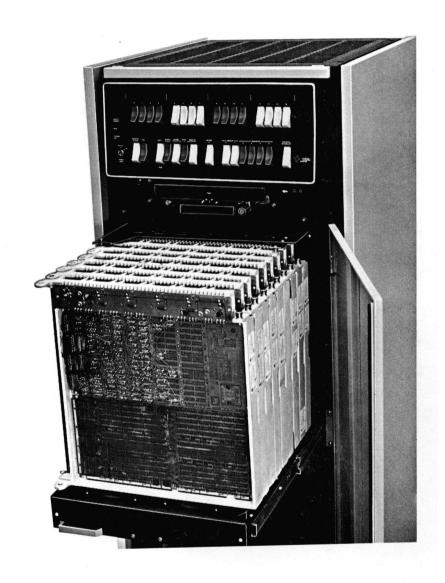
reference manual MODCOMP II computer modular computer systems our name, our claim.

# MODCOMP II COMPUTER REFERENCE MANUAL

OCTOBER 1972

210-102000-000 PRICE: \$7.00 Modular Computer Systems 1650 West McNab Road Fort Lauderdale, Florida 33309 (305) 974-1380



MODCOMP II Computer with Rack open and Planes exposed.

©Copyright 1972 by Modular Computer Systems, Inc. All rights reserved Printed in the United States of America

# **CONTENTS**

I.	MODCOMP II CHARACTERISTICS	1- ]
	GENERAL CHARACTERISTICS	1- 1
	Memory System	1- 3
	General Register File	1- 3
	Arithmetic Module	1- 3
	Read-Only Memory Controller	1- 3
	Input/Output System	1- 3
	Modular Bus Control	1- 4
	Interrupt System	1- 4
	Control Panel	1- 4
	Physical Characteristics	1- 4
	MODCOMP SOFTWARE	1- 4
	Executive Systems	1- 4
	Language Processors	1- 5
	Diagnostics, Utilities, Math Library	1- 6
	MODCOMP DATA PROCESSING PERIPHERALS	1- 6
	MEASUREMENT, CONTROL AND COMMUNICATION EQUIPMENT	1- 6
	High Level Analog Input Subsystem	1- 6
	Wide Range Solid State Analog Input Subsystem	1- 7
	Wide Range Relay Analog Input Subsystem	1- 7
	Input/Output Interface Subsystem	1- 7
	MODAC Subsystems	1- 7
	Communications Multiplexers	1- 8
	Communications Channels	1- 8
	SYSTEM EXPANDABILITY	1- 8
	MULTIPROCESSOR CONFIGURATIONS	1- 9
II.	CENTRAL PROCESSOR DESCRIPTION	2- 1
	INFORMATION FORMATS	2- 1
	Basic Formats	2- 1
	Arithmetic Data Formats	2- 2
	Character Formats	2- 3
	REGISTER FILE	2- 3
	ADDRESSING	2- 5
	Memory Word Addressing	2- 5
	Byte Addressing	2- 7
	Bit Addressing	2- 7
	DEDICATED MEMORY LOCATIONS	2- 8
	SECOND MEMORY PORT	2- 8
	READ-ONLY MEMORY CONTROLLER	2- 9
	OPERATIONAL INTEGRITY FEATURES	2- 9
	Memory Parity	2- 9
	Overflow	2- 9
	Carry Save	2 2

Unimplemented and Call Instructions	2-10
Undefined Instructions	2-10
Floating Point Overflow	2-10
Doubleword Operand Register Storage	2-10
Power Fail Safe/Auto Start	2-10
System Protect Feature	2-11
REAL-TIME CLOCK	2-11
III. INSTRUCTION SET	3- 1
OVERVIEW	3- 1
LOAD, STORE AND TRANSFER INSTRUCTIONS	3- 4
LDM	3- 4
LDI	3- 4
LDS	3- 4
LDX	3- 5
STM	3- 5
STI	3- 5
STS	3- 5
STX	3- 6
LBX	3- 6
Effective Byte Address Generation	3- 6
SBX	3- 7
LFM	3- 7
LFS	3- 7
LFX	3- 8
SFM	3- 8
SFS	3- 8
SFX	3- 8
TRR	3- 9
TRRB	3- 9
ARITHMETIC INSTRUCTIONS	3-10
ADM	3-11
ADI	3-11
ADS	3-11
ADX	3-12
ADMM	3-12
ADMB	3-12
ADSM	3-13
ADSB	3-13
ADXM	3-13
ADXB	3-14
ADR	3-14
ADRB	3-14
DAR	3-15
SUM	3-15
SUI	3-15
SUS	3-16
SUX	3-16
SUR	3-16

SURB		3-16
MPM		3-17
		3-17
MPS		
MPX		3-17 3-18
MPR		
DVM		3-18
DVS		3-18
DVX		3-19
DVR		3-19
CRMB		3-19
CRSB		3-20
CRXB		3-20
TRO		3-20
TTR		3-21
TTRB		3-21
LOGICAL INST	RUCTIONS	3-22
ETM		3-22
ETI		3-22
ETS		3-22
ETX		3-23
ETMM		3-23
ETMB		3-23
ETSM		3-24
ETSB		3-24
ETXM		3-24
ETXB		3-25
ETR		3-25
ETRB		3-25
ORM		3-26
ORI		3-26
ORS		3-26
ORX		3-26
ORMM		3-27
ORSM		3-27
ORXM		3-27
ORR		3-27
ORRB		3-28
MOX		3-28
XOI		3-28
XOS		3-29
XOX		3-29
XOR		3-29
XORB		3-29
TOR		3-30
TRMB		3-30
TRSB		3-30
TRXB		3-31

TERB	3-31
FLOATING POINT INSTRUCTIONS	3-32
Introduction	3-32
Data Formats	3-32
Floating Point Instruction Mnemonics	3-33
General Rules	3-33
Overflow	3-34
FAR	3-34
FSR	3-35
FMR	3-35
FDR	3-35
FARD	3-36
FSRD	3-36
FMRD	3-37
FDRD	3-37
FAM	3-37
FSM	3-37
FMM	
	3-38
FDM	3-38
FAMD	3-39
FSMD	3-39
FMMD	3-39
FDMD	3-40
SHIFT INSTRUCTIONS	3-41
LAD	3-41
RAD	3-41
LAS	3-42
RAS	3-42
LLD	3-42
RLD	3-42
LLS	3-43
RLS	3-43
LRS	3-43
BIT MANIPULATION INSTRUCTIONS	3-44
LBR	3-44
LBRB	3-44
ABMM	3-45
ABMB	3-45
ABSM	3-45
ABSB	3-46
ABXM	3-46
ABXB	3-46
ABR	
ABRB	3-47
SBR	3-47
	3-47
SBRB	3-48
ZBMM	3-48
ZBMB	3-48

ZBSM	3-49
ZBSB	3-49
ZBXM	3-49
ZBXB	3-49
ZBR	3-50
ZBRB	3-50
ОВММ	3-50
OBSM	3-50
OBXM	3-51
OBR	3-51
OBRB	3-51
XBR	3-51
XBRB	3-52
TBMB	3-52
TBSB	3-52
TBXB	3-53
TBRB	3-53
СВМВ	3-53
CBSB	3-54
CBXB	3-54
GMR	3-55
GMRB	3-55
BYTE MANIPULATION INSTRUCTIONS	3-56
MUR	3-56
MLR	3-56
MBR	3-56
MBL	3-57
IBR	3-57
UNCONDITIONAL BRANCH INSTRUCTIONS	3-58
BLM	3-58
BLI	3-58
BRU	3-58
НОР	3-59
BRX	3-59
CONTROL INSTRUCTIONS	3-60
HLT	3-60
NOP	3-60
SPR	3-60
SGP	3-61
SLP	3-61
SUP	3-61
INTERRUPT AND CALL INSTRUCTIONS	3-62
SIE	3-62
RIE	3-62
SIR	3-63
RIR	3-63
SIA	3-63
RIA	3-63
REX	3-63

	RMI	3-64
	CAR	3-64
	CIR	3-64
	INPUT/OUTPUT INSTRUCTIONS	3-65
	ISA	3-65
	ISB	3-65
	ISC	3-65
	ISD	3-65
	IDA	3-66
	IDB	3-66
	IDC	3-66
	IDD	3-66
	OCA	3-66
	OCB	3-66
	OCC	3-66
	OCD	3-66
	ODA	3-67
	ODB	3-67
	ODC	3-67
	ODD	3-67
IV.	PRIORITY INTERRUPTS	4- 1
	OVERVIEW	4- 1
	LEVEL ASSIGNMENTS	4- 1
	INTERRUPT OPERATION AND PROGRAM CONTROL	4- 3
	INTERRUPT SUB-LEVEL OPERATION AND PROGRAM CONTROL	4- 4
	TRAPS	4- 5
	Unimplemented Instruction Trap	4- 5
	Memory Parity Trap	4- 6
	System Protect	4- 6
	Floating Point Overflow	4- 6
	POWER FAIL SAFE/AUTO START INTERRUPT	4- 7
J.	INPUT/OUTPUT	5- 1
	OVERVIEW	5- 1
	INSTRUCTION EXECUTION SEQUENCE	5- 1
	TRANSFER FORMATS	5- 3
	REGISTER I/O TRANSFER MODES	5- 3
	INPUT/OUTPUT INTERRUPTS	5-4
	DIRECT MEMORY PROCESSOR	5- 5
	Transfer Initiation	5 <b>-</b> 5
	Data Chaining	5- 5
	Register File	5- 6
	PERIPHERAL DEVICE ASSIGNMENTS	5- 6
	PROGRAMMING CONSIDERATIONS	5- 6
	REGISTER I/O INTERRUPT MODE SEQUENCE	5- 6
	New Command Initiation	5- 6
	Response to Data Interrupt	5- 8
	Response to Service Interrupt	5- 8 5- 9
ID:		

	REGISTER I/O TEST AND TRANSFER MODE	5- 9
	DIRECT MEMORY PROCESSOR I/O MODE	5- 9
	New Command Initiation	5- 9
	Response to Data Interrupt	5- 9
	Response to Service Interrupt	5-10
	OUTPUT COMMAND FORMATS	5-10
	Select Format	5-10
	Control Format	5-11
	No Op Command	5-12
	Interrupt Disconnection and Termination	5-12
	Transfer Initiate	5-12
	INPUT STATUS FORMAT	5-13
	I/O BUS INTERFACE	5-14
	Signal Levels	5-17
VI.	OPERATOR CONTROLS	6- 1
	INDICATORS	6- 1
	Data	6- 1
	Parity Error	6- 1
	Run	6- 1
	Power On	6- 1
	SWITCHES	6- 1
	Data Entry	6- 1
	Panel Lock	6- 1
	Master Clear	6- 2
	Fill	6- 2
	Run/Halt	6- 2
	Single Cycle	6- 2
	Enter	6- 2
	Step P	6- 2
	Console Interrupt	6- 2
	Display	6- 3
	Enter R	6- 3
	Register Select	6- 3
	CONTROL PANEL OPERATION	6- 4
	Display Register	6- 4
	Load Register	6- 4
	Load Memory	6- 4
	Display Memory	6- 4
	Start Program	6- 4
	Single Cycle Program	6- 4
	FILL	6- 5

#### FIGURES

# I. MODCOMP II CHARACTERISTICS

MODCOMP II is an 800-nanosecond, 16-bit computer having many of the characteristics of 32-bit computers. It is designed to be efficient in executing higher level software including real-time multiprogramming systems, disc operating systems and interactive language systems. The large instruction set and general register file also permit highly efficient machine code to be written.

MODCOMP II consists of a set of functional modules implemented with the present state of the art in IC, MSI, LSI and core memory technology designed to be upgraded when new advances in component technology are available. All data transfers and manipulations within the computer are controlled by a highly-flexible solid state LSI memory (ROM) controller. The ROM controller provides a rich instruction set including bit, byte, word, doubleword, tripleword (including floating point) and file manipulation instructions.

The MODCOMP II is the intermediate member of the MODCOMP computer family. It has the same standard instruction set as the larger MODCOMP III computer. The same set of optional instructions is also available with both computers. Therefore, all of the software available with MODCOMP III is also available with MODCOMP II.

MODCOMP II is a superset of MODCOMP I. All MODCOMP I programs are executable in MODCOMP II.

#### GENERAL CHARACTERISTICS

The organization of MODCOMP II is shown in Figure 1-1.

The machine is packaged in two basic versions. The MODCOMP II/5 contains two CPU planes and either one or two memory planes. Each memory plane can contain a 4K, 8K, or 16K word core memory module or 512 to 2,048 words of solid state memory.

The MODCOMP II/20 and higher numbered models contain two CPU planes, one to four memory planes, and up to three option planes. The block diagram designates which options are available only in the larger computer package.

MODCOMP II consists of storage, processing and input/output modules and a modular bus through which all inter-module transfers are made. The major features of each module are described on the following pages.

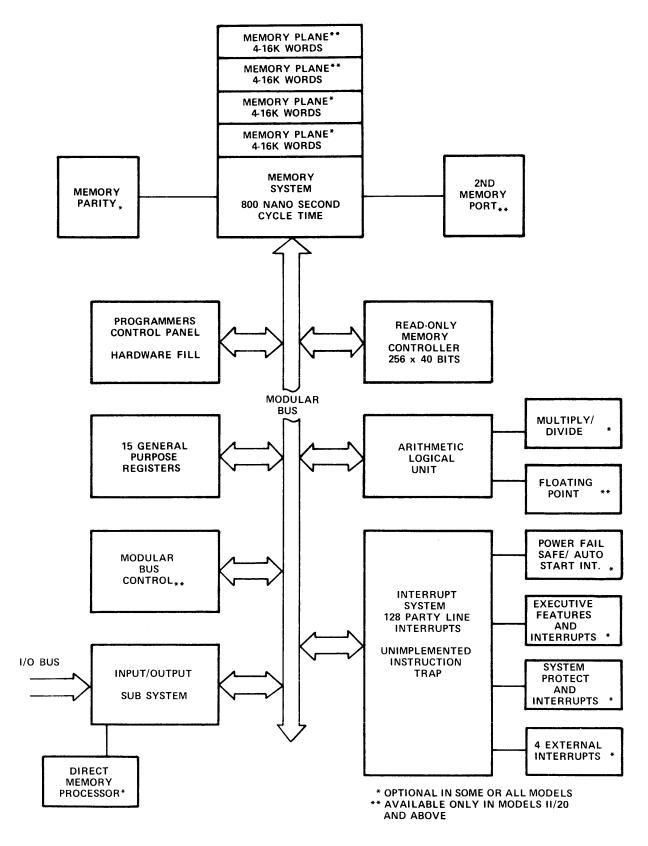


Figure I-I MODCOMP II BLOCK DIAGRAM

#### Memory System

- . 4,096 to 65,536 16-bit words, expandable by 4K, 8K, or 16K word modules
- . 400 nanosecond access time
- . 800 nanosecond full cycle time
- . All memory locations directly addressable
- . Seven memory addressing modes provided including indirect, indexed and immediate
- . Dual, concurrent access available in multiprocessor configurations
- . Memory Protect option
- . Memory Parity option

#### General Register File

- . 15 addressable, 16 bit, general purpose registers
- . 7 of the general registers usable as index registers
- . All 15 registers usable for short indexing operations
- . 800 nanoseconds execution time for typical register-to-register instructions

#### Arithmetic Module

- . Parallel operation
- . Full set of arithmetic, logical, compare, and shift capabilities
- . Execution times

```
Add, Subtract, And, Or, Exclusive Or (Reg.-to-Reg.) = 0.8 \mu sec. Add, Subtract, And, Or, Exclusive Or (Mem.-to-Reg.) = 1.6 \mu sec. Multiply (Reg.-to-Reg.) = 6.7 \mu sec., (Mem.-to-Reg.) = 7.2 \mu sec. Divide (Reg. by Reg.) = 11.0 \mu sec., (Reg. by Mem.) = 11.4 \mu sec.
```

. Implemented with four MSI modules

#### Read-Only Memory Controller

- . 267 nanosecond cycle time
- . 40-bit word length
- . 256 words in basic computer
- . Optional instructions including floating point arithmetic and fixed point multiply/divide

#### Input/Output System

- . Program controlled transfers to/from 63 peripheral devices
- . Transfers synchronized by interrupts
- . Transfers can be made from any general register to any device
- . Transfers are made over a buffered input/output bus which isolates the computer from external cable and controller delays
- . Direct Memory Processor available for automatic block transfers to/from 8 peripheral devices on a multiplexed basis
- . Controller for ASR-33, ASR-35, or KSR-35 Teletype
- . High-Speed paper tape reader, in addition to Teletype, can be operated from the controller.

#### Modular Bus Control

- . Permits Direct Memory Access transfers at rates up to 1.25 M words per second
- . Makes all machine processing logic available to an external controller
- . Enables custom macro instructions to be added to the CPU using ROM control

#### Interrupt System

- . Up to 16 unique priority levels
- . Two standard input/output levels
- . Each of these two priority levels (C,D) are connected to 17 unique sub-priority levels which can be connected to up to 64 sublevels, each with a unique (dedicated) memory pointer.
- . Standard unimplemented instruction trap
- . Complete program control of the Request, Enable\*, and Active states of each level
- . System Protect Feature includes memory protect and privileged instruction trap capabilities which enable the computer to operate in either a protected or an unprotected mode.
- . Executive features includes a real-time clock (200 Hz), console interrupt, task scheduler interrupt, and floating point overflow interrupt

#### Control Panel

- . Capability to display or modify the contents of any memory location, general register or most non-programmable registers
- . Program fill switch
- . Control panel lock switch
- . Master Clear to clear computer and peripherals
- . Optional Console Interrupt

#### Physical Characteristics

- . 0-55°C operating ambient temperature range
- . 120 + 10% VAC, 48 to 62 Hz
- . Packaged for mounting in a standard 19-inch cabinet. Occupies 8.75" (II/5) or 21" (II/20) vertically.

## MODCOMP SOFTWARE

#### Executive Systems

Three Modular Application Executive (MAX) systems are available with MODCOMP II computers to meet the requirements of a wide range of machine operating environments. A Special Application Executive (SAX) is also available for dedicated real-time applications.

MAX I is a core resident operating system which improves machine utilization efficiency in assembling, debugging and related operations.

<sup>\*</sup>Internal error exception interrupts are always enabled.

MAX II is a disc operating system which accepts a batch job input consisting of assemblies, compilations and/or executions. A core resident version is also available for non-disc systems.

MAX III is a real-time multiprogramming executive which provides complete task scheduling, initiation, termination and I/O services. This system will control the execution of any mixture of foreground/middleground and background tasks. Unprotected (middleground) tasks can be brought on-line without disturbing other protected (foreground) tasks. Batch processing can be performed in the background. A core-only version is available for dedicated applications.

<u>SAX</u> is a real-time executive which provides task scheduling, I/O services and a flexible operator communications package.

#### Language Processors

Several language processors are available with MODCOMP systems.

FORTRAN IV - The MODCOMP FORTRAN compiler meets the ANSI FORTRAN specifications. It is designed to produce efficient code by using all machine capabilities such as all registers in the register file and all instructions. It produces assembly language output, permitting the programmer to optimize further. The programmer can also write programs in any desired mixture of compiler and assembly languages. It is available in a core-resident or overlay version under MAX II and MAX III.

Extended Fortran IV - This FORTRAN compiler is an extension of FORTRAN IV as defined above containing random access I/O operations through DEFINE FILE. This compiler contains block level optimization to produce efficient object code. It is available in core-resident or overlay versions under MAX II or MAX III.

<u>BASIC</u> - This multi-user system is a subset of the Dartmouth BASIC system operating under either MAX II or MAX III. It enables users having no previous programming experience to write programs in a simple, quickly learned language.

<u>Macro Assembler</u> - This big machine class assembler has an extensive set of directives and error diagnostics as well as a macro processor. It accepts conditional assembly statements, assembly time branches and macro exits. It is a two-pass assembler, operating under MAX II and MAX III. It is available in core-resident or overlay versions.

<u>Assembler</u> - The assembler is a subset of the macro assembler. It generates relocatable as well as absolute object code and operates under MAX I, II or III.

FORTRAN Coded Assembler - The assembler is available in FORTRAN source language. This assembler operates in IBM 360/370 or CDC 6000 series computers and is compatible with the MODCOMP II ASSEMBLER in both syntax and binary output. The user can therefore assemble programs on larger machines then run them on the MODCOMP II with no modifications. It operates under OS or DOS in 65K bytes.

Compatible Assembler - This assembler accepts MODCOMP I Assembler source code and produces object code executable in a MODCOMP I, II or III computer.

#### Diagnostics, Utilities, Math Library

An advanced set of computer and peripheral diagnostics are available as maintenance aids. Utilities include source and object file editing, media-to-media conversion, and program debug capabilities. The math library meets ANSI FORTRAN standards.

# MODCOMP DATA PROCESSING PERIPHERALS

Modular peripherals are available for a broad spectrum of applications including program preparation, data processing and system support functions. All peripherals are supported by the appropriate MAX system. The basic specifications for each device are summarized below.

Page Printers - ASR-33, ASR-35, KSR-35 Teletypes

Paper Tape Reader - 625 characters per second

Paper Tape Reader and Punch - 625 characters per second read, 110 characters per

second punch

Card Readers - 300-1000 cards per minute

Card Punch - 100 cards per minute

High Speed Serial Printer - 50-150 lines per minute, 132 columns Line Printers - 600 lines per minute, 80-132 columns

- 12.5/45 IPS, 7/9 track, 556/800 BPI, industry compatible NRZ. 45 IPS, 9 track, 1600 BPI industry com-Magnetic Tape Units

patible Phase Encoded.

Moving-Head Discs - 20 millisecond average latency

Capacity range - over 1.2M, 13M and 26M words Transfer rates - 97.8K words and 156K words per

second

# MEASUREMENT, CONTROL AND COMMUNICATION EQUIPMENT

A complete range of analog input, analog output, digital input, digital output and communication equipment is available to operate with MODCOMP computer systems. equipment has all been designed together expressly to operate with MODCOMP systems. Therefore, hardware formats, interfaces, cabling and power supplies are the same in all units to facilitate customer usage and minimize spares requirements.

## High Level Analog Input Subsystem

Channel Capacity - 16-128 Channels single-ended or 8-128 Channels differential

Input Range - +10.24 volts full scale or +102.4 volts full scale

Throughput Rate - 50,000 Channels per second max.

Overall Accuracy -  $\pm 0.05$ % Full Scale  $\pm 1/2$  LSB

#### Wide Range Solid State Analog Input Subsystem

Channel Capacity - 8-128 Channels

Input Range - 12 Programmable ranges from +5 MV to +10.24V Full Scale

Throughput Rate - 20,000 Channels per second max.

Overall Accuracy - ±0.05% Full Scale ±1/2 LSB

Auto Ranging - With 4,000 Channels per second throughput

Zero Suppression - Optional

#### Wide Range Relay Analog Input Subsystem

Channel Capacity - 8-512 Channels

Input Ranges - 12 Programmable ranges from +5 MV to +10.24V Full Scale

Throughput Rate - 200 Channels per second max.

Overall Accuracy - +10 Microvolts or +0.05% Full Scale

Auto Ranging - Standard Zero Suppression - Optional

#### Input/Output Interface Subsystem

Channel Capacity - 16 Input/Output channels of 16 bits each plus expander

chassis (up to 2048, 16 bit channels)

Digital Inputs - Micrologic, positive voltage, negative voltage, bipolar

voltage, contact sense. (Isolated and filtered inputs)

Digital Outputs - Micrologic, positive voltage, negative voltage, elec-

tronic switch, contact closure, pulse output, and AC

output (TRIAC)

Analog Outputs - 12 Bits binary, including sign

-  $\pm$ 10 volts,  $\pm$ 20 volts, 1 to 5 ma, 4 to 20 ma, 10 to 50 ma

Serial Communica-

tions Interface - RS 232 or 20 ma current loop (TTY compatible)

Interval Timer - Provides programmable timing interrupt or 'watchdog'

timer

I/O Interrupts - Provides 8 data interrupts and/or 8 service interrupts

External Interrupts - Provides signal conditioning and drivers for 16 external

interrupts

Synchronizer - Provides 'handshake' data transfer

#### MODAC Subsystem

Provides a flexible combination of analog and digital data acquisition modules.

Analog Input Module - 32 High level (+10.24V) channels, 20K SPS

Analog Output Module - Eight D-to-A converters, 12 bits binary, current and

voltage outputs

Dual Word Input Module - Two 16-bit digital input channels
Dual Word Output Module - Two 16-bit digital output channels

# Communications Multiplexers

Types - Universal, operates in synchronous and/or asynchronous mode.

Asynchronous, operates in asynchronous mode only.

Channel Capacity - Universal, 4 to 32 full duplex channels expandable in groups

of 4 up to 64 full duplex channels.

Asynchronous, 2 to 32 full duplex channels expandable in

groups of 2 up to 128 full duplex channels.

#### Communications Channels

#### ASYNCHRONOUS

Clocking Mode - Asynchronous

Communication

Interfaces - EIA RS-232-C Modems, TTL Modems, TTY 60/20 ma Current loop

Baud Rate - Patchable from 75 to 9600 baud with a maximum of five different

baud rates per multiplexer

Codes - Program selectable - 5, 6, 7, or 8 bits plus parity

Stop bits - Program selectable - 1 or 2

Parity - Program selectable - none, odd, even

Echo - Program selectable - Echos on full duplex line

#### SYNCHRONOUS

Clocking Mode - Synchronous

Communications

Interfaces - EIA RS-232-C Modems, TTL Modems

Baud Rate - Patchable to 50K baud with a maximum of five different baud

rates per multiplexer

Code - Program selectable - 5, 6, 7, or 8 bits plus parity

Parity - Program selectable - none, odd, even

Sync Character - Patchable

#### SYSTEM EXPANDABILITY

The modular design makes the MODCOMP II computer easily expandable. The 21 inch high assembly is capable of containing all system features. Core memory up to a total of 64K words is available on a total of four memory planes. The interrupts are also modular and are field expandable. Even the concurrent memory access path (second port) can be added in the field. Therefore, a MODCOMP system can be expanded as system requirements grow. It can even be converted into a multiprocessor system if the need for a substantial increase in computing capability arises.

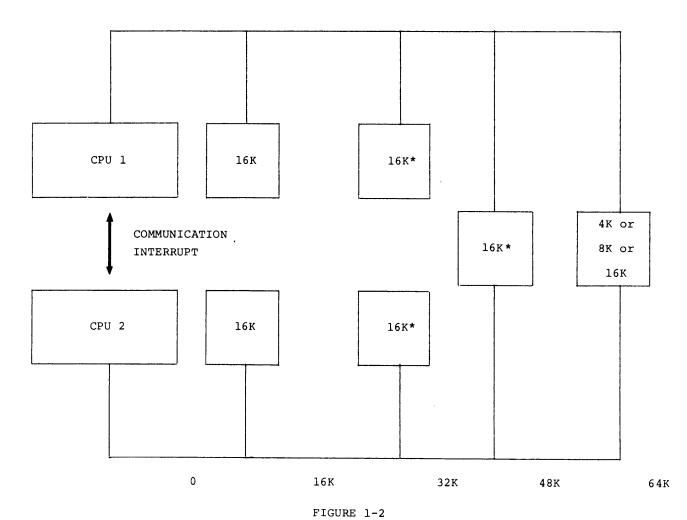
# MULTIPROCESSOR CONFIGURATIONS

The MODCOMP II is available as a multiprocessor having two CPU's and both private and shared memory modules. The range of multiprocessor configurations available is shown in Figure 1-2.

Each of the two computer cabinets can contain from 16K to 64K words of memory, and each CPU can address up to 64K words. Memory can be connected to the CPU in the other cabinet on a 16K word basis, except for the highest memory section which can be 4K, 8K or 16K words.

The lower 16K memory section cannot be shared because the lower memory locations in each computer are dedicated interrupt and I/O locations. Either one shared or two private modules can be connected between the 16K and 32K address boundaries and also between the 32K and 48K boundaries.

The CPU-to-CPU communication interrupt is generated by execution of the Request Multiprocessor Interrupt instruction. Whenever this instruction is executed in one CPU, an interrupt signal is sent to Level 3 in the other CPU.



\* Can be private or shared

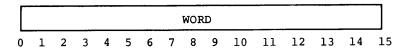
# II. CENTRAL PROCESSOR DESCRIPTION

# INFORMATION FORMATS

#### Basic Formats

The 16-bit word is the basic information format of the MODCOMP II computer. The bit designations in the computer word are:

#### WORD FORMAT



Some instructions operate on doublewords which consist of 32 bits of data stored in two consecutive register or memory locations.

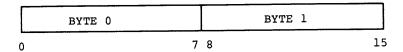
#### DOUBLEWORD FORMAT

EVEN REGISTER	MORE SIGNIFICANT WORD	
	0	15
ODD REGISTER	LESS SIGNIFICANT WORD	
	0	15

To be an operand for a doubleword instruction which operates on register contents, the more significant word must be stored in a register location having an even address and the less significant word must be stored in the next higher (odd) register.

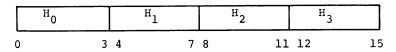
Many instruction and peripheral devices operate on eight-bit bytes which are packed two per register or memory location in the format:

BYTE DESIGNATIONS



Hexadecimal (base 16) digits are often used as a convenient means of representing binary byte, word or double word values. The hexadecimal word format is:

#### HEXADECIMAL DIGIT DESIGNATIONS



Where  $\mathbf{H}_{\mathbf{0}}$  is the most significant digit in the word.

Where  ${\bf H}_0$  is the most significant digit in the word. Hexadecimal numbers and the equivalent decimal numbers are listed in Appendix A. In the text hexadecimal numbers appear in the form  ${\bf N}_{16}$ .

#### Arithmetic Data Formats

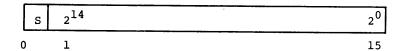
Fixed Point Binary Integer Format - This is the standard arithmetic data format in the MODCOMP II and consists of a sign bit and 15 or 31 data bits. The most significant bit is the sign bit, which is defined:

Sign = 0, Positive or Zero Quantity

Sign = 1, Negative Quantity

Two's complement representation is used for negative numbers. The principal fixed point arithmetic formats are:

SINGLE PRECISION FIXED POINT DATA FORMAT



## DOUBLE PRECISION FIXED POINT DATA FORMAT

s 2 <sup>30</sup>	2 <sup>16</sup>
0 1	15
2 <sup>15</sup>	20
0	15

Floating Point Format - This consists of a nine bit binary exponent and a 22 or 38 bit signed binary fraction. The exponent values are defined:

Exponent	Floating-Point			
Value 16	Number	<u>Value</u>		
000	2- <sup>256</sup> x	Fraction Value (1> $F \ge 0$ )		
400	2° x	Fraction Value		
7FC	2 <sup>255</sup> x	Fraction Value		

The value of zero is represented by  $00000...0_{16}$ . Hardware operations resulting in a zero fraction set the exponent to all zeroes. A negative number is represented as the integer two's complement of the absolute value so that integer compare and negate operations are valid with both fixed point and floating point operands.

The floating point formats are:

SINGLE PRECISION FLOATING POINT DATA FORMAT

s	EXPONENT		FRACTION	
0	1	9	10	15
	LEAST SIGNIFIC	ANT	BITS OF FRACTION	
0				15
	DOUBLE PRECISION	FLO	ATING POINT DATA FORM	AT
s	EXPONENT		FRACTION	
)	1	9	10	15
	]	FRAC	TION	
)				15
	LEAST SIGNIFIC	CANT	BITS OF FRACTION	
0				15

#### Character Formats

The ASCII code is the standard character code in MODCOMP computers and peripherals. Appendix B contains the character code definitions.

#### **REGISTER FILE**

MODCOMP II contains 16 addressable registers. Fifteen are fast access flip-flop registers having general register capabilities. Operands can be transferred between any of these registers and any other register or any memory location. In addition, the execution of many instructions produces a result stored in one or more of the general registers. All 15 of the general registers may be used in short indexed operations and 7 may be used as index registers.

One of the 16 addressable registers is the 16-bit switch register located on the control panel. This register is provided as one means of communication between the operator and program.

The designations and dedicated functions of the sixteen addressable registers are:

REGISTER FILE DESIGNATIONS AND DEDICATED FUNCTIONS

LOWER GENERAL REGISTER FILE AND INDEX REGISTERS (R1-R7)

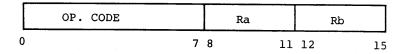
R0	Switch Reg.
Rl	Base Reg.
R2	
R3	
R4	
R5	
R6	
R7	

UPPER GENERAL REGISTER FILE

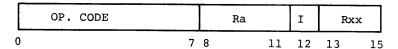
Register Rl has a dedicated hardware function in addition to being a general register. In one of the modes of memory address generation, a displacement value contained in the instruction is added to the contents of register Rl to produce the effective memory address. Registers Rl-R7 may be used as index registers. All registers R0-R15 may be used in short indexed operations.

The registers are designated by four-bit fields in the formats of instructions which invoke register operation. Typical register designations are shown in the following examples:

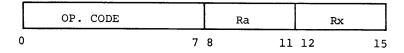
#### REGISTER-TO-REGISTER INSTRUCTION FORMAT



#### INDEXED INSTRUCTION FORMAT



#### SHORT INDEXED INSTRUCTION FORMAT



- Ra Specifies one operand register ( $0 \le a \le 15$ ) and the destination register. The destination register should not be R0, the switch register, unless the operation result is to be discarded, which is sometimes convenient in conditional branch instructions.
- Rb Specifies the second operand register (0  $\leq$  b  $\leq$  15).
- Rxx Specifies the index register  $(1 \le xx \le 7)$ .
- Rx Specifies the effective address register for short indexed instructions (0  $\langle$  x  $\langle$  15).

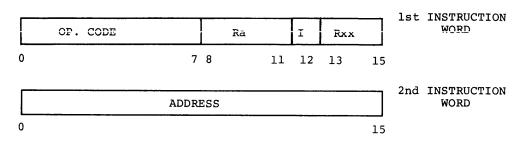
#### **ADDRESSING**

#### Memory Word Addressing

A total of seven memory addressing modes are provided in MODCOMP II instructions which operate on word operands. In each of these modes, a 16-bit effective word address (EWA) is produced in the central processing unit (CPU) and sent to the memory system along with a read or write request. The 16-bit contents of the location specified by the EWA are then either read from memory or replaced by the word transferred from the CPU. The 16-bit EWA provides a direct addressing range of 65,536 words.

The first four of the seven memory addressing modes are derived from this instruction format:

#### BASIC MEMORY ADDRESS FORMAT



Ra - Register Address

Rxx - Index Register Address (1  $\leq$  xx  $\leq$  7) where 0 = no indexing

I - Indirect Address Bit

 $\underline{\text{Direct}}$   $\underline{\text{Address}}$   $\underline{\text{Mode}}$  - If Rxx = 0 and I = 0, the 16-bit address contained in the second instruction word becomes the EWA.

Indexed Address Mode - If  $Rxx \neq 0$  and I = 0, the contents of register Rxx are added to the 16-bit address contained in the second instruction word. The least significant 16 bits of the result become the EWA. The contents of the index register may be either positive or negative to produce either positive or negative displacement indexing.

#### EXAMPLE:

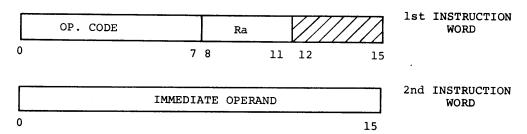
The indexing operation does not increase instruction execution time.

Indirect Address Mode - If Rxx= 0 and I = 1, the 16-bit address contained in the second instruction word specifies the memory location which contains the EWA. The indirect address capability is single level. One cycle time is added to instruction execution time by the indirect address word fetch.

Indexed and Indirect Address Mode - If  $Rxx \neq 0$  and I = 1, the contents of register Rxx are added to the 16-bit address contained in the second instruction word. The resulting address then specifies the location of the EWA. One cycle time is added to instruction execution time.

Immediate Mode - This two word memory reference instruction accesses operands or stores operands in the second instruction word. The program register is advanced by two to skip this location. The instruction format is:

#### IMMEDIATE OPERAND FORMAT



Short Displaced Mode - This single-word memory reference instruction format is provided for processing lists of operands occupying 16 or less consecutive memory locations. The instruction format is:

#### SHORT DISPLACEMENT FORMAT

	OP. CODE			Ra			DF	
0		7	8		11	12		15

DF = Displacement Field (0  $\leq$  DF  $\leq$  15)

In this mode of addressing memory, the positive displacement quantity DF and the contents of register Rl are added, to generate the 16-bit EWA. The contents of Rl are not modified by the EWA computation:

$$(R1) + DF = EWA$$

The 16-bit contents of Rl specify the base location (lowest address) of the list stored in memory.

When Branch instructions are executed in the short displaced mode, the Program Register rather than register Rl is used as the base register.

Short Indexed Format - This single-word memory reference instruction enables the contents of any of the 16 addressable registers to become the EWA. The instruction format is:

#### SHORT INDEXED FORMAT

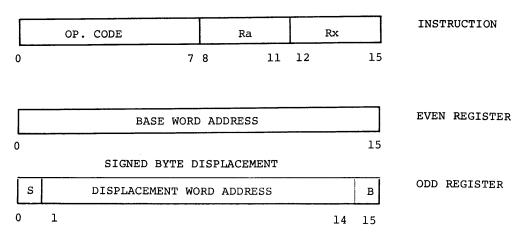
	OP. CODE		Ra			Rx		
0		7	8		11	12		15

<sup>,</sup> where Rx specifies the register which contains the EWA.

#### Byte Addressing

A byte may be addressed in any memory word with a special form of the short indexed format. In this case Rx specifies an even/odd pair of general registers.

#### BYTE ADDRESS FORMAT



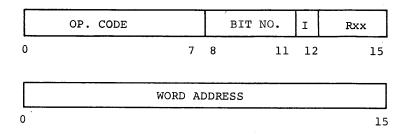
- B = 0 Specifies the byte contained in bits 0-7, and
- B = 1 Specifies the byte contained in bits 8-15 of the memory location specified by the EWA

The effective byte address EBA is obtained by adding the 16-bit base address to the signed byte displacement which is first shifted right one bit position. This produces an EWA which enables the accessing of the location containing the specified byte. The proper byte is then accessed from this location depending upon the state of B.

#### Bit Addressing

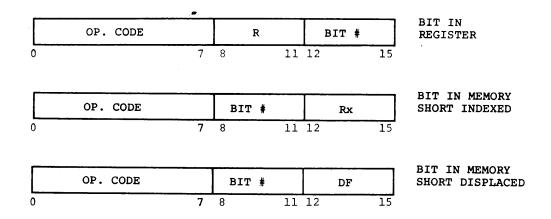
Any bit in memory can be addressed by the instruction format:

BIT ADDRESSING FORMAT



BIT NO. = 0 to 15, where 0 specifies the bit at the most significant end of the word.

The register-to-register, short displaced and short indexed forms are also used with these instructions.



# **DEDICATED MEMORY LOCATIONS**

Table 2-1 shows the area of memory which is dedicated to interrupt linkages and input/output transfer parameters.

Memory Locations 16	Dedicated Function
00-1F	Bootstrap Loader (00-2D),
	(Overlaps Interrupt Locations 20-2D)
20-5F	Interrupt Entry and Return
60-6F	DMP Transfer Count
70-7F	DMP Transfer Address
80-BF	I/O Data Interrupt Entry
C0-FF	I/O Service Interrupt Entry

Table 2-1 Dedicated Memory Locations

# SECOND MEMORY PORT

This optional feature provides a second memory port, or access path. Each port can have access to all 64K words (max.) contained in one MODCOMP II/20 or II/25. The CPU packaged with the memory is connected to the first (higher priority) port. The second port, connected to an external CPU can be connected to any combination of 16K word memory sections. (0-16K, 16-32K, 32-48K, 48-64K).

Each port can obtain a memory access in a different memory section simultaneously. If a simultaneous access is attempted in the same 16K section by both ports, the higher priority port will obtain the next cycle and the lower priority port the following cycle.

#### READ-ONLY MEMORY CONTROLLER

All standard MODCOMP II instructions are executed by a sequence of micro instructions stored in a 256 x 40 bit solid-state LSI read-only memory (ROM) module. In addition the operations performed by the Direct Memory Processor are controlled by ROM stored micro instructions. Micro instructions are executed at the rate of 3.75 million instructions per second, which corresponds to three ROM cycles per main memory cycle.

The ROM can be expanded on a CPU option plane to enable macro instructions to be added to the standard instruction set.

#### **OPERATIONAL INTEGRITY FEATURES**

Continuous checking is performed for the principal conditions, for which valid checks can be made, that can cause machine stoppage or abnormal program operation. The error signals are connected to interrupt levels, either as standard or optional features, to facilitate operation of the computer in real-time environments.

#### Memory Parity

A parity bit is stored in all memory word locations, when the memory parity checking feature is present. Each time a memory access is made, either for a byte or a word, the parity of both bytes in the word is generated or checked. If an error is detected, the execution of the instruction is aborted, further execution is halted, and the machine attempts to trap to the optional parity priority interrupt level. (Level 1) If the optional System Protect Feature is included in the computer, the parity error signal is connected to the interrupt level (Level 1). Since the instruction execution is aborted when the error is detected, the signal which interrupts the computer is classified as a trap, rather than an interrupt signal. (See Traps - Pg. 4-5) The parity error light is reset by the interrupt.

The parity error indicator is set whenever a parity error is detected and will remain on until a priority interrupt occurs or the machine is normalized.

#### Overflow

An overflow signal is generated in arithmetic operations if the result exceeds the capacity designated for the result. The specific overflow conditions are defined with the individual instruction descriptions. The general instruction types which can cause overflow are: Add, Subtract, Divide, Two's Complement, and Left Arithmetic Shift.

If an overflow occurs during the execution of one of these instructions, the overflow latch will be set regardless of its previous condition. A special machine instruction (TRO,R) is used to read the latch and reset it. Another instruction (GMR,R,O) may be used to set the overflow latch unconditionally. (Displayed in register #3D bit 0).

#### Carry Save

A carry save signal is generated in arithmetic operations if the result produces a carry. The general instruction types which can produce a carry are: Add, Subtract, Multiply, Divide, and Two's Complement.

If a carry occurs as a result of execution of one of these instructions, the carry save latch (Register #3D bit 15) will be set regardless of its previous state. Conversely, if no carry is generated, the latch is reset regardless of its previous state. A special instruction (TRO,R) is used to read the latch. Any GMR,R instruction will unconditionally reset the carry save latch.

#### Unimplemented and Call Instructions

Optional instructions such as floating point and custom macro op code groups are trapped in MODCOMP II computers not containing these options. The trap routine can execute all of these instructions as subroutines. Therefore, programs which contain these optional instructions can be executed in all MODCOMP II computers.

The trap level, which is present in all machines, is Level 4.

A special instruction Request Executive Service (REX) always generates the Unimplemented Instruction trap. This instruction is used for communication with the resident executive.

# <u>Undefined</u> <u>Instructions</u>

All undefined instruction op codes and unassigned op codes will execute as No Operation (NO OP) instructions in  $800\ \mathrm{nsec}$ .

# Floating Point Overflow

Floating point overflow is a separate trap from Overflow (above). Floating point overflow/underflow will occur if the resultant exponent of a floating point operation cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

The floating point unit trap mechanism used to indicate an overflow or underflow condition is the same function as the CPU trap implementation. The trap mechanism terminates the normal FPU flow of events and does not allow any results to be transferred back to the CPU register file. Therefore, the original register operands are maintained in the CPU register file and may be interrogated for further overflow-underflow clarification.

The trap level, when present in the system, is Level 5.

# Doubleword Operand Register Storage

Doubleword operands must be stored in register pairs in which the more significant word is stored in an even numbered register and the less significant word is stored in the next higher (odd register). The even register number must be used in the instruction to designate the doubleword. The use of an odd register number to designate doublewords will produce unspecified results except for multiplication operations. Refer to the descriptions of multiply instructions for more information.

# Power Fail Safe/Auto Start

When the a-c power is turned on or off in MODCOMP II computers having the PFS/AS feature, an interrupt is generated which overrides all other machine conditions, except the Halt condition.

This level is always enabled. When power fails, a minimum of 200 execution cycles are available after the interrupt occurs. After this time interval, memory writing is disabled to insure that the magnetic states of all cores remain unchanged when the power is turned off. When power is applied to the system, memory writing is also inhibited until proper initial conditions have been established for operation. At this time an interrupt is generated which can be used for automatic program initialization if the Halt/Run switch is in the RUN position or the CP is locked.

The PFS/AS interrupt level is Level 0.

#### System Protect Feature

The MODCOMP II offers a hardware system protect option which may utilize either a triple boundary method or a single (MCIII compatible) boundary method to assign protected memory. The rules governing the allocation of memory are as follows:

Triple Protect Boundary Registers

Two nine bit and one eight bit hardware registers are provided with the option to allow the selection of the protection boundaries for the memory system. These registers allow the boundaries to be assigned at any 128 word increment from 128 to 64K words of memory for the lower protect boundary, from 64K to 128 words of memory for the upper protect boundary, and at any 256 word increment from 256 to 64K words of memory for the third protect boundary.

1. For the Lower Boundary:
 (LPR) x (128) + 128 = Lower Boundary

-Where LPR = Contents of the 9 bit Lower Protect Boundary Register. All memory below the lower boundary is protected (or privileged).

2. For the Upper Boundary:
 (UPR) x (128) + 128 = Upper Boundary

-Where UPR = Contents of the 9 bit Upper Protect Boundary Register
All memory at or above the upper boundary but below the third boundary is protected
(or privileged).

3. For the Third Boundary:
 (B3R) x (256) + 256 = Third Boundary

-Where B3R = Protect Boundary Register #3.

Therefore, all memory at and above the lower boundary but below the upper boundary or all memory at or above the third boundary is unprotected (or unprivileged).

Single Protect Boundary Register

One five bit hardware register is provided to allow selection of the protection boundary for the memory system. This register allows the boundary to be assigned at any 2K word increment from 2K to 64K words of memory.

(PBR) x (2K) + 2K = Protect Boundary -Where PBR = Protect Boundary Register

All memory below the protect boundary is protected (or privileged) and all memory at and above the boundary is unprotected (or unprivileged).

This protect scheme is program compatible with the system used in the MCIII.

General Program Protect Characteristics

These options provide the capability to prevent a background program(s) (located outside of protected memory) from erroneously altering, executing or permanently inhibiting the foreground (or protected) program(s). In addition, the protect logic is connected to an optional priority level to notify the computer of any attempt on its part to violate the protect structure even though the attempt was aborted (except for branch violations) by the hardware. When the protect option is invoked, the protect logic traps the execution of I/O, Interrupt and Halt instructions if the program is operating in unprotected memory.

The Master Clear Console switch and Power Normalization signals always fill all three protect registers thereby yielding all of existent memory protected.

The System Protect Feature consists of two types of protection:

Memory Write Protection is included to prevent programs from modifying other resident programs. In MODCOMP II, two boundaries can be established by program control at any 128 word boundaries in memory. Programs stored between these boundaries cannot modify or branch into locations outside of these two boundaries. If an illegal attempt is made, a trap is generated at interrupt Level 2.

The Request Executive Service instruction is used for communication between programs in unprotected memory and the resident executive, which is located in protected memory.

<u>Privileged Instruction Execution</u> capability is provided to prevent unprotected programs from executing any input/output, protect status, interrupt instructions or the Halt instruction. A trap is generated at interrupt Level 2 if the execution of any privileged instruction is attempted.

The standard memory parity error is connected to interrupt Level 1, as part of the System Protect Feature. This grouping of integrity features is the result of monitor requirements and the physical grouping of the interrupts.

### REAL-TIME CLOCK

The real-time clock, which is part of the Executive Features, produces an interrupt signal at five millisecond intervals. The real-time clock interrupt is Level 6.

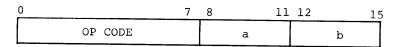
# III. INSTRUCTION SET

## **OVERVIEW**

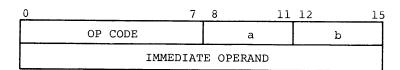
All MODCOMP II instructions are described in this chapter. The instructions are grouped in the functional classes:

- . Load, Store and Transfer
- . Arithmetic
- . Floating Point
- . Logical
- . Shift
- . Bit Manipulation
- . Byte Manipulation
- . Unconditional Branch
- . Control
- . Interrupt and Call
- . Input/Output

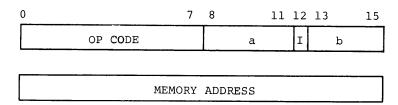
The principal MODCOMP instruction formats are:



Single Word Format



Immediate Operand Format



Two Word Format

Where: a and b define operand registers, index registers, bit address within a word, displacement address (up to 16 locations) with respect to a base address, shift count, interrupt level or peripheral device address and I specifies indirect addressing.

The general format for the instruction description is:

INSTRUCTION NAME

0 3	4	7	8	11 12	15
OP	CODE		Ra		Rb

Execution Description

MNEMONIC

#### Affected:

The Mnemonic is a three or four letter representation of the instruction name.

The Instruction Name briefly describes the function performed by the execution of the instruction.

EXECUTION TIME

The Execution Time is maximum (not average or minimum) and includes access time.

The Operation Code value is shown as two hexadecimal digits. The two right digits contain binary coded register addresses in many instructions and other binary coded fields in other instructions, as described. In all instructions in which the contents of register Rb, either with or without manipulation, are transferred to register Ra, the two register addresses may be made the same to produce a single register operation. For example, the contents of a register can be one's complemented by making the Ra and Rb addresses equal in the instruction Transfer One's Complement Register to Register.

Many instructions contain a second and some a third instruction word used for 16-bit memory addresses or immediate operands. The address in the Program Register (PR) referenced in the description of these instructions is that of the first instruction word.

The Execution Description covers all program controlled functions performed in the computer which comprise the instruction execution. In addition, the contents of the Program Register are advanced to the first word of the next instruction.

The Affected line lists all general registers and memory cells in which the contents are modified as a result of the execution of the instruction. In addition, if the execution of the instruction can cause overflow, the word "overflow" is included in the listing.

The symbols and abbreviations used in the instruction descriptions are listed alphabetically in the following table.

В	<ul> <li>Byte designator bit (0 = left byte, 1 = right byte)</li> </ul>
DF	- Displacement Field, which is used in the short displaced
	addressing mode and has the value range 0 $\leq$ DF $\leq$ 15
ŁA	- Effective memory address, which is the address that results
	after all specified address manipulation operations have been
	completed
I	- Indirect address bit
PR	- Program Register, which is a 16-bit register containing the
	current program location
Ra	- General register Ra, which is the operand destination register
	for many instructions
Ra, Ra	Vl - Doubleword consisting of the concatenated values stored in
	register Ra (more significant half) and register RaVl (less
	significant half), where Ra is even numbered register*
Ra <sub>n</sub>	- Bit n of register Ra
Rb	- General register Rb, which is the operand source register for
	many instructions
Rxx	- General register Rxx, $(1 \le xx \le 7)$ is the index register for
	many instructions. When $Rxx = 0$ , no index operation occurs
Rx	- Effective address register for short indexed instructions
	$(0 \leq x \leq 15)$
S	- Sign bit
μs	- Microseconds
()	- Contents of
<b>→</b>	- Replace the contents of
+	- Addition operator
-	- Subtraction operator
X	- Multiplication operator
÷	- Division operator
Λ	- Logical AND operator
V	- Logical OR operator
$\underline{\mathbb{Q}}$	- Logical Exclusive OR operator
( )	- Logical NOT (One's complement) operator*

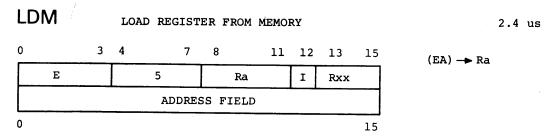
TABLE 3-1 Symbols and Abbreviations

\*Ra,RaVl normally indicate an even/odd register pair, 4 and 5 for example. RaVl indicates that a binary one is logically OR'ed with Ra (hex value) so it follows that Ra,RaVl cannot describe an even/odd register pair. If Ra = 5 then RaVl also = 5.

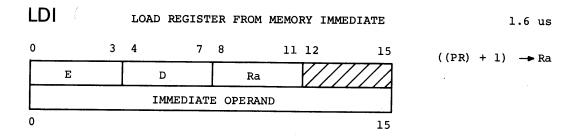
<sup>\*\*</sup>The 'Contents of' symbol ( ) is shown merely to show the physical position of the overline and is not necessarily part of the NOT symbol.

# LOAD, STORE AND TRANSFER INSTRUCTIONS

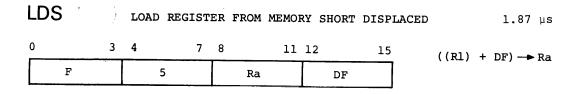
This instruction group provides the capability to transfer information from memory to the general register file (load), from the general register file to memory (store) and from register to register (transfer). Either a byte, word or file consisting of from one to eight words can be transferred by single instruction execution. The word transfer instruction set includes all seven memory addressing modes - direct, indexed, indirect, indirect and indexed, immediate, short displaced and short indexed.



The contents of the effective memory location replace the contents of register Ra. Affected: Ra



The contents of the second instruction word replace the contents of register Ra. Affected: Ra



The contents of the memory location specified by the displacement field DF added to the contents of register Rl replace the contents of register Ra.

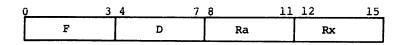
Affected: Ra

Load, Store, Transfer Instructions

LDX

LOAD REGISTER FROM MEMORY SHORT INDEXED

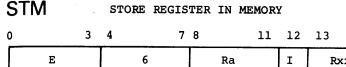
1.87 us



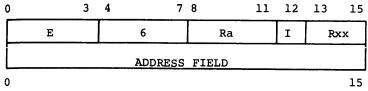
((Rx)) → Ra

The contents of the memory location specified by the contents of register Rx replace the contents of register Ra.

Affected: Ra



2.4 us



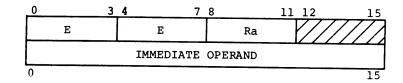
(Ra) → EA

The contents of register Ra replace the contents of the effective memory location. Affected: (EA)

STI

STORE REGISTER IN MEMORY IMMEDIATE

1.6 us



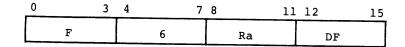
 $(Ra) \rightarrow (PR) + 1$ 

The contents of register Ra replace the contents of the second instruction word. Affected: ((PR) + 1)

STS

STORE REGISTER IN MEMORY SHORT DISPLACED

1.87 us

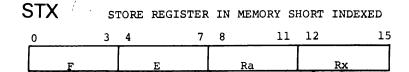


(Ra) → (R1) + DF

The contents of register Ra replace the contents of the memory location specified by the displacement field DF added to the contents of register Rl.

Affected: (EA)

Load, Store, Transfer
Instructions

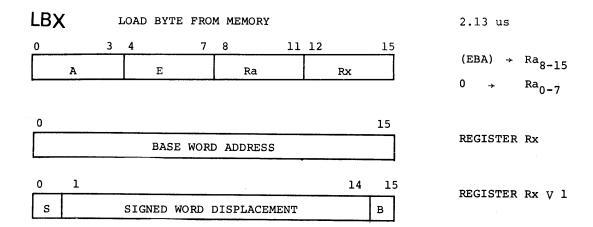


1.87 us

(Ra) → Rx

The contents of register Ra replace the contents of the memory location specified by the contents of register Rx.

Affected: (EA)



The contents of the effective byte location replace the right byte in register Ra. Zeroes replace the left byte in register Ra. Register Rx specifies an even/odd pair of general registers which contain the base word address and the signed byte displacement. The byte designator B specifies the byte within the memory word (0 = left, 1 = right).

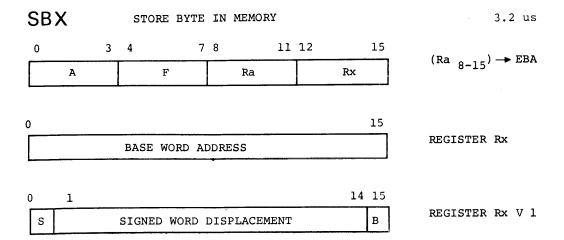
Affected: Ra

### Effective Byte Address Generation

Byte addressing is a special form of short indexed addressing. The effective byte address is generated by the addition of the base word address and the signed byte displacement which consists of the signed word displacement and a byte designator B. During the instruction execution the signed word displacement is right shifted by one bit position and is then added to the base word address to form an effective word address. The equation can be interpreted as: EBA = RX + RX + V + 1

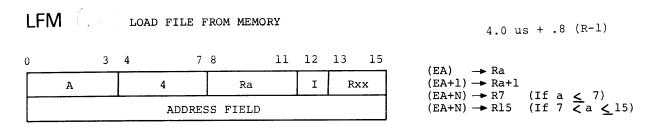
B = 0 Specifies the byte contained in bits 0-7

B = 1 Specifies the byte contained in bits 8-15



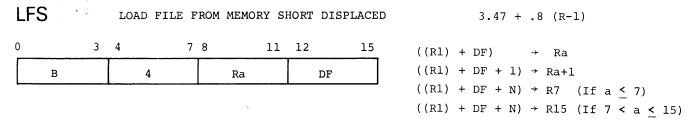
The right byte in register Ra replaces the contents of the effective byte location. The other byte in the memory word is not affected. The byte designator B specifies the byte within the memory (0 = left, 1 = right). See Effective Byte Address Generation under the description of the Load Byte From Memory instruction.

Affected: (EBA)



The contents of from one to eight consecutive memory location starting with the effective memory location replace the contents of register Ra through R7, if a  $\leq$  7, or register Ra through R15, if 7  $\langle$  a  $\leq$  15.

Affected: Ra through R7/15



The contents of from one to eight consecutive memory locations starting with the location specified by the displacement field DF added to the contents of register R1 replace the contents of registers Ra through R7, if a  $\leq 7$ , or register Ra through R15, if 7  $\leq$  a  $\leq 15$ .

Affected: Ra through R7/15

LFX

LOAD FILE FROM MEMORY SHORT INDEXED

$$3.47 + .8 (R-1)$$

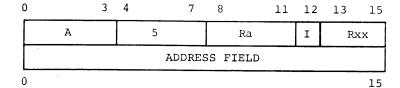
0	. 3	4	7 8	11 12	15
	В	С	Ra	F	₹x

$$\begin{array}{cccc} ((Rx)) & \longrightarrow & Ra \\ ((Rx)+1) & \longrightarrow & Ra+1 \\ ((Rx)+N) & \longrightarrow & R7 & (If & a \leq 7) \\ ((Rx)+N) & \longrightarrow & R15 & (If & 7 < a \leq 15) \end{array}$$

The contents of from one to eight consecutive memory locations starting with the location specified by the contents of Rx replace the contents of registers Ra through R7, if a  $\leq 7$ , or register Ra through R15, if 7 < a  $\leq 15$ . Affected: Ra through R7/15

SFM

STORE FILE IN MEMORY



4.0 us + .8 (R-1)

(Ra) 
$$\longrightarrow$$
 EA  
(Ra+1)  $\longrightarrow$  EA+1  
(R7)  $\longrightarrow$  EA+N (If a  $\leq$  7)  
(R15)  $\longrightarrow$  EA+N (If 7  $<$  a  $\leq$  15)

The contents of registers Ra through R7, if  $a \le 7$ , or registers Ra through R15, if  $7 < a \le 15$ , replace the contents of from one to eight consecutive memory locations starting with the effective memory location.

Affected: (EA) ... (EA+N)

**SFS** 

STORE FILE IN MEMORY SHORT DISPLACED

$$3.47 \text{ us} + .8 \text{ (R-1)}$$

0 3	4 7	8 11	12	15 (Ra) (Ra+1)	$\vec{\neg}$
В	5	Ra	DF	(	_
L	<u> </u>		L	(R15)	-

(Ra)	-	(R1)+DF			
(Ra+1)	-	(R1)+DF+1			
(R7)	-	(R1) + DF + N	(If	a	≤ 7)
(R15)	-	(R1)+DF+N	(If	7 <	a < 15)

The contents of registers Ra through R7, if  $a \le 7$ , or registers Ra through R15, if  $7 < a \le 15$ , replace the contents of from one to eight consecutive memory locations starting with the location specified by the displacement field DF added to the contents of register R1.

Affected: (EA) ... (EA+N)

SFX

STORE FILE IN MEMORY SHORT INDEXED

$$3.47 \text{ us} + .8 \text{ (R-1)}$$

0 3	4 7	8 11	12 15
В	D	Ra	Rx

The contents of registers Ra through R7, if  $a \le 7$ , or registers Ra through R15, if  $7 < a \le 15$ , replace the contents of from one to eight consecutive memory locations starting with the location specified by the contents of Rx.

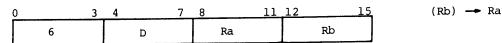
Affected: (EA) ... (EA+N)

Load, Store, Transfer Instructions

TRR

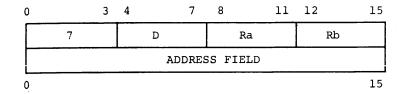
TRANSFER REGISTER TO REGISTER

0.8 us



The contents of register Rb replace the contents of register Ra. Affected: Ra

TRRB TRANSFER REGISTER TO REGISTER AND BRANCH IF NONZERO 1.6 us



(Rb)  $\longrightarrow$  Ra If Result  $\neq 0$ , EA  $\longrightarrow$  PR If Result =0, (PR)+2  $\longrightarrow$  PR

The contents of register Rb replace the contents of register Ra.

If the results are unequal to zero, a branch is executed to the effective word location. If the results equal zero, the next instruction in sequence is executed.

### **ARITHMETIC INSTRUCTIONS**

This instruction group includes the add, double-precision add, subtract, multiply, divide, compare, and two's complement instructions.

All instructions assume fixed-point operands, which may be scaled at any bit position. The double-precision add and divide instructions assume doubleword operands. All other instructions assume word operands.

All instructions, except the multiply and compare, produce an overflow if the conditions described with each instruction are met.

The multiply/divide instructions are a compatible set. Not only is the relationship true:  $(A \times B) \div A = B$ , but also the positioning of the operands and results are consistent. In multiply operations, if Ra specifies an even numbered general register, the doubleword product is then stored in the even-odd register pair consisting of Ra and RaVl. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the multiplier in Ra. In divide, the doubleword dividend must be stored in an even-odd register pair Ra and RaVl. The quotient is then stored in RaVl and the remainder in Ra. Therefore, the multiplier and quotient occupy the same register positions, which simplifies computations.

The maximum values of the products for word operand pairs having all combinations of signs are:

Operand Signs	Maximum Operands		Maximum Product
(+ x +)	$(2^{15}-1) \times (2^{15}-1)$	= ,	$2^{30} - 2^{16} + 2^{0}$
(+ x <sup>-</sup> )	$(2^{15}-1) \times 2^{15}$	=	2 <sup>30</sup> - 2 <sup>15</sup>
(- x -)	$2^{15} \times 2^{15}$	=	2 <sup>30</sup>

-where minus full scale = 1000 0000 0000 0000<sub>2</sub> =  $2^{15}$ .

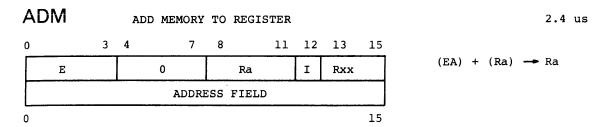
None of these numbers exceed the capacity of a doubleword and therefore overflow cannot occur.

In the divide operation, overflow will occur if the quotient exceeds 16 bits in length. Two checks are made by the overflow checking logic to determine if this error condition exists:

- The sign and most significant bit of the dividend are compared. They must be equal; otherwise overflow will occur.
- (2) The dividend is shifted left one bit position and then the divisor is subtracted from the most significant half. Overflow will occur if the absolute magnitude of the most significant half of the shifted dividend is not less than the absolute magnitude of the divisor.

As a result of the overflow logic the absolute magnitude of the largest permissable dividend is  $2^{30} - 2^{15} - 2^0$ .

Divide scaling is described in Appendix D.



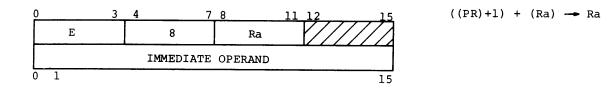
The contents of the effective memory location are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow



ADD MEMORY TO REGISTER IMMEDIATE

1.6 us



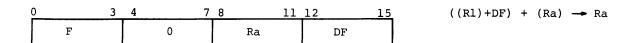
The contents of the second instruction word are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow

### **ADS**

ADD MEMORY TO REGISTER SHORT DISPLACED

1.87 us



The contents of the memory location specified by the displacement field DF added to the contents of register Rl are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

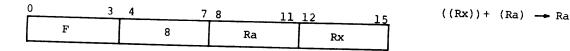
Affected: Ra, Overflow

Arithmetic Instructions

ADX

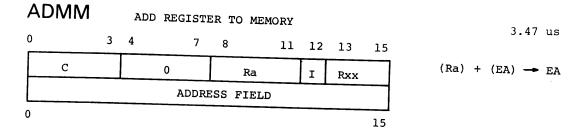
ADD MEMORY TO REGISTER SHORT INDEXED

1.87 us

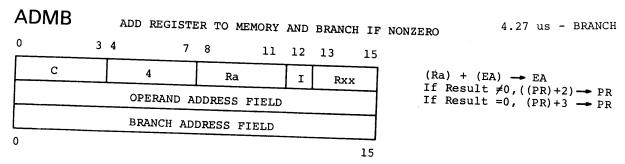


The contents of the memory location specified by the contents of register Rx are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow



The contents of register Ra are algebraically added to the contents of the effective memory location. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign. Affected: Overflow, (EA)



The contents of register Ra are algebraically added to the contents of the effective memory location. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed for the branch address. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow, (EA)

**ADSM** 

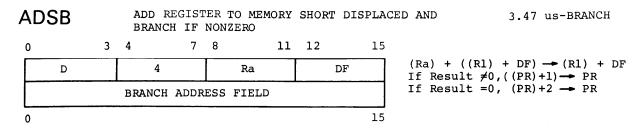
ADD REGISTER TO MEMORY SHORT DISPLACED

2.67 us

0 3 4 7 8 11 12 15 
$$(Ra) + ((R1) + DF) \rightarrow (R1) + DF$$
D Ra DF

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the displacement field DF added to the contents of register Rl. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)



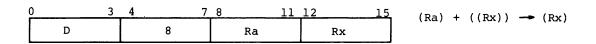
The contents of register Ra are algebraically added to the contents of the effective memory location specified by the displacement field DF added to the contents of register Rl. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the memory location specified by the contents of the second instruction word. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow , (EA)

**ADXM** 

ADD REGISTER TO MEMORY SHORT INDEXED

2.67 us



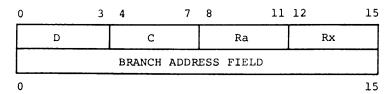
The contents of register Ra are algebraically added to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Overflow, (EA)

**ADXB** 

ADD REGISTER TO MEMORY SHORT INDEXED AND BRANCH IF NONZERO

3.47 us-BRANCH



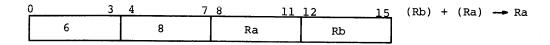
(Ra) + ((Rx))  $\longrightarrow$  (Rx) If Result  $\neq$ 0,((PR)+1)  $\longrightarrow$  PR If Result =0, (PR)+2  $\longrightarrow$  PR

The contents of register Ra are algebraically added to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the memory location specified by the contents of the second instruction word. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign. Affected: Overflow, (EA)

**ADR** 

ADD REGISTER TO REGISTER

0.8 us



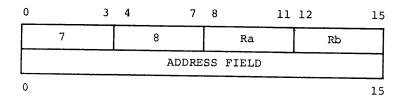
The contents of register Rb are algebraically added to the contents of register Ra. The result is stored in register Ra. An overflow occurs if both operands have like signs but the result has the opposite sign.

Affected: Ra, Overflow

### ADRB

ADD REGISTER TO REGISTER AND BRANCH IF NONZERO

1.6 us



(Rb) + (Ra)  $\longrightarrow$  Ra If Result  $\neq 0$ , EA  $\longrightarrow$  PR If Result =0, (PR)+2  $\longrightarrow$  PR

The contents of register Rb are algebraically added to the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if both operands have like signs but the result has the opposite sign.

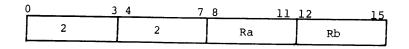
Affected: Ra, Overflow

Arithmetic Instructions

DAR

DOUBLE PRECISION ADD REGISTER TO REGISTER

2.13 us



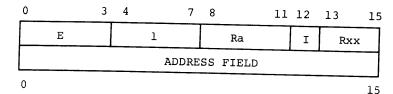
(Rb, RbV1) + (Ra, RaV1) → Ra, RaVl

The contents of registers Rb and RbVl (with register Rb containing the more significant half and register RbVl containing the less significant half of a double precision data word) are algebraically added to the contents of registers Ra and RaVl (with register Ra containing the more significant half and register RaVl containing the less significant half of a double precision data word). The sum replaces the contents of registers Ra and RaVl. Ra and Rb must specify even-numbered general registers. Affected: Ra, RaVl, Overflow

SUM

SUBTRACT MEMORY FROM REGISTER

2.4 us



(Ra) - (EA) → Ra

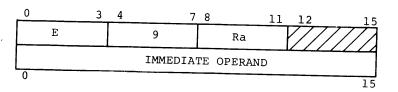
The contents of the effective memory location are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUL

SUBTRACT MEMORY FROM REGISTER IMMEDIATE

1.6 us



 $(Ra) - ((PR)+1) \longrightarrow Ra$ 

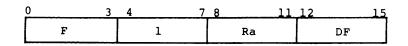
The contents of the second instruction word are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

# SUS

SUBTRACT MEMORY FROM REGISTER SHORT DISPLACED

1.87 us



$$(Ra) - ((R1) + DF) \longrightarrow Ra$$

The contents of the memory location specified by the displacement field added to the contents of register Rl are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs if the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUX ( subtract memory from register short indexed

1.87 us

0 3	4 7	8 11	12 15
F	9	Ra	Rx

$$(Ra) - ((Rx)) \longrightarrow Ra$$

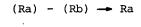
The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SUR ( SUBTRACT REGISTER FROM REGISTER

0.8 us

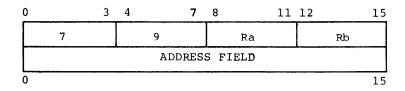
0 3	4 7	8 11	12 15
6	9	Ra	Rb



The contents of register Rb are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow

SURB ( ) SUBTRACT REGISTER FROM REGISTER AND BRANCH IF NONZERO



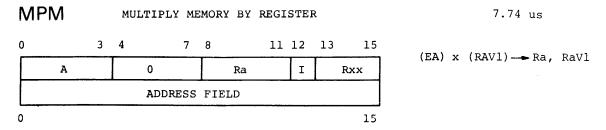
(Ra) - (Rb) 
$$\longrightarrow$$
 Ra  
If Result  $\neq 0$ , EA  $\longrightarrow$  PR  
If Result =0, (PR)+2  $\longrightarrow$  PR

1.6 us

The contents of register Rb are algebraically subtracted from the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next

instruction in sequence is executed. An overflow occurs when the sign of the result is the same as the sign of the subtrahend but is different from the sign of the minuend.

Affected: Ra, Overflow



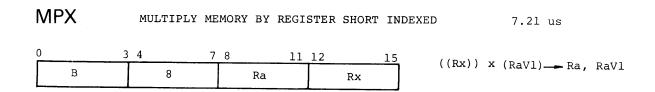
The contents of the effective memory location (multiplicand) are multiplied by the contents of register RaVl (multiplier). Ra normally specifies an even register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaVl. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaVl

	MPS		MULTIPLY	MEMORY	ВУ	REGI	STER	SHORT	DI	SPLACED	7.21 us	
	0	3	4	7 8		11	12		15	(/D1\  DE\	(5)	
i	В		0		Ra			DF		((RI)+DF)	$x (RaV1) \longrightarrow Ra,$	RaVl

The contents of the memory location specified by the displacement field DF added to the contents of register Rl (multiplicand) are multiplied by the contents of register RaVl(multiplier). Ra normally specifies an even register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaVl. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaVl



The contents of the memory location specified by the contents of Rx (multiplicand) are multiplied by the contents of register RaVl(multiplier). Ra normally specifies an even numbered register so that the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaVl. The sign of the

product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of register Ra.

Affected: Ra, RaVl

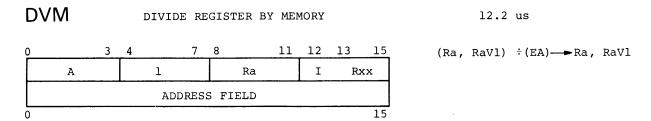
 MPR
 MULTIPLY REGISTER BY REGISTER
 6.67 us

 0
 3 4
 7 8
 11 12
 15
 (Rb) x (RaV1) → Ra, RaV1

 2
 0
 Ra
 Rb

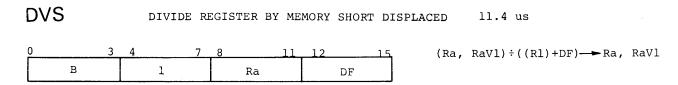
The contents of register Rb (multiplicand) are multiplied by the contents or register RaV1 (multiplier). Ra normally specifies an even numbered register so that the more significant half of the product replaces the contents of register Ra and the less significant half of the product replaces the contents of register RaV1. The sign of the product replaces the sign bit of register Ra. If Ra specifies an odd numbered register, the least significant 16 bits of the product replace the contents of Ra.

Affected: Ra, RaVl



The contents of the effective memory location (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.

Affected: Ra, Rayl Overflow



The contents of the memory location specified by the displacement field DF added to the contents of register Rl (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the

sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.

Affected: Ra, RaVl Overflow

DVX

DIVIDE REGISTER BY MEMORY SHORT INDEXED

11.4 us

0 3	4 7	8 11	12 15
В	9	Ra	Rx

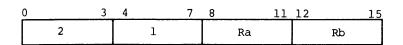
 $(Ra, RaVl) \div ((Rx)) \rightarrow Ra, RaVl$ 

The contents of the memory location specified by the contents of register Rx (divisor) are divided into the contents of registers Ra and Rayl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits. Affected: Ra, RaVl Overflow

DVR

DIVIDE REGISTER BY REGISTER

11.0 us



(Ra, RaVl)÷(Rb) → Ra, RaVl

The contents of register Rb (divisor) are divided into the contents of registers Ra and RaVl (dividend). The quotient replaces the contents of register RaVl and the remainder replaces the contents of register Ra. The sign of the quotient replaces the sign bit of register RaVl. Ra must specify an even numbered register. Overflow will occur if the quotient exceeds 16 bits.

Affected: Ra, RaVl Overflow

CRMB

COMPARE MEMORY AND REGISTER

4.53 us

0		3	4	7	8	11	12	13	15	
	С		7			Ra	I		Rxx	
	OPERAND ADDRESS FIELD									
	BRANCH ADDRESS FIELD (Ra) = (EA)									
L		]	BRANCH	ADDRES	SS FII	ELD (Ra)	<	(EA)		

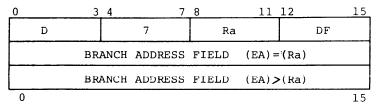
If (Ra) - (EA) = 0,  $((PR)+2) \rightarrow PR$ If (Ra) - (EA) < 0,  $((PR)+3) \rightarrow PR$ If (Ra) - (EA) > 0,  $(PR)+4 \rightarrow PR$ 

The contents of the effective memory location are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the third instruction word. If the result is negative, a branch is executed to the location specified by the fourth instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. Affected: None

**CRSB** 

COMPARE MEMORY AND REGISTER SHORT DISPLACED

4.0 us



```
If (Ra)-((R1)+DF)=0, ((PR)+1) \rightarrow PR

If (Ra)-((R1)+DF)<0, ((PR)+2) \rightarrow PR

If (Ra)-((R1)+DF)>0, (PR)+3 \rightarrow PR
```

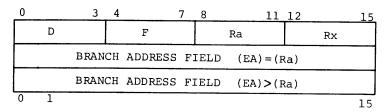
The contents of the memory location specified by the displacement field added to the contents of register Rl are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed.

Affected: None

**CRXB** 

COMPARE MEMORY AND REGISTER SHORT INDEXED

4.0 us



If 
$$(Ra)-((Rx))=0$$
,  $((PR)+1) \rightarrow PR$   
If  $(Ra)-((Rx))<0$ ,  $((PR)+2) \rightarrow PR$   
If  $(Ra)-((Rx))>0$ ,  $(PR)+3 \rightarrow PR$ 

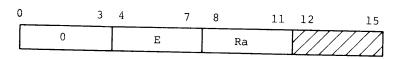
The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the contents of register Ra. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed.

Affected: None

TRO

TRANSFER AND RESET OVERFLOW STATUS

0.8 us



(OVERFLOW) 
$$\rightarrow$$
 Ra<sub>0</sub>  
(CARRY SAVE)  $\rightarrow$  Ra<sub>15</sub>  
0  $\rightarrow$  OVERFLOW, Ra<sub>1-14</sub>

The content of the overflow latch is transferred into the most significant bit of Ra and the last adder carry out saved is transferred into the least significant bit of Ra. Bits 1-14 of register Ra are set to zero and the overflow latch is reset by the execution of this instruction.

Affected: Ra, Overflow

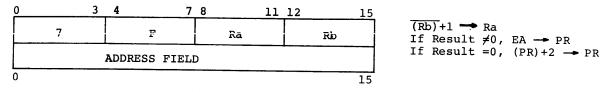
 TRANSFER TWO'S COMPLEMENT REGISTER TO REGISTER
 0.8 us

 0
 3 4
 7 8
 11 12 15 (Rb) +1 → Ra
 6
 Ra
 Rb

The contents of register Ra are replaced by the two's complement of register Rb. An overflow occurs if the operand is minus full scale.

Affected: Ra, Overflow

TRANSFER TWO'S COMPLEMENT REGISTER TO REGISTER AND BRANCH 1.6 us

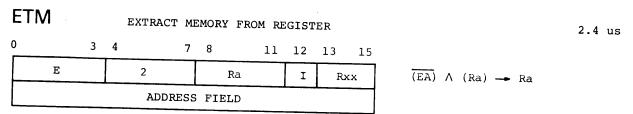


The contents of register Ra are replaced by the two's complement of the contents of register Rb. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed. An overflow occurs if the operand is minus full scale.

Affected: Ra, Overflow

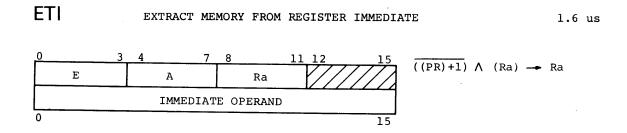
### LOGICAL INSTRUCTIONS

This group consists of the Extract  $(\overline{A} \wedge B)$ , OR  $(A \vee B)$ , Exclusive OR  $(A \vee B)$ , One's Complement, and Test instructions. All of these instructions operate on 16-bit operands. They produce a logical product (Extract), sum (OR), modulo-two sum (Exclusive OR), or complement and all but the Test instructions store the result in a general register or memory location. The Test instructions enable a comparison to be made between two operands without modifying either.



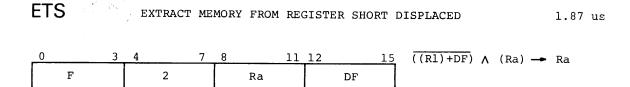
The one's complement of the contents of the effective memory location are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

Affected: Ra



The one's complement of the contents of the second instruction word are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

Affected: Ra



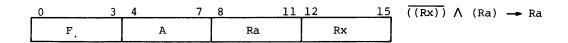
The one's complement of the contents of the memory location specified by the displacement field DF added to the contents of register Rl are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra. Affected: Ra

Logical Instructions

ETX

EXTRACT MEMORY FROM REGISTER SHORT INDEXED

1.87 us

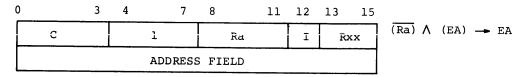


The one's complement of the contents of the memory location specified by the contents of register Rx are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

Affected: Ra



3.47 us



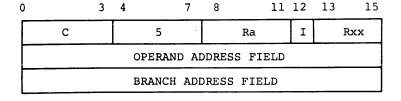
The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location. The result is stored in the effective memory location.

Affected: (EA)



EXTRACT REGISTER FROM MEMORY AND BRANCH IF NONZERO

4.27 us-NO BRANCH



(Ra)  $\wedge$  (EA)  $\longrightarrow$  EA If Result  $\neq$ 0,((PR)+2)  $\longrightarrow$  PR If Result =0, (PR)+3  $\longrightarrow$  PR

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

**ETSM** 

EXTRACT REGISTER FROM MEMORY SHORT DISPLACED

2.67 us

0	3	4 7	8 11	12 15
	D	1	Ra	DF

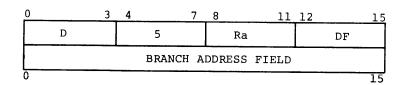
 $\overline{(Ra)} \wedge ((R1)+DF) \longrightarrow (R1)+DF$ 

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the displacement field DF added to the contents of register Rl. The result is stored in the effective memory location.

Affected: (EA)

**ETSB** 

EXTRACT REGISTER FROM MEMORY SHORT DISPLACED AND BRANCH 3.47 us



(Ra)  $\wedge$  ((R1)+DF)  $\longrightarrow$  (R1)+DF If Result  $\neq$ 0 ((PR)+1)  $\longrightarrow$  PR If Result =0 (PR)+2  $\longrightarrow$  PR

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the displacement field DF added to the contents of register Rl. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

**FTXM** 

EXTRACT REGISTER FROM MEMORY SHORT INDEXED

2.67 us

0 3	4 7	8 11	12 15
D	9	Ra	Rx

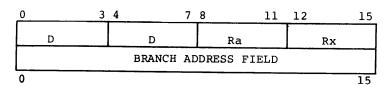
 $\overline{(Ra)} \wedge ((Rx)) \longrightarrow (Rx)$ 

The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location.

Affected: (EA)

**ETXB** 

EXTRACT REGISTER FROM MEMORY SHORT INDEXED AND BRANCH 3.47 us



 $\overline{(Ra)} \wedge ((Rx)) \longrightarrow (Rx)$ If Result  $\neq 0$   $((PR)+1) \longrightarrow PR$ If Result = 0  $(PR)+2 \longrightarrow PR$ 

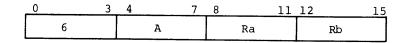
The one's complement of the contents of register Ra are logically multiplied (AND function) by the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location. If the result does not equal zero, a branch is executed to the location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Affected: (EA)

**FTR** 

EXTRACT REGISTER FROM REGISTER

0.8 us

1.6 us

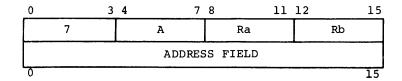


(Rb) ∧ (Ra) → Ra

The one's complement of the contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra. Affected: Ra

**ETRB** 

EXTRACT REGISTER FROM REGISTER AND BRANCH IF NONZERO



 $\overline{(Rb)} \land (Ra) \longrightarrow Ra$ If Result  $\neq 0$ , EA  $\longrightarrow$  PR
If Result =0, (PR)+2  $\longrightarrow$  PR

The one's complement of the contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is stored in register Ra.

If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Logical Instructions

ORM
OR MEMORY AND REGISTER

0 3 4 7 8 11 12 13 15

E 3 Ra I Rxx

ADDRESS FIELD

2.4 us

(EA)  $\vee$  (Ra)  $\rightarrow$  Ra

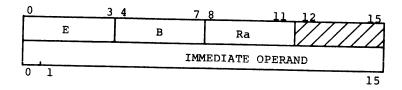
The contents of the effective memory location are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.

Affected: Ra

### ORI

OR MEMORY AND REGISTER IMMEDIATE

1.6 us



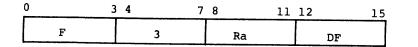
((PR)+1) V (Ra) → Ra

The contents of the second instruction word are logically added (OR function) to the contents of register Ra. The result is stored in register Ra. Affected: Ra

# **ORS**

OR MEMORY AND REGISTER SHORT DISPLACED

1.87 us



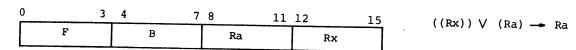
 $((R1)+DF) V (Ra) \longrightarrow Ra$ 

The contents of the memory location specified by the displacement field DF added to the contents of register Rl are logically added (OR function) to the contents of register Ra. The result is stored in register Ra. Affected: Ra

### ORX

OR MEMORY AND REGISTER SHORT INDEXED

1.87 us



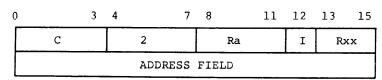
The contents of the memory location specified by the contents of register Rx are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.

Logical Instructions

**ORMM** 

OR REGISTER AND MEMORY

3.47 us



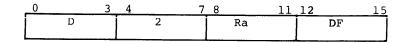
(Ra)  $\vee$  (EA)  $\longrightarrow$  EA

The contents of register Ra are logically added (OR function) to the contents of the effective memory location. The result is stored in the effective memory location. Affected: (EA)

ORSM

OR REGISTER AND MEMORY SHORT DISPLACED

2.67 us



 $(Ra) \lor ((R1)+DF) \longrightarrow (R1)+DF$ 

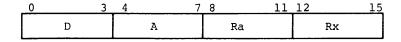
The contents of register Ra are logically added (OR function) to the contents of the effective memory location specified by the displacement field DF added to the contents of register Rl. The result is stored in the effective memory location.

Affected: (EA)

**ORXM** 

OR REGISTER AND MEMORY SHORT INDEXED

2.67 us



(Ra) V ((Rx))  $\rightarrow$  (Rx)

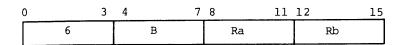
The contents of register Ra are logically added (OR function) to the contents of the effective memory location specified by the contents of register Rx. The result is stored in the effective memory location.

Affected: (EA)

ORR

OR REGISTER AND REGISTER

0.8 us



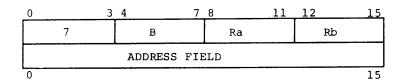
(Rb) V (Ra) → Ra

The contents of register Rb are logically added (OR function) to the contents of register Ra. The result is stored in register Ra.

### **ORRB**

OR REGISTER AND REGISTER AND BRANCH IF NONZERO

1.6 us



(Rb) 
$$\lor$$
 (Ra)  $\longrightarrow$  Ra  
If Result  $\neq 0$ , EA  $\longrightarrow$  PR  
If Result =0, (PR)+2 $\longrightarrow$  PR

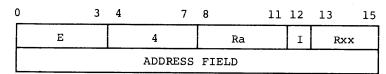
The contents of register Rb are logically added (OR function) to the contents of register Ra. The result is stored in register Ra. If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra

# **XOM**

EXCLUSIVE OR MEMORY AND REGISTER

2.4 us

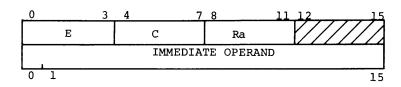


The contents of the effective memory location are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra. Affected: Ra

### XOI

EXCLUSIVE OR MEMORY AND REGISTER IMMEDIATE

1.6 us



((PR)+1) **(** (Ra) → Ra

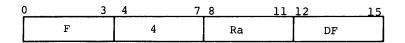
The contents of the second instruction word are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra. Affected: Ra

Logical Instructions

XOS

EXCLUSIVE OR MEMORY AND REGISTER SHORT DISPLACED

1.87 us



((R1)+DF)  $\bigvee$   $(Ra) \longrightarrow Ra$ 

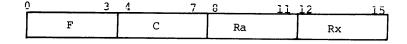
The contents of the memory location specified by the displacement field DF added to the contents of register Rl are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

Affected: Ra

XOX

EXCLUSIVE OR MEMORY AND REGISTER SHORT INDEXED

1.87 us



((Rx))  $(Ra) \rightarrow Ra$ 

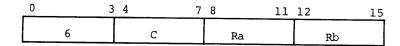
The contents of the memory location specified by the contents of register Rx are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

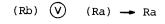
Affected: Ra

**XOR** 

EXCLUSIVE OR REGISTER AND REGISTER

0.8 us

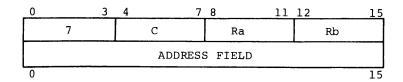


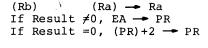


The contents of register Rb are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra. Affected: Ra

**XORB** 

EXCLUSIVE OR REGISTER AND REGISTER AND BRANCH IF NONZERO 1.6 us





The contents of register Rb are logically added modulo two (Exclusive Or function) to the contents of register Ra. The result is stored in register Ra.

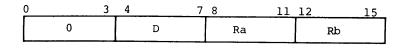
If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Logical Instructions

TOR

TRANSFER ONE'S COMPLEMENT REGISTER TO REGISTER

0.8 us



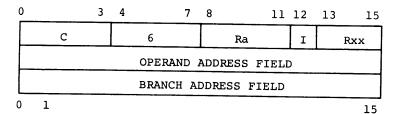
(Rb) - Ra

The one's complement of the contents of register Rb replaces the contents of register Ra.

Affected: Ra

**TRMB** 

TEST REGISTER AND MEMORY AND BRANCH IF ANY ONES COMPARE 3.73 us



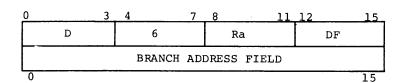
(Ra)  $\wedge$  (EA)  $\neq 0$ , ((PR) +2)  $\longrightarrow$  PR (Ra)  $\wedge$  (EA) =0, (PR) +3  $\longrightarrow$  PR

The contents of the effective memory location are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to the location specified by the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Affected: None

**TRSB** 

TEST REGISTER AND MEMORY SHORT DISPLACED AND BRANCH
1F ANY ONES COMPARE
2.99 us



(Ra)  $\Lambda$  ((R1)+DF) $\neq 0$ , ((PR)+1)  $\longrightarrow$  PR (Ra)  $\Lambda$  ((R1)+DF)=0, (PR)+2  $\longrightarrow$  PR

The contents of the memory location specified by the displacement field added to the contents of register Rl are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Affected: None

**TRXB** 

TEST REGISTER AND MEMORY SHORT INDEXED AND BRANCH IF ANY ONES COMPARE

2.93 us

0		3	4	7	8 11	12	15
	D		E		Ra	Rx	l
			BRANCH	ADI	DRESS FIELD		
0							15

(Ra)  $\Lambda$  ((Rx)) $\neq 0$ , ((PR)+1)  $\longrightarrow$  PR (Ra)  $\Lambda$  ((Rx))=0, (PR)+2  $\longrightarrow$  PR

The contents of the memory location specified by the contents of register Rx are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to location specified by the second instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Affected: None

**TERB** 

TEST REGISTER AND REGISTER AND BRANCH IF ANY ONES COMPARE

7 8 11 12 15
E Ra Rb

ADDRESS FIELD

(Rb)  $\land$  (Ra)  $\longrightarrow$  RESULT If Result  $\neq 0$ , EA  $\longrightarrow$  PR If Result =0, (PR)+2  $\longrightarrow$  PR

1.6 us

The contents of register Rb are logically multiplied (AND function) by the contents of register Ra. The result is not stored.

If the result does not equal zero, a branch is executed to the effective word location. If the result equals zero, the next instruction in sequence is executed.

Affected: None

7

### FLOATING POINT INSTRUCTIONS

#### INTRODUCTION

The optional floating point arithmetic instructions provide the capability to process very large or very small magnitude operands with precise results.

Floating point numbers consist of three parts: a sign, an exponent and a fraction. The sign bit applies only to the fraction. The exponent is a biased nine-bit binary number. The fraction is a binary number with an assumed radix point to the left of the high-order digit. The quantity that the floating-point number represents is obtained by raising the fraction value to the power expressed in the exponent value.

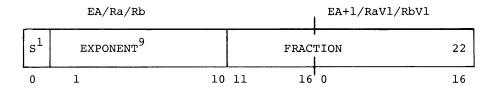
#### Data Formats

Floating point numbers are fixed in length and are either two word single precision or three word double precision in format.

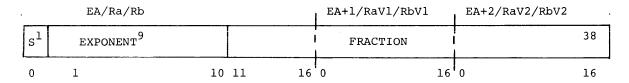
The first bit (bit 0) in both formats is the sign of the fraction. A one (1) bit represents a minus sign and a zero bit (0) represents a positive sign. The next nine bits  $(2^1-2^{10})$  represent a biased binary exponent. The fraction contains a 22 bit binary number (single-precision format) or a 38 bit binary number (double precision format).

The single precision format allows faster processing and uses less storage. The double precision format while providing greater precision, requires more processing time and use of an additional register and/or memory location.

Single Precision Floating Point Number



Double Precison Floating Point Number



Ra OR Rb FIELD	SINGLE PRECISION OPERAND (OR RESULTS) REGISTERS USED	DOUBLE PRECISION OPERAND (OR RESULTS) REGISTERS USED
0	0, 1	0, 1, 2
1	1, 1	1, 1, 3
2	2, 3*	2, 3, 2
3	3, 3	3, 3, 3
4	4, 5*	4, 5, 6
5	5, 5	5, 5, 6
6	6, 7*	6, 7, 6
7	7, 7*	7, 7, 7
8	8, 9*	8, 9, A*
9	9, 9	9, 9, B
A	A, B*	A, B, A
В	В, В	В, В, В
С	C, D*	C, D, E*
D	D, D	D, D, F
E	E, F*	E, F, E
F	F, F	F, F, F

<sup>\*</sup> Indicates all normally useful selections

TABLE 3-2 Floating Point Register Selections

### FLOATING POINT INSTRUCTION MNEMONICS

This group of 16 optional instructions is made up of the four arithmetic operations; add, subtract, multiply and divide. Each of the four arithmetic operations can be executed in register-to-register or memory-to-register formats with either single precision or double precision operands.

<u>Add</u>	Subtract	Multiply	Divide
FAR	FSR	FMR	FDR Reg-Reg
FARD	FSRD	FMRD	FDRD R-R Double
FAM	FSM	FMM	FDM Mem-Reg
FAMD	FSMD	FMMD	FDMD M-Reg Double

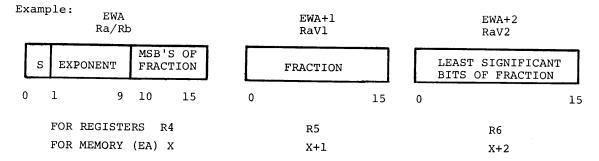
#### GENERAL RULES

When floating point instructions specify Ra,RaVl and Rb,RbVl, Ra and Rb must specify even numbered registers which will contain the more significant half of a single precision floating point operand, and RaVl, RbVl will specify the next sequential odd numbered registers and hold the less significant half of a single precision floating point operand. Refer to Table 3-2 for normally useful register selections.

Operands presented to the Floating Point Unit will be in normalized form and likewise, operation results will always be normalized.

The exceptions to this rule are when an unnormalized number is presented to the Floating Point Unit for normalization (e.g. 0 + an unnormalized number will yield the same number in a normalized format), and when a zero fraction is used.

The storage of floating point operands, both in CPU registers and in memory, follow the same rules used for fixed point operands handled in the standard MODCOMP II. That is, the most significant word of the operands is stored in the lower memory location or lower general purpose register number.



#### Overflow

Floating point overflow/underflow occurs if the resultant exponent of a floating point operation cannot be expressed within the range of the nine bit binary exponent field of the floating point format.

A trap occurs at interrupt level 5 if floating point overflow/underflow is detected. See Sections II and IV for a detailed explanation of traps.

	FAR	FLC	DATING	POINT	ADD	REG.	ТО	REG.			
(	)	3	4	7	8		11	12		15	15 us
	3		0			Ra			Rb		(Ph) (Ph)(1) (Pa) (Pa)(1) (Pa)
											(Rb),(RbVl)+(Ra),(RaVl)→Ra,RaVl

The contents of registers Rb and RbVl (with register Rb containing the more significant half and register RbVl containing the less significant half of a single precision floating point operand) are algebraically added to the contents of registers Ra and RaVl (with register Ra containing the more significant half and register RaVl containing the less significant half of a single precision floating point operand). The sum replaces the contents of registers Ra and RaVl. Ra and Rb must specify even-numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl

FSR FLOATING POINT SUBTRACT REG FROM REG

					15 us
i	3	1	Ra	Rb	
١					(Ra),(RaVl)-(Rb),(RbVl)+Ra,RaVl

The contents of registers Rb and RbVl (with register Rb containing the more significant half and register RbVl containing the less significant half of a single precision floating point operand) are algebraically subtracted from the contents of registers Ra and RaVl (with register Ra containing the more significant half and register RaVl containing the less significant half of a single precision floating point operand). The result is stored in registers Ra and RaVl. Ra and Rb must specify even numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl

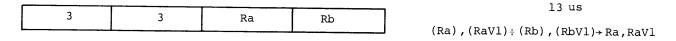
FMR FLOATING POINT MULTIPLY REG BY REG

				. 12.5 us
3	2	Ra	Rb	
			145	(Rb),(RbVl)x(Ra),(RaVl)→Ra,RaVl

The contents of registers Rb and RbVl (with register Rb containing the more significant half and register RbVl containing the less significant half of a single precision floating point multiplicand) are multiplied by the contents of registers Ra and RaVl (with register Ra containing the more significant half and register RaVl containing the less significant half of a single precision floating point multiplier). The product is stored in registers Ra and RaVl. Ra and Rb must specify even numbered registers. A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl

FDR FLOATING POINT DIVIDE REG BY REG



The contents of registers Rb and RbVl (with register Rb containing the more significant half and register RbVl containing the less significant half of a single precision floating point divisor) are divided into the contents of registers Ra and RaVl (with register Ra containing the more significant half and register RaVl

Floating Point Instructions

containing the less significant half of a single precision floating point dividend). The quotient replaces the contents of registers Ra and RaVl. Ra and Rb must specify even-numbered registers. A floating point overflow will occur if the divisor is equal to zero or if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl

FARD FLOATING	POINT	ADD	REG	то	REG	DOUBLE
---------------	-------	-----	-----	----	-----	--------

•					
	3	4	Ra	Rb	20.5 us

(Rb), (RbV1), (RbV2)+(Ra), (RaV1), (RaV2)+Ra, RaV1, RaV2

The contents of registers Rb,RbV1, and RbV2 (with these registers arranged in significance as described in Table 3-2 under floating point double precision operand formats) are algebraically added to the contents of registers Ra,RaV1, and RaV2 (with these registers arranged in significance as described in Table 3-2 under floating point double precision operand formats). The sum replaces the contents of registers Ra,RaV1, and RaV2. Ra and Rb must specify general purpose register four ( $^4_{16}$ ), eight ( $^8_{16}$ ), or  $^6_{16}$ . A floating point overflow will occur if the resultant exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl, RaV2

### FSRD FLOATING POINT SUBTRACT REG FROM REG DOUBLE

				1
3	5	Ra	Rb	20.5 us

(Ra), (RaV1), (RaV2)-(Rb), (RbV1), (RbV2)+Ra, RaV1, RaV2

The contents of registers Rb,RbVl,RbV2 are algebraically subtracted from the contents of registers Ra,RaVl,RaV2. The result is stored in registers Ra,RaVl,RaV2. Ra and Rb must specify general purpose register four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ . Floating point overflow may occur as described previously.

Affected: Ra, RaVl, RaV2

## FMRD FLOATING POINT MULTIPLY REG BY REG DOUBLE

3 6 Ra Rb

16 us

(Rb), (RbV1), (RbV2) x (Ra), (RaV1, (RaV2)  $\rightarrow$  Ra, RaV1, RaV2

The contents of registers Rb,RbVl,RbV2 containing a double precision floating point multiplicand are multiplied by the contents of Ra,RaVl,RaV2 which hold the double precision floating point multiplier. The product is stored in registers Ra,RaVl, RaV2. Ra and Rb must specify general purpose registers four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ . Floating point overflow may occur as described previously.

Affected: Ra,RaVl,RaV2

### FDRD FLOATING POINT DIVIDE REG BY REG DOUBLE

				3.0 =
3	7	Ra	Rb	16.5 us

(Ra), (RaV1), (RaV2); (Rb), (RbV1), (RbV2)  $\rightarrow Ra$ , RaV1, RaV2

The contents of registers Rb,RbVl,RbV2 containing a double precision floating point divisor are divided into the contents of registers Ra,RaVl,RaV2 which hold the double precision floating point dividend. The quotient replaces the contents of registers Ra,RaVl,RaV2. Ra and Rb must specify general purpose registers four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ . Floating Point overflow may occur as described previously.

Affected: Ra, RaV1, RaV2

# FAM FLOATING POINT ADD MEMORY TO REGISTER

3	8	Ra	Rx
	ADDRES	S WORD	

17.5 us

(EA), (EA+1)+(Ra),  $(RaV1)\rightarrow Ra$ , RaV1

The contents of the effective memory location and the effective memory location plus one (with (EA) containing the less significant half of a single precision floating point operand and (EA+1) containing the more significant half of a single precision floating point operand) are algebraically added to the contents of registers Ra and RaV1 (with register Ra containing the more significant half and register RaV1 containing the less significant half of a single precision floating point operand). The sum replaces the contents of Ra and RaV1. Ra must specify an even-numbered register. A floating point overflow will occur if the result exponent cannot be expressed within the range of the nine (9) bit binary exponent field of the floating point format.

Affected: Ra, RaVl

Floating Point
Instructions

### FSM FLOATING POINT SUBTRACT MEMORY FROM REGISTER

3	9	Ra	Rx
	ADDRESS	WORD	

17.5 us

(Ra), (RaV1)-(EA), (EA+1)+Ra, RaV1

The contents of the EA and EA+1 are algebraically subtracted from the contents of Ra, RaV1. The remainder replaces the contents of Ra, RaV1. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra, RaVl

### **FMM**

FLOATING POINT MULTIPLY MEMORY BY REGISTER

3	A	Ra	Rx
	ADDRESS WORD	(EA)	

14.5 us

(EA), A+1x(Ra), (RaV1) $\rightarrow$ Ra, RaV1

The contents of the EA and EA+1 containing a single precision fixed point multiplicand are multiplied by the contents of Ra,RaVl containing a single precision floating point multiplier. The product replaces the contents of Ra,RaVl. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra, RaVl

### **FDM**

FLOATING POINT DIVIDE MEMORY INTO REGISTER

3	В	Ra	Rx		
ADDRESS WORD					

15.5

(Ra), (RaV1): (EA), (EA+1)  $\rightarrow$  Ra, RaV1

The contents of Ra and RaVl containing a single precision floating point dividend are divided by the contents of EA and EA+l containing a single precision floating point divisor. The quotient replaces the contents of Ra,RaVl. Ra must specify an even numbered register.

Floating point overflow may occur as described previously.

Affected: Ra, RaVl

Floating Point Instructions

### FAMD FLOATING POINT ADD MEMORY TO REGISTER DOUBLE

3	С	Ra	Rx		
ADDRESS WORD					

22.5 us

(EA), (EA+1), (EA+2)+(Ra), (RaV1), (RaV2)+Ra, RaV1, RaV2

The contents of EA,EA+1,EA+2 containing a double precision floating point augend are algebraically added to the contents of Ra,RaV1,RaV2 containing a double precision floating point addend. The sum replaces the contents of Ra,RaV1,RaV2. Ra must specify general purpose register four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ .

Floating point overflow may occur as described previously.

Affected: Ra, RaV1, RaV2

# FSMD FLOATING POINT SUBTRACT MEMORY FROM REG DOUBLE

3	D	Ra	Rx		
ADDRESS WORD					

22.5 us

(Ra), (RaV1), (RaV2) - (EA), (EA+1), (EA+2)  $\rightarrow$  Ra, RaV1, RaV2

The contents of EA,EA+1 and EA+2 contain a double precision floating point subtrahend which is subtracted from Ra,RaVl,RaV2 containing a double precision floating point minuend. The remainder replaces the contents of Ra,RaVl,RaV2. Ra must specify general purpose register four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ .

Floating point overflow may occur as described previously.

Affected: Ra, RaVl, RaV2

# FMMD FLOATING POINT MULTIPLY MEMORY BY REGISTER DOUBLE

3	Е	Ra	Rx			
	ADDRESS WORD					

18 us

(EA), (EA+1), (EA+2)x (Ra), (RaV1), (RaV2)+Ra, RaV1, RaV2

The double precision floating point multiplicand contained in EA,EA+l and EA+2 is multiplied by the double precision floating point multiplier contained in registers Ra,RaVl and RaV2. The product replaces the contents of Ra,RaVl and RaV2. Ra must specify general purpose register four  $(4_{16})$ , eight  $(8_{16})$ , or  $(C_{16})$ .

Floating point overflow may occur as described previously.

Affected: Ra, RaV1, RaV2

Floating Point Instructions

FDMD FLOATING POINT DIVIDE MEMORY INTO REG DOUBLE

3	F	Ra	Rx		
ADDRESS WORD					

19 us

(Ra), (RaV1), (RaV2) ÷ (EA), (EA+1), (EA+2)→Ra, RaV1, RaV2

The double precision floating point dividend contained in Ra,RaVl and RaV2 is divided by the double precision floating point divisor contained in EA,EA+l and EA+2. The quotient replaces the contents of Ra,RaVl and RaV2. Register Ra must specify general purpose register four  $(4_{16})$ , eight  $(8_{16})$  or  $(C_{16})$ .

Floating point overflow may occur as described previously.

Affected: Ra, RaV1, RaV2

## SHIFT INSTRUCTIONS

The ten instructions in this group are used to reposition bits left or right within a single or a pair of adjacent registers. All combinations of arithmetic and logical, left and right, and single register and double register shift operations are provided. In addition, a left rotate instruction is included.

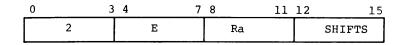
The execution of each shift instruction may shift the operand zero to 15 bit positions as defined by the binary coded shift field (bits 12-15) in each instruction except LRS.

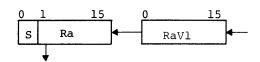
In all double register shift operations, the register specified by the instruction word must be the even register of an even-odd register pair consisting of two adjacent registers in the general register file (Ra (even) and RaVl (odd)). In doubleword arithmetic shifts, the more significant half of the operand is assumed to be in the even register and the less significant half in the odd register.

LAD

SHIFT LEFT ARITHMETIC DOUBLE

1.87 us + 267(N-1) ns





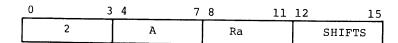
The contents of register Ra and register RaVl are shifted left zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift. Zeros are shifted into the LSB position of RaVl and the MSB of RaVl is shifted into the least significant bit position of Ra with each shift step. The next to MSB of Ra (bit position 1) is shifted out of the register and is lost. Ra must specify an even general register.

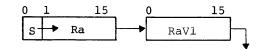
Affected: Ra, RaVl, Overflow

RAD

SHIFT RIGHT ARITHMETIC DOUBLE

1.87 us + 267(N-1) ns





The contents of register Ra and register Ra+l are shifted right zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift.\* The LSB(s) of Ra are shifted to the MSB position of RaVl and the LSB(s) of RaVl are shifted out of the register and are lost. Ra must specify an even general register.

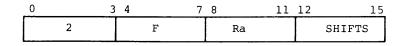
Affected: Ra, RaVl

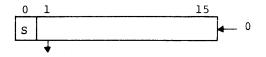
<sup>\*</sup>The sign bit is propogated right the number of places specified by the shift count.

LAS

SHIFT LEFT ARITHMETIC SINGLE

1.6 us + 267(n-1) ns



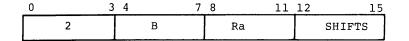


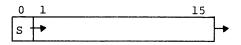
The contents of register Ra are shifted left zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift. The next to MSB of Ra (bit position 1) is shifted out of the register and is lost. Zeros are shifted into the LSB position(s) of Ra. Affected: Ra, (Overflow)

RAS

SHIFT RIGHT ARITHMETIC SINGLE

1.6 us + 267(N-1) ns





The contents of register Ra are shifted right zero to 15 bit position(s) as specified by the shift count control field. The sign bit of Ra does not change either during or after the shift\* The least significant bit(s) of Ra are shifted out of the register and are lost.

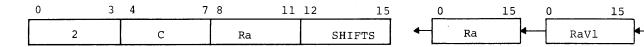
Affected: Ra

\*See previous page.

LLD

SHIFT LEFT LOGICAL DOUBLE

1.87 us + 267(N-1) ns

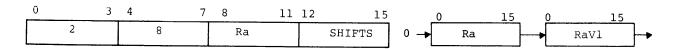


The contents of register Ra and register Ra+1 are shifted left zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the least significant bit position(s) of RaV1 and the most significant bit(s) of RaV1 are shifted into the least significant bit position(s) of Ra. The most significant bit(s) of Ra are shifted out of Ra and are lost. Ra must specify an even general register. Affected: Ra, RaV1

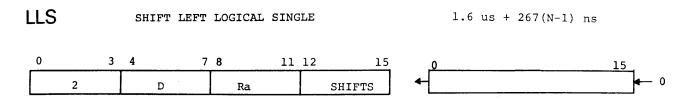
RLD

SHIFT RIGHT LOGICAL DOUBLE

1.87 us + 267(N-1) ns

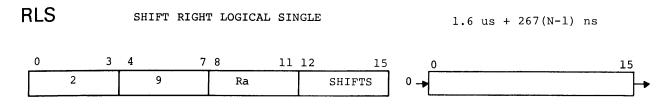


The contents of register Ra and register RaVl are shifted right zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the most significant bit position(s) of Ra and the least significant bit position(s) of Ra are shifted into the most significant bit position(s) of RaVl. The least significant bit(s) of RaVl are shifted out of RaVl and are lost. Ra must specify an even general register. Affected: Ra, RaVl



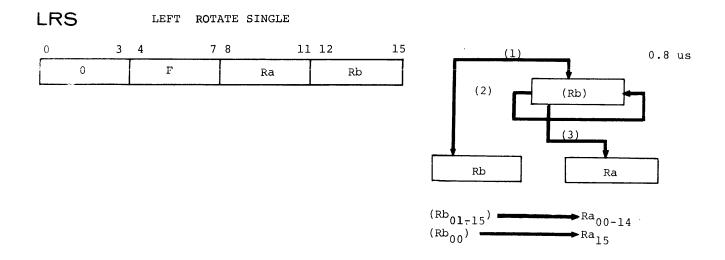
The contents of register Ra are shifted left zero to 15 bit position(s) as specified by the shift count control field. Zeros are shifted into the least significant bit position(s) and the most significant bit position(s) are shifted out of Ra<sub>0</sub> and are lost.

Affected: Ra



The contents of register Ra are shifted right zero to 15 bit positions as specified by the shift count control field. Zeros are shifted into the most significant bit position(s) and the least significant bit position(s) are shifted out of Ra<sub>15</sub> and are lost.

Affected: Ra



The contents of register Ra are replaced by the contents of register Rb shifted left one bit position with the most significant bit of Rb rotated into bit position 15 of register Ra. The contents of Rb are unaffected.

Affected: Ra

## BIT MANIPULATION INSTRUCTIONS

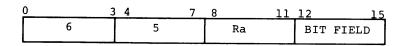
The bit manipulation instruction group includes the Load, Add, Subtract, Zero, OR, Exclusive OR, Test, and Compare instructions. In all instructions except Zero and Test, one operand is a bit literal of value one and the other operand is the 16-bit contents of the effective memory location or designated register. For example the Add Bit In Memory instruction causes a bit of value one to be added to the contents of the effective memory location. Carry is propagated through to the left through the sign bit.

The position of the bit literal is designated by the four bit, binary coded Bit Field in each instruction. Any bit in the word can be designated. The value of the Bit Field (n) specifies a 16-bit binary number of value  $+2^{15-n}$ .

Since the value of the bit literal is always one in the OR instruction, execution of this instruction causes the designated bit in memory or a general register to be set to one. For the same reason, execution of the Exclusive OR instruction causes the designated bit to be complemented (inverted).

LBR LOAD BIT IN REGISTER

0.8 us

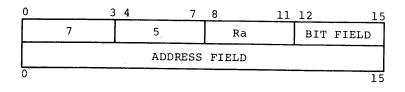


$$\begin{array}{ccc}
1 & -Ra_n \\
0 & -Ra_{0-(n-1)} \\
0 & -Ra_{(n+1)-1}
\end{array}$$

A one is stored in register Ra in bit position n, where n is specified by the contents of the Bit Field. Zeros are stored in all other bit positions in register Ra. Affected: Ra

LBRB (Let) Load bit in register and branch unconditionally

1.6 us



$$0's \rightarrow Ra_{0-(n-1)}$$

$$0's \rightarrow Ra_{(n+1)-15}$$

1 → Ra<sub>n</sub>

A one is stored in register Ra in bit position n, where n is specified by the contents of the Bit Field. Zeros are stored in all other bit positions in register Ra.

A branch is then executed unconditionally to the location specified by the contents of the second instruction word.

Affected: Ra

ABMM ADD BIT IN MEMORY

0 3 4 7 8 11 12 13 15

8 0 BIT FIELD I Rxx

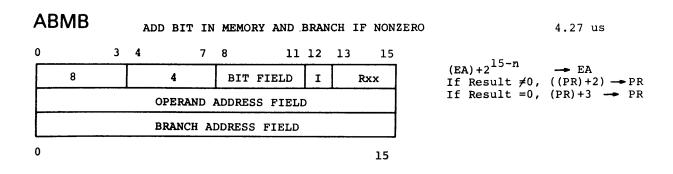
ADDRESS FIELD

3.47 us

(EA)+2<sup>15-n</sup> → EA

The contents of the effective memory location are incremented by one in the bit position (n) designated in the Bit Field of the first instruction word. The result is stored in the effective memory location. Overflow will occur if the result is greater than  $2^{15}$ -1.

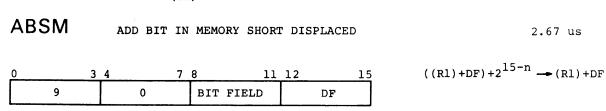
Affected: (Overflow), (EA)



The contents of the effective memory location are incremented by one in the bit position (n) designated in the Bit Field of the first instruction word. The result is stored in the effective memory location.

If the result is unequal to zero, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed. Overflow will occur if the result is greater than  $2^{15}$ -1.

Affected: Overflow, (EA)



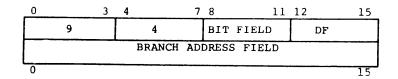
The contents of the effective memory location specified by the displacement field DF added to the contents of register Rl are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than  $2^{15}$ -1.

Affected: Overflow, (EA)

**ABSB** 

ADD BIT IN MEMORY SHORT DISPLACED AND BRANCH IF NONZERO

3.74 us



$$((R1) + DF) + 2^{15-n} \longrightarrow (R1) + DF$$
  
If Result  $\neq 0$ ,  $((PR) + 1) \longrightarrow PR$   
If Result =0,  $(PR) + 2 \longrightarrow PR$ 

The contents of the effective memory location specified by the displacement field DF added to the contents of register Rl are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than  $2^{15}-1$ .

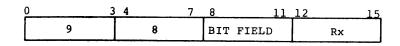
If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.

Affected: Overflow (EA)

**ABXM** 

ADD BIT IN MEMORY SHORT INDEXED

2.67 us



$$((Rx))+2^{15-n} \rightarrow (Rx)$$

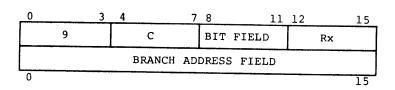
The contents of the effective memory location specified by the contents of register Rx are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than  $2^{15}-1$ .

Affected: Overflow , (EA)

**ABXB** 

ADD BIT IN MEMORY SHORT INDEXED AND BRANCH IF NONZERO

3.74 us



$$((Rx))+2^{15-n} \longrightarrow (Rx)$$
If Result  $\neq 0$   $((PR)+1) \longrightarrow PR$ 
If Result =0  $(PR)+2 \longrightarrow PR$ 

The contents of the effective memory location specified by the contents of register Rx are incremented by one in the bit position (n) designated by the Bit Field. The result is stored in the effective memory location. Overflow will occur if the result is greater than  $2^{15}$ -1.

If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.

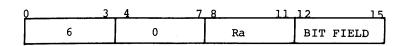
Affected: Overflow, (EA)

Bit Manipulation
Instructions

**ABR** 

ADD BIT IN REGISTER

0.8 us



 $(Ra)+2^{15-n} \rightarrow Ra$ 

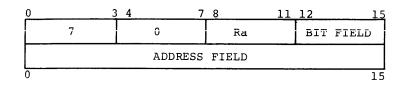
The contents of register Ra are incremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Over-flow will occur if the result is greater than  $2^{15}$ -1.

Affected: Ra, Overflow

**ABRB** 

ADD BIT IN REGISTER AND BRANCH IF NONZERO

1.6 us



 $(Ra)+2^{15-n} \longrightarrow Ra$ If Result  $\neq 0$ , EA  $\longrightarrow$  PR
If Result =0,  $(PR)+2 \longrightarrow PR$ 

The contents of register Ra are incremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Over-flow will occur if the result is greater than  $2^{15}$ -1.

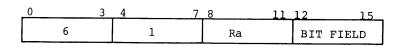
If the result is unequal to zero, a branch is executed to the effective memory location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra Overflow

**SBR** 

SUBTRACT BIT IN REGISTER

0.8 us



 $(Ra)-2^{15-n} \longrightarrow Ra$ 

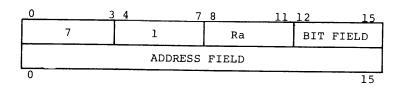
The contents of register Ra are decremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Overflow will occur if the result is less than  $-2^{15}$ .

Affected: Ra, Overflow

**SBRB** 

SUBTRACT BIT IN REGISTER AND BRANCH IF NONZERO

1.6 us



 $(Ra)-2^{15-n} \longrightarrow Ra$ If Result  $\neq 0$ , EA  $\longrightarrow$  PR
If Result = 0,  $(PR)+2 \longrightarrow PR$ 

The contents of register Ra are decremented by one in the bit position (n) designated in the Bit Field of the instruction word. The result is stored in register Ra. Over-flow will occur if the result is less than  $-2^{15}$ .

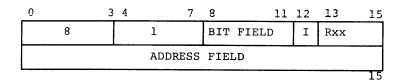
If the result is unequal to zero, a branch is executed to the effective memory location. If the result equals zero, the next instruction in sequence is executed.

Affected: Ra, (Overflow)

**ZBMM** 

ZERO BIT IN MEMORY

3.47 us



 $0 \rightarrow (EA)_n$ 

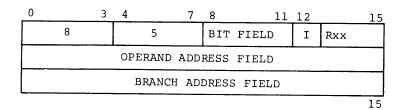
The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is cleared to zero. The other bits contained in the word are unaffected.

Affected: (EA)

**ZBMB** 

ZERO BIT IN MEMORY AND BRANCH IF NONZERO

4.8 us



 $0 \longrightarrow (EA)_{n}$ If Result  $\neq 0$ ,  $((PR)+2) \longrightarrow PR$ If Result =0,  $(PR)+3 \longrightarrow PR$ 

The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is cleared to zero. The other bits contained in the word are not affected. If the resulting word is unequal to zero, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

**ZBSM** 

ZERO BIT IN MEMORY SHORT DISPLACED

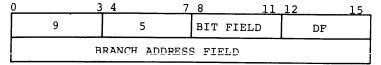
2.67 us

0 3	4 7	8	11	12	15
9	1	віт	FIELD	DF	

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register Rl is cleared to zero. The other bits contained in the word are unaffected.

Affected: (EA)

ZBSB zero bit in memory short displaced and branch if nonzero 4.27 us



$$0 \longrightarrow ((R1)+DF)_{n}$$
If Result  $\neq 0$ ,  $(PR)+1) \longrightarrow PR$ 
If Result  $=0$ ,  $(PR)+2 \longrightarrow PR$ 

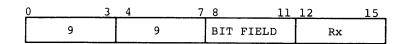
 $0 \rightarrow ((R1) + DF)_n$ 

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register Rl is cleared to zero. The other bits contained in the word are unaffected. If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed. Affected: (EA)

**ZBXM** 

ZERO BIT IN MEMORY SHORT INDEXED

2.93 us





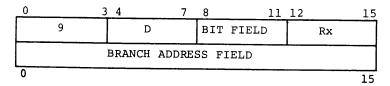
The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is cleared to zero. The other bits contained in the word are unaffected.

Affected: (EA)

**ZBXB** 

ZERO BIT IN MEMORY SHORT INDEXED AND BRANCH IF NONZERO

4.27 us



```
\begin{array}{l} 0 \longrightarrow ((Rx))_n \\ \text{If Result } \neq 0, \ (PR) + 1) \longrightarrow PR \\ \text{If Result } = 0, \ (PR) + 2 \longrightarrow PR \end{array}
```

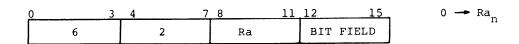
The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is cleared to zero. The other bits contained in the word are unaffected. If the resulting word is unequal to zero, a branch is executed to the location specified by the second instruction word. The branch address may only be generated by the direct address mode without indexing. If the result equals zero, the next instruction in sequence is executed.

Affected: (EA)

**ZBR** 

ZERO BIT IN REGISTER

0.8 us

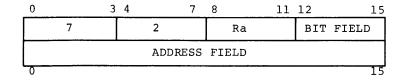


The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is cleared to zero. The other bits in register Ra are not affected. Affected:  $Ra_n$ 

ZBRB 5 10

ZERO BIT IN REGISTER AND BRANCH IF NONZERO

1.6 us



$$\begin{array}{l} 0 \longrightarrow \text{Ra}_n \\ \text{If Result } \neq 0 \text{, EA} \longrightarrow \text{PR} \\ \text{If Result } = 0 \text{, } (\text{PR}) + 2 \longrightarrow \text{PR} \end{array}$$

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is cleared to zero. The other bits in register Ra are not affected.

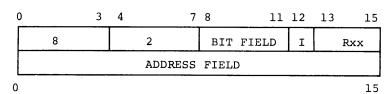
If the contents of Ra are not equal to zero, a branch is executed to the effective memory location. If the contents of Ra equals zero, the next instruction in sequence is executed.

Affected: Ran

**OBMM** 

OR BIT IN MEMORY

3.47 us





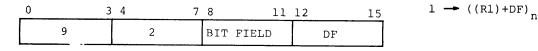
The bit contained in the position in the effective memory location designated by the contents of the Bit Field (n) is set to one. The other bits contained in the word are unaffected.

Affected: (EA)

OBSM

OR BIT IN MEMORY SHORT DISPLACED

2.67 us

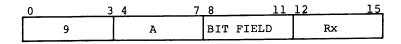


The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field DF added to the contents of register Rl is set to one. The other bits contained in the word are unaffected.

Affected: (EA)

# OBXM (5) OR BIT IN MEMORY SHORT INDEXED

2.94 us



 $1 \rightarrow ((Rx))_n$ 

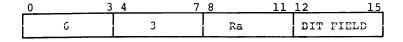
The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is set to one. The other bits contained in the word are unaffected.

Affected: (EA)

**OBR** 

OR BIT IN REGISTER

0.8 us



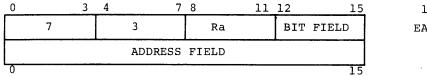
1 → Ra<sub>n</sub>

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is set to one. The other bits in register Ra are not affected. Affected: Ran

# **OBRB**

OR BIT IN REGISTER AND BRANCH UNCONDITIONALLY

1.6 us





The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is set to one. The other bits in register Ra are not affected.

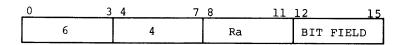
An unconditional branch is then executed to the effective memory location.

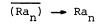
Affected: Ran

**XBR** 

EXCLUSIVE OR BIT IN REGISTER

0.8 us





The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is complemented. The other bits in register Ra are not affected. Affected:  $Ra_n$ 

**XBRB** 

EXCLUSIVE OR BIT IN REGISTER AND BRANCH IF NONZERO

1.6 us



$$(Ra_n) \longrightarrow Ra_n$$
If Result  $\neq 0$ , EA  $\longrightarrow$  PR
If Result =0,  $(PR)+2 \longrightarrow PR$ 

The bit contained in the position in register Ra designated by the contents of the Bit Field (n) is complemented. The other bits in register Ra are not affected.

If the contents of Ra are unequal to zero, a branch is executed to the effective memory location. If the contents of Ra equal zero, the next instruction in sequence is executed.

Affected: Ran

**TBMB** 

TEST BIT IN MEMORY AND BRANCH IF ONE

0 3 4 7 8 11 12 13 15

8 6 BIT FIELD I RXX

OPERAND ADDRESS FIELD

BRANCH ADDRESS FIELD

0 15

3.47 us

If Effective Bit =1,
 ((PR)+2) → PR
If Effective Bit =0,
 (PR)+3 → PR

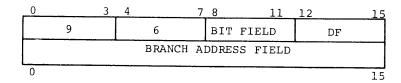
The bit contained in the position in the effective memory location designated by the contents of the Bit Field(n)is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the third instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed.

Affected: None

**TBSB** 

TEST BIT IN MEMORY SHORT DISPLACED AND BRANCH IF ONE

3.2 us



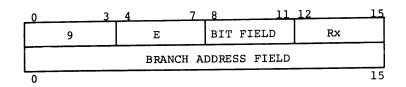
If Effective Bit =1, ((PR)+1) → PR If Effective Bit =0, (PR)+2 → PR

The bit n, designated by the Bit Field, contained in the memory location specified by the displacement field added to the contents of register Rl is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the second instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed. Affected: None

**TBXB** 

TEST BIT IN MEMORY SHORT INDEXED AND BRANCH IF ONE

3.2 us



If Effective Bit =1, ((PR)+1) → PR If Effective Bit =0, (PR)+2→ PR

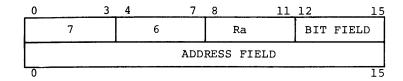
The bit n, designated by the Bit Field, contained in the memory location specified by the contents of register Rx is tested. If the bit is equal to one, a branch is executed to the location specified by the contents of the second instruction word. Only the direct address mode without indexing is performed. If the tested bit is equal to zero, the next instruction in sequence is executed.

Affected: None

**TBRB** 

TEST BIT IN REGISTER AND BRANCH IF ONE

1.6 us



If 
$$(Ra_n)=1$$
, EA  $\longrightarrow$  PR  
If  $(Ra_n)=0$ ,  $(PR)+2 \longrightarrow$  PR

The bit contained in the position in Ra designated by the contents of the Bit Field (n) is tested. If the bit is equal to one, a branch is executed to the effective memory location. If the bit equals zero, the next instruction in sequence is executed.

Affected: None

**CBMB** 

0

COMPARE BIT AND MEMORY

C		3	4	7	8	]	L1	12	13	15
	8		7		BIT	FIELI		Ι	R	xx
			OPERAND	ADDI	RESS I	FIELD				
			BRANCH	ADDRE	ESS F	ELD	(E	A)=2	15-n	
		•	BRANCH			ELD	(E	A) >2	15-n	

If 
$$2^{15-n} - (EA) = 0$$
  $((PR) + 2) \longrightarrow PR$   
If  $2^{15-n} - (EA) < 0$   $((PR) + 3) \longrightarrow PR$   
If  $2^{15-n} - (EA) > 0$   $(PR) + 4 \longrightarrow PR$ 

The contents of the effective memory location are algebraically subtracted from the value  $+2^{15-n}$ , where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the third instruction word. If the result is negative, a branch is executed to the location specified by the fourth instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.

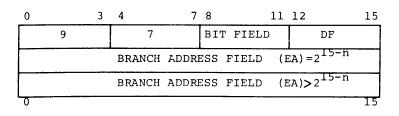
15

Affected: None

**CBSB** 

COMPARE BIT AND MEMORY SHORT DISPLACED

3.7 us



```
If 2^{15-n} - (EA) = 0
If 2^{15-n} - (EA) < 0
If 2^{15-n} - (EA) > 0
                                                         ((PR)+3) \rightarrow PR
```

The contents of the memory location specified by the displacement field DF added to the contents of register Rl are algebraically subtracted from the value 2<sup>15-n</sup>, where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word.

Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.

Affected: None

# **CBXB**

COMPARE BIT AND MEMORY SHORT INDEXED

3.7 us

If 
$$2^{15-n}$$
 -(EA)=0 ((PR)+2)  $\longrightarrow$  PR  
If  $2^{15-n}$  -(EA)<0 ((PR)+3)  $\longrightarrow$  PR  
If  $2^{15-n}$  -(EA)>0 (PR)+4  $\longrightarrow$  PR

The contents of the memory location specified by the contents of register Rx are algebraically subtracted from the value  $2^{15-n}$ , where n is designated by the Bit Field in the first instruction word. If the result equals zero, a branch is executed to the location specified by the second instruction word. If the result is negative, a branch is executed to the location specified by the third instruction word. Only the direct addressing mode without indexing is permitted for the branch operation. If the result is greater than zero, the next instruction in sequence is executed. The contents of the memory location are not altered.

Affected: None

**GMR** 

GENERATE MASK IN REGISTER

0.8 us

0	3	4 7	8 11	12 15
6		7	Ra	BIT FIELD

1's 
$$\rightarrow Ra_{0-n}$$
0's  $\rightarrow Ra_{n+1-15}$ 

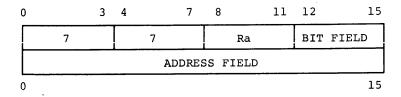
Ones are stored in register Ra in bit position  $Ra_0$  through  $Ra_n$ , where n is specified by the contents of the bit field. Zeroes are stored in register Ra in bit positions  $Ra_{n+1}$  through  $Ra_{15}$ . Overflow will result and  $PR_0$  will be set if n=0.

Affected: Ra, (Overflow)

**GMRB** 

GENERATE MASK IN REGISTER AND BRANCH UNCONDITIONALLY

1.6 u



1's 
$$\rightarrow$$
 Ra<sub>0-n</sub>  
0's  $\rightarrow$  Ra<sub>n+1-15</sub>  
EA  $\rightarrow$  PR

Ones are stored in register Ra in bit positions  $\mathrm{Ra}_0$  through  $\mathrm{Ra}_n$ , where n is specified by the contents of the bit field. Zeroes are stored in register Ra in bit positions  $\mathrm{Ra}_{n+1}$  through  $\mathrm{Ra}_{15}$ . Overflow will result and be set if n=0.

A branch is then executed unconditionally to the location specified by the contents of the second instruction word.

Affected: Ra, Overflow

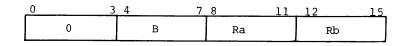
# BYTE MANIPULATION INSTRUCTIONS

These instructions enable bytes to be moved and interchanged in the general register file. All of these instructions contain two register addresses Ra and Rb. By making Ra equal to Rb, either byte or both bytes can be moved within one register. By making Ra unequal to Rb bytes can be moved from register to register. The move instructions cause one byte to be cleared to zero in the destination register, whether Ra=Rb or Ra≠Rb.

## **MUR**

MOVE UPPER BYTE REGISTER TO REGISTER

0.8 us



$$(Rb_{0-7}) \longrightarrow Ra_{0-7}$$
 $0 \longrightarrow Ra_{8-15}$ 

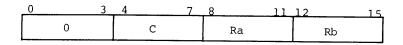
The more significant byte stored in register Rb is transferred to the more significant byte position of register Ra. Zeros are transferred to the less significant byte position in register Ra. If Ra=Rb, the instruction becomes a "Clear Lower Byte in Register" instruction.

Affected: Ra

# **MLR**

MOVE LOWER BYTE REGISTER TO REGISTER

0.8 us



$$\begin{array}{c} (Rb_{8-15}) & \longrightarrow Ra_{8-15} \\ 0 & \longrightarrow Ra_{0-7} \end{array}$$

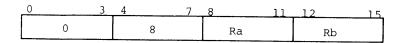
The less significant byte stored in register Rb is transferred to the less significant byte position of register Ra. Zeros are transferred to the more significant byte position in register Ra. If Ra=Rb, this instruction becomes a "Clear Upper Byte in Register" instruction.

Affected: Ra

# **MBR**

MOVE BYTE RIGHT REGISTER TO REGISTER

0.8 us



$$(Rb_{0-7}) \longrightarrow Ra_{8-15}$$
 $0 \longrightarrow Ra_{0-7}$ 

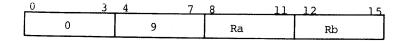
The more significant byte stored in register Rb is transferred to the less significant byte position of register Ra. Zeros are transferred to the more significant byte position in register Ra. If Ra=Rb, this instruction becomees a fast "Logical Right Shift Eight Bits" instruction.

Affected: Ra

MBL

MOVE BYTE LEFT REGISTER TO REGISTER

0.8 us



$$(Rb_{8-15}) \longrightarrow Ra_{0-7}$$
 $0 \longrightarrow Ra_{8-15}$ 

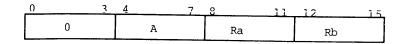
The less significant byte stored in register Rb is transferred to the more significant byte position of register Ra. Zeros are transferred to the less significant byte position in register Ra. If Ra=Rb, this instruction becomes a fast "Logical Left Shift Eight Bits" instruction.

Affected: Ra

**IBR** 

INTERCHANGE BYTES REGISTER TO REGISTER

0.8 us



$$(Rb_{0-7}) \longrightarrow Ra_{8-15}$$

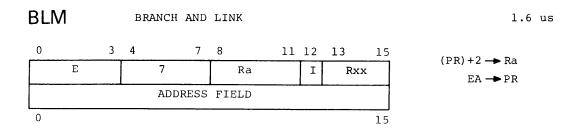
$$(Rb_{8-15}) \longrightarrow Ra_{0-7}$$

The less significant byte stored in register Rb is transferred to the more significant byte position in register Ra, and the more significant byte stored in register Rb is transferred to the less significant byte position in register Ra. If Ra = Rb, this instruction becomes a "Rotate Eight Bits" instruction.

Affected: Ra

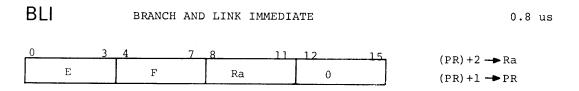
# **UNCONDITIONAL BRANCH INSTRUCTIONS**

This group includes the Branch and the Branch and Link instructions. All are unconditional branch instructions. Each time a branch is executed all 16 bits of the Program Register, are replaced.



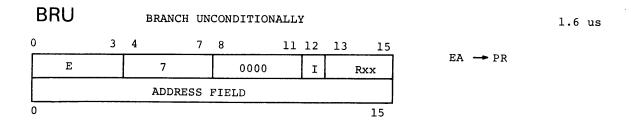
The 16-bit contents of the Program Register replace the contents of register Ra and then the effective memory address replaces the contents of the Program Register.

Affected: Ra



The 16-bit contents of the Program Register replace the contents of register Ra and then the next instruction in sequence is executed.

Affected: Ra



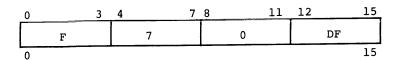
The 16 bits of the effective memory address replace the contents of the Program Register.

Affected: None

HOP

BRANCH SHORT DISPLACED

1.07 us



The contents of the displacement field DF are added to the contents of the Program Register. The result is then stored in the Program Register. Therefore a branch is executed which has a range of from zero to +15 locations with respect to the current program location.

Affected: None

BRX

BRANCH SHORT INDEXED

1.07 us

0 3	4 7	8 11	12 15
F	F	0	Rx

(Rx) - PR

 $(PR) + DF \rightarrow PR$ 

The 16 bit contents of register Rx replace the contents of the Program Register. Affected: None

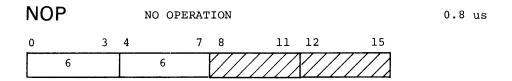
#### **CONTROL INSTRUCTIONS**

This group includes Halt, No Operation, Set Protect Register, Set Lower Protect Register and set Upper Protect Register.



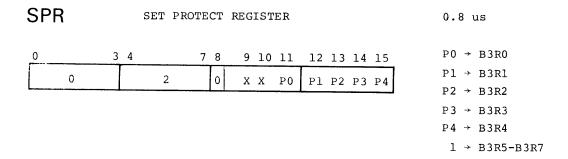
Program Execution is halted until the program is manually restarted. The halt occurs after the Program Register is advanced and the next instruction is transferred to the instruction register.

Affected: None



The execution of this instruction does not modify the contents of any general register or memory location.

Affected: None



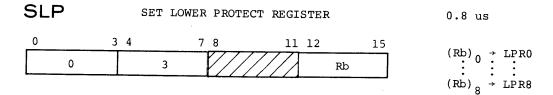
The five least significant bits of the Instruction Word (IRl1-IR15) are transferred directly to the Protect Boundary Register #3 (Bits 0 thru 4). Ones are loaded into bits five thru seven of this register therby yielding a 2K word protect granularity.

Affected: Protect Boundary Register #3.

SGP SET GLOBAL PROTECT REGISTER	0.8 us
0 3 4 7 8 9 10 11 12 13 14 15 0 2 1 X X X Rb	(Rb) 0 B3R0 (Rb) 1 B3R1 (Rb) 2 B3R2 (Rb) 3 B3R3 (Rb) 4 B3R4 (Rb) 5 B3R5 (Rb) 6 B3R6 (Rb) 7 B3R7

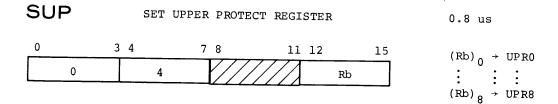
The eight most significant bits of the GPR specified by the Rb field are transferred directly to Protect Boundary Register #3 (Bits 0 thru 7) thereby yielding a 256 word protect granularity.

Affected: Protect Boundary Register #3.



The nine most significant bits of the GPR specified by the Rb field are transferred directly to Lower Protect Boundary Register, (Bits 0 thru 8) thereby yielding a 128 word protect granularity.

Affected: Lower Protect Boundary Register.



The nine most significant bits of the GPR specified by the Rb field are transferred directly to the Upper Protect Boundary Register, (Bits 0 thru 8) thereby yielding a 128 word protect granularity.

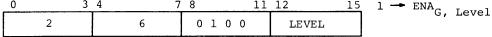
Affected: Upper Protect Boundary Register.

## INTERRUPT AND CALL INSTRUCTIONS

The interrupt instructions provide the capability for complete program manipulation of the states of all three latches present in each priority interrupt level. The Request Executive Service is the executive call instruction and the Request Multiprocessor Interrupt instruction enables each CPU in a multiprocessor configuration to produce an interrupt in the other cpu.

Each interrupt instruction contains a binary coded level field. This field permits each of the 16 (maximum) levels to be addressed and operated on individually.

SIE SET INTERRUPT ENABLE 1.33 us

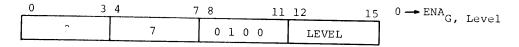


Enable the priority interrupt level specified by the Level selection field.

The PFS/AS, Memory Parity, Unimplemented Instruction, System Protect and Floating Point Overflow Trap interrupt levels are always enabled when present in the system. The Enable latch state of these levels cannot be altered by instruction execution.

Affected: None

RIE RESET INTERRUPT ENABLE 1.33 us



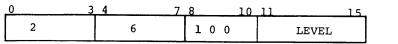
Disable the priority interrupt level specified by the Level selection field.

The PFS/AS, Memory Parity, Unimplemented Instruction, System Protect and Floating Point Overflow Trap interrupt levels are always enabled when present in the system. The Enable latch state of these levels cannot be altered by instruction execution.

SIR

SET INTERRUPT REQUEST

1.33 us



1  $\rightarrow$  REQ<sub>G</sub>, Level

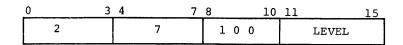
Set the Request latch of the priority interrupt level specified by the Level selection field. NOTE: Levels  $\rm C_{16}$  and  $\rm D_{16}$  should not be requested by the program.

Affected: None

RIR

RESET INTERRUPT REQUEST

1.33 us



 $0 \rightarrow \text{REQ}_{G, \text{Level}}$ 

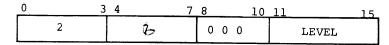
Reset the Request latch of the priority interrupt level specified by the Level selection field.

Affected: None

SIA

SET INTERRUPT ACTIVE

1.33 us



1 → ACT<sub>G</sub>, Level

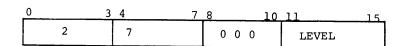
Activate the priority interrupt level specified by the Level selection field. NOTE: Level 0 (Power Fail/Auto Start) may not be set active by the program.

Affected: None

RIA

RESET INTERRUPT ACTIVE

1.33 us



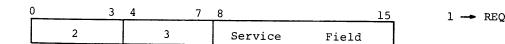
 $0 \rightarrow ACT_{G, Level}$ 

Deactivate the priority interrupt level specified by the Level selection field. Affected: None

**REX** 

REQUEST EXECUTIVE SERVICE

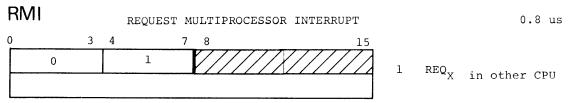
1.07 us



An interrupt request signal is sent to the Unimplemented Instruction Trap level. The executive service requested is defined by the contents of the Service Field. The program count is not advanced before the trap is generated. Therefore the stored PSW contains the address of the REX service.

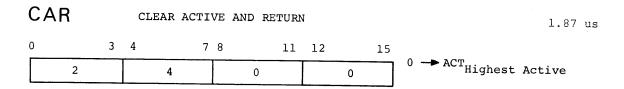
The Unimplemented Instruction Trap level will become active and the interrupt routine entered at the completion of execution of the REX instruction, provided that neither this level nor any higher level is already active. If this level or a higher level is active, execution of the REX cannot be completed. The machine must be manually cleared and restarted if this error condition occurs.

Affected: None



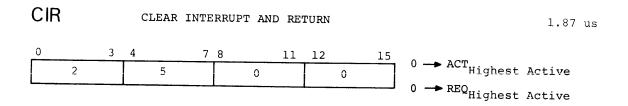
A pulse is generated by the executing CPU which requests the interprocessor communication interrupt in the other CPU.

Affected: None



The Active latch of the highest active interrupt level is cleared and the 16 bit contents of the memory location dedicated to the highest active level and transferred to the Program Register. If no interrupt is active when the CAR instruction is executed, the contents of location 0 are transferred to the Program Register. If an Interrupt is requesting when the CAR instruction is executed, (the interrupt) will not go in service until 1 instruction after the CAR is executed.

Affected: None



Both the Active and Request latches of the highest active interrupt level are cleared and the 16 bit contents of the memory location dedicated to the highest active level are transferred to the Program Register. If no interrupt is active when the CIR instruction is executed, the contents of location 0 are transferred to the Program Register.

Affected: None

## INPUT/OUTPUT INSTRUCTIONS

Two input instructions are provided to enable a data or status word to be transferred from any peripheral device to any general register. Two output instructions are provided to enable a data or command word to be transferred from any general register to any peripheral device. Some peripheral devices such as the disc transfer data only under control of the Direct Memory Processor. Therefore, only command and status words are transferred under program control to/from these devices.

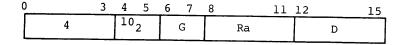
Up to 64 peripheral devices, consisting of four groups of 16 each, are addressable by each instruction. The group address is obtained from the two least significant bits of the operation code field. Therefore four operation codes and mnemonics are assigned to each instruction.

```
I/O GROUP A Consists of device addresses 00-0F
I/O GROUP B Consists of device addresses 10-1F
I/O GROUP C Consists of device addresses 20-2F
I/O GROUP D Consists of device addresses 30-3F
```

All instructions are executed in the fixed length of time contained in each instruction description.

ISA	ISA	(48)	Input	Status	From	I/O	Group	Α
ISB	ISB	(49)	Input	Status	From	I/O	Group	В
ISC	ISC	(4A)	Input	Status	From	1/0	Group	С
ISD	ISD	(4B)	Input	Status	From	1/0	Groun	ח

1.6 us



G, D  $\longrightarrow$  I/O Address Lines Device Status  $\longrightarrow$  Ra

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

Up to 16 bits of status are then transferred from the addressed device over the I/O bus to replace the contents of register Ra. Affected: Ra

IDA	IDA	(4C)	Input	Data	From	1/0	Group	A
IDB	IDB	(4D)	Input	Data	From	I/O	Group	В
IDC	IDC	(4E)	Input	Data	From	I/O	Group	С
IDD	IDD	(4F)	Input	Data	From	I/O	Group	D

1.6 us

0	3	4	5	6	7	8	11	12	15
4		11	2		G		Ra	D	

G, D  $\rightarrow$  I/O Address Lines Device Data  $\rightarrow$  Ra

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

Up to 16 bits of data are then transferred from the addressed device over the  $\mbox{I/O}$  bus to replace the contents of register Ra.

Affected: Ra

OCA	OCA	(40)	Output	Command	То	I/O	Group	Α
OCB	осв	(41)	Output	Command	то	I/O	Group	В
OCC	occ	(42)	Output	Command	То	I/O	Group	С
OCD	OCD	(43)	Output	Command	то	I/O	Group	D

1.33 us

0_		3	4	5	6	7	8	11	12 15
L	4		0 (	2	G			Ra	D

G, D  $\longrightarrow$  I/O Address Lines (R<sub>a</sub>)  $\longrightarrow$  I/O Data Lines

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

The 16 bit output command stored in register Ra is then transferred to the I/O register and placed on the I/O bus data lines.

Lines

ODA	ODA	(44)	Output	Data	То	1/0	Group	A
ODB	ODB	(45)	Output	Data	То	1/0	Group	В
ODC	ODC	(46)	Output	Data	то	I/O	Group	С
ODD	ODD	(47)	Output	Data	То	1/0	Group	D

1.33 us

0	3	4	5	6	7	8	11	12	15	G, D - I/O Address Lin
4		01	2	G			Ra		D	$(R_a) \rightarrow I/O$ Data Lines

The group (G) and device (D) numbers contained in the instruction word are placed on the I/O bus address lines.

The 16 bit data word stored in register Ra is then transferred to the I/O register and placed on the I/O bus data lines.

Three signals are needed to enable command decode. They are:

DRIOFN - Input/Output Function
DRCDFN - Command/Data Function

DRIOSN - I/O Sync

Data flow is determined by DRIOFN. If DRIOFN is true (low), data is input. If DRIOFN is false (high), data is output.

DFCDFN determines whether the instruction is a command or data. If DFCDFN is true (low), the instruction is data. If DRCDFN is false (high), the instruction is a command (or status).

These lines are interrogated at I/O sync time, DRIOSN.

	DRCDFN	DRCDFN	
DRIOFN	Input Data IDA	Input Status ISA	
DRIOFN	Output Data ODA	Output Command OCA	

# IV. PRIORITY INTERRUPTS

#### **OVERVIEW**

The MODCOMP II priority interrupt system contains three standard levels expandable in increments of four levels up to a total of 16 levels including the optional Power Fail Safe/Auto Start. Each external level can be selectively enabled and disabled under program control. Internal levels 0, 1, 2, 4, and 5, when present, are always enabled. In addition, the recognition of interrupt signals can be deferred for all interrupt levels below a selected level. Furthermore, interrupt request signals can be generated by instruction execution.

Of the three standard interrupt levels, two are I/O interrupt levels which have party line interrupt structures with 17 sub-priorities each and automatic source identification for up to 64 devices.

Each priority level is assigned two dedicated memory locations for the entry and return addresses unique to that level. The entry address of the interrupt processing routine is stored in one dedicated location. The return address, which is the contents of the Program Register (PR), is stored in the other dedicated location at the time the interrupt routine was entered. The 64 sub-levels of each I/O interrupt level share the return address of that level but are assigned unique entry address locations.

Nested interrupt routine execution is automatically handled for the 16 priority levels. The sub-levels of each I/O priority interrupt level cannot interrupt each other, but if several attempt to interrupt at the same time, the highest priority sub-level is recognized first.

#### LEVEL ASSIGNMENTS

The dedicated memory locations for each interrupt level and the signals connected to these levels are shown in Table 4-1.

There are three standard interrupt levels (4,C, and D) present in each MODCOMP II and they are connected to Unimplemented Instruction Trap and to the I/O Data and Service party lines. Power Fail Safe/Auto Start is standard in some models and optional in others. It is connected to the highest priority level (0).

The first optional group of interrupts (5,6,E, and F) are dedicated to the Executive Features Option. The second optional group of interrupts (1,2,3, and 7) are assigned to the System Protect option with level 7 available for an external interrupt signal. Priority level F is dedicated to the Task Scheduler Interrupt which allows the MAX III Executive to maintain a software task priority queue below the hardware priority queue. Levels 8,9,A,B are available as an optional group for connection to external equipment.

MODEL NUMBER	MEMORY LOCATION	LEVEL <sub>16</sub>	PROGRAM LINKAGE	INTERRUPT SIGNAL
3739 20 0		Return	Power Fail Safe/Auto Start	
	21		Entry	
3731	22	1	Return	Memory Parity
	23		Entry	
3731	24	2	Return	System Protect
	25		Entry	
3731	26	3	Return	Multiprocessor Communications
	27		Entry	
STD.	28	4	Return	Unimplemented Instruction Trap
	29		Entry	
3730	2A	5	Return	Floating Point Overflow
	2B		Entry	
3730	2C	6	Return	Real Time Clock
	2D		Entry	
3731	2E	7.	Return	External
	2F		Entry	
3732	30	8	Return	External
	31		Entry	
3732	32	9	Return	External
	33		Entry	
3732	34	A	Return	External
	35		Entry	
3732	36	В	Return	External
	37		Entry	
STD.	38	c	Return	I/O Data Party Line
	39		Not Used	
STD.	3A	D	Return	I/O Service Party Line
	3B		Not Used	-
3730	3C	E	Return	Console Interrupt
	3D		Entry	-
3730	3E	F	Return	Task Scheduler
	3F		Entry	
			-	

TABLE 4-1 INTERRUPT LEVEL ASSIGNMENTS

## INTERRUPT OPERATION AND PROGRAM CONTROL

Each interrupt level contains three flip-flops which collectively define the state of the level.

The <u>Request</u> flip-flop is set by the external interrupt request signal or by execution of the Set Interrupt Request (SIR) instruction. The purpose of this flip-flop is to store the request until it can be processed by the computer. It is reset by execution of either the Clear Interrupt and Return (CIR) or the Reset Interrupt Request (RIR) instruction.

The <u>Enable</u> flip-flop, when set, permits the stored request to interrupt the program. This flip-flop is set by execution of the Set Interrupt Enable (SIE) instruction and reset by execution of the Reset Interrupt Enable (RIE) instruction.

The <u>Active</u> flip-flop is set when the program interrupt signal is generated. It is not reset, except by execution of the Reset Interrupt Active (RIA) instruction, until the Clear Interrupt and Return (CIR) instruction is executed to exit an interrupt routine. Therefore, it indicates that an interrupt was being processed at this level and enables program control to be returned to the level if one or more higher priority interrupts occurred while the level was being serviced. The Clear Active and Return (CAR) operates just as the Clear Interrupt and Return (CIR) except that the request latch is not reset, thus allowing new reponses to be acknowledged that may have occurred while the level was active. The Set Interrupt Active (SIA) and Reset Interrupt Active (RIA) instructions are provided to enable a level to be made active without causing a program interruption. Program interruption can be deferred from the level made active down through all lower levels by execution of these two instructions.

Operation of the Master Clear switch resets all three flip-flops in each level, except the Enable flip-flops for levels 0, 1, 2, 4, and 5 (Power Fail-Safe/Auto Start, Memory Parity, System Protect, Unimplemented Instruction Trap and Floating Point Overflow).

Based on the operation of the three interrupt level flip-flops, the conditions necessary for interrupting the computer from a given interrupt level are:

- . The level must be enabled
- . A request signal must have occurred
- . No higher priority level must be active
- . The execution of the current instruction must be completed

When these conditions are met, program switching occurs. The current 16-bit contents of the Program Register are stored in the Return location assigned to the interrupting level. The 16-bit contents of the Entry location assigned to the level are then transferred to the Program Register, and the execution of the interrupt routine is started. Program switching requires 2.4  $\mu sec$ .

The interrupt level is cleared by execution of the Clear Interrupt and Return instruction. Execution of this instruction clears the Request and Active flip-flops of the highest active level and transfers the 16-bit contents of the dedicated return location to the Program Register.

## INTERRUPT SUB-LEVEL OPERATION AND PROGRAM CONTROL

The I/O data and Service Interrupt levels, C and D, each provide 17 sub-priorities which are assigned to peripheral devices and to external equipment (refer to Table 4-2). Addresses and dedicated memory locations are available for a total of 64 sub-levels for each of the two interrupt levels. The higher transfer rate devices such as the discs and analog input subsystems are assigned to the higher priority sub-levels. The data and service interrupt priorities are identical for each peripheral device. Each data and service sub-level is identified by the dedicated memory locations for storing subroutine entry addresses. Sub-levels of a given priority interrupt level cannot interrupt each other, but if several attempt to interrupt at the same time, the highest priority sub-level is recognized first. Any data interrupt sub-level can interrupt any service interrupt sub-level so that data transfers have precedence over error or status checking routines.

I/O PRIORITY SUB-LEVEL	INTERRUPT DATA	LOCATION SERVICE	PERIPHERAL DEVICE
0 1 2 3 4 5 6 6 7	81 82 90,91 98,99 83 92,93 A0-AF A0-AB 84	C1 C2 D1 D8,D9 C3 D3 E0-E7 E0-EB C4 C5	Moving Head Disc Fixed Head Disc High Level Analog Input Subsystems Communications Multiplexer High Performance Magnetic Tape Wide Range Analog Input Subsystem (#1) Input/Output Interface Subsystem (#1) MODAC Subsystem (#1) Moderate Performance Magnetic Tape Card Readers (300-1000 CPM)
9	86	C6	Card Punch
10 11 12	94 87 88	D4 C7 C8	Wide Range Relay Analog Input Subsystem  Line Printer (600 LPM)  X-Y Plotter
12	88	C8	Electrostatic Printer/Plotter
13	89	C9	High Speed Paper Tape Punch
14	8B	СВ	Line Printer 50-150 LPM
15	8A	CA	Teletype/Paper Tape Reader
16	B0-BF	F0-FF	Input/Output Interface Subsystem (#2)
16	во-вв	F0-FB	MODAC Subsystem (#2)

TABLE 4-2 Sub-Level Assignments

When a Data Interrupt is serviced (level  $C_{16}$ ), the contents of the Program Register are stored in memory location 38 and are replaced by the contents of the data interrupt entry location (80 to 3F) for the highest priority peripheral device. The contents of the entry location point to the unique interrupt subroutine for that I/O device.

The interrupt subroutine is exited with a CIR instruction which clears the interrupt level and branches to the address contained in memory location 38. If another Data Interrupt request is pending, the interrupt processing routine is re-entered immediately.

When a <u>Service Interrupt</u> is serviced, the operation is identical except that the entry and return addresses are 3A and 3B with dedicated sub-level interrupt locations CO-FF.

For additional information on the programming of the  ${\rm I/O}$  interrupts, refer to Section  ${\rm V.}$ 

#### **TRAPS**

Traps are defined as conditions which cause the execution of the current instruction to be aborted before completion and generate an interrupt request signal. Only these internal conditions operate as traps:

Unimplemented Instruction
Memory Parity
System Protect Violation
Floating Point Overflow

## Unimplemented Instruction Trap

An Unimplemented Instruction Trap occurs upon the execution of a REX instruction or when macro instructions (Op Codes  $1X_{16}$ ,  $3X_{16}$ , or  $5X_{16}$ ) are attempted. The floating point set belongs to the macro group and therefore floating point instructions are trapped when this option is not present in a computer.

When this trap occurs, the contents of the Program Register point to the memory location which contains the unimplemented instruction. Therefore, the instruction can be examined and be simulated by a subroutine. Keeping the contents of the Program Register from being advanced until the Unimplemented Instruction interrupt (level 4) becomes active, means that unimplemented instructions must not be present in any higher level interrupt routines.

The Unimplemented Instruction interrupt level is always enabled. It cannot be disabled by instruction execution. This condition prevents the possibility of stalling the machine due to an unimplemented instruction occurring when the level is disabled.

The Opcodes which have no assigned mnemonic are undefined instructions and their operation is unspecified. These undefined instructions will not generate an unimplemented instruction trap nor will they be executed as NOPs. These Opcodes should not be used.

## Memory Parity Trap

Instructions which result in a memory parity trap are aborted. If the optional memory parity interrupt level is not present, the computer will suspend all operations until the master clear switch is depressed. If the memory parity interrupt level is present, the interrupt will be processed in the normal fashion. If a parity error occurs during a higher priority interrupt subroutine (Power Fail Safe/Auto Start) the interrupt will not occur until the higher level is cleared. The memory parity interrupt is always enabled.

#### System Protect

This trap is used by the MAX III Modular Applications Executive to allow the checkout and execution of programs in unprotected areas of memory without interfering with the execution or integrity of programs residing in the protected areas of memory. This trap occurs if a memory protect violation, privileged instruction violation, or illegal branch is attempted by the unprotected program. The trap mechanism returns control to MAX III and results in the immediate aborting of the offending program.

A memory protect violation occurs when an attempt is made to write into protected memory by an instruction contained in unprotected memory. At this time a trap will occur and prevent the illegal write operation from occurring. If a branch is attempted into protected memory either directly or indirectly (a short indexed operation, for example), by an instruction or indirect address contained in unprotected memory, the branch occurs before the trap is implemented. The PR will be updated by the branch and will contain the address of protected memory into which the branch was made.

A privileged instruction violation occurs when a program in unprotect memory attempts to execute any CONTROL instruction, INPUT/OUTPUT instruction or INTERRUPT AND CALL instruction except REX.

The System Protect feature is enabled and disabled by the console key switch.

#### Floating Point Overflow

Floating point operands presented to the floating point unit must be normalized. However, floating point overflow or underflow will occur if the resultant exponent of a floating point operation is unable to be expressed within the range of the nine bit binary exponent field of the floating point format. If the resultant floating point fraction must be left shifted to normalize, the binary exponent must be decremented by one for each bit position left shifted. If the exponent decrements past all zeroes to all ones, floating point underflow has occurred.

If the resultant floating point fraction must be right shifted to be normalized, the binary exponent must be incremented by one for each bit position right shifted. If the exponent increments past all ones to all zeroes, floating point overflow has occurred.

Either occurrence causes the floating point overflow trap mechanism to terminate the normal FPU operation and does not allow any results to be transferred back to the CPU register file. The original register operands are maintained in the CPU register file and may be interrogated for further overflow/underflow clarification.

The floating point overflow trap level, when present in the system, is Level 5 and is always enabled.

# POWER FAIL SAFE/AUTO START INTERRUPT

When the a-c line voltage drops below 105 volts, an interrupt is generated a minimum of 200 memory cycles before the memory write current is disabled. This feature allows the inclusion and execution of a user supplied power failure interrupt routine to store all operands and I/O status, for example, which protects the integrity of the operating program stored in memory during transient or long term power failure conditions.

Upon the generation of a power failure interrupt, the Program Register is stored in memory location 20 and the power failure routine is entered using the address stored in location 21.

When a-c power is restored, the system is normalized, an interrupt is generated, and the start-up routine is entered using the address stored in location 21. The initial address of the auto-start subroutine should be stored in location 21 by the power failure subroutine. This level is always enabled.

# V. INPUT OUTPUT

#### **OVERVIEW**

The basic I/O facility of the MODCOMP II computer consists of a time-shared (party line) I/O bus capable of transferring data, commands and device status. Data can be transferred between any general register and any of up to 64 addressable peripheral devices. Up to 16 bits can be transferred in parallel over the bus under program control. In addition, the Direct Memory Processor (DMP) is available as an optional I/O facility which permits transfer of blocks of data to and from memory on a cycle stealing basis.

Figure 5-1 is the input/output subsystem block diagram. The I/O bus and a typical peripheral device controller are shown in addition to the computer I/O subsystem.

## INSTRUCTION EXECUTION SEQUENCE

The execution sequence for all I/O instructions - Input Data, Input Status, Output Data, Output Command - consists of:

- (1) The device address consisting of bits 6, 7 and 12-15 are transferred from the instruction register, through the I/O Control (Figure 5-1) to the addressed peripheral device controller.
- (2) A set of control signals are sent to the addressed controller which define the operation Input Data, Input Status, Output Data, or Output Command.
- (3) If the control signals call for an input, the device places a data word or status word on the 16 data lines of the I/O bus and this word is then transferred to register Ra as specified by the instruction. The fixed execution time for all input instructions is 1.6 microseconds. If the control signals call for an output, the contents of register Ra, as defined in the instruction word, are transferred to the output buffer register and then placed on the 16 data lines of the I/O bus. Execution of all output instructions is completed in a total of 1.33 microseconds.

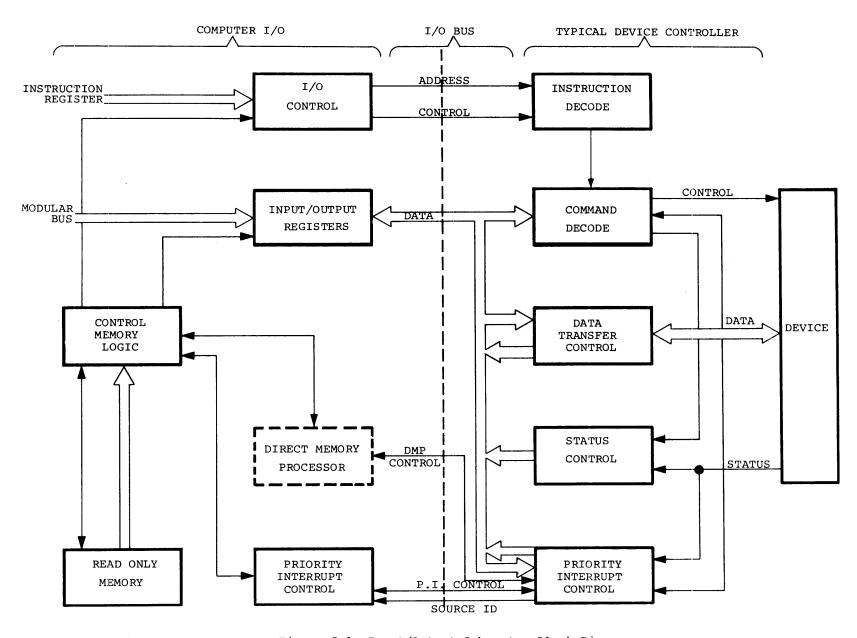
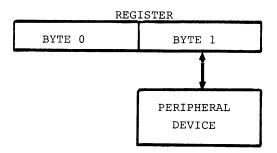


Figure 5-1 Input/Output Subsystem Block Diagram

#### TRANSFER FORMATS

Data is transferred over the I/O Bus as a 16-bit word. If a peripheral device requires or generates a data word of less than 16 bits, the device data word occupies the least significant bits of the 16-bit CPU data word with the unused bits appearing as zeroes. When less than 16 bits are output to a device, the outputs are taken from the less significant end of the register Ra or memory and zeroes are stored in the otherwise unused bits at the most significant end. This format is consistent with the operation of the byte manipulation instructions.

BYTE TRANSFER FORMAT



## REGISTER I/O TRANSFER MODES

Three transfer modes are available for program controlled transfers.

The <u>interrupt mode</u> can be used with any device which generates a transfer request signal. This group of devices includes all standard computer peripherals. The transfer request signal is connected to an interrupt level. Interrupt service routines can perform transfers and all required overhead functions at rates up to approximately 60K words per second.

The test and transfer mode is performed by first testing a device by means of the Input Status instruction. When the "Data Ready" status bit equals zero, a transfer can be made to or from the addressed device. The maximum transfer rate in this mode is determined almost entirely by the timing of the device.

The <u>burst mode</u> can be used with devices which can perform a word transfer any time the computer executes an I/O instruction addressed to the device. Output bursts of up to 15 words (one per register) can be performed at the burst rate of 750K words per second. Input bursts can be performed at 625K words per second. This mode is useful in applications such as updating a group of digital-to-analog converter registers.

## INPUT/OUTPUT INTERRUPTS

Two standard I/O priority interrupts are provided to initiate transfers between peripheral devices and the CPU. The higher priority data interrupt, level  $\rm C_{16}$ , is used to initiate a data word or byte transfer. The service interrupt, level  $\rm D_{16}$ , is used to initiate service routines for end of record, error, and similar signals. Under program control, each I/O priority interrupt can be connected or disconnected within each peripheral device. If a peripheral device has a stored interrupt request when a command is issued to disconnect the interrupt, the request will be reset. No interrupt signals are stored in a disconnected controller. Therefore when a controller is reconnected, all interrupt signals are cleared.

Even though all peripheral devices share the two standard I/O priority interrupts, the party line system used provides rapid response to interrupt requests. When a peripheral device requires a data transfer, the data request flip-flop in the device controller is set. Since the data request line is common to all peripheral devices, any data request flip-flop that is set will cause the data request line to be true. If no higher priority interrupt level is active, the CPU I/O subsystem will issue a data queue update command. In response to this command, all peripheral devices that have their data request flip-flop set, place their priority level on the data lines and their source ID on the source ID lines. The data lines are used to provide priority sub-levels for the data interrupt. The highest sub-level corresponds to data bit 0 on the data lines and the lowest sub-level to data bit 15. During the data queue update command each peripheral device examines all of the data lines corresponding to a higher priority sub-level than its own. If a higher priority sub-level is detected, the peripheral device removes its source ID from the source ID lines. At the end of the data queue update command the following occur:

- . The CPU internally stores the source ID of the highest priority peripheral device, to be used for defining the interrupt entry location.
- . The highest priority peripheral device resets its data request flip-flop and removes its source ID from the source ID Bus.
- . All peripheral devices remove their priority sub-levels from the data lines.

Next, the CPU stores the current contents of the Program Register in location  $^{38}_{16}$  and branches to the address contained in one of sixty-four dedicated locations  $^{(80-BF)}_{16}$  specified by the source ID. The subroutine is then entered to transfer data.

The service interrupt operates in the same manner as the data interrupt, except the dedicated return location is  $3A_{16}$  and the dedicated entry locations are  $^{\rm C0}_{16}{}^{\rm -FF}_{16}$ .

Refer to Section IV for more information on the I/O interrupts.

#### DIRECT MEMORY PROCESSOR

The DMP provides direct memory access capability for 8 peripheral device controllers. All 8 controllers can perform transfers of blocks of data to/from computer memory at the same time on an inter-leaved basis. Devices connected to DMP channels also accept Input Data and Output Data commands when not performing DMP controlled block transfers.

A pair of dedicated memory locations are assigned for each of the 8 controllers. Each controller is assigned a Transfer Count (TC) location  $(60-67)_{16}$  and a Transfer Address (TA) location  $(70-77)_{16}$  having the formats:

#### DMP TRANSFER PARAMETER FORMATS

	TRANSFER COUNT		
C <sub>*</sub> S	NEGATIVE WORD COUNT <16384	15	Transfer Single Block Transfer Chain of Blocks
	TRANSFER ADDRESS		
	ADDRESS FIELD		
0		15	

#### Transfer Initiation

Once a peripheral device is appropriately selected and initialized, a data transfer is started by storing the desired starting address for the transfer in the TA location and the negative number of words to be transferred in the TC location. An output command instruction in the transfer initiate format is then executed. Transfers occur automatically at the rate requested by the device. The TA and negative TC are incremented after each transfer. The maximum length of a single block is 16384 words.

When TC equals zero, a data interrupt is generated. If this interrupt is connected by the program to the interrupt (level  ${\it C}_{16}$ ) party line and if the level is enabled, the computer will be interrupted as soon as the data level reaches the top of the interrupt queue.

When the device can accept a new command, the service interrupt (level  $D_{16}$ ) is generated, if program connected to the service interrupt party line.

The use of both the data and service interrupts provides a choice between two "end of block" signals One occurs as soon as the last word has been transferred and the other occurs when the device is ready to be commanded again, which is often milliseconds after the last transfer.

#### Data Chaining

If bit 0 of the Transfer Count is set to zero (C=0) before the initiate command is executed, a new block of words will be transferred automatically after the transfer

of the current block is completed. The data interrupt signifies the completion of each block. The TA and TC parameters for the new block are obtained from the two memory locations immediately following those occupied by the current block. TC is taken from the first location and TA from the second location after the data block.

If C=0 in the TC parameter, data chaining will continue until a TC parameter is encountered with C=1.

#### Register File

All eight DMP channels are supplied with a pair of registers in which the current TA and TC are stored. Each time a block transfer is initiated, the contents of the TA and TC dedicated memory locations are automatically transferred to the two registers associated with the channel. Transfers can be made over these channels at rates up to 416K (input) or 340K (output) words per second, which are determined by the I/O subsystem timing.

At the end of a transfer sequence, just prior to SI generation, the final TA is stored in a dedicated location from the appropriate channel register.

#### PERIPHERAL DEVICE ASSIGNMENTS

All programming parameters for MODCOMP peripheral devices are listed in Table 5-1. Unassigned dedicated locations have been left for other peripheral devices, analog input subsystems, communication subsystems, custom devices and future system expansion.

# PROGRAMMING CONSIDERATIONS

The sequence of programming steps necessary to perform typical input/output functions are described in this section. The descriptions are general purpose and therefore are designed to cover all contingencies.

# REGISTER I/O INTERRUPT MODE SEQUENCE

#### New Command Initiation:

- . Store interrupt subroutine starting addresses in the data and service interrupt level dedicated locations.
- . Reset previous error and interrupt status by the execution of an Output Command instruction with a No Op output command, disconnecting both interrupts.
- . Test present device status by executing an Input Status instruction.
- . If status indicates inoperability or an invalid (all zero) status word, exit to error routine; otherwise continue.

I/O PRIORITY	INTERRUPT DATA	LOCATIONS SERVICE	DMP TC	LOCATIONS TA	DEVICE ADDRESS	PERIPHERAL DEVICE
0	81	Cl	61	71	01	Moving Head Disc
1	82	C2	62	72	02	Fixed Head Disc
2	90 91	Dl	60 63	70 73	10 11	High Level Analog Input Subsystem -Channel Output -Data Input
3	98 99	D8 D9	6F	7F	18 19	Communications Multip. -Controller -Channels
4	83	C3	63	73	03	High Performance Mag- netic Tape
5	92 93	D3	65 66	75 76	12 13	Wide Range Analog Input System -Channel Output -Data Input
6	A0-A7	E0-E7			20-27	Input/Output Interface Subsystem
7	84	C4	64	74	04	Moderate Performance Magnetic Tape
8	.85	C5			05	Card Readers (300 and 1,000 CPM)
9	86	C6			06	Card Punch
10	94	D <b>4</b>			14	Wide Range Relay Analog Input Subsystem
11	87	C7			07	Line Printer (600 LPM)
12	88	C8			08	X-Y Plotter
13	89	С9			09	Paper Tape Punch
14	8B	СВ			0B	Line Printer (50-150LPM)
15	8A	CA			0A	Teletype/Paper Tape Reader
16	A8-AF	E8-EF			28-2F	Input/Output Interface Subsystem

TABLE 5-1 Peripheral Device Interrupt Assignments

. Execute an output Command Instruction specifying register mode, input or output, connection of interrupts and any other modifiers required by the particular peripheral device. Exit and wait for interrupt.

The controller is now busy and will not respond to new initiation commands. It will respond to Input Status, Input Data and Output Data Instructions and an Output Command Instruction with a Terminate Command. The controller will produce the data interrupt when a data transfer is required and the service interrupt if a malfunction occurs or at the end of the media record.

#### Response to Data Interrupt:

- . The data interrupt processing routine is automatically entered when the requesting controller has the highest priority.
- . Preserve original contents of Rl execute an STM, Rl,A. Repeat for all other registers to be used as working registers.
- . Check word count, if transfer not complete, perform input or output operation as required. If output, load new data into appropriate place in register, execute Output Data Instruction and update word and byte counts appropriately. If input, execute Input Data Instruction and move or store data as required before updating word and byte counts.
- . If the last word required was transferred, an Output Command Instruction should be executed, issuing a Terminate Command to the controller. This will stop further transfers and reset the data interrupt request.
- . Restore the previous contents of the working registers.
- . Execute a CIR Instruction to exit the routine and return to the original program.

#### Response to Service Interrupt:

- . The Service Interrupt Processing routine is automatically entered if the requesting controller has the highest priority. This interrupt is generated after all hardware checks are complete and the controller can accept a new initiation command.
- . Preserve the original contents of Rl execute an STM,Rl,A. Repeat for all registers to be used as working registers.
- . Check validity of the transfer by issuing an Input Status instruction. If an abnormality is indicated, exit to error recovery routine.
- . If previous checks are satisfactory and no further tasks required for controller, restore the previous contents of the working registers and execute a CIR.
- . If previous checks are satisfactory and another transfer sequence is desired, execute an Output Command Instruction with a new initiation command. This command will reset any status conditions that may be set. Execute a CIR to exit the subroutine.

## REGISTER I/O TEST AND TRANSFER MODE

Register I/O transfers may be accomplished without the use of data interrupts. A "Data Ready" bit is provided in the standard status word for this purpose. To operate in this mode, the data interrupt is disconnected by the initiation command and the data ready bit is tested during each transfer sequence. Device control is performed in the same manner as in the interrupt mode.

## DIRECT MEMORY PROCESSOR I/O MODE

The optional DMP mode frees the program from the task of handling individual data word transfers, and increases net throughput capabilities. The software initiation and termination sequences are described in this Section in the most general manner possible.

#### New Command Initiation:

- . Store interrupt subroutine starting addresses in the two dedicated interrupt level locations.
- . Reset previous error or interrupt status by execution of an Output Command, which also disconnects both interrupt levels.
- . Test present status by executing an Input Status Instruction. If inoperability is indicated or an invalid (all zero) status word, exit to error routine.
- . Store a transfer address and a word count in the two DMP dedicated locations.
- . Execute an Output Command Instruction with an Initiate Command specifying DMP mode, input or output, connection of service interrupt, optional connection of data interrupt, and any other modifiers required by the particular peripheral.
- . Exit and wait for the interrupt.

The controller is now busy and will not respond to new initiation commands. It will respond only to an Input Status Instruction or an Output Command Instruction with a Terminate or No Op Command.

#### Response to Data Interrupt:

- . The data interrupt processing routine is automatically entered when the controller requesting has highest priority.
- . Preserve the original contents of Rl execute an STM,Rl,A. Repeat as required for all working registers.
- . The occurrence of this interrupt, in this mode, designates that the transfer of the block of data has been completed (TC=0). The program may use this fact to gain time to manipulate data prior to the completion of the physical media operation.
- . Execute a CIR, which will return control to the point of interruption.

#### Response to Service Interrupt:

- . The service interrupt processing routine is automatically entered when the requesting controller is the highest in the interrupt queue. This interrupt is generated after all hardware status checks are complete and the controller is ready to accept a new initiation command.
- . Preserve the original contents of Rl execute an STM,Rl,A. Repeat for all other registers to be used as working registers.
- . Check validity of the transfer by executing an Input Status instruction. If an abnormality is indicated, exit to an error recovery routine.
- . Check final transfer address in dedicated location. If improper, exit to error routine.
- . If the previous checks are satisfactory and no further tasks are required, execute a CIR instruction to return to the original program.
- . If the previous checks are satisfactory and another block transfer is desired, load the word count and transfer address into the dedicated locations. Then execute an Output Command Instruction with a new Initiation Command. This command will reset any status conditions that may be set.
- . Execute a CIR instruction to exit the routine.

#### OUTPUT COMMAND FORMATS

An Output Command Instruction transfers the 16 bit output command stored in register Ra to the I/O register where it is placed on the I/O bus data lines. There are three basic command formats; Select, Control and Transfer Initiate. The bit designations for each group are defined below. All standard peripheral controllers follow these format conventions. All Commands except End-of-Block and Terminate reset all stored status if the device is not busy. The specific commands and tests for each peripheral device are listed in Appendix C. The standard controllers interpret the command as follows:

#### Select Format

0	1	2	15
0	0		

#### BITS FUNCTION

- 0,1 Must both be zero. These bits specify the select format.
- 2-15 Specify a set up condition such as unit number (multi-unit controllers), density, head number, etc.

#### Control Format

0	1	2	3	4	5	6	7	15
0	1.	D	s	Е	т		P <sub>E</sub>	

#### BIT FUNCTION

0 Must be zero.

1 Must be one. This bit is used in conjunction with bit 0 to specify the control format.

Specifies the state of the data interrupt:

Zero - Disconnects the device controller and resets the request in the controller if present.

One - Connects the device controller sub-level to the data interrupt level.

If the DMP mode had previously been specified by a Transfer Initiate command, the data interrupt will occur when the Word Count = 0.

If the register I/O mode had previously been specified by a Transfer Initiate command, the data interrupt is defined as Data Request.

3 Specifies the state of the service interrupt.

Zero - Disconnects and resets the request if active.

One - Connect the interrupt, allowing it to become active.

The service interrupt may be caused by a variety of conditions such as end of record or error. The interrupt condition depends on controller design.

Specifies End-of-Block command when equal to one. No effect when equal to zero. The End-of-Block command causes the controller (except the Teletype Controller) to immediately generate a data interrupt if that interrupt had previously been connected. This function is useful for diagnostic and debugging purposes. An End-of-Block command will be accepted even when a controller is busy or operating in the DMP mode.

The responding device ignores all bits of the control format except 0, 1, 4 and 5.

Specifies Terminate command when equal to one. No effect when equal to zero. The Terminate command stops data transfer to/from the specified device and resets any non-active data interrupt, or DMP Data request. A Terminate command will be accepted when a controller is busy or in the DMP mode.

The Terminate command will also condition a controller to generate a service interrupt when the controller is subsequently ready to respond to another Transfer Initiate command. If the controller is not busy and the service interrupt has been previously connected, this interrupt will occur immediately. The responding device will ignore all bits of the control format except 0, 1, 4 and 5.

If a terminate command is issued with bit 7 set to one, a controller with DMP facilities will set its memory parity error status indicator. This command is normally issued automatically by the DMP I/O system if such an error is detected.

If bit 7 is set to one within a terminate command to some devices (TTY for example), an immediate operation abort occurs.

- Normally used to distinguish between a normal control command and a No-Op. See No-Op below.
- 7\* Specify a control function such as rewind, advance record, seek cylinder, etc..

#### No Op Command

When bits 4, 5 and 6 are all zero, bits 7-15 are ignored. The No Op command alters interrupt connection per the values of bits 2 and 3 whether or not the device is busy. The No Op command also resets all device status if the device is not busy.

#### Interrupt Disconnection and Termination

An interrupt may be reset by means of a Disconnect or Terminate command whenever the interrupt level is Active. However, if the level is not active but might become active immediately, an invalid request might occur on the I/O level. To accommodate this situation, requests at levels 80 and CO should execute a CIR and return to the interrupted program.

#### Transfer Initiate

0	1	2	3	4	5	15	
1	М	D	S	Ι			1

#### BITS FUNCTION

- 0 Must be one. This bit specifies the Transfer Initiate Format.
- 1 Specifies mode selection for subsequent data transfer.

Zero - Sets the device to the programmed register I/O mode. The device sends a data interrupt request, if connected, each time it requires a data transfer, including the first transfer.

One - Sets the device to the DMP transfer mode.

- Specifies the state of the data interrupt:
  - Zero Disconnects the device controller and resets the request in the controller if present.

One - Connects the device controller sub-level to the data interrupt level.

If the DMP mode had previously been specified by a Transfer Initiate command, the data interrupt will occur when the TC = 0.

If the register I/O mode had previously been specified by a Transfer Initiate command, the data interrupt is defined as Data Request.

<sup>\*</sup>Except as already noted.

3 Specifies the state of the service interrupt.

Zero - Disconnects and resets the request if active.

One - Connects the interrupt, allowing it to become active.

The service interrupt may be caused by a variety of conditions such as the end of the record or error. The interrupt condition depends on controller design.

4 Specifies the direction of data transfer.

Zero - Sets the device to the output transfer mode.

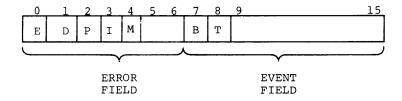
One - Sets the device to the input transfer mode.

5-15 Specifies a transfer initiate function such as write record or read card. (See Appendix C).

#### **INPUT STATUS FORMAT**

An Input Status instruction causes the contents of the 16 data lines to be transferred to the specified register Ra. The controller, as selected by the device address, puts its status word on the data lines and then the transfer occurs.

One basic format exists which is common to all controllers. This format encompasses two groups: Errors and Events. The error group has a pointer bit indicating if any error is set. The status format is so defined that a status word of all zeros is invalid, indicating a malfunctioning or non-existant controller.



#### BIT STATUS

0 Error Pointer Bit

Zero - An error has occurred and is defined in the field of bits 1 through 6.

One - No error has occurred.

#### BIT STATUS

Data transfer error.

Zero - No error.

One - Overflow or underflow error.

2 Parity or checksum error.

Zero - No error.

One - Device parity error.

- 3 Inoperable.
  - Zero Device operable (on-line).
  - One Device inoperable (off-line, interlock open, etc.)
- 4 Memory parity error.
  - Zero No error.
  - One A memory parity error was detected during a DMP transfer.
- 5-6 Specify error conditions unique to a device such as seek error.
- 7 Busy status of device controller.
  - Zero = Device controller not busy.
  - One Device controller busy.
- 8 Transfer Status. (Normally used when not operating in the interrupt mode).
  - Zero Device controller ready to transfer a data word.
  - One Device controller not ready to transfer a data word.
- 9-15 Specify device unique event conditions.

## I/O BUS INTERFACE

The MODCOMP II I/O Bus is a time-shared (party line) bi-directional bus capable of transferring data, commands, and device status. Since the bus is a party line, each device controller must request use of the bus on one or more of the three request lines available. When the bus is available for use, the CPU will issue interlock signals which inform the specified controller when it may use the bus.

The I/O cable is connected serially (daisy chained) to each Peripheral Controller Interface and Input Output Interface Subsystem. All controllers and the CPU are connected to the I/O bus through a cable driver/receiver set which isolates the logic from the effect of I/O bus loading and delays.

The I/O cable connection rules are:

- (1) The maximum combined cable length per computer is 100 feet.
- (2) All connections must be made via a cable driver/receiver set.
- (3) A maximum of eight cable driver/receiver sets can be connected to the cable.
- (4) Up to four device controllers can be connected to each driver/receiver set.

The block diagram of the computer I/O subsystem (Figure 5-2) shows the types of signals in the I/O cable (which is cut by the dashed line in the figure). Thirtynine pairs of lines are provided for communication between the CPU and external controllers.

The signals are described below.

Address (6 pairs DA00N through DA05N) - Designates which device controller is to respond to the control lines. The six-bit binary value is obtained from the I/O instruction word (bits 6-7, 12-15).

Control (5 pairs) - The four I/O instruction functions are coded on two lines.
(DACSN) - Command (false) / Data (true) and (DAION) - Input (true) / Output (false).

The I/O Sync signal (IOISN) indicates that an I/O transfer is occurring. When this signal is true, the Command/Data, Input/Output and Address signals are valid.

The other two control signals are Master Clear (IOICB), which is generated at the computer, and Clock (IOCLKN), which is a 5 MHZ square wave generated in the computer. The Master Clear signal normalizes all units and the Clock signal provides a timing signal to controllers, eliminating the need for an internal time base.

Data (16 pairs IOB00N through IOB15N) - All commands, status, data and controller priority are transmitted over this bi-directional bus.

 $\underline{\text{DMP}}$  Control (2 pairs) - The DMP Request signal (DMRQN) is sent to the computer by controllers operating under DMP Control. It signifies a request for a word transfer in the direction the DMP channel was last initialized.

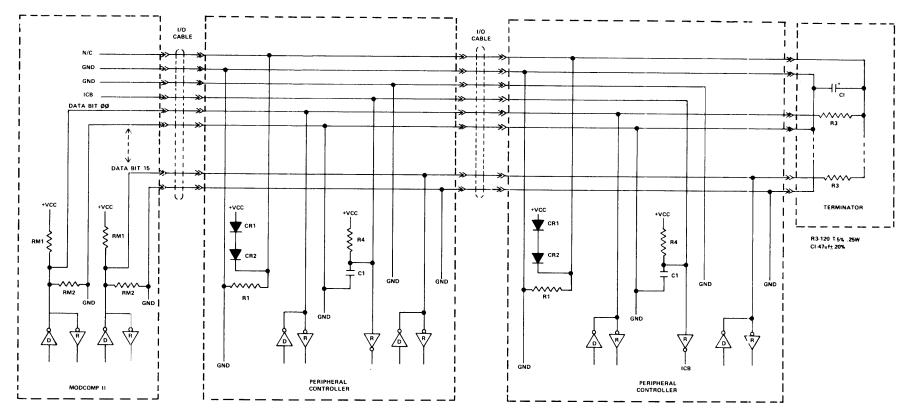
The DMP responds with a DMP Queue Update signal (UDDMQN). In response to this signal, each (if more than one) controller having a current DMP request places a true signal on one of the 16 data bus lines, indicating one of 16 DMP channel priorities.\*

The Source ID of the unit, which is a six-bit pointer to the dedicated registers for the channel is also placed on the Source ID lines. Each requesting unit examines the data bus lines to determine if a higher priority unit is also requesting a transfer. If so, the lower priority unit removes its priority and Source ID signals.

Priority Interrupt Control (4 pairs) - These signals consist of a Request signal and an Update Queue signal for both of the party line interrupts. Therefore the four signals are Data Request (DIRQN), Data Queue Update (UDDIQN), Service Request (SIRQN), and Service Queue Update (UDSIQN). These two pairs of signals are handled just like the DMP Request and DMP Queue Update except they are serviced at priority interrupt levels  $C_{16}$  and  $D_{16}$ , and DMP Transfers are serviced in the DMP independently of the interrupt priority structure. Each requesting controller places its priority bit on one of the 16 data bus lines and its Source ID on the Source ID lines. All but the highest priority requesting controller remove their signals during the Update Queue signal before the Source ID signals are transferred to the computer.

Source ID (6 pairs SID00N through SID05N) - Each device has a unique Source ID code which is sent to the computer in response to an update queue command. This code is used to identify the dedicated memory location assigned to the interrupt sublevel or the DMP channel.

<sup>\*</sup> Although 16 channel priorities are available, only eight (8) may be used.



RM1-220 a ±3%, 140MW

RM2-330 A ±3%, 140MW

D-SINKS 70 MA MIN. FOR LOGICAL 1 (.4V MAX.)

-100,A MAX. LEAKAGE FOR A LOGICAL Ø.(5V MAX.)

R-Ø OUT FOR INPUT > 2.0V -1 OUT FOR INPUT > 8.V MAX SOURCE CURRENT-2MA R1-220 A. ±5%, .25W

C1-.01, F, CERAMIC (-20%, +80%)

CR1, CR2-IN4001 or EQUIV.

R4-1.2K±5%, .25W

R1-220 A ±5%, .25W

C1-.01<sub>p</sub>F, CERAMIC (-20%+80%)

CR1, CR2-IN4001 OR EQUIV.

R4-1.2K±5%, .25W

FIGURE 5-2 INPUT/OUTPUT CABLE DIAGRAM

#### Signal Levels

The signal levels at the cable driver/receiver interface to the controller logic are:

Logic 0: } 2.4 to 5.5 Volts Except IOICB
Logic 1: 0 to 0.4 Volts which is inverted.

Figure 5-3 illustrates a MODCOMP II I/O Bus system with the driver/receiver sets and the associated termination network.

The I/O Clock (IOCLKN) a 5 MHz square wave, is distributed on the I/O bus for general purpose use in the controllers.

When the IOICB (Initial Condition Bus) comes true, each controller should normalize (not busy, no interrupt or DMP Requests, no interrupt enabled, device normalized, etc). This signal is present if the CPU power is going down and is present to normalize the system when power is returned to the CPU and when the master clear switch on the CPU console is depressed.

All signals are ground true except IOICB. This allows each controller to recognize the absence of a current sink in the CPU and normalize when CPU power is absent.

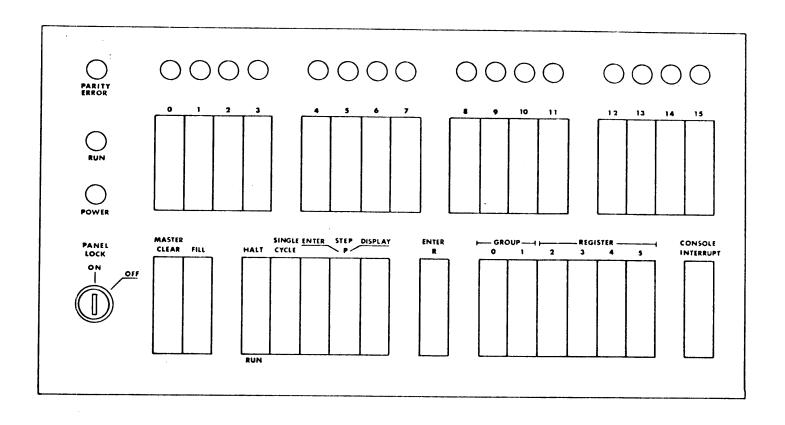


Figure 6-1 Control Panel

## VI. OPERATOR CONTROLS

The MODCOMP II control panel enables programs to be loaded into memory and executed under manual control. It also provides a number of debugging and maintenance aids.

#### **INDICATORS**

#### Data

The 16 Data Indicators display the contents of the register designated by the Register Select switches when the computer is halted. The bus traffic is displayed when the computer is in the run mode.

#### Parity Error

This indicator is lighted when the computer is halted due to a parity error, if no System Protect Feature, or until the memory parity interrupt is serviced, if the System Protect Feature is present.

#### Run

This indicator is lighted when the computer is in the run mode, which means not halted manually or by execution of the Halt instruction.

#### Power On

This indicator is lighted when a-c power is applied to the computer. The circuit breaker for switching power is located behind a hinged panel in the top front of the system cabinet which contains the computer.

#### **SWITCHES**

#### Data Entry

The 16 Data Entry switches are used to enter data into any register or memory location. The lowered position corresponds to a one value and the raised (normal) position corresponds to a zero value.

#### Panel Lock

In the ON position; this keyswitch disables all other control panel switches except the 16 Data Switches and the Console Interrupt switch. In the ON position it also enables the System Protect operation. All switches are enabled and the System Protect is disabled when the Panel Lock switch is in the OFF position.

#### Master Clear

Depressing this switch causes the computer and peripheral devices to be cleared. All interrupts and control signals and the contents of the Program and Instruction Registers are reset to the zero or cleared state.

#### Fill

Depressing this switch causes a bootstrap routine to be transferred to main memory (locations  $0-2D_{16}$ ) from read-only memory. The bootstrap routine automatically fills from the selected device. The device is selected by setting the proper device number in the Data Entry switches prior to depressing the Fill switch:

Fill From	Data Entr	У	Sv	vitches <sub>16</sub>
Paper Tape Reader	0	0	0	A
Card Reader	0	0	0	5
Mag tape	0	0	0	4
Fixed Head Disc	0	0	0	2
Moving Head Disc	0	0	0	1

#### Run/Halt

This switch is used to manually place the computer in either of the modes indicated by the switch positions. When the computer is manually halted, the Program Register points to the next instruction and the Instruction Register contains this next instruction. To resume operation at a new location, the Master Clear switch should be depressed to clear the Instruction Register, and the new location minus one should be manually entered into the Program Register. The Halt/Run switch should then be raised to the Run position.

#### Single Cycle

Depressing this switch causes the instruction presently stored in the Instruction Register to be executed. The Program Register is then advanced to the next instruction, and this instruction is accessed from memory and transferred to the Memory Data and Instruction Register. It can be displayed from the Memory Data Register.

#### Enter

When this switch is depressed, the word corresponding to the position of the Data Entry switches is stored in the memory location specified by the contents of the Program Register. The Program Register is not advanced.

#### Step P

The contents of the Program Register are incremented by one and the contents of the new memory location are entered into the MDR when this switch is depressed. The switch is provided to facilitate modifying or displaying the contents of consecutive memory locations.

#### Console Interrupt

Depressing this switch, in computer models having the Executive Features, causes an interrupt request signal to be sent to interrupt Level E.

#### Display

The contents of the memory location designated by the contents of the Program Register are displayed and entered into IR when this switch is depressed, providing the Register Select switches designate the Memory Data Register.

#### Enter R

Depressing this switch causes the contents of the Data Entry switches to be stored in the register specified by the Register Select switches.

#### Register Select

These switches are used to specify the register, the contents of which are to be displayed or modified. Whenever the computer is halted, the Data indicators display the contents of the specified register. When the Enter R switch is depressed, the specified register contents are replaced by the word specified by the Data Entry switches.

	COURCE Ly Register)	(Er	DESTINATION nter Register)
Switch Setting	Register Name	Switch Setting	Register Name
ØØ - ØF -	GPR FILE	øø	- NO DEST.
10 - 17 -	UNASSIGNED	Ø1 - ØF	- GPR FILE
18 -	NO SOURCE	100	- AR
19 - 1F -	GPR 1-7	11	- AR & TRB
2Ø - 2F -	AUX FILE	12, 13	- UNASSIGNED
3ø -	ACTIVE	14	- Ra PAUSE COUNTER
31 -	TRB	15	- Rb → PAUSE COUNTER
32 -	OPT PLØ	16	- BUS - PAUSE COUNTER
33 -	OPT PL1	17	- Rb PAUSE COUNTER
34 -	ENABLE	18	- HLT FF
35 -	TRA-	19	- RMI FF
36 -	P.I. QUEUE	1A, 1B	- UNASSIGNED
37 -	LITERAL	1C	- ROM ADDRESS REG.
38 -	REQUEST	1D	- UBR
39 -	UNASSIGNED	1E	- LBR
3A -	INPUT BUFFER	1F	- LBR, ZUBR
3B	UNASSIGNED	20 - 2F	- AUX FILE
3C -	PR	3Ø	- ACTIVE
. 3D -	OFLO (BITØ)	31	- TRB
	CARRY SAVE (BIT 15)	32	- OPT PLØ
3E -	UNASSIGNED	33	- OPT PL1
3F	MDR	34	- ENABLE
		35	- TRA
		36	- MA
		37	- MA & TRA
		38	- REQUEST
		39	- TRA & TRB
		3A	- OUTPUT BUFFER
		3B	- UNASSIGNED
		3C	- P & MA & TRA
		3D	- LSH MDR
		3E	- MSH MDR
		3F	- BOTH MDR

TABLE 6-1 REGISTER SELECT

#### **CONTROL PANEL OPERATION**

#### DISPLAY REGISTER

- 1. HLT/RUN switch to HLT
- 2. Register select switches to the desired register (Lights display contents of register)

#### LOAD REGISTER

- 1. HLT/RUN switch to HLT
- 2. Register select switches to the desired register
- 3. Place data into RO (switch register)
- 4. Press 'ENTER REGISTER'

#### LOAD MEMORY

- 1. HLT/RUN switch to HLT
- 2. Load Rll (PR) with desired starting address
- 3. Load RO (switch register) with desired data
- 4. Press 'ENTER MEMORY'

Note: To load sequential locations, press 'STEP' one time, and repeat steps 3 and 4.

#### DISPLAY MEMORY

- 1. HLT/RUN switch to HLT
- 2. Load Rll (PR) with desired starting address
- 3. Set the register select switches to R17
- 4. Press 'DISPLAY MEMORY' (Lights will display the contents of the selected memory location)

Note: To display sequential locations, press 'STEP' switch for each additional location to be displayed.

#### START PROGRAM

- 1. HLT/RUN switch to HLT
- 2. Load Rll (PR) with desired starting address
- 3. Press 'DISPLAY MEMORY'
- 4. HLT/RUN switch to RUN

#### SINGLE CYCLE PROGRAM

- 1. HLT/RUN switch to HLT
- 2. Register select switches to R17 (MDR)
- 3. Press 'DISPLAY MEMORY'
- 4. Press 'SINGLE CYCLE' for each instruction to be executed. R17 will display the contents of the first word of the next instruction to be executed.

Note: Interrupts will be ignored during single cycle.

#### FILL

- 1. HLT/RUN switch to HLT
- 2. RO bits 12-15 to the device address of filling device
  - A. 1 moving head disc
  - B. 2 fixed head disc
  - C. 4 mag tape
  - D. 5 card reader
  - E. A TTY or paper tape
- 3. Mag tape only

RO bits 1-7 to file for mag tape fill

#### Disc only

R0 bits 1-7 to  $\frac{\text{Starting Sector}}{\#100}$  (Starting Sector divided by 100)

- 4. Press 'MASTER CLEAR'
- 5. Press 'FILL'
- 6. HLT/RUN switch to RUN

# APPENDIX A. HEXADECIMAL TO DECIMAL CONVERSION

This appendix enables direct conversion of decimal numbers to/from hexadecimal numbers in the ranges:

HEXADECIMAL

9000

A000

B000

DECIMAL

36864

40960

HEXADECIMAL

DECIMAL

000 to FFF

HEXADECIMAL

1000

2000

3000

0000 to 4095

DECIMAL

4096

8192

12288

For numbers outside the range of the table, add the following values to the table figures:

			4000 5000 6000 7000	) ) )	•	12288 16384 20480 24576 28672 32768				B000 C000 D000 E000 F000			45056 49152 53248 57344 61440			
,	0	1	2	3					8	9	A	В	С	D	E	F
000 010 020 030	0032	0017	0016	0019	0020 0036 0052	0021 0037 0053	0022 0038 005 <b>4</b>	0023 0039 0055	0024 0040 0056	0025 0041 0057	0026 0042 0058	0027 0043 0059	0028 0044 0060	0029 0045 0061	0030 0046 0062	0015 0031 0047 0063
040 050 060 070	0064 0080 0096 0112	0065 0081 0097 0113	0066 0082 0098 0114	0067 0083 0099 0115	0068 0084 0100 0116	0069 0085 0101 0117	0070 0086 0102 0118	0071 0087 0103 0119	0072 0088 0104 0120	0073 0089 0105 0121	0074 0090 0106 0122	0075 0091 0107 0123	0076 0092 0108 0124	0077 0093 0109 0125	0078 0094 0110 0126	0079 0095 0111 0127
080 090 0A0 0B0	0160 0176	0161 0177	0146 0162 0178	0147 0163 0179	0148 0164 0180	0149 0165 0181	0150 0166 0182	0151 0167 0183	0152 0168 0184	0153 0169 0185	0138 0154 0170 0186	0155 0171 0187	0156 0172 0188	0157 0173 0189	0158 0174 0190	0159 0175 0191
0C0 0D0 0E0 0F0	0192 0208 0224 0240	0193 0209 0225 0241	0194 0210 0226 0242	0195 0211 0227 0243	0196 0212 0228 0244	0197 0213 0229 0245	0198 0214 0230 0246	0199 0215 0231 0247	0200 0216 0232 0248	0201 0217 0233 0249	0202 0218 0234 0250	0203 021 <b>9</b> 0235 0251	0204 0220 0236 0252	0205 0221 0237 0253	0206 0222 0238 0254	0207 0223 0239 0255
100 110 120 130	0256 0272 0288 0304	0257 0273 0289 0305	0258 0274 0290 0306	0259 0275 0291 0307	0260 0276 0292 0308	0261 0277 0293 0309	0262 0278 0294 0310	0263 0279 0295 0311	0264 0280 0296 0312	0265 0281 0297 0313	0266 0282 0298 0314	0267 0283 0299 0315	0268 0284 0300 0316	0269 0285 0301 0317	0270 0286 0302 0318	0271 0287 0303 0319
140 150 160 170	0320 0336 0352	0321 0337 0353	0322 0338 0354	0323 0339 0355	0324 0340 0356	0325 0341 0357	0326 0342 0358	0327 0343 0359	0328 0344 0360	0329 0345	0330 0346 0362 0378	0331	0332 0348 0364	0333	0334 0350 0366	0335 0351 0367
180 190 1A0 1B0	0384 0400 0416 0432	0385 0401 0417 0433	0386 0402 0418 0434	0387 0403 0419 0435	0388 0404 0420 0436	0389 0405 0421 0437	0390 0406 0422 0438	0391 0407 0423 0439	0392 0408 0424 0440	0393 0409 0425 0441	0394 0410 0426 0442	0395 0411 0427 0443	0396 0412 0428 0444	0397 0413 0429 0445	0398 0414 0430 0446	0399 0415 0431 0447
1C0 1D0 1E0 1F0	0448 0464 0480	0449 0465 0481	0450 0466 0482	0451 0467 0483	0452 0468 0484	0453 0469 0485	0454 0470 0486	0455 0471 0487	0456 0472	0457	0458 0474 0490 0506	0459	0460 0476	04 <b>6</b> 1 0477	0462 0478	0463 0479

200 210 220 230	0528 0544	0529 0545	0530 0546	0531 0547	0532 0548	0533 0549	0534 0550	0535 0551	0536 0552	0537 0553	0522 0538 0554 0570	0539 0555	0540 0556	0541 0557	0542 0558	0543 0559
240 250 260 270	0592 0608	0593 0609	0594 0610	0595 0611	0596 0612	0597 0613	0598 0614	0599 0615	0600 0616	0601 0617	0586 0602 0618 0634	0603 0619	0604 0620	0605 0621	0606 0622	0607 0623
280 290 2A0 2B0	0656 0672	0657 0673	0658 0674	0659 0675	0660 0676	0661 0677	0662 0678	0663 0679	0664 0680	0665 0681	065 <b>0</b> 0666 0682 0698	0667 0683	0668 0684	0669 0685	0670 0686	0671 0687
2C0 2D0 2E0 2F0	0720 0736	0721 0737	0722 0738	0723 0739	0724 0740	0725 <b>074</b> 1	0726 0742	0727 0 <b>74</b> 3	0728 0744	0729 0745	0714 0730 0746 0762	0731 0 <b>747</b>	0732 0 <b>74</b> 8	0733 0 <b>749</b>	0734 0750	0735 0751
300 310 320 330	0784 0800	0785 0801	0786 0802	0787 0803	0788 0804	0789 0805	0790 0806	0791 0807	0792 080 <b>8</b>	0793 0809	0778 0794 0810 0826	0795 0811	0796 0812	0797 0813	0798 081 <b>4</b>	0799 0815
340 350 360 370	0848 0864	0849 0865	0850 0866	0851 0867	0852 0868	0853 0869	0854 0870	0855 0871	0856 0872	0857 0873	0842 0858 0874 0890	0859 0875	0860 0876	0861 0877	0862 0878	0863 0879
380 390 3A0 3B0	0912 0928	0913 0929	0914 0930	0915 0931	0916 0932	0917 0933	0918 0934	0 <b>9</b> 19 0935	0920 0936	0921 0937	0906 0922 0938 0954	0923 0939	0924 0940	0925 0941	0926 0942	0927 0943
3C0 3D0 3E0 3F0	0976 0992	0977 0993	0978 0994	0979 0995	0980 0996	0981 0997	0982 0998	0983 0999	0984 1000	0985 1001	0970 0986 1002 1018	0987 1003	0988 1004	0989 1005	0990 1006	0991 1007
<u> </u>	0	1	2	3	4	5	6	7	8	9	A	В	С	D	E	F
400 410 420 430	1040 1056	1041 1057	1042 1058	1043 1059	1044 1060	1045 1061	1046 1062	1047 1063	1048 1064	1049 1065	1034 1050 1066 1082	1051 1067	1052 1068	1053 1069	1054 1070	1055 1071
440 450 460 470	1104 1120	1105 1121	1106 1122	1107 1123	1108 1124	1109 1125	1110 1126	1111 1127	1112 1128	1113 1129	1098 1114 1130 1146	1115 1131	1116 1132	1117 1133	1118 1134	1119 1135
480 490 4A0 4B0	1152 1168 1184	1153 1169 1185	1154 1170 1186	1155 1171 1187	1156 1172 1188	1157 1173 1189	1158 1174 1190	1159 1175 1191	1160 1176 1192	1161 1177 1193		1163 1179 1195	1164 1180 1196	1165 1181 1197	1166 1182 1198	1167 1183 1199
4C0 4D0 4E0 4F0	1216 1232 1248	1217 1233 1249	1218 1234 1250	1219 1235 1251	1220 1236 1252	1221 1237 1253	1222 1238 1254	1223 1239 1255	1224 12 <b>4</b> 0 1256	1225 1241 1257	1226 1242 1258 1274	1227 1243 1259	1228 1244 1260	1229 1245 1261	1230 1246 1262	1231 1247 1263
500 510 520 530	1280 1296 1312	1281 1297 1313	1282 1298 1314	1283 1299 1315	128 <b>4</b> 1300 1316	1285 1301 1317	1286 1302 1318	1287 1303 1319	1288 1304 1320	1289 1305 1321	1274 1290 1306 1322 1338	1291 1307 1323	1292 1308 1324	1293 1309 1325	1294 1310 1326	1295 1311 1327
540 550 560 570	1344 1360 1376	1345 1361 1377	1346 1362 1378	1347 1363 1379	1348 1364 1380	1349 1365 1381	1350 1366 1382	1351 1367 1383	1352 1368 1384	1353 1369 1385	1354 1370 1386 1402	1355 1371 1387	1356 1372 1388	1357 1373 1389	1358 1374 1390	1359 1375 1391
580 590 5A0 5B0	1408 1424 1440	1409 1425 1441	1410 1426 1442	1411 1427 1443	1412 1428 1444	1413 1429 1445	1414 1430 1446	1415 1431 1447	1416 1432 1448	1417 1433 1449	1418 1434 1450 1466	1419 1435 1451	1420 1436 1452	1421 1437 1453	1422 1438 1454	1423 1439 1455
5C0 5D0 5E0 5F0	1472 1488 1504	1473 1489 1505	1474 1490 1506	1475 1491 1507	1476 1492 1508	1477 1493 1509	1478 1494 1510	1779 1495 1511	1480 1496 1512	1481 1497 1513	1482	1483 1499 1515	148 <b>4</b> 1500 1516	1485 1501 1517	1486 1502 1518	1487 1503 1519

	0	1	2	3	4	5	6	7	8	9	A	В	С	D	E	F
600 610 620 630	1552 1568	1553 1569	7 1538 3 1554 9 1570 6 1586	1555 1571	1556 1572	1557 1573	1558 1574	1559 1575	1560 1576	1561 1577	1562 1578	1563 1579	15 <b>64</b> 1580	1565 1581	1566 1582	1567 1583
640 650 660 670	1616 1632	1617 1633	1602 1618 1634 1650	1619 1635	1620 1636	1621 1637	1622 1638	1623 1639	1624 1640	1625 1641	1626 1642	1627 1643	1628 1644	1629 1645	1630 1646	1631 1647
680 690 6A0 6B0	1680 1696	1681 1697	1666 1682 1698 1714	1683 1699	1684 1700	1685 1701	1686 1702	1687 1703	1688 1704	1689 1705	1690 1706	1691 1707	1692 1708	1693 1709	1694 1710	1695 1711
6C0 6D0 6E0 6F0	1744 1760	1745 1761	1730 1746 1762 1778	1747 1763	1748 1764	1749 1765	1750 1766	1751 1767	1752 1768	1753 1769	1754 1770	1755 1771	1756 1772	1757 1773	1758 1774	1759 <b>1</b> 775
700 710 720 730	1808 1824	1809 1825	1794 1810 1826 1842	1811 1827	1812 1828	1813 1829	1814 1830	1815 1831	1816 1832	1817 1833	1818 1834	1819 1835	1820 1836	1821 1837	1822 1838	1823 1839
740 750 760 770	1872 1888	1873 1889	1858 1874 1890 1906	1875 1891	1876 1892	1877 1893	1878 1894	1879 1895	1880 1896	1881 1897	1882 1898	1883 1899	1884 1900	1885 1901	1886 1902	188 <b>7</b> 1903
780 790 7A0 7B0	1936 1952	1937 1953	1922 1938 1954 1970	1939 1955	1940 1956	1941 1957	1942 1958	1943 1959	1944 1960	1945 1961	1946 1962	1947 1963	1948 1964	1949 1965	1950 1966	1951 1967
7C0 7D0 7E0 7F0	2000	2001 2017	1986 2002 2018 2034	2003 2019	2004 2020	2005 2021	2006 2022	2007 2023	2008 2024	2009 2025	2010 2026	2011 2027	2012 2028	2013 2029	2014 2030	2015
	0	1	2	3	4	5	6	7	8	9	A	В	С	D	E	F
800 810 820 830	2048 2064 2080	2049 2065 2081	2 2050 2066 2082 2098	2051 2067 2083	2052 2068 2084	2053 2069 2085	2054 2070 2086	2055 2071 2087	2056 2072 2088	2057 2073 2089	2058 2074 2090	2059 2075 2091	2060 2076 2092	2061 2077 2093	E 2062 2078 2094	2063 2079 2095
810 820	2048 2064 2080 2096 2112 2128 2144	2049 2065 2081 2097 2113 2129 2145	2050 2066 2082	2051 2067 2083 2099 2115 2131 2147	2052 2068 2084 2100 2116 2132 2148	2053 2069 2085 2101 2117 2133 2149	2054 2070 2086 2102 2118 2134 2150	2055 2071 2087 2103 2119 2135 2151	2056 2072 2088 2104 2120 2136 2152	2057 2073 2089 2105 2121 2137 2153	2058 2074 2090 2106 2122 2138 2154	2059 2075 2091 2107 2123 2139 2155	2060 2076 2092 2108 2124 2140 2156	2061 2077 2093 2109 2125 2141 2157	E 2062 2078 2094 2110 2126 2142 2158	2063 2079 2095 2111 2127 2143 2159
810 820 830 840 850 860	2048 2064 2080 2096 2112 2128 2144 2160 2176 2192 2208	2049 2065 2081 2097 2113 2129 2145 2161 2177 2193 2209	2050 2066 2082 2098 2114 2130 2146	2051 2067 2083 2099 2115 2131 2147 2163 2179 2195 2211	2052 2068 2084 2100 2116 2132 2148 2164 2180 2196 2212	2053 2069 2085 2101 2117 2133 2149 2165 2181 2197 2213	2054 2070 2086 2102 2118 2134 2150 2166 2182 2198 2214	2055 2071 2087 2103 2119 2135 2151 2167 2183 2199 2215	2056 2072 2088 2104 2120 2136 2152 2168 2184 2200 2216	2057 2073 2089 2105 2121 2137 2153 2169 2185 2201 2217	2058 2074 2090 2106 2122 2138 2154 2170 2186 2202 2218	2059 2075 2091 2107 2123 2139 2155 2171 2187 2203 2219	2060 2076 2092 2108 2124 2140 2156 2172 2188 2204 2220	2061 2077 2093 2109 2125 2141 2157 2173 2189 2205 2221	E 2062 2078 2094 2110 2126 2142 2158 2174 2190 2206 2222	2063 2079 2095 2111 2127 2143 2159 2175 2191 2207 2223
810 820 830 840 850 860 870 880 890 8A0	2048 2064 2080 2096 2112 2128 2144 2160 2176 2192 2208 2224 2240 2256 2272	2049 2065 2081 2097 2113 2129 2145 2161 2177 2193 2209 2225 2241 2257 2273	2050 2066 2082 2098 2114 2130 2146 2162 2178 2194 2210	2051 2067 2083 2099 2115 2131 2147 2163 2179 2195 2211 2227 2243 2259 2275	2052 2068 2084 2100 2116 2132 2148 2164 2196 2212 2228 2244 2260 2276	2053 2069 2085 2101 2117 2133 2149 2165 2181 2197 2213 2229 2245 2261 2277	2054 2070 2086 2102 2118 2134 2150 2166 2182 2198 2214 2230 2246 2262 2278	2055 2071 2087 2103 2119 2135 2151 2167 2183 2199 2215 2231 2247 2263 2279	2056 2072 2088 2104 2120 2136 2152 2168 2184 2200 2216 2232 2248 2248 2264 2280	2057 2073 2089 2105 2121 2137 2153 2169 2185 2201 2217 2233 2249 2265 2281	2058 2074 2090 2106 2122 2138 2154 2170 2186 2202 2218 2234 2250 2266 2282	2059 2075 2091 2107 2123 2139 2155 2171 2187 2203 2219 2235 2251 2267 2283	2060 2076 2092 2108 2124 2140 2156 2172 2188 2204 2220 2236 2252 2268 2284	2061 2077 2093 2109 2125 2141 2157 2173 2189 2205 2221 2237 2253 2269 2285	E 2062 2078 2094 2110 2126 2142 2158 2174 2190 2206 2222 2238 2254 2270 2286	2063 2079 2095 2111 2127 2143 2159 2175 2191 2207 2223 2239 2255 2271 2287
810 820 830 840 850 860 870 880 890 8A0 8B0 8C0 8D0 8E0	2048 2064 2080 2096 2112 2128 2144 2160 2176 2192 2208 2224 2240 2256 2272 2288 2304 2320 2336	2049 2065 2081 2097 2113 2129 2145 2161 2177 2193 2209 2225 2241 2257 2273 2289 2305 2321 2337	2050 2066 2082 2098 2114 2130 2146 2162 2178 2194 2210 2226 2242 2258 2274	2051 2067 2083 2099 2115 2131 2147 2163 2179 2195 2211 2227 2243 2259 2275 2291 2307 2323 2339	2052 2068 2084 2100 2116 2132 2148 2164 2196 2212 2228 2244 2260 2276 2292 2308 2324 2344	2053 2069 2085 2101 2117 2133 2149 2165 2181 2197 2213 2229 2245 2261 2277 2293 2309 2325 2341	2054 2070 2086 2102 2118 2134 2150 2166 2182 2198 2214 2230 2246 2262 2278 2294 2310 2326 2342	2055 2071 2087 2103 2119 2135 2151 2167 2183 2199 2215 2231 2247 2263 2279 2295 2311 2327 2343	2056 2072 2088 2104 2120 2136 2152 2168 2184 2200 2216 2232 2248 2264 2280 2296 2312 2328 2324	2057 2073 2089 2105 2121 2137 2153 2169 2185 2201 2217 2233 2249 2265 2281 2297 2313 2329 2345	2058 2074 2090 2106 2122 2138 2154 2170 2186 2202 2218 2234 2250 2266 2282 2298 2314 2330 2346	2059 2075 2091 2107 2123 2139 2155 2171 2187 2203 2219 2235 2251 2267 2283 2299 2315 2331 2347	2060 2076 2092 2108 2124 2140 2156 2172 2188 2204 2220 2236 2252 2268 2284 2300 2316 2332 2348	2061 2077 2093 2109 2125 2141 2157 2173 2189 2205 2221 2237 2253 2269 2285 2301 2317 2333 2349	E 2062 2078 2094 2110 2126 2142 2158 2174 2190 2206 2222 2238 2254 2270 2286 2302 2318 2334 2350	2063 2079 2095 2111 2127 2143 2159 2175 2191 2207 2223 2239 2255 2271 2287 2303 2319 2335 2351
810 820 830 840 850 860 870 880 890 8A0 8B0 8C0 8F0 900 910 920	2048 2064 2080 2096 2112 2128 2144 2160 2176 2192 2208 2224 2240 2256 2272 2288 2304 2320 2336 2352 2368 2384 2400	2049 2065 2081 2097 2113 2129 2145 2161 2177 2193 2209 2225 2241 2257 2273 2289 2305 2321 2337 2353 2369 2385 2401	2050 2066 2082 2098 2114 2130 2146 2162 2178 2194 2210 2226 2242 2258 2274 2290 2306 2322 2338	2051 2067 2083 2099 2115 2131 2147 2163 2179 2195 2211 2227 2243 2259 2275 2291 2307 2323 2339 2355 2371 2387 2403	2052 2068 2084 2100 2116 2132 2148 2164 2196 2212 2228 2244 2260 2276 2292 2308 2324 2340 2356 2372 2388 2404	2053 2069 2085 2101 2117 2133 2149 2165 2181 2197 2213 2229 2245 2261 2277 2293 2309 2325 2341 2357 2373 2389 2405	2054 2070 2086 2102 2118 2134 2150 2166 2182 2198 2214 2230 2246 2262 2278 2294 2310 2326 2342 2358 2374 2390 2406	2055 2071 2087 2103 2119 2135 2151 2167 2183 2199 2215 2231 2247 2263 2279 2295 2311 2327 2343 2359 2375 2391 2407	2056 2072 2088 2104 2120 2136 2152 2168 22168 22216 2232 2248 2264 2280 2296 2312 2328 2328 2344 2360 2376 2376 2392 2408	2057 2073 2089 2105 2121 2137 2153 2169 2185 2201 2217 2233 2249 2265 2281 2297 2313 2329 2345 2361 2377 2393 2409	2058 2074 2090 2106 2122 2138 2154 2170 2186 2202 2218 2234 2250 2266 2282 2298 2314 2330 2346 2362 2378 2378 2410	2059 2075 2091 2107 2123 2139 2155 2171 2187 2203 2219 2235 2251 2267 2283 2299 2315 2331 2347 2363 2379 2395 2411	2060 2076 2092 2108 2124 2156 2172 2188 2204 2220 2236 2252 2268 2284 2300 2316 2332 2348 2364 2380 2396 2412	2061 2077 2093 2109 2125 2141 2157 2173 2189 2205 2221 2237 2253 2269 2285 2301 2317 2333 2349 2365 2381 2397 2413	E 2062 2078 2094 2110 2126 2142 2158 2174 2190 2206 2222 2338 2254 2270 2286 2302 2318 2334 2350 2366 2382 2398 2414	2063 2079 2095 2111 2127 2143 2159 2175 2191 2207 2223 2239 2255 2271 2287 2303 2319 2335 2351 2367 2383 2399 2415
810 820 830 840 850 860 870 880 890 8A0 8B0 8C0 8F0 910 920 930 940 950 960	2048 2064 2096 2112 2128 2144 2160 2176 2192 2208 2224 2240 2256 2272 2288 2304 2320 2336 2352 2368 2342 2400 2416 2432 2448 2448 2464	2049 2065 2081 2097 2113 2129 2145 2161 2177 2193 2209 2225 2241 2257 2273 2289 2305 2321 2337 2353 2369 2369 2417 2417 2433 2449 2465	2050 2066 2082 2098 2114 2130 2146 2162 2178 2194 2210 2226 2242 2258 2274 2290 2306 2322 2338 2354 2370 2386 2402	2051 2067 2083 2099 2115 2131 2147 2163 2179 2195 2211 2227 2243 2259 2275 2291 2307 2323 2339 2355 2371 2403 2419 2435 2467	2052 2068 2084 2100 2116 2132 2148 2164 2196 2212 2228 2244 2260 2276 2292 2308 2324 2340 2356 2372 2388 2404 2420 2436 2452 2468	2053 2069 2085 2101 2117 2133 2149 2165 2181 2197 2213 2229 2245 2261 2277 2293 2309 2325 2341 2357 2373 2389 2405 2421 2437 2453 2469	2054 2070 2086 2102 2118 2134 2150 2166 2182 2198 2214 2230 2246 2262 2278 2294 2310 2326 2342 2358 2374 2390 2406 2422 2438 2454 2470	2055 2077 2087 2103 2119 2135 2151 2167 2183 2295 2215 2231 2247 2263 2279 2295 2311 2327 2343 2359 2375 2391 2472 2473 2473 2473 2473 2473 2473 2473	2056 2072 2088 2104 2120 2136 2152 2168 2184 22016 2232 2248 2296 2312 2328 2344 2360 2376 2376 2376 2476 2440 2456 2472	2057 2078 2089 2105 2121 2137 2169 2185 2201 2217 2233 2249 2265 2281 2297 2313 2329 2345 237 2393 2425 2441 2457 2473	2058 2074 2090 2106 2122 2138 2154 2170 2186 2202 2218 2234 2250 2282 2298 2314 2330 2346 2362 2378 2394 2426 2426 2428 2474	2059 2075 2091 2107 2123 2139 2155 2171 2187 2203 2219 2235 2251 2283 2299 2315 2331 2347 2363 2379 2427 2443 2427	2060 2076 2092 2108 2124 2140 2156 2172 2188 2204 22236 2252 2284 2300 2316 2332 2348 2384 2384 2386 2424 2428	2061 2077 2093 2109 2125 2141 2157 2173 2189 2205 2221 2237 2253 2269 2285 2301 2317 2333 2349 2365 2381 2397 2445 2461 2477	E 2062 2078 2094 2110 2126 2142 2158 2174 2190 2206 2222 2238 2254 2270 2286 2302 2318 2334 2350 2366 2382 2318 2446 2462 2478	2063 2079 2075 2111 2127 2143 2159 2175 2191 2207 2223 2239 2255 2271 2287 2303 2319 2335 2351 2367 2383 2399 2415 2447 2463 2479

		0		L :	2	3 4	5	6	5 7	' 8	9	) A	В	C	D	E	F
A	.00 10 20 30	259	2 259	77 25 93 259	78 25 94 259	79 258 95 259	0 258 6 259	1 258 7 259	2 258 8 259	3 258 9 260	4 258 0 260	35 258	6 258	1 257 7 258	2 2573 8 2589	2574 2590	
A	40 50 60 70	262 264 265	4 262 0 264 6 265	25 262 1 264 57 265	26 262 12 264 58 256	27 262 13 264 59 266	8 2629 4 2649 0 266	9 263 5 264 1 266	0 263 6 264 2 266	1 263 7 264 3 266	2 263 8 264 4 266	3 263 9 265 5 266	4 263 0 265 6 266	5 2630 1 2653	6 2627	2638 2654	2639 2655
A!	80 90 A0 B0	2688 2704 2720	8 268 4 270 0 272	9 269 5 270 1 272	00 269 06 270 22 272	1 269 17 270 23 272	2 2693 8 2709 4 2729	3 269 9 271 5 272	4 269 0 271 6 272	5 269 1 271 7 272	6 269 2 271 8 272	7 269 3 271	8 2699 4 2719	9 2700 5 2710	4 2685 0 2701 6 2717 2 2733 8 2749	2702 2718	2703 2719
1		2752 2768 2784	2 275 3 276 1 278	3 275 9 277 5 278	4 275 0 277 6 278	5 275 1 277 7 278	6 2757 2 2773 8 2789	7 275 3 277 1 2 <b>7</b> 91	8 275 4 277	9 2760 5 2770	0 276 6 277	1 276; 7 2778	2 2763 3 2779	3 2764 9 2780	3 2749 4 2765 0 2781 5 2797 2 2813	2766 2782	2767 2783
B0 B1 B2 B3	LO 20	2816 2832 2848	281 283 284	7 281 3 283 9 285	8 281 4 283 0 285	9 2820 5 2830 1 2852	0 2821 5 2837 2 2853	2822 2838 2854	2 2823 3 2839 4 2859	3 2824 9 2840 5 2856	1 282 284 3 285	5 2826 1 2842	2827 2 2843	2828 2844	2 2813 3 2829 4 2845 9 2861 5 2877	2830 2846	2831 2847
B4 B5 B6 B7	0	2880 2896 2912	2883 2891 2913	1 288 7 289 3 291	2 288 8 289 4 291	3 2884 9 2900 5 291 <i>6</i>	1 2885 2901 2917	2886 2902 2918	2887 2 2903 3 2919	7 2888 3 2904	2889	9 2890 5 2906	2891 2907	2892	2877 2893 2909 2925 2941	2894 2910	2895 2911
B8 B9 BA BB	0	2944 2960 2976	2945 2961 2977	5 2940 L 2962 7 2978	6 294° 2 296° 8 297°	7 2948 3 2964 9 2980	2949 2965	2950 2966	2951 2967	2952 7 2968	2953	2954 2970	2955 2971	2956 2972	2957 2973 2989 3005	2958 2974	2959 2975
BC BD BE BF	0 0	3008 3024 3040	3009 3025 3041	3010 3026 3042	0 3013 5 3023 2 3043	L 3012 7 3028 3 3044	3013 3029 3045	3014 3030	3015 3031	3016	3017	3018 3034	3019 3035	3020 3036	3021 3037 3053 3069	3022 3038	3023 3039
		0	1	2	3	4	5	6	7	8	9	A	В	C	D		
C10 C10 C20 C30	0   :	3104	3105	3106	3107	3108	3109	3110	3079 3095	3080 3096	3081 3097	3082 3098	3083 3099	3084 3100	3085 3101 3117 3133	3102	3103
C40 C50 C60 C70		3136 3152 3168 3184	3137 3153 3169 3185	3138 3154 3170 3186	3139 3155 3171 3187	3140 3156 3172 3188	3141 3157 3173 3189	3142 3158 3174 3190	3143 3159 3175 3191	3144 3160 3176 3192	3145 3161 3177 3193	3146 3162 3178 3194	3147 3163 3179 3195	3148 3164 3180 3196	3149 3165 3181 3197	3150 3166 3182 3198	3151 3167 3183
C80 C90 CA0	3 3	3200 3216 3232 3248	3201 3217 3233 3249	3202 3218 3234 3250	3203 3219 3235 3251	3204 3220 3236 3252	3205 3221 3237 3253	3206 3222 3238 3254	3207 3223 3239 3255	3208 3224 3240 3256	3209 3225 3241 3257	3210 3226 3242 3258	3211 3227 3243 3259	3212 3228 3244 3260	3213 3229 3245 3261	3214 3230 3246 3262	3215 3231 3247
CC0 CD0 CE0 CF0	3 3	3264 3280 3296 3312	3265 3281 3297 3313	3266 3282 3298 3314	3267 3283 3299 3315	3268 3284 3300 3316	3269 3285 3301 3317	3270 3286 3302 3318	3271 3287 3303 3319	3272 3288 3304 3320	3273 3289 3305 3321	3274 3290 3306 3322	3275 3291 3307 3323	3276 3292 3308 3324	3277 3293 3309 3325	3278 3294 3310 3326	3279 3295 3311
D00 D10 D20 D30	3 3	344 360 376	3329 3345 3361 3377	3330 3346 3362 3378	3331 3347 3363 3379	3332 3348 3364 3380	3333 3349 3365 3381	3334 3350 3366 3382	3335 3351 3367 3383	3336 3352 3368 3384	3337 3353 3369 3385	3338 3354 3370 3386	3339 3355 3371 3387	33 <b>4</b> 0 3356 3372 3388	3341 3357 3373	3342 3358 3374	3343 3359 3375
D40 D50 D60 D70	3,	408 424 440	3409 3425 3441	3394 3410 3426 3442	3395 3411 3427 3443	3396 3412 3428 3444	3397 3413 3429 3445	3398 3414 3430 3446	3399 3415 3431 3447	3400 3416 3432 3448	3401 3417 3433 3449	3402 3418 3434 3450	3403 3419 3435 3451	3404 3420 3436 3452	3405 3421 3437 3453	3406 3422 3438	3407 3423 3439
D80 D90 DA0 DB0	34	472 : 488 : 504 :	3457 3473 3489 3505	3458 3474 3490 3506	3459 3475 3491 3507	3460 3476 3492 3508	3461 3477 3493 3509	3462 3478 3494 3510	3463 3479 3495 3511	3464 3480 3496 3512	3465 3481 3497 3513	3466 3482 3498 3514	3467 3483 3499 3515	3468 3484 3500 3516	3469 3 3485 3 3501 3	3470 : 3486 : 3502 :	3471 3487 3503
DC0 DD0 DE0 DF0	35	536 3 552 3	3521 3537 3553	3522 3538 3554	3523 3539 3555	3524 3540 3556	3525 ( 3541 ( 3557 (	3526 3542	3527 3543	3528 3544	3529 3545	3530 3546	3531 3547	3532 3548	3533 3 3549 3 3565 3 3581 3	3534 3 3550 3	3535 3551

	0	1	2	3	4	5	6	7	8	9	A	В	С	D	Е	F
E00 E10 E20 E30	3600 3616	3601	3618		3604	3621	3622	3591 3607 3623 3639	3608 3624	3625	3610 3626	3611 3627	3628	3597 3613 3629 3645	3598 3614 3630 3646	3599 3615 3631 3647
E40 E50 E60 E70	3664 3680	3649 3665 3681 3697	3682	3651 3667 3683 3699	3684	3653 3669 3685 3701	3670 3686	3655 3671 36 <b>8</b> 7 3703	3672 3688	3673 3689	3674		3676	3661 3677 3693 3709	3662 3678 3694 3710	3663 3679 3695 3711
E80 E90 EA0 EB0	3728 3744	3729 3745	3714 3730 3746 3762	3731 3747	3748	3733 3749	3750	3719 3735 3751 3767	3736 3752	3753	3738 3754	3739 3755		3757	3726 3742 3758 3774	3759
EC0 ED0 EE0 EF0	3792 3808	3793 3809		3795 3811	3796 3812	3797 3813	3798 3814	3783 3799 3815 3831	3800 3816	3801 3817	3802 3818	3803 3819	3804 3820	37 <b>89</b> 3805 3821 3837	3822	
F00 F10 F20 F30	3856 3872	3841 3857 3873 3889		3843 3859 3875 3891	3860 3876	3861 3877	3862 3878	3847 3863 3879 3895	3864 3880	3865 3881	3866		3868	3853 3869 3885 3901		
F40 F50 F60 F70	3920 3936	3905 3921 3937 3953	3922 3938	3923 3939	3924 3940	3925 3941			3928 3944		3930 3946	3947	3932 3948	3917 3933 3949 3965	3918 3934 3950 3966	3919 3935 3951 3967
F80 F90 FA0 FB0	3984	3985 4001	3986 4002	3987	3988 4004	3989 4005	3990 4006	3975 3991 4007 4023	3992 4008	3993 4009	3994 4010	3995 4011	4012	3981 3997 4013 4029	3982 3998 4014 4030	3983 3999 4015 4031
FC0 FD0 FE0 FF0	4048 4064	4049 4065	4034 4050 4066 4082	<b>4</b> 051 <b>4</b> 067	4052 4068	4053 4069	4054 4070	4039 4055 4071 4087	4056 4072	4057 <b>4</b> 073	4058 4074	4059 4075	4060 4076	4061 4077	4046 4062 4078 4094	4079

# APPENDIX B. CHARACTER CODES

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
NUL	00			M.P.	12-0-9-8-1	NUL	80	NUL	00		
SOH	01			M.P.	12-9-1	SOM	81	SOH	01		
STX	02			M.P.	12-9-2	EOA	82	STX	02	770000	
ETX	03			M.P.	12-9-3	ЕОМ	83	ETX	03		
EOT	04			M.P.	9-7	EDT	84	EOT	37		
ENQ	05			M.P.	0-9-8-5	WRU	85	ENQ	2D		
ACK	06			M.P.	0-9-8-6	RU	86	ACK	2E		
BEL	07			M.P.	0-9-8-7	BEL	87	BEL	2F		
BS	08			M.P.	11-9-6	FE	88	BS	16		
HT	09			M.P.	12-9-5	нт	89	нт	05		
$_{ m LF}$	0A			M.P.	0-9-5	LF	8A	LF	25		
VT	0B			M.P.	12-9-8-3	VT	8B	VT	0в		
FF	0C			M.P.	12-9-8-4	FORM	8C	FF	0C		
CR	0D			M.P.	12-9-8-5	RETURN	8D	CR	0D		
S0	0E			M.P.	12-9-8-6	so	8E	S0	0E		
SI	0F			M.P.	12-9-8-7	sı	8F	SI	OF		
DLE	10		İ	M.P.	12-11-9-8-1		90	DLE:	10		
DCl	11			M.P.	11-9-1	X-ON	91	DC1	11		
DC2	12			M.P.	11-9-2	TAPE	92	DC2	12		
DC3	13			M.P.	11-9-3	X-OFF	93	DC3	13		
DC4	14			M.P.	9-8-4	TAPE	94	DC4	3C		
NAK	15			M.P.	9-8-5	ERROR	95	NAK	3D		
SYN	16			M.P.	9-2	SYC	96	SYN	32		
ETB	17			M.P.	0-9-6	LEM	97	ETB	26		
CAN	18		İ	M.P.	11-9-8	so	98	CAN	18		

M.P. = Multi-punch

CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	ТТҮ	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
EM	19			M.P.	11-9-8-1	S1	99	EM	19		
SUB	1A			M.P.	9-8-7	S2	9A	SUB	3F		
ESC	18	İ		M.P.	0-9-7	S3	9B	ESC	27		
FS	1C			M.P.	11-9-8-4	S4	9C	FS	1C		
GS	1D			M.P.	11-9-8-5	S5	9 D	GS	1D		
RS	$1\mathbf{E}$			M.P.	11-9-8-6	s6	9E	RS	1E		
US	1F			M.P.	11-9-8-7	s7	9F	US	1F		
SPACE	20	SPACE	20	SPACE B	AR -	SPACE	A0	SPACE	40	SPACE	0
!	21	!	21	1	12-8-7	!	Al	!	4 F		
"	22	"	22	11	8-7	"	A2	п	7F		
#	23	#	23	#	8-3	#	A3	#	7B		
\$	24	\$	24	\$	11-8-3	\$	A4	\$	5B	\$	39
8	25	ક્ર	25	ક્ર	0-8-4	8	A5	ક	6C		
&	26	&	26	&	12	&	A6	&	50		
•	27	ļ '	27	•	8-5	•	A7	1	<b>7</b> D		
(	28	(	28	(	12-8-5	(	A8	(	4D		
)	29	)	29	)	11-8-5	)	A9	)	5D		
*	2A	*	2A	*	11-8-4	*	AA	*	5C		
+	2B	+	2B	+	12-8-6	+	AB	+	4E		
,	2C	,	2C	,	0-8-3	,	AC	,	6B		
-	2D	-	2D	- ′	11	_	AD	-	60		
•	2E		2E		12-8-3		AE		4B	•	38
/	2F	/	2F	/	0-1	/	AF	/	61		
0	30	0	30	0	0	0	В0	0	F0	0	27
1	31	1	31	1	1	1	Bl	1	Fl	1	28
2	32	2	32	2	2	2	B2	2	F2	2	29
3	33	3	33	3	3	3	В3	3	F3	3	30
4	34	4	34	4	4	4	В4	4	F4	4	31
5	35	5	35	5	5	5	В5	5	F5	5	32
6	36	6	36	6	6	6	В6	6	F6	6	33

CI	HAR. C	CODE	L.P.	CODE	CARD READER	CODE 029	ТТҮ	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
7	3	37	7	37	7	7	7	в7	7	F7	7	34
8	3	88	8	38	8	8	8	в8	3	F8	8	35
9	3	19	9	39	9	9	9	В9	9	F9	9	36
:	3	BA	:	3 <b>A</b>	:	8-2	:	BA		7A	:	37
;	3	ВВ	;	3B	;	11-8-6	;	BB	,	5E		
<	3	C	<	3C	<	12-8-4	<	вс	<	4C		
=	3	D	=	3D	=	8-6	=	BD	:=	7E		
>	3	E	>	3E	>	0-8-6	>	BE	>	6E	.	
?	3	F	?	3F	?	0-8-7	?	BF	7	6F		
@	4	0	@	40	@	8-4	@	C0	ı	7C		
A	4	1	A	41	Α.	12-1	A	C1	A	Cl	A	1
В	4	2	В	42	В	12-2	В	C2	В	C2	В	2
С	4	3	С	43	С	12-3	С	C3	С	C3	C	3
D	4	4	D	44	D	12-4	D	C4	D	C4	D	4
E	4	5	E	45	E	12-5	E	C5	Е	C5	E	5
F	4	6	F	46	F	12-6	F	C6	F	C6	F	6
G	4	7	G	47	G	12-7	G	C7	G	C7	G	7
Н	4	8	Н	48	н	12-8	Н	C8	Н	C8	н	8
I	4	9	I	49	I	12-9	I	C9	1.	C9	ı	9
J	4.	A	J	4A	J	11-1	J	CA	J	Dl	J	10
K	4	В	K	4B	K	11-2	K	СВ	К	D2	K	11
L	4	c	L	4C	L	11-3	L	CC	L	D3	L	12
М	4:	D	M	4D	M	11-4	М	CD	M	D4	м	13
N	4:	E	N	4E	N	11-5	N	CE	N	D5	N	14
0	4:	F	0	4F	0	11-6	0	CF	0	D6	0	15
P	5	0	P	50	P ·	11-7	P	D0	P	D7	P	16
Q	5	1	Q	51	Q	11-8	Q	D1	Ω	D8	Q	17
R	5	2	R	52	R	11-9	R	D2	R	D9	R	18
s	5	3	S	53	s	0-2	S	D3	S	E2	S	19
. <b>T</b>	5	4	T	54	T	0-3	T	D4	T	E3	T	20

	CHAR.	CODE	L.P.	CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
	U	55	Ū	55	Ŭ	0-4	Ū	D5	U	E4	Ū	21
	V	56	V	56	V	0-5	V	D6	V	<b>E</b> 5	v	22
	W	57	W	57	W	0-6	W	D7	W	E6	W	23
	X	58	Х	58	Х	0-7	Х	D8	Х	E7	Х	24
	Y	59	Y	59	Y	0-8	Y	D9	Y	E8	Y	25
	Z	5A	Z	5A	Z	0-9	Z	DA	Z	E9	Z	26
	[	5B	Ţ.	5B	¢	12-8-2	[	DB	¢	<b>4</b> A		
		5C		5C	0-8-2	0-8-2		DC		E0		
	]	5D	]	5D	1	11-8-2	]	DD	1	5A		
	Λ	5E	Λ	5E	<del></del> ,	11-8-7	<b>†</b>	DE	Λ	5F		
		5F		5F		0-8-5	<b>←</b>	DF		6D		
	\	60			M.P.	8-1			\	79		
	a	61			M.P.	12-0-1			a	81		
	b	62			M.P.	12-0-2			b	82		
	С	63			M.P.	12-0-3			С	83		
	đ	64			M.P.	12-0-4			đ	84	•	
	е	65			M.P.	12-0-5			е	85		
	f	66			M.P.	12-0-6			f	86		
	g	67			M.P.	12-0-7			g	87		
	h	68			M.P.	12-0-8			h	88		
	i	69			M.P.	12-0-9			i	89		
	j	6A			М.Р.	12-11-1			j	91		
	k	6B			M.P.	12-11-2			k	92		
	1	6C			M.P.	12-11-3			1	93	ĺ	
	m	6D			M.P.	12-11-4			m	94		
ŀ	n	6E			M.P.	12-11-5	·		n	95		
	0	6F			M.P.	12-11-6			0	96		
	р	70			M.P.	12-11-7			р	97		
	q	71			м.Р.	12-11-8	·		q	98		
	r	72			M.P.	12-11-9			r	99		

M.P. = Multi-punch

CHAR. CODE	L.P. CODE	CARD READER	CODE 029	TTY	CODE	EBCDIC	CODE	COMP. CHAR.	CAN CODE
S 73 t 74 u 75 v 76 w 77 x 78 y 79 z 7A { 7B	L.P. CODE	M.P. M.P. M.P. M.P. M.P. M.P. M.P. M.P.	11-0-2 11-0-3 11-0-4 11-0-5 11-0-6 11-0-7 11-0-8 11-0-9 12-0 12-11 11-0 11-0-1 12-9-7	RUB OUT		EBCDIC  S  U  V  W  X  Z  DEL	A2 A3 A4 A5 A6 A7 A8 A9 C0 6A D0 A1 07	CHAR.	CODE

# APPENDIX C

# MOVING HEAD DISC (CONTROLLER ADDRESS $01_{16}$ )

			· · · · · · · · · · · · · · · · · · ·	·		,	,	<del>,                                      </del>	<del></del>	+							
		0	1	2	3	4	5	6	7	8	9	10	11	12	. 13	14	15
TRANSFER	WRITE	1	М	CONN. DI	CONN. SI	0	EOD	EOF	IGN	scs	IGNO	RED	s	s	S	S	S
INITIATE	READ	1	М	CONN. DI	CONN. SI	1	IGN	ORED		scs	IGNO	RED	s	s	S	S	S
	CYLINDER SELECT	0	1	CONN. DI	CONN. SI	0	0	1	*C	С	С	С	С	С	С	С	С
CONTROL	TERM/ EOB	0	1	IGNO	ORED	EOB	TERM.	IGN	MPE		4	<u> </u>	IGNO	RED	<u> </u>	L	<u> </u>
	HEAD/ DRIVE SELECT	0	0	IGN	*HEAD SEL	*HEAD SEL	*HEAD SEL	*HEAD SEL	HEAD SEL	CONTIN- UOUS SCAN		IGNO	RED		1= PREP MODE	U	Ū
	NO-OP	. 0	1	CONN. DI	CONN. SI	0	0	0				1	GNORED		1.000		
	STATUS	0= ERROR	O/F U/F	CRC	IN OP	MPE	WLO	SEEK ERROR	l= BUSY	0= DATA READY	EOD	EOF	EOR	DEV. SEEK	SEEK COMP.	Ū	ŭ

S = SECTOR, C = CYLINDER, DMP LOCATIONS: TC = 61, TA = 71

# FIXED HEAD DISC (CONTROLLER ADDRESS $02_{16}$ )

											10						
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	WRITE	1	М	CONN. DI	CONN. SI	0	EOD	EOF		I	GNORED	I	S	s	S	S	s
	READ	1	М	CONN. DI	CONN.	1			IGNO	ORED	·		S	s	S	S	s
	HEAD SELECT	0	1	CONN. DI	CONN. SI	0	0	1	IGN	Н	Н	Н	Н	Н	Н	Н	Н
CONTROL	TERM/ EOB	0	1	IGNO	RED	EOB	TERM.	IGN.	1 IGN H H H H H					<u></u>	l		
	UNIT* SELECT	0	0				<u> </u>	L	IGNO	RED						Ū	U
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	ø					IGNOR	ED			<u> </u>
	STATUS	0= ERROR	O/F U/F	CK SUM	IN OP	MPE	WLO	ø	l= BUSY	0= DATA READY	EOD	EOF	EOR	0	0	0	0

S = SECTOR, H = HEAD, DMP LOCATIONS: TC = 62, TA = 72

 $<sup>{\</sup>tt U} = {\tt UNIT} \ {\tt NO.}$ ,  ${\tt UP} \ {\tt TO} \ {\tt 4} \ {\tt UNITS} \ {\tt MAY} \ {\tt BE} \ {\tt CONNECTED} \ {\tt TO} \ {\tt A} \ {\tt CONTROLLER}$ 

M = MODE, IF BIT 1 = 0 PROGRAMMED I/O IS SELECTED RATHER THAN DMP

<sup>\*</sup> THESE BITS ARE USED ONLY IN DISC PACK MOVING HEAD DISC CONTROLLERS.

U = UNIT NO., UP TO 4 UNITS MAY BE CONNECTED TO A CONTROLLER

M = MODE, IF BIT 1 = 0 PROGRAMMED I/O IS SELECTED RATHER THAN DMP

<sup>\*</sup> THIS COMMAND NOT USED WITH MODEL 4103, 4104, 4105.

# MAGNETIC TAPE (DEVICE ADDRESS, HIGH SPEED $03_{16}$ , LOW SPEED $04_{16}$ )

	-				— т												15
	ļ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	WRITE	1	М	CONN. DI	CONN. SI	0	0= BINARY 1= INTER- CHANGE	0= N-GAP 1= L-GAP	CYCLE	0= ODD 1= EVEN	0= 800 CPI 1= 556 CPI	0	0	0	,0	υ	υ
TRANSFER INITIATE	READ	1	М	CONN. DI	CONN. SI	1	0= BINARY 1= INTER- CHANGE	0	1=SCS	0= ODD 1= EVEN	0= 800 CPI 1= 556 CPI	0	0	0	0	U	U
	WRITE EOF	0	1	CONN.	CONN. SI	0	0	1	1=SCS	0	0= 800 CPI 1= 556 CPI	WRITE EOF 1	0	0	0	Ū	Ü
	SPACE	0	1	CONN. DI	CONN. SI	0	0	1	1=SCS	0	0= 800 CPI 1= 556 CPI	"	SPACE 1	0= FORWARD 1= REVERSE	1=	U	U
CONTROL	EOB/ TERM	0	1	IGN	IGN	1=EOB	1=TERM	IGN	1=MPE				IGNO	RED			
	REWIND	0	1	CONN.	CONN. SI	0	0	1	1= LOCK OUT	REWIND	0	0	0	0	0	U	U
	TRANSPORT	0	1	CONN. DI	CONN. SI	0	0	1	1= CONT. SCAN	0	0	0	0	0	0	U	U
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0					IGNO	ORED			
STATUS	1	0= ERROR	l= OVER/ FLOW OR B>C	l= DEVICE PARITY ERROR		l= MEMOR' PARIT' ERROR	PROTECT	l= TAPE DETEC	1= CONT BUSY	0= DATA READY	l= EOT	l= EOF	l= BOT	1= DEVICE OFFLINE OR REWIND		Ū	U

## CARD READER (DEVICE ADDRESS 0516)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	1	1=BIN 0=XLATE					IGN	ORED		·····		
	EOB/ TERM	0	1	IGNO	RED	EOB	TERM.										
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0					IGNO	ORED			
STATUS		0= ERROR	1= OVER- FLOW	1= READ CHECK ERROR	l= IN OP	0	0	0	l= BUSY	0= DATA BUFFER READY	l= HOPPER EMPTY/ STACKER FULL	0	0	l= HOLD	I= PICK FAIL	0	0

# CARD PUNCH (DEVICE ADDRESS 06<sub>16</sub>)

		0	1	2	3 .	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI	0		B EE BLE*	l= OFFSET STACK			L	IGN	ORED	<u> </u>		<b>L</b>
CONTROL	EOB/ TERM	0	1	IGNO	DRED	EOB	TERM.			IGNORED							
	NO-OP	0	1	CONN. DI	CONN SI	0	0	0				IGNORED					
STATUS		0= ERROR	l= UNDER- FLOW	1= PUNCH ERROR		0	0	0	l= CONT. BUSY		l= HOPPER EMPTY/ STACKER FULL	0	0	0	1= TRANS- PORT	0	0

*AB	OUTPUT TRANSLATION MODES
11	ILLEGAL
01	XLATE, FROM ASCII (7 BIT) TO HOLLERITH
10	12 BIT BINARY (1 TO 1)
00	8 BIT BINARY EXPANDED TO 12 COL. PUNCH

# LINE PRINTER 600 LPM (DEVICE ADDRESS $07_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFEI		1	0	CONN. DI	CONN. SI	0		IGNOF	RED		1=VFU 0=L.CNT	LINE	LINE COUNT	COUNT		LINE COUNT OR VFU	LINE COUNT OR VFU
CONTROL	EOB/ TERM	0	1	IGN	IORED	EOB	TERM.	1=LINE FEED	Ι	GN	1=VFU 0=L.CNT	LINE COUNT	LINE	COUNT	LINE COUNT OR VFU	LINE COUNT OR VFU	LINE COUNT OR VFU
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	D			<u> </u>	IG	NORED				
STATUS		0= ERROR	0	0	l= IN OP	0	0	0	l= BUSY	0= DATA BUFFER READY		1= BOTTOM OF FORM	0	l= HOLD	0	0	0

## ELECTROSTATIC PRINTER/PLOTTER (DEVICE ADDRESS 0816)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	М	CONN. DI	CONN. SI	0	*PLOT	**SPP		1GNORED							
	FOB/ TERM	Û	1	1 GN	ORED	ЕОВ	TERM	1GN	MPE	IGNORED							
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0					IGNORED•				
STATUS		0= ERROR	0	0	INOP	MPE	0	0	l= BUSY	0= DATA READY	PAPER LOW	0	0	0	0	0	0

# X-Y PLOTTER (DEVICE ADDRESS $08_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
TRANSFER INITIATE		1	0	CONN.	CONN. SI	0						IGNORED				·			
	EOB/ TERM	0	1	IGN	ORED	l= EOB	l= TERM												
CONTROL	OUTPUT DATA		IGNORED										1=PEN UP	1=DRUM DOWN	UP	1=CARR. RIGHT	1=CARR. LEFT		
	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0				I	IGNORED						
STATUS		0= ERROR	0	0	l= IN OP	0	0	0	l= BUSY	0= DATA	0	0	0	0	0	0	0		

\*\*MUTUALLY EXCLUSIVE

# PAPER TAPE PUNCH (DEVICE ADDRESS $09_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER 1 0 CONN. CONN. O DI SI							IGNORED										
CONTROL	EOB/ TERM	0	1	IGN	ORED	ЕОВ	TERM.	IGN	l= ABORT	IGNORED							
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0				IGNO	RED			,	
STATUS		1	0	0	0	0	0	0	l= BUSY	0= DATA READY	1= TAPE LOW	0	0	0	0	0	0

# CONSOLE TTY/PAPER TAPE READER (DEVICE ADDRESS $0A_{16}$ )

				l	Γ		T	г			<del></del>		,					
		0	1	2 .	3	4	5	6	7	8	9	10	11	12	13	14	15	
TRANSFER INITIATE	,	1	0	CONN. DI	CONN. SI	l=IN 0=OUT	l= KEY BD.	ENABLE CLOCK						0=REV.* 1=FOR.	:	IGNORED		
CONTROL	EOB/ TERM	0	1	IGN	ORED	0	TERM	IGN	l=ABORT	IGNORED								
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0				IGNO	RED					
STATUS		1	0	0	0	0	0	0	1= BUSY	0= DATA READY	0	0	0	0	0	0	0	

<sup>\*</sup>NOTE: HIGH SPEED READER NORMALLY OPERATES IN REVERSE MODE.

# LINE PRINTER 50-150 LPM (DEVICE ADDRESS $0B_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
TRANSFER 1 0 CONN. CONN. 1NITIATE 1 0 IGNORED										l	<u> </u>	<u></u>							
CONTROL	EOB/ TERM	0	1	I	GNORED	EOB	TERM	IGNORED											
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	0 IGNORED										
STATUS		0= ERROR	0	0	l= IN OP	0	0	0	l= CONT. BUSY	0= DATA READY	1= LOW PAPER	1= BOTTOM OF FORM	0	1= HOLD	0	0	0		

#### HIGH LEVEL AIS OUTPUT UNIT (DEVICE ADDRESS $10_{16}$ )

													<del></del>				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	м	CONN. DI	IGN	1=SEQ 0=RAN		IGN	ORED		*END	ADDRES	S				
<del></del>	EOB/ TERM	ū	1	IGN	IORED	EOB	TERM	IGN	ORED		*ST#	AT ADDR	ESS				
CONTROL	NO-OP	0	1	CONN.	IGN	0	0	0				IGNO	RED —				
STATUS		ERROR	-		IGNORE.	D			l= BUSY	0= DATA READY	-			IGNORED			
DATA WORD		-				- IGNOREI	)——						* CHAN	NEL ADD	RESS		

<sup>\*</sup> ADDRESSES ARE IN BINARY (0-127)

#### HIGH LEVEL AIS INPUT UNIT (DEVICE ADDRESS $11_{16}$ )

								r					T	l			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	М	CONN.	CONN. SI						IGNORE	D					
	EOB/ TERM	0	1	IGN	IORED	EOB	TERM					I	GNORED				
CONTROL	NO-OP	0	1	CONN.	CONN.	0	0	0						IGNORED			
STATUS	I	0= ERROR			IGN	ORED			l= BUSY	0= DATA READY				IGNORED			
DATA WORL	)	SIGN	1	0	1	1	2 <sup>10</sup>	29	28	2 7	26	DATA 5	24	23	22	21	] 2 <sup>0</sup>

### WIDE RANGE SOLID STATE AIS OUTPUT UNIT (DEVICE ADDRESS $12_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	М	CONN. DI	IGN	ORED	l= RE- CYCLE	-					- IGNO	RED -			
	EOB/ TERM	0	1	IGNO	RED	EOB	TERM	-					IGNOI	RED ——			<del></del>
CONTROL	NO-OP	0	1	CONN.	IGN	0	0	0	-				IGNO	RED —			
STATUS		ERROR		IG	NORED				1= BUSY	0= DATA READY			IGNO				
DATA WORD				G	AIN		1=	1=					*	CHANNE	L ADDRES	SS (0-12	7)2
1		IGN	23	22	21	20	ZERO SUPPR.	AUTO	IGNO	RED	26	2 <sup>5</sup>	2 <sup>4</sup>	23	22	21	20
SUPPRESSIC WORD	ИС	SIGN	IGN	2 <sup>13</sup>	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	29	28	27	26	2 <sup>5</sup>	24	23	22	21	20

<sup>\*</sup> ADDRESSES ARE IN BINARY (0-127)

#### WIDE RANGE SOLID STATE AIS INPUT UNIT (DEVICE ADDRESS $13_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	М	CONN. DI	CONN. SI	-					IGNORED						
	EOB/ TERM	0	1	IGNC	RED	ЕОВ	TERM	-					IGNORED				
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0	-					IGNORED•		-	
STATUS ERROR IGNORED BUSY DATA READY IGNORED																	
DATA WORD	)	SIGN	23	2 <sup>2</sup>	AIN 2 <sup>1</sup>	20	2 <sup>10</sup>	29	28	27	2 <sup>6</sup>	2 <sup>5</sup>	24	23	22	21	2

#### WIDE RANGE RELAY AIS (DEVICE ADDRESS $14_{16}$ )

		0	1	2	3	4,	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE		1	0	CONN. DI	CONN. SI						IGNO:	RED					
	EOB/ TERM	0	1	IGNO	ORED	EOB	TERM					I	GNORED				
CONTROL	NO-OP	0	1	CONN. DI	CONN. SI	0	0	0						IGNO:	RED		
STATUS		ERROR			IGI	NORED			BUSY	INPUT DATA READY			IGNO	RED			OUTPUT DATA READY
DATA WORD		1=RES. CHECK MODE		GA	IN		l= ZERO SUPPR.	l= AUTO				* CH	ANNEL A	DDRESS			
SUPPRESSI WORD	ON	SIGN	IGN.	2 <sup>13</sup>	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	29	28	27	26	25	24	23	22	21	20

<sup>\*</sup> ADDRESSES ARE IN BINARY (0-511)

SUBSYSTEM 1 (DEVICE ADD.  $20_{16}$  – $2B_{16}$ ) MODAC SUBSYSTEM 2 (DEVICE ADD.  $30_{16}$  – $3B_{16}$ )

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TRANSFER INITIATE	AIM	1	М	CONN. DI	CONN. SI	l= SEQ	IGN	24	2 <sup>3</sup>	ND ADDRE 2 <sup>2</sup>	21	20	24	2 <sup>3</sup> ST	ART ADDI	RESS 2	20
	EOB/TERM AIM	0	1	IGNO		EOB	TERM	-				IGN	ORED -				
CONTROL	NO-OP AIM	0	1	CONN. DI	CONN. SI	0	0	0	-				IGNO				<del></del>
	INTERRUPT EXTENDER	0	1	CONN. DI	CONN. SI	IGN	IORED		R	0	1	2	NTERRUP'	CHANN	EL 5	6	7
	AIM	ERROR	-		IGN	ORED -	•		BUSY	0=IN DATA RDY 1=XFER	4			IGNORE	D	<b>→</b> DA	O=OUT ATA RDY
STATUS	SYNCHRONIZER INTERRUPT	-	<b></b>		IGN DATA	ORED -			<b>==</b>	REQ	•	2	SERVI	IGNORE CE INTE	RRUPT	1 6	==
I P No. 16, wines for all the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the s	EXTENDER OUTPUT AIM	0	1	2	3	4	5 IGNORED	6	/ 1	0	1		24	23	CHANNEL 2 <sup>2</sup>	21	20
	INPUT AIM	-		SIGN			2 <sup>10</sup>	29	28	27	— DATA 2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	23	22	21	20
	TAGE)	SIGN	ADDI 2 <sup>1</sup>	20	CRT CONT.	CRT CONT.	2 <sup>10</sup>	29	28	27	DATA 6	25	2 <b>4</b>	23	22	21	20
	ANALOG OUT- PUT (CUR- RENT)	IGN	ADDI 2 <sup>1</sup>	ress 2 <sup>0</sup>	IGN	2 <sup>11</sup>	2 <sup>10</sup>	29	28	27	2 <sup>6</sup>	2 <sup>5</sup>	24	23	22	21	20

### I/UIS#1 (DEVICE ADDRESS 20-2F<sub>16</sub>) I/OIS#2 (DEVICE ADDRESS 30-3F<sub>16</sub>)

_		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1
CONTROL	I/O INTERRUPT COUPLER	0	1	CONN.	CONN. SI		IGNOREI	)	1=RES REQ.	0	1 1	1 2	INTERRUI	T CHAN	NEL	1 6	<del> </del>
	INTERVAL TIMER	0	1				IGNO	RED			0=SINGLE 1=RECYCLE			-IGNO	RED-		
	CHANNEL MULTIPLEXER	0	1	<b>—</b>			IGNO	RED —				EXPA ADDR	NDER		• CHAN	NEL ADD	RESS
	SACI	0	1		DRED	0	т	!←			<del></del>	IGNO	RED				
	SACI NO-OP	0_	1	CONN.	CONN.	0	0	0					IGNO	RED			
TRANSFER INITIATE	SACI	1	0	CONN.	CONN. SI	l= IN	l= KEY BD.	-				-IGNO	RED				
	SACI	ERROR	IGNO	RED	l=DATA SET NOT READY		IGN	ORED	BUSY	DATA READY			IGNOF	RED			
STATUS	SYNCHRONIZER	SYS. CONN.	-		1G1	NORED -	•			l= XFER REQ.			IGNO	RED			
*	SUBSYSTEM	SYS CONN.															
	ANALOG OUTPUT	SIGN	0=CH1 1=CH2	CRT CONT.	CRT CONT.	CRT CONT.	210	29	28	27	2 <sup>6</sup> DA'	TA 5	2 <sup>4</sup>	23	2 <sup>2</sup>	2 <sup>1</sup>	2
DATA	SACI				IGNO	RED				<del></del>			ASCII	DATA	·	<u> </u>	

### APPENDIX D. DIVIDE

During execution of a divide instruction, the contents of registers Ra, RaVl (where a is an even number  $\neq 0$ ) form a double precision dividend and are divided by the single-precision divisor specified by the instruction. If the dividend is a single-precision number, RaVl should be cleared prior to executing the divide instruction or erroneous results may occur. Although a double-length dividend is used, divide is a single-precision operation and should not be confused with a double-precision divide operation that would use a double-length divisor and would produce a double-length quotient.

After execution of the divide, the single-precision quotient replaces the contents of RaVl and the remaining portion of the dividend that has not been divided (undivided remainder) replaces the contents of Ra. The quotient is signed in accordance with algebraic convention, that is, positive if dividend and divisor signs are alike, but negative otherwise. However, only 15 magnitude bits are generated by the divide and, if the magnitude of the quotient is so small as to require more than 15 bits to resolve, a zero quotient may be generated regardless of the required sign, but the remainder will still reflect the undivided portion of the original dividend. The binary scaling of the quotient is equal to the dividend scale factor minus the divisor scale factor.

The undivided remainder replaces the contents of Ra and has the same sign as the dividend. It has the same scaling as the divisor. By definition, the undivided remainder is that quantity which must be added to the product of the divisor and the quotient to produce the original dividend. The results of the divide instruction are consistent with this definition. It should be noted that the remainder must be added to the least significant part of the product of the divisor and the quotient to maintain proper scaling. Overflow is possible and the Overflow Indicator will be set if:

$$\left| \frac{(Ra, RaVl)}{(M)} \right| \geq 1$$
 Reference section on Overflow (Page 3-9)

EXAMPLE: Let 
$$(Ra, RaV1) = 0004$$
 E800  $(M) = 3C00$ 

The dividend can be represented as a decimal 628 with an equivalent binary point at  $2^{21}$ . The divisor may be represented as a decimal 30 with its binary point at  $2^6$ . The resulting binary scaling of the quotient is  $2^{21}-2^6=2^{15}$ . The remainder is scaled at  $2^6$ .

$$Q = 0014 (20_{10} \text{ at } 2^{15})$$
  $R = 3800 (28_{10} \text{ at } 2^6)$ 

## APPENDIX E. INSTRUCTION LIST

MNEMO	NIC OF	. CODE	NAME	EXECTION	
LOAD,	STORE AND	TRANSFER	•	TIME (µs)	PAGE
LDM	E5	Load	Register from Memory	2.4	3-4
LDI	ED	Load	Register from Memory Immediate	1.6	3-4
LDS	F5	Load	Register from Memory Short Displace	d 1.87	3-4
LDX	FD	Load	Register from Memory Short Indexed	1.87	3-5
STM	E6	Store	Register in Memory	2.4	3-5
STI	EE	Store	Register in Memory Immediate	1.6	3 <b>-</b> 5
STS	F6	Store	Register in Memory Short Displaced	1.87	3-5
STX	FE	Store	Register in Memory Short Indexed	1.87	3-6
LBX	AE	Load	Byte From Memory	2.13	3-6
SBX	AF	Store	Byte in Memory	3.2	3-7
LFM	A4	Load	File from Memory	40 + .8 (R-1)	3-7
LFS	B4	Load	File from Memory Short Displaced	3.47 + .8 (R-1)	3-7
LFX	BC	Load	File from Memory Short Indexed	3.47 + .8 (R-1)	3-8
SFM	A5	Store	File in Memory	40 + .8 (R-1)	3-8
SFS	В5	Store	File in Memory Short Displaced	3.47 + .8 (R-1)	3-8
SFX	BD	Store	File in Memory Short Indexed	3.47 + .8 (R-1)	3-8
TRR	6D		fer Register to Register	0.8	3-9
TRRB	<b>7</b> D	Trans	fer Register to Register and Branch	1.6	3-9
		if	Nonzero		
ARITHM	ETIC				
ADM	FIO	232 14			
ADI	E0		emory to Register	2.4	3-11
ADS	E8 F0		emory to Register Immediate	1.6	3-11
ADX	F0 F8		emory to Register Short Displaced	1.87	3-11
ADMM	co		emory to Register Short Indexed	1.87	3-12
ADMB	C4		egister to Memory	3.47	3-12
ADSM	D0		egister to Memory and Branch if Nonz	ero 4.27	3-12
ADSB	D0		egister to Memory Short Displaced	2.67	3-13
		Bran	egister to Memory Short Displaced an ach if Nonzero	d 3.47	3-13
ADXM	D8	Add Re	egister to Memory Short Indexed	2.67	3-13
ADXB	DC	Add Re Bran	egister to Memory Short Indexed and ach if Nonzero	3 • 47	3-14
ADR	68	Add Re	gister to Register	0.8	3-14
ADRB	78	Add Re Nonz	gister to Register and Branch if ero	1.6	3-14

MNEMONIC	OP. CO	DDE NAME	EXECUTION	
ARITHMETIC	(CONTINU	JED)	TIME (us)	PAGE
DAR	22	Double Precision Add Register to Register	2.13	3-15
SUM	El	Subtract Memory from Register	2.4	3-15
SUI	E9	Subtract Memory from Register Immediate	1.6	3-15
SUS	Fl	Subtract Memory from Register Short Displac	ed 1.87	3-16
SUX	F9	Subtract Memory from Register Short Indexed	1.87	3-16
SUR	69	Subtract Register from Register	8.0	3-16
SURB	79	Subtract Register from Register and Branch if Nonzero	1.6	3-16
MPM	A0	Multiply Memory by Register	7.74	3-17
MPS	В0	Multiply Memory by Register Short Displaced	7.21	3-17
MPX	В8	Multiply Memory by Register Short Indexed	7.21	3-17
MPR	20	Multiply Register by Register	6.67	3-18
DVM	Al	Divide Register by Memory	12.2	3-18
DVS	Bl	Divide Register by Memory Short Displaced	11.4	3-18
DVX	В9	Divide Register by Memory Short Indexed	11.4	3-19
DVR	21	Divide Register by Register	11.0	3-19
CRMB	C7	Compare Memory and Register	4.53	3-19
CRSB	D7	Compare Memory and Register Short Displaced	4.0	3-20
CRXB	DF	Compare Memory and Register Short Indexed	4.0	3 – 20
TRO	0E	Transfer and Reset Overflow Status	0.8	3- 20
TTR	6F	Transfer Two's Complement Register to Register	0.8	3-21
TTRB	7F	Transfer Two's Complement Register to Register and Branch if Nonzero	1.6	3-21
LOGICAL				
ETM	E2	Extract Memory from Register	2.4	3-22
ETI	EA	Extract Memory from Register Immediate	1.6	3-22
ETS	F2	Extract Memory from Register Short Displace	d 1.87	3-22
ETX	FA	Extract Memory from Register Short Indexed	1.87	3-23
ETMM	Cl	Extract Register from Memory	3.47	3-23
ETMB	C5	Extract Register from Memory and Branch if Nonzero	4.27	3-23
ETSM	Dl	Extract Register from Memory Short Displace	d 2.67	3- 24
ETSB	D5	Extract Register from Memory Short Displace and Branch if Nonzero	ed 3.47	3- 24
ETXM	D9	Extract Register from Memory Short Indexed	2.67	3-24
ETXB	DD	Extract Register from Memory Short Indexed and Branch if Nonzero	3.47	3- 25
ETR	6A	Extract Register from Register	0.8	²- 25
ETRB	7A	Extract Register from Register and Branch if Nonzero	1.6	3- 25
ORM	E3	OR Memory and Register	2.4	3- 26
ORI	EB	OR Memory and Register Immediate	1.6	3- 26
ORS	F3	OR Memory and Register Short Displaced	1.87	3- 26
ORX	FB	OR Memory and Register Short Indexed	1.87	3- 26

MNEMONIC	OP. CC	DDE NAME	EXECUTION	
LOGICAL (COM	TINUED)	-	TIME (us)	PAGE
ORMM	C2	OR Register and Memory	3.47	3-27
ORSM	D2	OR Register and Memory Short Displaced	2.67	3-27
ORXM	DA	OR Register and Memory Short Indexed	2.67	3-27
ORR	6B	OR Register and Register	0.8	3-27
ORRB	7B	OR Register and Register and Branch if Nonzero	1.6	3-28
XOM	E4	Exclusive OR Memory and Register	2.4	3-28
XOI	EC	Exclusive OR Memory and Register Immediate	1.6	3-28
XOS	F4	Exclusive OR Memory and Register Short Displaced	1.87	3-29
XOX	FC	Exclusive OR Memory and Register Short Indexed	1.87	3-29
XOR	6C	Exclusive OR Register and Register	0.8	3-29
XORB	7C	Exclusive OR Register and Register and Branif Nonzero	nch 1.6	3-29
TOR	0D	Transfer One's Complement Register to Register	0.8	3-30
TRMB	C6	Test Register and Memory and Branch if Any Ones Compare	3.73	3-30
TRSB	D6	Test Register and Memory Short Displaced and Branch if Any Ones Compare	2.99	3-30
TRXB	DE	Test Register and Memory Short Indexed and Branch if Any Ones Compare	2.93	3-31
TERB	7E	Test Register and Register and Branch if Any Ones Compare	1.6	3-31
FLOATING POI	NT			
FAR	30	Floating Add Register to Register	15.0	3-34
FSR	31	Floating Subtract Register from Register	15.0	3-35
FMR	32	Floating Multiply Register by Register	12.5	3-35
FDR	33	Floating Divide Register by Register	13.0	3-35
FARD	34	Floating Add Register to Register Double	20.5	3-36
FSRD	35	Floating Subtract Register from Register Double	20.5	3-36
FMRD	36	Floating Multiply Register by Register Double	16.0	3-37
FDRD	37	Floating Divide Register by Register Double	16.5	3-37
FAM	38	Floating Add Memory to Register	17.5	3-37
FSM	39	Floating Subtract Memory from Register	17.5	3-38
FMM	3A	Floating Multiply Memory by Register	14.5	3-38
FDM	3B	Floating Divide Memory into Register	15.5	3-39
FAMD	3C	Floating Add Memory to Register Double	22.5	3-39
FSMD	3D	Floating Subtract Memory from Register Double	22.5	3-39
FMMD	3E	Floating Multiply Memory by Register Double	18.0	3-40
FDMD	3F	Floating Divide Memory into Register Double	19.0	3-40

MNEMONIC	OP. COL	DE: NAME ~-	CUTION ME (µs)	PAGE
SHIFT				
LAD :	2E	Shift Left Arithmetic Double 1.87	+ 267(N-1)	3-41
RAD /	2A	Shift Right Affenders	+ 267(N-1)	3-41 3-42
LAS L	2F	Shift Left Affenders Dange	+ 267(N-1)	3-42
RAS & A	2B	Shift Right Allemmette Danger	+ 267(N-1)	3-42
LLD	2C	Shirt Bere Zegrena	+ 267(N-1)	3-42
RLD F	28	Sillic Right Deglera	+ 267(N-1)	3-42
LLS L	2D	Shirt here bogroup	+ 267(N-1)	3-43
RLS *	29	Shift Right Logical Single 1.6	+ 267(N-1)	3-43
LRS L L	OF	Left Rotate Single	0.8	2-42
BIT MANIPUL	ATION			
T.BR L	65	Load Bit in Register	0.8	3-44
LBRB	75	Load Bit in Register and Branch Unconditionally	1.6	3-44
ABMM ANG	80	Add Bit in Memory	3.47	3-45
ABMB A		Add Bit in Memory and Branch if Nonzero	4.27	3-45
ABSM ABSM		Add Bit in Memory Short Displaced	2.67	3-45
ABSB /	94	Add Bit in Memory Short Displaced and Branch if Nonzero	3.74	3-46
ABXM	- 98	Add Bit in Memory Short Indexed	2.67	3-46
ABXB (No.7)		Add Bit in Memory Short Indexed and Branch	3.74	3-46
ADAD		if Nonzero	0.8	3-47
ABR A	60	Add Bit in Register Add Bit in Register and Branch if Nonzero	1.6	3-47
ABRB A			0.8	3-47
SBR	61	Subtract Bit in Register Subtract Bit in Register and Branch if	1.6	3-48
SBRB	71	Nonzero	3.47	3-48
ZBMM 28H, El	<b>%</b> 81	Zero Bit in Memory	4.8	3-48
ZBMB	P#1 85	Zero Bit in Memory and Branch if Nonzero	2.67	3-49
ZBSM 27 MH E		Zero Bit in Memory Short Displaced	4.27	3-49
ZBSB FF & TO	√ ± <b>9</b> 5	Zero Bit in Memory Short Displaced and Branch if Nonzero	2.93	3-49
ZBXM - EF-14 (	in 99	Zero Bit in Memory Short Indexed		3-49
ZBXB@thtvf	1 Mz 1 <b>9D</b>	Zero Bit in Memory Short Indexed and Branch if Nonzero	0.8	3-50
ZBR 👬 🐇	62	Zero Bit in Register	1.6	
ZBRB 🚁 : .	72	Zero Bit in Register and Branch if Nonzero	3.47	3-50
OBMM OR	h 82	OR Bit in Memory	2.67	3-50
OBSM ORE	ng 92	OR Bit in Memory Short Displaced	2.94	3-50 3-51
OBXM P	9A	OR Bit in Memory Short Indexed	0.8	3-51
OBR Sp. 1		OR Bit in Register		3-51
OBRB GARA	73	OR Bit in Register and Branch Unconditional	0.8	3-51
XBR		Exclusive OR Bit in Register Exclusive OR Bit in Register and Branch if		3-52
XBRB	74	Nonzero	3.47	3-52
TBMB	86	Test Bit in Memory and Branch if One	3.47	3-52
TBSB T	96	Test Bit in Memory Short Displaced and Branch if One	3.2	3-53
TBXB TO	9E	Test Bit in Memory Short Indexed and Branch if One	1.6	3-53
TBRB 7	<del>-</del> 76	Test Bit in Register and Branch if One	4.27	3-53
CBMB (	<i>i;</i>	Compare Bit and Memory	3.7	3-54
CBSB	97	Compare Bit and Memory Short Displaced	3.7 3.7	3-54
CBXB	9F	Compare Bit and Memory Short Indexed	0.8	3-55
GMR (=	67	Generate Mask in Register	1.6	3-55
GMRB	77	Generate Mask in Register and Branch Unconditionally		

MNE	MONIC	OP. O	CODE NAME	EXECUTION	PAGE
BYTE	MANIPU	ULATION		TIME (µs)	
MUR	fact.	0В	Move Upper Byte Register to Register	0.8	3-56
MLR	111.1	0C	Move Lower Byte Register to Register	0.8	3-56
MBR	H = 1.	08	Move Byte Right Register to Register	0.8	3-56
MBL	. 4 ,*	09	Move Byte Left Register to Register	0.8	3-57
IBR		0A	Interchange Bytes Register to Register	0.8	3-57
UNCC	NDITION	IAL BRAN	СН		
BLM	7 .	—— ——— Е7		1.6	3 50
BLI	ĹP	EF	Branch and Link Immediate	0.8	3-58
BRU	2	E7	Branch Unconditionally		3-58
HOP	j.	F7	Branch Short Displaced	1.6	3-58
BRX		FF	Branch Short Indexed	1.07 1.07	3-59 3-59
HLT	He	00	Halt	1.07	
NOP	N. T	66	No Operation	0.8	3−60 3−6 <b>£</b> )
1101	1 5 1	•	no operation	0.0	3-0 <b>1</b> 2
CONT	ROL				
SPR		02-0	Set Protect Register	0.8	3-60
SGP		02-8	Set Global Protect Register	0.8	3-61
SLP		03	Set Lower Protect Register	0.8	3-61
SUP		04	Set Upper Protect Register	0.8	3-61
INTE	RRUPT A	ND CALL			
SIE	5.1	26-4	Set Interrupt Enable	1.2	3-61
RIE	s it	27-4	Reset Interrupt Enable	1.33	3-61
SIR	100	26-8	Set Interrupt Request	1.33	3-6 <b>2</b>
RIR	р., ч. В.	27-8	Reset Interrupt Request	1.33	3-62
SIA	* <i>p</i>	26-0	Set Interrupt Active	1.33	3-62
RIA	$\sim 7$	27-0	Reset Interrupt Active	1.33	3-62
REX	$r \cdot \star$	23	Request Executive Service	1.07	3-62
RMI		01	Request Multiprocessor Interrupt	0.8	3-63
CAR	. 600	24	Clear Active and Return	1.87	3-63
CIR	y . N	25	Clear Interrupt and Return	1.87	3-63
INPU'	r/OUTPU'	<u>r</u>			
ISA	<i>i</i> 5	48	Input Status from I/O Group A	1.6	3-64
ISB		49	Input Status from I/O Group B	1.6	3-64
ISC		4A	Input Status from I/O Group C	1.6	3-64
ISD		4B	Input Status from I/O Group D	1.6	3-64
IDA		4C	Input Data from I/O Group A	1.6	3-65
IDB		4D	Input Data from I/O Group B	1.6	3-65
IDC		4E	Input Data from I/O Group C	1.6	3-65
IDD		4F	Input Data from I/O Group D	1.6	3-65
OCA		40	Output Command to I/O Group A	1.33	3-65
OCB		41	Output Command to I/O Group B	1.33	3-65
occ		42	Output Command to I/O Group C	1.33	3-65
OCD		43	Output Command to I/O Group D	1.33	3-65
ODA		44	Output Data to I/O Group A	1.33	3-66
ODB		45	Output Data to I/O Group B	1.33	3-66
ODC		46	Output Data to I/O Group C	1.33	3-66
ODD		47	Output Data to I/O Group D	1.33	3-66
			/		- 50

# APPENDIX F. TABLE OF POWERS OF TWO AND SIXTEEN

		16 <sup>k</sup>													
		2 <sup>n</sup>	n	k	2 <sup>-n</sup>										
		1	0	0	1.0										
		2	1		0.5										
		4	2		0.25										
		8	3		0.12	5									
		16	4	1	0.06	2 5									
		32	5		0.03	1 25									
		64	6		0.01	5 62	5								
		128	7		0.00	7 81:	2 5								
		256	8	2	0.003	3 906	5 25								
		512	9		0.00	L 953	3 125	5							
	1	024	10		0.000	976	562	2 5							
	2	048	11		0.000	488	281	. 25							
	4	096	12	3	0.000	244	140	625	5						
	8	192	13		0.000	122	070	312	2 5						
		384	14		0.000	061	035	156	25						
	32	768	15		0.000	030	517	578	125	;					
		536	16	4	0.000	015	258	789	062	 : 5					
		072	17		0.000	007	629	394	531	25					
		144	18		0.000	003	814	697	265	625					
_	524	288	19		0.000	001	907	348	632	812	5				
	048		20	5	0.000	000	953	674	316	406	25				
	097		21		0.000	000	476	837	158	203	125				
	194		22		0.000	000	238	418	579	101	562	5			
8	388	608	23		0.000	000	119	209	289	550	781	25			
	777		24	6	0.000	000	059	604	664	775	390	625			
	554		25		0.000	000	029	802	322	387	695	312	5		
	108		26		0.000	000	014	901	161	193	847	656	25		
134	217	728	27		0.000										
	435		28	7	0.000	000	003	725	290	298	461	914	062	- 5	
	870		29		0.000										
	741		30		0.000										
147	483	648	31		0.000										5
294	967	296	32	8	0.000	000	000	232	830	643	653	869	628	906	25

## APPENDIX G.

#### FXAMPLES OF FLOATING FOINT NUMBERS

LUMBER	MACHINE	 Taos	The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
*** SINGLE IH	REPRESENTAI	LIOB				
16.000	41660000					
15.000	4130000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
14.000	41380000					
13.000	41340000		The state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the s		***	
12.000	413000C0					
11.000	4120000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
10,006	41280000					
9.00	41240000		a contract the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the			
8.000	41200000					The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon
70.0.0.	40E60000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
6.007	4010000		•		*	The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon
5.000	46F_80000					
4.000	40E0000C			*		
3.009	4,0 <u>6,000</u> 0					
\$ <b>⁻ ບ</b> ບ ເ	4670 <b>000</b>				· · · · · · · · · · · · · · · · · · ·	The state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the s
1.00€	406 <u>0000</u> 0					
0.000	G					
-1.000	BEA00000					
-2.000	FF600000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
-3.00/	11560000					
-4.000	BF200000		•			
<b>-5.00</b> 0	EF180000					
-6.000	FF100000					The second section of the second section of the second section of the second section of the second section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section section sec
-7.000	EF08000C					
-8.000	PEF00000		A CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR OF THE CONTRACTOR			
-9.00r	BELCCOOC					
-10.00c	FF.E.60000					
-11.000	6FU40000					
-12.000	BF100000		And the second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second s			the second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon
-13.000	FLCC0000					
-14.000	H U80000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
-15.000	EF C40000					
1.000	400000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
n.936	485C <b>000G</b>					
0.875	4638 <b>000</b> 6		the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of the second of th			
0.812	40340000			•		
0.750	4030 <b>000C</b>	*	The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
0.688	46200000					
0.625	40110000		The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon			
0 • 6 € . 0 • 5 € 2	40770000 40740000					
0.5(0						
0.430	402(0000					
	3/1/0066					
9.374	21 + 600CO					
0.313	31+10000					
0.250	244 60000					
0.180	21 60000					
0.12	SF/ L0000					e de la companya de la companya de la companya de la companya de la companya de la companya de la companya de
0.063	<0.0000 cm					
0.000	0					
-0.06s	C 0 / 0 0 0 0 0					
-0.12	( 66 60 000					The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon
-0.188	C 05.6 <b>6060</b>					The second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second second secon
-0.250	00500000		A constrained by the first			TO THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF THE PERSON OF
-c.31:	C 0 1 4 0 0 0 C					
-0.375	((160000					
-0.437	C C C 8 O O O C					the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the state of the s
-0.500	EFE 00000					
-0.£6.	L+100000					
+0.62t	00005					
<b>-೧.€</b> 8₺	FFE40000					
-0.75(	EFF-60000					
-0.813	` → € € 6 <b>0 0 0</b> 0					
-n.E7!	11101000					
-0.93/	FF C40000					the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the first of the f