Welcome to E-XFL.COM

Understanding Embedded - DSP (Digital Signal Processors)

Embedded - DSP (Digital Signal Processors) are specialized microprocessors designed to perform complex mathematical computations on digital signals in real-time. Unlike general-purpose processors, DSPs are optimized for high-speed numeric processing tasks, making them ideal for applications that require efficient and precise manipulation of digital data. These processors are fundamental in converting and processing signals in various forms, including audio, video, and communication signals, ensuring that data is accurately interpreted and utilized in embedded systems.

Applications of Embedded - DSP (Digital Signal Processors)

Details	
Product Status	Active
Type	Floating Point
Interface	SCI, SPI
Clock Rate	25MHz
Non-Volatile Memory	-
On-Chip RAM	-
Voltage - I/O	-
Voltage - Core	5.00V
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	68-BCPGA
Supplier Device Package	68-PGA (26.92x26.92)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68882crc25a

THE COPROCESSOR CONCEPT

The MC68882 functions as a coprocessor in systems where the MC68020 or MC68030 is the main processor via the M68000 coprocessor interface. 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 or MC68030 registers and instruction set in a manner which is transparent to the programmer. The programmer perceives the MPU/FPCP 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 or MC68030 and the MC68882. For optimum performance, however, the coprocessor interface allows concurrent operations in the MC68882 with respect to the MC68020 or MC68030 whenever possible. In order to simplify the programmer's model, the coprocessor interface is designed to emulate, as closely as possible, non-concurrent operation between the MC68020 or MC68030 and the MC68882.

The MC68882 is a non-DMA type coprocessor which uses a subset of the general purpose coprocessor interface supported by the MC68020 or 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 which require a trap to be taken are serviced by the main processor at the request of the MC68882.
- Support of virtual memory/virtual 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 or MC68030 as a coprocessor. This device fully supports the MC68020 or 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 is required, and HMOS structures are used where minimum silicon area is desired. Using this technology increases speed performance 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 or 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 or MC68030 programmer can view the MC68882 registers as though the registers are resident in the MC68020 or MC68030. Thus, a MC68020 or MC68030 and an MC68882 device pair functions 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 is shown in Figures 1 through 6 and consists of the following:

- 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 that contains enable bits for each class of exception trap, and mode bits to set the user-selectable rounding and precision modes.
- A 32-bit status register that contains floating-point condition codes, quotient bits, and exception status information.
- A 32-bit instruction address register that 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.

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

As shown in Figure 8, the MC68882 is internally divided into three processing elements: the bus interface unit

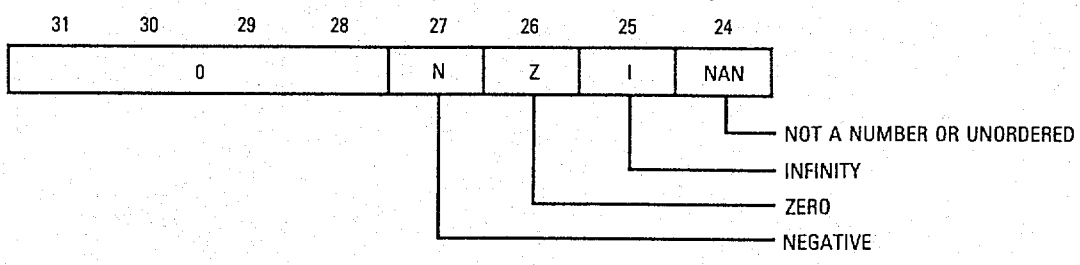


Figure 4. Condition Code Byte

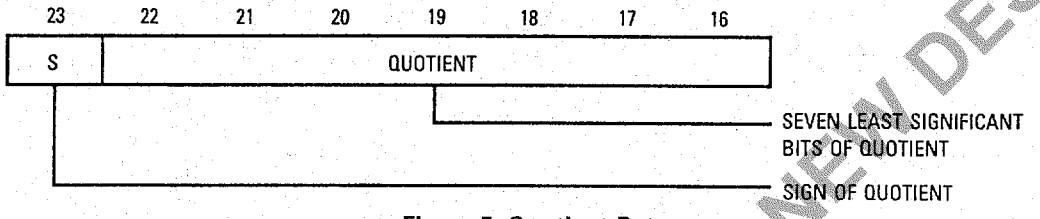


Figure 5. Quotient Byte

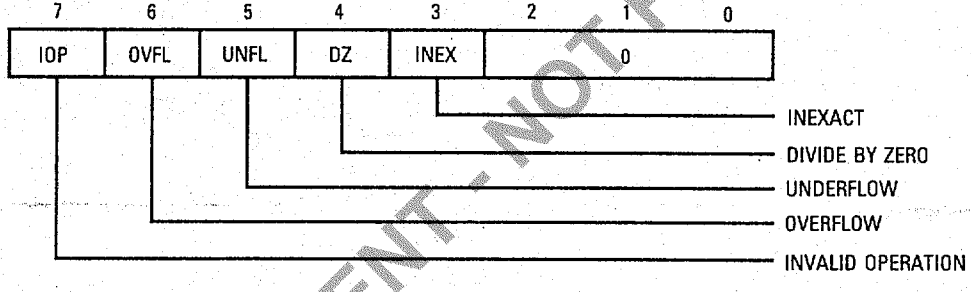


Figure 6. Accrued Exception Byte

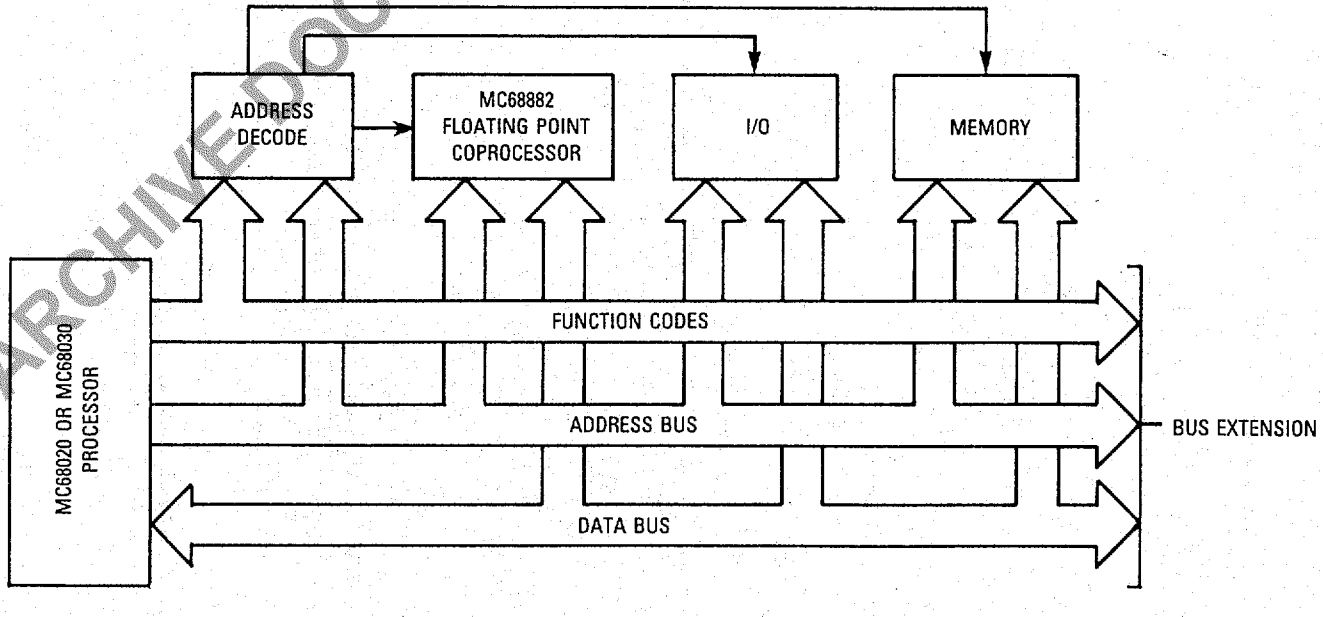


Figure 7. Typical Coprocessor Configuration

(BIU), the conversion unit (CU), and the arithmetic processing unit (APU). The BIU communicates with the MC68020 or MC68030, the CU performs data conversion for binary real data formats, and the APU executes all MC68882 instructions.

The BIU contains the coprocessor interface registers (CIRs). In addition to these registers, the register select and DSACK timing control logic is contained in the BIU. Finally, 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 to these registers, 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 bit to 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 of the special test environment supported by VLSI test equipment.

BUS INTERFACE UNIT

All communications between the MC68020 or 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 coprocessor interface registers (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 and MC68030 implement this general purpose coprocessor interface protocol in hardware and microcode.

When the MC68020 or MC68030 detects a general type MC68882 instruction, the MC68020 or 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 or MC68030 on behalf of the MC68882. For example, the response may request that the MC68020 or MC68030 fetch an operand from the evaluated effective address and transfer the operand to the operand CIR. Once the MC68020 or MC68030 fulfills the coprocessor request(s), the MC68020 or 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 or MC68030 issues a CPU address space function code during the cycle. (The function codes are generated by the M68000 Family processors to identify eight separate address spaces.) Thus, the memory-mapped coprocessor interface registers do not infringe upon instruction or data address spaces. The MC68020 or MC68030 places a coprocessor ID field from the coprocessor instruction onto three of the upper address lines during coprocessor accesses. This ID, along with the CPU address space function code, is 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 8-bit MC68008, the 16-bit MC68000, or the 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 processors to the MC68020 or MC68030. The software migrates without change to the next generation equipment using the MC68020 or MC68030.

Since the bus is asynchronous, the MC68882 need not run at the same clock speed as the main processor. Total system performance may therefore be customized. For a given CPU performance requirement, 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 or MC68030 designs. The interface partitions MPU and coprocessor operations so that the MC68020 or 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 or 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 the great majority of MC68882 instructions may be overlapped with the execution of MC68020 or MC68030 instructions, concurrency is completely transparent to the programmer. The MC68020 and MC68030 single-step and program flow (trace) modes are fully supported by the MC68882 and the M68000 Family coprocessor interface.

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

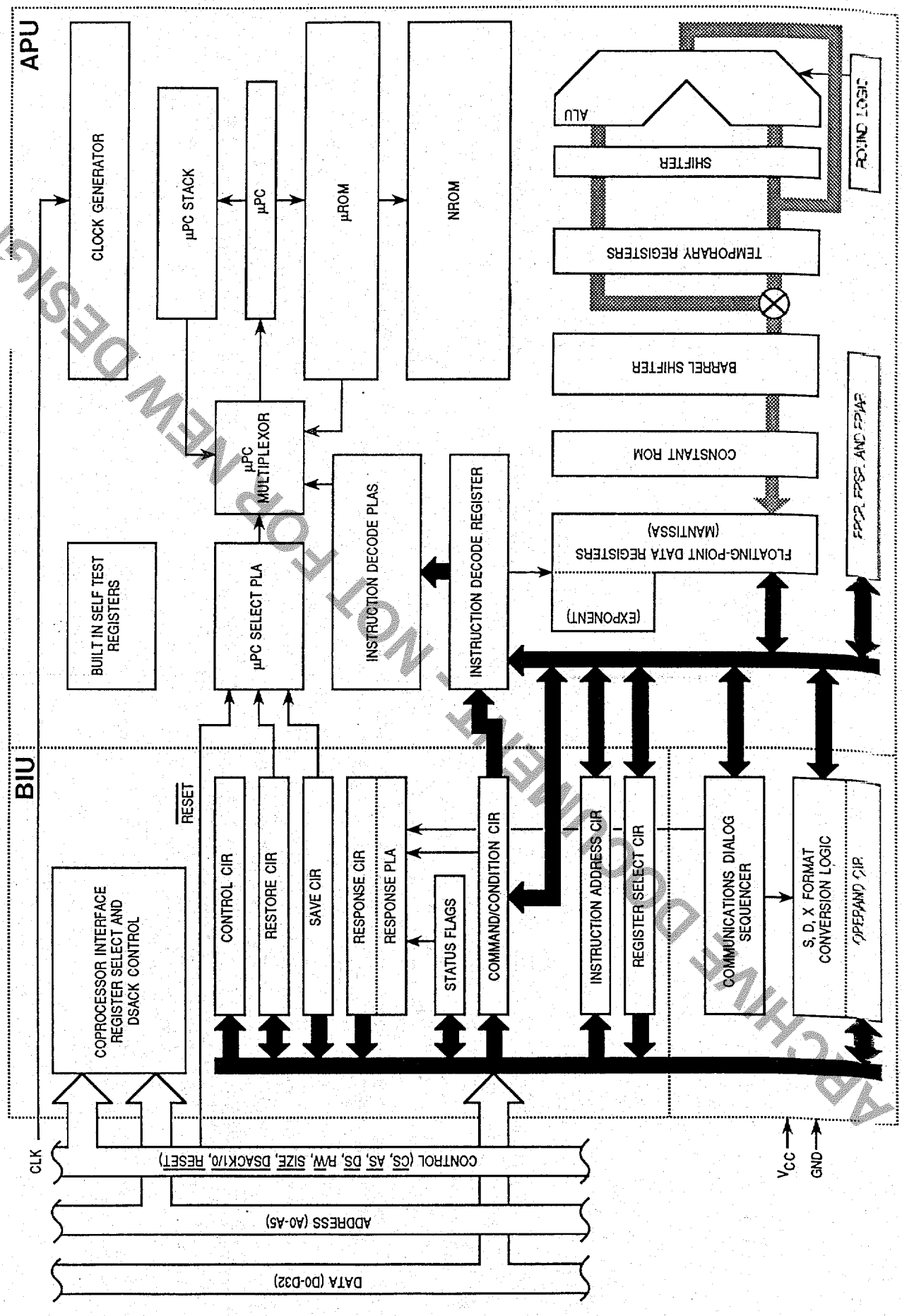


Figure 8. MC68882 Simplified Block Diagram

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

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. It is anticipated that extended precision data will not be stored in large arrays due to the amount of memory required by each value.

PACKED DECIMAL STRING REAL DATA FORMAT

The packed decimal 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). As is the case with all data formats when packed BCD strings are supplied to the MC68882, the strings are automatically converted to extended precision real values. This conversion allows 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 described above are supported orthogonally by all arithmetic and transcendental operations and by all appropriate MC68020 or MC68030 addressing modes. For example, all of the following are legal instructions:

```
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.

INSTRUCTION SET

The MC68882 instruction set is organized into six major classes:

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

MOVES

On all moves from memory (or from an MC68020 or 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 or 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:

```
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 or MC68030 effective address operand.
.<fmt> is the data format size.
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 through FP7 can be moved to or from memory with one instruction. These registers are always moved as 96-bit extended data with no conversion (hence 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

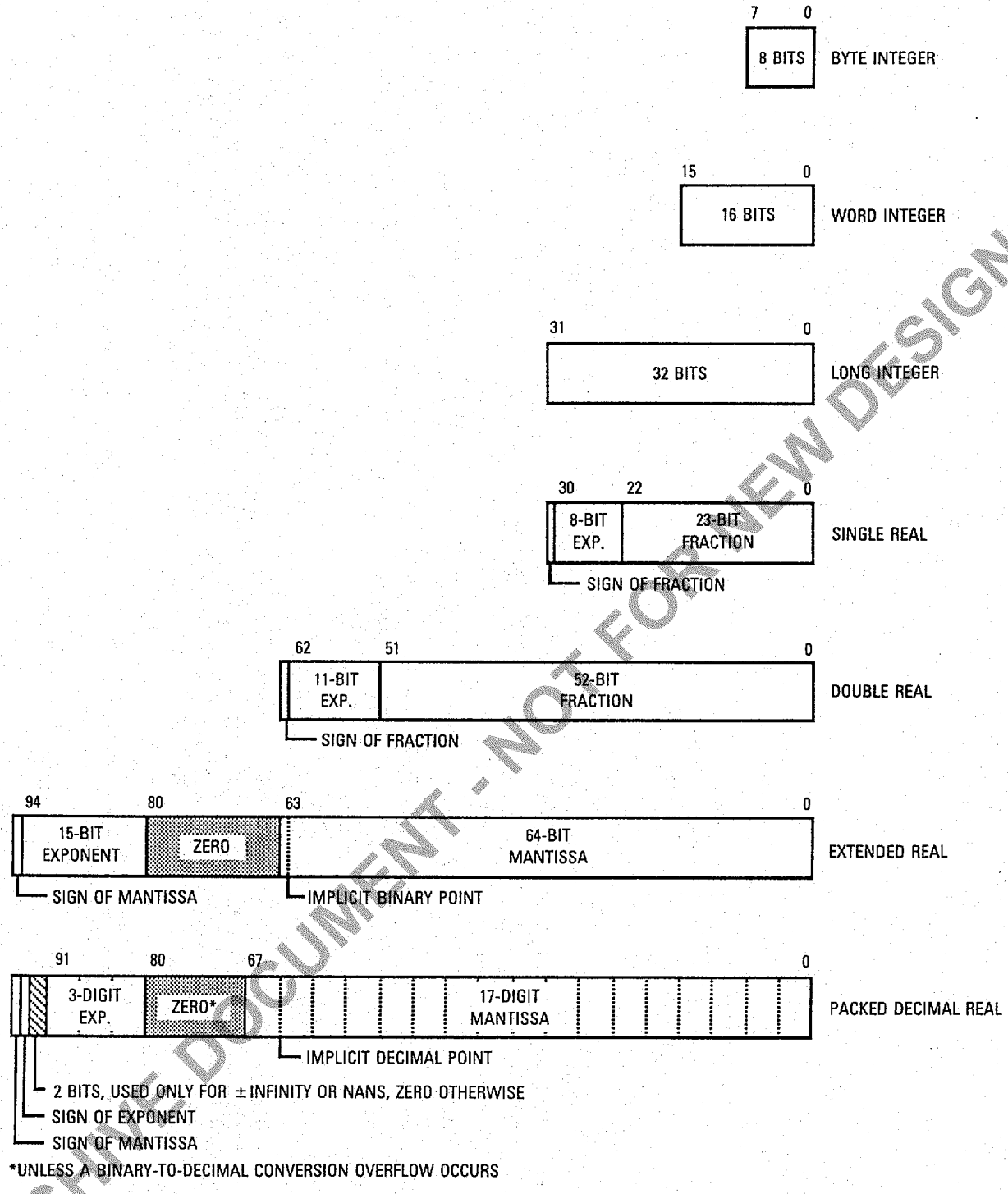


Figure 9. MC68882 Data Format Summary

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. In order 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 or MC68030 data register. The result is always stored in a floating-point data register. For example, the syntax for square root is:

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

The MC68882 monadic operations available 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 or 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:

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

The dyadic operations available with the MC68882 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 or MC68030 whether the condition is true or false. The MC68020 or MC68030 then takes the appropriate action. Since the MC68882 and MC68020 or MC68030 are closely coupled, the floating-point branch operations execute very quickly.

The MC68882 conditional operations are:

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 as given in Table 2.

Table 2. Floating-Point Conditional Test Specifiers

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

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 or MC68030 to transfer an operand via the coprocessor interface, the MC68020 or MC68030 performs the addressing mode calculations requested in the instruction. In this case, the instruction is encoded specifically for the MC68020 or 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 and 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 writer to think of the MC68882 as though it is part of the MC68020 or MC68030. There are no special restrictions 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 idle and busy state frames (generated by the FSAVE instruction) are both 32 bytes larger with the MC68882 than 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 BSUN (Branch or Set on Unordered), SNAN (Signaling Not-A-Number), OPERR (Operand Error), OVFL (Overflow), DZ (Divide by Zero) and INEX (Inexact result) floating-point exception handlers must have these minimum requirements:

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

The above 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.

FUNCTION 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.

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 through A4)

These active-high address line inputs are used by the main processor to select the coprocessor interface register locations located in the CPU address space. These lines control the register selection as listed in Table 3.

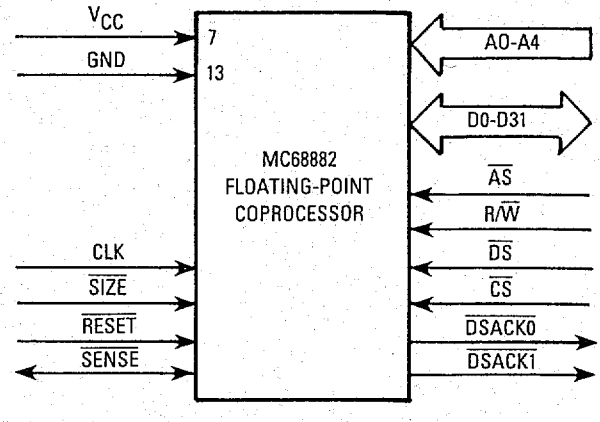


Figure 10. MC68882 Input/Output Signals

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{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{W}$	Operand
1010x	$14	16	Read	Register Select
1011x	$16	16	—	(Reserved)
110xx	$18	32	Read	Instruction Address
111xx	$1C	32	R/$\overline{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 coprocessor interface registers. When the MC68882 is configured to operate over a 16- or 32-bit system data bus, both the A0 and the $\overline{SIZE}$ pins are strapped high and/or low as listed in Table 4.

Table 4. System Data Bus Size Configuration

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

DATA BUS (D0 through 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 inter-processor 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 system data bus. Depending upon the system data bus configuration, both the A0 and $\overline{SIZE}$ pins are configured specifically for the applicable bus configuration. (Refer to **ADDRESS BUS (A0 through A4)** and **SIZE ($\overline{SIZE}$)** for further details).

SIZE ($\overline{SIZE}$)

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

ADDRESS STROBE ($\overline{AS}$)

This active-low input signal indicates that there is a valid address on the address bus, and both the chip select ($\overline{CS}$) and read/write (R/$\overline{W}$) signal lines are valid.

CHIP SELECT ($\overline{CS}$)

This active-low input signal enables the main processor access to the MC68882 coprocessor interface registers. 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/$\overline{W}$)

This input signal 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/$\overline{W}$ signal must be valid when $\overline{AS}$ is asserted.

DATA STROBE ($\overline{DS}$)

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

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

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

If the bus cycle is a main processor read, the MC68882 asserts $\overline{DSACK0}$ and $\overline{DSACK1}$ signals to indicate that the informatin on the data bus is valid. (Both DSACK 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, $\overline{DSACK0}$ and $\overline{DSACK1}$ are used to acknowledge acceptance of the data by the MC68882.

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

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 coprocessor interface registers 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 DSACK 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 both $\overline{DSACK0}$ and $\overline{DSACK1}$ high between bus cycles. In order to reduce the signal rise time, the $\overline{DSACK0}$ and $\overline{DSACK1}$ lines are actively pulled up (negated) by the

Table 5. DSACK Assertions

Data bus	A4	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	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

MC68882 following the rising edge of $\overline{\text{AS}}$ or $\overline{\text{DS}}$, and both DSACK lines are then three-stated (placed in the high-impedance state) to avoid interference with the next bus cycle.

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

This active-low input signal 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 the $\overline{\text{RESET}}$ line asserted for a minimum of four clock cycles after V_{CC} is within tolerance. This assures correct initialization of the MC68882 when power is applied. For compatibility with all M68000 Family devices, 100 milliseconds 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, the $\overline{\text{RESET}}$ line 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 kohm) 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 apropriate capacitive decoupling.

NO CONNECT (NC)

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

SIGNAL SUMMARY

Table 6 provides a summary of all the MC68882 signals described in the above 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	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

MC68882/MC68010 OR MC68030 INTERFACING

The following paragraphs describe how to connect the MC68882 to an MC68020 or 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 the A0 and SIZE pins are connected to V_{CC}.

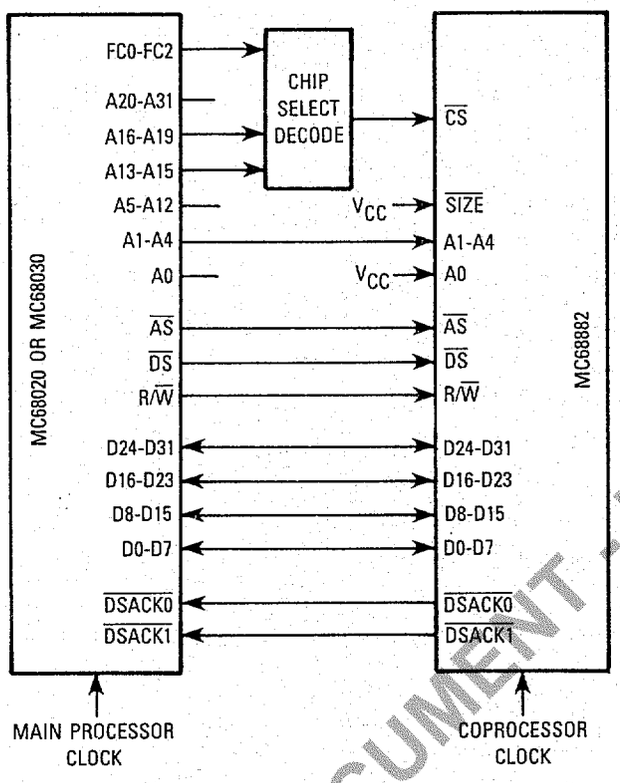


Figure 11. 32-Bit Data Bus Coprocessor Connection

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 the SIZE pin is connected to V_{CC}, and the A0 pin is connected to GND. The sixteen least-significant data pins (D0-D15) must be connected to the sixteen 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 DSACK pins of the two devices are directly connected, although it is not necessary to connect the DSACK0 pin since the MC68882 never asserts it in this configuration.

8-Bit Data Bus Coprocessor Connection

Figure 13 illustrates the connect 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 the SIZE pin is connected to GND. The twenty-four least-significant data pins (D0-D23) must be connected to the eight most-significant data pins (D24-

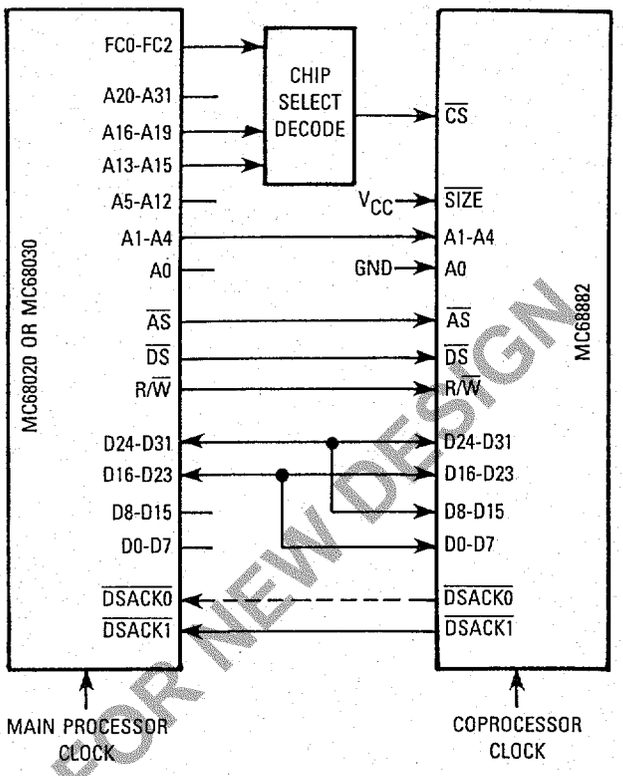


Figure 12. 16-Bit Data Bus Coprocessor Connection

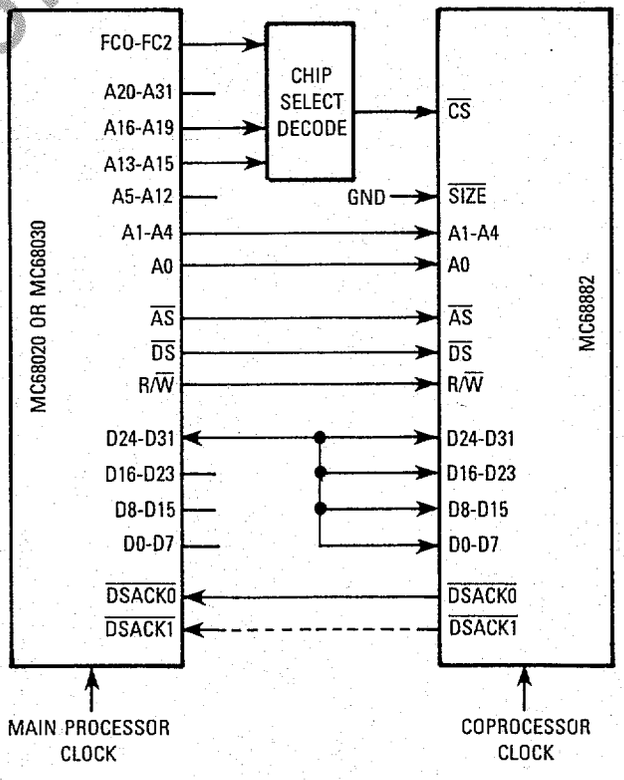


Figure 13. 8-Bit Data Bus Coprocessor Connection

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 DSACK pins of the two devices are directly connected, although it is not necessary to connect the $\overline{\text{DSACK1}}$ pin since the MC68882 never asserts it in this configuration.

MC68882-MC68000/MC68008/MC68010 INTERFACING

The following paragraphs describe how to connect the MC68882 to an MC68000, MC68008, or MC68010 processor for opertion 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 the $\overline{\text{SIZE}}$ pin is connected to V_{CC}, and the A0 pin is connected to GND. The sixteen least-significant data pins (D0-D15) must be connected to the sixteen 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 $\overline{\text{DSACK1}}$ pin of the MC68882 is connected to the $\overline{\text{DTACK}}$ pin of the main processor, and the $\overline{\text{DSACK0}}$ pin is not used.

When connected as a peripheral processor, the MC68882 chip select ($\overline{\text{CS}}$) decode is system dependent. If the MC68000 is used as the main processor, the MC68882 $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ decode logic for such systems may be the same as in an MC68020/MC68030 system, such 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 the $\overline{\text{SIZE}}$ pin is connected to GND. The eight least-significant data pins (D0-D7) must be connected to the twenty-four 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 $\overline{\text{DSACK0}}$ pin of the MC68882 is connected to the $\overline{\text{DTACK}}$ pin of the MC68008, and the $\overline{\text{DSACK1}}$ pin is not used.

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

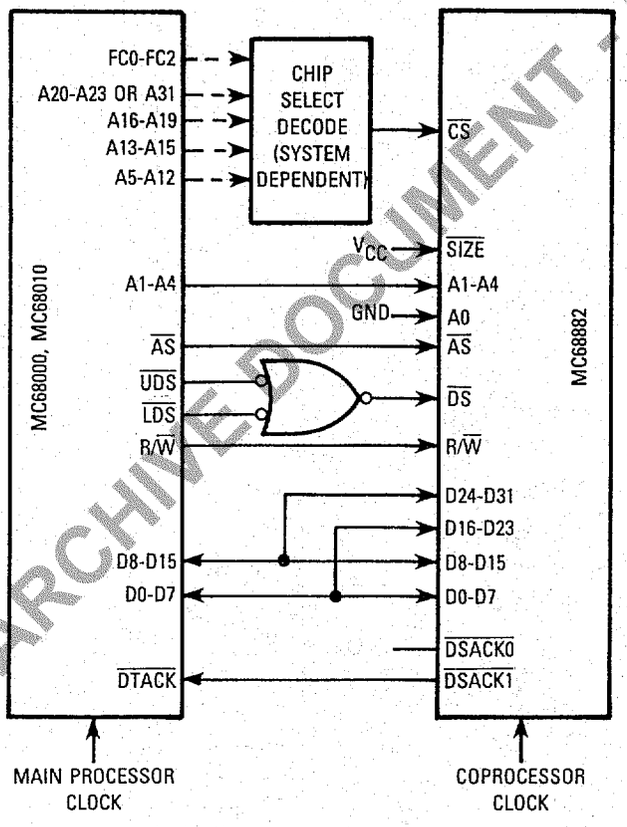


Figure 14. 16-Bit Data Bus Peripheral Processor Connection

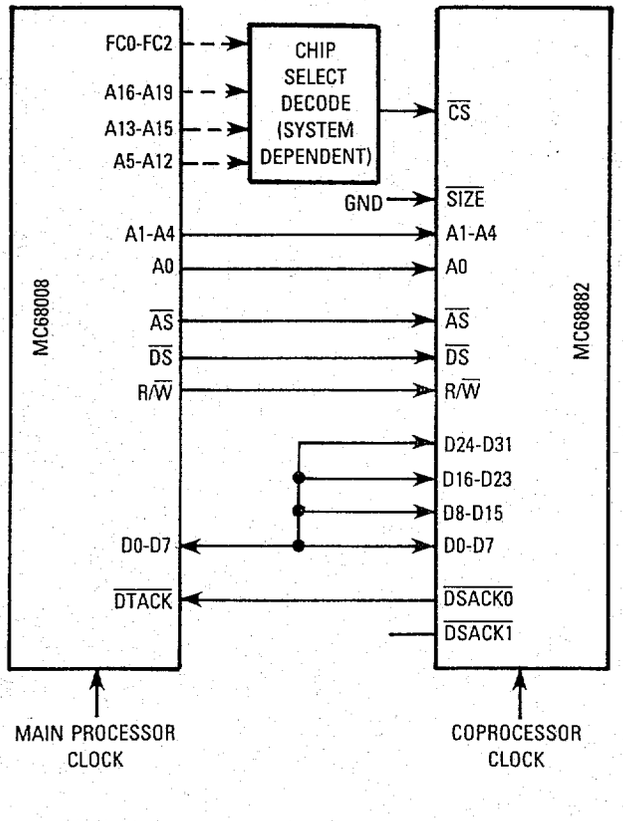


Figure 15. 8-Bit Data Bus Peripheral Processor Connection

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 Junction to Ambient Junction to Case	 θ_{JA} θ_{JC}	 33 15	°C/W

POWER CONSIDERATIONS

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

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

where:

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

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

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

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

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

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

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

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

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

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling, and thermal convention. 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.

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

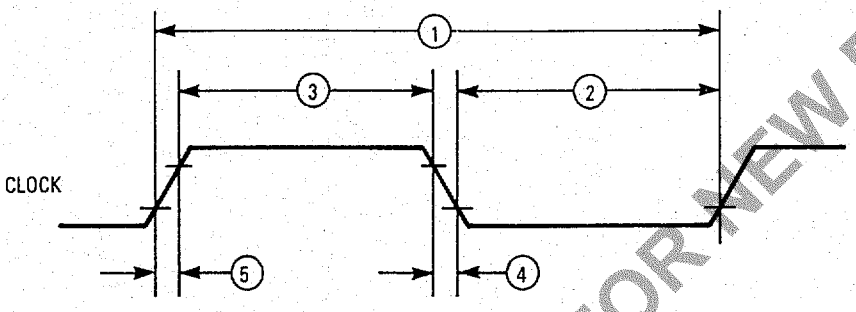
Characteristic		Symbol	Min	Max	Unit
Input High Voltage		V_{IH}	2.0	V_{CC}	V
Input Low Voltage		V_{IL}	GND −0.5	0.8	V
Input Leakage Current @ 5.25 V	CLK, $\overline{RESET}$, R/$\overline{W}$, A0–A4, $\overline{CS}$, $\overline{DS}$, $\overline{AS}$, $\overline{SIZE}$	I_{in}	—	10	μA
Hi-Z (Off State) Input 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 CHARACTERISTICS — CLOCK INPUT

(V_{CC} = 5.0 Vdc ± 5%; GND = 0 Vdc; T_A = 0 to 70°C; refer to Figure 16)

Num	Characteristic	16.67 MHz		20 MHz		25 MHz		33.33 MHz		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
	Frequency of Operation	8	16.67	12.5	20	12.5	25	16.7	33.33	MHz
1	Cycle Time	60	125	50	80	40	80	30	60	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	ns
4,5	Rise and Fall Times	—	5	—	5	—	4	—	3	ns



NOTE:

1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, 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 volts and 2.0 volts.

Figure 16. 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; refer to Figures 17, 18, and 19)

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

— Continued —

AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

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

NOTES:

1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, 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 volts and 2.0 volts.
2. These specifications only apply if the MC68882 has completed all internal operations initiated by the termination of the previous bus cycle when $\overline{DS}$ 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 $\overline{SIZE}$ pin is ***not*** strapped to either V_{CC} or GND, it must have the same setup times as do addresses.
6. If the $\overline{SIZE}$ 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 nanoseconds if $\overline{DSACK0}$ and $\overline{DSACK1}$ 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 $\overline{START} = \overline{CS} + \overline{AS} + R/\overline{W} \cdot \overline{DS}$.
9. If a subsequent access is not a FPCP access, $\overline{CS}$ must be negated before the assertion of $\overline{AS}$ and/or $\overline{DS}$ on the non-FPCP access. These specifications replace the old specifications 8 and 8A (the old specifications implied that in all cases, transitions in $\overline{CS}$ must not occur simultaneously with transitions of $\overline{AS}$ or $\overline{DS}$. This is not a requirement of the MC68882).

AC ELECTRICAL SPECIFICATION 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 20. In order to test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in Figure 20. 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 measurement 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.

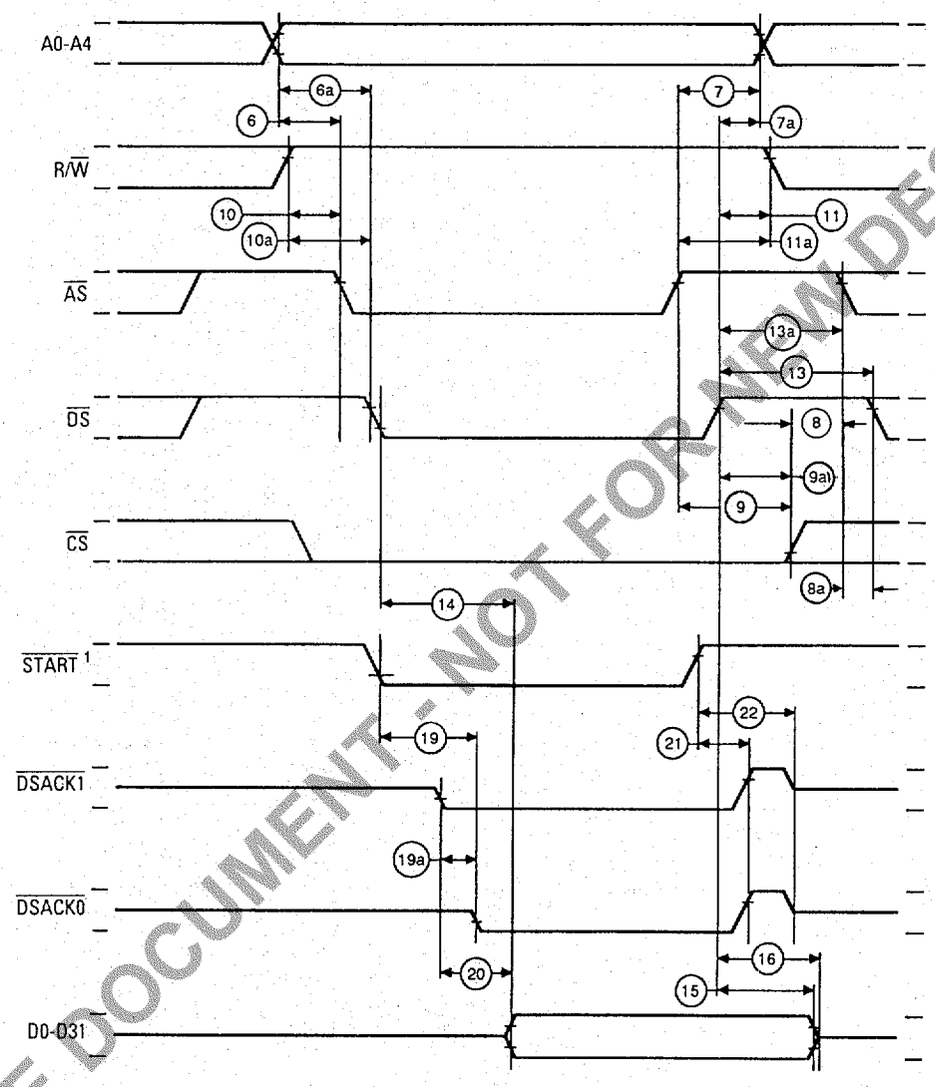


Figure 17. Asynchronous Read Cycle Timing Diagram

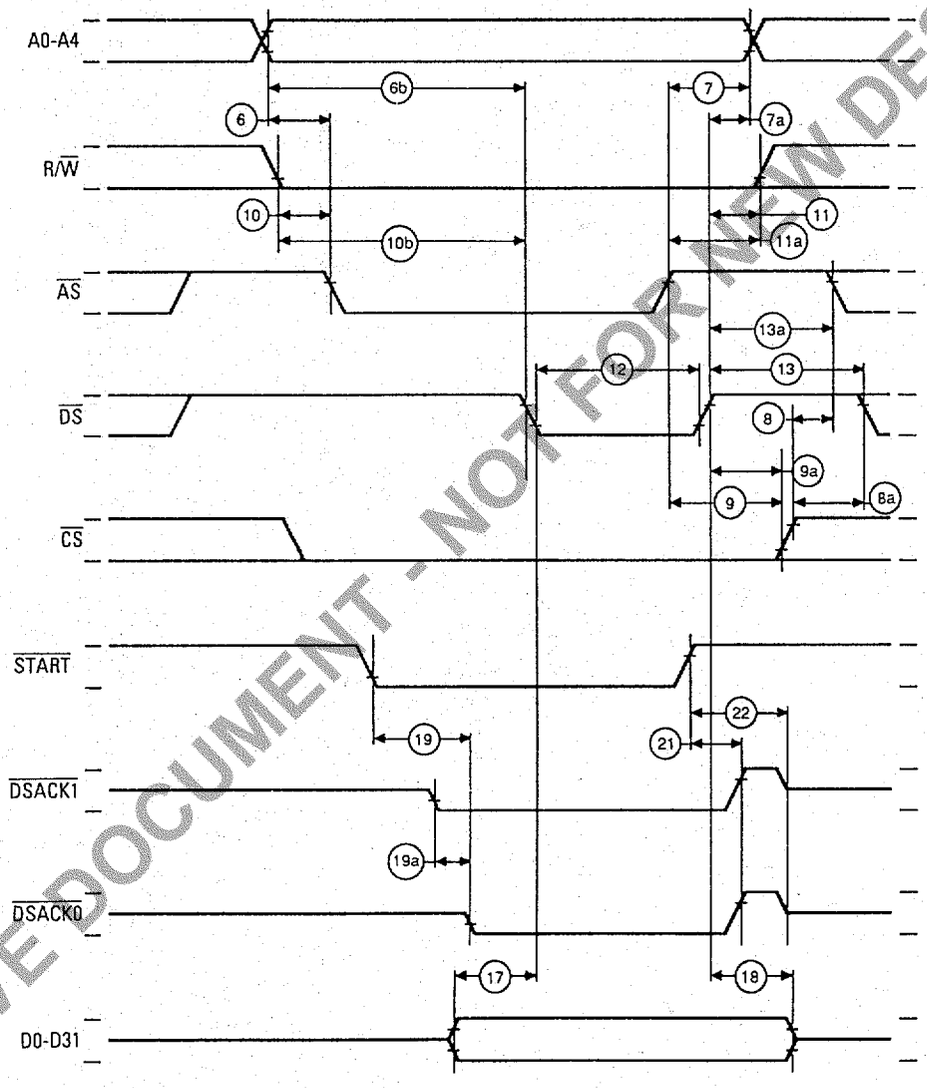
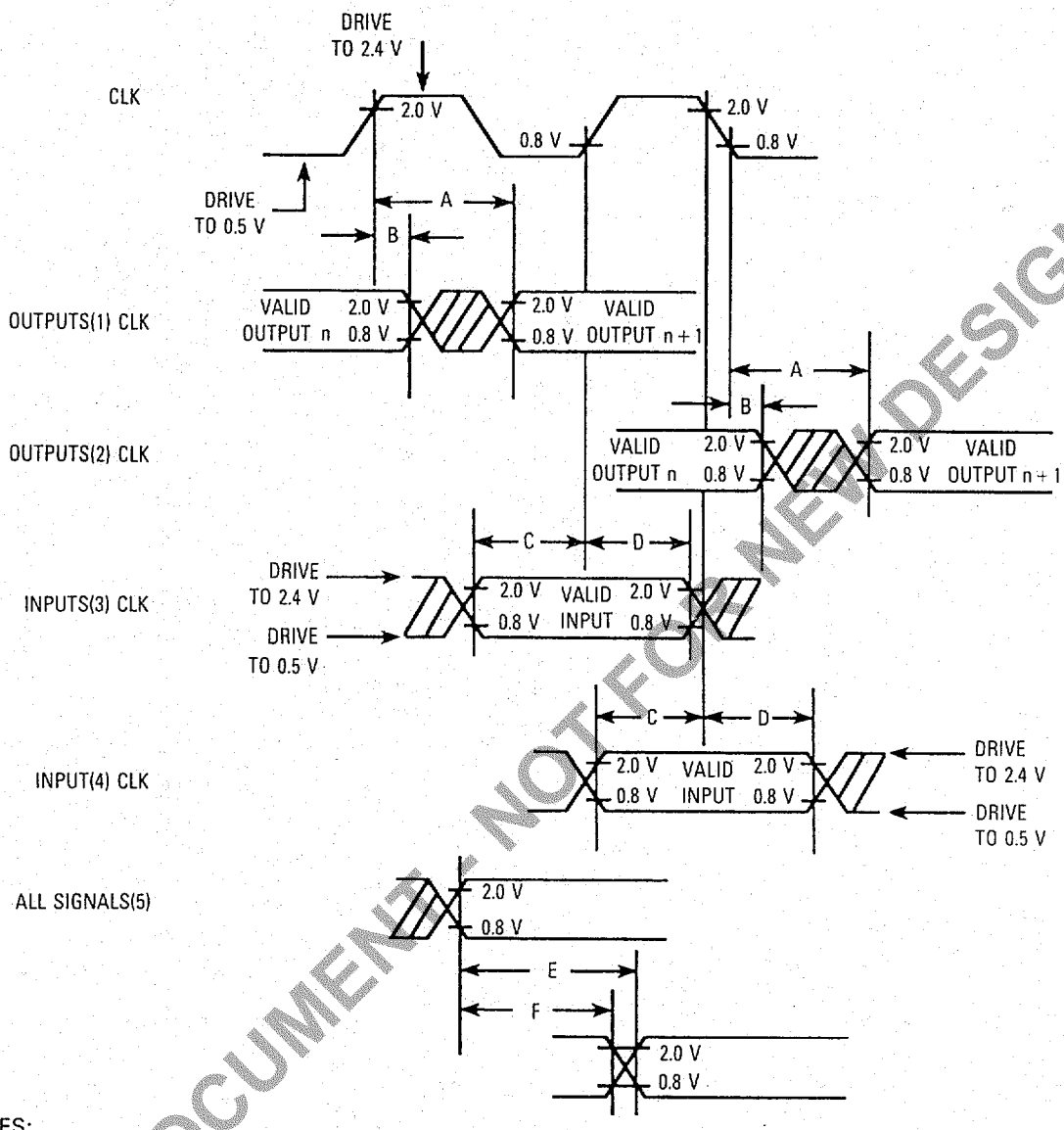


Figure 18. Asynchronous Write Cycle Timing Diagram



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.

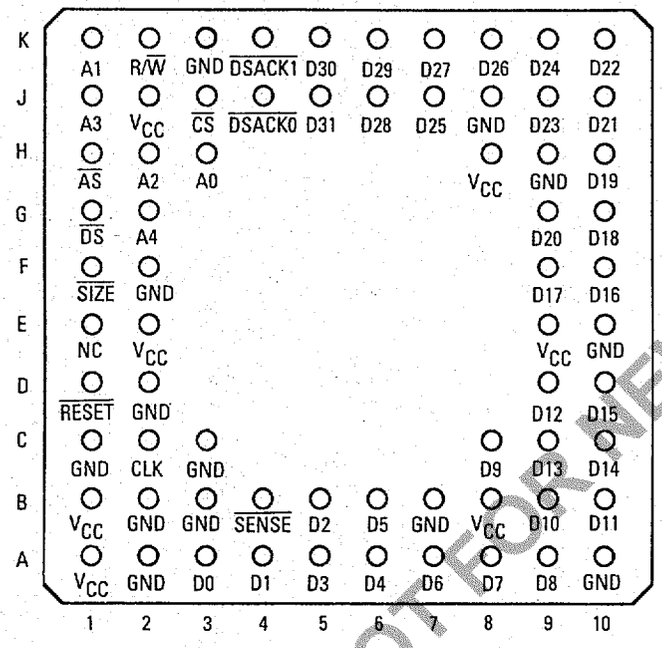
LEGEND:

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).

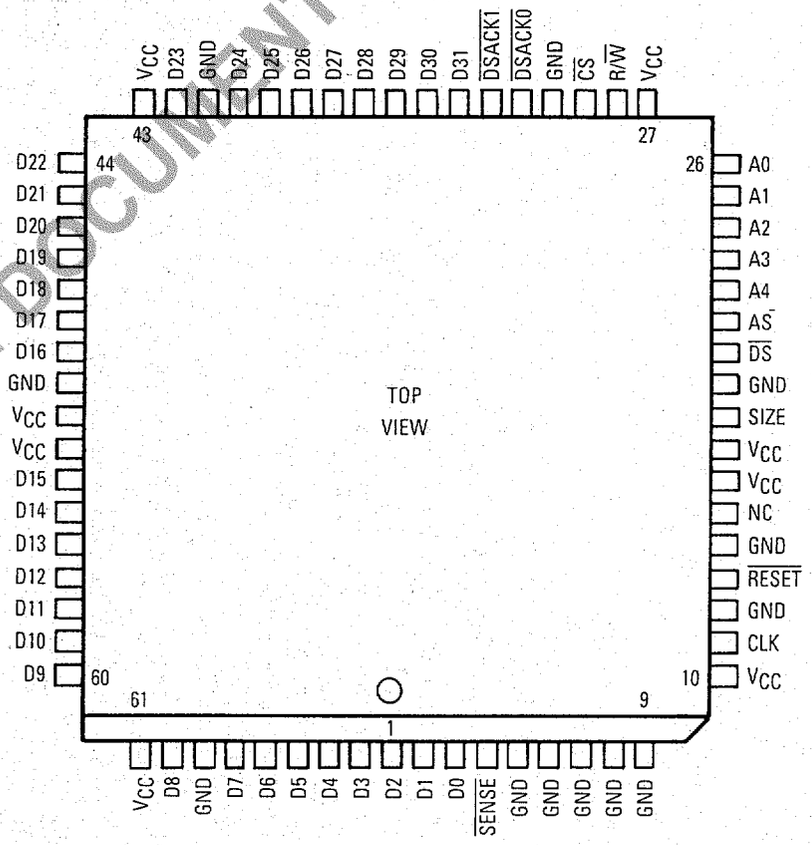
Figure 20. Drive Levels and Test Points for AC Specifications

PIN ASSIGNMENTS

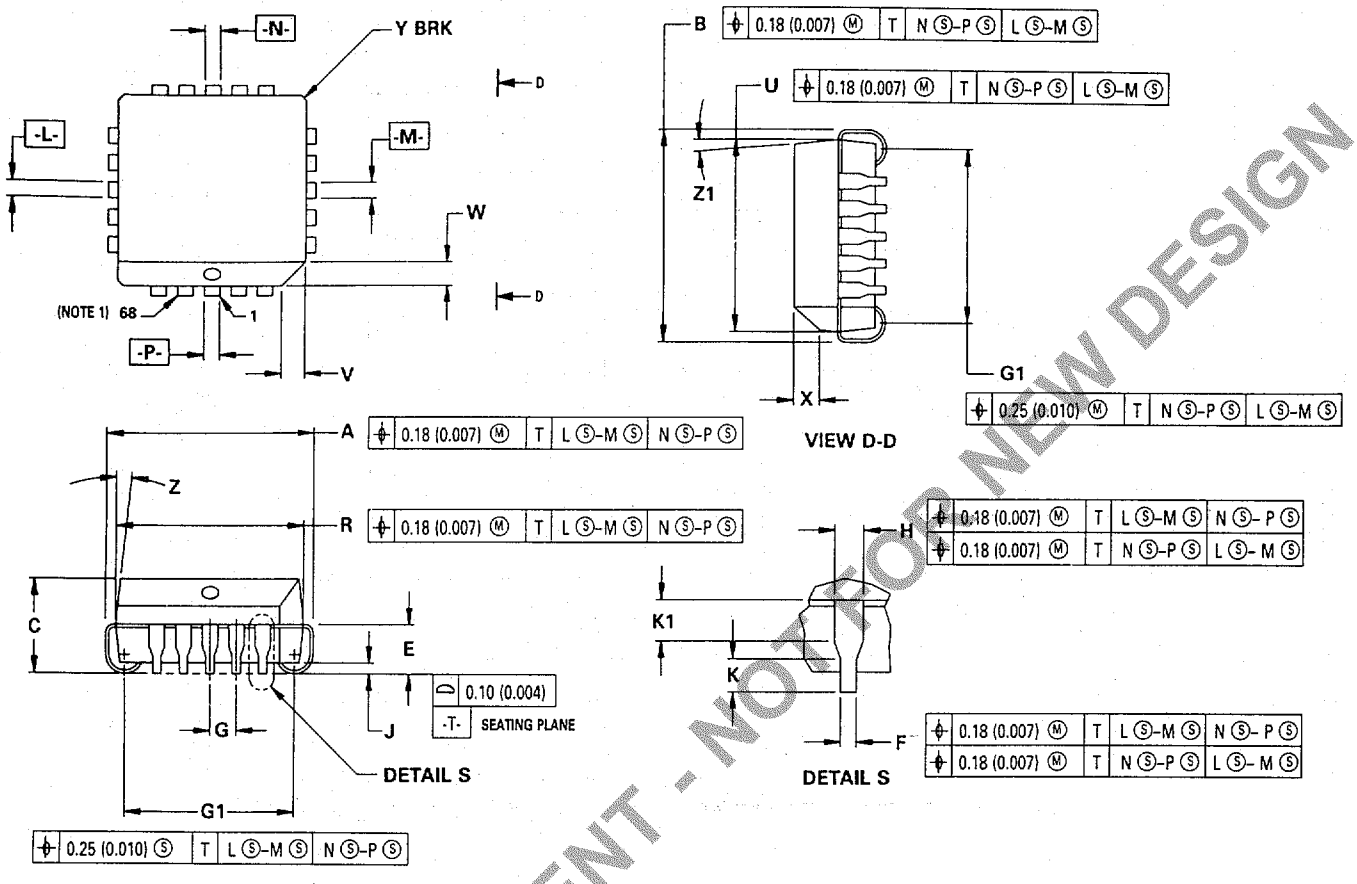
PIN GRID ARRAY



PLASTIC LEADED CHIP CARRIER



FN SUFFIX
PLASTIC LEADED CHIP-CARRIER
CASE 779-02



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	25.02	25.27	0.985	0.995
B	25.02	25.27	0.985	0.995
C	4.20	4.57	0.165	0.180
E	2.29	2.79	0.090	0.110
F	0.33	0.48	0.013	0.019
G	1.27 BSC		0.050 BSC	
H	0.66	0.81	0.026	0.032
J	0.51	—	0.020	—
K	0.64	—	0.025	—
R	24.13	24.28	0.950	0.956
U	24.13	24.28	0.950	0.956
V	1.07	1.21	0.042	0.048
W	1.07	1.21	0.042	0.048
X	1.07	1.42	0.042	0.056
Y	—	0.50	—	0.020
Z	2°	10°	2°	10°
G1	23.12	23.62	0.910	0.930
K1	1.02	—	0.040	—
Z1	2°	10°	2°	10°

NOTES:
1. DUE TO SPACE LIMITATION, CASE 779-02 SHALL BE REPRESENTED BY A GENERAL (SMALLER) CASE OUTLINE DRAWING RATHER THAN SHOWING ALL 68 LEADS.
2. DATUMS -L-, -M-, -N-, AND -P- DETERMINED WHERE TOP OF LEAD SHOULDER EXIT PLASTIC BODY AT MOLD PARTING LINE.
3. DIM G1, TRUE POSITION TO BE MEASURED AT DATUM -T-, SEATING PLANE.
4. DIM R AND U DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.25 (0.010) PER SIDE.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
6. CONTROLLING DIMENSION: INCH.