



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China






M68000

8-/16-/32-Bit Microprocessors User's Manual

Ninth Edition

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

©MOTOROLA INC., 1993

**For More Information On This Product,
Go to: www.freescale.com**

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
Section 1 Overview		
1.1	MC68000	1-1
1.2	MC68008	1-2
1.3	MC68010	1-2
1.4	MC68HC000	1-2
1.5	MC68HC001	1-3
1.6	MC68EC000	1-3
Section 2 Introduction		
2.1	Programmer's Model	2-1
2.1.1	User's Programmer's Model	2-1
2.1.2	Supervisor Programmer's Model	2-2
2.1.3	Status Register	2-3
2.2	Data Types and Addressing Modes	2-3
2.3	Data Organization In Registers	2-5
2.3.1	Data Registers	2-5
2.3.2	Address Registers	2-6
2.4	Data Organization In Memory	2-6
2.5	Instruction Set Summary	2-8
Section 3 Signal Description		
3.1	Address Bus	3-3
3.2	Data Bus	3-4
3.3	Asynchronous Bus Control	3-4
3.4	Bus Arbitration Control	3-5
3.5	Interrupt Control	3-6
3.6	System Control	3-7
3.7	M6800 Peripheral Control	3-8
3.8	Processor Function Codes	3-8
3.9	Clock	3-9
3.10	Power Supply	3-9
3.11	Signal Summary	3-10

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 4		
8-Bit Bus Operations		
4.1	Data Transfer Operations	4-1
4.1.1	Read Operations	4-1
4.1.2	Write Cycle	4-3
4.1.3	Read-Modify-Write Cycle	4-5
4.2	Other Bus Operations	4-8
Section 5		
16-Bit Bus Operations		
5.1	Data Transfer Operations	5-1
5.1.1	Read Operations	5-1
5.1.2	Write Cycle	5-4
5.1.3	Read-Modify-Write Cycle	5-7
5.1.4	CPU Space Cycle	5-9
5.2	Bus Arbitration	5-11
5.2.1	Requesting The Bus	5-14
5.2.2	Receiving The Bus Grant	5-15
5.2.3	Acknowledgment of Mastership (3-Wire Arbitration Only)	5-15
5.3	Bus Arbitration Control	5-15
5.4	Bus Error and Halt Operation	5-23
5.4.1	Bus Error Operation	5-24
5.4.2	Retrying The Bus Cycle	5-26
5.4.3	Halt Operation	5-27
5.4.4	Double Bus Fault	5-28
5.5	Reset Operation	5-29
5.6	The Relationship of \overline{DTACK} , \overline{BERR} , and \overline{HALT}	5-30
5.7	Asynchronous Operation	5-32
5.8	Synchronous Operation	5-35
Section 6		
Exception Processing		
6.1	Privilege Modes	6-1
6.1.1	Supervisor Mode	6-2
6.1.2	User Mode	6-2
6.1.3	Privilege Mode Changes	6-2
6.1.4	Reference Classification	6-3
6.2	Exception Processing	6-4
6.2.1	Exception Vectors	6-4
6.2.2	Kinds Of Exceptions	6-5
6.2.3	Multiple Exceptions	6-8

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 6		
Exception Processing		
6.2.4	Exception Stack Frames	6-9
6.2.5	Exception Processing Sequence	6-11
6.3	Processing of Specific Exceptions	6-11
6.3.1	Reset	6-11
6.3.2	Interrupts	6-12
6.3.3	Uninitialized Interrupt	6-13
6.3.4	Spurious Interrupt	6-13
6.3.5	Instruction Traps	6-13
6.3.6	Illegal and Unimplemented Instructions	6-14
6.3.7	Privilege Violations	6-15
6.3.8	Tracing	6-15
6.3.9	Bus Errors	6-16
6.3.9.1	Bus Error	6-16
6.3.9.2	Bus Error (MC68010)	6-17
6.3.10	Address Error	6-19
6.4	Return From Exception (MC68010)	6-20

Section 7 8-Bit Instruction Timing

7.1	Operand Effective Address Calculation Times	7-1
7.2	Move Instruction Execution Times	7-2
7.3	Standard Instruction Execution Times	7-3
7.4	Immediate Instruction Execution Times	7-4
7.5	Single Operand Instruction Execution Times	7-5
7.6	Shift/Rotate Instruction Execution Times	7-6
7.7	Bit Manipulation Instruction Execution Times	7-7
7.8	Conditional Instruction Execution Times	7-7
7.9	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	7-8
7.10	Multiprecision Instruction Execution Times	7-8
7.11	Miscellaneous Instruction Execution Times	7-9
7.12	Exception Processing Instruction Execution Times	7-10

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 8		
16-Bit Instruction Timing		
8.1	Operand Effective Address Calculation Times	8-1
8.2	Move Instruction Execution Times	8-2
8.3	Standard Instruction Execution Times	8-3
8.4	Immediate Instruction Execution Times	8-4
8.5	Single Operand Instruction Execution Times	8-5
8.6	Shift/Rotate Instruction Execution Times	8-6
8.7	Bit Manipulation Instruction Execution Times	8-7
8.8	Conditional Instruction Execution Times	8-7
8.9	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	8-8
8.10	Multiprecision Instruction Execution Times	8-8
8.11	Miscellaneous Instruction Execution Times	8-9
8.12	Exception Processing Instruction Execution Times	8-10
Section 9		
MC68010 Instruction Timing		
9.1	Operand Effective Address Calculation Times	9-2
9.2	Move Instruction Execution Times	9-2
9.3	Standard Instruction Execution Times	9-4
9.4	Immediate Instruction Execution Times	9-6
9.5	Single Operand Instruction Execution Times	9-6
9.6	Shift/Rotate Instruction Execution Times	9-8
9.7	Bit Manipulation Instruction Execution Times	9-9
9.8	Conditional Instruction Execution Times	9-9
9.9	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	9-10
9.10	Multiprecision Instruction Execution Times	9-11
9.11	Miscellaneous Instruction Execution Times	9-11
9.12	Exception Processing Instruction Execution Times	9-13
Section 10		
Electrical and Thermal Characteristics		
10.1	Maximum Ratings	10-1
10.2	Thermal Characteristics	10-1
10.3	Power Considerations	10-2
10.4	CMOS Considerations	10-4
10.5	AC Electrical Specifications Definitions	10-5
10.6	MC68000/68008/68010 DC Electrical Characteristics	10-7
10.7	DC Electrical Characteristics	10-8
10.8	AC Electrical Specifications—Clock Timing	10-8

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 10		
Electrical and Thermal Characteristics		
10.9	MC68008 AC Electrical Specifications—Clock Timing	10-9
10.10	AC Electrical Specifications—Read and Write Cycles	10-10
10.11	AC Electrical Specifications—MC68000 To M6800 Peripheral	10-15
10.12	AC Electrical Specifications—Bus Arbitration	10-17
10.13	MC68EC000 DC Electrical Specifications	10-23
10.14	MC68EC000 AC Electrical Specifications—Read and Write	10-24
10.15	MC68EC000 AC Electrical Specifications—Bus Arbitration	10-28
Section 11		
Ordering Information and Mechanical Data		
11.1	Pin Assignments	11-1
11.2	Package Dimensions	11-7
Appendix A		
MC68010 Loop Mode Operation		
Appendix B		
M6800 Peripheral Interface		
B.1	Data Transfer Operation	B-1
B.2	Interrupt Interface Operation	B-4

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
2-1	User Programmer's Model	2-2
2-2	Supervisor Programmer's Model Supplement	2-2
2-3	Supervisor Programmer's Model Supplement (MC68010)	2-3
2-4	Status Register	2-3
2-5	Word Organization In Memory	2-6
2-6	Data Organization In Memory	2-7
2-7	Memory Data Organization (MC68008)	2-3
3-1	Input and Output Signals (MC68000, MC68HC000, MC68010)	3-1
3-2	Input and Output Signals (MC68HC001)	3-2
3-3	Input and Output Signals (MC68EC000)	3-2
3-4	Input and Output Signals (MC68008 48-Pin Version)	3-3
3-5	Input and Output Signals (MC68008 52-Pin Version)	3-3
4-1	Byte Read-Cycle Flowchart.....	4-2
4-2	Read and Write-Cycle Timing Diagram.....	4-2
4-3	Byte Write-Cycle Flowchart	4-4
4-4	Write-Cycle Timing Diagram	4-4
4-5	Read-Modify-Write Cycle Flowchart	4-6
4-6	Read-Modify-Write Cycle Timing Diagram.....	4-7
5-1	Word Read-Cycle Flowchart	5-2
5-2	Byte Read-Cycle Flowchart.....	5-2
5-3	Read and Write-Cycle Timing Diagram.....	5-3
5-4	Word and Byte Read-Cycle Timing Diagram	5-3
5-5	Word Write-Cycle Flowchart	5-5
5-6	Byte Write-Cycle Flowchart	5-5
5-7	Word and Byte Write-Cycle Timing Diagram	5-6
5-8	Read-Modify-Write Cycle Flowchart	5-7
5-9	Read-Modify-Write Cycle Timing Diagram.....	5-8
5-10	CPU Space Address Encoding	5-9
5-11	Interrupt Acknowledge Cycle Timing Diagram	5-10
5-12	Breakpoint Acknowledge Cycle Timing Diagram	5-11
5-13	3-Wire Bus Arbitration Flowchart (NA to 48-Pin MC68008 and MC68EC000	5-12
5-14	2-Wire Bus Arbitration Cycle Flowchart	5-13

LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
5-15	3-Wire Bus Arbitration Timing Diagram (NA to 48-Pin MC68008 and MC68EC000	5-13
5-16	2-Wire Bus Arbitration Timing Diagram.....	5-14
5-17	External Asynchronous Signal Synchronization	5-16
5-18	Bus Arbitration Unit State Diagrams.....	5-17
5-19	3-Wire Bus Arbitration Timing Diagram—Processor Active	5-18
5-20	3-Wire Bus Arbitration Timing Diagram—Bus Active	5-19
5-21	3-Wire Bus Arbitration Timing Diagram—Special Case	5-20
5-22	2-Wire Bus Arbitration Timing Diagram—Processor Active	5-21
5-23	2-Wire Bus Arbitration Timing Diagram—Bus Active	5-22
5-24	2-Wire Bus Arbitration Timing Diagram—Special Case	5-23
5-25	Bus Error Timing Diagram	5-24
5-26	Delayed Bus Error Timing Diagram (MC68010).....	5-25
5-27	Retry Bus Cycle Timing Diagram	5-26
5-28	Delayed Retry Bus Cycle Timing Diagram	5-27
5-29	Halt Operation Timing Diagram.....	5-28
5-30	Reset Operation Timing Diagram.....	5-29
5-31	Fully Asynchronous Read Cycle	5-32
5-32	Fully Asynchronous Write Cycle.....	5-33
5-33	Pseudo-Asynchronous Read Cycle	5-34
5-34	Pseudo-Asynchronous Write Cycle.....	5-35
5-35	Synchronous Read Cycle.....	5-37
5-36	Synchronous Write Cycle	5-38
5-37	Input Synchronizers	5-38
6-1	Exception Vector Format.....	6-4
6-2	Peripheral Vector Number Format	6-5
6-3	Address Translated from 8-Bit Vector Number	6-5
6-4	Exception Vector Address Calculation (MC68010)	6-5
6-5	Group 1 and 2 Exception Stack Frame	6-10
6-6	MC68010 Stack Frame	6-10
6-7	Supervisor Stack Order for Bus or Address Error Exception	6-17
6-8	Exception Stack Order (Bus and Address Error)	6-18
6-9	Special Status Word Format	6-19
10-1	MC68000 Power Dissipation (P_D) vs Ambient Temperature (T_A)	10-3
10-2	Drive Levels and Test Points for AC Specifications	10-6
10-3	Clock Input Timing Diagram	10-9
10-4	Read Cycle Timing Diagram	10-13
10-5	Write Cycle Timing Diagram.....	10-14
10-6	MC68000 to M6800 Peripheral Timing Diagram (Best Case)	10-16

LIST OF ILLUSTRATIONS (Concluded)

Figure Number	Title	Page Number
10-7	Bus Arbitration Timing	10-18
10-8	Bus Arbitration Timing	10-19
10-9	Bus Arbitration Timing—Idle Bus Case	10-20
10-10	Bus Arbitration Timing—Active Bus Case	10-21
10-11	Bus Arbitration Timing—Multiple Bus Request	10-22
10-12	MC68EC000 Read Cycle Timing Diagram	10-26
10-13	MC68EC000 Write Cycle Timing Diagram	10-27
10-14	MC68EC000 Bus Arbitration Timing Diagram	10-29
11-1	64-Pin Dual In Line	11-2
11-2	68-Lead Pin Grid Array	11-3
11-3	68-Lead Quad Pack	11-4
11-4	52-Lead Quad Pack	11-5
11-5	48-Pin Dual In Line	11-6
11-6	64-Lead Quad Flat Pack	11-7
11-7	Case 740-03—L Suffix	11-8
11-8	Case 767-02—P Suffix	11-9
11-9	Case 746-01—LC Suffix	11-10
11-10	Case — Suffix	11-
11-11	Case 765A-05—RC Suffix	11-12
11-12	Case 778-02—FN Suffix	11-13
11-13	Case 779-02—FN Suffix	11-14
11-14	Case 847-01—FC Suffix	11-15
11-15	Case 840B-01—FU Suffix	11-16
A-1	DBcc Loop Mode Program Example	A-1
B-1	M6800 Data Transfer Flowchart	B-1
B-2	Example External \overline{VMA} Circuit	B-2
B-3	External \overline{VMA} Timing	B-2
B-4	M6800 Peripheral Timing—Best Case	B-3
B-5	M6800 Peripheral Timing—Worst Case	B-3
B-6	Autovector Operation Timing Diagram	B-5

LIST OF TABLES

Table Number	Title	Page Number
2-1	Data Addressing Modes	2-4
2-2	Instruction Set Summary	2-11
3-1	Data Strobe Control of Data Bus	3-5
3-2	Data Strobe Control of Data Bus (MC68008)	3-5
3-3	Function Code Output	3-9
3-4	Signal Summary	3-10
5-1	\overline{DTACK} , \overline{BERR} , and \overline{HALT} Assertion Results	5-31
6-1	Reference Classification	6-3
6-2	Exception Vector Assignment	6-7
6-3	Exception Grouping and Priority	6-9
6-4	MC68010 Format Code	6-11
7-1	Effective Address Calculation Times	7-2
7-2	Move Byte Instruction Execution Times	7-2
7-3	Move Word Instruction Execution Times	7-3
7-4	Move Long Instruction Execution Times	7-3
7-5	Standard Instruction Execution Times	7-4
7-6	Immediate Instruction Execution Times	7-5
7-7	Single Operand Instruction Execution Times	7-6
7-8	Shift/Rotate Instruction Execution Times	7-6
7-9	Bit Manipulation Instruction Execution Times	7-7
7-10	Conditional Instruction Execution Times	7-7
7-11	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	7-8
7-12	Multiprecision Instruction Execution Times	7-9
7-13	Miscellaneous Instruction Execution Times	7-10
7-14	Move Peripheral Instruction Execution Times	7-10
7-15	Exception Processing Instruction Execution Times	7-11
8-1	Effective Address Calculation Times	8-2
8-2	Move Byte Instruction Execution Times	8-2
8-3	Move Word Instruction Execution Times	8-3
8-4	Move Long Instruction Execution Times	8-3

LIST OF TABLES (Concluded)

Table Number	Title	Page Number
8-5	Standard Instruction Execution Times	8-4
8-6	Immediate Instruction Execution Times	8-5
8-7	Single Operand Instruction Execution Times	8-6
8-8	Shift/Rotate Instruction Execution Times	8-6
8-9	Bit Manipulation Instruction Execution Times	8-7
8-10	Conditional Instruction Execution Times	8-7
8-11	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	8-8
8-12	Multiprecision Instruction Execution Times	8-9
8-13	Miscellaneous Instruction Execution Times	8-10
8-14	Move Peripheral Instruction Execution Times	8-10
8-15	Exception Processing Instruction Execution Times	8-11
9-1	Effective Address Calculation Times	9-2
9-2	Move Byte and Word Instruction Execution Times	9-3
9-3	Move Byte and Word Instruction Loop Mode Execution Times	9-3
9-4	Move Long Instruction Execution Times	9-4
9-5	Move Long Instruction Loop Mode Execution Times	9-4
9-6	Standard Instruction Execution Times	9-5
9-7	Standard Instruction Loop Mode Execution Times	9-5
9-8	Immediate Instruction Execution Times	9-6
9-9	Single Operand Instruction Execution Times	9-7
9-10	Clear Instruction Execution Times	9-7
9-11	Single Operand Instruction Loop Mode Execution Times	9-8
9-12	Shift/Rotate Instruction Execution Times	9-8
9-13	Shift/Rotate Instruction Loop Mode Execution Times	9-9
9-14	Bit Manipulation Instruction Execution Times	9-9
9-15	Conditional Instruction Execution Times	9-10
9-16	JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times	9-10
9-17	Multiprecision Instruction Execution Times	9-11
9-18	Miscellaneous Instruction Execution Times	9-12
9-19	Exception Processing Instruction Execution Times	9-13
10-1	Power Dissipation and Junction Temperature vs Temperature ($\theta_{JC} = \theta_{JA}$)	10-4
10-2	Power Dissipation and Junction Temperature vs Temperature ($\theta_{JC} = \theta_{JC}$)	10-4
A-1	MC68010 Loop Mode Instructions	A-3

SECTION 1 OVERVIEW

This manual includes hardware details and programming information for the MC68000, the MC68HC000, the MC68HC001, the MC68008, the MC68010, and the MC68EC000. For ease of reading, the name M68000 MPUs will be used when referring to all processors. Refer to M68000PM/AD, *M68000 Programmer's Reference Manual*, for detailed information on the MC68000 instruction set.

The six microprocessors are very similar. They all contain the following features

- 16 32-Bit Data and Address Registers
- 16-Mbyte Direct Addressing Range
- Program Counter
- 6 Powerful Instruction Types
- Operations on Five Main Data Types
- Memory-Mapped Input/Output (I/O)
- 14 Addressing Modes

The following processors contain additional features:

- MC68010
 - Virtual Memory/Machine Support
 - High-Performance Looping Instructions
- MC68HC001/MC68EC000
 - Statically Selectable 8- or 16-Bit Data Bus
- MC68HC000/MC68EC000/MC68HC001
 - Low-Power

All the processors are basically the same with the exception of the MC68008. The MC68008 differs from the others in that the data bus size is eight bits, and the address range is smaller. The MC68010 has a few additional instructions and instructions that operate differently than the corresponding instructions of the other devices.

1.1 MC68000

The MC68000 is the first implementation of the M68000 16-/32 bit microprocessor architecture. The MC68000 has a 16-bit data bus and 24-bit address bus while the full architecture provides for 32-bit address and data buses. It is completely code-compatible with the MC68008 8-bit data bus implementation of the M68000 and is upward code compatible with the MC68010 virtual extensions and the MC68020 32-bit implementation of the architecture. Any user-mode programs using the MC68000 instruction set will run unchanged on the MC68008, MC68010, MC68020, MC68030, and MC68040. This is possible because the user programming model is identical for all processors and the instruction sets are proper subsets of the complete architecture.

1.2 MC68008

The MC68008 is a member of the M68000 family of advanced microprocessors. This device allows the design of cost-effective systems using 8-bit data buses while providing the benefits of a 32-bit microprocessor architecture. The performance of the MC68008 is greater than any 8-bit microprocessor and superior to several 16-bit microprocessors.

The MC68008 is available as a 48-pin dual-in-line package (plastic or ceramic) and 52-pin plastic leaded chip carrier. The additional four pins of the 52-pin package allow for additional signals: A20, A21, \overline{BGACK} , and $\overline{IPL2}$. The 48-pin version supports a 20-bit address that provides a 1-Mbyte address space; the 52-pin version supports a 22-bit address that extends the address space to 4 Mbytes. The 48-pin MC68008 contains a simple two-wire arbitration circuit; the 52-pin MC68008 contains a full three-wire MC68000 bus arbitration control. Both versions are designed to work with daisy-chained networks, priority encoded networks, or a combination of these techniques.

A system implementation based on an 8-bit data bus reduces system cost in comparison to 16-bit systems due to a more effective use of components and byte-wide memories and peripherals. In addition, the nonmultiplexed address and data buses eliminate the need for external demultiplexers, further simplifying the system.

The large nonsegmented linear address space of the MC68008 allows large modular programs to be developed and executed efficiently. A large linear address space allows program segment sizes to be determined by the application rather than forcing the designer to adopt an arbitrary segment size without regard to the application's individual requirements.

1.3 MC68010

The MC68010 utilizes VLSI technology and is a fully implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes. The vector base register (VBR) allows the vector table to be dynamically relocated

1.4 MC68HC000

The primary benefit of the MC68HC000 is reduced power consumption. The device dissipates an order of magnitude less power than the HMOS MC68000.

The MC68HC000 is an implementation of the M68000 16-/32 bit microprocessor architecture. The MC68HC000 has a 16-bit data bus implementation of the MC68000 and is upward code-compatible with the MC68010 virtual extensions and the MC68020 32-bit implementation of the architecture.

1.5 MC68HC001

The MC68HC001 provides a functional extension to the MC68HC000 HCMOS 16-/32-bit microprocessor with the addition of statically selectable 8- or 16-bit data bus operation. The MC68HC001 is object-code compatible with the MC68HC000, and code written for the MC68HC001 can be migrated without modification to any member of the M68000 Family.

1.6 MC68EC000

The MC68EC000 is an economical high-performance embedded controller designed to suit the needs of the cost-sensitive embedded controller market. The HCMOS MC68EC000 has an internal 32-bit architecture that is supported by a statically selectable 8- or 16-bit data bus. This architecture provides a fast and efficient processing device that can satisfy the requirements of sophisticated applications based on high-level languages.

The MC68EC000 is object-code compatible with the MC68000, and code written for the MC68EC000 can be migrated without modification to any member of the M68000 Family.

The MC68EC000 brings the performance level of the M68000 Family to cost levels previously associated with 8-bit microprocessors. The MC68EC000 benefits from the rich M68000 instruction set and its related high code density with low memory bandwidth requirements.

SECTION 2 INTRODUCTION

The section provide a brief introduction to the M68000 microprocessors (MPUs). Detailed information on the programming model, data types, addressing modes, data organization and instruction set can be found in M68000PM/AD, *M68000 Programmer's Reference Manual*. All the processors are identical from the programmer's viewpoint, except that the MC68000 can directly access 16 Mbytes (24-bit address) and the MC68008 can directly access 1 Mbyte (20-bit address on 48-pin version or 22-bit address on 52-pin version). The MC68010, which also uses a 24-bit address, has much in common with the other devices; however, it supports additional instructions and registers and provides full virtual machine/memory capability. Unless noted, all information pertains to all the M68000 MPUs.

2.1 PROGRAMMER'S MODEL

All the microprocessors executes instructions in one of two modes—user mode or supervisor mode. The user mode provides the execution environment for the majority of application programs. The supervisor mode, which allows some additional instructions and privileges, is used by the operating system and other system software.

2.1.1 User' Programmer's Model

The user programmer's model (see Figure 2-1) is common to all M68000 MPUs. The user programmer's model, contains 16, 32-bit, general-purpose registers (D0–D7, A0–A7), a 32-bit program counter, and an 8-bit condition code register. The first eight registers (D0–D7) are used as data registers for byte (8-bit), word (16-bit), and long-word (32-bit) operations. The second set of seven registers (A0–A6) and the user stack pointer (USP) can be used as software stack pointers and base address registers. In addition, the address registers can be used for word and long-word operations. All of the 16 registers can be used as index registers.

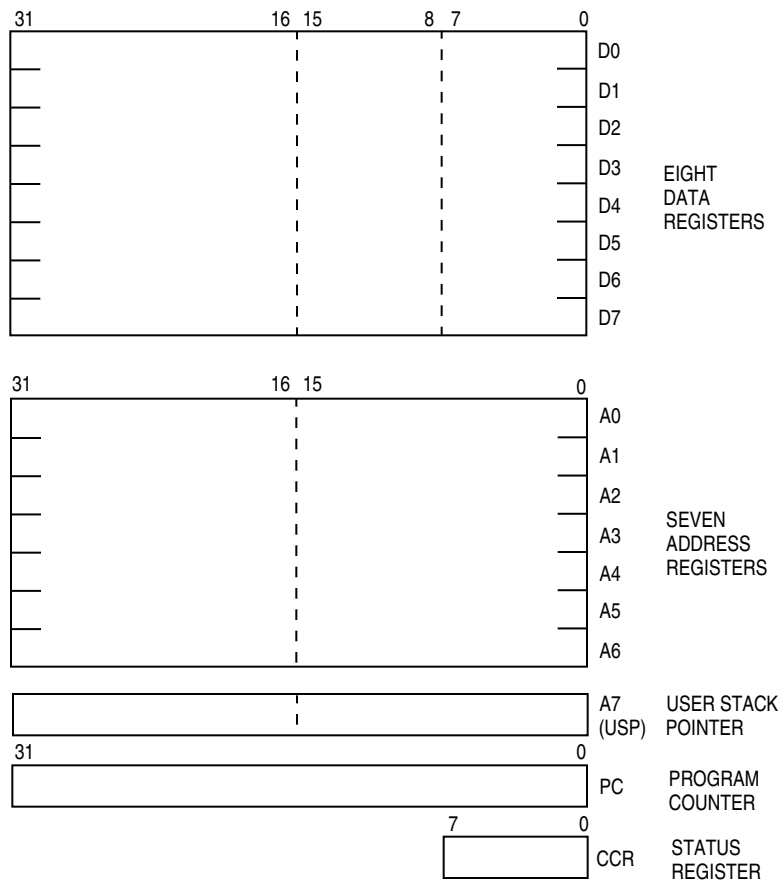


Figure 2-1. User Programmer's Model (MC68000/MC68HC000/MC68008/MC68010)

2.1.2 Supervisor Programmer's Model

The supervisor programmer's model consists of supplementary registers used in the supervisor mode. The M68000 MPUs contain identical supervisor mode register resources, which are shown in Figure 2-2, including the status register (high-order byte) and the supervisor stack pointer (SSP/A7').

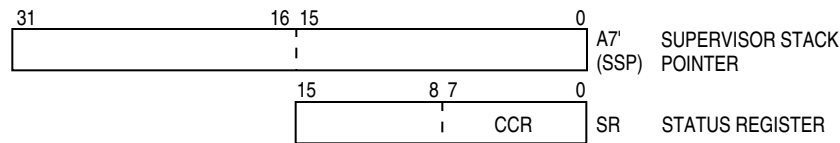


Figure 2-2. Supervisor Programmer's Model Supplement

The supervisor programmer's model supplement of the MC68010 is shown in Figure 2-3. In addition to the supervisor stack pointer and status register, it includes the vector base register (VRB) and the alternate function code registers (AFC). The VBR is used to determine the location of the exception vector table in memory to support multiple vector

tables. The SFC and DFC registers allow the supervisor to access user data space or emulate CPU space cycles.

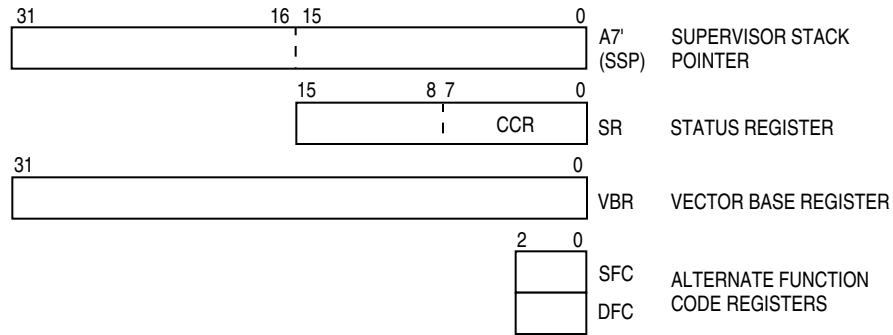


Figure 2-3. Supervisor Programmer's Model Supplement (MC68010)

2.1.3 Status Register

The status register (SR), contains the interrupt mask (eight levels available) and the following condition codes: overflow (V), zero (Z), negative (N), carry (C), and extend (X). Additional status bits indicate that the processor is in the trace (T) mode and/or in the supervisor (S) state (see Figure 2-4). Bits 5, 6, 7, 11, 12, and 14 are undefined and reserved for future expansion

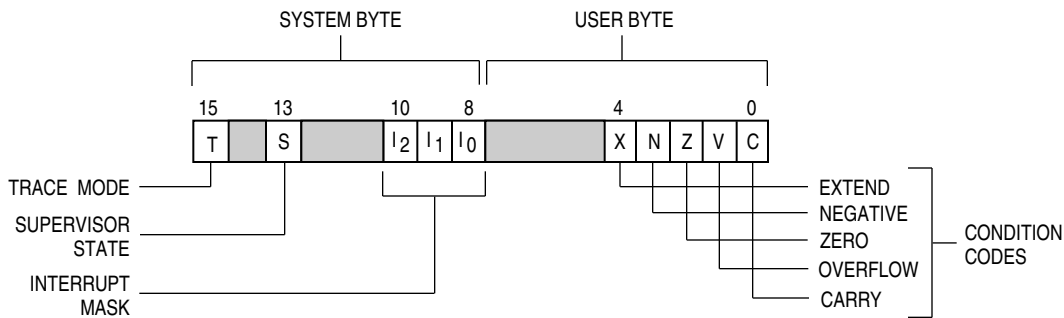


Figure 2-4. Status Register

2.2 DATA TYPES AND ADDRESSING MODES

The five basic data types supported are as follows:

1. Bits
2. Binary-Coded-Decimal (BCD) Digits (4 Bits)
3. Bytes (8 Bits)
4. Words (16 Bits)
5. Long Words (32 Bits)

In addition, operations on other data types, such as memory addresses, status word data, etc., are provided in the instruction set.

The 14 flexible addressing modes, shown in Table 2-1, include six basic types:

1. Register Direct
2. Register Indirect
3. Absolute
4. Immediate
5. Program Counter Relative
6. Implied

The register indirect addressing modes provide postincrementing, predecrementing, offsetting, and indexing capabilities. The program counter relative mode also supports indexing and offsetting. For detail information on addressing modes refer to M68000PM/AD, *M68000 Programmer Reference Manual*.

Table 2-1. Data Addressing Modes

Mode	Generation	Syntax
Register Direct Addressing Data Register Direct Address Register Direct	EA=Dn EA=An	Dn An
Absolute Data Addressing Absolute Short Absolute Long	EA = (Next Word) EA = (Next Two Words)	(xxx).W (xxx).L
Program Counter Relative Addressing Relative with Offset Relative with Index and Offset	EA = (PC)+d ₁₆ EA = (PC)+d ₈	(d ₁₆ ,PC) (d ₈ ,PC,Xn)
Register Indirect Addressing Register Indirect Postincrement Register Indirect Predecrement Register Indirect Register Indirect with Offset Indexed Register Indirect with Offset	EA = (An) EA = (An), An ← An+N An ♦ An-N, EA=(An) EA = (An)+d ₁₆ EA = (An)+(Xn)+d ₈	(An) (An)+ -(An) (d ₁₆ ,An) (d ₈ ,An,Xn)
Immediate Data Addressing Immediate Quick Immediate	DATA = Next Word(s) Inherent Data	#<data>
Implied Addressing¹ Implied Register	EA = SR, USP, SSP, PC, VBR, SFC, DFC	SR,USP,SSP,PC, VBR, SFC,DFC

NOTES: 1. The VBR, SFC, and DFC apply to the MC68010 only

- EA = Effective Address
- Dn = Data Register
- An = Address Register
- () = Contents of
- PC = Program Counter
- d₈ = 8-Bit Offset (Displacement)
- d₁₆ = 16-Bit Offset (Displacement)
- N = 1 for byte, 2 for word, and 4 for long word. If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.
- ♦ = Replaces
- Xn = Address or Data Register used as Index Register
- SR = Status Register
- USP = User Stack Pointer
- SSP = Supervisor Stack Pointer
- CP = Program Counter
- VBR = Vector Base Register

2.3 DATA ORGANIZATION IN REGISTERS

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers and the active stack pointer support address operands of 32 bits.

2.3.1 Data Registers

Each data register is 32 bits wide. Byte operands occupy the low-order 8 bits, word operands the low-order 16 bits, and long-word operands, the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or a destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

2.3.2 Address Registers

Each address register (and the stack pointer) is 32 bits wide and holds a full, 32-bit address. Address registers do not support byte-sized operands. Therefore, when an address register is used as a source operand, either the low-order word or the entire long-word operand is used, depending upon the operation size. When an address register is used as the destination operand, the entire register is affected, regardless of the operation size. If the operation size is word, operands are sign-extended to 32 bits before the operation is performed.

2.4 DATA ORGANIZATION IN MEMORY

Bytes are individually addressable. As shown in Figure 2-5, the high-order byte of a word has the same address as the word. The low-order byte has an odd address, one count higher. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long-word operand is located at address n (n even), then the second word of that operand is located at address $n+2$.

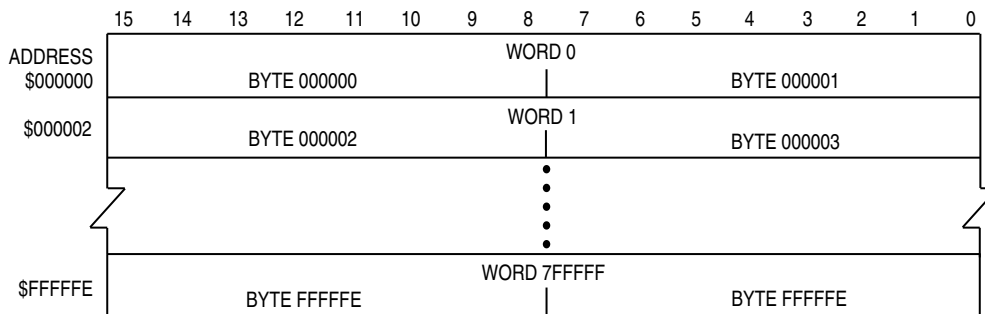


Figure 2-5. Word Organization in Memory

The data types supported by the M68000 MPUs are bit data, integer data of 8, 16, and 32 bits, 32-bit addresses, and binary-coded-decimal data. Each data type is stored in memory as shown in Figure 2-6. The numbers indicate the order of accessing the data from the processor. For the MC68008 with its 8-bit bus, the appearance of data in memory is identical to the all the M68000 MPUs. The organization of data in the memory of the MC68008 is shown in Figure 2-7.

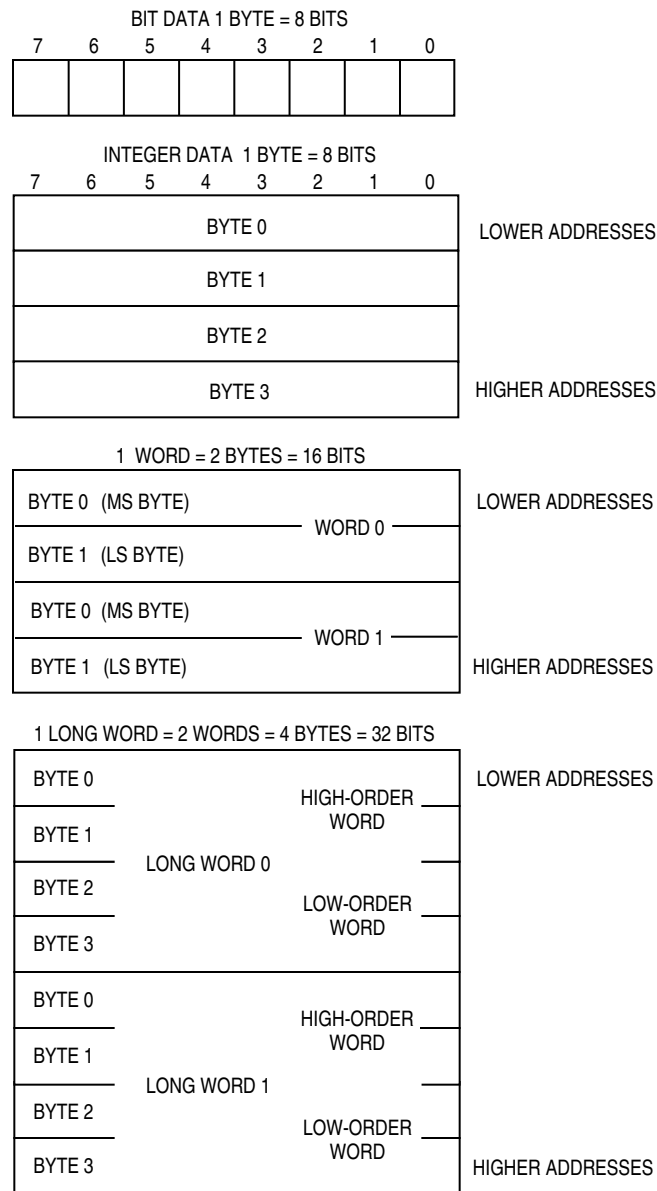


Figure 2-7. Memory Data Organization of the MC68008

2.5 INSTRUCTION SET SUMMARY

Table 2-2 provides an alphabetized listing of the M68000 instruction set listed by opcode, operation, and syntax. In the syntax descriptions, the left operand is the source operand, and the right operand is the destination operand. The following list contains the notations used in Table 2-2.

Notation for operands:

- PC — Program counter
- SR — Status register
- V — Overflow condition code
- Immediate Data — Immediate data from the instruction
- Source — Source contents
- Destination — Destination contents
- Vector — Location of exception vector
- +inf — Positive infinity
- inf — Negative infinity
- <fmt> — Operand data format: byte (B), word (W), long (L), single (S), double (D), extended (X), or packed (P).
- FPm — One of eight floating-point data registers (always specifies the source register)
- FPn — One of eight floating-point data registers (always specifies the destination register)

Notation for subfields and qualifiers:

- <bit> of <operand> — Selects a single bit of the operand
- <ea>{offset:width} — Selects a bit field
- (<operand>) — The contents of the referenced location
- <operand>10 — The operand is binary-coded decimal, operations are performed in decimal
- (<address register>) — The register indirect operator
- (<address register>) — Indicates that the operand register points to the memory
- (<address register>)+ — Location of the instruction operand—the optional mode qualifiers are -, +, (d), and (d, ix)
- #xxx or #<data> — Immediate data that follows the instruction word(s)

Notations for operations that have two operands, written <operand> <op> <operand>, where <op> is one of the following:

- — The source operand is moved to the destination operand
- ↔ — The two operands are exchanged
- + — The operands are added
- — The destination operand is subtracted from the source operand
- × — The operands are multiplied
- ÷ — The source operand is divided by the destination operand
- < — Relational test, true if source operand is less than destination operand
- > — Relational test, true if source operand is greater than destination operand
- V — Logical OR
- ⊕ — Logical exclusive OR
- ∧ — Logical AND

shifted by, rotated by — The source operand is shifted or rotated by the number of positions specified by the second operand

Notation for single-operand operations:

~<operand> — The operand is logically complemented

<operand>sign-extended — The operand is sign-extended, all bits of the upper portion are made equal to the high-order bit of the lower portion

<operand>tested — The operand is compared to zero and the condition codes are set appropriately

Notation for other operations:

TRAP — Equivalent to Format/Offset Word → (SSP); SSP-2 → SSP; PC → (SSP); SSP-4 → SSP; SR → (SSP); SSP-2 → SSP; (vector) → PC

STOP — Enter the stopped state, waiting for interrupts

If <condition> then — The condition is tested. If true, the operations after "then" are performed. If the condition is false and the optional "else" clause is present, the operations after "else" are performed. If the condition is false and else is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.

<operations> else

<operations>

Table 2-2. Instruction Set Summary (Sheet 1 of 4)

Opcode	Operation	Syntax
ABCD	Source ₁₀ + Destination ₁₀ + X → Destination	ABCD Dy, Dx ABCD -(Ay), -(Ax)
ADD	Source + Destination → Destination	ADD <ea>, Dn ADD Dn, <ea>
ADDA	Source + Destination → Destination	ADDA <ea>, An
ADDI	Immediate Data + Destination → Destination	ADDI # <data>, <ea>
ADDQ	Immediate Data + Destination → Destination	ADDQ # <data>, <ea>
ADDX	Source + Destination + X → Destination	ADDX Dy, Dx ADDX -(Ay), -(Ax)
AND	Source \wedge Destination → Destination	AND <ea>, Dn AND Dn, <ea>
ANDI	Immediate Data \wedge Destination → Destination	ANDI # <data>, <ea>
ANDI to CCR	Source \wedge CCR → CCR	ANDI # <data>, CCR
ANDI to SR	If supervisor state then Source \wedge SR → SR else TRAP	ANDI # <data>, SR
ASL, ASR	Destination Shifted by <count> → Destination	ASd Dx, Dy ASd # <data>, Dy ASd <ea>
Bcc	If (condition true) then PC + d → PC	Bcc <label>
BCHG	\sim (<number> of Destination) → Z; \sim (<number> of Destination) → <bit number> of Destination	BCHG Dn, <ea> BCHG # <data>, <ea>
BCLR	\sim (<bit number> of Destination) → Z; 0 → <bit number> of Destination	BCLR Dn, <ea> BCLR # <data>, <ea>
BKPT	Run breakpoint acknowledge cycle; TRAP as illegal instruction	BKPT # <data>
BRA	PC + d → PC	BRA <label>
BSET	\sim (<bit number> of Destination) → Z; 1 → <bit number> of Destination	BSET Dn, <ea> BSET # <data>, <ea>
BSR	SP - 4 → SP; PC → (SP); PC + d → PC	BSR <label>
BTST	\sim (<bit number> of Destination) → Z;	BTST Dn, <ea> BTST # <data>, <ea>
CHK	If Dn < 0 or Dn > Source then TRAP	CHK <ea>, Dn
CLR	0 → Destination	CLR <ea>
CMP	Destination—Source → cc	CMP <ea>, Dn
CMPA	Destination—Source	CMPA <ea>, An
CMPI	Destination—Immediate Data	CMPI # <data>, <ea>
CMPM	Destination—Source → cc	CMPM (Ay)+, (Ax)+
DBcc	If condition false then (Dn - 1 → Dn); If Dn \neq -1 then PC + d → PC)	DBcc Dn, <label>