Motorola

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD AUSTIN TEXAS 78721

MC68020

Product Preview

THE MC68020 ENHANCED M68000 MICROPROCESSOR

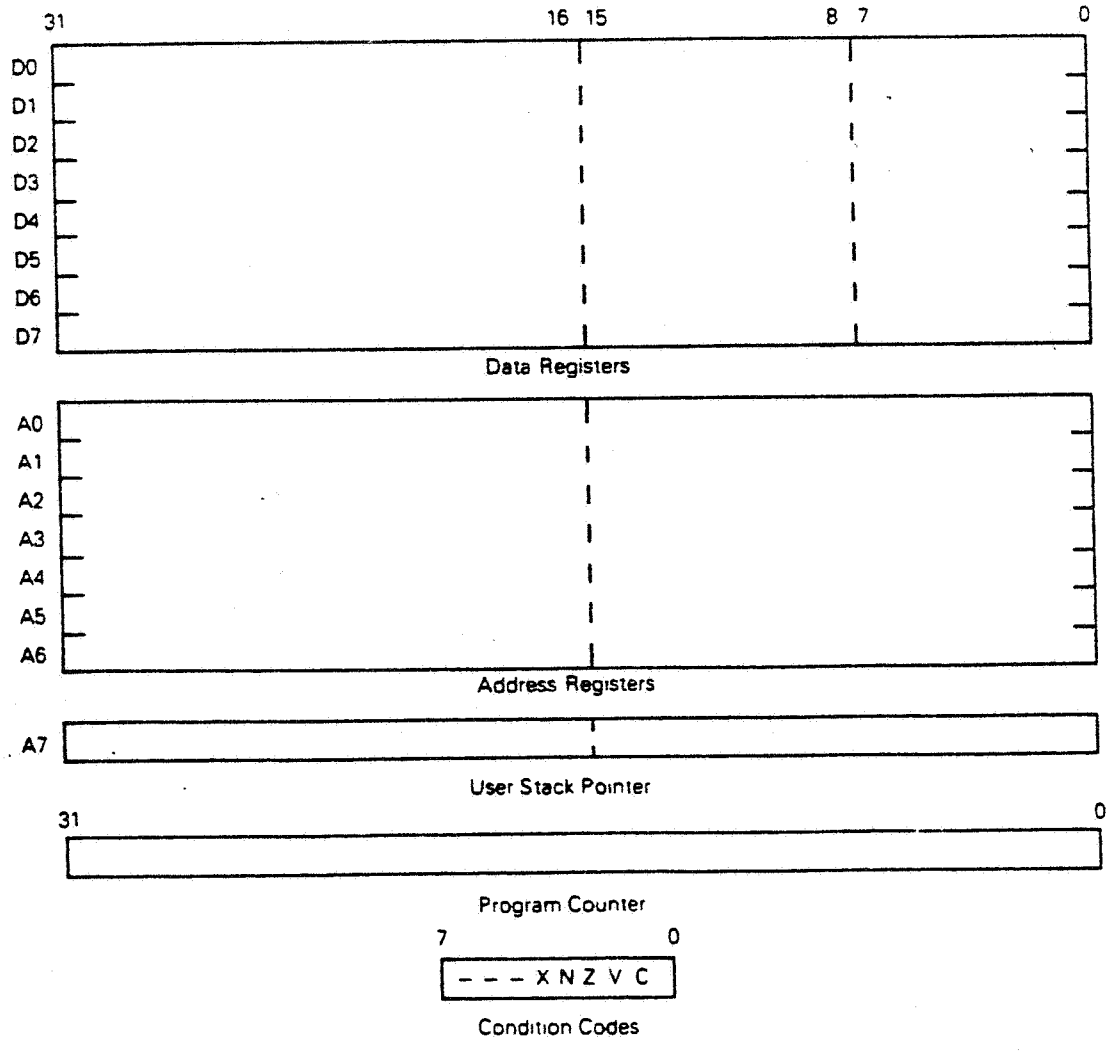
The MC68020 (Redwood) is another in a series of M68000 high performance 16/32-bit microprocessors from Motorola. Motorola's original MC68000 is the basis for this processor. This processor retains compatibility with the MC68008, MC68000 and the MC68010.

The MC68020 is the first true 32-bit microprocessor. The power of the MC68020 allows its use in any application; even those previously limited to mainframes. Just as the MC68000 set the industry standard for 16-bit microprocessor performance, the MC68020 will be the standard for all 32-bit systems.

Motorola did not stop at merely rounding out the 32-bit architecture, but added instruction set enhancements (including floating point operations), co-processor operations, improved operating system support, cache for performance advantages, and innovative techniques to improve bus efficiency.

The MC68020 is designed to accommodate M68000 co-processors through a special co-processor interface and using new co-processor commands. Use of co-processors extends the capability of an M68000 system beyond the limits of a single processing element to numerous tightly-coupled processing units, each of which may be tailored to a particular data type, task, performance, etc. To further enhance special operations in an M68000 system, an instruction cache is included in the MC68020. This cache will allow short repeated instruction streams to execute more quickly while leaving the external buses available to other processors.

FIGURE 1 — USER PROGRAMMING MODEL



As an Instruction Set Processor the MC68020 supports many data types, addressing modes, and includes structures and instructions to aid compilers and operating systems. The MC68020 includes ASCII and bit field operations, expanded addressing modes, and many new instructions including improved and new system operations. To increase throughput, the MC68020 will also incorporate reduced instruction cycle count and improved bus timing.

The User Programming Model for the MC68020 is shown in Figure 1. The Supervisor Programming Model additions are shown in Figure 2.

There are many techniques for increasing both processor and system throughput. Increasing processor performance may be a simple matter of using a processor with a higher rated clocking frequency. Increasing system performance not only encompasses such raw processor improvements, but also includes means of getting more intelligent units operating concurrently on the same or different tasks.

Parallel co-processors are one means of achieving increased system performance. Another technique involves providing a small, very fast, local (perhaps on-board) piece of memory from which data or instructions may be fetched. This cache allows the processor to operate within immediate resources. This avoids delay-inducing buffering, address translations, excessive signal skewing, and the need to access secondary resources that other processors might be sharing. Consequently, the processor can execute faster, and the secondary resources are available more often to the other processors, allowing faster total throughput. The MC68020 incorporates all of these techniques.

INSTRUCTION CACHE

An instruction cache is on the MC68020. This cache is a block of memory which is written to each time the microprocessor goes off-chip to fetch an opcode or extension word following an opcode. Should the need arise to re-execute a recently executed instruction, there are good chances that the entire instruction is valid within the cache. A controller detects this to be the case and simply reads the instruction information right out of the cache, foregoing the need to access off-chip resources. Such a case arises frequently in code which makes liberal use of short branches and loops.

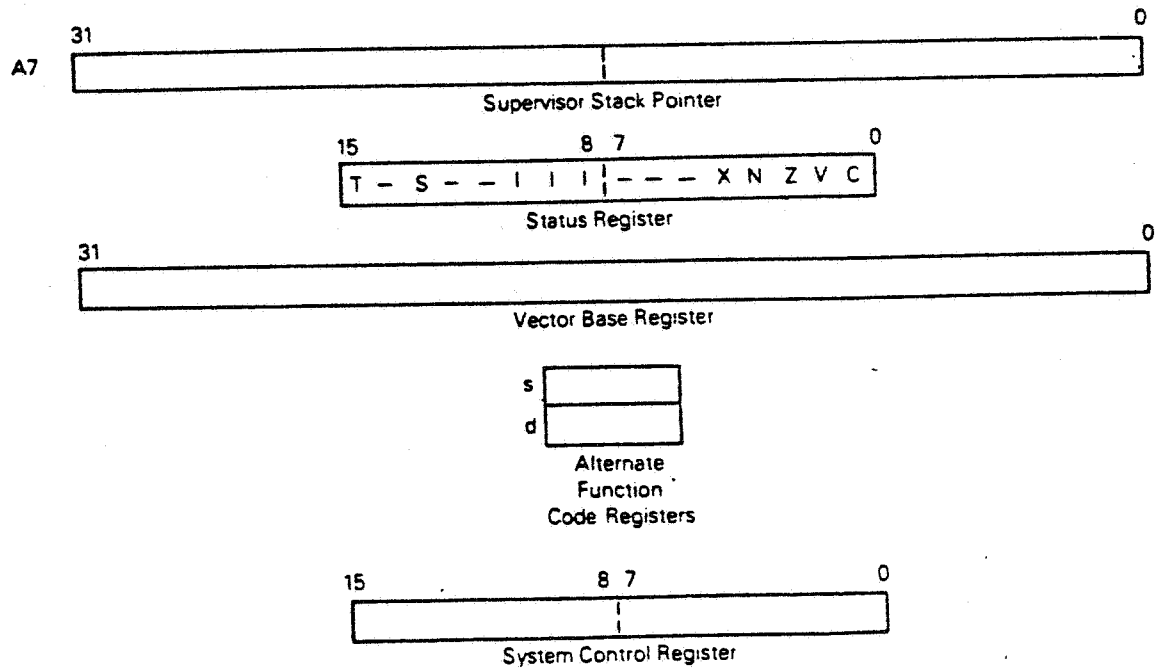
The instruction cache holds only opcodes and extension words. Therefore, any time memory data is required by an instruction (no matter where the instruction was fetched from) external resources will be accessed. This assures that only the latest data is used (as in system where another processor also has access to, and may modify, the same data). A general picture of the cache in the MC68020 is shown in Figure 3.

The cache in the MC68020 is quite useful because it not only speeds up the execution of instructions found valid in the cache (processor performance increase), but also because it leaves the external data bus free for use by other processors or DMA controllers in the same system (system performance increase).

Many repeated operations will perform much quicker in the MC68020 because of the instruction cache. A complete set of string instructions in a microprocessor can not be attained. A partial set of string instructions tends to be very limiting and only occasionally useful. But with the M68000 instructions such as MOVE and DBcc, and knowing that the cache retains instructions, custom, user-written string operations will execute faster than any other microprocessor, and will perform exactly the function the user desires. Only accesses to fetch and write data will be made after the initial pass through the operation.

Many other special repeated operations can be formulated by the user utilizing the MC68020 instructions. Successive approximations, matrix operations, and data building operations will perform much faster in the cache, and the buses will be used much less than with other microprocessors.

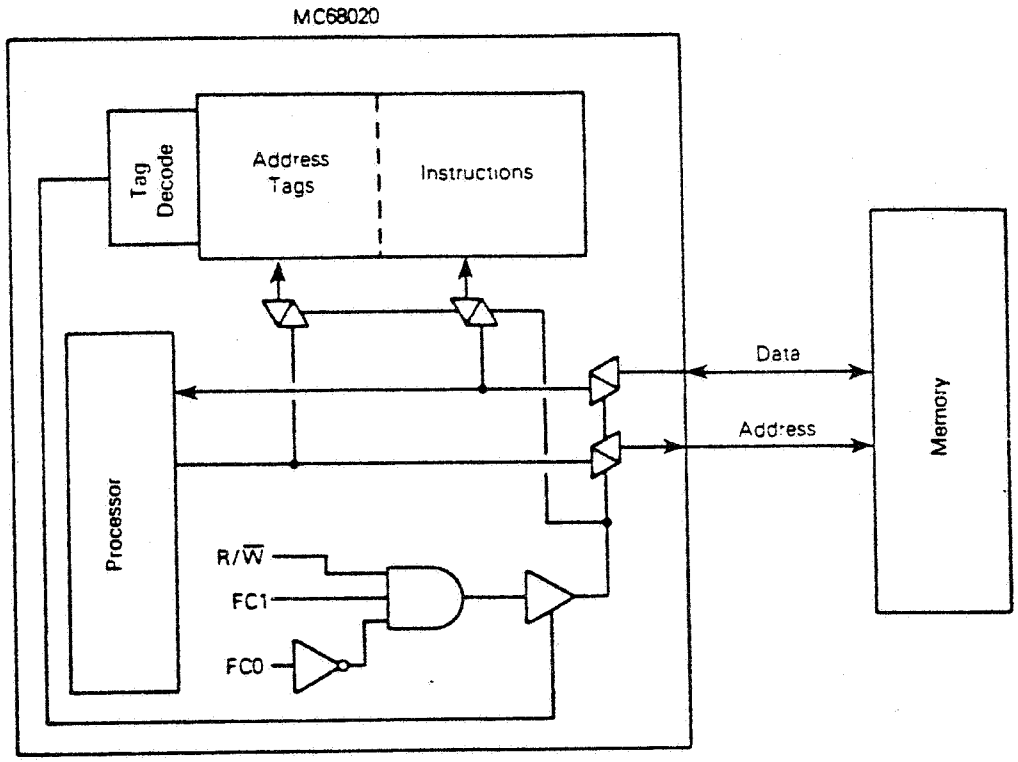
FIGURE 2 — SYSTEM PROGRAMMING MODEL SUPPLEMENT



CO-PROCESSOR OPERATIONS

The MC68020 has a M68000 co-processor interface. This co-processor interface allows the user to connect co-processors such as the MC68881 Floating Point co-processor to the MC68020. To properly operate these co-processors data and status must be passed to and from the co-processor and the CPU. The operation code line "F" (first four bits = 1111), unused in the MC68000 and MC68010, will be used for co-processor operations in the MC68020. These operations take the form of instructions but they automatically activate the special protocol that constitutes the co-processor bus interface.

FIGURE 3 — INSTRUCTION CACHE



INSTRUCTION SET

The MC68020 will execute all of the MC68000 and MC68010 instructions. Some of the instructions will execute more quickly due to streamlining of internal operations and additional resources. Additionally, the MC68020 will perform some new operations that will aid programmers in many fields, as well as simplifying operating systems and compilers.

ADDRESSING MODES

The 14 addressing modes available in the MC68000 and the MC68010 span the most necessary addressing needs. Internal address calculation is very efficient with the full 32-bit addresses being processed in Arithmetic Units all at once. In the MC68020, there are some extentions to the addressing modes.

Branches (BRA), conditional branches (Bcc), and branches to subroutines (BSR) over the entire 32-bit addressing range are available in the MC68020. The displacements used in indexed addressing can be up to a full 32-bit value. Also indexed addressing modes can utilize a new additional level of deferral in the calculation of the EA, allowing pointer tables in memory to be better supported. The addressing modes available in the MC68020 are enumerated in Table 1.

TABLE 1 — MC68020 ADDRESSING MODES

Dn	An	
(An)	– (An)	(An) +
(An + d.W)	(An + Xn)	(An + Xn.L)
(An + d.B)	(An + d.B + Xn)	(An + d.B + Xn.L)
(An + d.L)	(An + d.L + Xn)	(An + d.L + Xn.L)
(PC + d.W)	(PC + Xn)	(PC + Xn.L)
(PC + d.B)	(PC + d.B + Xn)	(PC + d.B + Xn.L)
(PC + d.L)	(PC + d.L + Xn)	(PC + d.L + Xn.L)
	((An) + Xn)	((An) + Xn.L)
((An + d.B))	((An + d.B) + Xn)	((An + d.B) + Xn.L)
((An + d.L))	((An + d.L) + Xn)	((An + d.L) + Xn.L)
	((PC) + Xn)	((PC) + Xn.L)
((PC + d.B))	((PC + d.B) + Xn)	((PC + d.B) + Xn.L)
((PC + d.L))	((PC + d.L) + Xn)	((PC + d.L) + Xn.L)
	Absolute.W	Absolute.L
Immediate.B	Immediate.W	Immediate.L
CCR	SR	
VBR	FCR	SCR

ENHANCED INSTRUCTIONS

Some of the MC68000 and MC68010 instructions in the MC68020 have greater capability, additional data types, or execute faster. These enhancements provide a more well rounded set of instructions, including the 32-bit operations.

The shift and rotate instructions in the MC68020 execute much quicker. 32-bit operands, displacements, and limits are extended to the Check, Link, Unlink, Byte extend, Multiply, and Divide instructions.

Some additional instructions have been included in the MC68020 which will simplify programming and improve performance in operating systems, compilers, and new data structures. These include Block Move to Alternate Address Space (MOVESB) for the operating system to move around parameter blocks, and a more sophisticated Enter and Leave operation for procedure transitions. Moving around variously sized bit fields will be facilitated with MOVEF instructions, and converting between ASCII and BCD will be provided in PACK and UNPK.

MOTOROLA Semiconductor Products Inc.

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC

MOTOROLA
SEMICONDUCTORS
6501 ED BLUESTEIN BLVD AUSTIN TEXAS 78723

MC68881

Product Preview

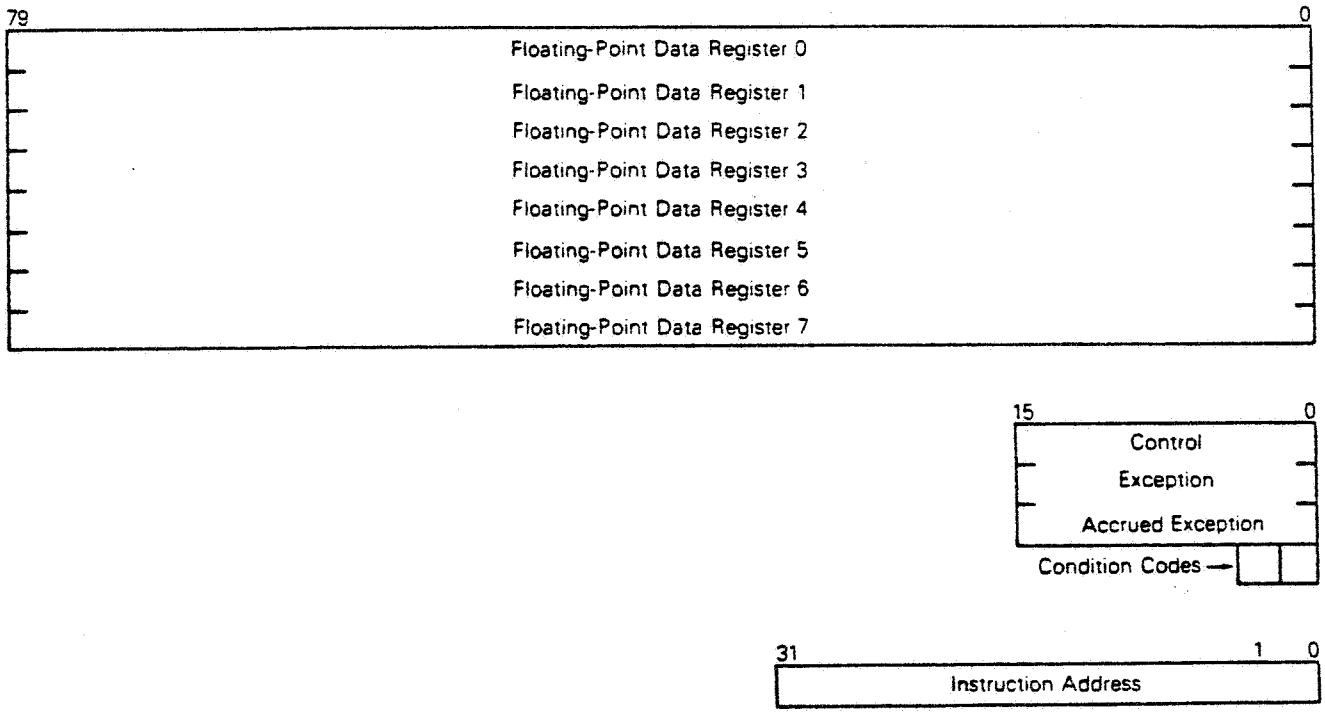
THE MC68881 FLOATING-POINT CO-PROCESSOR

The MC68881 is a high-performance HMOS floating-point processor designed to interface with the advanced MC68020 microprocessor. It can also be used as a peripheral in systems with other processors. The MC68881 is a comprehensive floating-point co-processor that provides a wide range of floating point capabilities seldom found even in a large main frame computer. System performance with the MC68020 is the overriding design goal of the MC68881.

ARCHITECTURE

The architecture of the MC68881 was defined as an extension to the architecture of the M68000 Family. It is a register-oriented processor. The programmer's model for the MC68881 is shown in Figure 1.

FIGURE 1 — PROGRAMMER'S MODEL



There are eight 80-bit floating-point data registers. These registers always hold full extended precision numbers.

The control word contains the user selectable modes. The accrued exception word contains the logical inclusive OR of the exceptions for all operations since the last clear of the accrued exception register. The exception word contains the exception(s) of the last operation only. The condition code register holds the result of the last compare instruction.

The instruction address register contains the address in main processor memory of the last instruction executed by the co-processor. This address can be used during an error trap to determine the address of the faulty instruction.

ARCHITECTURAL DETAILS

Data Types

The MC68881 incorporates four new data types. They are:

Single Precision Real (S)
Double Precision Real (D)
Double-Extended Precision Real (X)
Packed Real BCD String (P)

In the assembly language syntax these new data types are handled in the same manner as the existing byte, word, and long word data types. The suffixes S, D, X, and P are appended to the opcode.

Operation Types

The operations on the MC68881 can be broken into five major types. They are:

Dyadic Operations (2 operands)
Monadic Operations (1 operand)
Moves and Conversions
Conditional Tests
Control Operations

Dyadic Operations – All dyadic operations have as their source argument a MC68020 memory location, a MC68020 data register, or a floating-point data register. The source is converted to double-extended precision, if not already such. The destination argument is always a floating-point data register. The result is returned to the floating-point data register defined as the destination argument.

Monadic Operations – The monadic instructions only have one argument. It is either in MC68020 memory, an MC68020 data register, or in a floating-point register. It is always converted to double-extended precision format, if it is not already. The destination is always a floating-point register.

Moves and Conversions – Conversion to double-extended precision format is implicit in the move-in portion of the dyadic or monadic operation. Similarly, data contained in floating-point registers may be converted to other formats as operands are moved out of the MC68881.

Conditional Test – The conditional instructions are the FBcc and FScc which are identical to the M68000 Family instructions Bcc and Scc except they use the MC68881's condition codes for determining the truth of the condition.

Control Operations – The control instructions are used to set modes in the control register and to read the exception, accrued exception, and instruction address registers.

Co-Processor Interface

The co-processor interface designed by Motorola is an integral part of the design of both the MC68020 and the MC68881 design. The interface is clean and simple with the MC68020 and MC68881 sharing the tasks of the interface. The MC68020 provides services for the MC68881 at the co-processor's request. The services provided by the MC68020 are the ones done more efficiently by the main processor.

On the other hand, the MC68881 does not depend on the MC68020 for all services as do some co-processor schemes. Once the MC68020 has provided the services requested by the MC68881 (which may be none) it is free to continue processing. Thus the choice of concurrency or non-currency is determined on an instruction-by-instruction basis and is determined by the co-processor. The great majority of MC68881 instructions are in fact overlapped in execution with MC68020 instructions.

Since the co-processor interface is simple and flexible, it opens up the possibility of user-created co-processors. For this and other reasons the co-processor interface allows multiple co-processors in a system. Furthermore, the same handshaking that occurs between the main CPU and the co-processor can be simulated in software on CPUs that do not have the co-processor interface, by treating the MC68881 as a peripheral.

Lastly, the co-processor interface was designed with the ever growing M68000 Family in mind. The MC68881 is fully compatible with all future and existing M68000 parts including the M68450 DMA Controller, the M68451 Memory Management Unit, and the MC68020's cache memory. It also supports true virtual memory.

Implementation

The MC68881 is a microcoded processor whose complexity is on the order of the MC68020 itself. It will be built using Motorola's advanced HMOS III process.

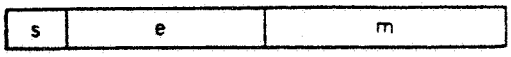
The hardware consists of a high-speed 67-bit ALU for manipulating mantissa bits. The hardware also includes a barrel shifter that can shift from 1 bit to 67 bits in one machine cycle. The barrel shifter not only speeds up standard arithmetic functions, but is also a fundamental part of transcendental function implementation. Since argument reduction for transcendental functions will be performed by the microcode, the number of functions provided will be dependent on the available microcode space.

THE IEEE FLOATING POINT STANDARD

The MC68881 is a conforming implementation of the proposed standard. In fact it not only supports all the *required* features and functions of the proposed standard, but also implements most of the *suggested* features as well. Further, the MC68881 conforms without the need for any software external to the processor. All operations take place in high-speed hardware.

Data Format Conformance

The MC68881 supports three data sizes defined by the proposed standard. They are single, double, and double-extended. (The single-extended type is redundant when these three are included; all references in this document which refer to "extended" imply "double-extended".) The format for all three data types has the basic organization of:



where:
s = sign
e = exponent
m = mantissa

The sizes of each field for the three floating-point formats are:

	Size in Bits		
	Single	Double	Extended
Sign	1	1	1
Exponent	8	11	15
Mantissa	23	52	64
Total	32	64	80

The three formats described above are the formats which are assumed by floating-point numbers in user's memory. Each time one of these numbers is transferred to the MC68881 it is converted into an extended real number. Thereafter, all operations in the co-processor take place with full extended precision. Even integers and BCD strings are converted into 80-bit numbers when they are loaded into an MC68881 data register. This means that the MC68881 supports mixed mode arithmetic.

Data Type Conformance

The proposed standard requires that not only must normalized numbers be recognized, but that special data types must also be recognized. The largest and smallest exponents are reserved for these special data types:

- Positive True Zero
- Negative True Zero
- Plus Infinity
- Minus Infinity
- Denormalized Numbers
- Not-a-Number (NaN's)

Operation Conformance

All operations specified by the proposed standard are supplied in full precision by the MC68881. The arithmetic operations provided are:

- Add
- Subtract
- Multiply
- Divide
- Remainder
- Compare
- Square Root
- Integer Part

MODE CONFORMANCE

Rounding Modes – The MC68881 supports all four rounding modes specified in the standard:

- Round to Nearest
- Round Towards Plus Infinity
- Round Towards Minus Infinity
- Round Towards Zero

Rounding Precisions – Even though the MC68881 does all arithmetic to full 80-bit precision, sometimes it is desirable to round the 80-bit result to the precision of a single or double result. The three choices are:

- Round to Extended (Default)
- Round to Double
- Round to Single

Infinity Closures – Two types of infinity closures are also defined by the standard and supported in the MC68881. *Affine* closure defines a number system where both plus and minus infinity exist and are at opposite ends of the number line.

In *projective* closure, infinity is unsigned and the number system can be thought of as a circle which includes all numbers.

Error Handling Conformance

The proposed standard provides for the hardware to trap if an error occurs. On the MC68020/MC68881 if an error occurs on an enabled trap, the MC68881 will signal the MC68020 to take a trap and will supply a vector number. In other words, floating point exception traps are handled just like any other MC68020 traps. No external glue parts are required and there is no possibility of dead-lock.

BEYOND THE IEEE PROPOSAL

The MC68881 offers many features and functions beyond those required or suggested by the IEEE.

Additional Instructions

Some of the additional instructions provided in the MC68881 are:

- Absolute Value
- Negate
- Scale Exponent
- Set Byte determined by Floating-Point Condition
- Branch on Floating-Point Condition
- Get Index Based on Floating-Point Type
- Move Constant to Floating-Point Register
- Get Fraction of Floating-Point Number
- Get Exponent of Floating-Point Number
- Modulo

Transcendentals

The MC68881 includes on-chip hardware for evaluation of transcendental functions. The functions planned are:

- Sine x
- Cosine x
- Arc Tangent x
- Log Base 2
- e^x
- Log Base e

The following functions will also be provided if there is adequate space in the microcode after the above functions are included:

- Tangent x
- Hyperbolic Arc Tangent
- Hyperbolic Sine
- Hyperbolic Cosine
- Hyperbolic Tangent
- Log Base 10
- Log Base 2
- 10^x
- y^x

Each of these functions is calculated to double-extended precision.

SUMMARY

The MC68881 is the most comprehensive floating-point processor. It provides all the required functions and features of the proposed IEEE standard in hardware. In addition many other functions are provided to round out the support necessary in most numeric programs. The architecture is a logical extension to the M68000 Family architecture and is clean and easy to use. Furthermore, it lends itself to being moved onto the main processor in the future. The co-processor interface was designed with a great deal of thought; not only to allow it to work well with the MC68881, but also to allow for future co-processors, multiple co-processors, and user defined co-processors. Lastly, the MC68881 is being designed with state-of-the-art hardware and all-out performance as the primary design goal.

MOTOROLA *Semiconductor Products Inc.*

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

OB 68K1

MC 68000 'MULTIBUS'* SINGLE BOARD COMPUTER

GENERAL DESCRIPTION

The OB68K1 is an extremely powerful, complete stand alone microcomputer board that is functionally interfaced to the Intel 'Multibus'* and is fully compatible with the IEEE 796 Bus Standard by way of the card edge connector. The heart of the OB68K1 is the Motorola MC68000 CPU.

FEATURES

* Powerful Motorola 68000 CPU (Addresses 16 M bytes)
* 'Multibus'* compatible (Memory and I/O)
* IEEE 796 Bus Compatible
* Up to 64K bytes EPROM (8-EPROM sockets, any combination of 2716, 2532, or 2564's in pairs)
* Up to 128K bytes of onboard RAM
* 7 Prioritized - Vectored Interrupts
* 1 Non-maskable Interrupt
* 16 MHz XTAL controlled clock (8 MHz processor speed)
* 2 - RS232C Serial Ports
* XTAL controlled baud rate generator with 16 standard rates (50-19.2K baud)
* 40 bits of onboard binary I/O utilizing 2-MC6821 PIA's
* CB2 and CA2 lines of all MC6821's are driven off the board with jumper option to select polarity.
* MC6840 triple 16 bit timer/counter with driven outputs
* 62K bytes offboard I/O space
* 2K byte onboard I/O space
* User programmable memory mapping PROM's
* Motorola MEX68KDM Software Compatible

SPECIFICATIONS

Board Dimensions:	6.75 in. (17.15 cm) X 12.00 in. (30.48 cm) 0.062 in. (0.16 cm) FR4 Printed Circuit Board (Single Layer). All IC's Socketed. Solder Mask both sides.	
Power Requirements: -32K	+5 Volts ±5% @ 3.0A +12 Volts ±5% @ 0.025A -12 Volts ±5% @ 0.005A	(ALL READINGS W/O ROM)
Temperature Range:	0 - 60° C. (non-condensing)	
Interface Compliance:	IEEE P796 MASTER D16 M24 116 VOL	

INTERRUPTS

Seven (7) levels of vectored, prioritized interrupts are supported, one of which is non-maskable. All the interrupt sources (8 from the 'Multibus' and 7 from the 0B68K1 onboard I/O) may be routed to the 7 inputs in any combination by use of jumper options.

TIMER

A MC6840 PTM (triple 16 bit timer/counter) is provided for generating system interrupts or external use (i.e. counting events, real time clock, etc.) via a 16 pin header. The outputs from each of the three timers are buffered with exclusive 'or' gates. Each output may be individually inverted.

MEMORY

The 0B68K1 uses dynamic memory with onboard hardware refresh and contains sufficient memory for most applications with 32K or 128K bytes of RAM. There are sockets for 4 pairs of EPROMs on the board. Each pair may be individually configured for 2716's, 2532's or 2564's. This gives the 0B68K1 from 16K bytes to 64K bytes of onboard EPROM capability.

The memory map is jumper selectable for Motorola MEX68KDM compatibility or RAM intensive organization. In the KDM mapping mode the 0B68K1 is software compatible with the Motorola MEX68KDM. In either case, all onboard addresses are protected from offboard use and all addresses not defined as "onboard" default to offboard memory. This feature allows multiple 0B68K1's to reside on the same bus and use identical memory maps.

I/O

The I/O capability of the 0B68K1 is impressive with 2-MC6821 peripheral interface adapters for a total of 40 bits of binary I/O (each PIA is organized as two 8 bit bytes, each byte having two control lines). The 2 PIA's are brought out to a 50 pin Ansley type header connector. Two serial ports, using the standard RS232C convention, are implemented with MC6850 ACIA's and RS232 buffers. Connection is made via 2 26 pin Ansley type header connectors. The factory standard configuration allows a terminal to be connected on port 0 and a modem on port 1 by use of mass terminated ribbon cable without the necessity to do custom wiring. If a different configuration is desired, jumper options are provided. A crystal controlled baud rate generator permits the selection of 1 of 16 standard baud rates (50-19.2K baud) and is independent of all other system timing. Each port has independent baud rate selection.

The 0B68K1 is fully compatible with the IEEE 796 I/O address space assignments. A full 64K address space is assigned for I/O in a manner that supports both the 8 bit and 16 bit I/O address field for the IEEE 796 standard.

AVAILABILITY: JULY 1981

ORDERING INFORMATION:

0B68K1-32K	32K byte RAM Version (Assembled & Tested)
0B68K1-128K	128K byte RAM Version (Assembled & Tested)

* 'Multibus' is a trademark of Intel Corporation

MOTOROLA

Advance Information
MICROSYSTEMS

MEX68KDM
MC68000 Design Module

- MC68000 System Real-Time Emulation
- EXORciser Compatible Bus
- 32K Bytes of Dynamic RAM
- 8K Bytes of System Monitor
- Extra ROM/EPROM User Sockets
- Download Host Computer Port
- Two 16-Bit Parallel I/O Ports
- Three 16-Bit Timers
- Wirewrap Area for User
- Extensive Debug Routines

Register and Memory Mode	Examine and Set
Trace and Display Mode	Trace One Instruction or To Address
Breakpoint Mode	Add/Delete/Clear All
Run Mode	Go from Program Counter or Address
Window Mode	Effective Address Display
Console Control Mode	Set Terminal Parameters and Speed
Offset Mode	Address Offset Option on Most Commands
Load/Dump/Print Mode	Program Read/Write to Terminal or Host
Symbol Mode	Assign Symbols for Command Symbol Reference
Transparent Mode	Host Computer Communications
I/O Port Mode	Port Assignment During Command Execution
Copy Mode	Move Blocks of Memory

The MEX68KDM provides the user with an easy and convenient means of designing with and evaluating the MC68000 processor. As shown in Figure 1 this module provides the necessary RAM, ROM, Timer, and I/O for initial system emulation or system prototyping. Also provided are two serial RS-232 ports for communication between the MEX68KDM module and a terminal or host computer. The terminal operator may download a machine file previously edited and assembled on a host computer, such as an IBM 370, PDP-11 or 6800 EXORciser, into the on-board 32K byte memory.

Once a memory file is resident in RAM, the user may begin his program debug phase using the extensive MACSBUG command structure provided in ROM. The MACSBUG Command Summary as provided in ROM is listed in Figure 2. Some key new debug commands beyond the usual trace and breakpoint modes should be noted.

MACSBUG now provides symbol referencing and printing, offset addressing, memory move, calls to utility routines, host computer communications, effective address windows to memory data, a memory test routine, I/O port assignment at the command level, and trace display selection.

The memory map for the MEX68KDM allows the system designer to use any of the on-card I/O as well as using additional external memory or I/O. A 6800 Bus interface card is provided to allow the MEX68KDM to read or write data to an external 6800 Memory or I/O card in either the upper or lower data byte positions.

The MEX68KDM Design Module may be used in one of three configurations. First it may be used stand-alone, where the user provides his own power supply and RS-232 terminal. Secondly the module may be used in an existing EXORciser Development system in the non-expanded bus mode, or thirdly it may be used in a card cage with other 6800 memory or I/O boards in the expanded mode.

ISSUE A

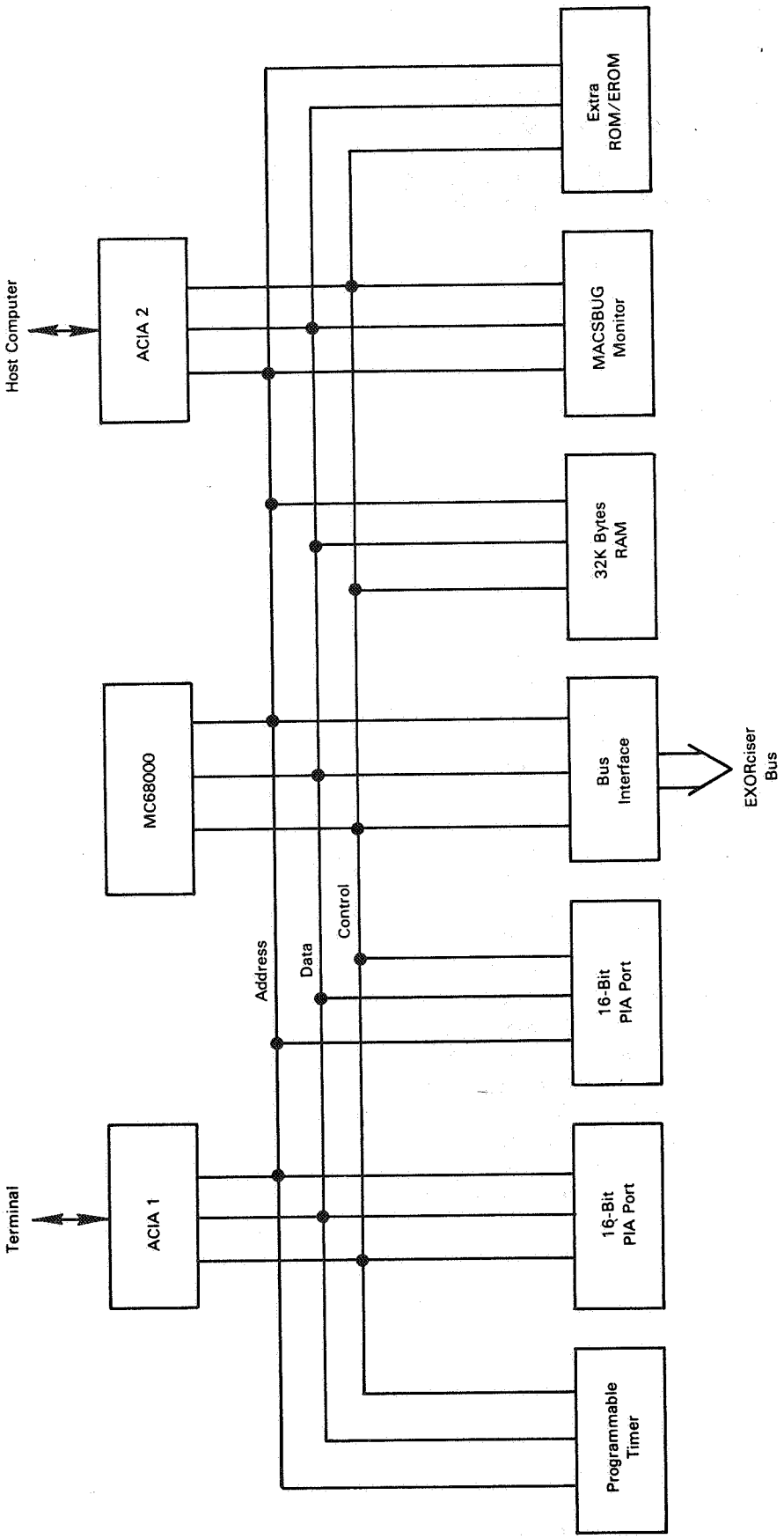


FIGURE 1 — BLOCK DIAGRAM

FIGURE 2 — MACSBUG COMMAND SUMMARY

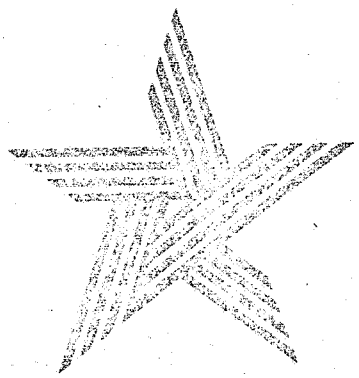
reg#	print a register#
reg# hexdata	put a hex value in the register #
reg# 'ASCII'	put hex — equivalent characters in register #
reg#/	print the old value and request new value
class	print all registers of a class (A or D)
class/	sequence through — print old value request new
R	print the trace display
R reg#. len	put a register in the display
R ALl	set all registers to appear in the display
R Clear	take all registers out of the display
T count	turn the trace mode on, trace count instructions (default 1)
T TILL Address	trace until this address
T OFF	turn trace display off — print only when stopping
LOop low high	suppress trace display when tracing through range
(CR)	carriage return — trace one instruction, trace must be on
hex number	trace this many instructions, trace mode must be on
Breakpoint	print all breakpoint addresses
B add/count	set a new breakpoint and optional count
B -address	clear a breakpoint
B CLear	clear all breakpoints
Go	start running from address in program counter
Go address	start running from this address
Go TILL add.	set temporary breakpoint
W#	print the effective address of the window
W#. len EA	define the length and addressing mode (address) of a window
M#	memory through a window (same syntax as register)
FOrmat hex	program/initialize an ACIA
NUll hex	set character null pads
CR hex	set carriage return null pads
SPeed baud	set null pads to default values
#decimal	convert decimal number to hex
#$hex	convert hex to decimal
P2	enter transparent mode, host and console linked
(CTL Q)	control 'Q' ends transparent mode
*..data..	transmit command to host
CAll address	JSR to user utility routine
COpy start end start2	copy memory from one location to another
PRint[S] start end	hex — ASCII memory dump
PUnch start end	print 'S' records (tape image)
REad	expect to receive 'S' records
VErify	check memory against 'S' records
SM address data	set memory with data
Offset address	define an offset
SYmbol NAME value	define and print symbols

MOTOROLA microsystems

MÜNCHENERSTRASSE 18, 8043 UNTERFÖHRING (WEST GERMANY)

E-153 — J.79-8.0

Printed in Switzerland



$20,000 system puts 68000 on Multibus, runs large Fortran 77 programs

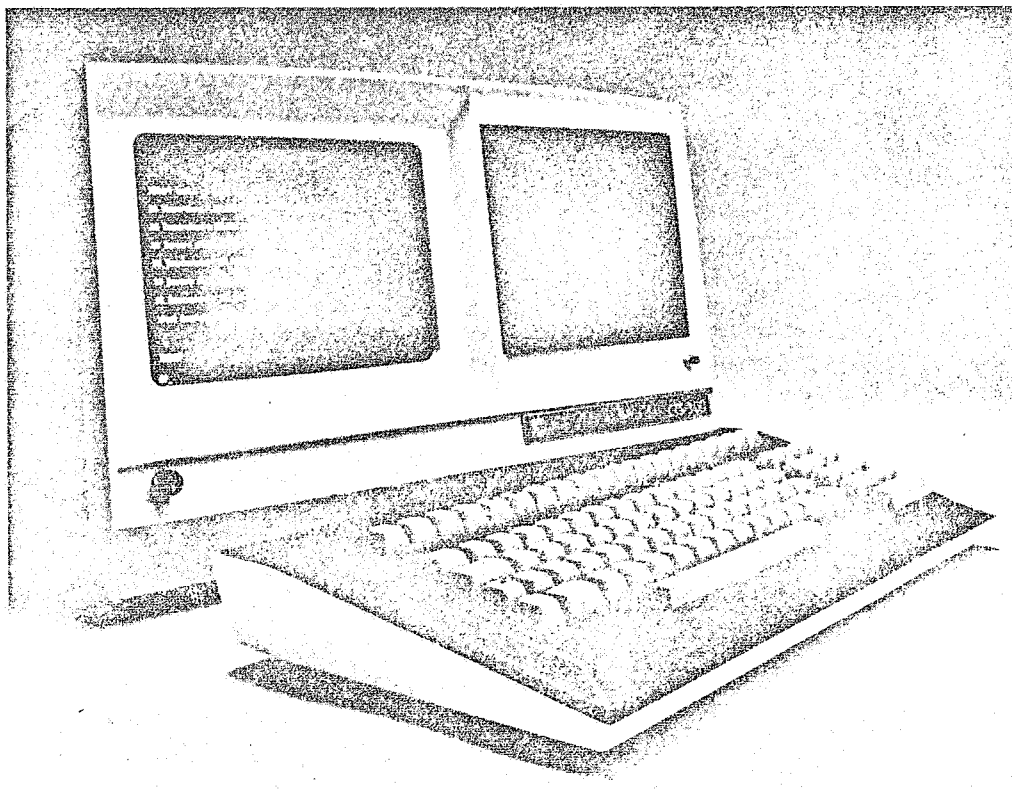
In the eyes of many, nothing would be more desirable than a combination of Motorola's 68000 microprocessor with Intel's Multibus (IEEE-P796). A number of Multibus-compatible boards are available to augment the functions of the microcomputer, whereas the 68000 is a powerful 16-bit processor. The use of 64-K random-access memories, subsidiary processors, a 5¼-in. Winchester disk drive, and several software and interface features further enhance the design of the CTS-300 integrated microcomputer system from Codata Systems Corp.

The CTS-300 crams 256-K bytes of RAM onto the same board as the central processing unit, so that the 68000 can take full advantage of the memory's 200-ns access time by using the RAM as a cache. In fact, the 8-MHz version of the 68000 employed can operate at full speed with no wait states at all when using the cache. The processor can directly address up to 2 megabytes of RAM on other Multibus cards. All RAM has parity checking, but not all has error checking and correction.

Compiler. Since so much main memory is directly addressable by the 68000, the CTS-300 can fully implement Fortran 77 with a compiler that costs $450 per copy. The compiler occupies about 100-K bytes of memory and works at speeds of about 7,000 lines per minute. When it becomes available in August, it will be possible to off-load into the CTS-300 from a mainframe any programs coded in Fortran 77, such as those translated from Fortran 4 into Fortran 77 by the compiler released this month for the IBM 370 series. Notes Codata director of marketing Beau Vrolyk, "For the first time, we are offering the user a personal computer capable of running large Fortran programs—and at a very reasonable cost."

A Pascal compiler (also $450 per copy) has been written in the native code of the 68000, and it can be run on the CTS-300. It occupies about 96-K bytes of memory and handles a superset, not subset, of the International Standards Organization's Pascal. Deliveries of the Pascal compiler will begin in June. The CTS-300 can also run a monitor program written in Pascal. "To the user, this monitor looks like Unix," notes Vrolyk. "It has directed I/O and subprocesses, but not concurrent processes."

The 68000 is aided by a series of controllers, each of which has its own 8085 microprocessor. One is a cathode-ray-tube and keyboard controller, with features including inverted video, blinking, underlining, and full cursor addressing. Another is a floppy-disk controller, including direct memory access, multiple-master mode, and a 4-K-byte data buffer for consecutive sector transfers. A separate controller for the Winchester disk drive has features similar to the floppy-disk controller's. The Winchester controller uses the faster Signetics 8X300 controller chip instead of an 8085. Also, there is a



serial input/output controller that can handle four channels, each of which is a concurrent-interrupt–driven RS-232-C port.

Tape interface. There is also an interface for a nine-track magnetic-tape storage system that can interface with either an Intephase- or Comark-type tape unit, which has a capacity over 10 megabytes using an IBM 45-in./s format.

The CTS-300 comes as an integrated microcomputer system, and the company has yet to decide whether it will make the 68000 CPU board available as a separate product. The system is packaged similarly to Codata's previous products, the CTS-100 and CTS-200, which are Multibus-oriented integrated systems using a Z80A under CP/M and a Z8000 under Xenix, respectively. The system includes a detachable keyboard with 18 programmable function keys, a 9-in. CRT displaying 80 characters by 25 lines, and two 5¼-in. disk drives. Depending upon the configuration ordered, the disk drives may be two double-sided, double-density floppy-disk units containing a total of 696-K bytes or one floppy-disk drive and one Winchester drive. The Winchester drive is the Seagate Technology ST 506 drive, which stores about 6 (unformatted) megabytes. When the Seagate ST 512 drive, with a capacity of about 12 megabytes, becomes available, it will be offered as an alternative to the ST 506.

The Multibus architecture allows multiple processors to be used in a master-slave arrangement, although the CTS-300 is initially configured as a single-user system. The Multibus card cage has nine slots, giving the user the option of adding four Multibus cards of his own for system expansion. The price of the CTS-300 system, with 68000 card and Winchester and floppy-disk drives, is about $20,000. First deliveries will take place this month.

Codata Systems Corp., 285 N. Wolfe Rd., Sunnyvale, Calif. 94086. Phone (408) 735-1744 [421]

1-Mb/s twisted-pair network connects up to 64 devices

Using RS-422 twisted-pair connections rather than coaxial cable, the Corvus Omninet is a low-cost way for 64 devices to share a local network. The types of hardware that may initially be interconnected include Apple 2 and 3, LSI-11, and Onyx computers, as well as any of Corvus's disk drives. Connection of Tandy's TRS-80– and S-100–based computers, as well as the most popular types of printers, will be added in the third quarter. The network will operate at a 1-Mb/s data rate, and each device will gain entry to the network through an interface device called a Transporter.

Appropriately enough, the Transporter operates at the fourth, or transport, level of the International Standards Organization's seven-layered networking highway. The Transporter units are designed around a Motorola 6801 processor and a custom gate array and are priced at $450 for the Apple computers, $600 for Onyx computers, and $750 for Digital Equipment Corp.'s LSI-11s. For the Corvus 5¼- and 8-in. Winchester disk drives, the Transporter unit is integrated into a disk server card, which costs $1,000 and adds security features; first in, first out communication channeling (which Corvus calls "pipes"); and disk spooling to the network. The entire Omninet can also serve as a base for the company's Constellation networking applications software, which operates at ISO levels five through seven.

The Transporters also include a unique structure for dismantling packets and demultiplexing. Using a direct-memory-access transfer technique and operating on the fly, the Transporter can separate a received data packet into two portions, such as status information and data, and transfer it directly into the main memory of the attached computer. It does this with only one small buffer, a 3-byte FIFO.

Almost virtual. Like other local network offerings such as those by Nestar, Ungermann-Bass, Zilog, and Xerox, Omninet is a carrier-sensing multiple-access system. It is different, however, in that it does not use collision-detection routines. The Corvus Omninet uses a DMA-transfer architecture based on "microvirtual circuits," a new term coined by Corvus. Notes Omninet project manager Phil Belanger, "This is more than a datagram, yet not quite a real virtual circuit." The difference is that the Omninet does not weed out extraneous packets, and it does not have all the security features of a true virtual circuit.

What it does have is a system of low-level acknowledgments, which are different types of packets from those that carry data. The network specification calls for a maximum interval of 15 μs between the time a data packet is put onto the net and the time an acknowledgment of that packet must start back to the sender. If it does not arrive within the time window, the data packet is retransmitted. The acknowledgment packet is given a higher priority than a data packet, unlike other packet schemes in which all packets are treated equally. "With our method, we can guarantee the arrival order of the data packets without requiring extensive packet-tracking overhead," adds engineering vice president Mark Hahn.

Although it does not maintain extensive collision-detection circuitry, Omninet has two methods for avoiding collisions. The first is a method that Corvus calls a "software level of randomness," which enters data packets into the net according to random numbers assigned to and based on the number (from 1 to 64) of the sending network member. The Transporters keep track of the number of collisions that have occurred recently on

ARTICLES

ColdFire: A Hot Architecture

This new implementation renews a proven embedded architecture

Joe Circello

Motorola's 68000 family of microprocessors has served both the computer and the embedded markets well. Now the PowerPC has created an opportunity for the 68000 family to refocus entirely on embedded systems, making it possible to redefine the architecture to achieve dramatic improvements in both cost and performance relative to the older 68000-family designs. A new architecture, called ColdFire, is the result of such a refocus and represents an approach targeted at the emerging needs of advanced consumer-electronics applications.

ColdFire's designers set several requirements for the architecture they would create for this new class of cost-sensitive embedded applications. Obviously, they demanded a low-cost architecture, most of which they achieved by using a small core size. A small die also let them integrate on-chip memories, system modules, and peripherals cost-effectively. ColdFire offers abundant processing power, so it can tackle computer-intensive jobs while consuming relatively little electrical power.

Finally, ColdFire employs a high-density ISA (instruction-set architecture), especially important because in many embedded systems the memory subsystem's cost far exceeds the processor's cost. A high-density ISA minimizes the application's storage requirements, which thus reduces overall system cost. The original 68000 processor ISA provided the starting point for ColdFire's ISA. Like the 68000 ISA, ColdFire defines a variable-length ISA to achieve optimum code density. This is accomplished in a RISC-based implementation that provides a very efficient silicon design.

Changes in Instruction

Other important changes were made to the 68000 ISA instruction set while still maintaining the original programming model. Certain operations had either reduced support or were eliminated, which makes for a simpler and smaller core. Examples of the changes to the instruction set are reduced support for byte- and word-size operands, reduced support for RMW (read-modify-write) instructions, and removal of instructions used primarily by desktop applications, such as the trap on overflow exception, and BCD (binary-coded decimal) arithmetic.

Let's look at these changes in more detail: For byte- and word-size operands, the instructions supporting arithmetic and logical operations were removed. However, ColdFire keeps those op codes performing simple assignments (e.g., `move`) and the Test and Clear functions. While support for RMW operations was reduced, ColdFire retains the op codes performing arithmetic and logical functions using a program-visible register and memory. A number of instructions, including those involving BCD operands, rotate op codes, and integer divides, were simply deleted. The Divide instruction was eliminated because the transistor count needed to support these op codes could not be justified. A software Divide routine has been developed that actually uses less machine cycles than does the 68000 for most operands.

A number of extensions were made to the original 68000 ISA when the 68020 microprocessor was introduced. The ColdFire architecture implements several important instructions from these additions, including a 32- by 32-bit integer multiply that produces a 32-bit result, a complete set of register sign-extension instructions, scale factors (x1, x2, x4) for indexed addressing modes, and multiple-word NOP (No Operation) instructions. Compilers use the latter to remove branch instructions.

Taking Exception

In addition to these areas of instruction-set simplification, the ColdFire exception processing model is streamlined. The architecture defines a single 8-byte frame created for all exception types on a self-aligning system stack (i.e., the stack pointer automatically compensates for misaligned data before creating an exception frame). After ColdFire creates the stack frame, it fetches an exception vector from a 1024-byte table that defines the location of the first instruction of the service routine. Thus, the processing of system calls and external interrupts remains exactly compatible with previous 68000-family designs. As a result of these simplifications, exception processing times are very fast. For most exceptions, the time from the faulting instruction until the first instruction in the service routine is a mere 12 cycles.

The resulting ColdFire ISA then represents a balance between the core size and code expansion, while retaining the 68000-family programming model with its powerful set of basic addressing modes. The static size of embedded applications in the ColdFire ISA is typically 20 percent to 40 percent less than fixed-length instruction sets. In relation to its predecessor, the ColdFire ISA produces object images that are considerably smaller than 68000 object images, but not as compact as objects targeted for the 68040.

A Tale of Two Pipelines

The hardware implementation of the ColdFire architecture uses a synthesis-driven, tools-based design philosophy. This allows the addition of optional hardware modules that provide custom functions and tune the processor's performance. It also provides design independence across different process technologies that target a range of operating frequencies and voltages. Finally, this approach also produces quick design cycles.

Two decoupled pipelines implement the ColdFire processor core: an IFP (Instruction Fetch Pipeline) and an OEP (Operand Execution Pipeline). A 12-byte FIFO (first-in/first-out) instruction buffer decouples the two pipelines (see the figure "ColdFire Processor Block Diagram"). Note that the core features a non-Harvard implementation to minimize die size and bus complexity. Studies indicate a full Harvard architecture provides only a minimal improvement in performance.

As the figure shows, the IFP itself consists of two stages, an IAG (Instruction Address Generation) stage and an IC (Instruction Fetch Cycle) stage. The OEP also consists of two stages, each of which can perform multiple functions, depending on the instruction type. The first stage of the OEP is the DSOC (Decode and Select/Operand Fetch Cycle), and the second stage is the operand AGEX (Address

Generation/Execute Cycle).

The IFP calculates the next instruction address and then fetches 32 bits of instruction data using the single-cycle processor/memory bus. Typically, the processor is connected to an on-chip memory, either in the form of a RAM/ROM structure or a unified cache. As the fetched instruction enters the processor, it is loaded into the FIFO instruction buffer. If the OEP is waiting for instruction data, the prefetched instruction is also gated directly into its instruction registers. The connection between the two pipelines is a 48-bit interface, a ColdFire instruction's maximum size. The ColdFire architecture's variable-length instructions include a 16-bit op code, an optional 16-bit extension word 1, and an optional 16-bit extension word 2. The IFP connected to the FIFO instruction buffer provides a very efficient mechanism for loading the variable-length ColdFire instructions into the OEP with a minimum of idle cycles.

The OEP is based on the traditional RISC compute engine structure with a dual read-ported register file feeding an ALU. Register-to-register instructions are executed in a single pipeline cycle with the operands fetched during the OC (Operand Fetch Cycle) phase of the OEP pipeline, and the actual execution is performed in the EX (execute) phase of the OEP pipeline.

The ColdFire ISA is not a pure load/store architecture, so there are numerous compound instructions that combine a load operation with some type of arithmetic or logical operation. These "embedded-load" instructions essentially pass through the OEP twice. This type of instruction begins by selecting the components needed to form the operand address in the DS (decode and select) phase of the OEP's first stage. Next, the ALU sums the components to form the operand address during the AG (address generation) phase in the pipeline's second stage. During the third cycle, memory is read and the desired operand returned to the core. At the same time, any required register operand is fetched during the OC phase in the OEP pipeline. Finally, the instruction is actually executed in the ALU during the EX phase in the OEP pipeline. Register store operations perform both functions (DS + OC, and AG + EX) simultaneously in each stage of the OEP to execute the instruction in a single cycle.

The results of the ColdFire design can be seen in the table above. It compares today's 68000 design (the 68EC000), the latest 68040 design, and a possible ColdFire implementation. The ColdFire architecture provides 68040 levels of performance at a given frequency in a core size smaller than the original 68000 design. The RISC-based implementation approach provides higher operating frequencies while still maintaining the advantages of a variable-length ISA. For cost-driven embedded systems, this variable-length ISA can provide substantial benefits over a fixed-length approach. Additionally, this new architecture maintains compatibility with the substantial 68000-family embedded development tool sets and preserves the knowledge base of engineers and programmers.

COMPARING COLDFIRE TO OTHER 68000 DESIGNS

	68EC000	**68040V**	**COLDFIRE**
Process technology	0.8 3.3 V, DLM	0.5 3.3 V, TLM	0.5 3.3 V, TLM
Core size (sq mm)	11.8	18.4	4.4
Frequency (MHz)	16.67	25	50
On-chip cache	None	4 KB instruction, 4 KB data	4 KB unified
External bus (bits)	16	32	32
Performance			
Embedded code	1.0x	11.6x	20.2x
Dhrystone MIPS	2.1	24.6	44.3
MIPS/watt	42	36	197

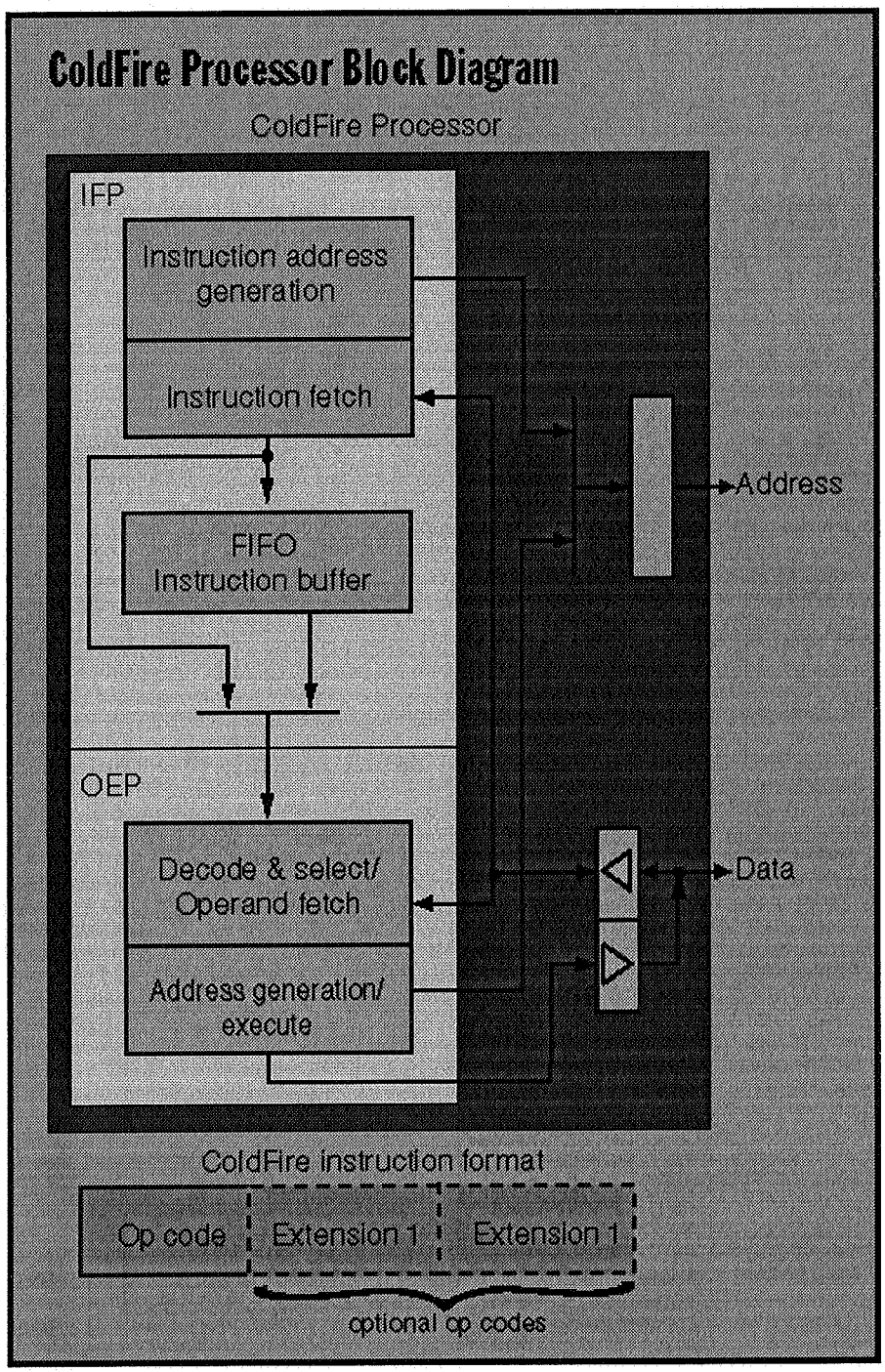
ColdFire Processor Block Diagram

illustration_link (14 Kbytes)

To improve performance, the ColdFire processor uses two pipelines. Note that the OEP's output is routed in such a way that compound instructions can pass through this pipeline twice.

Joe Circello is an advanced microprocessor architect for Motorola's High Performance Embedded Systems Division. You can reach him on the Internet at circello@oakhill.sps.mot.com or on BIX c/o "editors."

ColdFire Processor Block Diagram



To improve performance, the ColdFire processor uses two pipelines. Note that the OEP's output is routed in such a way that compound instructions can pass through this pipeline twice.