MAY 8 1991

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by MC68EC020/D

MC68EC020

Technical Summary
32-Bit Embedded Controller

The MC68EC020 is an economical high-performance embedded controller that has been designed specifically to suit the needs of the embedded controller market. The HCMOS MC68EC020 has an internal 32-bit architecture that is supported by a 32-bit data bus. This architecture and the unit's on-chip instruction cache provide a fast and efficient processing device that can satisfy the requirements of very sophisticated applications based on the high-level languages.

The main features of the MC68EC020 include:

- Object Code Compatible with M68000 Family
- A 24-Bit Address Bus with 16-Mbyte Direct Addressing Range
- A 32-Bit Data Bus with Dynamic Bus Sizing for 8-/16-/32-Bit Memories and Peripherals
- An On-Chip Instruction Cache for Faster Instruction Execution
- 16.67 MHz or 25 MHz Processor Speeds
- Plastic Pin Grid Array or Plastic Quad Flat Pack Packages

Additional features of the MC68EC020 include:

- Eighteen Addressing Modes
- Seven Data Types Including Bit Field Types for Applications such as Video Graphics
- Sixteen 32-Bit General-Purpose Data and Address Registers
- Two 32-Bit Supervisor Stack Pointers Plus Five 32-Bit Special-Purpose Control Registers
- Pipelined Architecture with Internal Parallelism
- Coprocessor Interface to the MC68881/MC68882 Floating-Point Coprocessors
- Low-Power HCMOS Circuitry

The MC68EC020 brings the performance level of the MC68020 to cost levels previously associated with only 16-bit microprocessors. The MC68EC020 benefits from the rich M68000 instruction set and its related high code density with low memory bandwidth requirements. The dynamically sized, asynchronous bus of the MC68EC020 allows the system designer to freely intermix 8-, 16-, or 32-bit data paths to optimize for best available cost/performance ratios.

MOTOROLA

INTRODUCTION

The MC68EC020 is a high-performance 32-bit HCMOS embedded controller that has been designed to fit the needs of the cost-sensitive embedded control market. Like the MC68020 on which it is based, the MC68EC020 is a further development of the M68000 Family and incorporates the best features of that product line, including code compatibility, common internal and bus architectures, and a complete offering of peripherals. All of these features make the MC68EC020 the logical choice for both system upgrade as well as new product family design.

In the MC68EC020, the popular MC68020 core has been optimized to provide a more economical design that still has a 32-bit external data bus to support the unit's internal 32-bit architecture. The data bus also has the flexibility to be dynamically sized to accommodate systems with 8-, 16-, and 32-bit memories and peripherals.

The MC68EC020 has a 24-bit address bus that is capable of addressing up to 16 Mbytes of memory. Eighteen addressing modes are available for maximum flexibility. The MC68EC020 has extended addressing modes that better support special-purpose applications and high-level languages.

The MC68EC020 includes a 256-byte on-chip instruction cache for greater system performance. The cache reduces bus activity by providing a local reference for a block of the user program. If the instructions required for the next operation are resident in the cache, the only necessary bus accesses will be for operand fetches. The cache also increases parallelism by allowing instructions that do not require the bus to execute while an operand store is being completed .

The MC68EC020 maintains full support of the M68000 coprocessor interface, allowing use of the MC68881 or MC68882 floating-point coprocessors.

Figure 1 shows a block diagram of the MC68EC020. The MC68EC020 has two main sections; the bus controller and the execution unit. The bus controller consists of the address and data bus circuits and the multiplexers required to support dynamic bus sizing. The bus controller also includes 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 an instruction cache with its associated control. The execution unit consists of an address section, an operand address section, and a data section. Microcode control is provided by a two-level store of microROM and nanoROM within the execution unit. The instruction pipe and other individual control sections provide the secondary decode of instructions and generate the actual control signals that decode and interpret the nanoROM and microROM information.

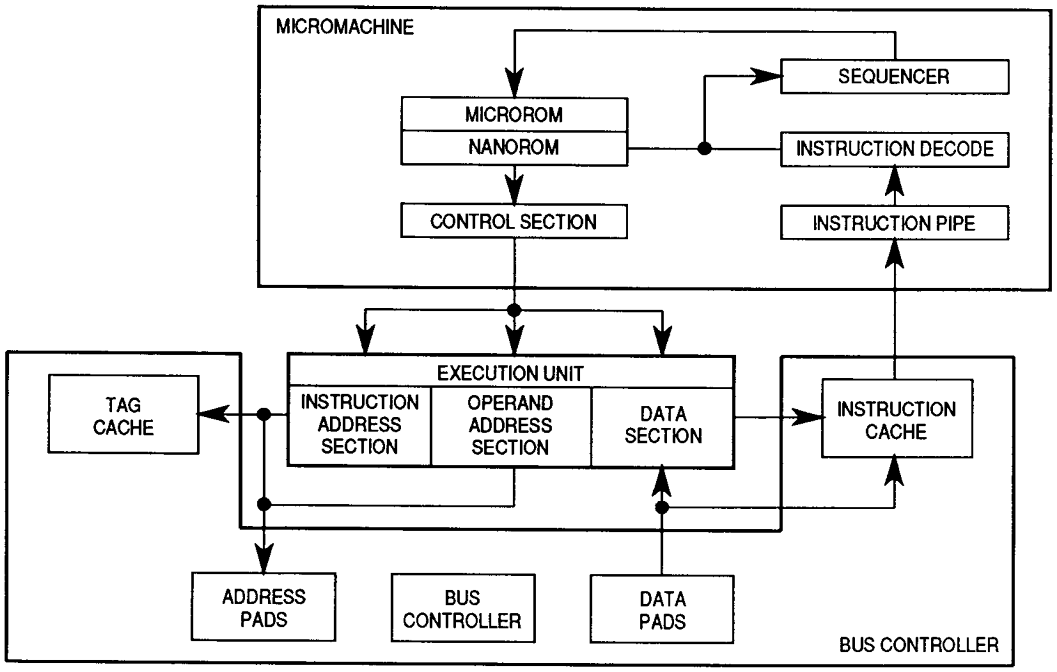


Figure 1. MC68EC020 Block Diagram

PROGRAMMING MODEL

As shown in the programming model (See Figure 2), the MC68EC020 has sixteen 32-bit general-purpose 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 are address registers; register A7 is the user stack pointer. These registers, as well as the interrupt and master stack pointers, can be used as software stack pointers or as 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.

The MC68EC020 provides a variety of useful control registers (see Figure 3), including a 32-bit program counter, two 32-bit supervisor stack pointers, a 32-bit vector base register, two 3-bit alternate function code registers, and two 32-bit cache control registers.

The vector base register is used to determine the base memory address location of the exception vector table. The displacement of an exception vector is added to the value in this register during the access to the vector table, supporting multiple vector tables; therefore, each process or task can properly manage exceptions independent of each other.

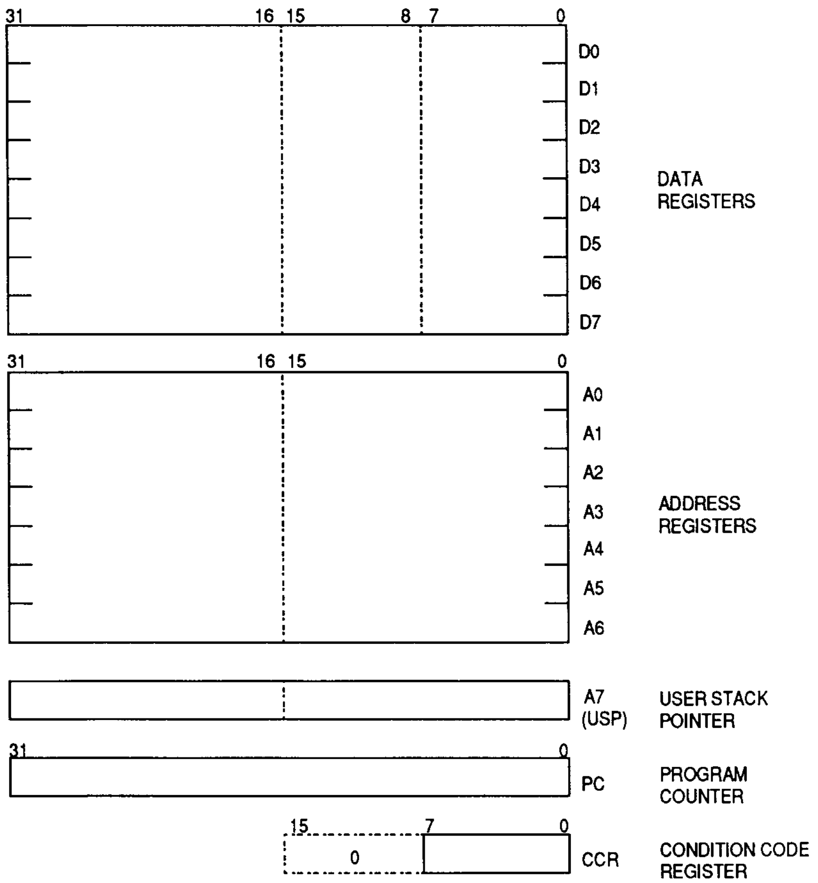


Figure 2. User Programming Model

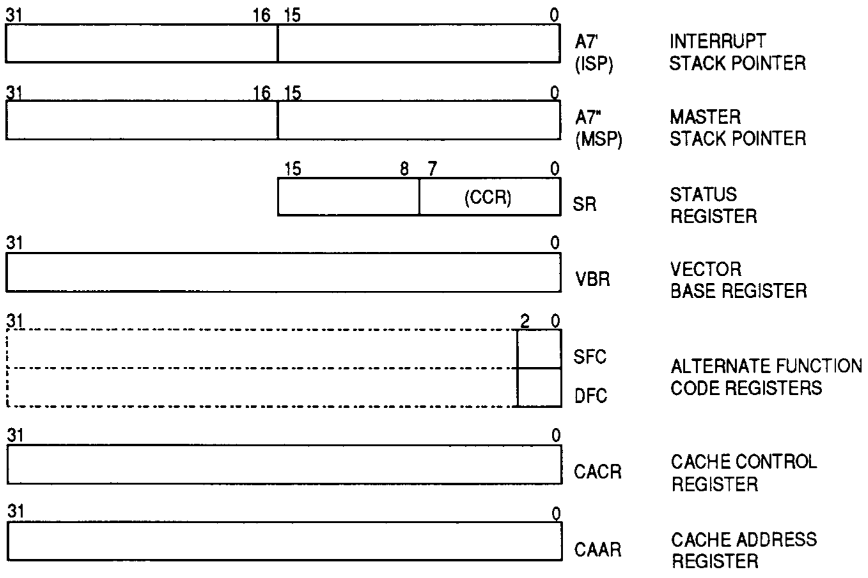


Figure 3. Supervisor Programming Model Supplement

All M68000 Family devices, including the MC68EC020, distinguish address spaces as supervisor/user and program/data. These four combinations are specified by the function code pins, FC0/FC1/FC2, during access to 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 3-bit 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 MC68EC020 has two registers for 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.

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

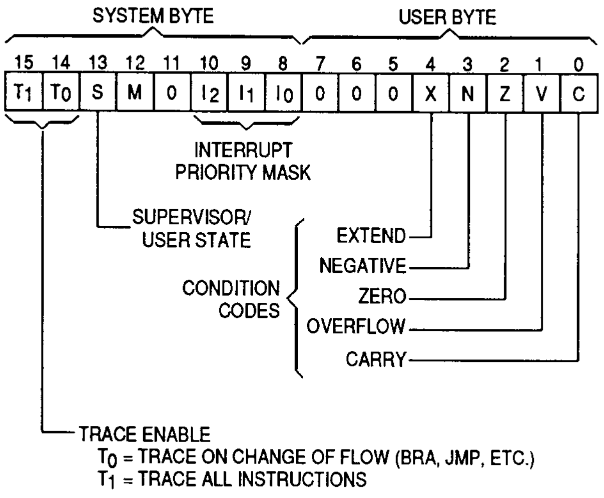


Figure 4. Status Register

The MC68EC020 and all processors and embedded controllers of the M68000 Family, support instruction tracing (via the T1 status bit in the MC68EC020). This tracing allows each executed instruction to be followed by a trap to a user-defined trace routine. The MC68EC020 also has the capability to trace only change-of-flow instructions (branch, jump, subroutine call and return, etc.) using the T0 status bit. These features are important for software program development and debug.

DATA TYPES AND ADDRESSING MODES

Seven basic data types are supported by the MC68EC020:

- Bits
- Bit Fields (String of consecutive bits, 1–32 bits long)
- BCD Digits (Packed: 2 digits/byte; Unpacked: 1 digit/byte)
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long-Word Integers (32 bits)
- 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/MC68882 floating-point coprocessor allows direct support of floating-point data types. The coprocessor mechanism supports specialized user-defined data types and functions.

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

- Register Direct
- Register Indirect
- Register Indirect with Index
- Memory Indirect
- Program Counter Indirect with Displacement
- Program Counter Indirect with Index
- Program Counter Memory Indirect
- Absolute
- Immediate

The register indirect addressing modes support postincrement, predecrement, offset, and index addressing. 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 the index and offset capabilities that can be required to support position-independent software. In addition to these addressing modes, the MC68EC020 provides data operand sizing and scaling.

Table 1. Addressing Modes

Addressing Modes	Syntax
Register Direct Addressing	
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 Data Addressing	
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
[ ]	=	Used as indirect address to long-word address.

INSTRUCTION SET OVERVIEW

The MC68EC020 instruction set is listed in Table 2. The MC68EC020 is source and object code compatible with the MC68020. The MC68EC020 is upward source and object code compatible with the M68000 Family in that it supports all instructions of MC68000, MC68008, MC68010, MC68020, MC68HC000, and MC68HC001 Family members. Additional instructions are provided by the MC68EC020 to support its advanced features. Special emphasis has been given to the 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 MC68EC020 to take advantage of the full 32-bit operation versus the 8-bit and 16-bit instructions on some earlier M68000 Family members. For detailed information on the MC68EC020 instruction set, refer to the *MC68EC020 User's Manual* (MC68EC020UM/AD) and the *M68000 Programmer's Reference Manual* (M68000PM/AD).

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
ANDI to CCR	Logical AND Immediate to Condition Code Register
ANDI to SR	Logical AND Immediate to Status Register
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
EORI to CCR	Logical Exclusive OR Immediate to Condition Code Register
EORI to SR	Logical Exclusive OR Immediate to Status Register
EXG	Exchange Registers
EXT, EXTB	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine

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 to CCR	Logical Inclusive OR Immediate to Condition Code Register
ORI to 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
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

BIT FIELD OPERATIONS

The MC68EC020 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. In addition to the bit manipulation instructions found on other M68000 Family products, the MC68EC020 has 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 MC68EC020 has 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

Some previous M68000 Family members have offered 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 MC68EC020 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

The system trap capabilities of the MC68EC020 have been expanded over what has been found in some earlier M68000 Family products. In addition to the current trap-on-overflow (TRAPV) instruction, a TRAPcc instruction is available that defines a format where any condition code is allowed to be the trapping condition. The TRAPcc instruction provides 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.

A breakpoint instruction (BKPT) is also available to support the program breakpoint function found in debug monitors and real-time in-circuit or hardware emulators. The operation of the BKPT instruction will be dependent on the actual system implementation. Execution of this instruction causes the MC68EC020 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 MC68EC020 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 MC68EC020 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 support multiple processor systems, the MC68EC020 has a compare and swap instruction (CAS). 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.

DYNAMIC BUS SIZING AND OPERAND TRANSFER

Though the MC68EC020 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 MC68EC020 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 MC68EC020 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, and 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 (DSACK) signals and occurs on a cycle-by-cycle basis. For example, if the MC68EC020 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 MC68EC020 latches all 32 bits of data and continues. If the port responds that it is 16 bits wide, the MC68EC020 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 MC68EC020 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 MC68EC020 always transfers the maximum amount of data on all bus cycles. It always assumes the port is 32 bits wide when beginning the bus cycle. In addition, the MC68EC020 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 MC68EC020 automatically runs the minimum bus cycles.

COPROCESSOR INTERFACE

The MC68EC020 is based on the MC68020 and shares that unit's ability to interface to coprocessors. Thus the MC68EC020 can be used with the MC68881/MC68882 floating-point coprocessors, enhancing the flexibility and applicability of the MC68EC020. Refer to MC68881UM/AD, *MC68881/MC68882 Floating-Point Coprocessor User's Manual* for more information.

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 signals, and reset requests. The interrupts are requests from peripheral devices for controller action. The bus error and reset pins are used for bus access control and controller 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 MC68EC020 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. Thus, it is highly desirable to have these instructions be accessible to the controller without having to perform any time-consuming fetches from memory. A cache is an on-chip memory that provides temporary storage for instructions as they are used so that they will be available again without the controller having to access external memory. Minimizing the bus access delays required for memory access and correspondingly freeing the bus for other devices can dramatically improve program performance.

One of the important benefits from the MC68EC020 embedded controller cache is the reduction in the controller's external bus activity. In a given MC68000 system, the controller will use approximately 80 to 90 percent (or greater) of the available bus bandwidth. In an MC68000 system with more than one bus master (such as a system with a controller and a DMA device, or a multiprocessor system) performance degradation can occur due to the demand being placed on the bus by many different devices. Reducing the load on the bus by limiting controller transactions effectively increases the available bus bandwidth for the other devices in the system, thereby improving total system performance.

A second major benefit of the on-chip instruction cache is the increase in effective CPU throughput as larger memory sizes or slower memories increase average access time. By placing a high-speed cache between the controller 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 MC68EC020 on-chip cache provides a substantial CPU performance increase or allows much slower and less expensive memories to be used for the same controller performance.

The throughput increase in the MC68EC020 is gained in two ways. First, the MC68EC020 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 MC68EC020 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 MC68EC020 is capable of executing several internal instructions (instructions that do not require the bus) while completing an operand access for another instruction.

The MC68EC020 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).

The MC68EC020 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, subsequent instruction prefetches can find the 16-bit instruction words in the cache and save the related bus cycle. However, even if the cache is not enabled, the bus controller will hold the full 32 bits of the previous fetch and will be used again, if it is the instruction required, saving another 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.

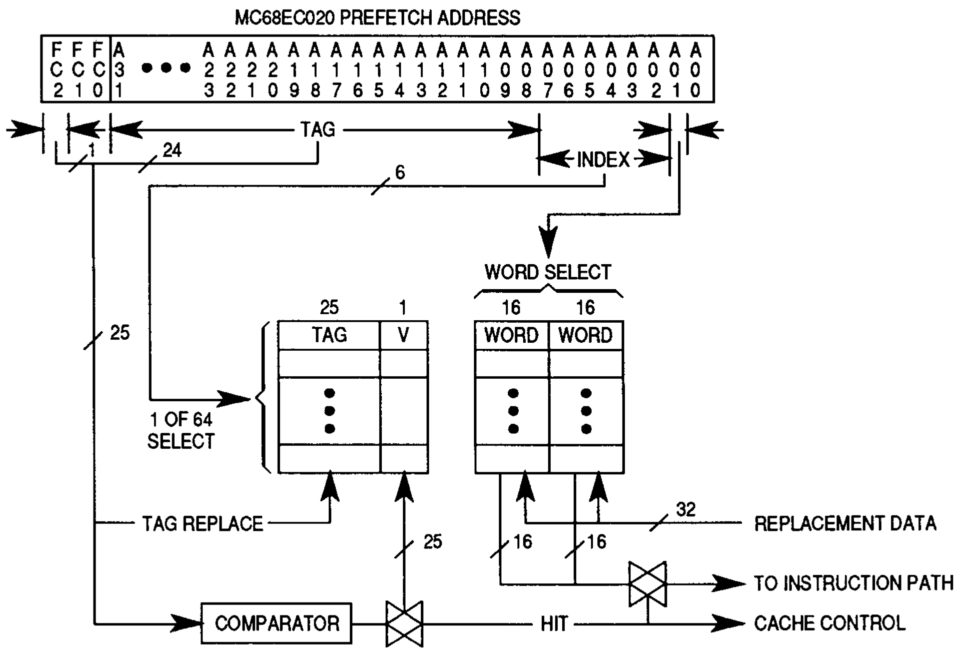

MC68EC020 PREFETCH ADDRESS
FC2 FC1 FC0 A31 ••• A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A09 A08 A07 A06 A05 A04 A03 A02 A01 A00
TAG
INDEX
1
24
6
25
WORD SELECT
25
1
TAG
V
16
16
WORD
WORD
1 OF 64 SELECT
TAG REPLACE
25
16
16
32
REPLACEMENT DATA
TO INSTRUCTION PATH
COMPARATOR
HIT
CACHE CONTROL

Figure 5. On-Chip Cache Organization

SIGNAL DESCRIPTION

The MC68EC020 is offered in a 100-lead plastic pin grid array (PGA) and a 100 pin quad flat pack (QFP). Figure 6 illustrates the functional signal groups, and Table 5 lists the signals and their function.

Table 5. Signal Index

Signal Name	Mnemonic	Function
Function Codes	FC0–FC2	A 3-bit function code used to identify the address space of each bus cycle.
Address Bus	A0–A23	A 24-bit address bus.
Data Bus	D0–D31	A 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.
Read/Write	R/$\overline{W}$	Defines the bus transfer as a controller 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 MC68EC020.
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.
Cache Disable	$\overline{CDIS}$	Dynamically disables the on-chip cache to assist emulator support.
Interrupt Priority Level	$\overline{IPL0}$–$\overline{IPL2}$	Provides an encoded interrupt level to the controller.
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.
Reset	$\overline{RESET}$	System reset.
Halt	$\overline{HALT}$	Indicates that the controller should suspended bus activity.
Bus Error	$\overline{BERR}$	Indicates that an erroneous bus operation is being attempted.
Clock	CLK	Clock input to the controller.
Power Supply	V_{CC}	+ 5 V ± 5% power supply.
Ground	GND	Ground connection.
No Connect	NC	Reserved by Motorola for future use. Do not connect externally.

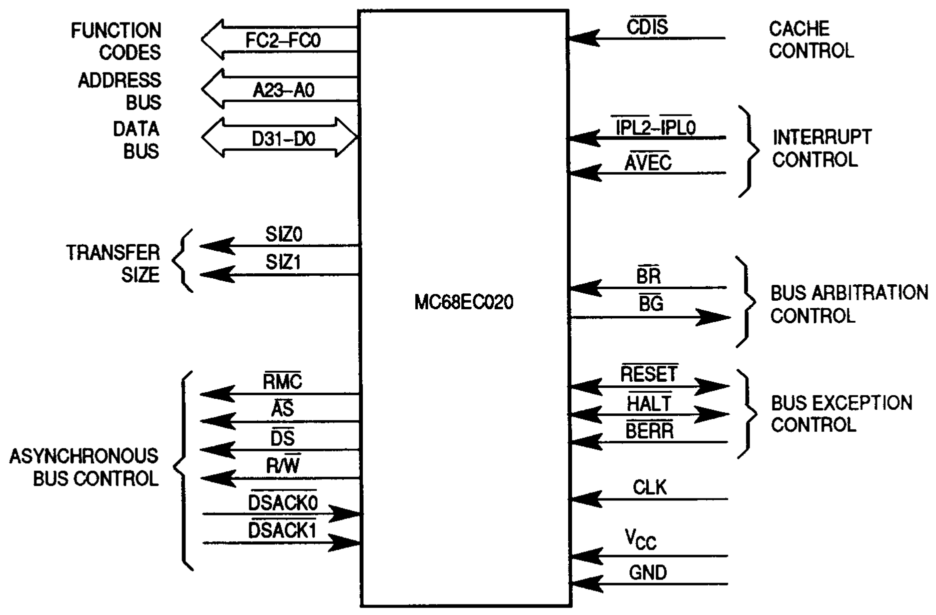


Figure 6. Functional Signal Groups

BUS ARBITRATION

The bus design of the MC68EC020 provides for a single bus master at any one time: either the controller or an external device. One or more of the external devices on the bus can have the capability of becoming bus master. Bus arbitration is the protocol by which an external device becomes bus master; the bus controller in the MC68EC020 manages the bus arbitration signals so that the controller has the lowest priority. External devices that need to obtain the bus must assert the bus arbitration signals in the sequences described in the following paragraphs. Systems having several devices that can become bus master require external circuitry to assign priorities to the devices, so that, when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first. The sequence of the protocol is as follows:

1. An external device asserts the bus request signal.
2. The controller asserts the bus grant signal to indicate that the bus will become available at the end of the current bus cycle.
3. The external device continues to assert the bus request signal throughout the duration of the external devices bus mastership.

Bus request ($\overline{BR}$) may be issued any time during a bus cycle or between cycles. Bus grant ($\overline{BG}$) is asserted in response to $\overline{BR}$; it is usually asserted as soon as $\overline{BR}$ has been synchronized and recognized, except when the MC68EC020 has made an internal decision to execute a bus cycle. Then, the assertion of $\overline{BG}$ is deferred until the bus cycle has begun. Additionally, $\overline{BG}$ is not asserted until the end of a read-modify-write operation (when $\overline{RMC}$ is negated) in response to a $\overline{BR}$ signal. When the requesting device receives $\overline{BG}$ and more than one external device can be bus master, the requesting device should begin whatever arbitration is required.

The external device asserts $\overline{\text{BR}}$ and maintains $\overline{\text{BR}}$ during the entire bus cycle (or cycles) for which it is bus master. The following conditions must be met for an external device to assume mastership of the bus through the normal bus arbitration procedure:

- It must have received $\overline{\text{BG}}$ through the arbitration process.
- $\overline{\text{AS}}$ must be negated, indicating that no bus cycle is in progress, and the external device must ensure that all appropriate controller signals have been placed in the high-impedance state (by observing specification #7.
- The termination signal ($\overline{\text{DSACKx}}$) for the most recent cycle must have become inactive, indicating that external devices are off the bus.
- No other bus master has claimed ownership of the bus.

Figure 7 is a flowchart of bus arbitration for a single device. Figure 8 is a timing diagram for the same operation. This technique allows processing of bus requests during data transfer cycles.

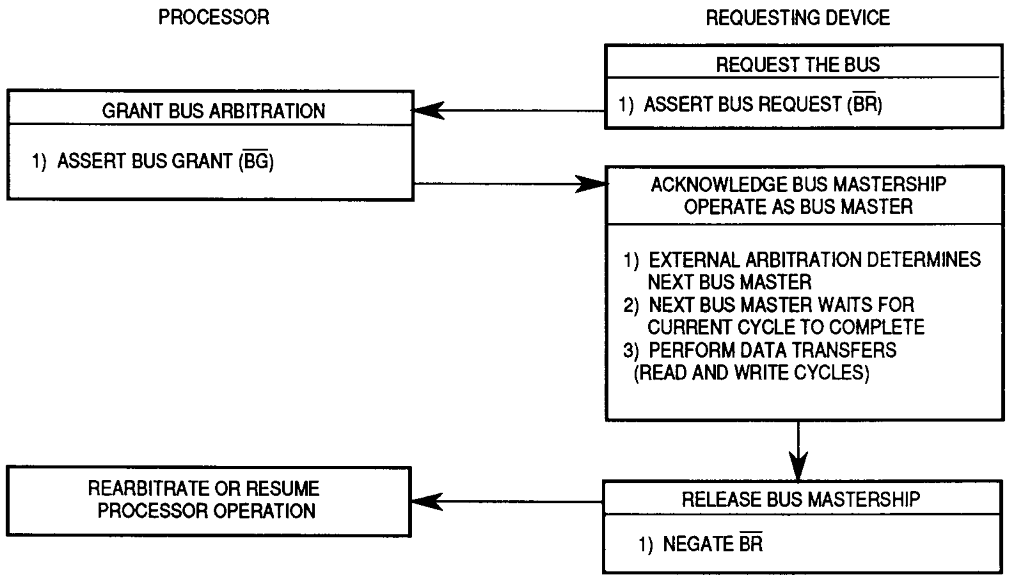

PROCESSOR
REQUESTING DEVICE
REQUEST THE BUS
1) ASSERT BUS REQUEST ($\overline{\text{BR}}$)
GRANT BUS ARBITRATION
1) ASSERT BUS GRANT ($\overline{\text{BG}}$)
ACKNOWLEDGE BUS MASTERSHIP OPERATE AS BUS MASTER
1) EXTERNAL ARBITRATION DETERMINES NEXT BUS MASTER
2) NEXT BUS MASTER WAITS FOR CURRENT CYCLE TO COMPLETE
3) PERFORM DATA TRANSFERS (READ AND WRITE CYCLES)
REARBITRATE OR RESUME PROCESSOR OPERATION
RELEASE BUS MASTERSHIP
1) NEGATE $\overline{\text{BR}}$

Figure 7. Bus Arbitration Flowchart for Single Request

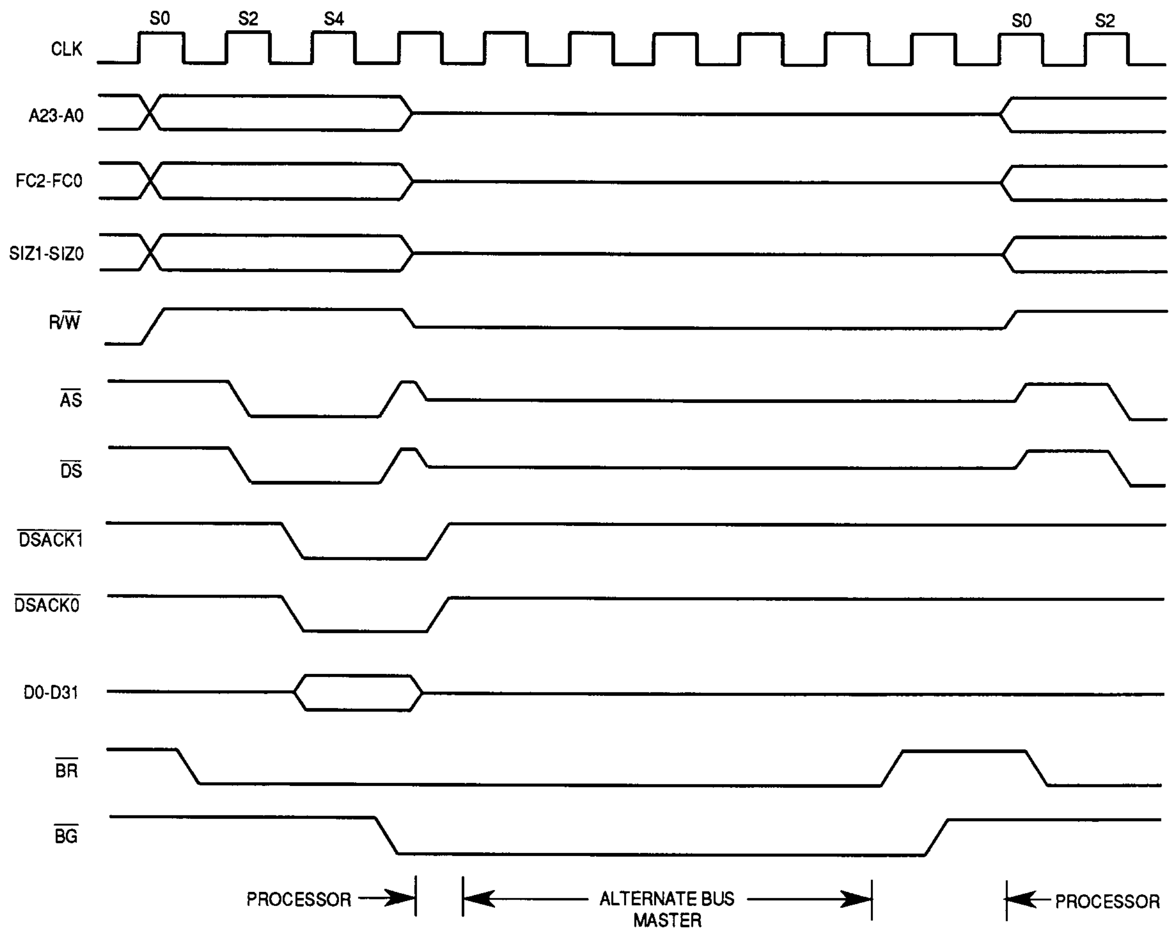


Figure 8. Bus Arbitration Operation Timing

Bus arbitration requests are recognized during normal processing, RESET assertion, HALT assertion, and even when the controller has halted due to a double bus fault.

BUS REQUEST

External devices capable of becoming bus masters request the bus by asserting BR. This can be a wire-ORed signal (although it need not be constructed from open-collector devices) that indicates to the controller that some external device requires control of the bus. The controller is effectively at a lower bus priority level than the external device and relinquishes the bus after it has completed the current bus cycle (if one has started).

BUS GRANT

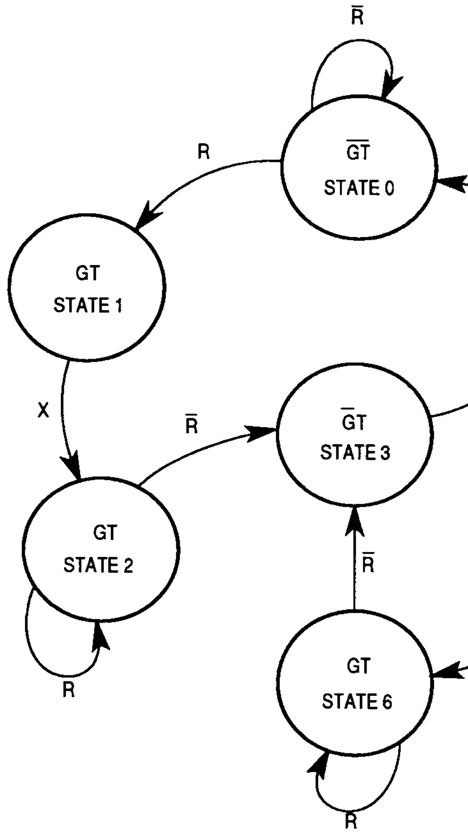
The controller asserts BG as soon as possible after receipt of BR. This is immediately following internal synchronization except during a read-modify-write cycle or following an internal decision to execute a bus cycle. During a read-modify-write cycle, the controller does not assert BG until the entire operation has completed. RMC is asserted to indicate that the bus is locked. In the case of an internal decision to execute another bus cycle, BG is deferred until the bus cycle has begun.

$\overline{BG}$ may be routed through a daisy-chained network or through a specific priority-encoded network. The controller allows any type of external arbitration that follows the protocol.

BUS ARBITRATION CONTROL

The bus arbitration control unit in the MC68EC020 is implemented with a finite state machine. As discussed previously, all asynchronous inputs to the MC68EC020 are internally synchronized in a maximum of two cycles of the controller clock.

As shown in Figure 9, the bus request input signal labeled R is an internally synchronized version of the $\overline{BR}$ signal. The bus grant output is labeled G, and the internal high-impedance control signal is labeled T. If T is true, the address, data, and control buses are placed in the high-impedance state after the next rising edge following the negation of $\overline{AS}$ and $\overline{RMC}$. All signals are shown in positive logic (active high), regardless of their true active voltage level.



R — BUS REQUEST
G — BUS GRANT
T — THREE-STATE CONTROL TO BUS CONTROL LOGIC
X — DON'T CARE

Figure 9. Bus Arbitration State Diagram

State changes occur on the next rising edge of the clock after the internal signal is valid. The $\overline{BG}$ signal transitions on the falling edge of the clock after a state is reached during which G changes. The bus control signals (controlled by T) are driven by the controller immediately following a state change when bus mastership is returned to the MC68EC020.

State 0, at the top center of the diagram, in which G and T are both negated, is the state of the bus arbiter while the controller is bus master. Request R keeps the arbiter in state 0 as long as it is negated. When a request R is received, both grant G and signal T are asserted (in state 1 at the top left). The next clock causes a change to state 2, at the lower left, in which G and T are held. The bus arbiter remains in that state until request R is negated. Then the arbiter changes to the center state, state 3, and negates grant G. The next clock takes the arbiter to state 4, at the upper right, in which grant G remains negated and signal T remains asserted. The arbiter returns to the original state, state 0, and negates signal T. This sequence of states follows the normal sequence of signals for relinquishing the bus to an external bus master. Other states apply to other possible sequences of R.

The MC68EC020 does not allow arbitration of the external bus during the read-modify-write sequence. For the duration of this sequence, the MC68EC020 ignores the $\overline{BR}$ input. If mastership of the MC68EC020 bus is required during a read-modify-write operation, the bus error signal must be used to abort the read-modify-write sequence. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in Figure 10.

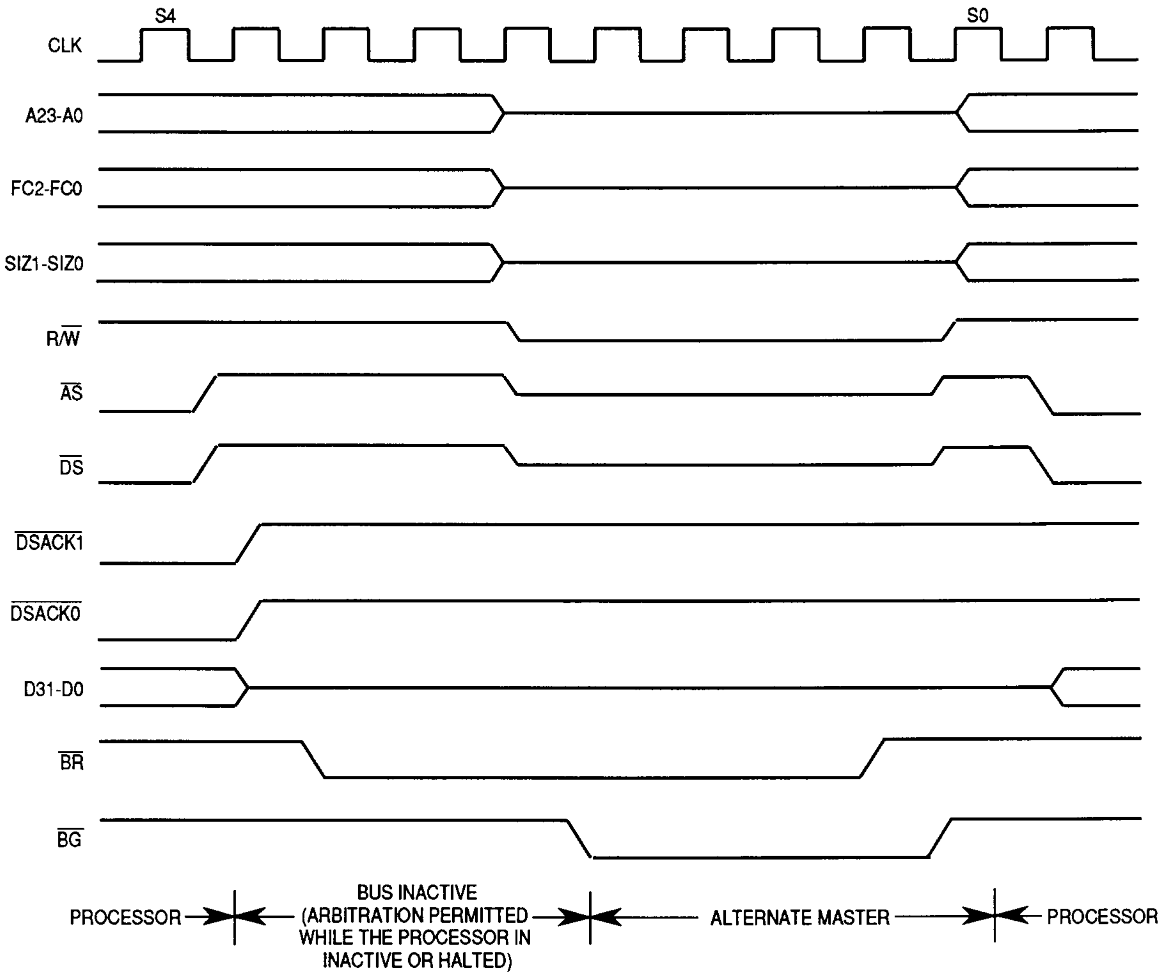


Figure 10. Bus Arbitration Operation (Bus Inactive)

The existing three-wire arbitration design ($\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{BGACK}}$) of some peripherals can be converted to the MC68EC020 two-wire arbitration with the addition of an AND gate and an OR gate. Figure 11 shows the combination of $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$ for a three-wire arbitration system to $\overline{\text{BR}}$ of the MC68EC020, and $\overline{\text{BR}}$ and $\overline{\text{BG}}$ for the MC68EC020 to $\overline{\text{BG}}$ of a three-wire arbitration system.

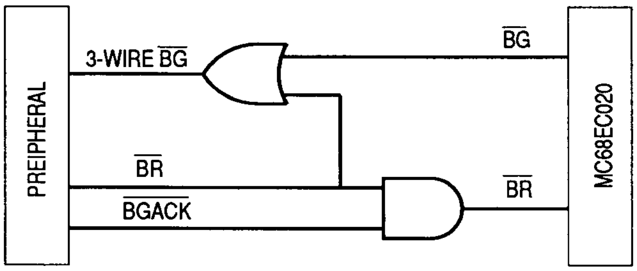


Figure 11. Interface for Three-Wire to Two-Wire Bus Arbitration

ELECTRICAL SPECIFICATIONS

MAXIMUM RATINGS

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	70	
Storage Temperature Range	T_{stg}	–55 to 150	°C

The device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, normal precautions should be taken to avoid application of voltages higher than maximum-rated voltages to these high-impedance circuits. Tying unused inputs to the appropriate logic voltage level (e.g., either GND or V_{CC}) enhances reliability of operation.

THERMAL CONSIDERATIONS

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

$$T_J = T_A + (P_D \bullet \theta_{JA}) \qquad (1)$$

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:

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

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

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

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:

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

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		
PPGA Package (RP Suffix)	32	TBD
QFP Package (FG Suffix)	55*	TBD

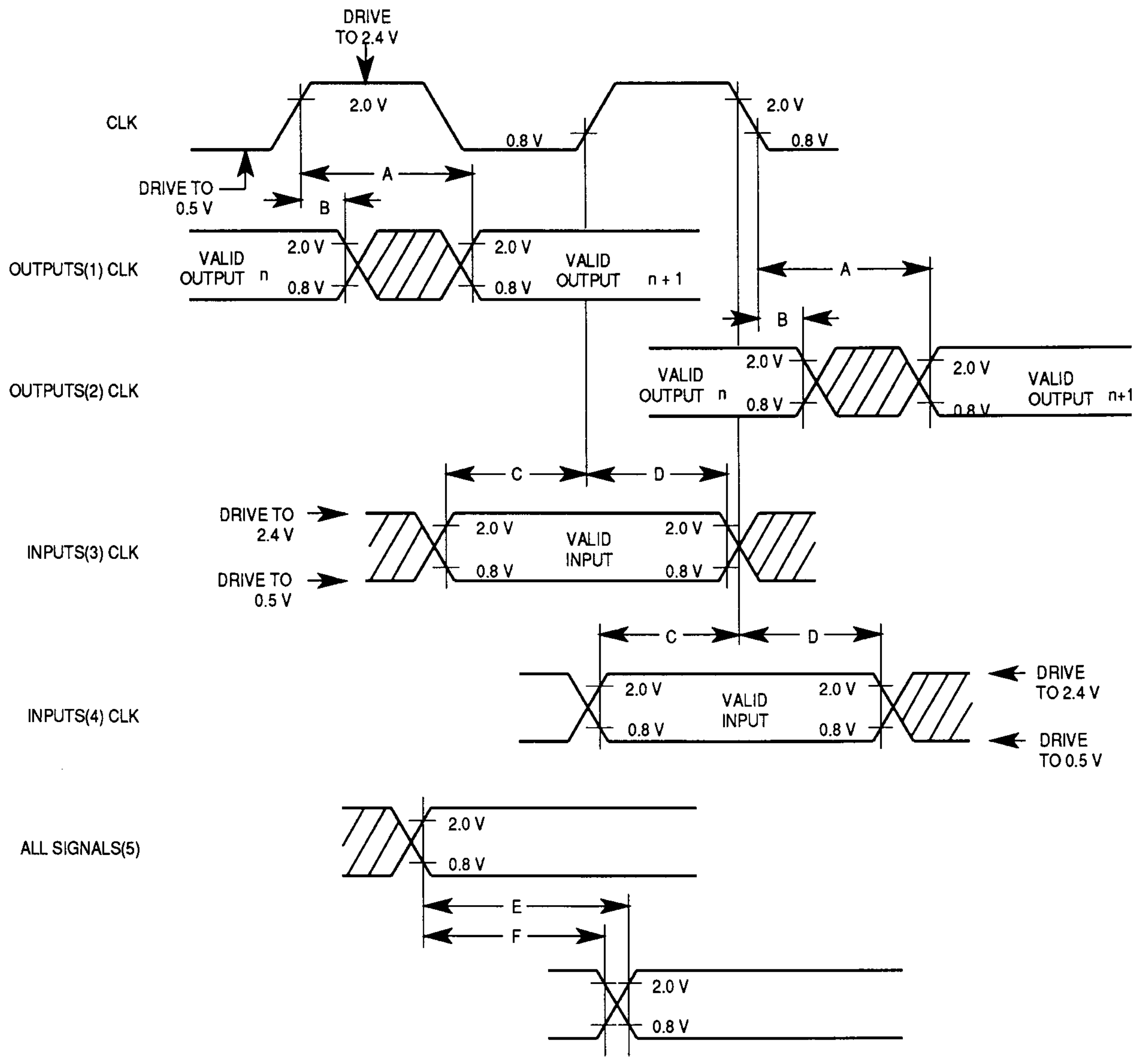
* Estimated

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 clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in Figure 12. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 12. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown in Figure 12. Inputs are specified with minimum setup and hold times, and are measured as shown. Finally, the measurement for signal-to-signal specifications is also shown.

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



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 12. Drive Levels and Test Points for AC Specifications

DC ELECTRICAL SPECIFICATIONS

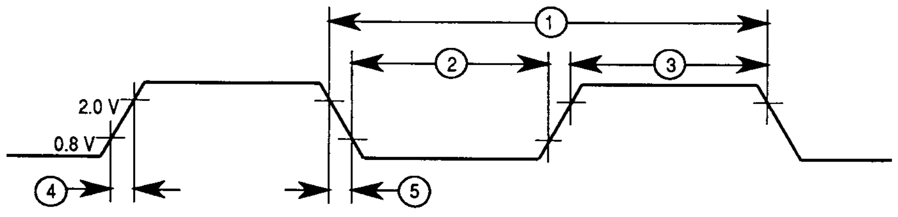
(V_{CC} = 5.0 Vdc ± 5%; GND = 0 Vdc; temperature in defined ranges)

Characteristics		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}	$\overline{BERR}$, $\overline{BR}$, CLK, $\overline{IPL0}$–$\overline{IPL2}$, $\overline{AVEC}$,$\overline{DSACK0}$, $\overline{DSACK1}$, $\overline{CDIS}$, $\overline{HALT}$, $\overline{RESET}$	I_{in}	–1.0 –20	1.0 20	µA
Hi-Z (Off-State) Leakage Current @ 2.4 V/0.5 V	A0–A23, $\overline{AS}$, $\overline{DS}$, D0–D31, FC0–FC2, R/$\overline{W}$, $\overline{RMC}$, SIZ0–SIZ1	I_{TSI}	–20	20	µA
Output High Voltage I_{OH} = 400 µA	A0–A23, $\overline{AS}$, $\overline{BG}$, D0–D31, $\overline{DS}$, R/$\overline{W}$, $\overline{RMC}$, SIZ0–SIZ1, FC0–FC2	V_{OH}	2.4	—	V
Output Low Voltage I_{OL} = 3.2 mA I_{OL} = 5.3 mA I_{OL} = 10.7 mA	A0–A23, FC0–FC2, SIZ0–SIZ1, $\overline{BG}$, D0–D31 $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, $\overline{RMC}$, $\overline{HALT}$, $\overline{RESET}$	V_{OL}	— — —	0.5 0.5 0.5	V
Power Dissipation (T_A = 0°C)	f = 25 MHz f = 16 MHz	P_{INT}	— —	1.6 1.3	W
Capacitance (see Note) V_{in} = 0 V, T_A = 25°C, f = 1 MHz		C_{in}	—	20	pF

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

AC ELECTRICAL SPECIFICATIONS — CLOCK INPUT (see Figure 13)

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



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

Figure 13. Clock Input Timing Diagram

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 5.0 Vdc ± 5%; GND = 0 Vdc; temperature in defined ranges; see Figures 14–16)

Num.	Characteristics	16.67MHz		25 MHz		Unit
		Min	Max	Min	Max	
6	Clock High to FC, Size, $\overline{RMC}$, Address Valid	0	30	0	25	ns
7	Clock High to Address, Data, FC, Size, $\overline{RMC}$ High Impedance	0	60	0	40	ns
8	Clock High to Address, FC, Size, $\overline{RMC}$ Invalid	0	—	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$ Asserted	3	30	3	18	ns
9A[1]	$\overline{AS}$ to $\overline{DS}$ Assertion Skew (Read)	–15	15	–10	10	ns
9B[7]	$\overline{AS}$ Asserted to $\overline{DS}$ Asserted (Write)	37	—	27	—	ns
11	Address, FC, Size, $\overline{RMC}$ Valid to $\overline{AS}$ (and $\overline{DS}$ Asserted, Read)	15	—	6	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$ Negated	0	30	0	15	ns
13	$\overline{AS}$, $\overline{DS}$ Negated to Address, FC, Size, $\overline{RMC}$ Invalid	15	—	10	—	ns
14	$\overline{AS}$ (and $\overline{DS}$ Read) Width Asserted	100	—	70	—	ns
14A	$\overline{DS}$ Width Asserted (Write)	40	—	30	—	ns
15	$\overline{AS}$, $\overline{DS}$ Width Negated	40	—	30	—	ns
15A[5]	$\overline{DS}$ Negated to $\overline{AS}$ Asserted	35	—	25	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	—	60	—	40	ns
17	$\overline{AS}$, $\overline{DS}$ Negated to R/$\overline{W}$ Invalid	15	—	10	—	ns
18	Clock High to R/$\overline{W}$ High	0	30	0	20	ns
20	Clock High to R/$\overline{W}$ Low	0	30	0	20	ns
21	R/$\overline{W}$ High to $\overline{AS}$ Asserted	15	—	5	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	75	—	50	—	ns
23	Clock High to Data-Out Valid	—	30	—	25	ns
25	$\overline{DS}$ Negated to Data-Out Invalid	15	—	5	—	ns
26	Data-Out Valid to $\overline{DS}$ Asserted (Write)	15	—	5	—	ns
27	Data-In Valid to Clock Low (Setup)	5	—	5	—	ns
27A	Late $\overline{BERR}$/$\overline{HALT}$ Asserted to Clock Low (Setup)	20	—	10	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACKx}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	0	80	0	50	ns
29	$\overline{AS}$, $\overline{DS}$ Negated to Data-In Invalid (Data-In Hold Time)	0	—	0	—	ns
29A	$\overline{AS}$, $\overline{DS}$ Negated to Data-In (High Impedance)	—	60	—	40	ns
31[2]	$\overline{DSACKx}$ Asserted to Data-In Valid	—	50	—	32	ns
31A[3]	$\overline{DSACKx}$ Asserted to $\overline{DSACKx}$ Valid (DSACK Asserted Skew)	—	15	—	10	ns
32	$\overline{RESET}$ Input Transition Time	—	1.5	—	1.5	Clks
33	Clock Low to $\overline{BG}$ Asserted	0	30	0	20	ns
34	Clock Low to $\overline{BG}$ Negated	0	30	0	20	Clks
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)	1.5	3.5	1.5	3.5	Clks
39	$\overline{BG}$ Width Negated	90	—	60	—	ns

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (CONTINUED)

Num.	Characteristics	16.67MHz		25 MHz		Unit
		Min	Max	Min	Max	
39A	$\overline{\mathrm{BG}}$ Width Asserted	90	—	60	—	ns
46	R/$\overline{\mathrm{W}}$ Width Valid (Write or Read)	150	—	100	—	ns
47A	Asynchronous Input Setup Time	5	—	5	—	ns
47B	Asynchronous Input Hold Time	15	—	10	—	ns
48[4]	$\overline{\mathrm{DSACKx}}$ Asserted to $\overline{\mathrm{BERR}}$, $\overline{\mathrm{HALT}}$ Asserted	—	30	—	18	ns
53	Data-Out Hold from Clock High	0	—	0	—	ns
55	R/$\overline{\mathrm{W}}$ Valid to Data Bus Impedance Change	30	—	20	—	ns
56	$\overline{\mathrm{RESET}}$ Pulse Width (Reset Instruction)	512	—	512	—	Clks
57	$\overline{\mathrm{BERR}}$ Negated to $\overline{\mathrm{HALT}}$ Negated (Rerun)	0	—	0	—	ns
59	$\overline{\mathrm{BG}}$ Negated to Bus Driven	1	—	1	—	Clks

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 $\overline{\mathrm{DSACKx}}$ low to data setup time (#31) and $\overline{\mathrm{DSACKx}}$ low to $\overline{\mathrm{BERR}}$ 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 $\overline{\mathrm{BERR}}$ must only satisfy the late $\overline{\mathrm{BERR}}$ low to clock low setup time (#27A) for the following clock cycle.
3. This parameter specifies the maximum allowable skew between $\overline{\mathrm{DSACK0}}$ to $\overline{\mathrm{DSACK1}}$ asserted or $\overline{\mathrm{DSACK1}}$ to $\overline{\mathrm{DSACK0}}$ asserted; specification #47A must be met by $\overline{\mathrm{DSACK0}}$ or $\overline{\mathrm{DSACK1}}$.
4. This specification applies to the first ($\overline{\mathrm{DSACK0}}$ or $\overline{\mathrm{DSACK1}}$) $\overline{\mathrm{DSACKx}}$ signal asserted. In the absence of $\overline{\mathrm{DSACKx}}$, $\overline{\mathrm{BERR}}$ is an asynchronous input using the asynchronous input setup time (#47A).
5. This specification guarantees operation with the MC68881/MC68882, which specifies a minimum time for $\overline{\mathrm{DS}}$ negated to $\overline{\mathrm{AS}}$ asserted (specification #13A in the *MC68881/MC68882 User's Manual)*. Without this specification, incorrect interpretation of specifications #9A and #15 would indicate that the MC68EC020 does not meet the MC68881/MC68882 requirements.
6. This specification allows system designers to qualify the $\overline{\mathrm{CS}}$ signal of an MC68881/MC68882 with $\overline{\mathrm{AS}}$ (allowing 7 ns for a gate delay) and still meet the $\overline{\mathrm{CS}}$ to $\overline{\mathrm{DS}}$ setup time requirement (specification 8B of the *MC68881/MC68882 User's Manual).*

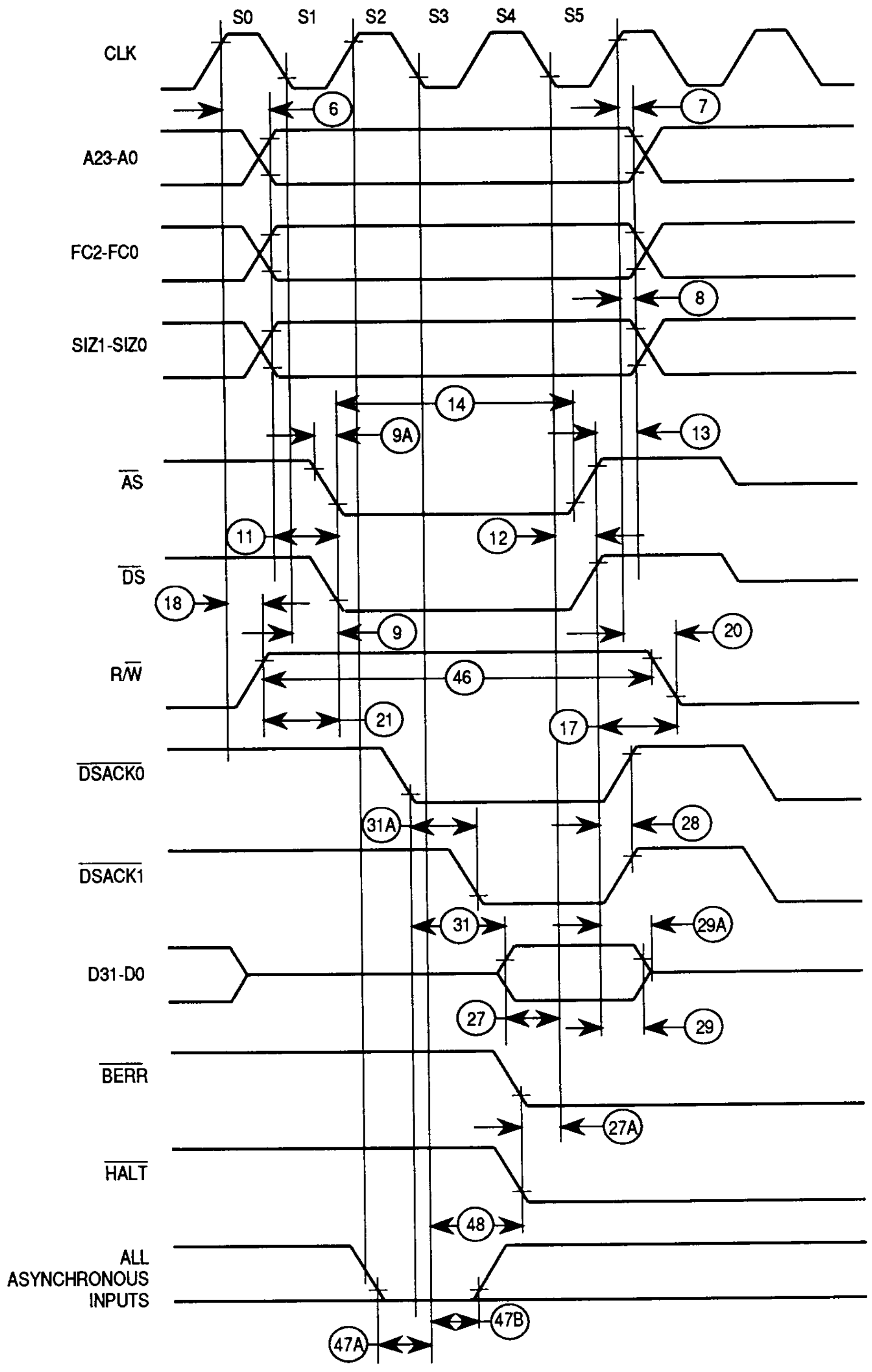


Figure 14. Read Cycle Timing Diagram

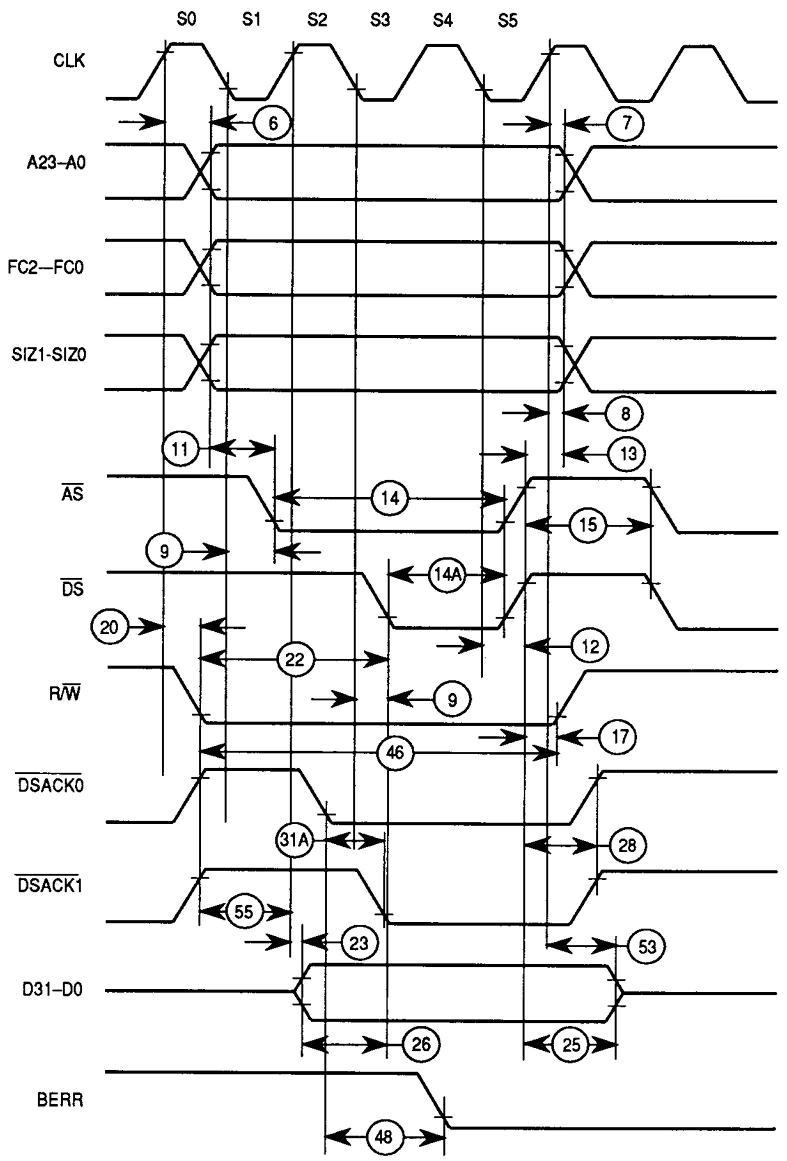
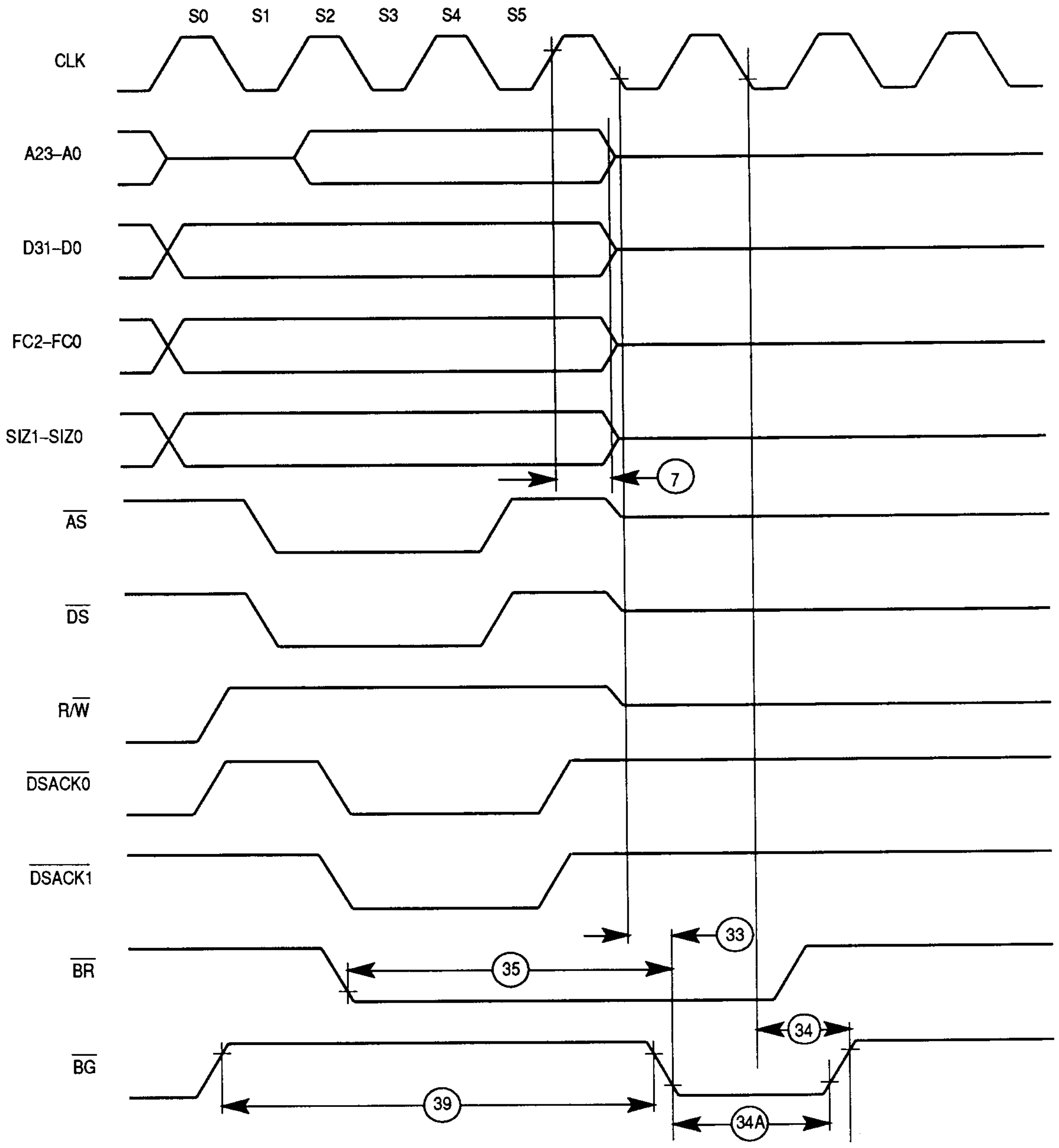


Figure 15. Write Cycle Timing Diagram

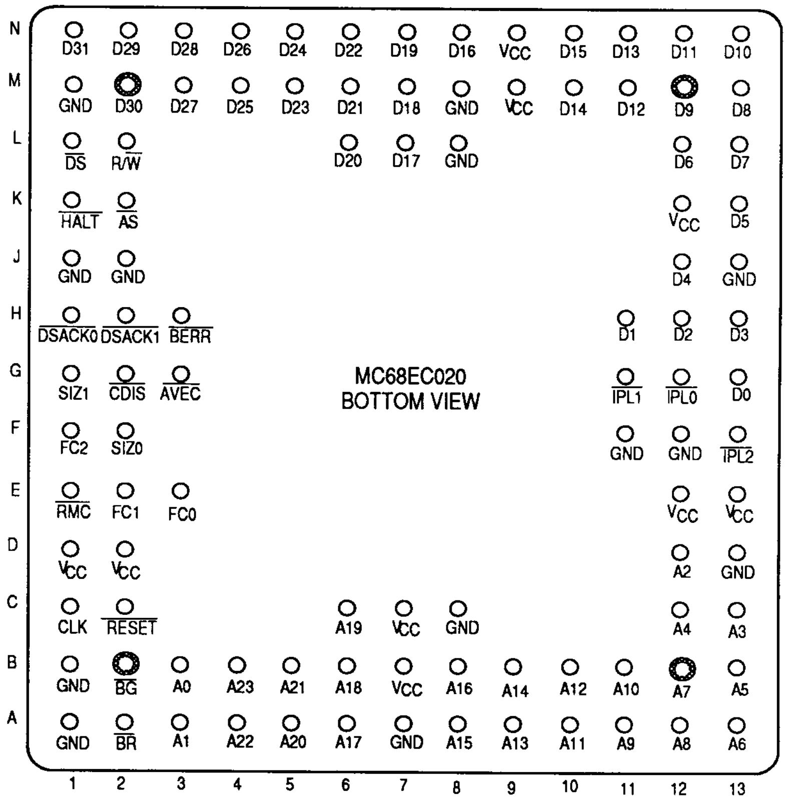


NOTE: Timing measurements are referenced to and from a low voltage of 0.8 V and high voltage of 2.0 V, unless otherwise noted. The voltage swing through this range should start outside and pass through the range so that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 16. Bus Arbitration Timing Diagram

MECHANICAL DATA

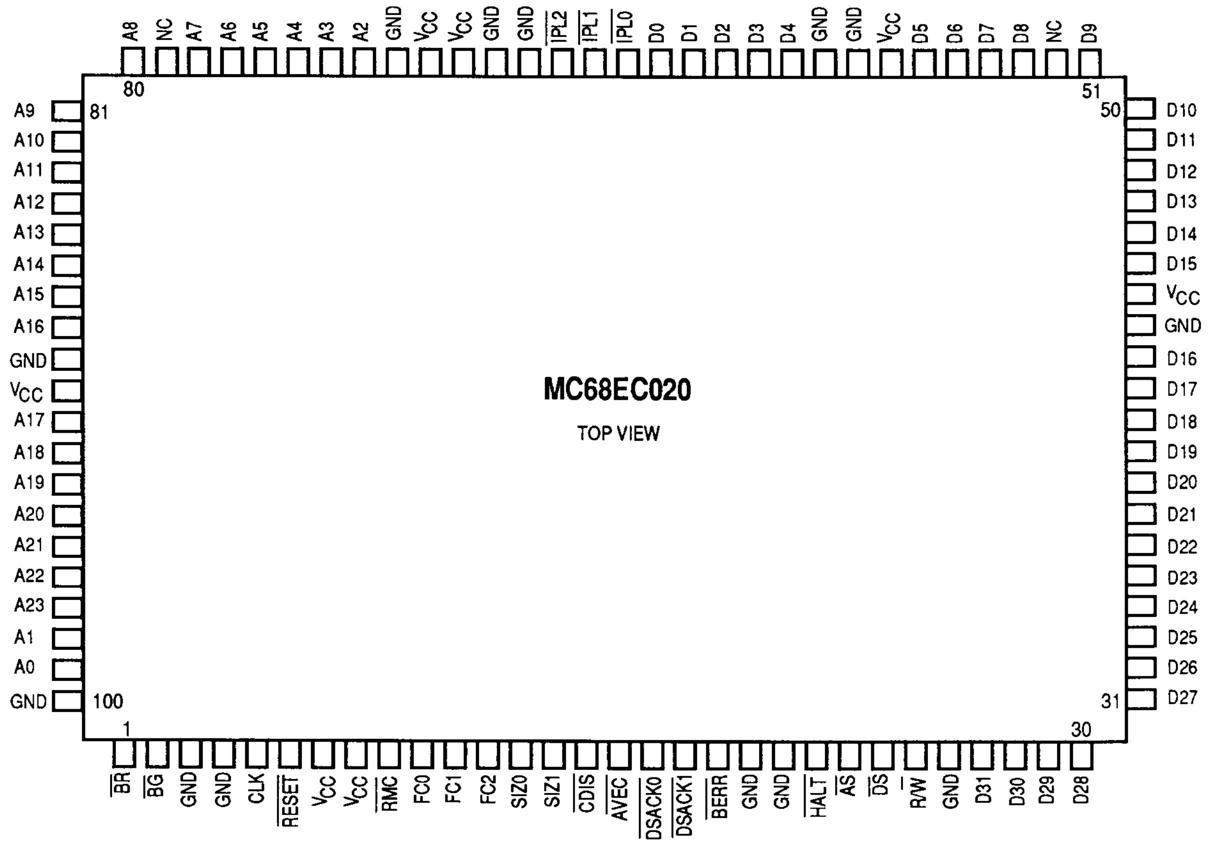
PIN ASSIGNMENTS — PIN GRID ARRAY (RP SUFFIX)


N D31 D29 D28 D26 D24 D22 D19 D16 VCC D15 D13 D11 D10
M GND D30 D27 D25 D23 D21 D18 GND VCC D14 D12 D9 D8
L DS R/W D20 D17 GND D6 D7
K HALT AS VCC D5
J GND GND D4 GND
H DSACK0 DSACK1 BERR D1 D2 D3
G SIZ1 CDIS AVEC IPL1 IPL0 D0
MC68EC020
BOTTOM VIEW
F FC2 SIZ0 GND GND IPL2
E RMC FC1 FC0 VCC VCC
D VCC VCC A2 GND
C CLK RESET A19 VCC GND A4 A3
B GND BG A0 A23 A21 A18 VCC A16 A14 A12 A10 A7 A5
A GND BR A1 A22 A20 A17 GND A15 A13 A11 A9 A8 A6
1 2 3 4 5 6 7 8 9 10 11 12 13

The V_{CC} and GND pins are separated into four groups to provide individual power supply connections for the address bus buffers, the data bus buffers, and all other output buffers and internal logic. It is recommended that all pins be connected to power and ground as indicated.

Group	V_{CC}	Ground
Address Bus	B7,C7	A1, A7, C8, D13
Data Bus	K12, M9, N9	J13, L8, M1, M8
Logic	D1, D2, E12, E13	F11, F12, J1, J2
Clock	—	B1

PIN ASSIGNMENTS — QUAD FLAT PACK (FG SUFFIX)

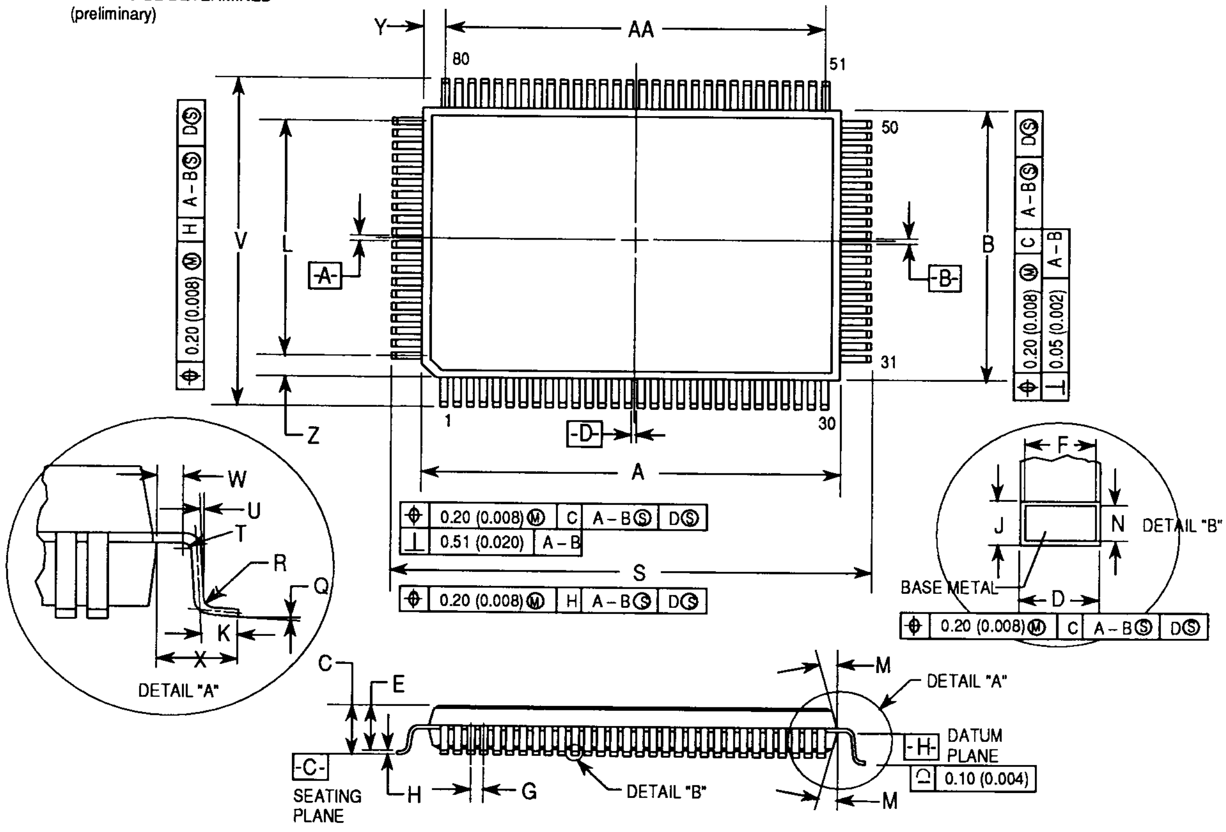


The V_{CC} and GND pins are separated into four groups to provide individual power supply connections for the address bus buffers, the data bus buffers, and all other output buffers and internal logic. It is recommended that all pins be connected to power and ground as indicated. NC pins are reserved by Motorola for future use and should have no external connection.

Group	V_{CC}	Ground
Address Bus	90	72, 89, 100
Data Bus	44,57	26,43,58,59
Logic	7,8,70,71	3,20,21,68,69
Clock	—	4

PACKAGE DIMENSIONS

FG SUFFIX
CASE TO BE DETERMINED
(preliminary)

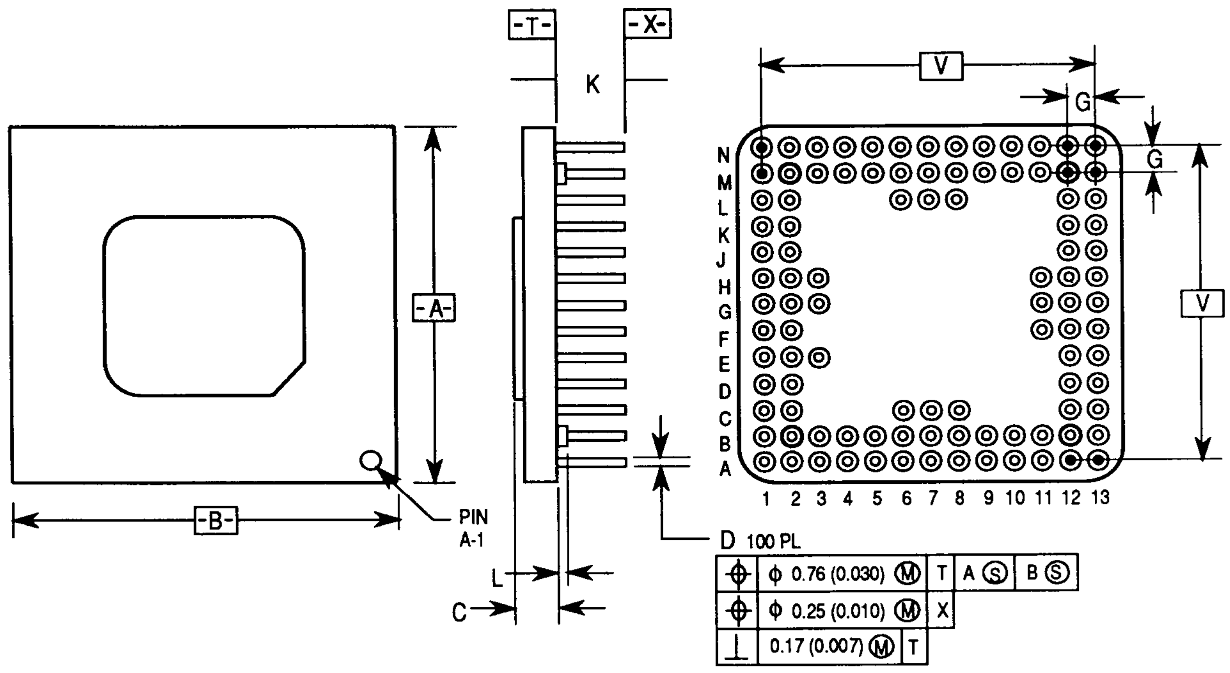


DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	19.90	20.0	.785	.789
B	13.90	14.0	0.549	0.553
C	2.8	3.1	.110	.122
D	0.22	0.38		
E	2.6	2.9	0.101	0.113
G	0.65 BSC		0.256 BSC	
H	.10	.36	.004	.014
J	.26	.38	.010	.015
K	.66	.94	.026	0.037
L	13.9	14.0	.549	.553
Q	0°	8°	0°	8°
S	23.6	24.2	.931	.951
V	17.7	18.2	.695	.715
X	1.60 REF			
AA	18.8 REF		0.742 REF	

NOTES:

1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. DATUM PLANE -H- LOCATED AT MOLD PARTING LINE AND COINCIDENT WITH LEAD, WHERE LEAD EXITS PLASTIC BODY AT BOTTOM OF PARTING LINE.
3. DATUMS A-B AND -D- TO BE DETERMINED WHERE CENTERLINE BETWEEN LEADS EXITS PLASTIC BODY AT DATUM PLANE -H-.
4. TO BE DETERMINED AT SEATING PLANE -C- .
5. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS .010" / 0.254mm ON D1 AND E1 DIMENSIONS.
6. "N" IS THE TOTAL NUMBER OF TERMINALS.
7. THESE DIMENSIONS TO BE DETERMINED AT DATUM PLANE -H- .
8. PACKAGE TOP DIMENSIONS ARE SMALLER THAN BOTTOM DIMENSIONS BY 0.20<.008> mm.
9. CONTROLLING DIMENSION: MILLIMETER. INCHES ARE IN "( )".
10. MAXIMUM ALLOWABLE DIE THICKNESS TO BE ASSEMBLED IN THIS PACKAGE FAMILY IS 0.51<.020> mm.

RP SUFFIX
CASE TO BE DETERMINED
(Preliminary)



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	34.04	35.05	1.340	1.380
B	34.04	35.05	1.340	1.380
C	2.92	3.18	0.115	0.135
D	0.44	0.55	0.017	0.022
G	2.54 BSC		0.100 BSC	
K	4.32	4.83	0.170	0.190
L	1.02	1.52	0.040	0.060
V	30.48 BSC		1.200 BSC	

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M,1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION D INCLUDES LEAD FINISH.