MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC68882

Technical Summary

HCMOS Enhanced Floating-Point Coprocessor

The MC68882 floating-point coprocessor fully implements the *IEEE Standard for Binary Floating-Point Arithmetic (754)* for use with the Motorola M68000 Family of microprocessors. An upgrade of the MC68881, it is pin and software compatible with an optimized microprocessor unit (MPU) interface providing in excess of 1.5 times the performance of the MC68881. It is implemented using VLSI technology to give systems designers the highest possible functionality in a physically small device.

5

Intended primarily for use as a coprocessor to the MC68020/MC68030 32-bit MPU, the MC68882 provides a logical extension to the main MPU integer data processing capabilities. This extension is achieved by providing a very high-performance floating-point arithmetic unit and a set of floating-point data registers that are analogous to the use of the integer data registers. The MC68882 instruction set is a natural extension of all earlier members of the M68000 Family, and it supports all addressing modes of the host MPU. Due to the flexible bus interface of the M68000 Family, the MC68882 can be used with any of the M68000 MPU devices and as a peripheral to non-M68000 processors.

The major features of the MC68882 are as follows:

- Eight general-purpose floating-point data registers, each supporting a full 80-bit extended-precision real data format (a 64-bit mantissa plus a sign bit and a 15-bit signed exponent)
- A 67-bit arithmetic unit to allow very fast calculations, with intermediate precision greater than the extended-precision format
- A 67-bit barrel shifter for high-speed shifting operations (for normalizing, etc.)

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

- Special-purpose hardware for high-speed conversion of binary real memory operands to and from the internal extended format
- Reduced coprocessor interface overhead to increase throughput
- 46 instructions, including 35 arithmetic operations
- Full conformation to the IEEE 754 standard, including all requirements and suggestions
- Support of functions not defined by the IEEE 754 standard, including a full set of trigonometric and transcendental functions
- Seven data types: byte, word and long-word integers; single-, double-, and extended-precision real numbers; and packed binary-coded decimal (BCD) string real numbers
- 22 constants available in the on-chip ROM, including π, e, and powers of 10
- Virtual memory/machine operations
- Efficient mechanisms for procedure calls, context switches, and interrupt handling
- Concurrent instruction execution with the main processor
- Concurrent instruction execution of multiple floating-point instructions
- Use with any host processor on an 8-, 16-, or 32-bit data bus

THE COPROCESSOR CONCEPT

The MC68882 functions as a coprocessor in systems where the MC68020 or MC68030 is the main processor. It functions as a peripheral processor in systems where the main processor is the MC68000, MC68008, or MC68010.

The MC68882 utilizes the M68000 Family coprocessor interface to provide a logical extension of the MC68020/MC68030 registers and instruction set in a manner transparent to the programmer. The programmer perceives the MPU/MC68882 execution model as if both devices are implemented on one chip.

A fundamental goal of the M68000 Family coprocessor interface is to provide the programmer with an execution model based upon sequential instruction execution by the MC68020/MC68030 and the MC68882. For optimum performance, however, the coprocessor interface allows concurrent operations in the MC68882 with respect to the MC68020/MC68030 whenever possible. To simplify the programmer's model, the coprocessor interface is designed to emulate, as closely as possible, nonconcurrent operation between the MC68020/MC68030 and the MC68882.

The MC68882 is a non-DMA type coprocessor using a subset of the general-purpose coprocessor interface supported by the MC68020/MC68030. Features of the interface implemented in the MC68882 are as follows:

- The main processor(s) and MC68882 communicate via standard M68000 bus cycles.
- The main processor(s) and MC68882 communications are not dependent upon the architecture of the individual devices (e.g., instruction pipes or caches, addressing modes).
- The main processor(s) and MC68882 may operate at different clock speeds.
- MC68882 instructions utilize all addressing modes provided by the main processor.
- All effective addresses are calculated by the main processor at the request of the coprocessor.
- All data transfers are performed by the main processor at the request of the MC68882.
- Overlapped (concurrent) instruction execution enhances throughput while maintaining the programmer's model of sequential instruction execution.
- Coprocessor detection of exceptions requiring a trap to be taken are serviced by the main processor at the request of the MC68882.
- Support of virtual memory/machine systems is provided via the FSAVE and FRESTORE instructions.
- Up to eight coprocessors may reside in a system simultaneously. Multiple coprocessors of the same type are allowed.
- Systems may use software emulation of the MC68882 without reassembling or relinking user software.

HARDWARE OVERVIEW

The MC68882 is a high-performance floating-point device designed to interface with the MC68020/MC68030 as a coprocessor. This device fully supports the MC68020/MC68030 virtual machine architecture and is implemented in HCMOS, Motorola's low-power small-geometry process. This process allows CMOS and HMOS (high-density NMOS) gates to be combined on the same device. CMOS structures are used where speed and low power are required, and HMOS structures are used where minimum silicon area is desired. Using this technology increases speed while using low-power consumption, yet still confines the MC68881 to a reasonably small die size.

The MC68882 can also be used as a peripheral processor in systems where the MC68020/MC68030 is not the main processor (e.g., MC68000, MC68008, MC68010). The configuration of the MC68882 as a peripheral processor or coprocessor may be completely transparent to user software (i.e., the same object code may be executed in either configuration).

The architecture of the MC68882 appears to the user as a logical extension of the M68000 Family architecture. Because of the coupling of the coprocessor interface, the MC68020/MC68030 programmer can view the MC68882 registers as though the registers were resident in the MC68020/MC68030. Thus, a MC68020/MC68030 and an MC68882 function as one processor with eight integer data registers, eight address registers, and eight floating-point data registers supporting seven floating-point and integer data types.

The MC68882 programming model, shown in Figures 1–6, consists of the following features:

- Eight 80-bit floating-point data registers (FP0–FP7). These registers are analogous to the integer data registers (D0–D7) and are completely general purpose (i.e., any instruction can use any register).
- A 32-bit control register contains enable bits for each class of exception trap and mode bits for setting the user-selectable rounding and precision modes.
- A 32-bit status register contains floating-point condition codes, quotient bits, and exception status information.
- A 32-bit instruction address register contains the main processor memory address of the last floating-point instruction that was executed. This address is used in exception handling to locate the instruction that caused the exception.

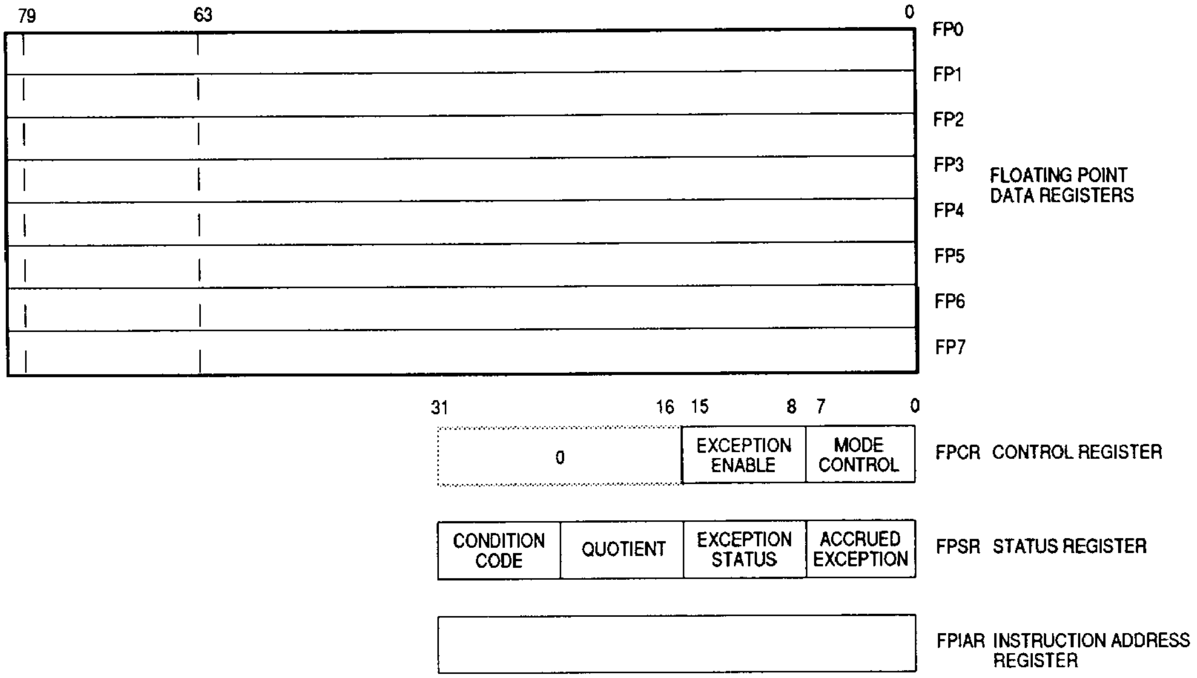


Figure 1. Programming Model

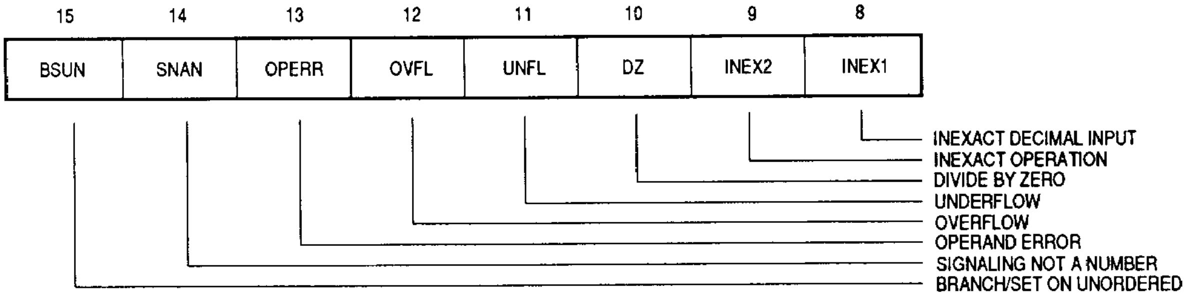


Figure 2. Exception Status/Enable Byte

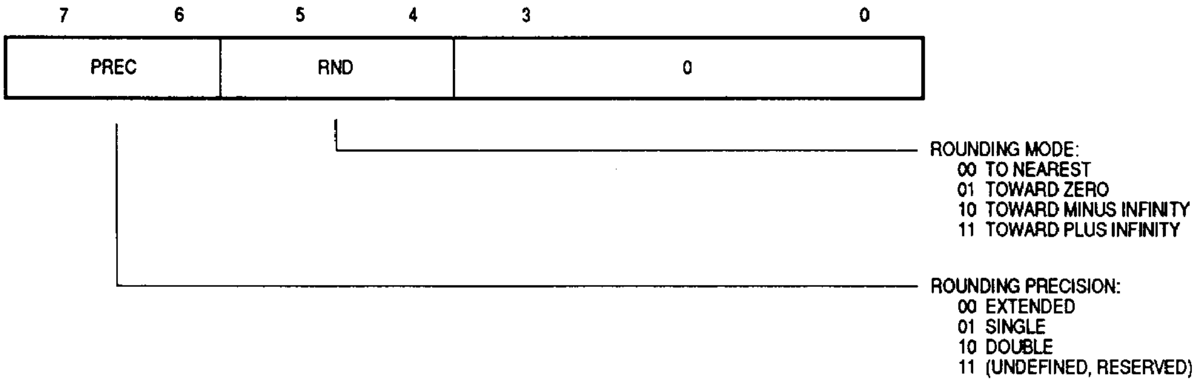


Figure 3. Mode Control Byte

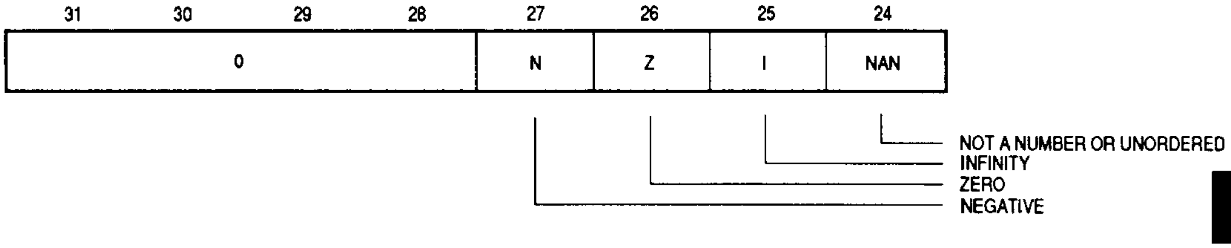


Figure 4. Condition Code Byte

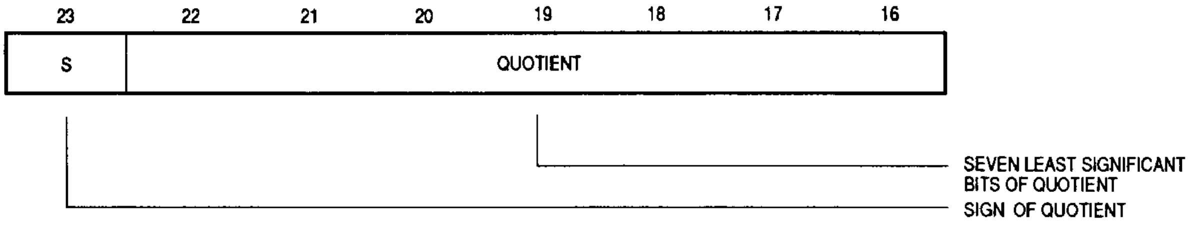


Figure 5. Quotient Byte

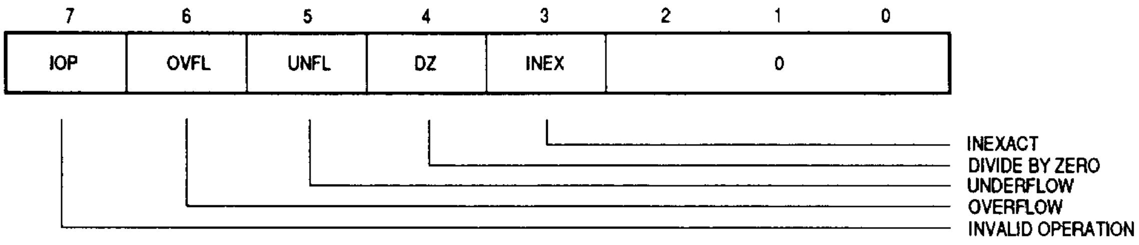


Figure 6. Accrued Exception Byte

The connection between the MC68020/MC68030 and the MC68882 is a simple extension of the M68000 bus interface. The MC68882 is connected as a coprocessor to the MC68020/MC68030, and the selection of the MC68882 is based on a chip select, which is decoded from the MC68020/MC68030 function codes and address bus. Figure 7 illustrates the MPU/coprocessor configuration.

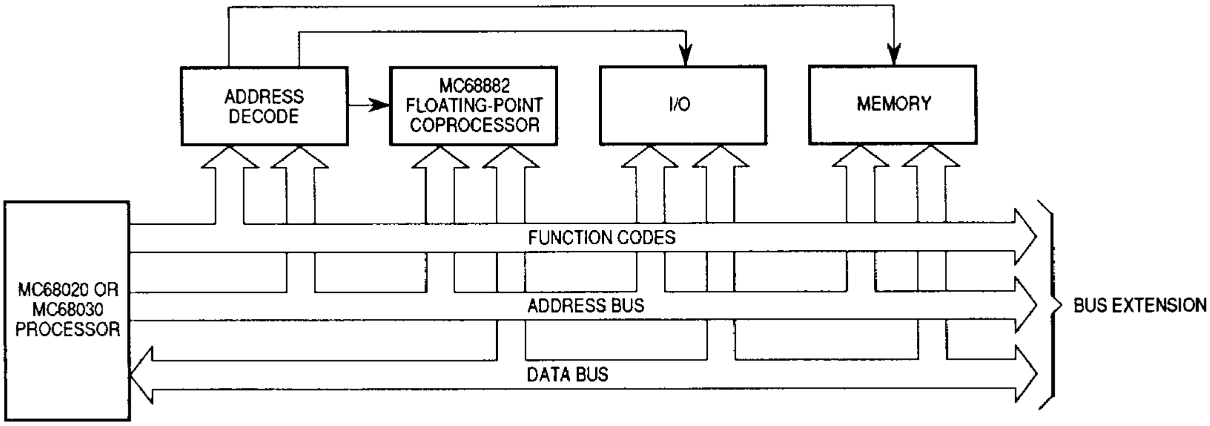


Figure 7. Typical Coprocessor Configuration

As shown in Figure 8, the MC68882 is internally divided into three processing elements: the bus interface unit (BIU), the conversion unit (CU), and the arithmetic processing unit (APU). The BIU communicates with the MC68020/MC68030, the CU performs data conversion for binary real data formats, and the APU executes all MC68882 instructions.

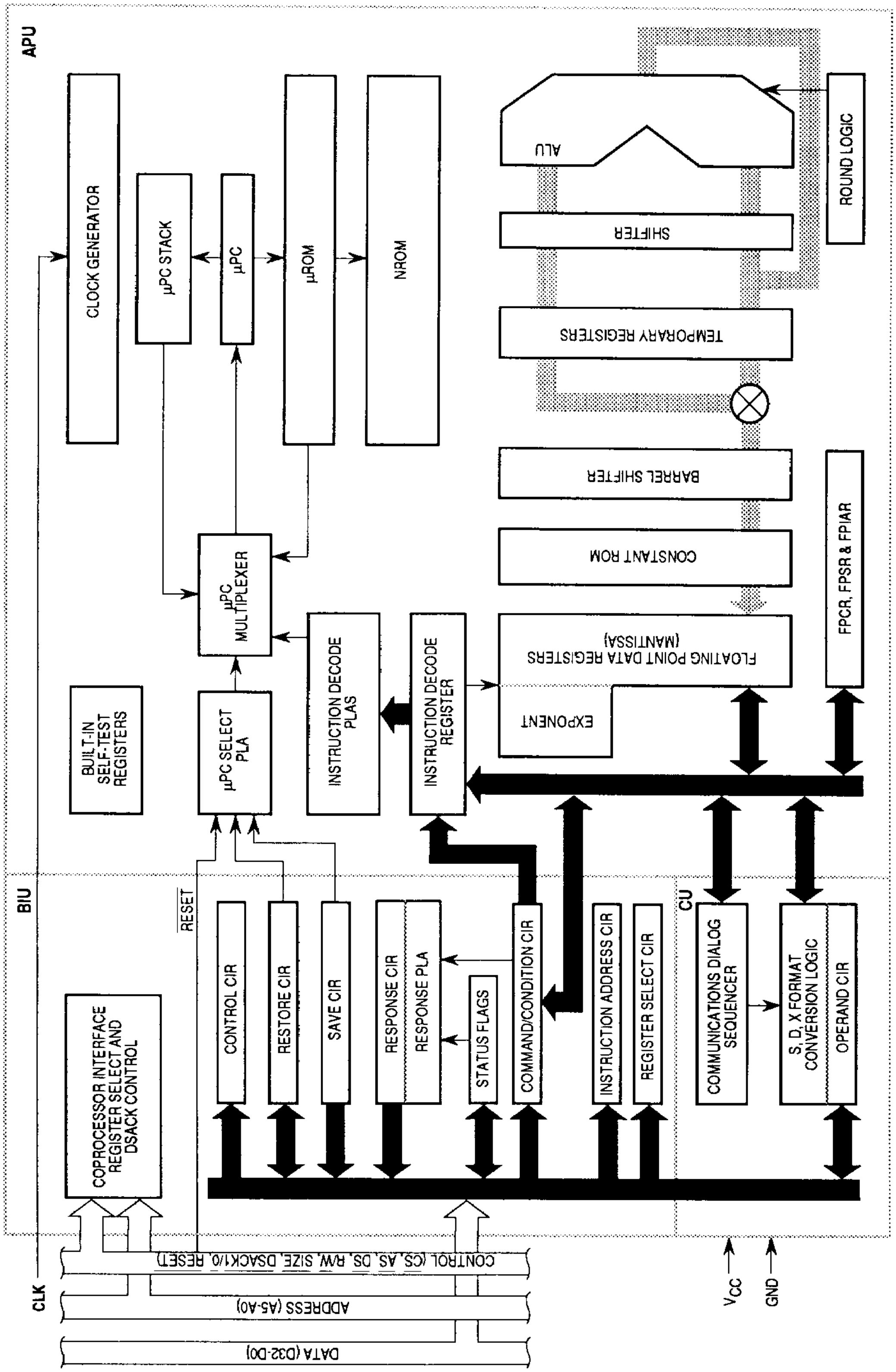


Figure 8. Simplified Block Diagram

The BIU contains the coprocessor interface registers (CIRs). In addition, the register select and DSACK timing control logic and the status flags used to monitor the status of communications with the main processor are contained in the BIU.

The CU contains special-purpose hardware that performs data format conversions between binary real data formats to and from the internal extended format. The CU relieves the APU of a significant work load and allows the MC68882 to execute data movement and preparation functions concurrently with arithmetic and transcendental calculations.

The eight, 80-bit, floating-point data registers (FP0–FP7) and the 32-bit control, status, and instruction address registers (FPCR, FPSR and FPIAR) are located in the APU. In addition, the APU contains a high-speed 67-bit arithmetic unit used for both mantissa and exponent calculations, a barrel shifter that can shift from 1–67 bits in one machine cycle, and ROM constants (for use by the internal algorithms or user programs). The control section of the APU contains the clock generator, a two-level microcode sequencer, the microcode ROM, and self-test circuitry. The built-in self-test capabilities of the MC68882 enhance reliability and ease manufacturing requirements; however, these diagnostic functions are not accessible outside the special test environment supported by VLSI test equipment.

BUS INTERFACE UNIT

All communications between the MC68020/MC68030 and the MC68882 occur via standard M68000 Family bus transfers. The MC68882 is designed to operate on 8-, 16-, or 32-bit data buses.

The MC68882 contains a number of CIRs that are addressed in the same manner as memory by the main processor. The M68000 Family coprocessor interface is implemented via a protocol of reading and writing to these registers by the main processor. The MC68020/MC68030 implements this general-purpose coprocessor interface protocol in hardware and microcode.

When the MC68020/MC68030 detects a general-type MC68882 instruction, the MC68020/MC68030 writes the instruction to the memory-mapped command CIR and reads the response CIR. In this response, the BIU encodes requests for any additional action required of the MC68020/MC68030 on behalf of the MC68882. For example, the response may request that the MC68020/MC68030 fetch an operand from the evaluated effective address and transfer the operand to the operand CIR. Once the MC68020/MC68030 fulfills the coprocessor request(s), the MC68020/MC68030 is free to fetch and execute subsequent instructions.

The only difference between a coprocessor bus transfer and any other bus transfer is that the MC68020/MC68030 issues a CPU address space function code during the cycle function codes are generated by the M68000 Family processors to identify eight separate address spaces. Thus, the memory-mapped CIRs do not infringe upon instruction or data address spaces. The MC68020/MC68030 places a coprocessor ID field from the coprocessor instruction onto three of the upper address lines during coprocessor accesses. This ID and the CPU address space function code are decoded to select one of eight coprocessors in the system.

Since the coprocessor interface protocol is based solely on bus transfers, the protocol is easily emulated by software when the MC68882 is used as a peripheral with any processor capable of memory-mapped I/O over an M68000-style bus. When used as a peripheral processor with the MC68008, MC68000, or MC68010, all MC68882 instructions are trapped by the main processor to an exception handler at execution time. Thus, the software emulation of the coprocessor interface protocol can be totally transparent to the user. The MC68882 can provide a performance option for MC68000-based designs by changing the main processor to the MC68020/MC68030. The software migrates without change to the next-generation equipment using the MC68020/MC68030.

Since the bus is asynchronous, the MC68882 need not run at the same clock speed as the main processor; therefore, total system performance can be customized. Thus, the floating-point performance can be selected to meet particular price/performance specifications, running the MC68882 at slower (or faster) clock speeds than the MPU clock.

COPROCESSOR INTERFACE

The M68000 Family coprocessor interface is an integral part of the MC68882 and MC68020/MC68030 designs. The interface partitions MPU and coprocessor operations so that the MC68020/MC68030 does not have to completely decode coprocessor instructions and the MC68882 does not have to duplicate main processor functions (such as effective address evaluation).

This partitioning provides an orthogonal extension of the instruction set by permitting MC68882 instructions to utilize all MC68020/MC68030 addressing modes and to generate execution time exception traps. Thus, from the programmer's view, the MPU and coprocessor appear to be integrated onto a single chip. While the execution of most MC68882 instructions may be overlapped with the execution of MC68020/MC68030 instructions, concurrency is completely transparent to the programmer. The MC68020/MC68030 single-step and program flow (trace) modes are fully supported by the MC68882 and the M68000 Family coprocessor interface.

Although the M68000 Family coprocessor interface permits coprocessors to be bus masters, the MC68882 is never a bus master. The MC68882 requests that the MC68020/MC68030 fetch all operands and store all results. In this manner, the MC68020/MC68030 32-bit data bus provides high-speed transfer of floating-point operands and results while simplifying the design of the MC68882.

Since the coprocessor interface is based solely upon bus cycles (to and from CPU space) and the MC68882 is never a bus master, the MC68882 can be placed on either the logical or physical side of the system memory management unit. Since the memory management unit of the MC68030 is on-chip, the MC68882 is always on the physical side of the memory management unit in an MC68030 system.

The virtual machine architecture of the MC68020/MC68030 is supported by the coprocessor interface and the MC68882 through the FSAVE and FRESTORE instructions. If the MC68020/MC68030 detects a page fault and/or a task timeout, the MC68020/MC68030 can force the MC68882 to stop whatever operation is in progress at any time and save the MC68882 internal state in memory. During the execution of a floating-point instruction, the MC68882 can stop at predetermined points as well as at the completion of the instruction.

The size of the saved internal state of the MC68882 is dependent upon the APU state at the time FSAVE is executed. If the MC68882 is in the reset state when FSAVE is received, only one word of state is transferred to memory, which can be examined by the operating system to determine that the coprocessor programmer's model is empty. If the coprocessor is in the idle state when FSAVE is received, only a few words of internal state are transferred to memory. If executing an instruction in the busy state, it may be necessary to save the entire internal state of the machine. Instructions completing execution in less time than it takes to save the larger state in mid-instruction are allowed to complete execution and then save the idle state. Thus, the size of the saved internal state is kept to a minimum. The ability to utilize several internal state sizes greatly reduces the average context switching time.

The FRESTORE instruction permits reloading of an internal state that was saved earlier and continues any operation that was previously suspended. FRESTORE of the null state frame re-establishes default register values, a function identical to the MC68882 hardware reset.

MC68882 PERFORMANCE ENHANCEMENTS

The high performance of the MC68882 is the result of executing multiple floating-point instructions concurrently. Concurrency utilizes the APU more efficiently by decreasing its idle time.

When the MC68882 receives an instruction, the BIU and the CU can initiate the instruction, fetch the necessary operands, and convert them to the internal extended format even though the APU is busy completing execution of a previous instruction. Although the MC68881 can only instruct the main processor to wait if the APU is busy, the MC68882 CU can proceed with the next instruction. When the APU is finally ready to perform the calculation, it can do so immediately without incurring delay due to data movement and preparation functions.

Another factor in obtaining increased performance in the MC68882 is the optimized FMOVE instructions for BCD real data formats. These FMOVE instructions execute twice as fast as the corresponding FMOVE instructions of the MC68881. The FMOVE instructions are also potentially fully concurrent and can be completely executed during the execution of a previous instruction.

The MC68882 also has a more optimized coprocessor interface than the MC68881. If an arithmetic instruction has data formats of single, double, or extended precision, the dialogs are designed to increase the potential overlap with subsequent instructions. This overlap can significantly decrease the effective instruction execution time.

OPERAND DATA FORMATS

The MC68882 supports the following data formats:

Byte Integer (B)
Word Integer (W)
Long-Word Integer (L)
Single-Precision Real (S)
Double-Precision Real (D)
Extended-Precision Real (X)
Packed BCD String Real (P)

The capital letters contained in parentheses denote suffixes added to instructions in the assembly language source that specify the data format to be used.

INTEGER DATA FORMATS

The three integer data formats (byte, word, and long word) are the standard twos-complement data formats supported in the M68000 Family architecture. Whenever an integer is used in a floating-point operation, the integer is automatically converted by the MC68882 to an extended-precision floating-point number before being used. For example, to add an integer constant of five to the number contained in floating-point data register 3 (FP3), the following instruction can be used:

```
FADD.W #5,FP3
```

(The Motorola assembler syntax "#" is used to
denote immediate addressing.)

The ability to effectively use integers in floating-point operations saves user memory since an integer representation of a number, if representable, is usually smaller than the equivalent floating-point representation.

FLOATING-POINT DATA FORMATS

The floating-point data formats, single precision (32 bits) and double precision (64 bits), are defined by the IEEE 754 standard. These data formats are the main floating-point formats and should be used for most calculations involving real numbers. Table 1 lists the exponent and mantissa size for single, double, and extended precision. The exponent is biased, and the mantissa is in sign and magnitude form. Since single and double precision require normalized numbers, the most significant bit of the mantissa is implied as a one and is not included, thus giving one extra bit of precision.

Table 1. Exponent and Mantissa Sizes

Data Format	Exponent Bits	Mantissa Bits	Bias
Single	8	23(+1)	127
Double	11	52(+1)	1023
Extended	15	64	16383

The extended-precision data format is also in conformance with the IEEE 754 standard, but the standard does not specify this format to the bit level as it does for single and double precision. The memory format on the MC68882 consists of 96 bits (three long words). Only 80 bits are actually used; the other 16 bits are for future expandability and for long-word alignment of floating-point data structures. Extended format has a 15-bit exponent, a 64-bit mantissa, and a 1-bit mantissa sign.

Extended-precision numbers are intended for use as temporary variables, intermediate values, or in areas where extra precision is needed. For example, a compiler might select extended-precision arithmetic for evaluation of the right side of an equation with mixed-sized data and then convert the answer to the data type on the left side of the equation. Extended-precision data will not be stored in large arrays due to the amount of memory required by each value.

PACKED BCD STRING DATA FORMAT

This data format allows packed BCD strings to be transferred to and from the MC68882. The strings consist of a 3-digit base-10 exponent and a 17-digit base-10 mantissa. Both the exponent and mantissa have a separate sign bit. All digits are packed BCD; an entire string fits in 96 bits (three long words). Like all data formats, when packed BCD strings are supplied to the MC68882, the strings are automatically converted to extended-precision real values, allowing packed BCD numbers to be used as inputs to any operation. For example,

```
FADD.P   #-6.023E+24,FP5
```

BCD numbers can be output from the MC68882 in a format readily used for printing by a program generated by a high-level language compiler. For example,

```
FMOVE.P   FP3,BUFFER{#-5}
```

This instruction converts the floating-point data register 3 (FP3) contents into a packed BCD string with five digits to the right of the decimal point (FORTRAN F format).

DATA FORMAT SUMMARY

All data formats previously described are supported orthogonally by all arithmetic and transcendental operations and by all appropriate MC68020/MC68030 addressing modes. For example, all of the following instructions are legal:

```
FADD.B   #0,FP0
FADD.W   D2,FP3
FADD.L   BIGINT,FP7
FADD.S   #3.14159,FP5
FADD.D   (SP)+,FP6
FADD.X   [(TEMP_PTR,A7)],FP3
FADD.P   #1.23E25,FP0
```

Most on-chip calculations are performed in the extended-precision format, and the eight floating-point data registers always contain extended-precision values. All operands used are converted to extended precision by the MC68882 before a specific operation is performed, and all results are in extended precision. The use of extended precision ensures maximum accuracy without sacrificing performance.

Refer to Figure 9 for a summary of the memory formats for the seven data formats supported by the MC68882.

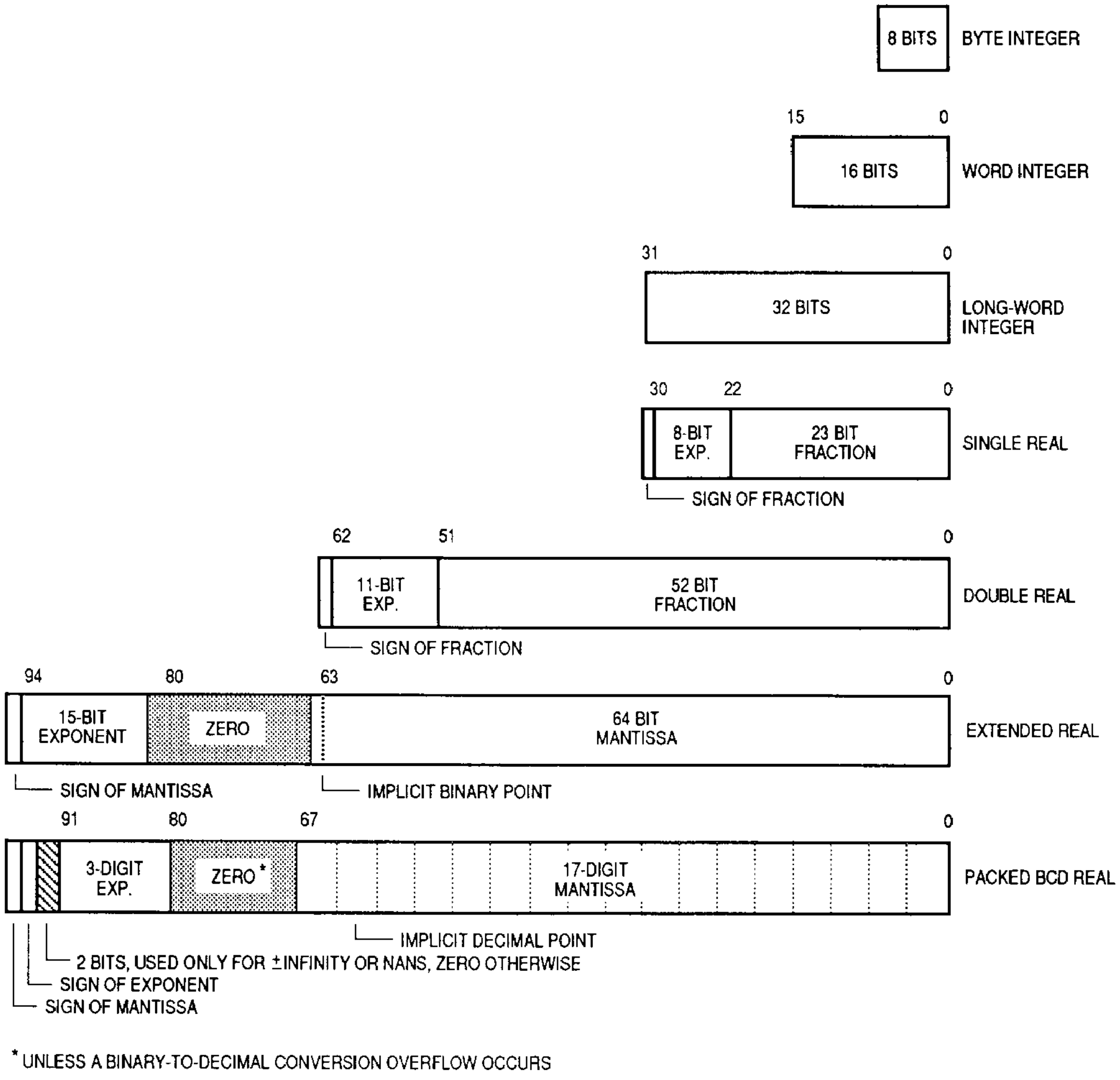


Figure 9. Data Format Summary

INSTRUCTION SET

The MC68882 instruction set is organized into six major classes:

1. Moves between the MC68882 and memory or the MC68020/MC68030 (in and out)
2. Move multiple registers (in and out)
3. Monadic operations
4. Dyadic operations
5. Branch, set, or trap conditionally
6. Miscellaneous

MOVES

On all moves from memory (or from an MC68020/MC68030 data register) to the MC68882, data is converted from the source data format to the internal extended-precision format. On all moves from the MC68882 to memory (or to an MC68020/MC68030 data register), data is converted from the internal extended-precision format to the destination data format. Note that data movement instructions perform arithmetic operations since the result is always rounded to the precision selected in the FPCR mode control byte. The result is rounded using the selected rounding mode and is checked for overflow and underflow.

The syntax for the move is as follows:

```
FMOVE.<fmt>  <ea>,FPn     Move to MC68882
FMOVE.<fmt>  FPm,<ea>     Move from MC68882
FMOVE.X      FPm,FPn      Move within MC68882
```

where <ea> is an MC68020/MC68030 effective address operand, .<fmt> is the data format size, and FPm and FPn are floating-point data registers.

MOVE MULTIPLE REGISTERS

The floating-point move multiple instructions on the MC68882 are much like the integer counterparts on the M68000 Family processors. Any set of the floating-point registers FP0–FP7 can be moved to or from memory with one instruction. These registers are always moved as 96-bit extended data with no conversion (no possibility of conversion errors). Some examples of the move multiple instruction are as follows:

```
FMOVEM   <ea>,FP0–FP3/FP7
FMOVEM   FP2/FP4/FP6,<ea>
```

The move multiple instructions are useful during context switches and interrupts to save or restore the state of a program. These moves are also useful at the start and end of a procedure to save and restore the register set of the calling routine. To reduce procedure call overhead, the list of registers to be saved or restored can be contained in a data register, thus enabling run-time optimization by allowing a called routine to save as few registers as possible. Note that no rounding or overflow/underflow checking is performed by these operations.

MONADIC OPERATIONS

Monadic operations have one operand. This operand may be in a floating-point data register, memory, or in an MC68020/MC68030 data register. The result is always stored in a floating-point data register. For example, the syntax for square root is one of the following:

```
FSQRT.<fmt>   <ea>,FPn
FSQRT.X       FPm,FPn
FSQRT.X       FPn
```

The available MC68882 monadic operations are as follows:

FABS	Absolute Value
FACOS	Arc Cosine
FASIN	Arc Sine
FATAN	Arc Tangent
FATANH	Hyperbolic Arc Tangent
FCOS	Cosine
FCOSH	Hyperbolic Cosine
FETOX	e to the x Power
FETOXM1	e to the x Power −1
FGETEXP	Get Exponent
FGETMAN	Get Mantissa
FINT	Integer Part
FLINTRZ	Integer Part (Truncated)
FLOG10	Log Base 10
FLOG2	Log Base 2
FLOGN	Log Base e
FLOGNP1	Log Base e of (x+1)
FNEG	Negate
FSIN	Sine
FSINCOS	Simultaneous Sine and Cosine
FSINH	Hyperbolic Sine
FSQRT	Square Root
FTAN	Tangent
FTANH	Hyperbolic Tangent
FTENTOX	10 to the x Power
FTST	Test
FTWOTOX	2 to the x Power

DYADIC OPERATIONS

Dyadic operations have two operands each. The first operand is in a floating-point data register, memory, or an MC68020/MC68030 data register. The second operand is the contents of a floating-point data register. The destination is the same floating-point data register used for the second operand. For example, the syntax for floating-point add is as follows:

```
FADD.<fmt>    <ea>,FPn
FADD.X        FPm,FPn
```

The available MC68882 dyadic operations are as follows:

FADD	Add
FCMP	Compare
FDIV	Divide
FMOD	Modulo Remainder
FMUL	Multiply
FREM	IEEE Remainder
FSCALE	Scale Exponent
FSGLDIV	Single-Precision Divide
FSGLMUL	Single-Precision Multiply
FSUB	Subtract

BRANCH, SET, AND TRAP ON CONDITION

The floating-point branch, set, and trap on condition instructions implemented by the MC68882 are similar to the equivalent integer instructions of the M68000 Family processors, except more conditions exist due to the special values in IEEE floating-point arithmetic. When a conditional instruction is executed, the MC68882 performs the necessary condition checking and reports to the MC68020/MC68030 whether the condition is true or false. The MC68020/MC68030 then takes the appropriate action. Since the MC68882 and MC68020/MC68030 are closely coupled, the floating-point branch operations execute very quickly.

The MC68882 conditional operations are as follows:

FBcc	Branch
FDBcc	Decrement and Branch
FScc	Set According to Condition
FTRAPcc	Trap on Condition (with an Optional Parameter)

where cc is one of the 32 floating-point conditional test specifiers listed in Table 2.

Table 2. Floating-Point Conditional Test Specifiers

<table>
<tr><th>Mnemonic</th><th>Definition</th></tr>
<tr><td colspan="2"><b>NOTE</b>
The following conditional tests do not set the BSUN bit in the status register exception byte under any circumstances.</td></tr>
<tr><td>F</td><td>False</td></tr>
<tr><td>EQ</td><td>Equal</td></tr>
<tr><td>OGT</td><td>Ordered Greater Than</td></tr>
<tr><td>OGE</td><td>Ordered Greater Than or Equal</td></tr>
<tr><td>OLT</td><td>Ordered Less Than</td></tr>
<tr><td>OLE</td><td>Ordered Less Than or Equal</td></tr>
<tr><td>OGL</td><td>Ordered Greater or Less Than</td></tr>
<tr><td>OR</td><td>Ordered</td></tr>
<tr><td>UN</td><td>Unordered</td></tr>
<tr><td>UEQ</td><td>Unordered or Equal</td></tr>
<tr><td>UGT</td><td>Unordered or Greater Than</td></tr>
<tr><td>UGE</td><td>Unordered or Greater or Equal</td></tr>
<tr><td>ULT</td><td>Unordered or Less Than</td></tr>
<tr><td>ULE</td><td>Unordered or Less or Equal</td></tr>
<tr><td>NE</td><td>Not Equal</td></tr>
<tr><td>T</td><td>True</td></tr>
<tr><td colspan="2"><b>NOTE</b>
All the following conditional tests set the BSUN bit in the status register exception byte if the NAN condition code bit is set when a conditional instruction is executed.</td></tr>
<tr><td>SF</td><td>Signaling False</td></tr>
<tr><td>SEQ</td><td>Signaling Equal</td></tr>
<tr><td>GT</td><td>Greater Than</td></tr>
<tr><td>GE</td><td>Greater Than or Equal</td></tr>
<tr><td>LT</td><td>Less Than</td></tr>
<tr><td>LE</td><td>Less Than or Equal</td></tr>
<tr><td>GL</td><td>Greater or Less Than</td></tr>
<tr><td>GLE</td><td>Greater, Less, or Equal</td></tr>
<tr><td>NGLE</td><td>Not (Greater, Less, or Equal)</td></tr>
<tr><td>NGL</td><td>Not (Greater or Less)</td></tr>
<tr><td>NLE</td><td>Not (Less or Equal)</td></tr>
<tr><td>NLT</td><td>Not (Less Than)</td></tr>
<tr><td>NGE</td><td>Not (Greater or Equal)</td></tr>
<tr><td>NGT</td><td>Not (Greater Than)</td></tr>
<tr><td>SNE</td><td>Signaling Not Equal</td></tr>
<tr><td>ST</td><td>Signaling True</td></tr>
</table>

MISCELLANEOUS INSTRUCTIONS

Miscellaneous instructions include moves to and from the status, control, and instruction address registers. Also included are the virtual memory/machine FSAVE and FRESTORE instructions that save and restore the internal state of the MC68882:

FMOVE	<ea>,FPcr	Move to Control Register(s)
FMOVE	FPcr,<ea>	Move from Control Register(s)
FSAVE	<ea>	Virtual Machine State Save
FRESTORE	<ea>	Virtual Machine State Restore

ADDRESSING MODES

The MC68882 does not perform address calculations. Thus, if the MC68882 instructs the MC68020/MC68030 to transfer an operand via the coprocessor interface, the MC68020/MC68030 performs the addressing mode calculations requested in the instruction. In this case, the instruction is encoded specifically for the MC68020/MC68030, and the execution of the MC68882 is dependent only on the value of the command word written to the MC68882 by the main processor.

This interface is flexible and allows any addressing mode to be used with floating-point instructions. For the M68000 Family, these addressing modes include immediate, postincrement, predecrement, data or address register direct, and the indexed/indirect addressing modes of the MC68020/MC68030. Some addressing modes are restricted for instructions consistent with the M68000 Family architectural definitions (e.g., program counter relative addressing is not allowed for a destination operand).

The orthogonal instruction set of the MC68882 and the flexible branches and addressing modes of the MC68020/MC68030 allow a programmer or a compiler to think of the MC68882 as though it were part of the MC68020/MC68030. No special restrictions are imposed by the coprocessor interface, and floating-point arithmetic is coded exactly like integer arithmetic.

MC68881 COMPATIBILITY

Using the MC68882 in an existing MC68881 socket does not require hardware changes nor user-software modifications. Implementation of multiple floating-point instruction execution concurrency gives the MC68882 a performance advantage over the MC68881. However, to guarantee that the floating-point exception model maintains the precepts of a sequential execution model, some systems-level software modifications are needed to upgrade the system to operate properly with an MC68882.

First, note that the MC68882 idle and busy state frames (generated by the FSAVE instruction) are both 32 bytes larger than those of the MC68881. The offsets for the exceptional operand, the operand register word, and the BIU flag word from the top of the saved idle state frame are 32 bytes more than that of the MC68881. However, a unique format word is generated by the MC68882, enabling the system software to detect this difference. The unique format word prevents a saved MC68881 context from being restored into an MC68882 and vice versa.

Second, the branch or set on unordered (BSUN), signaling not-a-number (SNAN), operand error (OPERR), overflow (OVFL), divide by zero (DZ), and inexact result (INEX) floating-point exception handlers must have these minimum requirements:

1. FSAVE must be executed before any other floating-point instruction.
2. A BSET or similar instruction sets bit 27 of the BIU flag word (located in the saved idle state frame).
3. FRESTORE must be executed before RTE.

These requirements are not applicable to interrupt handlers that do not contain any floating-point instructions. For interrupt handlers that have floating-point instructions, only requirements 1 and 3 must be implemented.

FUNCTIONAL SIGNAL DESCRIPTIONS

The following paragraphs contain a brief description of the input and output signals for the MC68882 floating-point coprocessor. The signals are functionally organized into groups as shown in Figure 10.

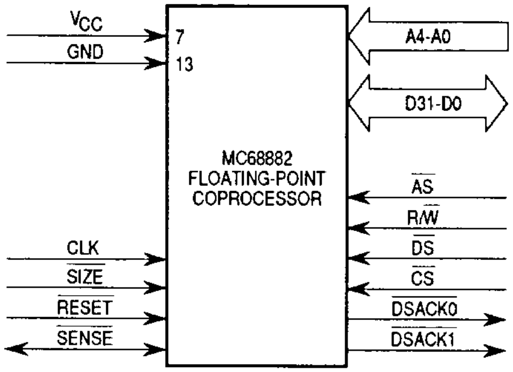


Figure 10. Functional Signal Groups

NOTE

The terms **assertion** and **negation** are used extensively to avoid confusion when describing active-low and active-high signals. The term **assert** or **assertion** is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term **negate** or **negation** is used to indicate that a signal is inactive or false.

ADDRESS BUS (A0–A4)

These active-high inputs are used by the main processor to select the CIR locations in the CPU address space. These lines control the register selection (see Table 3).

Table 3. Coprocessor Interface Register Selection

A4–A0	Offset	Width	Type	Register
0000x	$00	16	Read	Response
0001x	$02	16	Write	Control
0010x	$04	16	Read	Save
0011x	$06	16	R/$\overline{\text{W}}$	Restore
0100x	$08	16	—	(Reserved)
0101x	$0A	16	Write	Command
0110x	$0C	16	—	(Reserved)
0111x	$0E	16	Write	Condition
100xx	$10	32	R/$\overline{\text{W}}$	Operand
1010x	$14	16	Read	Register Select
1011x	$16	16	—	(Reserved)
110xx	$18	32	Read	Instruction Address
111xx	$1C	32	R/$\overline{\text{W}}$	Operand Address

When the MC68882 is configured to operate over an 8-bit data bus, the A0 pin is used as an address signal for byte accesses of the CIRs. When the MC68882 is configured to operate over a 16- or 32-bit data bus, both the A0 and the $\overline{\text{SIZE}}$ pins are strapped high and/or low as listed in Table 4.

Table 4. Data Bus Size Configuration

A0	$\overline{\text{SIZE}}$	Data Bus
—	Low	8-Bit
Low	High	16-Bit
High	High	32-Bit

DATA BUS (D0–D31)

This 32-bit, bidirectional, three-state bus serves as the general-purpose data path between the MC68020/MC68030 and the MC68882. Regardless of whether the MC68882 is operated as a coprocessor or a peripheral processor, all interprocessor transfers of instruction information, operand data, status information, and requests for service occur as standard M68000 bus cycles.

The MC68882 will operate over an 8-, 16-, or 32-bit data bus. Depending upon the system data bus configuration, A0 and $\overline{\text{SIZE}}$ are configured specifically for the applicable bus configuration. (Refer to **ADDRESS BUS (A0–A4)** and **SIZE ($\overline{\text{SIZE}}$)** for further details).

SIZE ($\overline{\text{SIZE}}$)

This active-low input is used in conjunction with the A0 pin to configure the MC68882 for operation over an 8-, 16-, or 32-bit data bus. When the MC68882 is configured to operate over a 16- or 32-bit data bus, $\overline{\text{SIZE}}$ and A0 are strapped high and/or low as listed in Table 4.

ADDRESS STROBE ($\overline{\text{AS}}$)

This active-low input indicates a valid address on the address bus and valid signals for chip select ($\overline{\text{CS}}$) and read/write (R/$\overline{\text{W}}$).

CHIP SELECT ($\overline{\text{CS}}$)

This active-low input enables the main processor access to the MC68882 CIRs. When operating the MC68882 as a peripheral processor, the chip-select decode is system dependent (i.e., like the chip select on any peripheral).

READ/WRITE (R/$\overline{\text{W}}$)

This input indicates the direction of a bus transaction (read/write) by the main processor. A logic high (1) indicates a read from the MC68882, and a logic low (0) indicates a write to the MC68882. The R/$\overline{\text{W}}$ signal must be valid when $\overline{\text{AS}}$ is asserted.

DATA STROBE ($\overline{\text{DS}}$)

This active-low input indicates is valid data on the data bus during a write bus cycle.

DATA TRANSFER AND SIZE ACKNOWLEDGE ($\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$)

These active-low, three-state outputs indicate the completion of a bus cycle to the main processor. The MC68882 asserts both $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ upon assertion of $\overline{\text{CS}}$.

If the bus cycle is a main processor read, the MC68882 asserts $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ to indicate information on the data bus is valid. (Both $\overline{\text{DSACKx}}$ signals may be asserted in advance of the valid data being placed on the bus.) If the bus cycle is a main processor write to the MC68882, $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ are used to acknowledge acceptance of the data by the MC68882.

The MC68882 also uses DSACK0 and DSACK1 signals to dynamically indicate to the MC68020/MC68030 the port size (data bus width) on a cycle-by-cycle basis. Depending upon which of the DSACKx pins is asserted in a given bus cycle, the MC68020/MC68030 assumes data has been transferred to/from an 8-, 16-, or 32-bit data port. Table 5 lists the DSACKx assertions that are used by the MC68882 for the various bus cycles over the various data bus configurations.

Table 5. DSACKx Assertions

Data Bus	A4	DSACK1	DSACK0	Comments
32-Bit	1	Low	Low	Valid Data on D31–D0
32-Bit	0	Low	High	Valid Data on D31–D16
16-Bit	x	Low	High	Valid Data on D31–D16 or D15–D0
8-Bit	x	High	Low	Valid Data on D31–D24, D23–D16, D15–D8, or D7–D0
All	x	High	High	Insert Wait States in Current Bus Cycle

Table 5 indicates that all accesses over a 32-bit bus where A4 equals zero are to 16-bit registers. The MC68882 implements all 16-bit CIRs on data lines D16–D31 (to eliminate the need for on-chip multiplexers); however, the MC68020/MC68030 expects 16-bit registers that are located in a 32-bit port at odd word addresses (A1 = 1) to be implemented on data lines D0–D15. For accesses to these registers when configured for 32-bit bus operation, the MC68882 generates DSACKx signals as listed in Table 5 to inform the MC68020/MC68030 of valid data on D16–D31 instead of D0–D15.

An external holding resistor is required to maintain DSACK0 and DSACK1 high between bus cycles. To reduce the signal rise time, DSACK0 and DSACK1 are actively pulled up (negated) by the MC68882 following the rising edge of AS or DS, and both DSACKx lines are then three-stated (placed in the high-impedance state) to avoid interference with the next bus cycle.

RESET (RESET)

This active-low input causes the MC68882 to initialize the floating-point data registers to non-signaling not-a-numbers (NANs) and clears the floating-point control, status, and instruction address registers.

When performing a power-up reset, external circuitry should keep RESET asserted for a minimum of four clock cycles after V_{CC} is within tolerance, assuring correct initialization of the MC68882 when power is applied. For compatibility with all M68000 Family devices, 100 ms should be used as the minimum.

When performing a reset of the MC68882 after V_{CC} has been within tolerance for more than the initial power-up time, $\overline{\text{RESET}}$ must have an asserted pulse width which is greater than two clock cycles. For compatibility with all M68000 Family devices, 10 clock cycles should be used as the minimum.

CLOCK (CLK)

The MC68882 clock input is a TTL-compatible signal that is internally buffered for development of the internal clock signals. The clock input should be a constant-frequency square wave with no stretching or shaping techniques required. The clock should not be gated off at any time and must conform to minimum and maximum period and pulse-width times.

SENSE DEVICE ($\overline{\text{SENSE}}$)

This pin may be used optionally as an additional GND pin or as as indicator to external hardware that the MC68882 is present in the system. This signal is internally connected to the GND of the die, but it is not necessary to connect it to the external ground for correct device operation. If a pullup resistor (which should be larger than 10 Ω) is connected to this pin location, external hardware may sense the presence of the MC68882 in a system.

POWER (V_{CC} and GND)

These pins provide the supply voltage and system reference level for the internal circuitry of the MC68882. Care should be taken to reduce the noise level on these pins with appropriate capacitive decoupling.

NO CONNECT (NC)

One pin of the MC68882 package is designated as a no connect (NC). Reserved for future use by Motorola, this pin should not be used for signal routing or connected to V_{CC} or GND.

SIGNAL SUMMARY

Table 6 provides a summary of all MC68882 signals described in the previous paragraphs.

Table 6. Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Three State
Address Bus	A0–A4	Input	High	—
Data Bus	D0–D13	Input/Output	High	Yes
Size	$\overline{\text{SIZE}}$	Input	Low	—
Address Strobe	$\overline{\text{AS}}$	Input	Low	—
Chip Select	$\overline{\text{CS}}$	Input	Low	—
Read/Write	$\text{R/}\overline{\text{W}}$	Input	High/Low	—
Data Strobe	$\overline{\text{DS}}$	Input	Low	—
Data Transfer and Size Acknowledge	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$	Output	Low	Yes
Reset	$\overline{\text{RESET}}$	Input	Low	—
Clock	CLK	Input	—	—
Sense Device	$\overline{\text{SENSE}}$	Input/Output	Low	No
Power Input	V_{CC}	Input	—	—
Ground	GND	Input	—	—

INTERFACING METHODS

The following paragraphs describe how to connect an MC68882 to an M68000 Family processor.

MC68882 TO MC68020/MC68030 INTERFACING

The following paragraphs describe how to connect the MC68882 to an MC68020/MC68030 for coprocessor operation via an 8-, 16-, or 32-bit data bus.

32-Bit Data Bus Coprocessor Connection

Figure 11 illustrates the coprocessor interface connection of an MC68882 to an MC68020/MC68030 via a 32-bit data bus. The MC68882 is configured to operate over a 32-bit data bus when both A0 and $\overline{\text{SIZE}}$ are connected to V_{CC}.

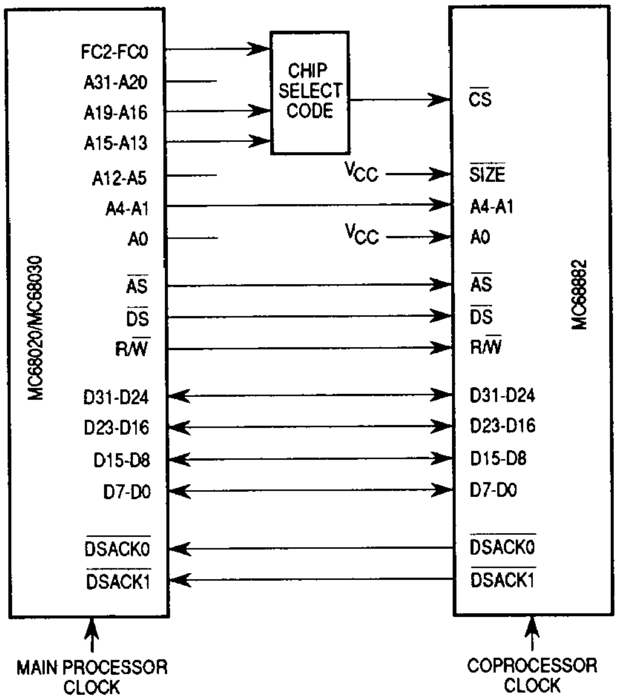


Figure 11. 32-Bit Data Bus Coprocessor Connection

16-Bit Data Bus Coprocessor Connection

Figure 12 illustrates the coprocessor interface connection of an MC68882 to an MC68020/MC68030 via a 16-bit data bus. The MC68882 is configured to operate over a 16-bit data bus when $\overline{\text{SIZE}}$ is connected to V_{CC} and A0 is connected to GND. The 16 least significant data pins (D0–D15) must be connected to the 16 most significant data pins (D16–D31) when the MC68882 is configured to operate over a 16-bit data bus (i.e., connect D0 to D16, D1 to D17, . . . and D15 to D31). The $\overline{\text{DSACKx}}$ pins of the two devices are directly connected, although it is not necessary to connect $\overline{\text{DSACK0}}$ since the MC68882 never asserts it in this configuration.

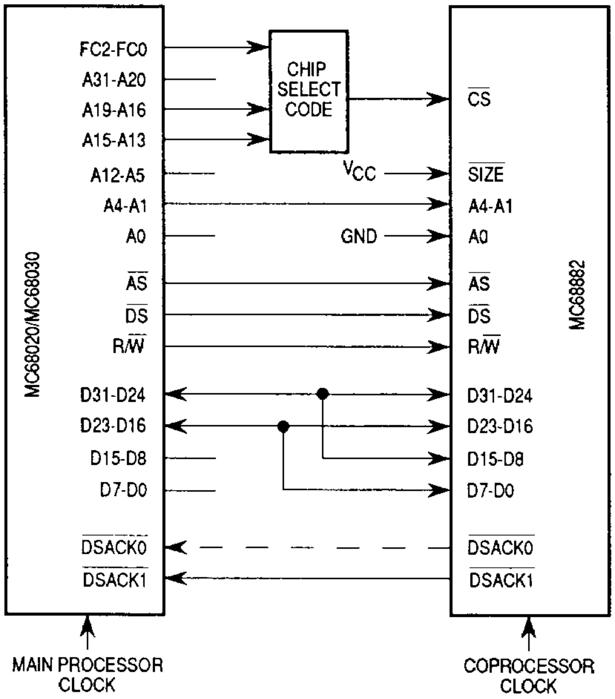


Figure 12. 16-Bit Data Bus Coprocessor Connection

8-Bit Data Bus Coprocessor Connection

Figure 13 illustrates the connection of an MC68882 to an MC68020/MC68030 as a coprocessor over an 8-bit data bus. The MC68882 is configured to operate over an 8-bit data bus when $\overline{\text{SIZE}}$ is connected to GND. The 24 least significant data pins (D0–D23) must be connected to the eight most significant data pins (D24–D31) when the MC68882 is configured to operate over an 8-bit data bus (i.e., connect D0 to D8, D16, and D24; D1 to D9, D17, and D25; . . . and D7 to D15, D23, and D31). The $\overline{\text{DSACKx}}$ pins of the two devices are directly connected, although it is not necessary to connect $\overline{\text{DSACK1}}$ since the MC68882 never asserts it in this configuration.

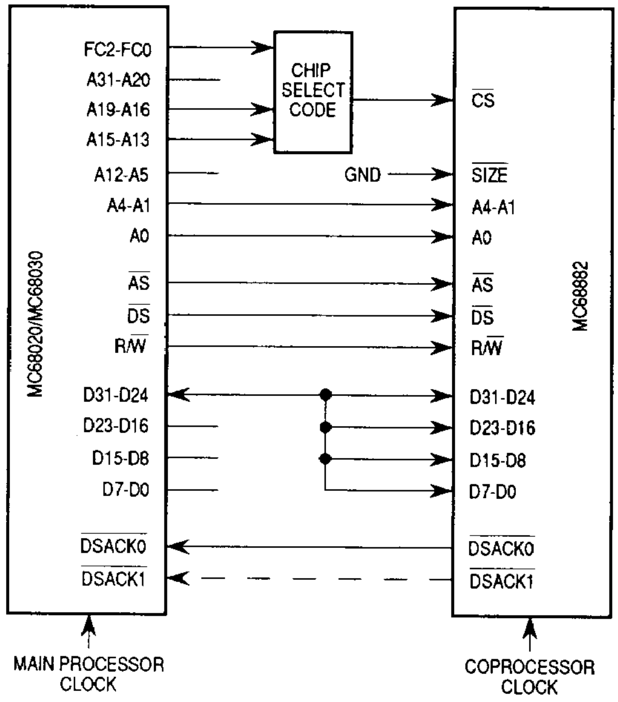


Figure 13. 8-Bit Data Bus Coprocessor Connection

MC68882 TO MC68000/MC68008/MC68010 INTERFACING

The following paragraphs describe how to connect the MC68882 to an MC68000/MC68008/MC68010 processor for operation as a peripheral via an 8- or 16-bit data bus.

16-Bit Data Bus Peripheral Processor Connection

Figure 14 illustrates the connection of an MC68882 to an MC68000 or MC68010 as a peripheral processor over a 16-bit data bus. The MC68882 is configured to operate over a 16-bit data bus when $\overline{\mathrm{SIZE}}$ is connected to V_{CC}, and A0 is connected to GND. The 16 least significant data pins (D0–D15) must be connected to the 16 most significant data pins (D16–D31) when the MC68882 is configured to operate over a 16-bit data bus (i.e., connect D0 to D16, D1 to D17, . . . and D15 to D31). The $\overline{\mathrm{DSACK1}}$ pin of the MC68882 is connected to the $\overline{\mathrm{DTACK}}$ pin of the main processor; $\overline{\mathrm{DSACK0}}$ is not used.

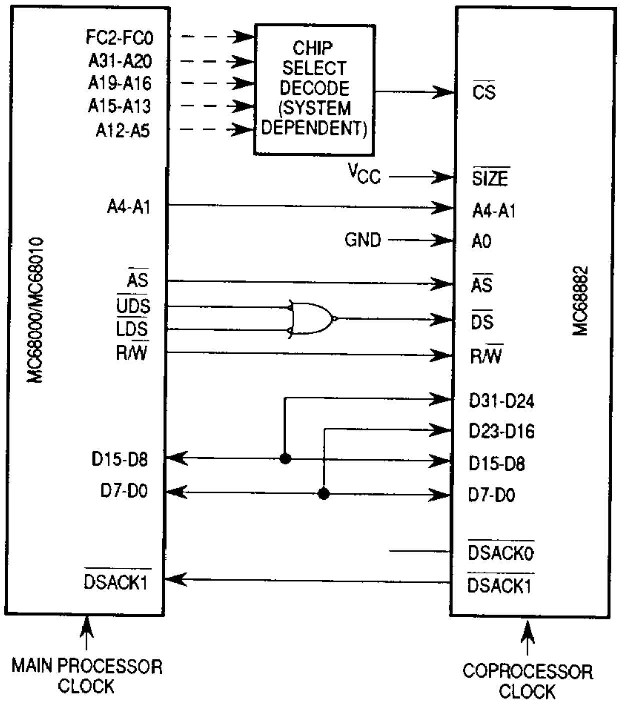


Figure 14. 16-Bit Data Bus Peripheral Processor Connection

When connected as a peripheral processor, the MC68882 chip-select decode is system dependent. If the MC68000 is used as the main processor, the MC68882 CS must be decoded in the supervisor or user data spaces. However, if the MC68010 is used for the main processor, the MOVES instruction may be used to emulate any CPU space access that the MC68020/MC68030 generates for coprocessor communications. Thus, the CS decode logic for such systems may be the same as in an MC68020/MC68030 system so that the MC68882 will not use any part of the data address spaces.

8-Bit Data Bus Peripheral Processor Connection

Figure 15 illustrates the connection of an MC68882 to an MC68008 as a peripheral processor over an 8-bit data bus. The MC68882 is configured to operate over an 8-bit data bus when SIZE is connected to GND. The eight least significant data pins (D0–D7) must be connected to the 24 most significant pins (D8–D31) when the MC68882 is configured to operate over an 8-bit data bus (i.e., connect D0 to D8, D16, and D24; D1 to D9, D17, and D25; . . . and D7 to D15, D23, and D31). The DSACK0 pin of the MC68882 is connected to the DTACK pin of the MC6800; DSACK1 is not used.

When connected as a peripheral processor, the MC68882 chip-select decode is system dependent, and $\overline{CS}$ must be decoded in the supervisor or user data spaces.

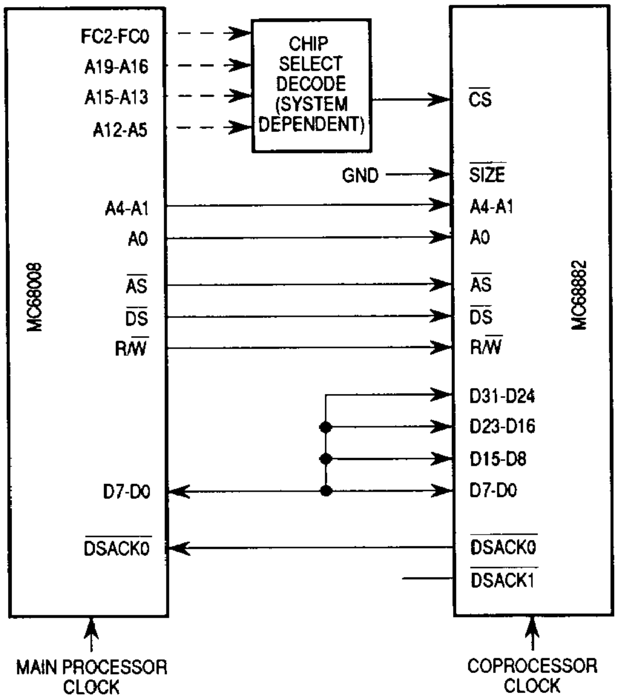


Figure 15. 8-Bit Data Bus Peripheral Processor Connection

ELECTRICAL SPECIFICATIONS

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature	T_A	0 to 70	°C
Storage Temperature	T_{stg}	−55 to +150	°C

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

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Rating
Thermal Resistance — Ceramic			°C/W
Junction to Ambient	θ_{JA}	33	
Junction to Case	θ_{JC}	15	
Thermal Resistance — PLCC			
Junction to Ambient	θ_{JA}	45	
Junction to Case	θ_{JC}	TBD	

TBD — To Be Determined

POWER 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.

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

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

5

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

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

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

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

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

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, ambient air cooling, and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in Equation (1) will result in a lower semiconductor junction temperature.

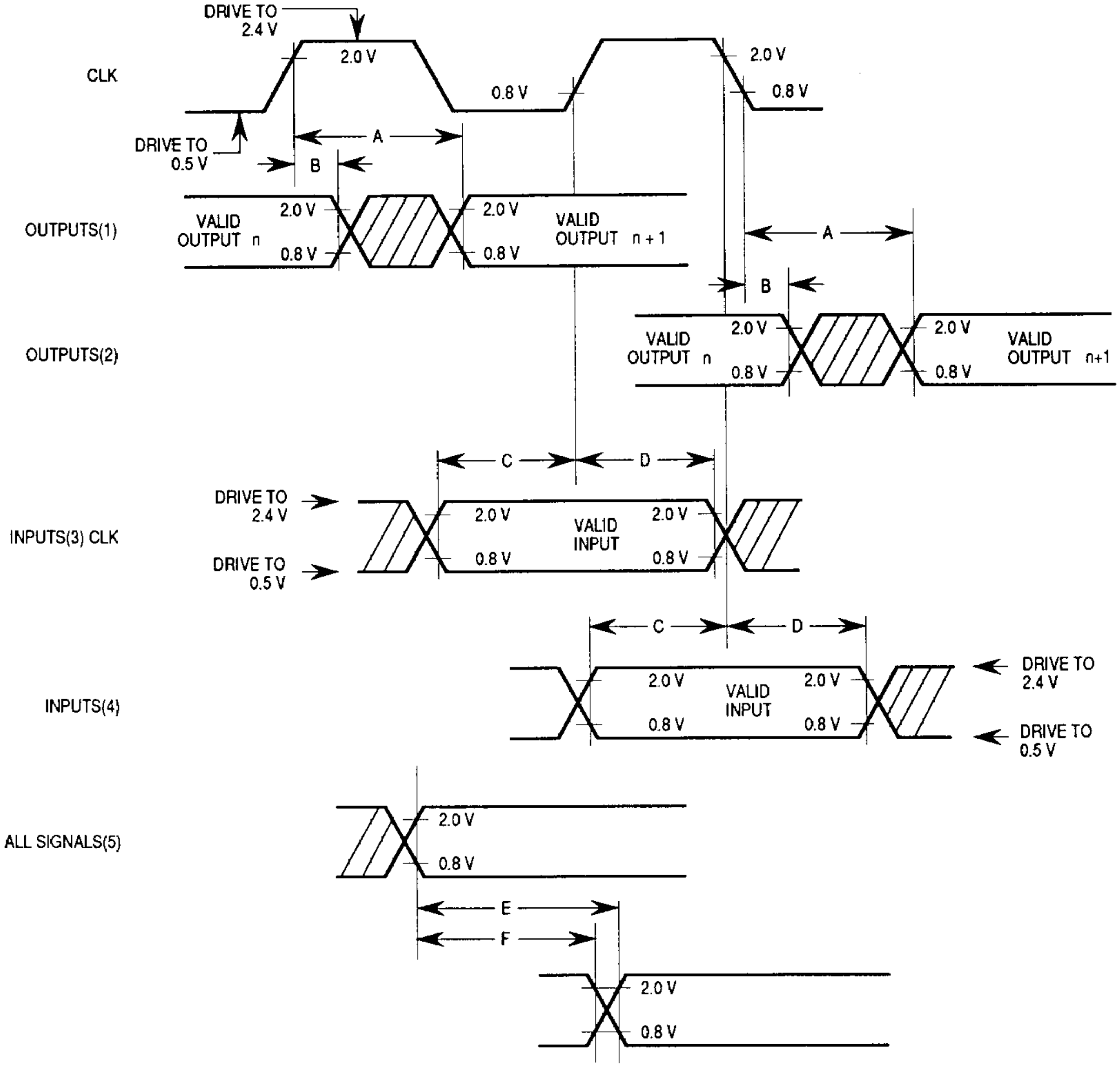
Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex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

AC ELECTRICAL SPECIFICATIONS DEFINITIONS

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

The measurement of the AC specifications is defined by the waveforms shown in Figure 16. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 16. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs 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 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 16. Drive Levels and Test Points for AC Specifications

DC ELECTRICAL SPECIFICATIONS (V_{CC}=5.0 Vdc±5%; GND=0 Vdc; T_A=0°C to 70°C)

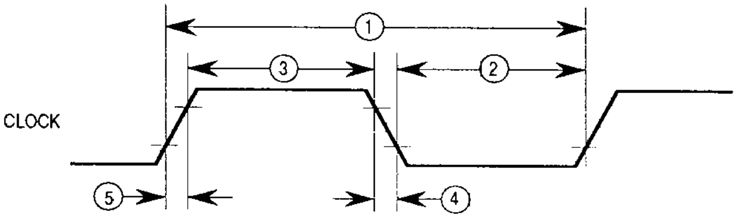
Characteristic		Symbol	Min	Max	Unit
Input High Voltage		V_{IH}	2.0	V_{CC}	V
Input Low Voltage		V_{IL}	GND −0.5	0.8	V
Input Leakage Current @ 5.25 V	CLK, $\overline{RESET}$, R/$\overline{W}$, A0–A4, $\overline{CS}$, $\overline{DS}$, $\overline{AS}$, $\overline{SIZE}$	I_{in}	—	10	μA
Hi-Z (Off State) Input Current @ 2.4 V/0.4 V	$\overline{DSACK0}$, $\overline{DSACK1}$, D0–D31	I_{TSI}	—	20	μA
Output High Voltage (I_{OH} = −400 μA)	$\overline{DSACK0}$, $\overline{DSACK1}$, D0–D31	V_{OH}	2.4	—	V
Output Low Voltage (I_{OL}=5.3 mA)	$\overline{DSACK0}$, $\overline{DSACK1}$, D0–D31	V_{OL}	—	0.5	V
Output Low Current (V_{OL}=GND)	$\overline{SENSE}$	I_{OL}	—	500	μA
Power Dissipation		P_D	—	0.75	W
Capacitance* (V_{in}=0, T_A=25°C, f=1 MHz)		C_{in}	—	20	pF
Output Load Capacitance		C_L	—	130	pF

*Capacitance is periodically sampled rather than 100% tested.

AC ELECTRICAL SPECIFICATIONS — CLOCK INPUT

(V_{CC}=5.0 Vdc±5%; GND=0 Vdc; T_A=0 to 70°C; see Figure 17)

Num	Characteristic	16.67 MHz		20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
	Frequency of Operation	8	16.67	12.5	20	12.5	25	16.7	33.33	25	40	25	50	MHz
1	Cycle Time	60	125	50	80	40	80	30	60	25	40	20	40	ns
2,3	Clock Pulse Width (Measured from 1.5 V to 1.5 V for 33 MHz)	24	95	20	54	15	59	14	66	11.5	29	9.5	30.5	ns
4,5	Rise and Fall Times	—	5	—	5	—	4	—	3	—	2	—	2	ns



NOTE: Timing measurements are referenced to and from a low voltage of 0.8V and a high voltage of 2.0V, 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 0.8V and 2.0V.

Figure 17. Clock Input Timing Diagram

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 5.0 Vdc ± 5%; GND = 0 Vdc; T_A = 0 to 70°C; see Figures 18, 19, and 20)

Num	Characteristic	16.67 MHz		20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
6[5]	Address Valid to $\overline{AS}$ Asserted	15	—	10	—	5	—	5	—	2	—	2	—	ns
6A[5]	Address Valid to $\overline{DS}$ Asserted (Read)	15	—	10	—	5	—	5	—	2	—	2	—	ns
6B[5]	Address Valid to $\overline{DS}$ Asserted (Write)	50	—	50	—	35	—	26	—	18	—	18	—	ns
7[6]	$\overline{AS}$ Negated to Address Invalid	10	—	10	—	5	—	5	—	2	—	2	—	ns
7A[6]	$\overline{DS}$ Negated to Address Invalid	10	—	10	—	5	—	5	—	2	—	2	—	ns
8[9]	$\overline{CS}$ Negated to $\overline{AS}$ Asserted	0	—	0	—	0	—	0	—	0	—	0	—	ns
8A[9]	$\overline{CS}$ Negated to $\overline{DS}$ Asserted (Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
8B	$\overline{CS}$ Asserted to $\overline{DS}$ Asserted (Write)	30	—	25	—	20	—	15	—	10	—	10	—	ns
9	$\overline{AS}$ Negated to $\overline{CS}$ Negated	10	—	10	—	5	—	5	—	2	—	2	—	ns
9A	$\overline{DS}$ Negated to $\overline{CS}$ Negated	10	—	10	—	5	—	5	—	2	—	2	—	ns
10	R/$\overline{W}$ High to $\overline{AS}$ Asserted (Read)	15	—	10	—	5	—	5	—	3	—	3	—	ns
10A	R/$\overline{W}$ High to $\overline{DS}$ Asserted (Read)	15	—	10	—	5	—	5	—	3	—	3	—	ns
10B	R/$\overline{W}$ Low to $\overline{DS}$ Asserted (Write)	35	—	30	—	25	—	25	—	23	—	23	—	ns
11	$\overline{AS}$ Negated to R/$\overline{W}$ Low (Read) or $\overline{AS}$ Negated to R/$\overline{W}$ High (Write)	10	—	10	—	5	—	5	—	2	—	2	—	ns
11A	$\overline{DS}$ Negated to R/$\overline{W}$ Low (Read) or $\overline{DS}$ Negated to R/$\overline{W}$ High (Write)	10	—	10	—	5	—	5	—	2	—	2	—	ns

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES
(Continued)

Num	Characteristic	16.67 MHz		20 MHz		25 MHz		33.33 MHz		40 MHz		50 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
12	$\overline{\mathrm{DS}}$ Width Asserted (Write)	40	—	38	—	30	—	23	—	13	—	13	—	ns
13	$\overline{\mathrm{DS}}$ Width Negated	40	—	38	—	30	—	23	—	13	—	13	—	ns
13A[4]	$\overline{\mathrm{DS}}$ Negated to $\overline{\mathrm{AS}}$ Asserted	30	—	30	—	25	—	18	—	14	—	14	—	ns
14[2]	$\overline{\mathrm{CS}}$, $\overline{\mathrm{DS}}$ Asserted to Data-Out Valid (Read)	—	80	—	60	—	45	—	30	—	25	—	20	ns
15	$\overline{\mathrm{DS}}$ Negated to Data-Out Invalid (Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
16	$\overline{\mathrm{DS}}$ Negated to Data-Out High Impedance (Read)	—	50	—	30	—	30	—	20	—	20	—	20	ns
17	Data-In Valid to $\overline{\mathrm{DS}}$ Asserted (Write)	15	—	10	—	5	—	5	—	2	—	2	—	ns
18	$\overline{\mathrm{DS}}$ Negated to Data-In Invalid (Write)	15	—	10	—	5	—	5	—	2	—	2	—	ns
19[2]	$\overline{\mathrm{START}}$ True to $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ Asserted	—	50	—	35	—	25	—	20	—	15	—	15	ns
19A[7]	$\overline{\mathrm{DSACK0}}$ Asserted to $\overline{\mathrm{DSACK1}}$ Asserted (Skew)	−15	15	−10	10	−10	10	—	5	—	3	—	3	ns
20	$\overline{\mathrm{DSACK0}}$ or $\overline{\mathrm{DSACK1}}$ Asserted to Data-Out Valid	—	50	—	43	—	32	—	17	—	10	—	5	ns
21[8]	$\overline{\mathrm{START}}$ False to $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ Negated	—	50	—	30	—	30	—	20	—	20	—	20	ns
22[8]	$\overline{\mathrm{START}}$ False to $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ High Impedance	—	70	—	40	—	40	—	30	—	30	—	30	ns
23[3,8]	$\overline{\mathrm{START}}$ True to Clock High (Synchronous Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
24[3]	Clock Low to Data-Out Valid (Synchronous Read)	—	105	—	80	—	60	—	45	—	36	—	36	ns
25[3,8]	$\overline{\mathrm{START}}$ True to Data-Out Valid (Synchronous Read)	— 1.5	105+ 2.5	— 1.5	80+ 2.5	— 1.5	60+ 2.5	— 1.5	45+ 2.5	— 1.5	36+ 2.5	— 1.5	36+ 2.5	ns Clks
26[3]	Clock Low to $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ Asserted (Synchronous Read)	—	75	—	55	—	45	—	30	—	23	—	23	ns
27[3,8]	$\overline{\mathrm{START}}$ True to $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ Asserted (Synchronous Read)	— 1.5	75+ 2.5	— 1.5	55+ 2.5	— 1.5	45+ 2.5	— 1.5	30+ 2.5	— 1.5	23+ 2.5	— 1.5	23+ 2.5	ns Clks

NOTES:

1. 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.
2. These specifications only apply if the MC68882 has completed all internal operations initiated by the termination of the previous bus cycle when $\overline{\mathrm{DS}}$ was negated.
3. Synchronous read cycles occur *only* when the save or response CIR locations are read.
4. This specification only applies to systems in which back-to-back accesses (read-write or write-write) of the operand CIR can occur. When the MC68882 is used as a coprocessor to the MC68020/MC68030, this can occur when the addressing mode is Immediate.
5. If the $\overline{\mathrm{SIZE}}$ pin is *not* strapped to either V_{CC} or GND, it must have the same setup times as do addresses.
6. If the $\overline{\mathrm{SIZE}}$ pin is *not* strapped to either V_{CC} or GND, it must have the same hold times as do addresses.
7. This number is reduced to 5 ns if $\overline{\mathrm{DSACK0}}$ and $\overline{\mathrm{DSACK1}}$ have equal loads.
8. START is not an external signal; rather, it is the logical condition that indicates the start of an access. The logical equation for this condition is $\overline{\mathrm{START}} = \overline{\mathrm{CS}} + \overline{\mathrm{AS}} + R/\overline{W} \bullet \overline{\mathrm{DS}}$.
9. If a subsequent access is not a FPCP access, $\overline{\mathrm{CS}}$ must be negated before the assertion of $\overline{\mathrm{AS}}$ and/or $\overline{\mathrm{DS}}$ on the non-FPCP access. These specifications replace the old specifications 8 and 8A. (The old specifications implied that, in all cases, transitions in $\overline{\mathrm{CS}}$ must not occur simultaneously with transitions of $\overline{\mathrm{AS}}$ or $\overline{\mathrm{DS}}$. This is not a requirement of the MC68882.)

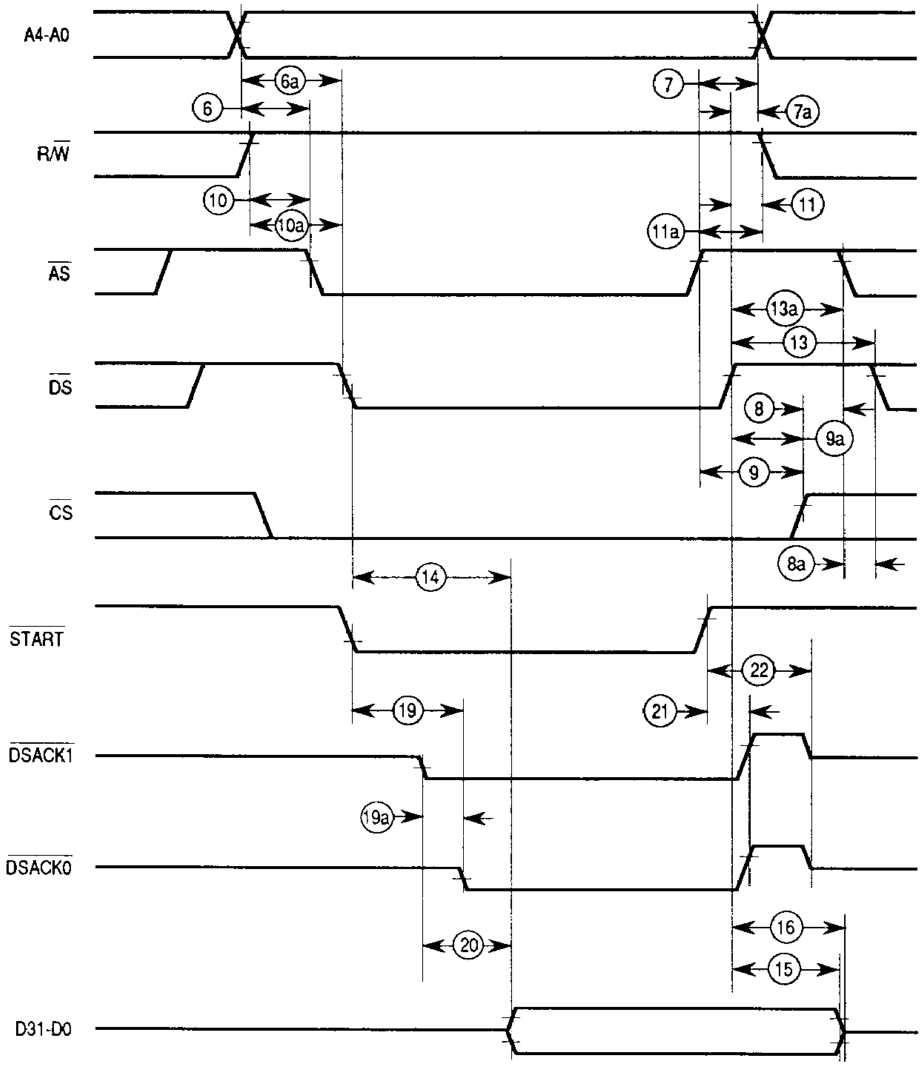


Figure 18. Asynchronous Read Cycle Timing Diagram

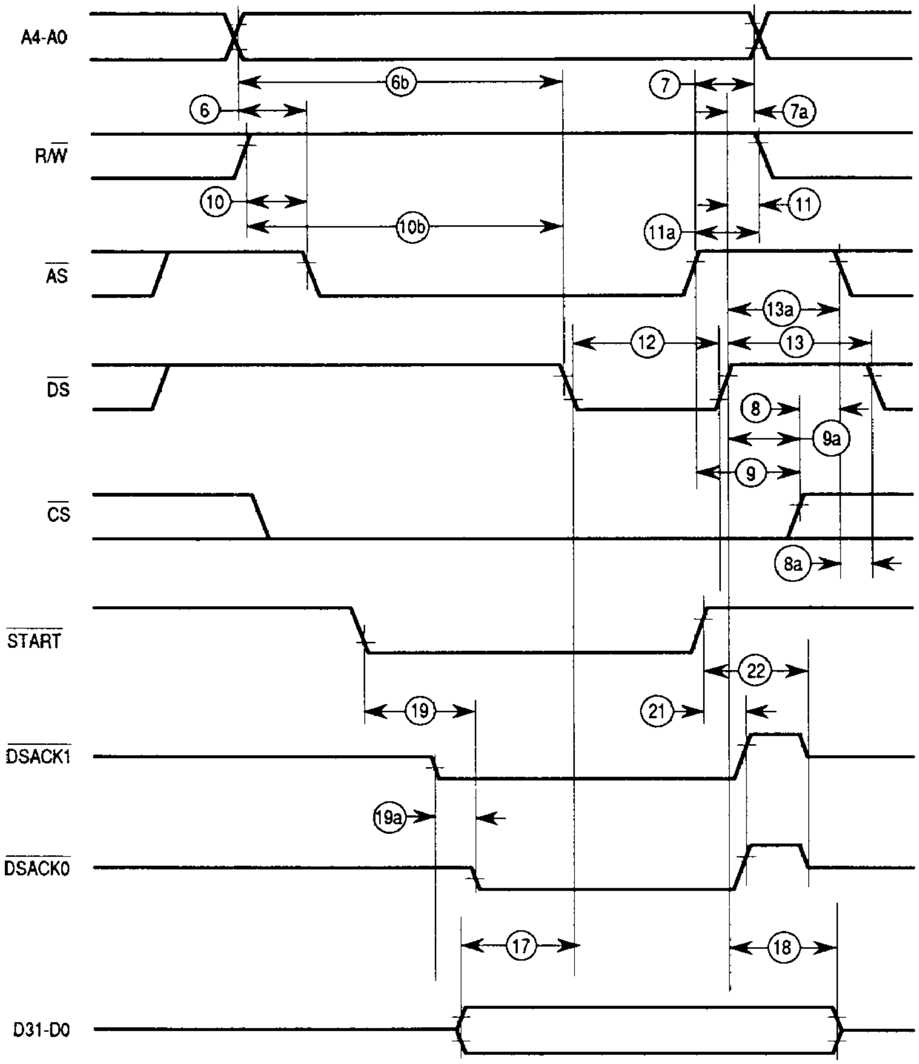


Figure 19. Asynchronous Write Cycle Timing Diagram

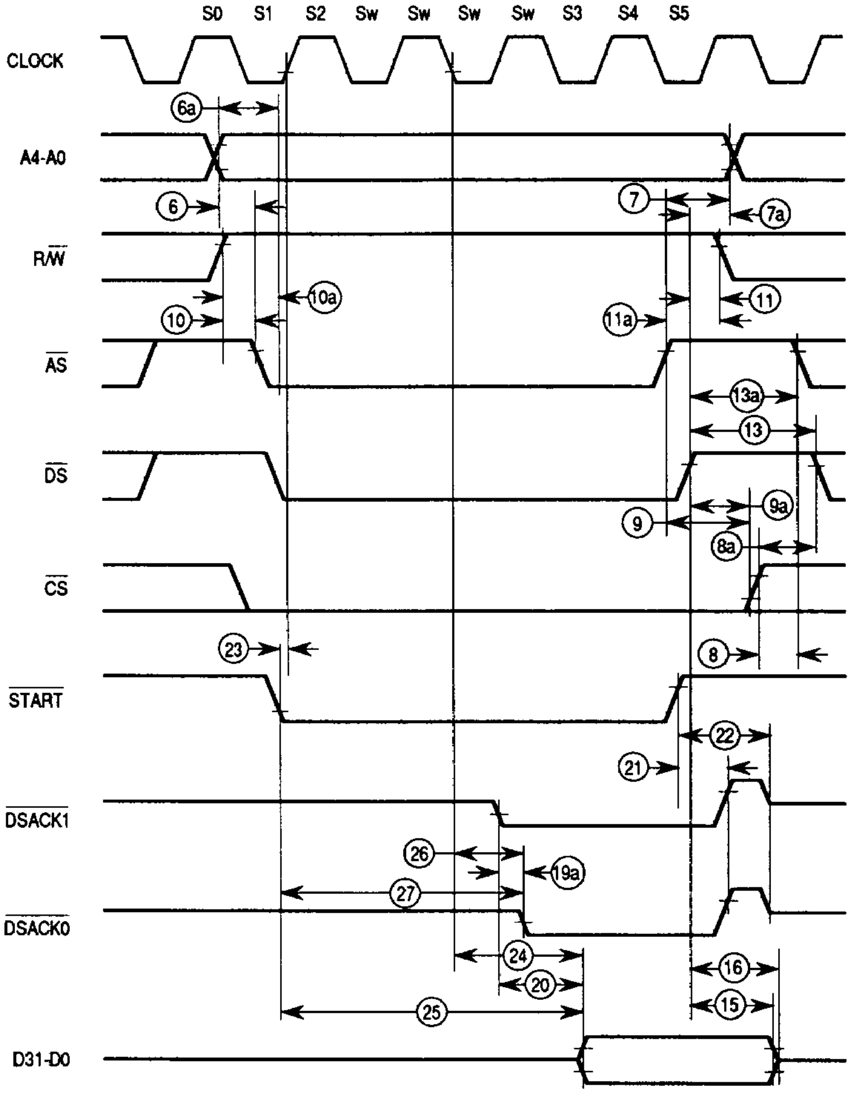


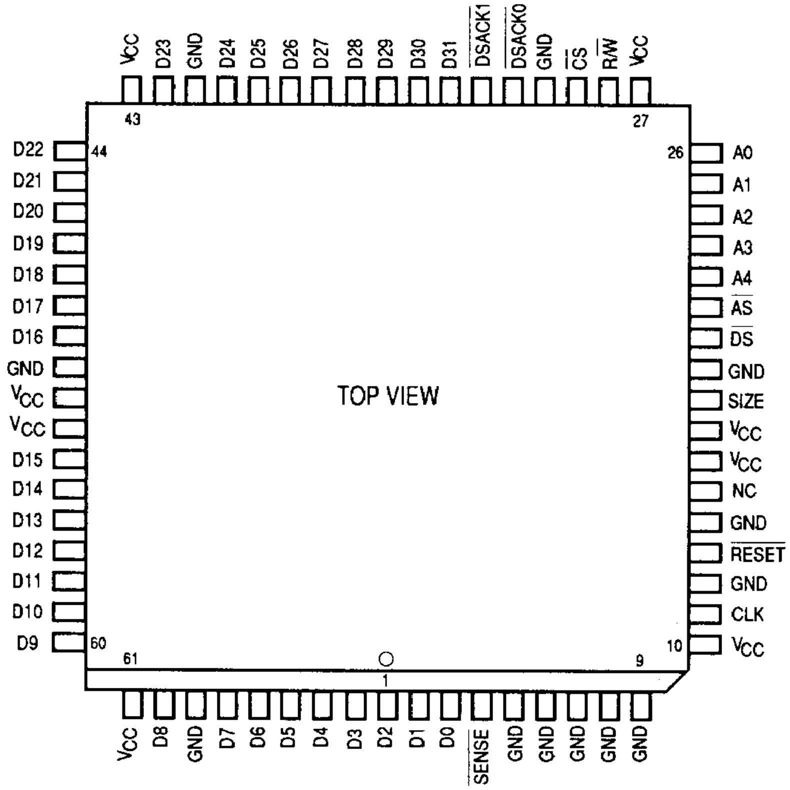
Figure 20. Synchronous Read Cycle Timing Diagram

PIN ASSIGNMENTS

68-LEAD PIN GRID ARRAY

K	A1	R/$\overline{W}$	GND	$\overline{DSACK1}$	D30	D29	D27	D26	D24	D22
J	A3	V_{CC}	$\overline{CS}$	$\overline{DSACK0}$	D31	D28	D25	GND	D23	D21
H	$\overline{AS}$	A2	A0					V_{CC}	GND	D19
G	$\overline{DS}$	A4							D20	D18
F	$\overline{SIZE}$	GND							D17	D16
E	NC	V_{CC}							V_{CC}	GND
D	$\overline{RESET}$	GND							D12	D15
C	GND	CLK	GND					D9	D13	D14
B	V_{CC}	GND	GND	$\overline{SENSE}$	D2	D5	GND	V_{CC}	D10	D11
A	V_{CC}	GND	D0	D1	D3	D4	D6	D7	D8	GND
	1	2	3	4	5	6	7	8	9	10

68-LEAD PLASTIC LEADED CHIP CARRIER


V_{CC}
D23
GND
D24
D25
D26
D27
D28
D29
D30
D31
$\overline{DSACK1}$
$\overline{DSACK0}$
GND
$\overline{CS}$
$R/\overline{W}$
V_{CC}
43
27
D22
D21
D20
D19
D18
D17
D16
GND
V_{CC}
V_{CC}
D15
D14
D13
D12
D11
D10
D9
44
60
26
10
A0
A1
A2
A3
A4
$\overline{AS}$
$\overline{DS}$
GND
SIZE
V_{CC}
V_{CC}
NC
GND
$\overline{RESET}$
GND
CLK
V_{CC}
TOP VIEW
61
1
9
V_{CC}
D8
GND
D7
D6
D5
D4
D3
D2
D1
D0
$\overline{SENSE}$
GND
GND
GND
GND
GND