Adhering to the sequential execution model, the coprocessor completes each instruction before it starts the next, and the MC68020 is allowed to proceed in concurrent fashion.

Coprocessors are divided into two types by their bus-utilization characteristics. A direct memory access (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 type 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 (see Table 3), which are defined for the M68000 Family coprocessor interface. The MC68020 hardware uses standard M68000 asynchronous bus cycles to access the registers. Thus, the coprocessor does not require special bus hardware; the bus interface implemented by a coprocessor for its interface register set must only satisfy the MC68020 address, data, and control signal timing to guarantee proper communication with the main processor. Because the MC68020 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 MC68020 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/$\overline{W}$

Other microprocessors in the M68000 Family can operate any M68000 coprocessor even though they may lack the hardware implementation of the coprocessor interface. Since the coprocessor is operated through the coprocessor interface registers, which are accessed via normal asynchronous bus cycles, the coprocessor can be used as a peripheral device. Software easily emulates the communication protocol by appropriately addressing the coprocessor interface registers and by passing the required coprocessor commands and operands.

The coprocessor interface registers are implemented by the coprocessor in addition to those registers implemented as extensions to the MC68020 programmer's model. For example, the MC68881/MC68882 implement the coprocessor interface registers shown in Table 3 and the registers in the programming model, including eight 80-bit floating-point data registers and three 32-bit control/status registers used by the MC68881/MC68882 programmer.

Up to eight coprocessors are supported in a single system with a system-unique coprocessor identifier encoded in the coprocessor instruction. When accessing a coprocessor, the MC68020 executes standard read and write 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 to access the coprocessor, 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 M68000 bus cycles.

COPROCESSOR PROTOCOL

Interprocessor transfers are all initiated by the main processor during coprocessor instruction execution. During the processing of 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 the M68000 bus cycles.

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

A. The main processor initiates the communication by writing control information to a location in the coprocessor interface.
B. The main processor reads the coprocessor response to that information.
    1. The response may indicate the coprocessor is busy and that 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 query the coprocessor again 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.

PRIMITIVES/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 for performing the coprocessor command. Table 4 summarizes the coprocessor primitives accepted by the MC68020.

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

EXCEPTIONS

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

EXCEPTION TYPES

Exceptions can be generated by either internal or external causes. The externally generated exceptions are interrupts, bus error, and reset requests. The interrupts are requests from peripheral devices for processor action; whereas, the bus error and reset pins are used for access control and processor restart. The internally generated exceptions come from instructions, address errors, tracing, or breakpoints. The TRAP, TRAPcc, TRAPV, cpTRAPcc, CHK, CHK2, and DIV instructions can all generate exceptions as part of their instruction execution. Tracing behaves like a very high priority, internally generated interrupt whenever it is processed. The other internally generated exceptions are caused by illegal instructions, instruction fetches from odd addresses, and privilege violations.

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 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 processing. If the exception is an interrupt and the M bit is set, the M bit is then cleared, and a short four-word exception stack frame is saved on the master stack, indicating the exception is saved on the bottom of the interrupt stack. If the exception is a reset, the M bit is simply cleared, then the reset vector is accessed.

The MC68020 provides an extension to the exception stacking process. 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., 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 multiplying 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 value 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.

ON-CHIP INSTRUCTION CACHE

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 loop mode operation, which allows certain instructions, when coupled with the DBcc instruction, to execute without the overhead of instruction fetches. In effect, this is a three-word cache. Although the cache hardware has been supplied in a full range of computer systems for many years, technology now allows this feature to be integrated into the microprocessor.

MC68020 CACHE GOALS

There were two primary design goals for the MC68020 microprocessor cache. The first design goal was to reduce the processor external bus activity. In a given M68000 system, the processor will use approximately 80 to 90 percent (or greater) of the available bus bandwidth due to the extremely efficient prefetching algorithm and the overall speed of the internal architecture. Thus, in an M68000 system with more than one bus master (such as a processor and a DMA device) or in a multiprocessor system, performance degradation can occur due to lack of available bus bandwidth. Therefore, an important goal for the MC68020 on-chip cache was to provide a substantial increase in the total available bus bandwidth.

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 access time now 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, an MC68020 on-chip cache provides a substantial CPU performance increase or allows much slower and less expensive memories to be used for the same processor performance.

The throughput increase in the MC68020 is gained in two ways. First, the MC68020 cache is accessed in two clock cycles versus the three cycles (minimum) required for an external access. Any instruction fetch that is currently resident in the cache will provide a 33-percent improvement over the corresponding external access. Second, the cache allows instruction stream fetches

and operand accesses to proceed in parallel. For example, if the MC68020 requires both an instruction stream access and an operand access and if the instruction is resident in the cache, the operand access proceeds unimpeded rather than being queued behind the instruction fetch. Similarly, the MC68020 is capable of executing several internal instructions (instructions that do not require the bus) while completing an operand access for another instruction.

INSTRUCTION CACHE

The MC68020 instruction cache is a 256-byte direct-mapped cache organized as 64 long-word entries. Each cache entry consists of a tag field composed of the upper 24 address bits, the FC2 (user/supervisor) value, one valid bit, and 32 bits of instruction data (see Figure 5).

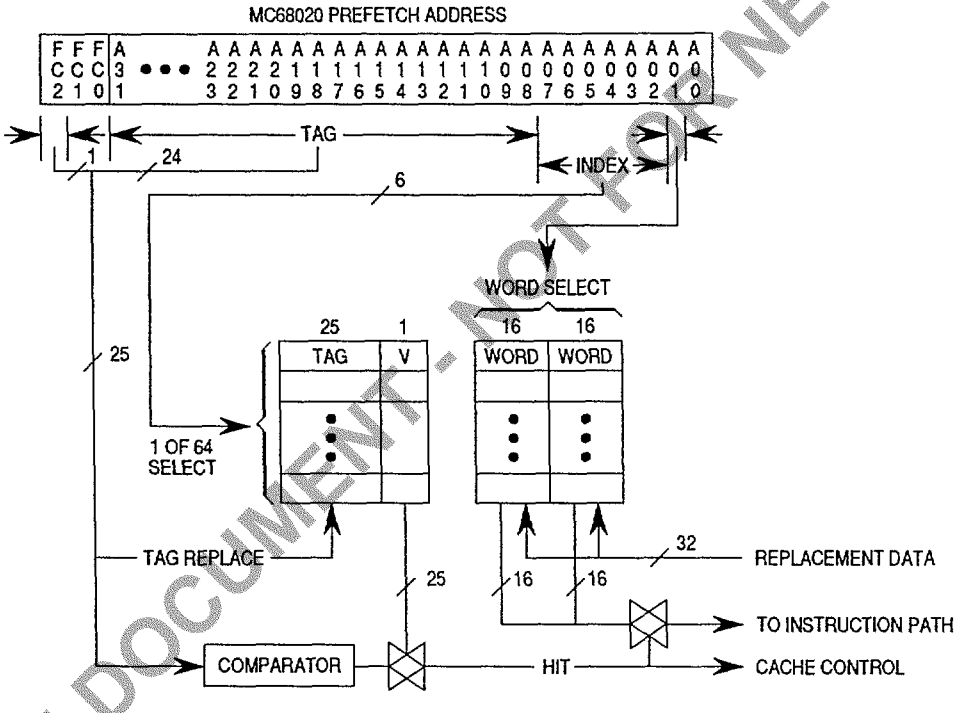


Figure 5. On-Chip Cache Organization

The MC68020 employs a 32-bit data bus and fetches instructions on long-word address boundaries. Hence, each 32-bit instruction fetch brings in two 16-bit instruction words that are written into the on-chip cache. When the cache is enabled, the subsequent prefetch finds the next 16-bit instruction word in the cache and saves the related bus cycle. However, even if the cache is not enabled, the bus controller still holds the full 32 bits and can satisfy the subsequent

prefetch and again save the related bus cycle. Thus, even when the on-chip instruction cache is not enabled, the bus controller can provide a savings of up to 50 percent of instruction prefetches.

SIGNAL DESCRIPTION

The MC68020 is offered in a 114-lead pin grid array (PGA) package. Figure 6 illustrates the functional signal groups, and Table 5 lists the signals and their function.

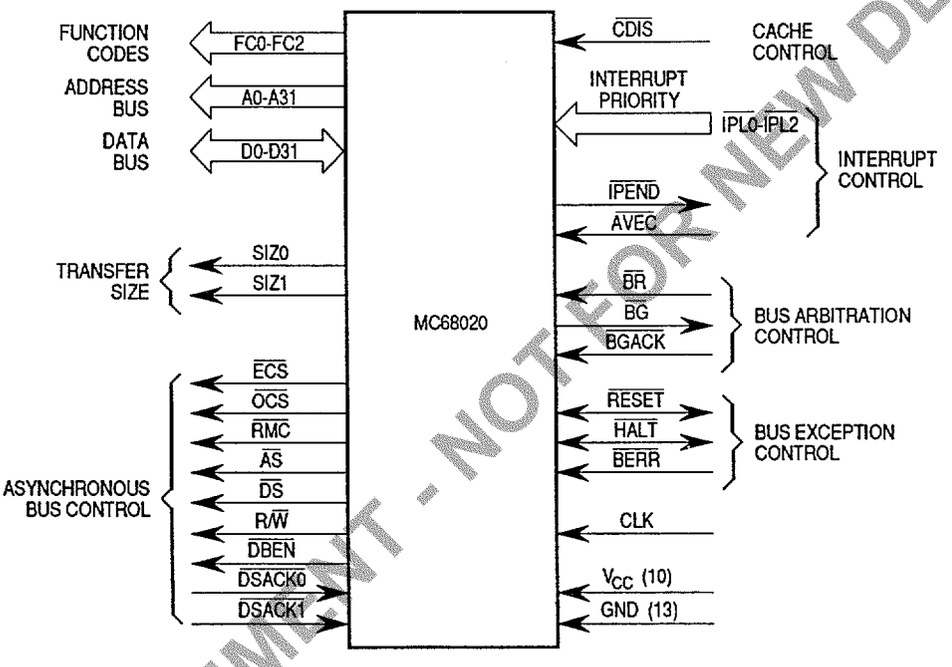


Figure 6. Functional Signal Groups

The V_{CC} and GND pins are separated into four groups to provide individual power supply connections for the address bus buffers, data bus buffers, and all other output buffers and internal logic.

Group	V_{CC}	GND
Address Bus	A9, D3	A10, B9, C3, F12
Data Bus	M8, N8, N13	L7, L11, N7, K3
Logic	D1, D2, E3, G11, G13	G12, H13, J3, K1
Clock	—	B1

Table 5. Signal Index

Signal Name	Mnemonic	Function
Function Codes	FC2–FC0	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 A1 and A0, define the active sections of the data bus.
External Cycle Start	$\overline{ECS}$	Provides an indication that a bus cycle is beginning.
Operand Cycle Start	$\overline{OCS}$	Identical operation to that of $\overline{ECS}$ except that $\overline{OCS}$ is asserted only during the first bus cycle of an operand transfer.
Read/Write	R/$\overline{W}$	Defines the bus transfer as a processor read or write.
Read-Modify-Write Cycle	$\overline{RMC}$	Provides an indicator that the current bus cycle is part of an indivisible read-modify-write operation.
Address Strobe	$\overline{AS}$	Indicates that a valid address is on the bus.
Data Strobe	$\overline{DS}$	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 MC68020.
Data Buffer Enable	$\overline{DBEN}$	Provides an enable signal for external data buffers.
Data Transfer and Size Acknowledge	$\overline{DSACK0}$/ $\overline{DSACK1}$	Bus response signals that indicate the requested data transfer operation has 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.
Interrupt Priority Level	$\overline{IPL0}$–$\overline{IPL2}$	Provides an encoded interrupt level to the processor.
Interrupt Pending	$\overline{IPEND}$	Indicates that an interrupt is pending.
Autovector	$\overline{AVEC}$	Requests an autovector during an interrupt acknowledge cycle.
Bus Request	$\overline{BR}$	Indicates that an external device requires bus mastership.
Bus Grant	$\overline{BG}$	Indicates that an external device may assume bus mastership.
Bus Grant Acknowledge	$\overline{BGACK}$	Indicates that an external device has assumed bus mastership.
Reset	$\overline{RESET}$	System reset.
Halt	$\overline{HALT}$	Indicates that the processor should suspend bus activity.
Bus Error	$\overline{BERR}$	Indicates that an erroneous bus operation is being attempted.
Cache Disable	$\overline{CDIS}$	Dynamically disables the on-chip cache to assist emulator support.
Clock	CLK	Clock input to the processor.
Power Supply	V_{CC}	Power supply.
Ground	GND	Ground connection.

ELECTRICAL SPECIFICATIONS

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	VCC	-0.3 to +7.0	V
Input Voltage	Vin	-0.5 to +7.0	V
Operating Temperature Range Minimum Ambient Temperature Maximum Ambient Temperature PGA , PPGA,PQFP Maximum Junction Temperature CQFP	 T_A T_A T_J	 0 70 110	°C

THERMAL 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.

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

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

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 thermal 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}. θ_{JC} represents the barrier to heat flow from the semiconductor junction to the package (case) surface, and θ_{CA} represents the barrier to heat flow from the case to the ambient air. 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, forced air cooling, and use of thermal convection to increase air flow over the device. Thus, good thermal design 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) results in a lower semiconductor junction temperature.

Thermal Resistance (°C/W)

The following table provides thermal resistance characteristic for junction to ambient and junction to case for the different packages with natural convection and no heatsink.

Characteristic – Natural Convection and No Heatsink	θ_{JA}	θ_{JC}
Thermal Resistance		
PGA Package	26	3
PPGA Package	32	TBD
CQFP Package	46	15
PQFP Package	42	20

Resistance is to bottom center (pin side) of case for PGA and PPGA packages, top center of case for CQFP package.

CQFP Package

Table 5 provide typical and worst case thermal charastics for the CQFP package both with and without a heatsink. The heatsink used is black anodized alluminum alloy, 0.72"x0.75"x0.6" high with an omnidirectional 5x6 array of fins. Attachment was made using Epolite 6400 one part epoxy.

Table 5. θ_{JA} Vs. Airflow - CQFP package

	Airflow in linear feet/minute		
	0*	200	500
θ_{JA} Maximum			
No Heatsink	46	28	24
With Heatsink	35	20	18
θ_{JA} Typical			
No Heatsink	43	25	21
With Heatsink	32	17	15

* Natural convection

Table 6 shows the relationship between clock speed and power dissipation for any package type. The worst case operating conditions are used for thermal management design, while typical values are used for reliability analysis.

Table 6. Power vs. Rated Frequency (at T_J Maximum = 110°C)

Rated Frequency	P_D Maximum	P_D Typical
MHz	Watts	Watts
33	1.4	0.84
25	1.2	0.72
20	1.0	0.60
16	0.9	0.54

Table 7 shows the relationship between board temperature rise and power dissipation in the test environment for the CQFP package. Derate θ_{JA} based on measurements made in the application by adding $(0.8/P_D)*[T_{ba(application)}-T_{ba(table)}]$ to the θ_{JA} values in the table. Board temperature was measured on the top surface of the board directly under the device.

Table 7. Temperature Rise of Board vs. P_D - CQFP Package

Natural Convection	P_D		
	0.6W	1.0W	1.75W
T_{ba} (°C) - No Heatsink	18	27	53

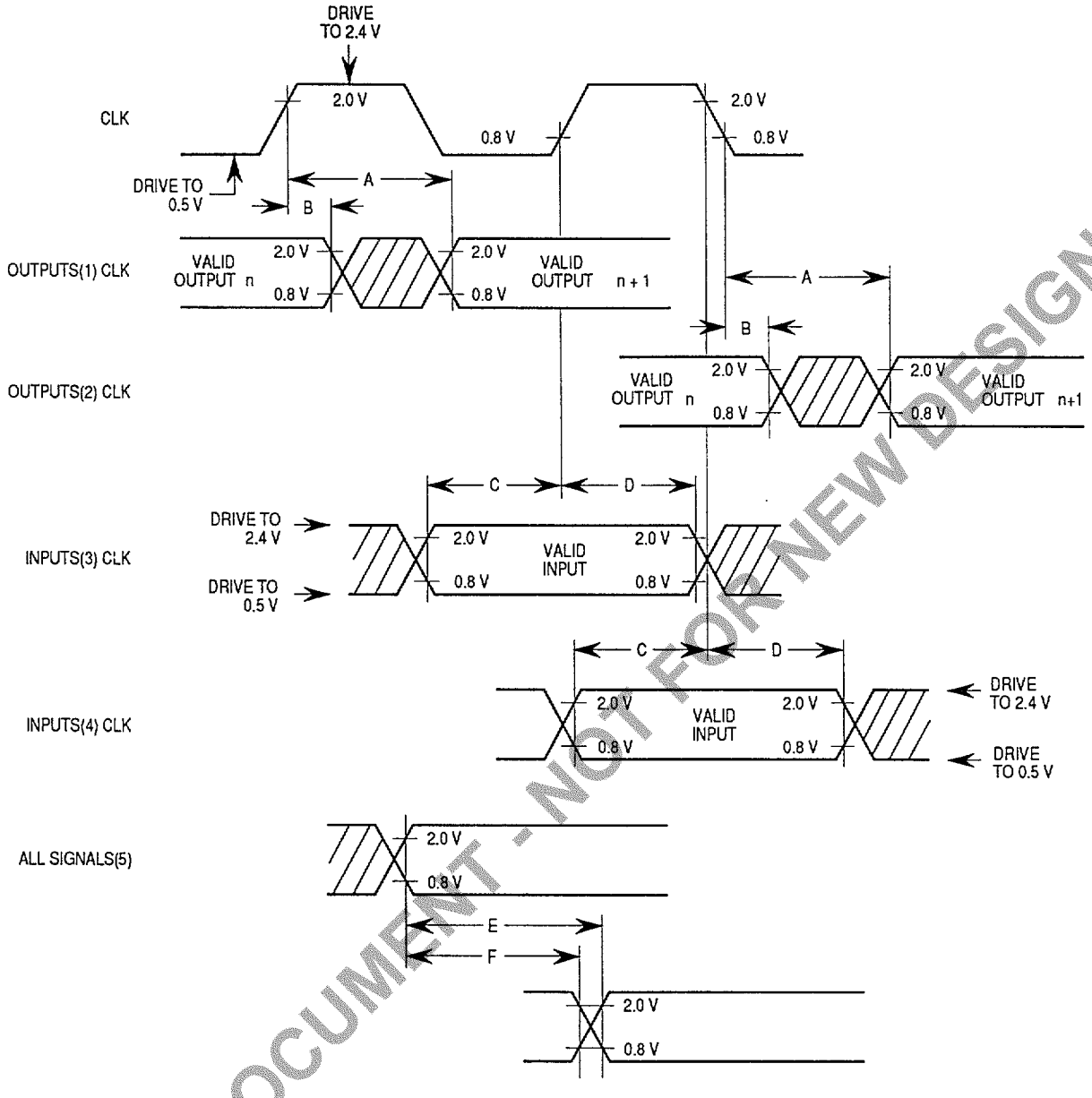
Values for thermal resistance presented in this document were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent 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 MC68020 clock input and, possibly, relative to one or more other signals.

The measurement of the AC specifications is defined by the waveforms in Figure 7. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 7. Outputs of the MC68020 are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs to the MC68020 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 MC68020 to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.



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 7. Drive Levels and Test Points for AC Specifications

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

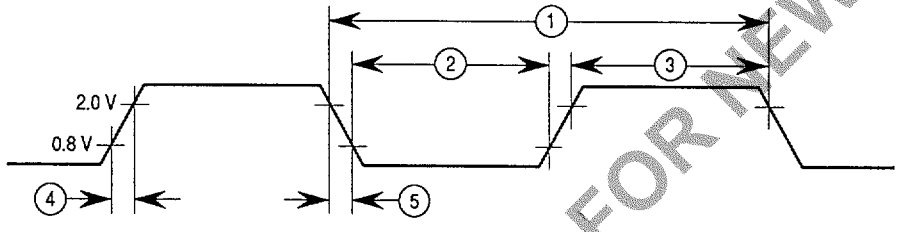
Characteristic		Symbol	Min	Max	Unit
Input High Voltage		V_{IH}	2.0	V_{CC}	V
Input Low Voltage		V_{IL}	GND −0.5	0.8	V
Input Leakage Current GND ≤ V_{in} ≤ V_{CC}	$\overline{BERR}$, $\overline{BR}$, $\overline{BGACK}$, CLK, $\overline{IPL0}$-$\overline{IPL2}$, $\overline{AVEC}$, $\overline{CDIS}$, $\overline{DSACK0}$, $\overline{DSACK1}$	I_{in}	−1	1.0	μA
	$\overline{HALT}$, $\overline{RESET}$		−20	20	
Hi-Z (Off-State) Leakage Current @ 2.4 V/0.5 V	A31–A0, $\overline{AS}$, $\overline{DBEN}$, $\overline{DS}$, D31–D0, FC2–FC0, R/$\overline{W}$, $\overline{RMC}$, SIZ1–SIZ0	I_{TSI}	−20	20	μA
Output High Voltage I_{OH} = 400 μA	A31–A0, $\overline{AS}$, $\overline{BG}$, D31–D0, $\overline{DBEN}$, $\overline{DS}$, $\overline{ECS}$, R/$\overline{W}$, $\overline{IPEND}$, $\overline{OCS}$, $\overline{RMC}$, SIZ1–SIZ0, FC2–FC0,	V_{OH}	2.4	—	V
Output Low Voltage		V_{OL}			V
I_{OL} = 3.2 mA	A31–A0, FC2–FC0, SIZ1–SIZ0, $\overline{BG}$, D31–D0		—	0.5	
I_{OL} = 5.3 mA	$\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, $\overline{RMC}$, $\overline{DBEN}$, $\overline{IPEND}$		—	0.5	
I_{OL} = 2.0 mA	$\overline{ECS}$, $\overline{OCS}$		—	0.5	
I_{OL} = 10.7 mA	$\overline{HALT}$, $\overline{RESET}$		—	0.5	
Power Dissipation (T_A = 0°C)		P_D	—	2.0	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
	All Other			130	

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

AC ELECTRICAL SPECIFICATIONS — CLOCK INPUT (see Figure 8)

Num.	Characteristic	16.67 MHz		20 MHz		25 MHz*		33.33 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
	Frequency of Operation	8	16.67	12.5	20	12.5	25	12.5	33.33	MHz
1	Cycle Time	60	125	50	80	40	80	30	80	ns
2, 3	Clock Pulse Width (Measured from 1.5 V to 1.5 V for 25 and 33.33 MHz)	24	95	20	54	19	61	14	66	ns
4, 5	Rise and Fall Times	—	5	—	5	—	4	—	3	ns

*These specifications represent an improvement over previously published specifications for the 25-MHz MC68020 and are valid only for product bearing date codes of 8827 and later.



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 8. Clock Input Timing Diagram

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

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

Num.	Characteristic	16.67 MHz		20 MHz		25 MHz*		33.33 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
6	Clock High to Address, FC, Size, $\overline{RMC}$ Valid	0	30	0	25	0	25	0	21	ns
6A	Clock High to $\overline{ECS}$, $\overline{OCS}$ Asserted	0	20	0	15	0	12	0	10	ns
7	Clock High to Address, Data, FC, Size, $\overline{RMC}$, High Impedance	0	60	0	50	0	40	0	30	ns
8	Clock High to Address, FC, Size, $\overline{RMC}$ Invalid	0	—	0	—	0	—	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$ Asserted	3	30	3	25	3	18	3	15	ns
9A[1]	$\overline{AS}$ to $\overline{DS}$ Assertion (Read) (Skew)	−15	15	−10	10	−10	10	−10	10	ns
9B[11]	$\overline{AS}$ Asserted to $\overline{DS}$ Asserted (Write)	37	—	32	—	27	—	22	—	ns
10	$\overline{ECS}$ Width Asserted	20	—	15	—	15	—	10	—	ns
10A	$\overline{OCS}$ Width Asserted	20	—	15	—	15	—	10	—	ns
10B[7]	$\overline{ECS}$, $\overline{OCS}$ Width Negated	15	—	10	—	5	—	5	—	ns
11	Address, FC, Size, $\overline{RMC}$ Valid to $\overline{AS}$ (and $\overline{DS}$ Asserted Read)	15	—	10	—	6	—	5	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	0	30	0	25	0	15	0	15	ns
12A	Clock Low to $\overline{ECS}$, $\overline{OCS}$ Negated	0	30	0	25	0	15	0	15	ns
13	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC, Size, $\overline{RMC}$ Invalid	15	—	10	—	10	—	5	—	ns
14	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted	100	—	85	—	70	—	50	—	ns
14A	$\overline{DS}$ Width Asserted Write	40	—	38	—	30	—	25	—	ns
15	$\overline{AS}$, $\overline{DS}$ Width Negated	40	—	38	—	30	—	23	—	ns
15A[8]	$\overline{DS}$ Negated to $\overline{AS}$ Asserted	35	—	30	—	25	—	18	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, $\overline{DBEN}$ High Impedance	—	60	—	50	—	40	—	30	ns
17	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	15	—	10	—	10	—	5	—	ns
18	Clock High to R/$\overline{W}$ High	0	30	0	25	0	20	0	15	ns
20	Clock High to R/$\overline{W}$ Low	0	30	0	25	0	20	0	15	ns
21	R/$\overline{W}$ High to $\overline{AS}$ Asserted	15	—	10	—	5	—	5	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	75	—	60	—	50	—	35	—	ns
23	Clock High to Data Out Valid	—	30	—	25	—	25	—	18	ns
25	$\overline{DS}$ Negated to Data Out Invalid	15	—	10	—	5	—	5	—	ns
25A[9]	$\overline{DS}$ Negated to $\overline{DBEN}$ Negated (Write)	15	—	10	—	5	—	5	—	ns
26	Data Out Valid to $\overline{DS}$ Asserted (Write)	15	—	10	—	5	—	5	—	ns
27	Data-In Valid to Clock Low (Data Setup)	5	—	5	—	5	—	5	—	ns
27A	Late $\overline{BERR}$/$\overline{HALT}$ Asserted to Clock Low Setup Time	20	—	15	—	10	—	5	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACKx}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	0	80	0	65	0	50	0	40	ns
29	$\overline{DS}$ Negated to Data-In Invalid (Data-In Hold Time)	0	—	0	—	0	—	0	—	ns
29A	$\overline{DS}$ Negated to Data-In (High Impedance)	—	60	—	50	—	40	—	30	ns
31[2]	$\overline{DSACKx}$ Asserted to Data-In Valid	—	50	—	43	—	32	—	17	ns
31A[3]	$\overline{DSACKx}$ Asserted to $\overline{DSACKx}$ Valid ($\overline{DSACK}$ Asserted Skew)	—	15	—	10	—	10	—	10	ns

AC ELECTRICAL SPECIFICATIONS (Continued)

Num.	Characteristic	16.67 MHz		20 MHz		25 MHz*		33.33 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
32	$\overline{\text{RESET}}$ Input Transition Time	—	1.5	—	1.5	—	1.5	—	1.5	Clks
33	Clock Low to $\overline{\text{BG}}$ Asserted	0	30	0	25	0	20	0	20	ns
34	Clock Low to $\overline{\text{BG}}$ Negated	0	30	0	25	0	20	0	20	ns
35	$\overline{\text{BR}}$ Asserted to $\overline{\text{BG}}$ Asserted ($\overline{\text{RMC}}$ Not Asserted)	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37	$\overline{\text{BGACK}}$ Asserted to $\overline{\text{BG}}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37A[6]	$\overline{\text{BGACK}}$ Asserted to $\overline{\text{BR}}$ Negated	0	1.5	0	1.5	0	1.5	0	1.5	Clks
39	$\overline{\text{BG}}$ Width Negated	90	—	75	—	60	—	50	—	ns
39A	$\overline{\text{BG}}$ Width Asserted	90	—	75	—	60	—	50	—	ns
40	Clock High to $\overline{\text{DBEN}}$ Asserted (Read)	0	30	0	25	0	20	0	15	ns
41	Clock Low to $\overline{\text{DBEN}}$ Negated (Read)	0	30	0	25	0	20	0	15	ns
42	Clock Low to $\overline{\text{DBEN}}$ Asserted (Write)	0	30	0	25	0	20	0	15	ns
43	Clock High to $\overline{\text{DBEN}}$ Negated (Write)	0	30	0	25	0	20	0	15	ns
44	R/$\overline{\text{W}}$ Low to $\overline{\text{DBEN}}$ Asserted (Write)	15	—	10	—	10	—	5	—	ns
45[5]	$\overline{\text{DBEN}}$ Width Asserted Read	60	—	50	—	40	—	30	—	ns
	Write	120	—	100	—	80	—	60	—	
46	R/$\overline{\text{W}}$ Width Valid (Write or Read)	150	—	125	—	100	—	75	—	ns
47A	Asynchronous Input Setup Time	5	—	5	—	5	—	5	—	ns
47B	Asynchronous Input Hold Time	15	—	15	—	10	—	10	—	ns
48[4]	$\overline{\text{DSACK}}$x Asserted to $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted	—	30	—	20	—	18	—	15	ns
53	Data Out Hold from Clock High	0	—	0	—	0	—	0	—	ns
55	R/$\overline{\text{W}}$ Valid to Data Bus Impedance Change	30	—	25	—	20	—	20	—	ns
56	$\overline{\text{RESET}}$ Pulse Width (Reset Instruction)	512	—	512	—	512	—	512	—	Clks
57	$\overline{\text{BERR}}$ Negated to $\overline{\text{HALT}}$ Negated (Rerun)	0	—	0	—	0	—	0	—	ns
58[10]	$\overline{\text{BGACK}}$ Negated to Bus Driven	1	—	1	—	1	—	1	—	Clks
59[10]	$\overline{\text{BG}}$ Negated to Bus Driven	1	—	1	—	1	—	1	—	Clks

*These specifications represent an improvement over previously published specifications for the 25-MHz MC68020 and are valid only for product bearing date codes of 8827 and later.

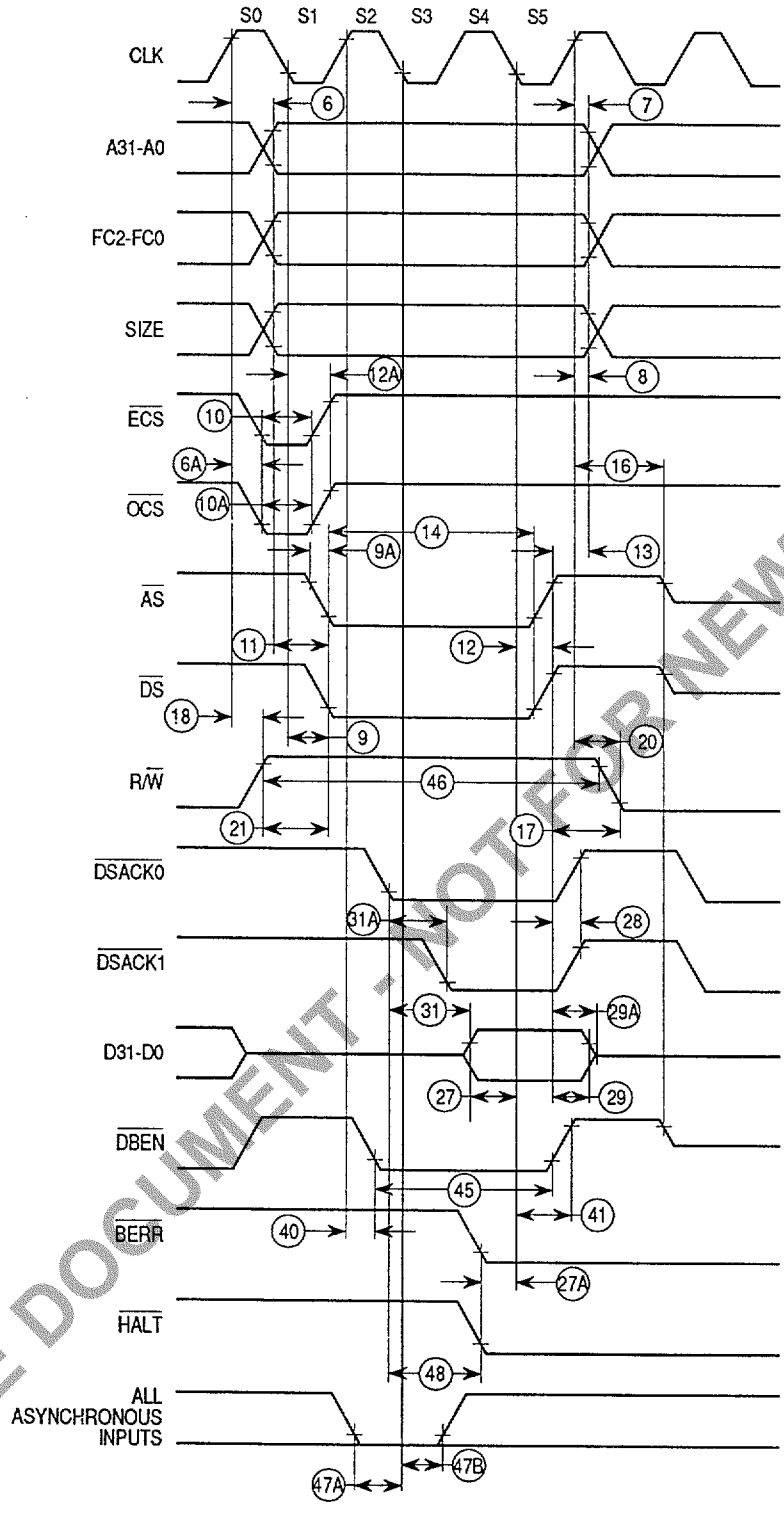
Temperature must be kept within described ranges in the Thermal Characteristics section.

NOTES:

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 $\overline{\text{DSACK}}$x low to data setup time (#31) and $\overline{\text{DSACK}}$x low to $\overline{\text{BERR}}$ low setup time (#48) can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle, and $\overline{\text{BERR}}$ must only satisfy the late $\overline{\text{BERR}}$ low to clock low setup time (#27A) for the following clock cycle.
3. This parameter specifies the maximum allowable skew between $\overline{\text{DSACK0}}$ to $\overline{\text{DSACK1}}$ asserted or $\overline{\text{DSACK1}}$ to $\overline{\text{DSACK0}}$ asserted; specification #47A must be met by $\overline{\text{DSACK0}}$ or $\overline{\text{DSACK1}}$.
4. This specification applies to the first ($\overline{\text{DSACK0}}$ or $\overline{\text{DSACK1}}$) $\overline{\text{DSACK}}$x signal asserted. In the absence of $\overline{\text{DSACK}}$x, $\overline{\text{BERR}}$ is an asynchronous input using the asynchronous input setup time (#47A).
5. $\overline{\text{DBEN}}$ may stay asserted on consecutive write cycles.
6. The minimum values must be met to guarantee proper operation. If this maximum value is exceeded, $\overline{\text{BG}}$ may be reasserted.
7. This specification indicates the minimum high time for $\overline{\text{ECS}}$ and $\overline{\text{OCS}}$ in the event of an internal cache hit followed immediately by a cache miss or operand cycle.
8. This specification guarantees operation with the MC68881/MC68882, which specifies a minimum time for $\overline{\text{DS}}$ negated to $\overline{\text{AS}}$ asserted (specification #13A in the *MC68881/MC68882 User's Manual*). Without this specification, incorrect interpretation of specifications #9A and #15 would indicate that the MC68020 does not meet the MC68881/MC68882 requirements.

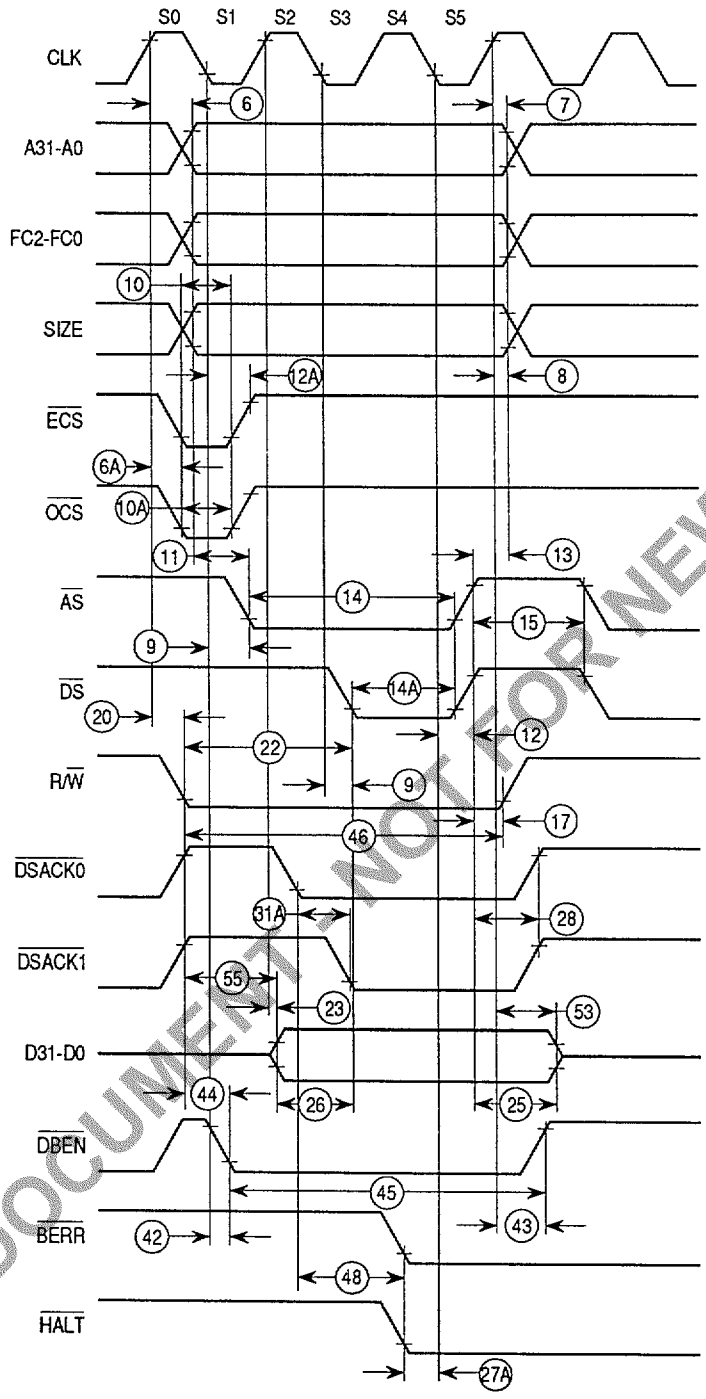
AC ELECTRICAL SPECIFICATIONS (Concluded)

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 $\overline{DBEN}$.
10. These specifications allow system designers to guarantee that an alternate bus master has stopped driving the bus when the MC68020 regains control of the bus after an arbitration sequence.
11. This specification allows system designers to qualify the $\overline{CS}$ signal of an MC68881/MC68882 with $\overline{AS}$ (allowing 7 ns for a gate delay) and still meet the $\overline{CS}$ to $\overline{DS}$ setup time requirement (specification #8B of the *MC68881/MC68882 User's Manual*).



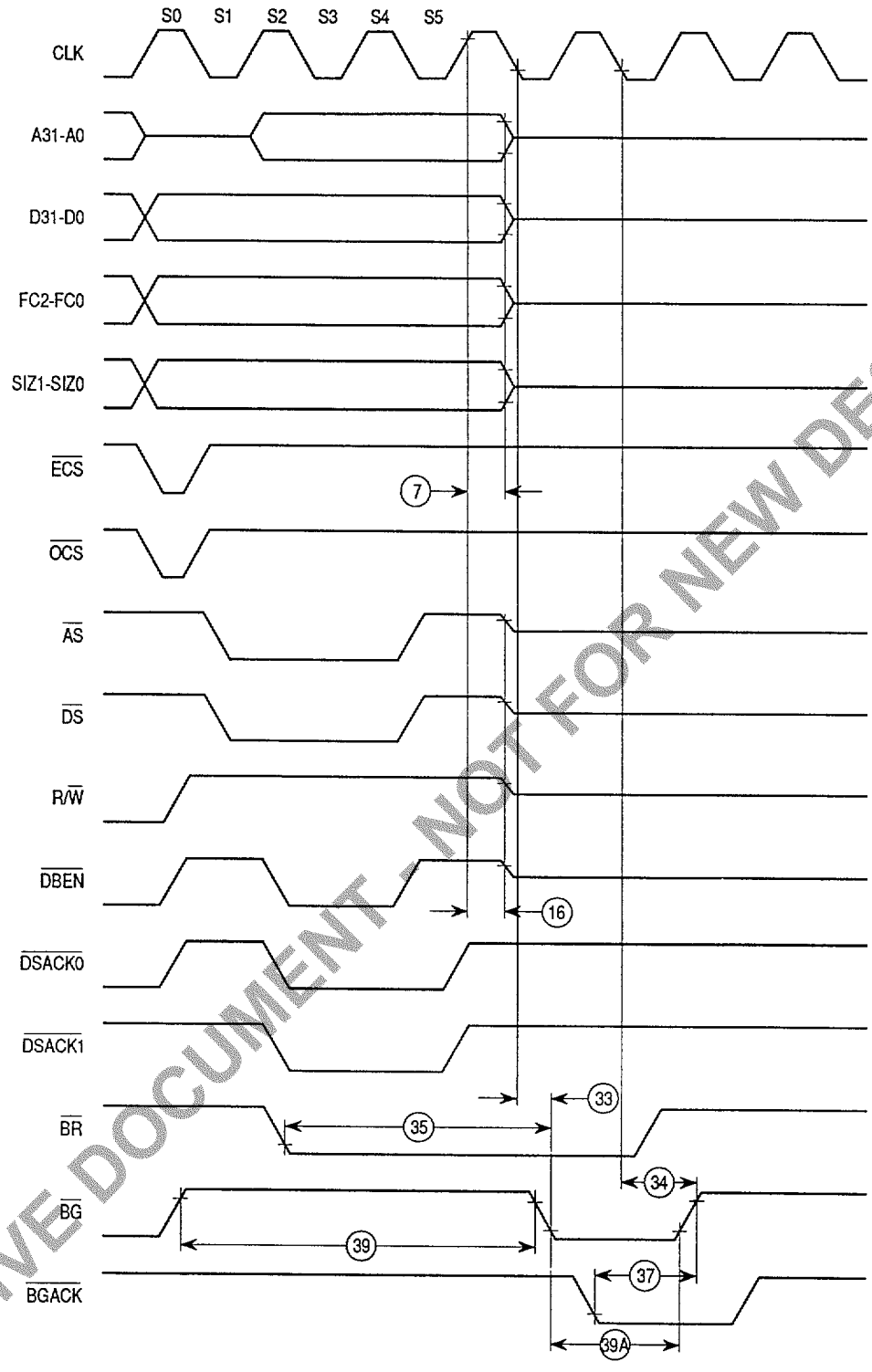
NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 9. Read 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 10. Write Cycle Timing Diagram



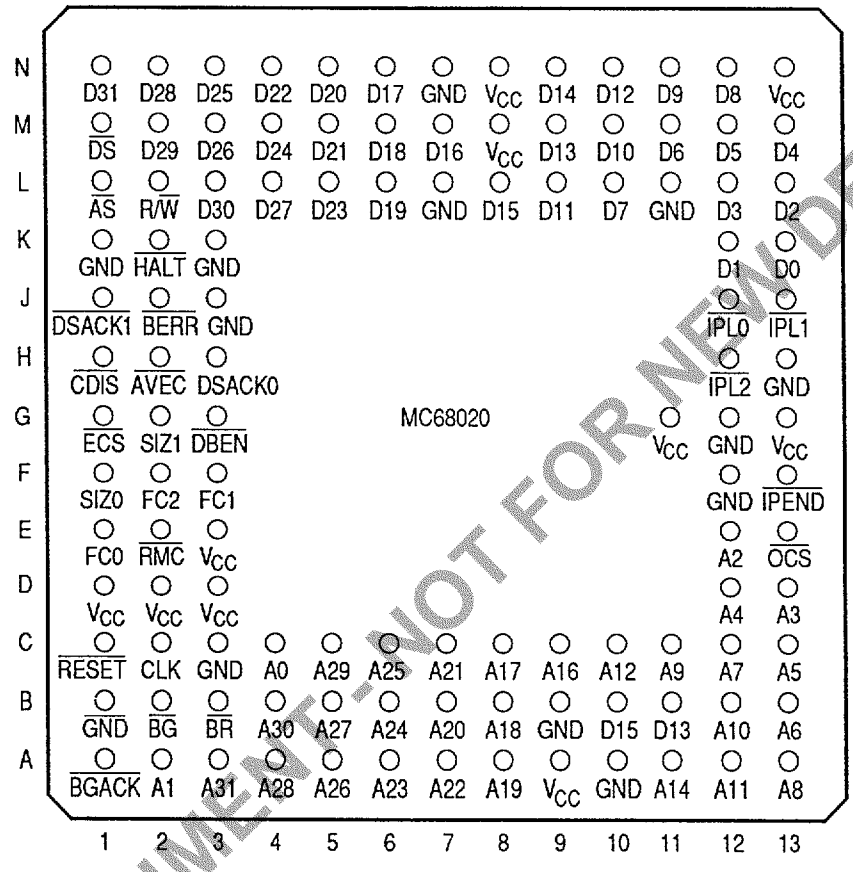
NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and a high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 11. Bus Arbitration Timing Diagram

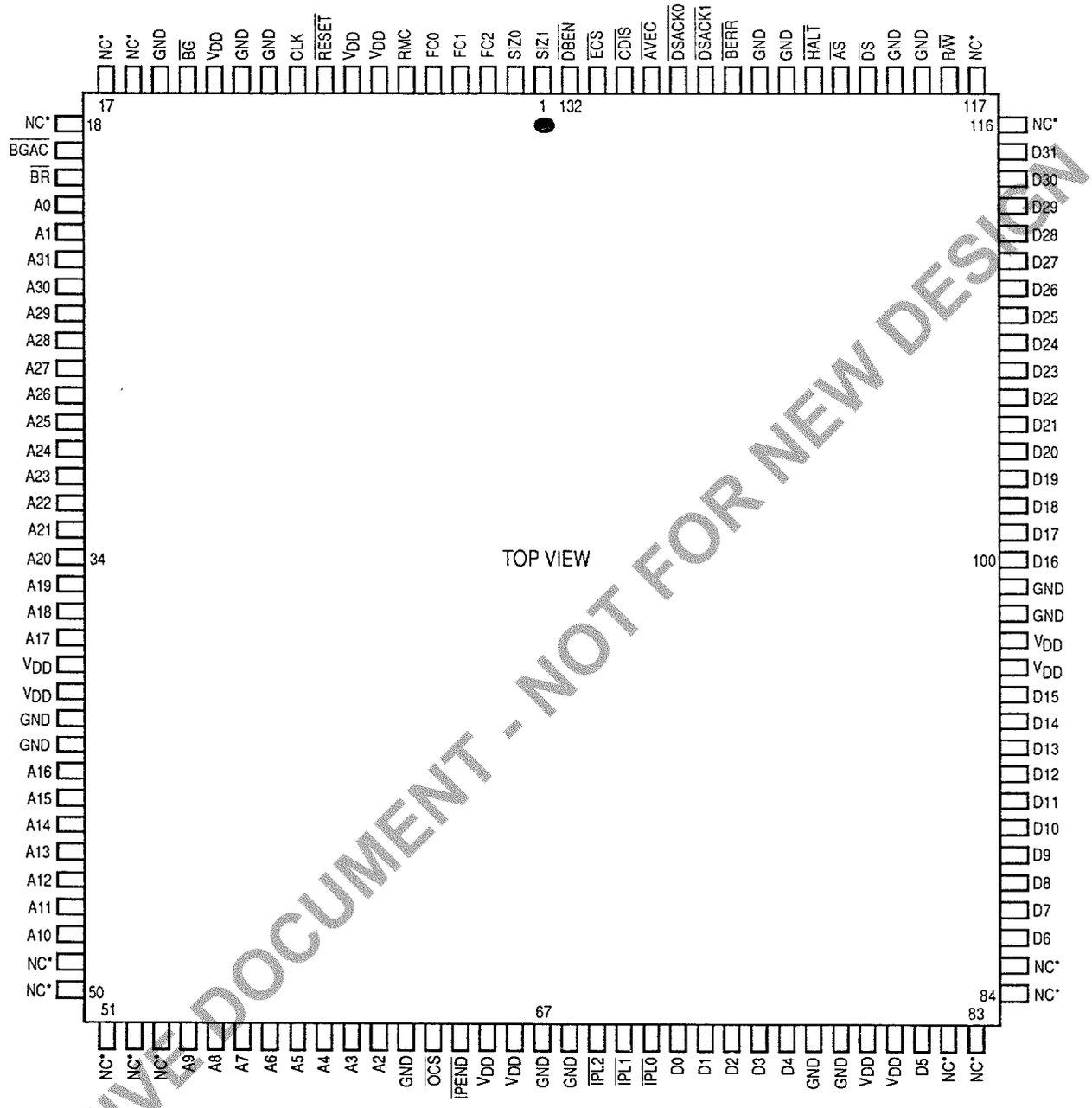
MECHANICAL DATA

PIN ASSIGNMENTS

Pin Grid Array (RC, RP Suffix)



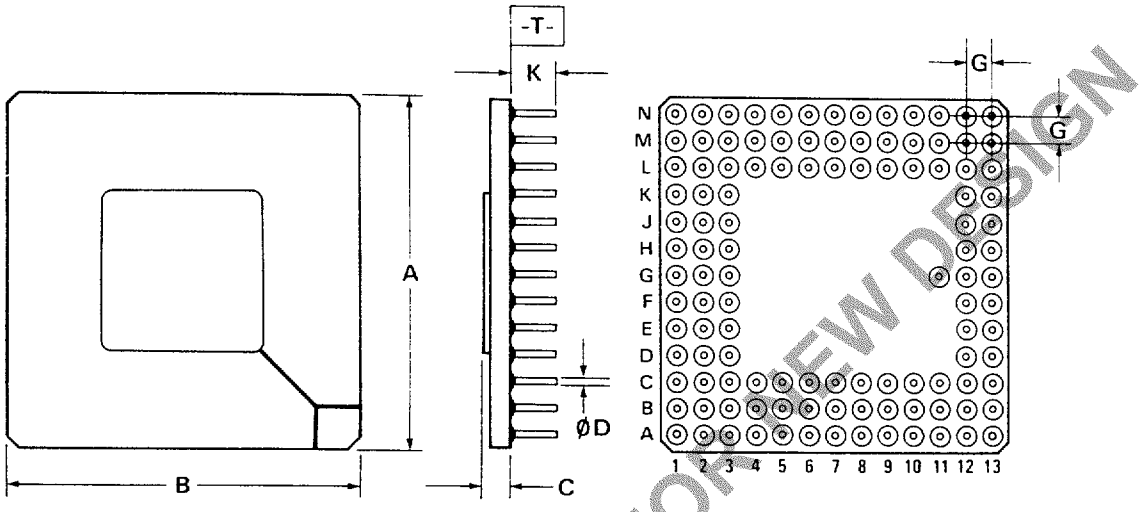
Ceramic Quad Flat Pack (FE Suffix)
Plastic Quad Flat Pack (FC Suffix)



*NC - Do not connect to this pin.

PACKAGE DIMENSIONS

RC Suffix
Case No. 791-01



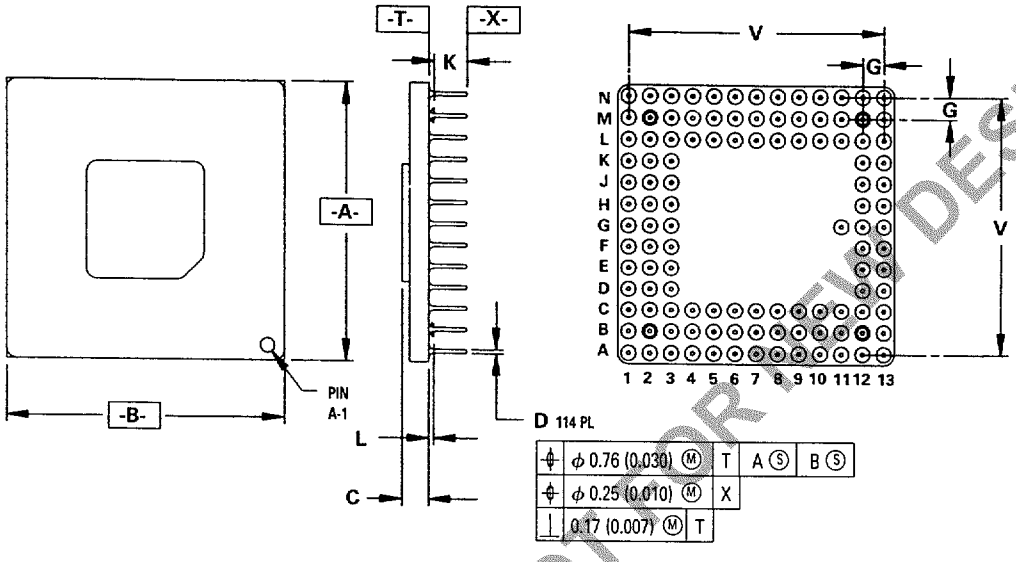
NOTES:
1. A AND B ARE DATUMS AND T IS A DATUM SURFACE.
2. POSITIONAL TOLERANCE FOR LEADS (114 PLACES).

⌖	0.13 (0.005) Ⓜ	T	A Ⓢ	B Ⓢ

3. DIMENSIONING AND TOLERANCING PER Y14.5M, 1982.
4. CONTROLLING DIMENSION: INCH

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	34.04	35.05	1.340	1.380
B	34.04	35.05	1.340	1.380
C	2.54	3.81	0.100	0.150
D	0.43	0.55	0.017	0.022
G	2.54 BSC		0.100 BSC	
K	4.32	4.95	0.170	0.195

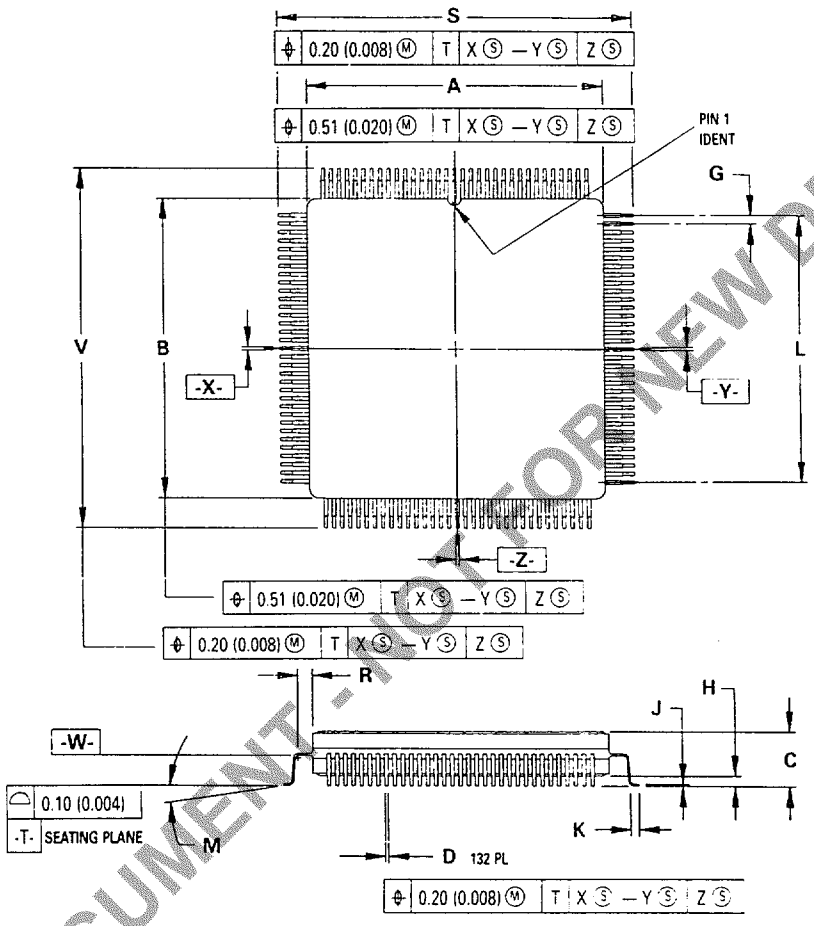
RP Suffix
Case No. 789E-02



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	34.68	35.17	1.365	1.385
B	34.68	35.17	1.365	1.385
C	4.07	4.57	0.160	0.180
D	0.44	0.55	0.017	0.022
G	2.54 BSC		0.100 BSC	
K	2.93	3.68	0.115	0.145
L	0.26	0.76	0.010	0.030
V	30.48 BSC		1.200 BSC	

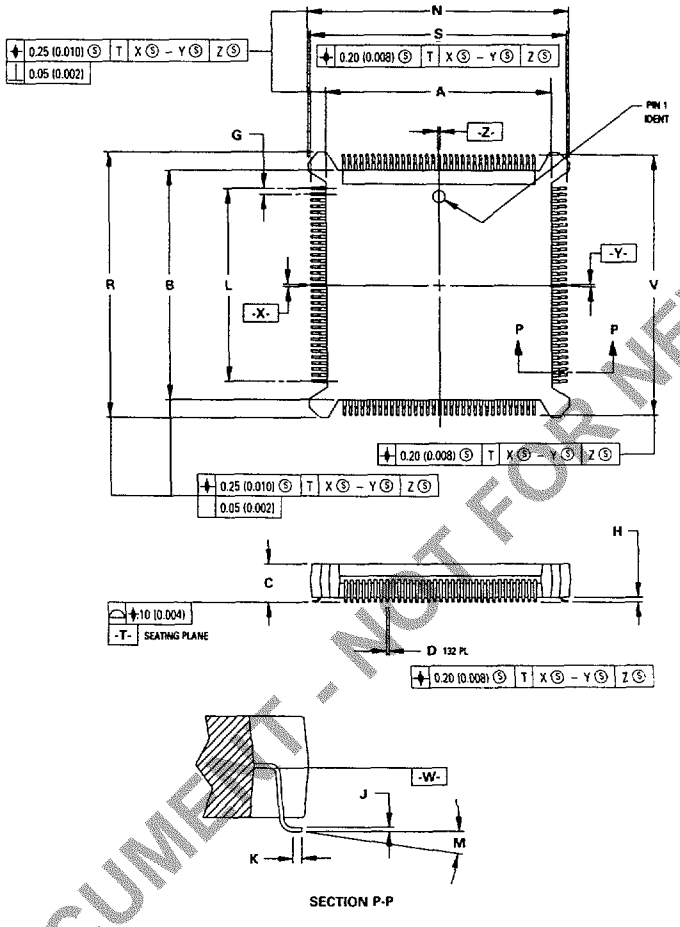
FE Suffix
Case No.831-01



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	21.85	22.86	0.860	0.900
B	21.85	22.86	0.860	0.900
C	3.94	4.31	0.155	0.170
D	0.204	0.292	0.0080	0.0115
G	0.64 BSC		0.025 BSC	
H	0.64	0.88	0.025	0.035
J	0.13	0.20	0.005	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
R	0.64	—	0.025	—
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIM A AND B DEFINE MAXIMUM CERAMIC BODY DIMENSIONS INCLUDING GLASS PROTRUSION AND MISMATCH OF CERAMIC BODY TOP AND BOTTOM.
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIM S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIM A AND B TO BE DETERMINED AT DATUM PLANE -W-.

RC Suffix
Case No. 831A-01



DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX
A	24.06	24.20	0.947	0.953
B	24.06	24.20	0.947	0.953
C	4.07	4.57	0.160	0.180
D	0.21	0.30	0.008	0.012
G	0.64 BSC		0.025 BSC	
H	0.51	1.01	0.020	0.040
J	0.16	0.20	0.006	0.008
K	0.51	0.76	0.020	0.030
L	20.32 REF		0.800 REF	
M	0°	8°	0°	8°
N	27.88	28.01	1.097	1.103
R	27.88	28.01	1.097	1.103
S	27.31	27.55	1.075	1.085
V	27.31	27.55	1.075	1.085

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSIONS A, B, N, AND R DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION FOR DIMENSIONS A AND B IS 0.25 (0.010), FOR DIMENSIONS N AND R IS 0.18 (0.007).
4. DATUM PLANE -W- IS LOCATED AT THE UNDERSIDE OF LEADS WHERE LEADS EXIT PACKAGE BODY.
5. DATUMS X-Y AND Z TO BE DETERMINED WHERE CENTER LEADS EXIT PACKAGE BODY AT DATUM -W-.
6. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE, DATUM -T-.
7. DIMENSIONS A, B, N, AND R TO BE DETERMINED AT DATUM PLANE -W-.

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

AR7739-15 PRINTED IN USA 1/91 IMPERIAL LITHO 77255 18,000 MPC YGABAA