

14122000
August 1980

PRELIMINARY

AN/AYK-14(V) INSTRUCTION SET
PROGRAMMER'S REFERENCE MANUAL

Standard Airborne Computer Set

AN/AYK-14(V)

CONTRACT: N00019-76-C-0697, Item 0017

CDRL Sequence Number J00D

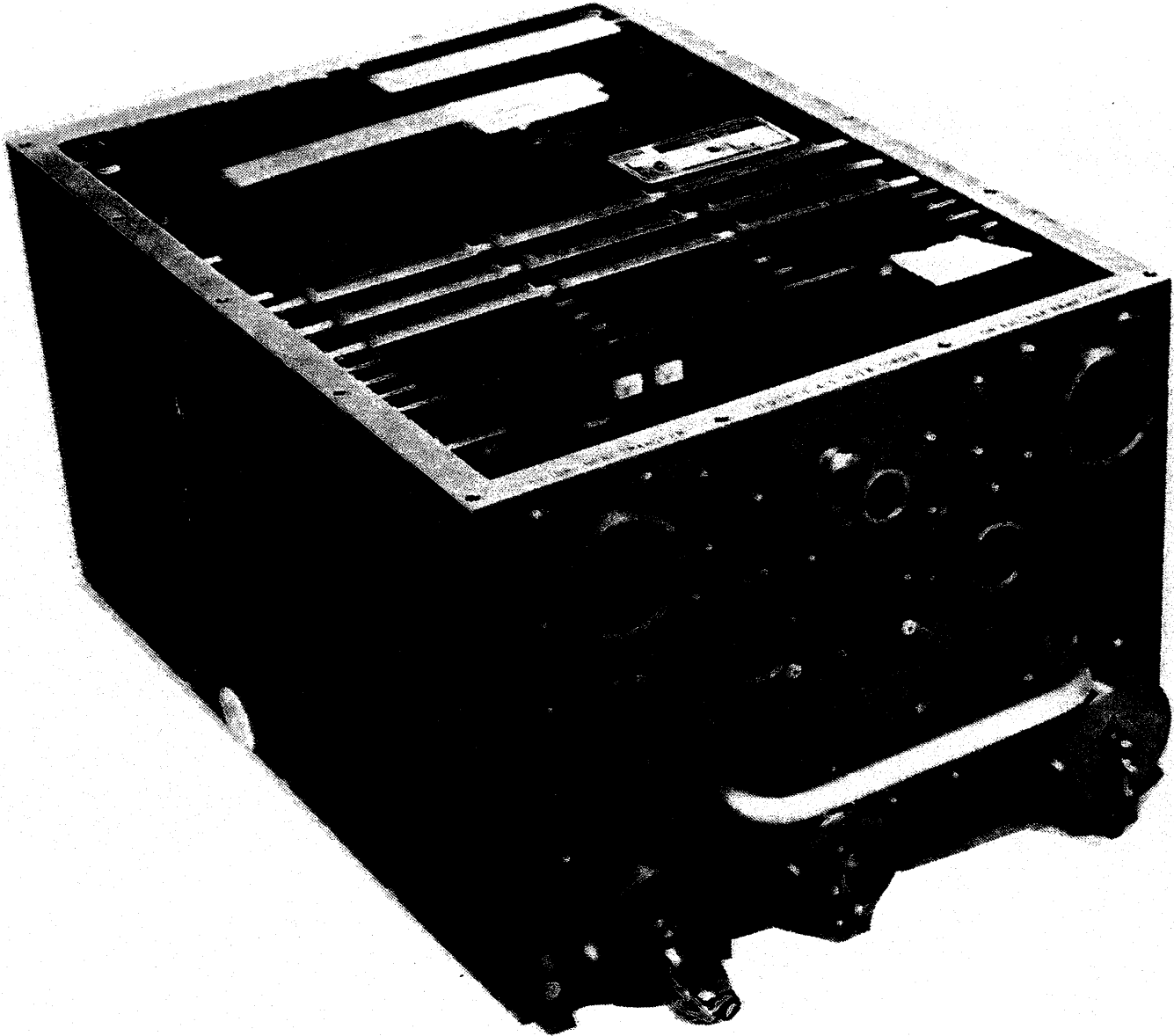
Prepared For:

DEPARTMENT OF THE NAVY
Naval Air Systems Command
Washington, D.C.

Prepared By:

CONTROL DATA CORPORATION
Aerospace Division
3101 East 80th Street
P.O. Box 609
Minneapolis, Minnesota 55440

(Doc Nos 0384A, 0386A, 0399A, 0511A, 0513A, 0517A 0565A)
(Disk Nos 0065A, 0069A, 0070A, 0076A, 0078A 0100A)



LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Rev.	Page	Rev.	Page	Rev.	Page	Rev.
Title Page	-						
ii	-						
iii/iv	-						
v - xii	-						
1-1 - 1-13/1-14	-						
2-1 - 2-16	-						
3-1 - 3-13/3-14	-						
4-1 - 4-10	-						
5-1 - 5-7/5-8	-						
6-1 - 6-4	-						
7-1 - 7-10	-						
8-1 - 8-40	-						
9-1 - 9-9/9-10	-						
10-1 - 10-103/ 10-104	-						
A-1 - A-244	-						
B-1 - B9/B10	-						
C-1 - C-6	-						

CONTENTS

Section	Page
1 GENERAL DESCRIPTION	1-1
Features of the AN/AYK-14(V)	1-1
Characteristics	1-1
Processing Subsystem	1-4
Intermodule Communication	1-4
Memory Subsystem	1-9
I/O Subsystem	1-9
Power Subsystem	1-10
Chassis Subsystem	1-10
Environment	1-10
Configuration Capability	1-10
2 ARCHITECTURE	2-1
Functional Organization	2-1
Memory Architecture	2-1
Memory Interfaces	2-1
Direct Memory Access (DMA) Capability	2-1
Memory Addressing	2-1
Memory Parity	2-2
Memory Protection	2-2
Assigned Memory Addresses	2-2
Read-Only Memory (ROM)	2-2
CPU Architecture	2-2
General Registers	2-3
Program Address Register	2-3
Real-Time Clock (RTC) and Monitor (MON) Clock Features	2-3
Power Failure and Thermal Protection Feature	2-3
Status Registers	2-5
Modularity	2-5
General Processor Module (GPM)	2-5
Processor Support Module (PSM)	2-5
Extended Arithmetic Unit (EAU)	2-6
Input/Output Processor (IOP)	2-6
I/O Modules	2-7
Discrete Interface Module (DIM)	2-7
Serial Interface Module (SIM)	2-8
NTDS Interface Modules (NIM)	2-8
RS-232-C Interface Module (RIM)	2-9
PROTEUS Interface Module (PIM)	2-9

CONTENTS (Cont.)

Section	Page
PIC/POC/SOC Module (PPSM)	2-9
Discrete Input/Output Module (DIOM)	2-13
Bus Extender Module (BEM)	2-13
Memory Modules	2-13
Memory Control Module (MCM)	2-13
Core Memory Module (CMM)	2-14
Semiconductor Memory Module (SMM)	2-14
Read/Write Expandable Module (RXM)	2-14
Power Converter Module (PCM)	2-14
Operation	2-15
3 INSTRUCTION, DATA, AND ADDRESSING FORMATS WITH ADDRESSING MODES	3-1
Instruction Formats	3-1
Data Formats	3-4
Addressing Formats	3-4
Arithmetic Instructions	3-7
Integer Arithmetic	3-7
Floating-Point Arithmetic	3-7
4 REGISTERS AND CLOCKS	4-1
General Registers	4-1
Status Registers	4-1
Status Register 1	4-1
Status Register 2	4-4
Breakpoint Registers	4-6
Program Address Register	4-7
AN/AYK-14(V) Clocks	4-7
Real-Time Clock	4-7
Monitor Clock	4-8
Built-In-Test Counter	4-9
5 MEMORY	5-1
Memory Subsystem	5-1
Interleaving	5-1
Word Interleaving	5-1
Bank Interleaving	5-1

CONTENTS (Cont.)

Section	Page
RXM Addressing	5-2
Bootstrap ROM Addressing	5-2
Memory Interlock	5-2
Paging	5-4
Paging Technique	5-4
Page Register 0	5-4
Memory Protection	5-6
Page Modification Indicator	5-6
Cautions on Memory Protection	5-6
6 COMPUTER SUPPORT EQUIPMENT	6-1
Computer Support Functions	6-1
Support Channel Operations	6-1
7 STACK AND QUEUE INSTRUCTIONS	7-1
List Processing Instructions	7-1
Stack Instructions	7-1
Queue Instructions	7-3
8 INSTRUCTION DESCRIPTIONS	8-1
General	8-1
Mnemonic Conventions	8-1
Double Precision	8-2
Repertoire of Instructions	8-3
Load Instructions	8-3
Store Instructions	8-6
Arithmetic Instructions	8-7
Logical Instructions	8-21
Compare Instructions	8-23
Shift Instructions	8-25
Jump Instructions (Unconditional)	8-27
Jump Instructions (Conditional)	8-29
Power-Out-of-Tolerance Jump Instructions	8-32
Miscellaneous Instructions	8-37
9 INTERRUPTS	9-1
Interrupt Processing	9-1
Class I Interrupts	9-3
Class II Interrupts	9-7
Class III Interrupts	9-8

CONTENTS (Cont.)

Section	Page
10 I/O CHANNEL OPERATIONS	10-1
General	10-1
Control Memory	10-1
Processor to I/O Channel Communication	10-5
Serial Interface Module	10-7
Message Formats	10-8
Word Formats	10-10
Control Memory Definition	10-10
SIM I/O Channel Instructions	10-15
SIM Interrupt Handling	10-17
SIM Programming Considerations in BC Mode	10-19
SIM Programming Considerations in RT Mode	10-21
Discrete Interface Module	10-22
Word Formats	10-22
Control Memory Definition	10-24
DIM I/O Channel Instructions	10-24
Message Formats	10-30
DIM Interrupt Handling	10-30
NTDS Interface Modules	10-33
Control Memory Definition	10-33
Message Formats (16-Bit Channel)	10-36
Message Formats (32-Bit Channel)	10-38
NIM I/O Channel Instructions	10-39
NIM Interrupt Handling	10-42
Programming Considerations (16-Bit Mode)	10-43
Programming Considerations (32-Bit Mode)	10-44
Programming Considerations (Serial Channel)	10-45
RS-232-C Interface Module	10-46
Data Formats	10-46
Message Formats	10-46
Control Memory Definition	10-46
RIM I/O Channel Instructions	10-52
RIM Interrupt Handling	10-54
Programming Considerations (Async)	10-56
Programming Considerations (Sync)	10-59
PROTEUS Interface Module	10-60
Channel Formats	10-60
Channel Sequences	10-63
Control Memory Definition	10-65

CONTENTS (Cont.)

Section	Page
PIM I/O Channel Instructions	10-68
PIM Interrupt Handling	10-70
Programming Considerations	10-74
PIC/POC/SOC Module	10-76
Message Formats	10-77
Control Memory Definition	10-78
PPSM I/O Channel Instructions	10-82
PIC/POC Interrupt Handling	10-85
SOC Interrupt Handling	10-88
Discrete Input/Output Module	10-88
Word Formats	10-88
Control Memory Definition	10-89
DIOM Channel Instructions	10-94
DIOM Interrupt Handling	10-97
DIOM Programming Considerations	10-100
APPENDIX A	INSTRUCTION REPERTOIRE AND INSTRUCTION EXECUTION TIMES
APPENDIX B	GENERAL REFERENCE TABLES
APPENDIX C	PSEUDO-OPS, COMMANDS, AND REQUESTS

ILLUSTRATIONS

Figure		Page
1-1	AN/AYK-14(V) System Elements	1-5
1-2	Minimum Configuration	1-12
1-3	Expanded Configuration	1-13
2-1	NTDS Slow, Fast, and ANEW Channel Interface	2-10
2-2	NTDS Serial Channel Interface and Message Format	2-11
2-3	RS-232-C Serial Channel Interface	2-12
2-4	PROTEUS Channel Pair	2-12
3-1	Instruction Formats	3-2
3-2	Operand Formation	3-3
3-3	Indirect Addressing Schemes	3-6
3-4	Integer Arithmetic Formats	3-8
3-5	Overflow and Carry Indications	3-9
3-6	Floating-Point Format	3-11
4-1	Status Register Number 1 Format	4-2
4-2	Status Register Number 2 Format	4-5
5-1	Bootstrap ROM Addressing	5-3
5-2	Memory Address Generations	5-5
7-1	Example of a Stack	7-2
7-2	Example of an SPT Operation	7-4
7-3	Example of an SGT Operation	7-5
7-4	Example of a Queue	7-6
7-5	Example of a QPT Operation	7-8
7-6	Example of a QPB Operation	7-9
7-7	Example of a QGT Operation	7-10
10-1	I/O Chain Program Initiation	10-3
10-2	I/O Chain Program Operation	10-4
10-3	SIM I/O Channel Type Message Formats	10-9
10-4	SIM Channel Type Word Formats	10-11
10-5	SIM Control Memory Map	10-12
10-6	SIM MCW Format	10-14
10-7	SIM Hardware Status Word 0 Format	10-18
10-8	SIM Hardware Status Word 1 Format	10-18
10-9	DIM Word Formats	10-23
10-10	DIM Control Memory Map	10-25
10-11	DIM BCW Format	10-26
10-12	DIM DSW Format	10-26
10-13	DIM I/O Channel Hardware Status Word 0 Format	10-29
10-14	DIM Hardware Status Word 1 Format	10-29
10-15	DIM MPW Formats	10-32
10-16	NIM Control Memory Map	10-34
10-17	NIM BCW Format	10-35
10-18	NIM Parallel MCW Format	10-37
10-19	NIM Serial MCW Format	10-37
10-20	NIM Parallel Mode Status Word	10-41

ILLUSTRATIONS (Cont.)

Figure		Page
10-21	NIM Serial Mode Status Word	10-41
10-22	NIM Hardware Status Word 0	10-42
10-23	RIM Sync/Async Data Formats	10-47
10-24	RIM Control Memory Map	10-48
10-25	RIM BCW Format	10-48
10-26	SMI Word Format	10-50
10-27	Hardware Status Word 0 Format	10-55
10-28	Hardware Status Word 1 Format	10-55
10-29	RIM Interrupt Word Format	10-56
10-30	RIM Async Output Data Transfer	10-57
10-31	RIM Async Input Data Transfer	10-58
10-32	PIM Control Frame Format	10-61
10-33	PIM Data Word Format	10-64
10-34	PIM Control Memory Map	10-66
10-35	PIM Sink Mode Control Word Format	10-67
10-36	PIM Source Mode Control Word Format	10-67
10-37	PIM Channel Hardware Status Word 0 Format	10-71
10-38	PIM Channel Hardware Status Word 1 Format	10-72
10-39	PIM Channel Hardware Status Word 2 Format	10-73
10-40	Command and Function Code Word Format	10-78
10-41	PIC/POC Control Memory Definition Map	10-79
10-42	SOC Control Memory Definition Map	10-80
10-43	PIC/POC Buffer Control Word	10-81
10-44	PIC/POC Buffer Control Word	10-81
10-45	SOC Buffer Control Word	10-83
10-46	SOC Mode Control Word	10-83
10-47	PIC/POC or SOC I/O Channel Status Word 0 Format	10-86
10-48	PIC/POC or SOC I/O Channel Status Word 1 Format	10-87
10-49	DIOM Control Memory Definition Map	10-91
10-50	DIOM Buffer Control Word (BCW)	10-92
10-51	DIOM Input Control Word (ICW)	10-92
10-52	DIOM Output Control Word (OCW)	10-93
10-53	DIOM Mode Control Word (MCW)	10-93
10-54	DIOM Status Word 0 Format	10-98
10-55	DIOM Status Word 1 Format	10-99
10-56	DIOM Interrupt Word Format	10-99

TABLES

Table		Page
1-1	AN/AYK-14(V) Computer System Specifications and Features . . .	1-2
1-2	Hardware Identification	1-6
2-1	Assigned Memory Addresses	2-4
2-2	PCM Capacities	2-16
2-3	Initial Conditions	2-16
3-1	Floating-Point Special Cases	3-12
4-1	Status Register 1 Condition Codes	4-4
9-1	Interrupt Lockout Effects	9-2
9-2	Assigned Memory Addresses	9-4
10-1	RIM Channel Speed Selection	10-51
10-2	Input Word Definitions	10-90
10-3	Output Word Definitions	10-90
10-4	Interrupt Word Codes	10-101

FEATURES OF THE AN/AYK-14(V)

The AN/AYK-14(V) computer system is a family of microprogrammed computers designed to provide low-cost standard airborne computers applicable to a wide range of vehicles and missions. The AN/AYK-14(V) computers operate in MIL-E-5400 environments; however, the basic module design is also applicable to configurations for shipboard and land environments.

The AN/AYK-14(V) system architectural philosophy is based on the following key features:

- 1) The AN/AYK-14(V) architecture and instruction set is upward compatible with that of the AN/UYK-20, permitting the adaptation and use of existing AN/UYK-20 support software.
- 2) The hardware is functionally partitioned into pluggable modules. These modules are the standard building blocks used in configuring functionally large or small computers.
- 3) Intermodule communications are standardized via uniform internal bus structures to permit reconfiguration and new module addition without impact on the architecture.

These combined features permit configuration of specific AN/AYK-14(V) computers to efficiently meet the processing requirements of a wide variety of military systems. Currently, the AN/AYK-14(V) computer system provides 19 module types which can be configured in various combinations in different chassis types. Configurations range from a two-module dedicated processor to multiple processors in multiple chassis with up to 524,288 words of memory and up to 17 I/O channels of various types.

CHARACTERISTICS

The AN/AYK-14(V) is a variable configuration, general purpose, 16-bit computer featuring a performance range of 400 to 800 KOPS (thousands of operations per second). The computer features a high degree of functional and mechanical modularity and is designed for flexible growth and extensive hardware commonality over a wide range of applications. The AN/AYK-14(V) architecture discernable to the user is not changed by modular hardware configuration changes, permitting common firmware and support software systems for all users. These design concepts are the key to providing a low-cost, versatile Navy standard airborne computer system. Table 1-1 summarizes the AN/AYK-14(V) specifications and features.

TABLE 1-1. AN/AYK-14(V) COMPUTER SYSTEM SPECIFICATIONS AND FEATURES

<p>GENERAL FEATURES</p> <p>GP, 16-bit digital computer Physically and functionally modular Expandable by plug-ins and additional enclosures Microprogrammed, emulates extended AN/UYK-20 LSI components ATR enclosures Variable configurations</p>	<p>PROCESSOR SUBSYSTEM (Cont.)</p> <p>I/O processor (optional)</p> <p>I/O controller capability Instruction subset compatible with central processor Microprogrammed Usable in conjunction with central processor or as standalone processor*</p> <p>Real-time clock 16-word by 16-bit general register set Addressing to 65,536 words Fixed-point, 16-bit arithmetic Interface to support equipment</p>
<p>PROCESSOR SUBSYSTEM</p> <p>Microprogrammed 2's complement arithmetic Executive and user states Two sets of 16-word by 16-bit general registers Two status registers Three-level interrupt system Addressing to 524,288 words Fixed and floating-point arithmetic 4-, 8-, 16-, and 32-bit operands 16- and 32-bit instructions Direct, indirect, and indexed addressing Optional hardware floating-point module (EAU) Loadable/readable 32-bit RTC clock, 1-MHz rate; 16-bit monitor clock, 10-kHz rate, Built-in test functions Bootstrap PROM memory Power failure shutdown/recovery I/O controller capability: Chaining capability Control memory for each channel Up to 16 channels in various combinations Interface to support equipment Sample instruction times: Shift 1.1 μs Add, subtract .9 Multiply 4.2 Divide 8.3 Basis: single GPM, core memory, overlapped access, interleaved addresses</p> <p>*32K words available (standalone)</p>	<p>MEMORY SUBSYSTEM</p> <p>Core memory module (CMM), 16K or 32K words of 18 bits Semiconductor memory module (SMM), 16K or 32K words of 18 bits Interchangeable core and semiconductor memory modules CMM has 900-nanosecond cycle time and 350-nanosecond access time SMM has 400-nanosecond cycle time and 300-nanosecond access time Interleaved or noninterleaved addressing Read/write expandable memory (RXM), 4K x 18-bit RAM with optional 4K PROM Parity bit per byte Protect features: Write protect Read protect Execute protect Block protect in paging system Memory controller with paging to 524,288 words</p>
	<p>CHASSIS SUBSYSTEM</p> <p>Bus extension module (BEM) extends all buses outside the enclosure (two-chassis system)</p>

TABLE 1-1. AN/AYK-14(V) COMPUTER SYSTEM SPECIFICATIONS AND FEATURES (Cont.)

<p>I/O SUBSYSTEM</p> <p>Discrete interface module (DIM) Eight program selectable external device interrupts 32 bidirectional input or output discretes 16 differential input discretes 16 switch closure input discretes</p> <p>Serial interface module (SIM) MIL-STD-1553A multiplexed bus 50-kHz, 16-bit word rate 32 terminals per bus Operation as bus controller or remote terminal</p>	<p>PIC/POC/SOC module (PPSM) Parallel input channel (PIC) Parallel output channel (POC) Serial output channel (SOC) 32-bit parallel data transfers Serial NRZ bit data transfers in 16-bit words Serial data rate of 200 kHz or 1 MHz selectable under program control Internal and external testability</p> <p>Discrete I/O module (DIOM) 144 output discretes 48 input discretes or interrupts Full duplex Internal testability</p>																														
<p>NTDS interface module (NIM) MIL-STD-1397 Parallel channels: NTDS slow (41,667 words/sec) NTDS fast (125K words/sec) ANEW (125K words/sec) 16-bit and 32-bit (dual channel)(125K words/sec) operation Computer-to-peripheral and computer-to-computer modes Externally specified addressing on dual channels Serial channels: 125K words/sec 16- or 32-bit (dual channel) message formats</p>	<p>PHYSICAL</p> <table border="1"> <thead> <tr> <th>Chassis</th> <th>Height</th> <th>Width</th> <th>Depth</th> <th>Weight*</th> </tr> </thead> <tbody> <tr> <td>XN-1</td> <td>7.62"</td> <td>10.12"</td> <td>19.56"</td> <td>45-55**</td> </tr> <tr> <td>XN-2</td> <td>7.62"</td> <td>10.12"</td> <td>14.00"</td> <td>34</td> </tr> <tr> <td>XN-3</td> <td>7.62"</td> <td>10.12"</td> <td>12.75"</td> <td>35-45**</td> </tr> <tr> <td>XN-4</td> <td>7.70"</td> <td>10.20"</td> <td>22.88"</td> <td>35-45**</td> </tr> <tr> <td>XN-5</td> <td>7.60"</td> <td>10.12"</td> <td>14.00"</td> <td>35-45**</td> </tr> </tbody> </table> <p>*Does not include fan **Weight varies as functions of optional modules installed.</p> <p>Service conditions as specified in MIL-E-5400 for class 1, 1A, 1B, and 2X equipment</p>	Chassis	Height	Width	Depth	Weight*	XN-1	7.62"	10.12"	19.56"	45-55**	XN-2	7.62"	10.12"	14.00"	34	XN-3	7.62"	10.12"	12.75"	35-45**	XN-4	7.70"	10.20"	22.88"	35-45**	XN-5	7.60"	10.12"	14.00"	35-45**
Chassis	Height	Width	Depth	Weight*																											
XN-1	7.62"	10.12"	19.56"	45-55**																											
XN-2	7.62"	10.12"	14.00"	34																											
XN-3	7.62"	10.12"	12.75"	35-45**																											
XN-4	7.70"	10.20"	22.88"	35-45**																											
XN-5	7.60"	10.12"	14.00"	35-45**																											
<p>RS-232-C interface module (RIM) Asynchronous 75 to 9600 baud Synchronous to 9600 baud</p> <p>PROTEUS interface module (PIM) 125K words/sec Serial transfer, 32-bit word format</p>	<p>PRIMARY POWER</p> <p>115 Vac, 400 cycle, three phase, wye connected as per MIL-STD-704 400 to 600 watts for XN-1* 250 to 400 watts for XN-2* 150 to 350 watts for XN-3* 150 to 350 watts for XN-4* 150 to 350 watts for XN-5*</p> <p>*Power varies as function of optional modules installed.</p>																														

TABLE 1-1. AN/AYK-14(V) COMPUTER SYSTEM SPECIFICATIONS AND FEATURES (Cont.)

AVAILABLE OPTIONAL BOLT-ON FAN COOLING			AVAILABLE OPTIONAL BOLT-ON FAN COOLING (Cont.)		
<u>Fan</u>	<u>Length</u>	<u>Diameter</u>	<u>Weight</u> (lb)	<u>Power</u> (at sea level)	<u>Altitude</u> (ft)
IMC 5026	3.10"	2.75"	2.00	100	30,000

The AN/AYK-14(V) consists of a family of pluggable modules, chassis, interconnecting buses, support equipment, software, firmware, documentation, and training necessary to provide the user with a completely supported computer system. Figure 1-1 depicts the system elements by subsystem and shows the functional modules applicable to each subsystem. Table 1-2 briefly defines the AN/AYK-14(V) element nomenclature.

PROCESSING SUBSYSTEM

The general processing module (GPM) contains all the microprogrammed control, arithmetic unit, registers, and bus interfaces. The processor support module (PSM) contains the supporting elements such as micromemory, real-time clocks, bootstrap memory, bus interfaces, and event (interrupt) logic required to complete the function of the GPM. Together they form a 16-bit central processing unit (CPU) of a general purpose computer. The extended arithmetic unit (EAU) provides high-speed, 32-bit, floating-point hardware and operates under the control of the GPM.

The IOP combines the basic functions of the GPM and PSM on one module with a reduced instruction set and performance level. The IOP is microprogrammed to serve either as an IOC or as a single-module, 16-bit, general purpose CPU without modification.

The IOP, when used in a dual processor configuration, performs all I/O operations and I/O event-related functions along with executing software programs initiated by IOP recognized instructions. The CPU performs all event-related functions associated with the memory and power subsystems along with executing software programs initiated by any of the instructions. The dual event system allows CPU-to-IOP and IOP-to-CPU communications.

The IOP, when used in a standalone configuration, performs all I/O operations, I/O events, memory subsystem events, power subsystem events, and executes software programs initiated by IOP recognized instructions.

INTERMODULE COMMUNICATION

The functional modules communicate via one or two identical internal buses: the CPUBUS and the IOBUS. These high-speed, 24-bit parallel buses are the principal data transfer paths between processing modules, memory, and the I/O channels. Additional control signals are transmitted via the EVENTBUS, which transfers interrupt and other event signals. Internal common module

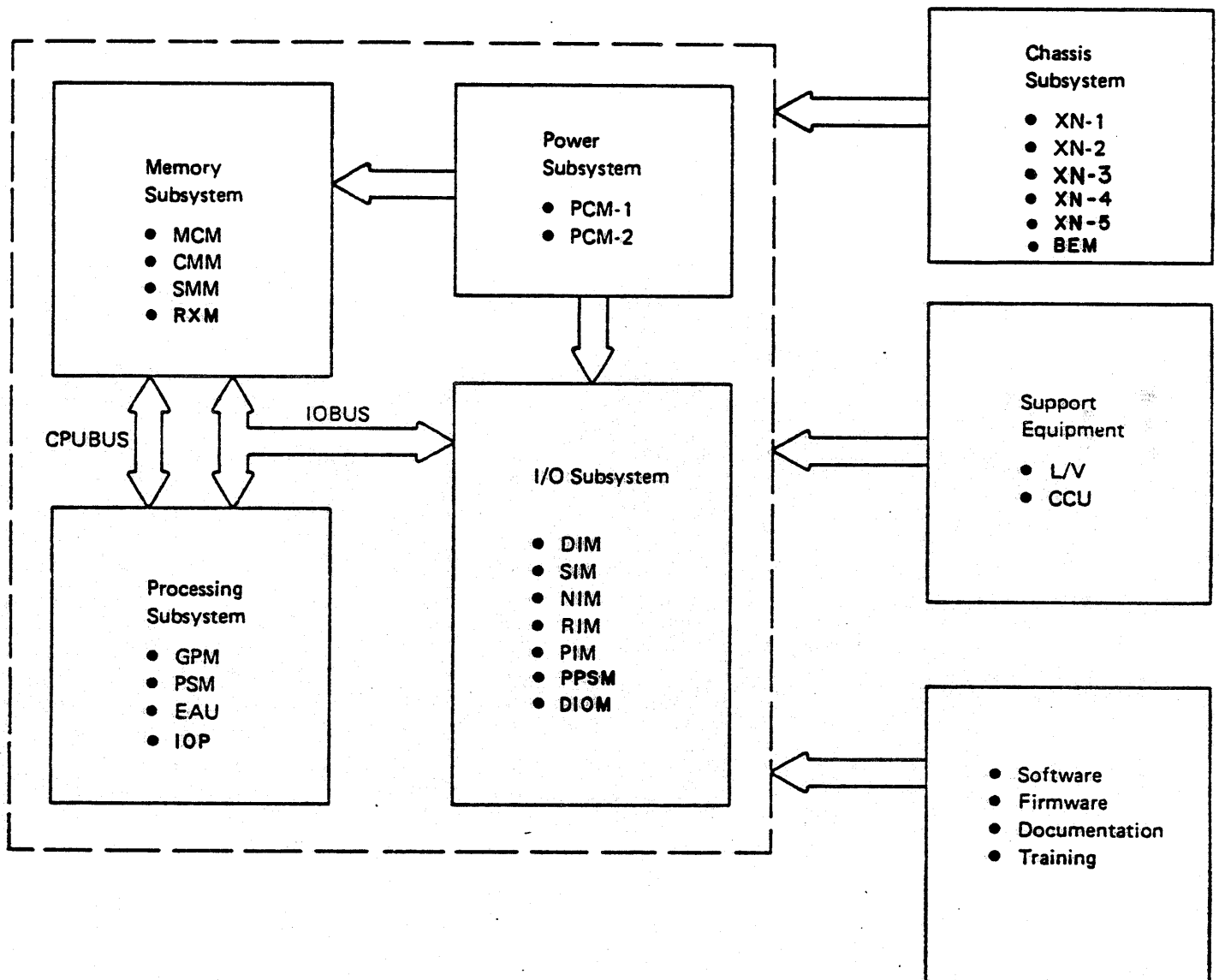


Figure 1-1. AN/AYK-14(V) System Elements

TABLE 1-2. HARDWARE IDENTIFICATION

Module	Typical Power (Watts)	Typical Weight (Lb)	Name	Function
GPM	49	2.13	General processor	16-bit processor with control, registers, and arithmetic unit
PSM	43	2.06	Processor support	Micromemory, real-time clock, interrupt system
EAU	40	2.13	Extended arithmetic unit	High-speed, floating-point arithmetic
IOP	44	2.18	I/O processor	16-bit computer with control, registers, arithmetic unit, micro-memory, and interrupt system
MCM	36	1.87	Memory control	Memory controller with paging, protect, parity, and two ports
CMM	31	3.10	Core memory	32K words by 18-bit core memory
SMM	17	2.50	Semiconductor memory	16K words by 18-bit semiconductor memory
RXM	9	0.95	Read/write expandable memory	4K words by 18-bit semiconductor RAM memory with optional addition of 4K ROM or PROM
DIM	14	1.02	Discrete I/O	32 input discretes, 32 I/O programmable discretes, eight interrupts
SIM	16	1.09	Serial I/O	1553A serial multiplex channel, 1-MHz bit rate
NIM	18	1.05	NTDS fast I/O	NTDS fast interface 125K words/second
NIM	17	1.05	NTDS slow I/O	NTDS slow interface, 41,667 words/second

TABLE 1-2. HARDWARE IDENTIFICATION (Cont.)

Module	Typical Power (Watts)	Typical Weight (Lb)	Name	Function
NIM	10	1.05	NTDS ANEW I/O	NTDS ANEW interface 125K words/second
NIM	10	1.05	NTDS serial I/O	NTDS serial interface, 125K words/second
RIM	15	1.06	RS-232-C I/O	One serial RS-232-C channel, 9600 bauds
PIM	13	1.04	PROTEUS I/O	I/O serial channel pair, 125K words/second
PPSM	23	1.25	Parallel I/O, serial output	32-bit parallel input/output, serial output 16-bit words
DIOM	28	1.87	Discrete I/O	144 output discrettes, 48 input discrettes or interrupts
BEM	29	1.05	Bus extender	Extends all AN/AYK-14(V) internal buses outside the enclosure
*PCM-1 PCM-2	Varies	9.50 11.58	Power converter	Regulated power supply. MIL-STD-704 power input, status outputs
Chassis				
XN-1	N/A	17.4		19.56" by 10.125" by 7.625" ATR enclosure and chassis
XN-2	N/A	11.84		14" by 10.125" by 7.625" ATR enclosure and chassis
XN-3	N/A	13.2		12.75" by 10.125" by 7.625" ATR enclosure and chassis for computer extension
XN-4	N/A	18.48		22.881" by 10.197" by 7.718" ATR enclosure and chassis

TABLE 1-2. HARDWARE IDENTIFICATION (Cont.)

Module	Typical Power (Watts)	Typical Weight (Lb)	Name	Function
Chassis				
XN-5	N/A	12.40		14" by 10.125" by 7.625" ATR enclosure and chassis
Support Equipment				
L/V	80	43	Loader/verifier	Portable militarized tape loader and control panel
CCU	440	383	Computer control unit with tape unit and formatter	Laboratory operator console for firmware and software debugging and maintenance; interfaces to commercial peripherals

*PCM-1 provides 390 watts of output power, $n \approx 71\%$
 PCM-2 provides 540 watts of output power, $n \approx 71\%$

interfaces permit flexible module configuration and ensure that module modification or addition of new types will not result in existing module modification.

MEMORY SUBSYSTEM

The memory subsystem includes interchangeable 16K and 32K word core memory modules (CMM) and 16K-word semiconductor memory modules (SMM) with 18-bit word length. The CMM cycle time is 900 nanoseconds and the SMM cycle time is 400 nanoseconds. The memory control module (MCM) interfaces between the GPM and the memory modules (CMM or SMM). The MCM has both CPUBUS and IOBUS interfaces which permit the GPM to use one bus for instruction access and the other for operands to enhance effective access time. The MCM also provides two channels to memory modules, the OMEMBUS and EMEMBUS, which can increase effective access time through interleaved addressing between two memory banks.

The read/write expandable memory module (RXM) is a 4K word by 18-bit semiconductor memory with 400-nanosecond cycle time which operates directly with the IOP or GPM via either the CPUBUS or the IOBUS. An optional addition of 4K read-only memory (ROM or PROM) is also available. The primary application of the RXM is to provide memory for use with the IOP as a small, dedicated two-card computer.

I/O SUBSYSTEM

The AN/AYK-14(V) system organization provides for up to 17 I/O channels, each on individual functional modules which communicate with the computer system via identical CPUBUS or IOBUS. The standardization of internal interfaces permits any I/O channel module type to be interchanged in the chassis I/O slots by simple plug-in replacement. Available chassis provide from four to six I/O channels. Expansion to more I/O channels requires the additional XN-3 enclosure. The following I/O module types are currently available to match standard I/O channel characteristics.

- MIL-STD-1553A avionics serial multiplex bus
- NTDS (fast, slow, ANEW, and serial) MIL-STD-1397
- RS-232-C
- PROTEUS

The I/O controller (IOC) functions can be executed by either the central processor (GPM and PSM) or the optional I/O processor (IOP). The incorporation of an IOP into an AN/AYK-14(V) system, operating in conjunction with the central processor, greatly enhances the processing thruput.

User-equipment interrupts can be brought into the system either through the associated I/O channel or via the discrete interface module (DIM), which also has provision for 32 input and 32 input or output discrete signals.

Software selectable internal wraparound provides a means to test some I/O channels. This allows the CPU or IOP to perform diagnostics on the I/O channels without testing the transmitters and receivers.

POWER SUBSYSTEM

Power for all modules in an enclosure is supplied by a power converter module (PCM) with appropriate regulated voltage and current capabilities. Present designs operate on MIL-STD-704 power, 115 Vac, 400 cycle, three phase, wye connected.

CHASSIS SUBSYSTEM

All modules plug into an ATR-type chassis equipped with slots to accommodate a combination of module types. Currently, three standard chassis types designed for MIL-E-5400, Class II environments, are available for 16-bit computers. Connector location and basic dimensions are shown. It should be noted that the XN-3 is an extension unit to be used with the 100 series (XN-1) or 200 series (XN-2) chassis to provide additional memory, processing, and/or I/O capability. Multiple 300 series (XN-3) chassis can be used to further expand the system.

Expansion of the computer beyond a single enclosure or implementation of direct memory access (DMA) I/O is effected through the use of the bus extender module (BEM), which provides a buffered extension of all internal computer buses to another enclosure.

ENVIRONMENT

The basic module of the AN/AYK-14(V) family is designed for use in MIL-E-5400 (airborne) when installed in a suitable enclosure. The total range of conditions includes temperatures of -54° to 71°C, altitudes to 70,000 feet, and levels of shock, vibration, humidity, and EMI appropriate to these environments.

Qualification of the chassis types (listed in Table 1-2) to MIL-E-5400 Class II requirements will occur under the current AN/AYK-14 contract from the U.S. Navy.

All modules are designed for conduction cooling via a heat sink backing the printed circuit boards. The modules have ramp clamps along both short edges to provide solid mechanical and thermal contact to the slots in the chassis. Heat is transferred from the chassis heat sink via an air plenum, which may be supplied by a vehicle cooling air system or optional bolt-on fan. No cooling air is needed over module components.

The rigid module structure, stiffened by the heat sink, withstands severe shock and vibration environments.

All modules are conformal coated for moisture resistance. The module design permits great flexibility in chassis cooling provisions to meet multiple application requirements.

CONFIGURATION CAPABILITY

The functional partitioning of the modules and the internal bus structures provide for flexible configuration of a wide range of AN/AYK-14(V) computers. The AN/AYK-14(V) system allows building up the system by addition

of modules to meet the problem computing bandwidth and capacity requirements. Some examples are given to show how these building blocks can be used to balance computer size, weight, power, and cost against performance.

Figure 1-2 shows the minimum AN/AYK-14(V) computer configuration which consists of an IOP as the 16-bit processor and an RXM 4K by 18-bit random access memory (RAM) semiconductor memory (with optional 4K PROM). This is a bare module configuration and assumes that the modules are incorporated as components into the user's equipment. The user's equipment power supply would provide regulated 5-Vdc power for the modules and the user would also provide the I/O adaptation to the IOBUS interface. IOP/RXM combinations can also be used effectively as computing elements in distributed processing systems.

An expanded configuration (Figure 1-3) yields a complete 16-bit, general purpose processor with high-speed floating-point hardware, hardware I/O controller, 128K words of 18-bit core memory, and up to 16 I/O channels of various types. This example illustrates the role of the identical CPUBUS and IOBUS in organizing the modules into a powerful computer. Since the GPM has two bus interfaces to the MCM, it is possible to overlap instruction and operand fetches from memory. In addition, it should be noted that the two memory channels, OMEMBUS and EMEMBUS, permit interleaving of memory addresses between memory banks for high, effective access speed.

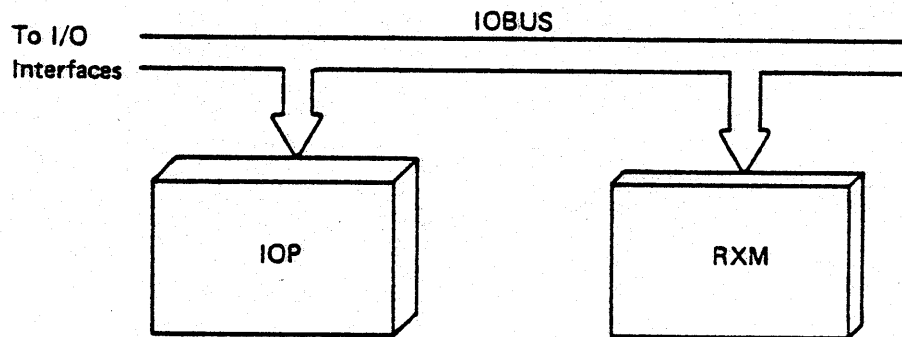


Figure 1-2. Minimum Configuration

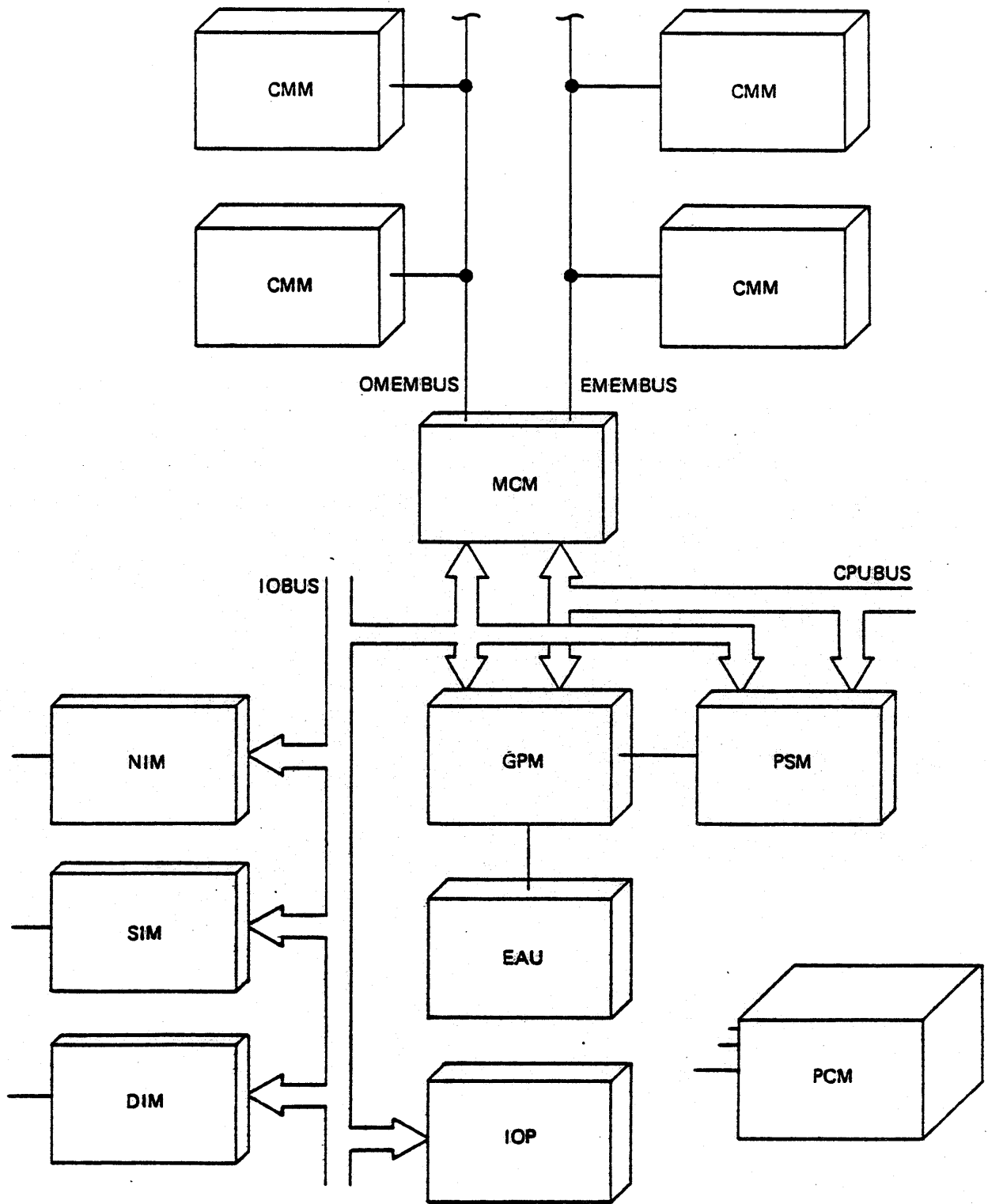


Figure 1-3. Expanded Configuration

FUNCTIONAL ORGANIZATION

The functional architecture of the AN/AYK-14(V), that is the architecture perceived by the programmer or other user, is implemented via the AN/AYK-14(V) microcode operating in a suitable configuration of AN/AYK-14(V) modules. The architecture is upward compatible with the AN/UYK-20 architecture. All instructions common to both AN/AYK-14(V) and AN/UYK-20 have identical formats and operation codes.

MEMORY ARCHITECTURE

The main memory consists of combinations of CMM and SMM modules, up to a maximum of 524,288 words, and an MCM to provide dual access port, paging, parity operation, and protect features.

Memory Interfaces

The memory interfaces to the CPU with the capability to overlap. Overlap is always used with the GPM which accesses instructions on the CPUBUS and operands on the IOBUS. The IOP interfaces to the memory system only via the IOBUS, and thus does not use overlap.

The AN/AYK-14(V) can interleave memory addresses between the memory modules interfacing on the OMEMBUS and with those on the EMEMBUS. This interleaving enhances effective access time in transferring sequentially addressable words. Whenever the configuration of memory modules is identical on both OMEMBUS and EMEMBUS, interleaving is automatically provided unless a jumper on a front panel connector is used to inhibit interleaving. Interleaving is not used when an odd number of memory modules are installed, or the assortment of memory types is not symmetrical on both memory buses.

Overlap and interleaving are independent features.

Direct Memory Access (DMA) Capability

The BEM provides for extending memory interfaces external to the chassis. This module permits a DMA capability through a user-provided external DMA controller. DMA transfer does not require processing by either the CPU or IOP.

Memory Addressing

The memory addressing capability provides addressing to 524,288 words through the paging features incorporated in the MCM. The 16-bit relative address from the CPU or IOP is converted to a 19-bit address. The lower 10 bits of the relative address specify one of 1024 words within a page, while the upper six bits specify which of the 64 page registers will be referenced to determine the 9-bit page base address. Software instructions B0 through B7 (hexadecimal) provide the capability for loading and storing the page registers.

Memory Parity

The memory system incorporates a parity bit for each 8-bit byte. Parity is generated and checked by the MCM and an interrupt is generated upon a parity error.

Memory Protection

Memory protection features on a 1024-word page basis are provided for data security and assurance of program integrity. Three types of protection are implemented via bits stored in the page register. These types are:

Execute Protection - generates an interrupt if instruction execution is performed from a protected page.

Write Lockout - generates an interrupt if a write operation is attempted in a protected page.

Read Protection - generates an interrupt if a read operation is performed from a protected page.

In addition to the three protection bits, a bit in each page register serves as a page modification register. This bit, when set, indicates that a write operation was made in the associated page. The load address register instructions permit modification of protected bits in the page registers.

Assigned Memory Addresses

In general, programs, constants, and data can be stored in any address. There are, however, some assigned locations (as shown in Table 2-1) which are associated with executive, interrupt, I/O functions, and ROM mode.

Read-Only Memory (ROM)

The memory system includes two segments of words of ROM memory containing the bootstrap program. This memory duplicates the address space of words 00₁₆ to 3F₁₆ and C0₁₆ through 3FF₁₆ of main memory and is entered upon initiation of operations. A bit in status register number 1 controls the selection of ROM or main memory. Bootstrap operations are provided via a 1553A I/O channel.

CPU ARCHITECTURE

The AN/AYK-14(V) operates in the following two modes.

Executive - used for executive functions. In this mode, all instructions can be executed.

Program - used for user program functions. In this mode, any instructions except executive instructions can be executed.

Modification of status registers and page address registers is restricted to executive mode.

This two-mode feature simplifies and increases the speed of the executive control and aids in integration of user program modules into the system software.

General Registers

The AN/AYK-14(V) has two sets of 16-word by 16-bit general registers, each set designated R0 through R15 and an instruction set tailored to their manipulation. The selection of the register set to be used is designated by status register 1, bit 14. These registers can be used as follows:

- Accumulators for arithmetic, shift, and logical operations
- Index registers for address and operand modification
- Temporary storage locations for addresses and operands.

The large number of general registers and the register-register instructions yield benefits in execution time and decreased storage requirements compared to architectures using an A/Q register organization or fewer registers.

The general registers are referenced by the register designator fields (a,m) of the AN/AYK-14(V) instructions.

Program Address Register

The program address register, P, holds the address of the instruction being executed in a program sequence. Its contents are automatically advanced by one each time a single-length (16-bit) instruction is executed and by two for a double-word (32-bit) instruction. Jump instructions load the P register with the entry address of the program that receives control.

Real-Time Clock (RTC) and Monitor (MON) Clock Features

The AN/AYK-14(V) contains an RTC and a MON clock which are loadable and readable under software control and provide interrupts when enabled. The RTC counts up a 32-bit register at a 1-MHz rate, allowing for timed intervals up to $(2^{32}-1)$ microseconds or approximately 1.19 hours.

The MON clock is a 16-bit counter which counts down at a 10-kHz rate to provide interrupts at intervals up to approximately 6.5 seconds. These features are useful for scheduling periodic processing activities, coordinating I/O operations, and timing real-time events. The high resolution of the AN/AYK-14(V) clock is particularly useful for signal processing applications and weapons control functions associated with high-speed vehicles.

Power Failure and Thermal Protection Features

The power failure and thermal protection features provide for orderly shut-down and preparation for recovery if the computer power falls below a safe threshold or if an overtemperature condition occurs. The PCM monitors power failure and thermal condition and, upon detection of one or the other, provides a signal to generate a CPU interrupt. The CPU then has about 300 microseconds to store desired registers and status before the PCM shuts down.

TABLE 2-1. ASSIGNED MEMORY ADDRESSES

Function	CPU						IOP					
	I		II		III		I		II		III	
	Hex	Octal	Hex	Octal	Hex	Octal	Hex	Octal	Hex	Octal	Hex	Octal
Store P	58	130	50	120	48	110	68	150	70	160	48	110
Store SR1	59	131	51	121	49	111	69	151	71	161	49	111
Store SR2	5A	132	52	122	4A	112	6A	152	72	162	4A	112
Store RTC lower	5B	133	53	123	4B	113	6B	153	73	163	4B	113
P reload	5C	134	54	124	4C	114	6C	154	74	164	4C	114
SR1 reload	5D	135	55	125	4D	115	6D	155	75	165	4D	115
SR2 reload	5E	136	56	126	4E	116	6E	156	76	166	4E	116
Store RTC upper	5F	137	57	127	4F	117						
I/O Command cell	60-61 ₁₆ , 140-141 ₈						62-63 ₁₆ , 142-143 ₈					
Auto start entrance	7F ₁₆ , 177 ₈											
External interrupt word storage	80-8F ₁₆ , 200-217 ₈						80-8F ₁₆ , 200-217 ₈					
ROM	0-3F ₁₆ & C0-3FF ₁₆ , 0-77 ₈ & 300-1777 ₈											

Status Registers

The AN/AYK-14(V) contains two 16-bit status registers, status register number 1 (SR1) and status register number 2 (SR2), which provide an indication of the computer state, error conditions, and interrupt lockouts. These registers are accessible to all programs, but can only be modified by software in the executive mode. Upon program interruption, SR1 and SR2 are automatically stored in memory, where they can be recalled and reinstated upon completion of the interrupt processing routine to allow continuation of the original program with the status existing before interruption.

MODULARITY

The modules described in this section represent the current AN/AYK-14(V) module designs. It is expected that new module types will be added to meet future applications requirements. New or special modules will be designed to interface with standard AN/AYK-14(V) internal buses to preserve system integrity.

GENERAL PROCESSOR MODULE (GPM)

The GPM is a 16-bit microprogrammable processor based on the AMD 2900 series microprocessor slice LSI devices. The architecture is augmented for high-speed performance with additional registers, internal data, and control transfer paths. The GPM features which contribute to its performance include:

- 48-bit microcommand control
- Microprogram address sequencing to 4K words
- 180-nanosecond microcommand cycle
- 256 by 16-bit word register file
- 256 by 16-bit word multiport CFILE
- Dual identical parallel bus interfaces (CPUBUS and IOBUS)
- Event interface
- Interface to micromemory on PSM
- Serial interface to support equipment
- Interface to EAU.

The GPM operators from microcommands stored on the PSM module (up to 4K words of micromemory).

PROCESSOR SUPPORT MODULE (PSM)

The PSM augments GPM functions to form a complete 16-bit computer in two modules. The partitioning of the functions between GPM and PSM was designed to allocate those functions to the PSM that might require modification as applications change. The PSM features include:

- Up to 4K by 48 bits of PROM micromemory for the GPM
- 1K by 16 bits of PROM bootstrap memory for computer system initiation via the 1553A I/O channel
- Two parallel bus interfaces (CPUBUS and IOBUS)
- Event interface
- Event monitor logic, which forms the basic hardware portion of the

- event (interrupt) processing
- 32-bit high-speed multiply logic
- BIT timer with 2.097-second increment, 4-bit count.

The AN/AYK-14(V) computer micromemory may contain commands for processing a variety of functions including:

- 1) AN/AYK-14(V) instruction set interpretation and maintenance of the computer status
- 2) Built-in-test (BIT) functions
- 3) Diagnostic and fault isolation functions
- 4) Special macroinstruction or algorithm processing.

EXTENDED ARITHMETIC UNIT (EAU)

The EAU is a 32-bit, high-speed, floating-point processor which operates under the control of the GPM and interfaces directly to it. The EAU utilizes the floating-point format which consists of a 7-bit exponent and a 24-bit mantissa. A GPM/PSM configuration will execute the floating-point add, subtract, multiply, and divide instructions via firmware. When an EAU module is added, the floating-point add, subtract, multiply, and divide are performed by the EAU as well as the nine trigonometric instructions which are only legal with the EAU present. The incorporation of the EAU automatically increases floating-point execution speed without firmware or software changes.

INPUT/OUTPUT PROCESSOR (IOP)

The IOP is a complete 16-bit processor combining the basic functions of the GPM and PSM on one module. The instruction set is a subset of the total AN/AYK-14(V) instruction set. To accomplish a one-module processor, the performance and features are reduced from the GPM/PSM capability. The IOP is intended for use in the following three general applications types.

- 1) As a small scale, standalone, general purpose processor with emulation capabilities.
- 2) As an I/O controller (IOC) in conjunction with a GPM/PSM as instruction processor.
- 3) As a combination IOC and instruction processor in conjunction with a GPM/PSM.

A summary of differences between the IOP and CPU is as follows:

- Only one set of 16 general registers is available (stack 0)
- Page register modification and indirect addressing are not incorporated.
- The operand breakpoint register is not emulated.
- The IOP real-time clock, when enabled, is incremented at a 1.024-microsecond rate.

- The RTC register is 16 bits.
- The IOP BIT timer is 3 bits wide
- No monitor clock.

Features of the IOP include:

- 48-bit microcommand control
- Up to 2K micromemory on the module
- 250-nanosecond microcommand cycle
- 256 by 16-bit word register file
- Single parallel bus interface (IOBUS)
- Event interface
- Serial interface to support equipment
- BIT timer, 2.097-second increment, 3-bit count
- Event monitor logic
- Microcommand format identical to GPM.

I/O MODULES

All I/O channel modules are physically the same size and are interchangeable in any I/O module chassis slot. Each I/O channel module implements a single I/O channel of a designated type and has a common set of intermodule interfaces including the IOBUS and event interface. Each I/O module type contains the logic to implement the specific channel type characteristics and operate with a standardized IOBUS communication procedure. All I/O modules have provision for a module test operation in which test data is looped through the module and returned to the processor. The standard I/O channel module set can be augmented with special channel modules to meet system requirements. The special channels will use the same IOBUS and event interface as the standard I/O modules.

Discrete Interface Module (DIM)

The DIM is used to provide a convenient interface for communicating single-bit status, event, or control information between user devices and the computer.

The DIM provides the following interface capabilities.

- Eight external device interrupts. These can have program selectable priority and can be individually masked.
- 32 bidirectional input or output discretes. These use differential TTL interface signals. They are program selectable as inputs or outputs in groups of four.
- 16 differential input discretes. These use differential lines, ac terminated.
- 16 switch closure input discretes.
- The 32 bidirectional discretes have a loop test capability.

Serial Interface Module (SIM)

The SIM implements a serial multiplex data channel meeting the channel control and format characteristics of MIL-STD-1553A. This channel type is the standard intersystem communication facility on board modern military aircraft. The module interfaces to two 1553A-type buses for redundant operation.

The module can operate with any MIL-STD-1553A protocol and can function as either a bus controller or remote terminal unit. Information is transferred on a single shielded, twisted pair line at a 1-MHz bit rate. Data is transferred in 20-microsecond words, each divided into 17-bit times of 1-microsecond and one 3-microsecond sync interval. All messages are addressed and use three types of words.

Command word - sent by bus controller to address appropriate terminal, specify message type, and set data word count for subsequent transfer.

Status word - set by a terminal in response to command word. Identifies terminal and reports status.

Data word - containing 16 bits of message data, sync pattern, and a parity bit.

Up to 32 terminals can interface on a single bus. All transmissions and receptions are initiated and controlled by the bus controller using message formats.

The SIM contains interfaces to two 1553A buses and has the capability of data transfer on one and monitoring the other at any time.

NTDS Interface Modules (NIM)

There are four types of NIMs, each capable of operating according to MIL-STD-1397.

- 1) NTDS Slow - 16-bit parallel transfer up to 41,667 words per second. Binary voltage levels of 0 Vdc (logical 1) and -15 Vdc (logical 0).
- 2) NTDS Fast - 16-bit parallel transfer of up to 125,000 words per second. Binary voltage of 0 Vdc (logical 1) and -3 Vdc (logical 0).
- 3) ANEW - 16-bit parallel transfer of up to 125,000 words per second. Binary voltage levels of 0 Vdc (logical 1) and 3.5 Vdc (logical 0).
- 4) Serial - serial data transfer of up to 125,000 words per second on one cable. Bipolar $\pm 3.25V$ signals.

Channel interface lines for NTDS fast, slow, and ANEW are shown in Figure 2-1 and for serial, in Figure 2-2. Two NIM parallel channels can be operated together to form a 32-bit wide parallel channel. Transfer operations on the serial channel involve the use of 3-bit control frames and 34-bit data frames (32-bit message data, function, or interrupt code, and 1-bit word ID,

1-bit sync) according to procedures defined in MIL-STD-1397. The modules support operation in computer-to-computer, computer-to-peripheral, externally specified addressing modes as described in MIL-STD-1397.

RS-232-C Interface Module (RIM)

The RIM provides a full-duplex RS-232-C serial channel operable at selectable baud rates from 75 to 9600 baud for the asynchronous mode and in synchronous mode, to 9600 baud. See Figure 2-3 for cable configuration.

The module can be converted to operate to MIL-STD-188C with some component changes but without circuit board modifications.

PROTEUS Interface Module (PIM)

The PIM contains the logic to implement a PROTEUS digital channel pair capable of full-duplex data transmission at a nominal 125K-words-per-second rate. The channel is designed to NADC Specification No. A30-15590.

Transmission on the PROTEUS channel is between a source and a sink, with initiation and control by the source. A source transmits 6-bit control frames and 34-bit data words (32 message bits, 1 parity bit, and 1 identifier bit). The sink responds to each source word or frame with an appropriate 6-bit control frame to accomplish a positive handshaking procedure on a word-by-word basis.

Parity is provided on both control and data words for error detection, and retransmission is used for error correction.

The channel pair uses a total of eight differential NRZ signals as depicted in Figure 2-4.

PIC/POC/SOC Module (PPSM)

The PPSM consists of a parallel input channel (PIC), a parallel output channel (POC), and a serial output channel (SOC). The PIC/POC portion of the module performs 32-bit parallel data transfer to and from external devices in full duplex operation. The SOC portion of the module provides serial NRZ data transfers to external devices at a 200-KHz or 1-MHz rate, selectable under program control. The data transfer can consist of any number of 16-bit words.

The PPSM has the following capabilities.

- Request-acknowledge type control logic (PIC/POC)
- Internal and external wraparound test (PIC/POC)
- External halt available for POC (generates an EII)
- Data and control signals are differential (SOC)
- Internal test capability (SOC)
- External suspend line available to regulate data transmissions (SOC).

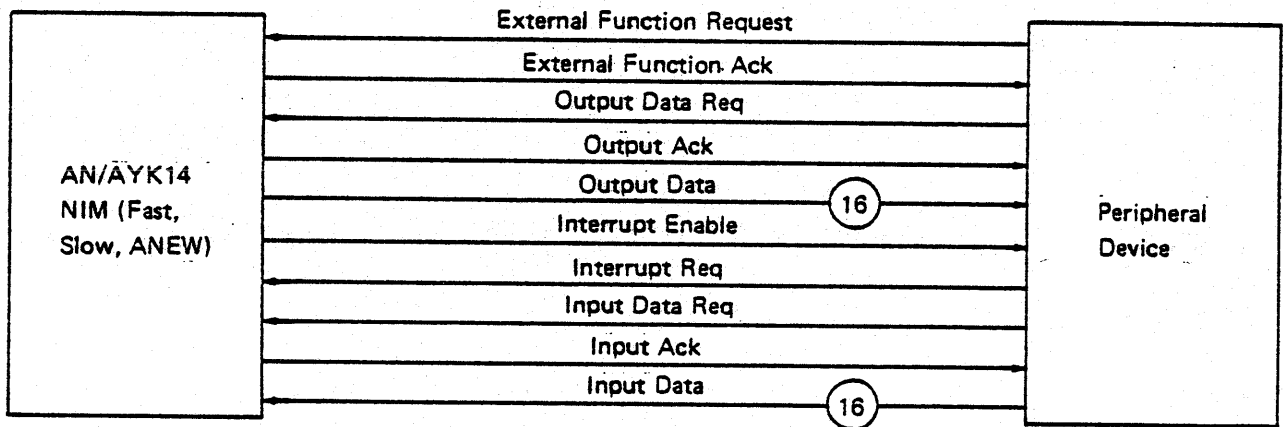
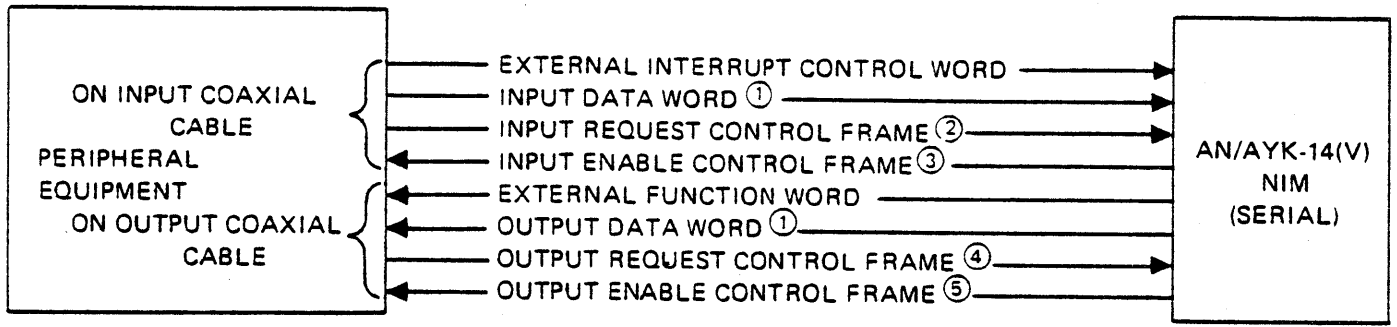


Figure 2-1. NTDS Slow, Fast, and ANEW Channel Interface



ARROWHEADS SHOW DIRECTION OF TRANSMISSION

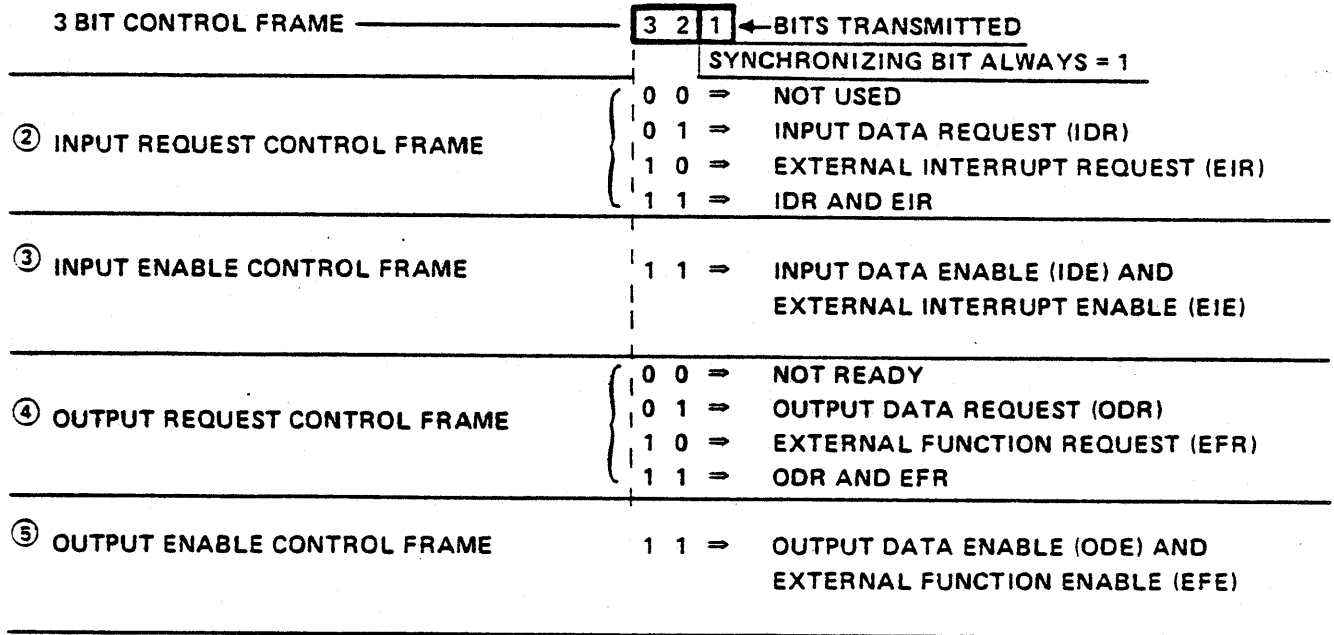
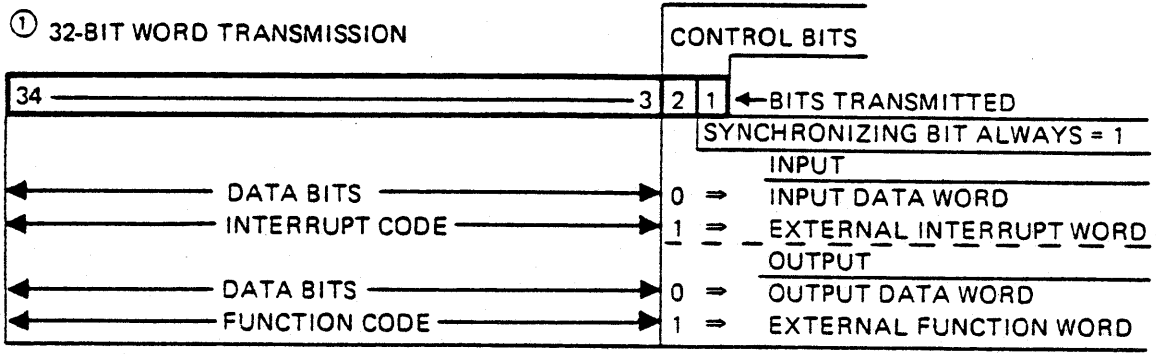


Figure 2-2. NTDS Serial Channel Interface and Message Format

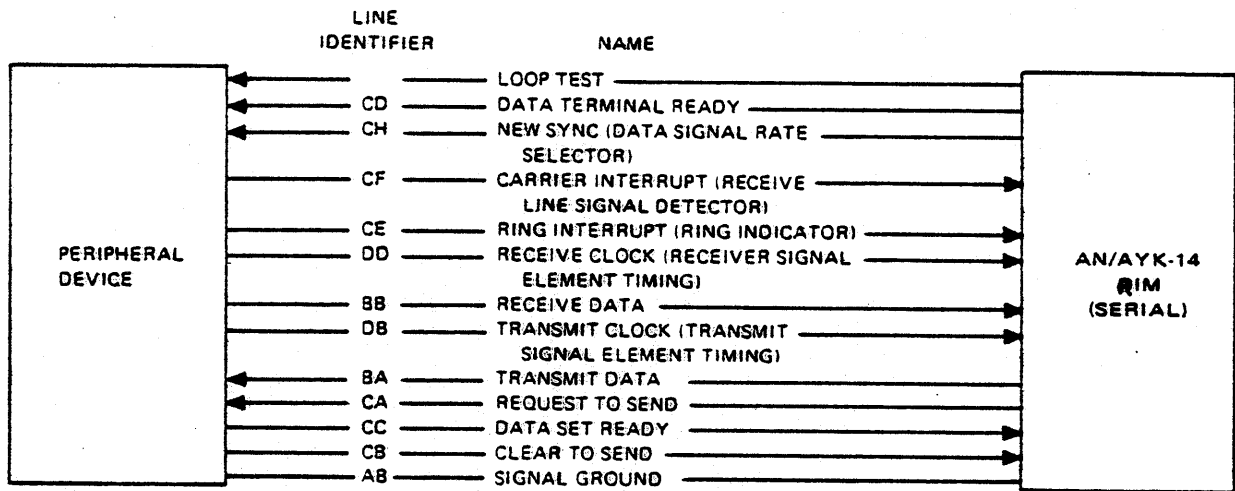


Figure 2-3. RS-232-C Serial Channel Interface

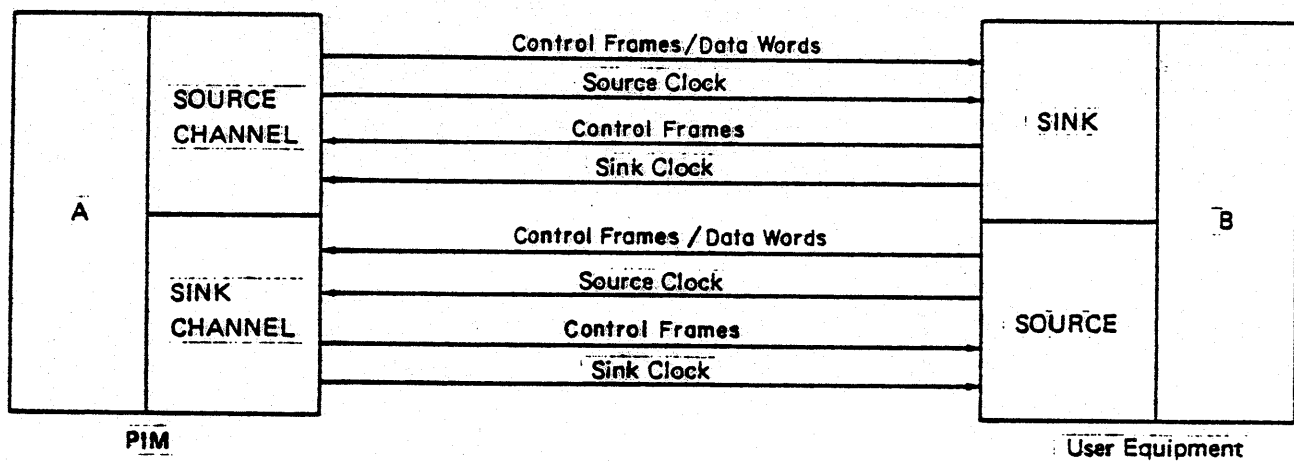


Figure 2-4. PROTEUS Channel Pair

Discrete Input/Output Module (DIOM)

The DIOM provides a full duplex I/O channel capable of initiating both input and output chains which may be active simultaneously.

The DIOM also provides the following interface capabilities.

- 144 output discrettes (114 active low TTL and 30 active low 15V outputs)
- 48 input discrettes (24 active low TTL and 24 active low 15V inputs)
- All 48 inputs may be used as inputs or interrupts, but not both
- The 48 inputs are individually prioritized in a fixed sequence for use as interrupts
- Interrupts are individually masked
- All outputs are internally wraparound testable including the transmitters
- Input scan logic can be tested internally and input receivers tested externally.

BUS EXTENDER MODULE (BEM)

The BEM provides an extension of the internal AN/AYK-14(V) buses and interfaces outside the enclosure to permit extension of memory, processor, and/or I/O subsystems to additional enclosures up to 15 feet (total cable length) from the computer. All voltage levels are TTL compatible and employ differential line drivers/receivers for all I/O lines. The electrical and logical design permits BEM-to-BEM communication. The BEM does not have a channel address as do other I/O modules, but instead appears transparent to bus operation. Any communication via the BEM results in a slight interface delay of approximately 75 nanoseconds in each direction, relative to direct module intercommunication. The BEM can be used to interface a DMA channel.

MEMORY MODULES

Memory Control Module (MCM)

The MCM provides a two-port paged interface to two independent, memory channels allowing simultaneous access by two users. The MCM contains the control, interface, and paging logic to operate core and SMMs with the AN/AYK-14(V) processor system. The MCM features include:

- Interfaces to CPUBUS and IOBUS
- Dual memory bus interfaces to memory modules OMEMBUS and EMEMBUS
- 16-bit address to 19-bit address paging system
- Interleaving of memory modules between memory buses
- Parity bit logic, 1 parity bit per byte
- Block protect in paging system:
 - Read protect
 - Write lockout
 - Execute protect.

Core Memory Module (CMM)

The CMM is available as a 32K by 18-bit word module. The CMM is a plug-in unit containing all of the specified core storage, associated drive and sense electronics, timing and control logic, and interface circuitry. The form-factor and electrical interface of the 32K CMM is identical to the SMM which provides for complete interchangeability.

The CMM features are:

- 900-nanosecond read/write cycle time
- 350-nanosecond access time
- Low power, average 31 watts for 32K words (based on half 1's, 50-percent standby), maximum 64 watts
- Byte operation
- Interface to OMEMBUS or EMEMBUS
- Mountable on 1.45-inch centers
- Read/modify/write capability
- Data guard, indicates power supply out of tolerance (optional).

Semiconductor Memory Module (SMM)

The SMM provides 16K by 18-bit words in a module which is compatible to and interchangeable with the CMM.

The SMM features are:

- 300-nanosecond read access time
- Low power, 16 watts average for 16K words, maximum 30 watts
- Interface to OMEMBUS or EMEMBUS
- Mountable on 1.45-inch centers.

Read/Write Expandable Module (RXM)

The RXM contains 4K words by 18 bits of read/write static semiconductor memory and an optional additional 4K words of read-only memory (ROM or PROM). Features include:

- 400-nanosecond cycle time
- 300-nanosecond access time
- Interface to IOBUS or CPUBUS
- Mountable on 0.45-inch centers
- Parity logic, 1 bit per byte
- If ROM or PROM option is desired, memory contents must be specified at time of order.

POWER CONVERTER MODULE (PCM)

The PCM provides the regulated dc power required to operate AN/AYK-14(V) modules from military aircraft power sources. Two sizes of PCMs are currently available to power various computer configurations. The PCMs are themselves modular, and new capacities can be developed to meet other power source or computer configuration requirements.

The PCMs operate from 115-Vac, three-phase, 400-Hz, wye-connected input power. The design is compatible with MIL-STD-704B and MIL-STD-461A, Notice 3. PCM capacities are given in Table 2-2.

PCM-1 and PCM-2 supply thermal protection and power failure signals as well as sequencing for power-up and power-down operations. Approximately 300 microseconds are available for saving machine state and registers upon input power loss detection.

OPERATION

The AN/AYK-14(V) is designed to be used on applications requiring unattended operation. These operations do not require a computer operator's panel. Instead, the computer is operated indirectly from interfaces provided on the mission system control panel(s).

Operator control for maintenance and program development and debugging is provided by the loader/verifier (L/V) and computer control unit (CCU). These pieces of equipment interface to either the CPU or IOP via the computer support interface, which consists of a high-speed serial channel and is accessible by front panel connector. The L/V is designed for flight line maintenance and program loading. The CCU provides full computer console display and control functions. Initial conditions for the AN/AYK-14(V) are shown in Table 2-3.

The computer support interface is in addition to the 16 AN/AYK-14(V) I/O channels and is not addressable as an I/O channel.

TABLE 2-2. PCM CAPACITIES

Module Type	Maximum Output				Size	Typical Efficiency
	+5V	+15V	-12V	-5V		
PCM-1	48A	3.3A	8.3A	1A	7" by 9" by 3.5"	71%
PCM-2	78A	3.3A	8.3A	1A	7" by 9" by 4.9"	71%

TABLE 2-3. INITIAL CONDITIONS

Register/Condition	Content/State	
Program Address Register	0000	
Program Address Breakpoint	0000	
Operand Address Breakpoint	0000	
Status Register 1	0000	
Status Register 2	0000	
Page Register 0	0000	
Page Register 1	0001	
⋮	⋮	
Page Register 63	003F	
General Register Set 0	All = 0000	} In UYK-20 these are unknown after power-up
General Register Set 1	All = 0000	
Real-Time Clock	Cleared and disabled	
Monitor Clock	Cleared and disabled	
Input/Output Channels	Cleared	
Memory Interface	Cleared*	
All Interrupts	Cleared and locked out	
Built-In Test Timer	Cleared and enabled	
Execution Mode	Running**	
Computer Support Interface		
Channel Busy	Not Busy	

*Memory content is not changed during the initialize sequence

**Execution mode is "stopped" if the computer support equipment is connected

INSTRUCTION FORMATS

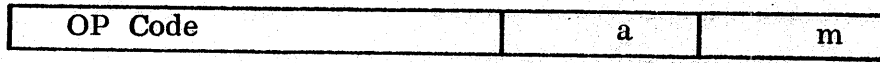
The instruction word formats for the AN/AYK-14(V) include both single-word (16-bit) and double-word (32-bit) types. The 16-bit instructions conserve memory and enhance processing speed while the double word provides memory addressing over the full range of addresses and for in-line storage of constants. Figure 3-1 defines the fields for the four types of instruction formats, and Figure 3-2 defines the associated operand formation. The various instruction types provide for a variety of operand addressing processes including direct, indirect, and indexed types.

Instructions using double-length (32-bit) operands use two sequential registers or memory locations as shown in Figure 3-2. Ra, Rm, or Y contains the most significant portion and sign bit; Ra \oplus 1, Rm \oplus 1, or Y \oplus 1 contains the least significant portion of the operand.

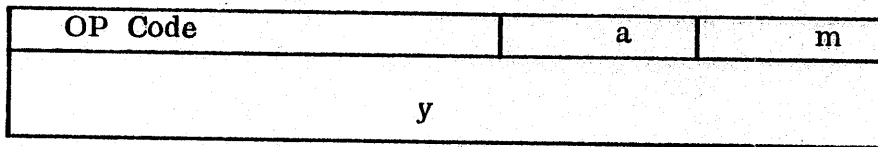
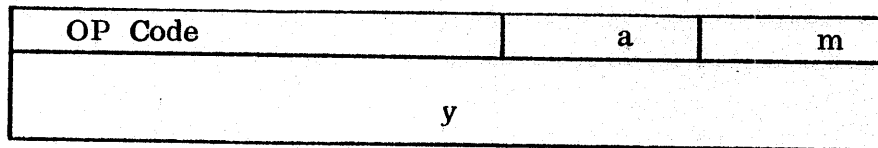
The instruction word formats of the AN/AYK-14(V) are identical to those of the AN/UYK-20.

- RR format instructions use general registers for operands instead of main memory. The a and m designators specify the general registers Ra and Rm, respectively, that are used in the operation.
- RL format instructions perform operations involving one or two general registers. The a designator selects Ra or Ra \oplus 1 depending on the particular instruction. The m designator is a 4-bit unsigned literal which can be used, for example, as a count or increment depending on the particular instruction.
- RI format, type 1 instructions are local jump instructions which increase or decrease (P) by the d value in the instruction. The effective jump address $Y=(P)+d$, where d is the 2's complement deviation value.
- RI format, type 2 instructions perform operations that involve general registers and a main memory reference. The a and m designators select general registers Ra and Rm, respectively. Rm in this case contains an address, Y, that is used for the main memory reference.
- RK format instructions are double-word instructions stored in sequential memory locations. The first word contains the operation code, a, and m designator fields. The second word is the value, y, that may be used as a operand or address.

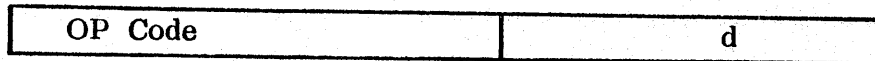
The a designator selects general register Ra. When $m = 0$, the operand or address Y equals y, no Rm is selected. $m \neq 0$ selects a general register Rm; the operand $Y = y + (Rm)$ (i.e., y is indexed



16-bit



32-bit



16-bit

- | | | | |
|---|---|---|---|
| a | General register or subfunction designator | d | Signed deviation value (2's complement) |
| m | General register or subfunction designator or 4-bit literal | y | Address or constant |

Register - Register Format (RR)

Register - Literal Format (RL)

Register - Immediate Format (RI-2)

Register - Constant Format (RK)

(y is a value)

Register - Indirect Format (RX)

(y is an address)

Register - Immediate Format (RI-1)

Figure 3-1. Instruction Formats

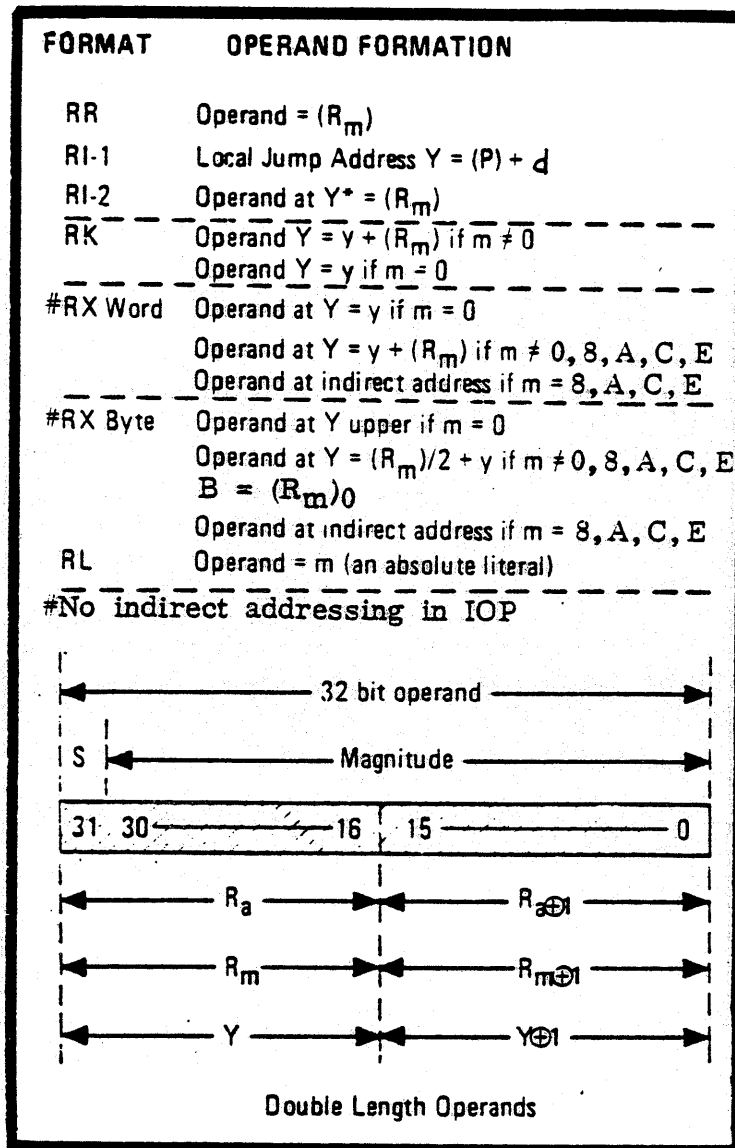


Figure 3-2. Operand Formation

by the contents of Rm). Also, operand Y is used as an address in RK format jump instructions and in remote execute instructions.

- RX format instructions are also double-word instructions. The first word contains the a and m designators, and the next word contains the y value. RX format instructions perform 8-bit byte, whole-word (16-bit), and double-word (32-bit) operations with general registers and main memory. The a designator selects Ra for all three types of operands.

RX instructions provide very flexible operand addressing including direct, indirect, cascaded indirect, and indexed types. The m designator selects the addressing as follows:

m = 0 - direct addressing without indexing

m = 1-7,9,B,D, or F (hexadecimal) - direct addressing with indexing, with the contents of Rm used as the index value.

m = 8,A,C,E (hexadecimal) - address scheme depends on bits set in SR28-15. If indirect addressing is indicated, a pair of indirect words (IW1 and IW2) will be used to define further operand address processing according to the scheme depicted in Figure 3-3. The address of IW1 is $Y = y$, if not indexed, and $Y = y + (Rm)$, if indexed. This process provides a means for cascading indirect addresses as desired.

DATA FORMATS

Data formats include, numbered from right to left, 4-bit literal, 8-bit byte signed or unsigned, 16-bit word signed or unsigned, and 32-bit double-word signed or unsigned. Double words reside in two adjacent registers or memory locations. The first 16 bits of a double word must be contained in an even-numbered register or memory location. The second 16 bits of a double word will then occupy the adjacent odd-numbered register or memory location. Floating-point operand format is shown in Figure 3-6.

ADDRESSING FORMATS

Double-length operands are assigned even-numbered addresses or registers, with the most significant portion in the even-numbered location and the least significant portion in the next location. Memory addressing for the first portion of a double-length operation is formed like that of the single-word operands.

For m values 1 through 7 and 9, B, D, F, the m-field specifies direct addressing Rm as an index register [i.e., operand is located at $Y = y + (Rm)$, where y is the value in the second 16 bits of the instruction]. For m values 8,A,C,E, one of the four pairs of bits in the upper half of status register 2 is checked to determine the addressing scheme to be used for the instruction. If the bit pair in question has values 00 or 01, direct addressing using Rm as an index register is selected. Bit patterns 10 and

11 specify indirect addressing using a pair of indirect words (labeled IW1 and IW2). The pattern 10 selects IW1 at y, and 11 selects IW1 at y + (Rm). Bits 14 through 12 of IW1 (labeled J) specify the interpretation of IW2. For J from 0 through 3, IW2 is interpreted as the address of the operand to be fetched using Rm, Rm + 1 or, Rx (where x is the value in bits 3 through 0 of IW1) as an index register. For J from 4 through 7, IW2 is interpreted as containing the address of a new indirect word after indexing by Rm, Rm + 1, or Rx.

For byte addressing, the least significant bit (LSB)(bit 0) of the indexing register is used to select the upper or lower byte of the word referenced. If the bit is 0, the more significant byte (upper) in the referenced address is selected; if the bit is 1, the less significant byte (lower) is selected. The following cases can arise:

β is the byte designator (0 for more significant byte)

$m = 0$, address $Y = y$ and $\beta = 0$

$m \neq 0$, $Y = y + (Rm)/2$, $\beta = \text{LSB}(Rm)$ before the indexing

under indirect addressing (see Figure 3-3)

$j = 0$, $y = (IW2)$, $\beta = 0$

$j = 1$, $Y = (IW2) + (Rx)/2$, $\beta = \text{LSB}(Rx)$ before the indexing

$j = 2$, $Y = (IW2) + (Rm)/2$, $\beta = \text{LSB}(Rm)$ before the indexing

$j = 3$, $Y = (IW2) + (Rm + 1)/2$, $\beta = \text{LSB}(Rm + 1)$ before the indexing

where m is the m-field of the instruction

y is the second 16 bits of the instruction

Y is the computed address

X is the lower 4 bits of IW1 (see Figure 3-3)

IW2 is the second of the pair of indirect words (see preceding discussion).

Indirect addresses are computed normally. For j value from 4 through 7, the indirect addressing is continued normally until a j value from 0 through 3 is encountered.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IW1	J				Not Used								X			
IW2	y															

J Value	Address Determination
0	Operand at address specified by IW2
1	Operand at address specified by IW2 + (Rx)
2	Operand at address specified by IW2 + (Rm)
3	Operand at address specified by IW2 + (Rm + 1)
4	Another indirect word at address specified by IW2
5	Another indirect word at address specified by IW2 + (Rx)
6	Another indirect word at address specified by IW2 + (Rm)
7	Another indirect word at address specified by IW2 + (Rm + 1)
10 - 17	Not assigned

Figure 3-3. Indirect Addressing Schemes

ARITHMETIC INSTRUCTIONS

INTEGER ARITHMETIC

The AN/AYK-14(V) provides both integer (fixed-point) and floating-point arithmetic. Integer arithmetic can be both single-precision and double-precision. Integer arithmetic formats are shown in Figure 3-4. Two distinct indicators of arithmetic results (the carry and overflow indicators in status register 1) are provided. The carry indicator is set when an arithmetic operation produces a result which carries out beyond the most significant bit (MSB)(either 16-bit or 32-bit operations). The overflow indicator is set when an operation exceeded the capacity of the machine and produces a result of a different sign. Overflow and carry indications for all possible combinations of operands in an addition operation are shown in Figure 3-5. The overflow and carry pattern is the same for 16-bit and 32-bit operands.

FLOATING-POINT ARITHMETIC

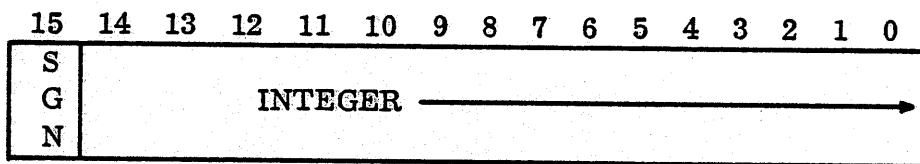
Figure 3-6 shows floating-point operand format. The characteristic is a 7-bit field computed as exponent + 40_{16} , the mantissa is 24 bits (6 hexadecimal digits). Floating-point over/underflow occurs when a normalized result's characteristic is larger than 7 bits and cannot be represented. The AN/AYK-14(V) floating-point instructions accept both normalized and unnormalized inputs and produce normalized outputs. A floating-point number is normalized when the most significant hexadecimal digit is nonzero. Normalization consists of shifting the mantissa left in hexadecimal fashion (4 bits at a time) until the high-order hexadecimal digit is nonzero and reducing the characteristic by 1 for each 4-bit shift.

Floating-point addition, subtraction, multiplication, and division may be performed with a normalized result with or without a residue. If the floating-point residue designator (bit 6) in status register 1 is cleared, the residue is dropped after normalization of floating-point results. If the residue designator is set, the residue is saved after normalization of results.

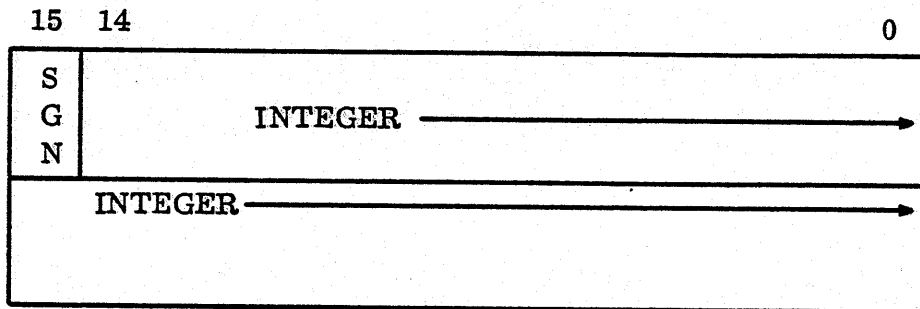
Word 1 of the operand contains the algebraic sign of the fractional mantissa, a biased characteristic in the range $0 \leq C \leq 7F$ (hexadecimal) and the two most significant hexadecimal digits of the fractional mantissa. A normalized floating-point number has a nonzero hexadecimal digit in the most significant 4 bits of the mantissa. When a residue is requested by the program (SR1, bit 6 set), the computer stores the floating-point normalized number in Ra and Ra + 1. The residue in floating-point data format (normalized or unnormalized) is stored in Ra + 2 and Ra + 3. The residue may or may not be normalized as normalization will not be performed on residue.

Floating-point residue for the floating-point subtract and add arithmetic operations will be as follows, if selected.

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the 48-bit result.



Single-Word Arithmetic Operand



Double-Word Arithmetic Operand

Figure 3-4. Integer Arithmetic Formats

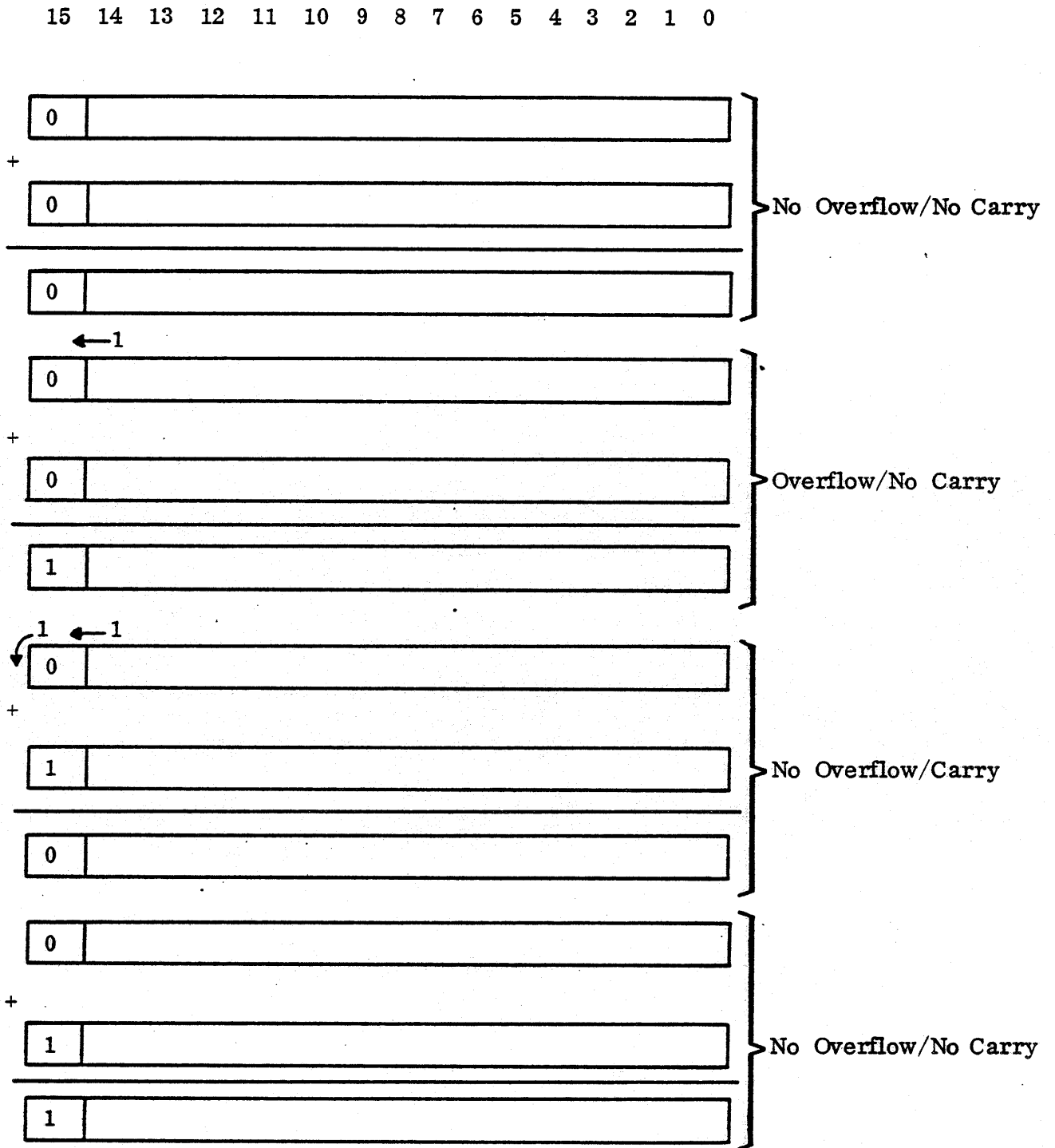


Figure 3-5. Overflow and Carry Indications (Sheet 1 of 2)

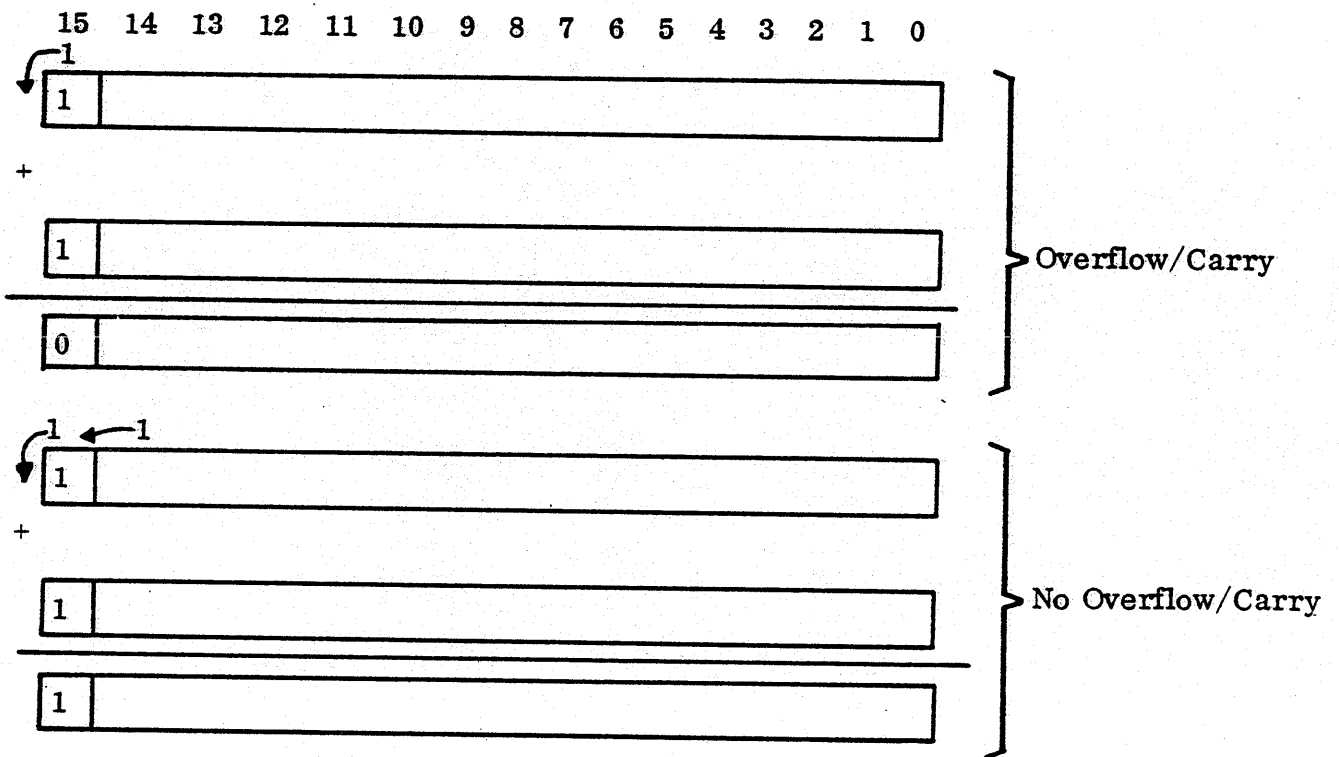
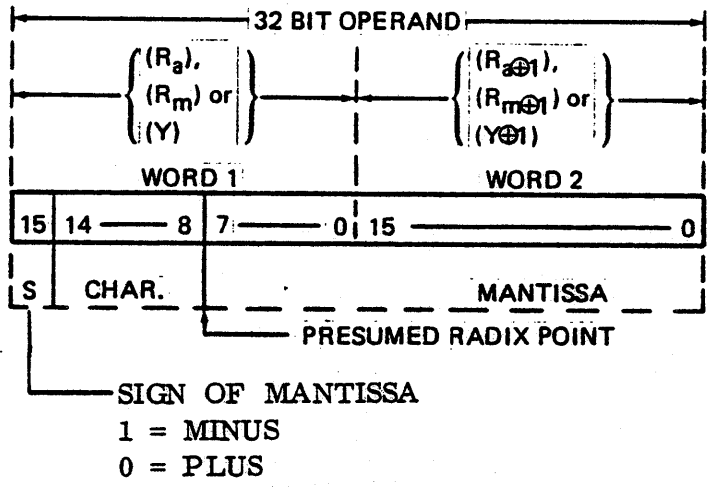


Figure 3-5. Overflow and Carry Indications (Sheet 2 of 2)



NOTE:

a, m, and address Y are even numbers.
Characteristic or exponent is formed
by adding 64 (40_{16}), the bias to the
exponent.

- + exponents above 40 ($40-7F$)
- exponents below 40 ($00-3F$).

Floating-Point Format Examples

+1 ₁₆	4110	0000
-1 ₁₆	C110	0000
-.01 ₁₆	BF10	0000
+.001E ₁₆	3E1E	0000
+3E6476.x16+3	493E	6476

Figure 3-6. Floating-Point Format

- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

Refer to the specific instruction descriptions for floating-point multiply and divide residue.

A change of 1 in the characteristic represents a factor of 16 change in the value of the number and represents a 4-bit position shift of the fractional mantissa. The magnitude range of floating point numbers is approximately $5.4 \times 10^{-79} \leq M \leq 7.2 \times 10^{75}$. A floating-point zero is any data value with a zero mantissa.

Floating-point definitions:

- 1) Floating-Point Zero - any data value where mantissa is zero
- 2) Machine Zero - any data value where sign, exponent, and mantissa are zero
- 3) Machine Infinity - any data value where exponent and mantissa are all ones. Either plus or minus infinity possible.

Machine zero and machine infinity are considered to be valid, normalized floating-point numbers. Refer to Table 3-1 for floating-point special cases.

TABLE 3-1. FLOATING-POINT SPECIAL CASES

Divide Special Case	Result	Residue	Exponent Overflow/Underflow
Infinity/infinity	One (sign is XOR of both input signs)	Machine zero	Overflow
Zero/zero	Machine zero	Machine zero	Overflow
Nonzero/zero	Infinity (with sign of Ra)	Same as result	Overflow
Zero/Nonzero	Machine zero	Machine zero	Neither
Infinity/x	Infinity (sign is XOR of both input signs)	Same as result	Overflow
x/infinity	Machine zero	Machine zero	Underflow
Multiply special case			
Zero times infinity, or infinity times zero	Machine zero	Machine zero	Overflow

TABLE 3-1. FLOATING-POINT SPECIAL CASES (Cont.)

Multiply Special Case	Result	Residue	Exponent Overflow/Underflow
Zero times x, or x times zero	Machine zero	Machine zero	Neither
Infinity times x, or x times infinity, or infinity times infinity	Infinity (sign is XOR of two input signs)	Same as result	Overflow
Add special case			
Infinity + infinity	Infinity (sign is negative if both operands negative, else positive)	Same as result	Overflow
Infinity + non- infinity	Infinity [sign is sign of (Ra, Ra+1)]	Same as result	Overflow
Noninfinity + Infinity	Infinity (sign is sign of (Y, Y+1))	Same as result	Overflow
Infinity - infinity	Infinity (sign negative if (Ra, Ra+1) nega- tive and (Y, Y+1) positive, else positive)	Same as result	Overflow
Infinity - non- infinity	Infinity (sign is sign of Ra, Ra+1)	Same as result	Overflow
Noninfinity - infinity	Infinity [sign is complement of sign of (Y, Y+1)]	Same as result	Overflow

GENERAL REGISTERS

Two stacks, each of sixteen 16-bit general registers are provided on the AN/AYK-14(V). The register stack in use is determined by the general register set designator, bit 14, in status register 1. All register references, fetching or storing data, indexing, etc., involve the register stack currently selected. The contents of the other stack are not affected. All registers are initialized to 0 on power up or after master clear, and register stack 0 is selected.

STATUS REGISTERS

STATUS REGISTER 1

Status register 1 (Figure 4-1) contains designations for system functions and indicates status for instruction executions. Four instructions are provided for modifying and examining the contents of status register 1.

Store SRL (OC a 1) (SSOR a)

The Store SRL instruction stores the value in SRL into the designated register, Ra, and sets the condition codes appropriately. A discussion of condition codes follows.

Load SRL (OC a 5) (LSOR a)

The Load SRL instruction loads SRL with the value in the designated register Ra. This is an executive mode instruction.

Load PSW (Immediate) (1D-m) (LPI m)

The Load PSW instruction loads the contents of memory address $Y^* + 1$ into SRL. This is an executive mode instruction.

Load PSW (Indirect) (1F-m) (LPy, m)

The Load PSW instruction loads the contents of memory address $Y + 1$ into SRL. This is an executive mode instruction.

Functions of status register 1 are as follows:

- 1) Executive Mode Designator - Instructions affecting overall system operation such as page register manipulation, enabling/disabling clock and interrupts, activating I/O chains, loading status registers, and stops are executive mode instructions. The execution of such instructions when not in executive mode causes an executive mode fault interrupt. Operation in executive mode is controlled by bit 15 of status register 1. On power up or master clear this bit is cleared by firmware and the system is placed in executive mode.

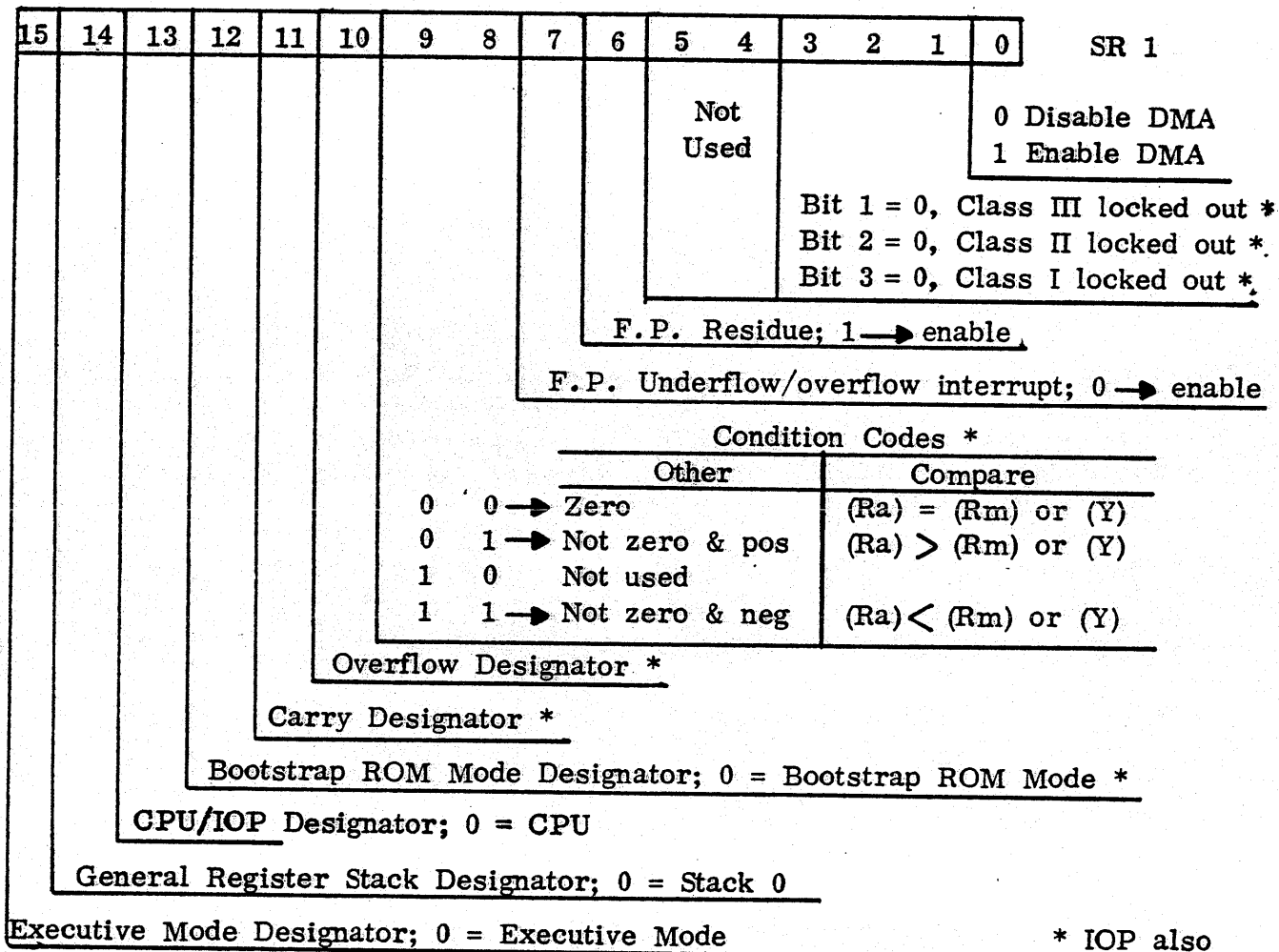


Figure 4-1. Status Register Number 1 Format

The bit is then software controllable with the Load SRI instruction, the Load PSW instructions, and via interrupt generation.

- 2) General Register Set Designator - If 0, this bit selects general register stack 0 and if 1, selects stack 1. Registers in the stack not selected are not affected by instructions. Register designators Ra and Rm within instructions refer to the stack currently selected and do not affect the other stack. All general registers are set to zero on power up or after a master clear, and bit 14 is cleared selecting stack 0. The bit is then software controllable.
- 3) CPU/IOP Designator - If 0, this bit indicates a CPU. If 1, this bit indicates an IOP. This bit is not software controllable.
- 4) Bootstrap ROM Designator - Various system control functions, including the AN/AYK-14(V) bootstrap, are permanently stored in bootstrap ROM memory. This memory can be accessed if the bootstrap ROM designator bit is cleared. The bootstrap ROM memory is located at locations 0 through 3F₁₆ and C0₁₆ through 3FF₁₆ in page 0. If the bootstrap ROM designator is set, all memory references through page 0 will access main memory. On power up or master clear, this bit is cleared selecting ROM mode. The bit is then software controllable.
- 5) Carry - This bit, if set, indicates a carry out of the most significant bit of the adder as a result of a fixed-point arithmetic operation or indicates an underflow condition, when overflow is set, for floating-point arithmetic operations.
- 6) Overflow - This bit, if set, indicates an overflow condition as a result of a fixed-point arithmetic operation or indicates a floating-point arithmetic overflow or underflow condition. If set for a floating-point operation, the carry bit is 0 if overflow and 1 if underflow.
- 7) Condition Codes - Bits 9 and 8 provide the condition codes indicating the results of operations as shown in Table 4-1.
- 8) Floating-Point Over/Underflow Interrupt - This bit, if cleared, allows a floating-point interrupt to be generated as a result of a characteristic out-of-bounds condition in a floating-point arithmetic operation. Cleared on power up or master clear. It is then software controllable.
- 9) Floating-Point Residue - When 1, the residue resulting from a floating-operation is saved. When 0, the residue is discarded. On power up or master clear, the bit is cleared. It is then software controllable.
- 10) Interrupt Lockout Designators - Bits 3 through 1 are the interrupt lockout designators; bit 3 for Class I interrupts, bit 2 for Class II interrupts, and bit 1 for Class III interrupts. When cleared,

TABLE 4-1. STATUS REGISTER 1 CONDITION CODES

Condition Code		Indicated Results of	
Bit 9	Bit 8	Stores, Shifts, Loads, Logicals, or Arithmetic Operations	Compare Operation
0	0	Zero	(Ra)=(Rm) or (Y) or m or y, bit=0
0	1	Nonzero and positive	(Ra) > (Rm) or (Y) or m or y, Bit≠0
1	0	Not used	Not used
1	1	Nonzero and negative	(Ra) < (Rm) or (Y) or m or y, Bit 15≠0

these bits designate that the respective class of interrupts is locked out. They are cleared at power up or master clear and then are software controllable.

- 11) DMA Designator - When set, this designator allows direct memory access (DMA) to AN/AYK-14(V) main memory from external equipment using the BEM. The AN/AYK-14(V) hardware operation and maintenance manuals provide details. On power up or master clear, this bit is cleared, disabling the DMA. The bit is then software controllable with the Load SR1 instruction.

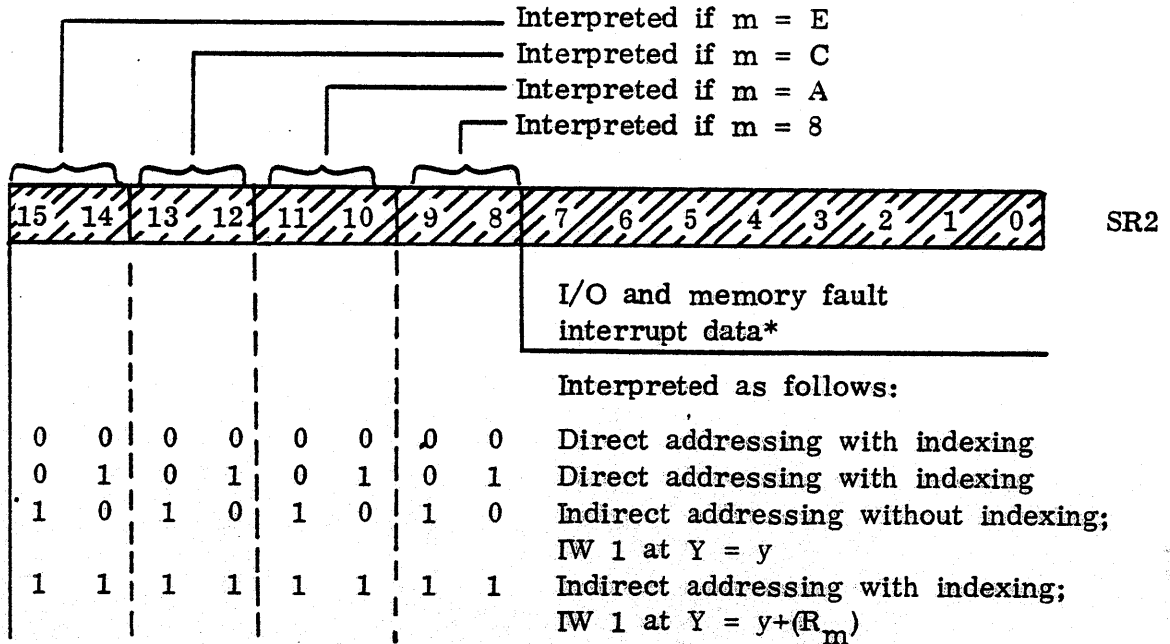
STATUS REGISTER 2

Status register 2 (Figure 4-2) contains designators for addressing schemes and information pertinent to various interrupt types. The following four instructions are provided for modifying and examining the contents of status register 2.

- 1) Store SR2 (OC a 2) (SSTR a) - Stores the value in SR2 into the designated register, Ra, and sets the condition codes appropriately.
- 2) Load SR2 (OC a 6) (LSTR a) - Loads SR2 with the value in the designated register Ra. This is an executive mode instruction.
- 3) Load PSW (Immediate) (1D-M) (LPI m) - Loads the contents of $Y^* + 2$ into SR2. This is an executive mode instruction.
- 4) Load PSW (Indirect) (1F-m) (LPy, m) - Loads the contents of $Y + 2$ into SR2. This is an executive mode instruction.

The functions of status register 2 are as follows:

- 1) Addressing Techniques - Bits 15 through 8 of status register 2 determine the addressing scheme to be used by the software. When the m-field in an RX format instruction is B, A, C, or E, bits interpreted specify direct addressing with indexing or indirect addressing with or without indexing.



*Interpreted as follows:

7	6	5	4	3	2	1	0
C	C	C	C	-	-	-	-
C	C	C	C	0	X	1	0
0	0	0	0	0	0	0	1
M	M	M	M	X	0	0	1
M	M	M	M	X	0	1	0
M	M	M	M	X	1	0	0
M ₀	M ₁	M ₂	M ₃	M ₄	M ₅	M ₆	M ₇

**I/O failure interrupt
 **I/O chain instruction fault
 CCCC = Chan No., X = 0 → Input, X = 1 → Output
 **I/O command instruction fault
 Memory time-out
 Memory parity error
 **Memory protect fault

MMMM = Module bank code
 X = 0, 1 Memory bus indicator
 0 = Even bus

**IOP also

Time-out or parity error (RXM)

M_i = 0 Bank i no error, M_i = 1 Bank i error

Figure 4-2. Status Register Number 2 Format

- 2) Interrupt Information - Bits 7 through 0 of status register 2 provide information regarding various interrupts. For details on generation of interrupts, see Section 5.
- 3) I/O Failure Interrupt - When an I/O failure interrupt occurs (software references a channel not provided in the hardware configuration), the firmware places the channel number referenced in bits 7 through 4 of status register 2.
- 4) I/O Command Instruction Fault - When an I/O instruction fault interrupt occurs (execution of an illegal command from the I/O command cell), the firmware inserts 0's in bits 7 through 4 and 0001 in bits 3 through 0 of status register 2.
- 5) I/O Input Chain Instruction Fault - When I/O instruction fault interrupt occurs (illegal instruction in a chain program), the firmware inserts the number of the channel whose chain program committed the fault in bits 7 through 4, 0010 in bits 3 through 0 of status register 2 if it was an input chain, and 0110 in bits 3 through 0 if it was an output chain.
- 6) Memory Fault Interrupts - For the three types of memory fault interrupts, the firmware inserts the module bank code (upper 4 bits of the 19-bit absolute address computed by the MCM) in bits 7 through 4. Bit 3 specifies whether the failure occurred out of an address accessed by the odd or even memory bus (bit 3 = 0 for even bus). The module bank code and odd/even bus indicator, together with the memory configuration and the type of interleaving (word/bank) specify the memory module where the failure occurred. For the memory time-out interrupt, the firmware sets bit 0; for memory parity error, the firmware sets bit 1; and for memory protect fault, the firmware sets bit 2.

If multiple failures occur on one memory reference, more than 1 bit in bits 2 through 0 will be set, depending on the faults which occurred. In such cases, only the highest priority interrupt is generated. Hence, a memory time-out interrupt may also indicate a memory parity error or a memory protect fault. Section 9 contains more details on interrupts.

BREAKPOINT REGISTERS

Two breakpoint registers are provided in the AN/AYK-14(V): a program address breakpoint and an operand address breakpoint. The CCU allows the user to specify breakpoint values and to enable and disable the breakpoints (see CCU user's manual). The breakpoint values are 16-bit paged addresses interpreted according to the current page register configuration. Program address breakpoints must be set to the address of the first word of two-word instructions. For program address breakpoints, the AN/AYK-14(V) will stop with (P) = breakpoint value and the instruction at P has not been executed. For operand breakpoint, the AN/AYK-14(V) will stop with (P) = breakpoint + 1 and the instruction at P has been executed. There is no operand breakpoint for the IOP.

PROGRAM ADDRESS REGISTER

The AN/AYK-14(V) 16-bit program address register provides the 16-bit address of the instruction being executed. However, due to the overlap of instruction execution and instruction fetching, and due to simultaneous execution of CPU and I/O instructions, the program address counter value at a given time may not be relevant to program status.

For certain interrupts, the value of the program address register may not indicate the exact instruction at which the interrupt occurred.

The interrupts where P equals something other than the address of the instruction at which the interrupt occurred are as follows:

Class II

Priority 1 - CP instruction fault - address + 1 or address + 2

Priority 2 - I/O instruction fault - address + 1

Priority 3 - Floating-point over/underflow - address + 1

Priority 4 - Executive return - address + 1

The event representing the interrupt may not be detected until after the instruction has been executed. For the CPU instruction fault, the program address will be the address of the instruction causing the fault.

The lookahead instruction read performed concurrently with current instruction execution requires caution in certain coding sequences. Instructions which can affect the next instruction to be read (e.g., a modification of page registers) should be avoided. Since the next instruction is read while the current instruction is executed, the modification does not take place until after the next instruction is read. Similar coding sequences which affect the address of the next instruction should be avoided.

AN/AYK-14(V) CLOCKS

REAL-TIME CLOCK

The AN/AYK-14(V) real-time clock (RTC) in the CPU is a 32-bit register which increments at a 1-MHz rate. When the clock is counting and the RTC overflow interrupt is enabled, the clock interrupt is generated each time the lower 16 bits are incremented end around from FFFF to 0000 (every 2^{16} microseconds). On power up or after a master clear, both the clock and its interrupt are disabled.

The RTC in the IOP is a 16-bit register which increments every 1024 microseconds when enabled. The 1024-microsecond time base is taken from a 16-bit register being incremented at a 1-MHz rate. This register is incremented whenever power is applied and cannot be controlled by software. When the RTC register is enabled and the RTC interrupt is enabled, the interrupt will be generated each time the 16 bits increment end around to zero. Master clear disables the RTC counting and RTC interrupt. IOP instructions are

provided to load and store clock values, and to enable and disable the clock count and RTC interrupt.

Software instructions are provided to load and store clock values and to enable and disable the clock count and the clock interrupt:

- 1) Store RTC (OC a 3) (SCR a) - Stores the contents of the least significant 16 bits (lower half) of the 32-bit real-time clock register into Ra and then sets the condition code.
- 2) Load RTC Lower (OC a 7) (LCR a) - Loads the value in Ra into the lower 16 bits of the RTC. The count and interrupt status (enabled or disabled) are not affected. This instruction is an executive mode instruction.
- 3) Enable Clock (OC - 8) (ECR) - Enables the clock count and the clock overflow interrupt. The clock oscillator operates independently of the clock count. Hence, counting begins at the point in the oscillator cycle where the count is enabled. This instruction is an executive mode instruction. Do not use this instruction in interrupt handlers other than the RTC interrupt handler.
- 4) Disable Clock (OC - 9) (DCR) - Disables the clock count and the clock overflow interrupt. This instruction is an executive mode instruction.
- 5) Load Double and Enable Clock (OC a C) (LCRD a) - Loads the values in Ra and Ra + 1 into the 32-bit clock and enables the clock count. Clock interrupt status (enable/disable) is not affected. This is an executive mode instruction.
- 6) Store RTC Double (OC a D) (SCRD a) - In the CPU, stores the contents of the 32-bit real-time clock register into Ra and Ra ⊕ 1. In the IOP, stores the contents of the RTC register into Ra and the 16-bit 1-MHz incrementing register value into Ra ⊕ 1. Clock count and interrupt status (enabled/disabled) are not affected.
- 7) Enable Clock Interrupt (OC - E) (ECIR) - Enables the RTC overflow interrupt which is generated when the lower 16 bits of the clock increment to all 0's. Clock count status (enable/disable) is not affected. This is an executive mode instruction. Ensure that this instruction is not the last instruction in a page in memory.
- 8) Disable Clock Interrupt (OC - F) (DCIR) - Disables the RTC overflow interrupt. Clock value and count status (enable/disable) are not affected. This is an executive mode instruction.

MONITOR CLOCK

The AN/AYK-14(V) 16-bit monitor clock is decremented at a 10-kHz frequency. When the clock is counting and the monitor clock interrupt is enabled, the monitor clock interrupt is generated when the clock decrements to 0. On power up, after a master clear, or after a monitor clock interrupt, both the

clock count and the interrupt are disabled. Software instructions are provided to load and store clock values and to enable and disable the clock count and the clock interrupt:

- 1) Load and Enable Monitor Clock (OC a A) (LEM a) - Loads the monitor clock with the value in the designated register Ra and enables the clock count and clock interrupt. The clock oscillator operates independently of the clock count. Hence, counting begins at the point in the oscillator cycle where the count is enabled. This is an executive mode instruction.
- 2) Disable Monitor Clock (OC - B) (DM) - Disables the monitor clock count and the clock interrupt. This is an executive mode instruction.
- 3) Store Monitor Clock (10 a 4) (SMC a) - Stores the current monitor clock value into the designated register Ra. Clock count and interrupt status (enable/disable) are not affected.

BUILT-IN-TEST COUNTER

The built-in-test (BIT) counter in the CPU is a 4-bit counter which is incremented once every 2^{21} microseconds (approximately 2.097 seconds) by the built-in-test timer. The time base is a 1-MHz oscillator on the PSM. The timer is free running and cannot be enabled or disabled by software. On power up or after a master clear, the BIT timer is cleared, the BIT count reset to 0, and counting is begun. The RESET BIT TIMER instruction allows the software to reset the BIT count to 0, queue count to 0, and clear the timer.

Each increment of the BIT counter generates the hardware fault warning interrupt; when the counter increments to all 1's, the hardware fault interrupt is generated. There are two methods of inhibiting the hardware fault warning interrupt. If Class I interrupts are locked out (see Section 9 on interrupts), up to 13 successive hardware fault warning interrupts are placed in a one-level queue. The fourteenth interrupt, however, is not inhibited and is received by the software. The same result can be achieved by executing the Diagnostic Jump instruction with bits 15 and 0 of R15 set. By this method, the other Class I interrupts need not be locked out. Executing the Diagnostic Jump with bit 15 of R15 set and bit 0 of R15 cleared, enables the interrupt for generation at each increment of the counter. If software execution is stopped and if CSE is connected to the AN/AYK-14(V), generation of the hardware fault warning interrupt is inhibited by the constant clearing of the BIT timer by the firmware. The following two instructions are associated with the BIT timer and BIT indicator.

- 1) Reset Bit Timer (08 - E) (RBT) - Resets the BIT count to 0, queue count to 0, and resets the BIT timer.
- 2) Set Bit Indicator (08 - F) (SBT) - Sets the external BIT indicator which, in turn, generates the hardware fault interrupt. It also resets the BIT counter, queue count, and BIT timer.

The built-in-test (BIT) counter in the IOP is a 3-bit counter which is incremented by the BIT timer. The BIT timer time base is 1 MHz. The timer is free running and cannot be enabled or disabled by software. On power up or after a master clear, the BIT timer is cleared, the BIT counter is reset to zero, the queue count is reset to zero, and counting is begun. When the IOP is in controller mode, the firmware will keep the BIT timer from timing out and incrementing the BIT counter which generates the hardware fault warning interrupt. When the IOP is in processor mode, it is a responsibility of the software via the RESET BIT TIMER instructions to keep the BIT timer from generating the interrupt.

Each increment of the BIT counter generates the hardware fault warning interrupt; when the counter increments to all 1's, the hardware fault interrupt is generated if an RBT instruction is not executed within .9375 seconds.

The same two methods of inhibiting the hardware fault warning interrupt apply in the IOP. If inhibited, six hardware fault warning interrupts are placed in a one-level queue, and the seventh interrupt is received by the software.

MEMORY SUBSYSTEM

The AN/AYK-14(V) memory subsystem includes the MCM and memory modules, either core or semiconductor memory. The MCM maintains page registers, computes absolute addresses using the established page configuration, and performs memory protection functions. This module also provides parity checking on data stored in memory and informs the firmware of memory failures. Memory modules are arranged on two memory buses labeled odd and even. Two interleaving methods are available: bank interleaving in which memory modules as units are accessed on one or the other memory bus, and word interleaving in which successive words in memory are accessed on alternate buses. Traffic on the two memory buses can occur independently and simultaneously to increase memory reference speed.

The IOP can access memory on the RXM without using the MCM when an RXM module is included in the system. IOP memory references through the MCM use the current paging as established by the CPU. The CPU cannot reference RXM memory.

INTERLEAVING

Word or bank interleaving provides alternate use of the memory buses to reduce access time during multiple memory references. The type of interleaving (word or bank) is determined by the hardware configuration, and number and size of memory modules.

Bank size can be 16K or 32K depending on the mix of memory module sizes used in a given configuration. Where there is a mix of memory module sizes (i.e., 16K and 32K), the bank size is 16K.

WORD INTERLEAVING

Word interleaving means that memory references are alternated between odd and even memory buses (i.e., word zero from the even bus and word one from the odd bus). In the case where there is a memory module configuration of 32K in location 0 and 32K in location 1 (locations 0 and 1 are hardware slots associated with even and odd memory buses), word interleaving is automatic. The first 32K will contain even memory addresses 00000 through OFFFE, and the second 32K will contain odd memory addresses 00001 through OFFFF. In a configuration such as this, word interleaving can be changed to bank interleaving by inserting a jumper wire between pins 86 and 87 of J01 [front connector on the AN/AYK-14(V)] for an XN-1.

BANK INTERLEAVING

Bank interleaving means that memory references are made to a bank of addresses on the even bus and then a bank of addresses on the odd bus (i.e., words 0 through n from the even bus and words n + 1 through x from the odd bus). In the case where there is a memory module configuration of 32K in

location 0 and 16K in location 1, bank interleaving is automatic. Bank size is defined as 16K, the smaller memory module size. Bank 0 is the first 16K of the 32K module, bank 1 is the 16K module, and bank 2 is the last 16K of the 32K module. Bank 0 would be addresses 00000 through 03FFF, bank 1 would be addresses 04000 through 07FFF, and bank 2 would be addresses 08000 through 0BFFF.

RXM ADDRESSING

This memory is unpagged and does not interface with the MCM. It is intended to operate directly with a processor via the IOBUS interface. Multiple RXMs can be used in a system up to a total of 65,536 words; however, the present AN/AYK-14(V) chassis [100 series (XN-1) and 300 series (XN-3)] provides space for only one RXM each. The primary application of RXMs is to provide memory functions for small AN/AYK-14(V) configurations using the IOP as a standalone processor. An RXM can also be used as a private program memory for the IOP when used in configurations employing both the IOP and CPU in combination. In the latter case, the CPU will not have access to the RXM. When installed in the 100 series or 300 series chassis, the RXM is assigned address ranges F000 to FFFF (hexadecimal) for the RAM portion and E000 to EFFF (hexadecimal) for the optional PROM portion.

BOOTSTRAP ROM ADDRESSING

The bootstrap loader routine resides in ROM and occupies approximately 896_{10} locations.

In order to reference ROM locations, SR1, bit 12 must be zero. In bootstrap mode, paging is not used with ROM locations.

Use of page 0 (ROM) for loading data is limited to non-ROM locations (i.e., 40_{16} -BF $_{16}$). When in bootstrap mode and a location outside of the ROM area is referenced, that location in main memory is accessed using paging. Also, when in bootstrap mode, main memory locations equivalent to the ROM area are not accessible using page register zero. Figure 5-1 shows the relationship between ROM and main memory addresses along with module execution of relative address references.

Interrupt trap locations are established by the bootstrap loader prior to exit. That way, should an interrupt occur after exit and before user intervention, the interrupt will be handled by the bootstrap ROM program.

MEMORY INTERLOCK

To facilitate interprocessor communications through main memory in systems with two processors (CPUs, IOPs, or both), a memory interlock function is provided for the stack, queue, and biased fetch instructions. The interlock applies to a page register accessed by a processor during execution of one of these instructions. When the page register is accessed by a processor, the MCM reserves further accesses to that page register for that processor. When the instruction is completed, the page register is released. During the interlock period, accesses to the page register by the other processor are held until the page register is released. After release, accesses by other processors are allowed. The memory interlock facilitates the stack,

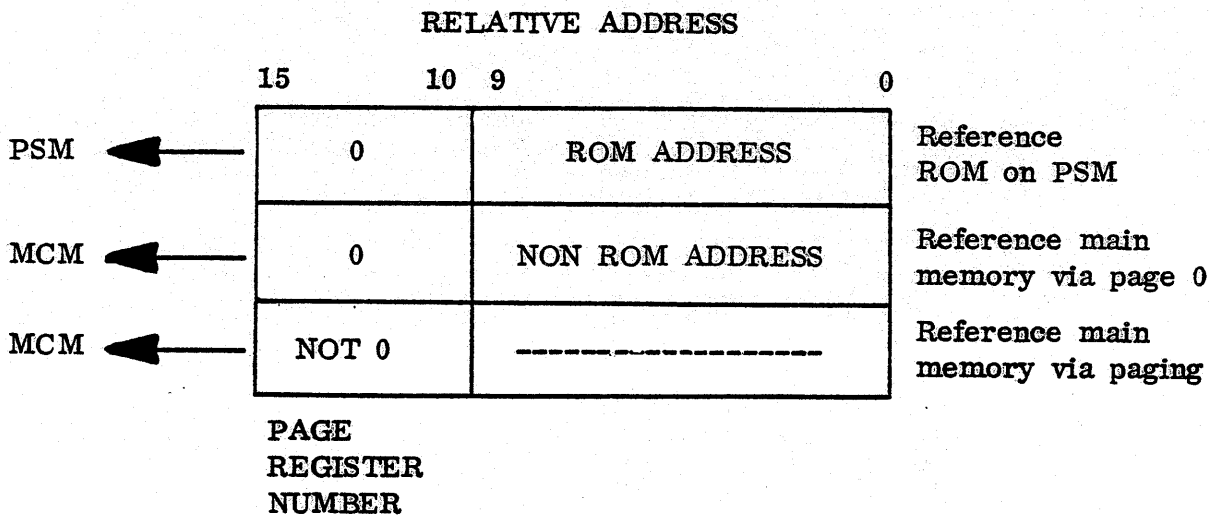
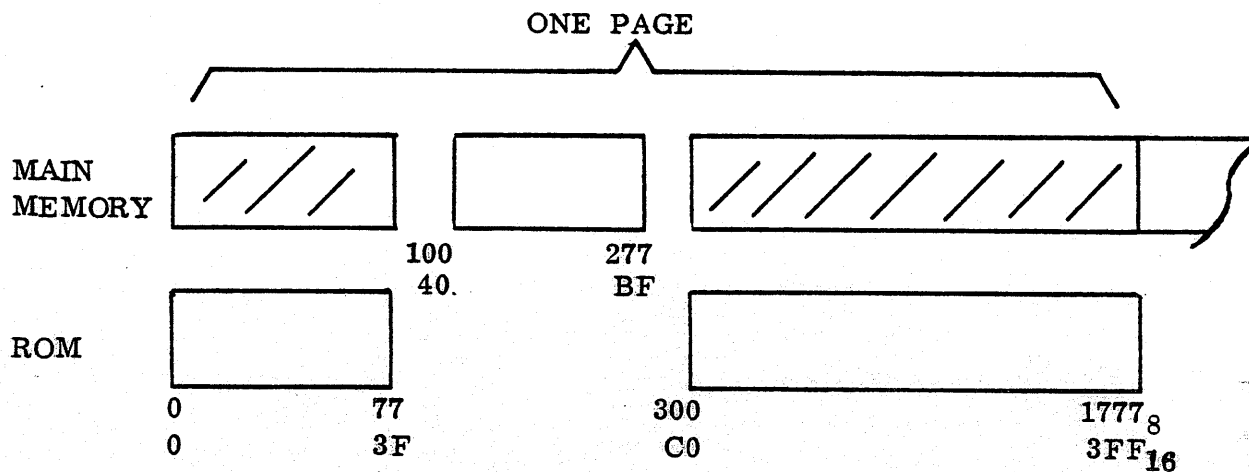


Figure 5-1. Bootstrap ROM Addressing.

queue, and biased fetch instructions in maintaining lists of tasks to be performed by the processors in the system.

PAGING

PAGING TECHNIQUE

The AN/AYK-14(V) CPU can access up to 512K words of memory using a paging system.

The MCM forms the 19-bit absolute memory addresses by concatenating fields from a 16-bit address supplied by the executing software and from one of 64 page registers as follows.

The 16-bit software address is broken into two fields. The lower 10 bits (bits 9 through 0) are used as the lower 10 bits of the absolute address. The upper 6 bits of the software address specify 1 of 64 page registers (0 through 63). Each page register contains 16 bits. The lower 9 bits (bits 8 through 0) of the selected page register are concatenated with the lower 10 bits of the software address, providing a 19-bit absolute address (Figure 5-2).

Effectively, each page register can point to one of 512 1024-location segments of memory (called pages). The page is specified by the lower 9 bits of the page register. Therefore, the 64 page registers can be set to point to a maximum of 64K of memory at one time. To access other portions of memory, the area of memory pointed to by the page registers must be changed. Page register values are modified using the Load Address Register instructions. (These instructions are executive mode instructions).

PAGE REGISTER 0

Page register 0 is used for a number of system functions and therefore requires special handling.

All firmware interrupt memory references, occur through page register 0. This includes storage of current program status word (PSW) and reloading of the interrupt routine PSW. Hence, if page register 0 is modified to point to other areas of memory, the interrupt information must be reloaded in the new area of memory.

If the bootstrap ROM designator (bit 12 of status register 1) is cleared, page 0 locations 00₁₆ through 3F₁₆ and C0₁₆ through 3FF₁₆ access ROM memory. These special locations contain the bootstrap loaders and other system control features. Attempts to write into ROM memory cause data to be lost without notification to the executing software. Page register zero references access to ROM memory if the ROM designator is 0, regardless of the location to which page register 0 is set. Main memory at absolute locations 00₁₆ through 3F₁₆ and C0₁₆ through 3FF₁₆ can be referenced through page registers other than page register 0, regardless of the status of the ROM designator.

Software Address Y (16 bits)

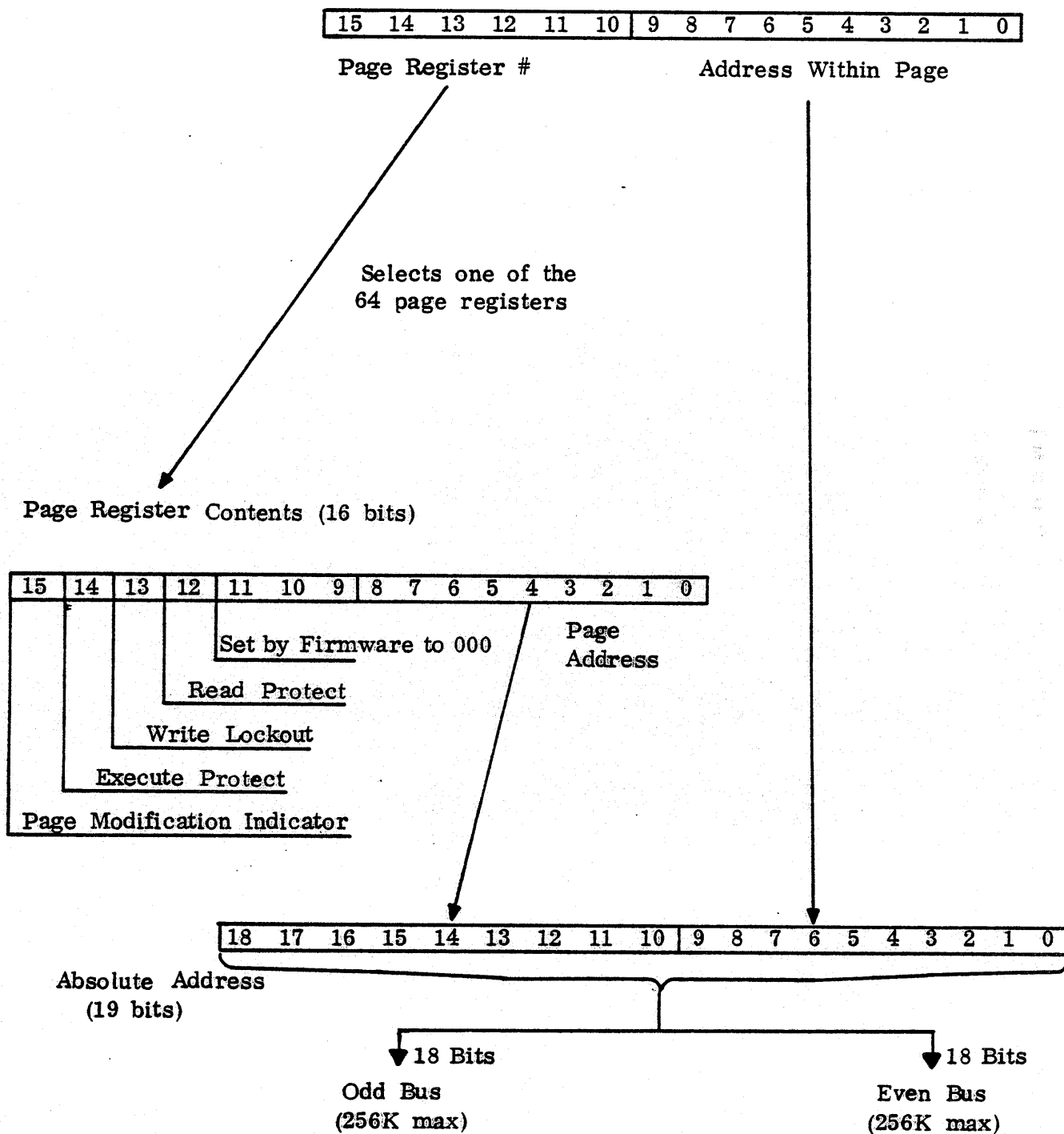


Figure 5-2. Memory Address Generation

MEMORY PROTECTION

The upper 4 bits of each page register (bits 15 through 12) perform the memory protection function. Three types of protection (execute protect, read protect, and write lockout) are provided. Descriptions of the three types of protections follow.

- 1) Execute Protection - When the execute protect bit (bit 14) of a page register is set, the memory protect fault is generated if an instruction or an indirect word is read from the page. The instruction read will be executed. Firmware interrupt processing and execution out of bootstrap ROM memory do not cause an interrupt if execute protect is set in page register 0.
- 2) Write Lockout - When the write protect bit (bit 13) of a page register is set, the memory protect fault is generated if an attempt is made to write into that page of memory. The write is inhibited. Firmware interrupt processing in page 0 does not cause an interrupt if the write lockout is set in page register 0.
- 3) Read Protection - When the read protect bit (bit 12) of a page register is set, the read protect fault will be generated if an operand is read from this page. Indirect word reads and instruction reads do not generate the interrupt. Firmware interrupt processing in page 0 does not cause the interrupt if read protect is set in page register 0.

PAGE MODIFICATION INDICATOR

The page modification indicator (bit 15) in a page register is set when a write operation is performed on the page of memory. The modification bit can be cleared only by a Load Address Register instruction or a master clear of the system.

When operating with a BEM and accessing memory via direct memory access (DMA), the only bit in the page registers that can be modified is the page modification bit (bit 15). It will be set for write operations and can only be cleared by a master clear of the system.

CAUTIONS ON MEMORY PROTECTION

The AN/AYK-14(V) overlaps the execution cycle of the current instruction with the reading of the next sequential instruction (or next 16 bits of a 32-bit instruction). This overlap requires caution in the use of memory protection features.

An example of a problem that may arise is given by the execution of an RR jump instruction (16 bits) located at a page boundary. If the next page is set for execute protect, the interrupt is generated even though the jump command sends the program into an unprotected page. In general, execution of instructions at boundaries between unprotected and protected memory should be performed with care.

In an IOP standalone operation, only one-half of memory is available. The IOP does not have page registers, indirect addressing, paging, or memory protection.

Details on the effect of the emulator instruction cycle overlap are discussed in Section 4.

COMPUTER SUPPORT FUNCTIONS

A portion of the AN/AYK-14(V) microcode is devoted to routines which provide the computer support functions. The CCU sends messages containing function requests to the AN/AYK-14(V). The function requests are combined by the CCU into a high-level interface for use in software development and controlling computer operations. Details may be found in the CCU User's Manual.

SUPPORT CHANNEL OPERATIONS

The AN/AYK-14(V) can communicate with peripheral equipment over a special RS-232-C type communications link supported by the CCU. This link can be operated at rates up to 4800 baud for one-way transmission to the AN/AYK-14(V) (no status) transmission and up to 2000 baud for full duplex communications. The AN/AYK-14(V) initiates all activity on the link. While a transfer between the AN/AYK-14(V) and the CCU is in progress, the SUPPORT CHANNEL BUSY (SCBY) indicator is set. The AN/AYK-14(V) clears this indicator when the transfer is completed. Data to be transferred across the link is placed in the support channel buffer (upper byte) where it can be accessed by the CPU or CCU. Five instructions are provided in the AN/AYK-14(V) to drive communications across this link. These five instructions become NOPs when executed in the AN/AYK-14(V) when it is not connected to the CCU.

- 1) Support Channel Input (08 a 7) (SCI a) - Transfers the contents of the support channel buffer to bits 15 through 8 of the register designated by Ra.
- 2) Initiate Support Channel I/O (0D a m) (SCIO a,m) - Performs one of the following functions depending on the m-field:
 - m - 0000: Request for one byte of input data across the support channel to the support buffer bits 15 through 8. The SUPPORT CHANNEL BUSY indicator is set until the data is received.
 - 0100: Transfers bits 15 through 8 of register Ra to the support channel buffer, and requests output of that byte across the support channel. The SUPPORT CHANNEL BUSY indicator is set until the data is transferred and acknowledged.
 - 1000: Request for one byte of status across the support channel to the support channel buffer bits 15 through 8. SUPPORT CHANNEL BUSY indicator is set until the data is received.

1100: Transfer bits 15 through 8 of register Ra to the support channel buffer, and requests output of that byte across the support channel. The SUPPORT CHANNEL BUSY indicator is set until the transfer is acknowledged. The byte will be interpreted by the CCU as either a mode or command byte.

0001: Reserved for shop replaceable assembly (SRA)

0101: communications with computer support equipment.

0010: Transfers bits 15 through 8 of register Ra to the support channel buffer, and requests output of that byte across support channel. The SUPPORT CHANNEL BUSY indicator is set until the byte is transferred. The CCU will interpret it as a speed select byte.

- 3) Jump Support Channel Busy RR, RK, RX (80 C m; 82 C m; 83 C m)
(JSCR m; JSC y, m; JSC *y, m) - Tests the SUPPORT CHANNEL BUSY indicator. If the indicator is set, the jump is executed; if the indicator is cleared (transfer complete), the next sequential instruction is executed.

To use this channel, the user must write his driver programs, select speed, parity, stop bits, do the handshaking, and transfer the data.

Codes used in controlling the support channel are expected by the CCU in the following sequence:

Mode byte (upper byte of
the designated register)
sets up the CCU

7	6	5	4	3	2	1	0
STOP		PARITY		LENGTH		FACTOR	

Bits

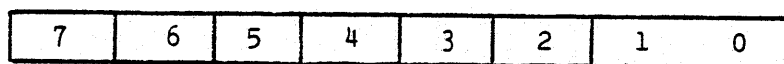
76 01 = 1 stop bit, 11 = 2 stop bit
54 00 = No, 01 = odd, 11 = even
32 00 = 5, 01 = 6, 10 = 7, 11 = 8
10 00 = Sync, 10 = Async

Command byte (upper byte of the designated register) directs the CCU



- 7 Not used (=0)
- 6 Reset to mode (next byte is a mode byte)
- 5 Request to send CA
- 4 Reset errors (clear error bits in status byte)
- 3 Send break
- 2 Receive enable
- 1 Data terminal ready CD
- 0 Transmit enable

Speed byte (upper byte of the designated register) selects baud rates



Not used

9600 baud	0	0	0
4800	0	0	1
2400	0	1	0
1800	0	1	1
1200	1	0	0
300	1	0	1
150	1	1	0
110	1	1	1

Status byte (upper byte of the designated register) from the CCU



- 7 Date set ready CC
- 6 Sync detected
- 5 Frame error
- 4 Overrun error
- 3 Parity error
- 2 Transmit empty [CCU can accept two bytes of data from AN/Ayk-14(V)]
- 1 Receive ready [byte ready for input to AN/AYK-14(V)]
- 0 Transmit ready [CCU can accept one byte of data from AN/AYK-14(V)]

The sequence of events for a support channel output is as follows:

Mode	SCIO a,m = 1100
Command	SCIO a,m = 1100
Speed	SCIO a,m = 0010
Data	SCIO a,m = 0100

The sequence of events for a support channel input is as follows:

Mode	SCIO a,m = 1100
Command	SCIO a,m = 1100
Speed	SCIO a,m = 0010
Status	SCIO m = 1000
Input	SCI a

Depending on the baud rate, a check for status may have to be performed for each output.

If a program is stopped and restarted, a mode byte looks like a command byte unless a command byte to reset mode is executed.

LIST PROCESSING INSTRUCTIONS

The AN/AYK-14(V) stack and queue instructions provide functions to maintain linked (threaded) lists of items within AN/AYK-14(V) main memory. These instructions are implemented using a memory interlock which facilitates access to the lists by multiple processors.

The stack and queue instructions can be used to manipulate lists with any number of items and items of any size and format. The first word of each item, however, is to be reserved for the list link.

The stack instructions [Stack Get Top (SGT), and Stack Put Top (SPT)] manipulate a list of one pointer (stack top pointer) and provide last-in, first-out (push/pop) lists. The queue instructions [Queue Get Top (QGT), Queue Put Top (QPT), and Queue Put Bottom (QPB)] manipulate lists with two pointers (queue top and bottom pointers) and provide output restricted lists. Tests are provided in the instructions to handle empty lists.

Detailed definitions of the lists and descriptions of the list processing instructions are given in the paragraphs which follow.

STACK INSTRUCTIONS

The stacks manipulated by the AN/AYK-14(V) stack instructions are linked lists of items of any size, format, and number, each with a pointer indicating the top of the stack (Figure 7-1). The location of the pointer shall be denoted by Y and the contents of this location (i.e., the address of the top item in the stack) shall be denoted as (Y). The first word of each item in the list is reserved for the link to the next item. The null link (i.e., the address pointed to by the link of the last item) shall be 0. The stack is initially set so that (Y) = 0000.

The stack instructions are in RX format (i.e., they are 32-bit instructions, with the second 16 bits used in obtaining the address of an operand in memory). No indirect addressing is allowed. The m-field is used to specify one of 16 general registers to be employed as an index register. Y, the address of the stack pointer, is computed as $y + (Rm)$, where y is the second 16 bits of the instruction and (Rm) is the contents of the register specified by the m-field.

NOTE:

All stack thread cells must reside in the first 32K of memory for a standalone IOP configuration. If the threadcell ((Y)) for the SGT instruction contains an address larger than $7FFF_{16}$, then (Y + 1) is stored into pointer Y rather than the initial thread cell.

Stackpointer

List of Linked Items

Location Y

(Y)

0100

4000

Points to top item in the list

Location

Items

4000

3000

2000

1000

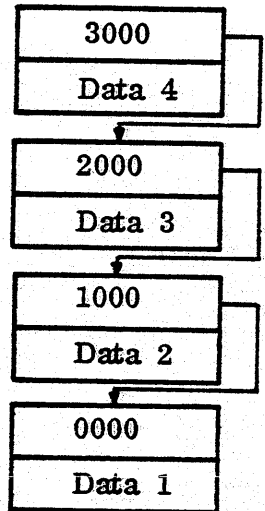
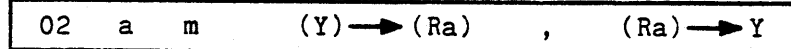


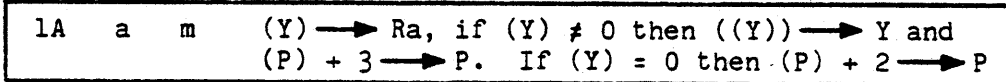
Figure 7-1. Example of a Stack

Stack Put Top (SPT a, y, m)



The SPT instruction adds a new item onto the stack. (Ra) is the address of the new item. (Y) is placed at (Ra) thus linking the new item to the item previously at the top of the stack. The address of the new item is then placed in Y, maintaining Y as the pointer to the top of the stack. $Y \neq y + (Rm)$ where y is the second 16 bits of the instruction and Rm is the register specified by the m-field. Figure 7-2 is a representation of the action of this instruction.

Stack Get Top (SGT a, y, m)



The SGT instruction removes the top item from the stack. The stack pointer contents (address of the top item) is placed in Ra and, if (Y) ≠ 0 stack not empty, the address of the item pointed to by the link of the item removed is then placed in Y. This maintains Y as pointing to the new top of the stack. The instruction at P + 3 is executed, skipping the next sequential instruction. If (Y) = 0, the stack is empty. In this case, the next sequential instruction (located at P + 2 since SGT is a 32-bit instruction) is executed. Figure 7-3 is a representation of the action of this instruction.

QUEUE INSTRUCTIONS

The queues manipulated by the AN/AYK-14(V) instructions are linked lists of items of any size, format, and number, each with a pair of pointers indicating the top and bottom of the queue. The location of the pointer to the top of the queue is Y, and that of the pointer to the bottom of the queue is $Y \oplus 1$ (logical OR of Y and 1) (Figure 7-4). The location of the item at the top of the queue is then (Y), and the location of the item at the bottom of the queue is $(Y \oplus 1)$. The first word of each item is reserved for the link to the next item. The null link (i.e., the contents of the link of the bottom item) shall be 0. The bottom item of the queue shall have a null link. The queue is set initially so that (Y) = 0000 and $(Y \oplus 1) = Y$.

The queue instructions are in RX format (i.e., they are 32-bit instructions, with the second 16 bits used to obtain the address of an operand in memory). No indirect addressing is allowed. The m-field of the instruction is used to specify one of 16 registers as an index register. Y, the address of the pointer to the top of the stack, is computed as $y + (Rm)$, where y is the second 16 bits of the instruction and (Rm) is the contents of the register specified by the m-field.

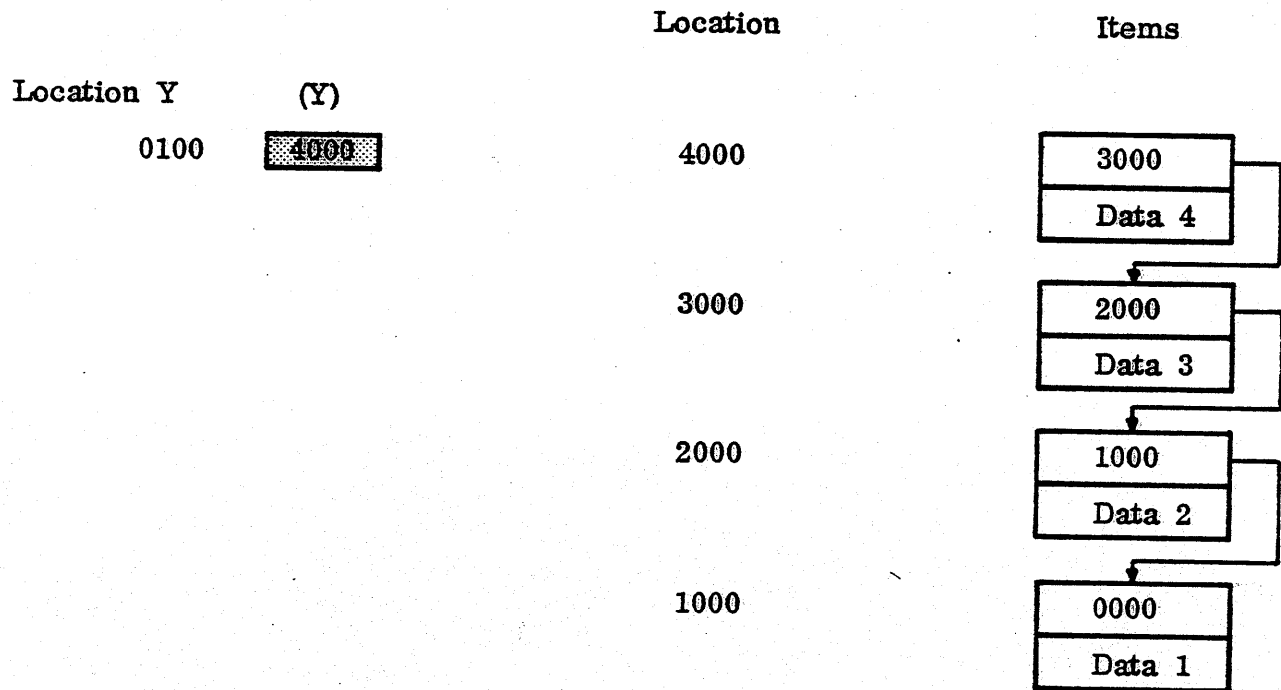
NOTE:

All queue thread cells must reside in the first 32K of memory for a standalone IOP configuration. If the thread cell ((Y)) for the QGT instruction contains an address larger than $7FFF_{16}$, the $(Y \oplus 1)$ is stored into pointer Y rather than the initial thread cell.

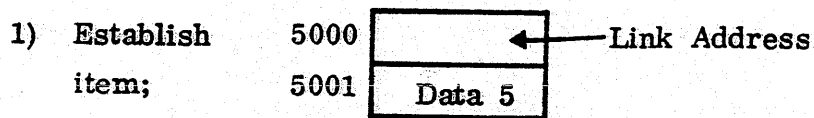
Operation: (Y) → (Ra), (Ra) → Y

Stackpointer

List of Linked Items



To add item to the stack



2) Put location of item in Ra; 5000 → Ra

3) Execute SPT a,y,m

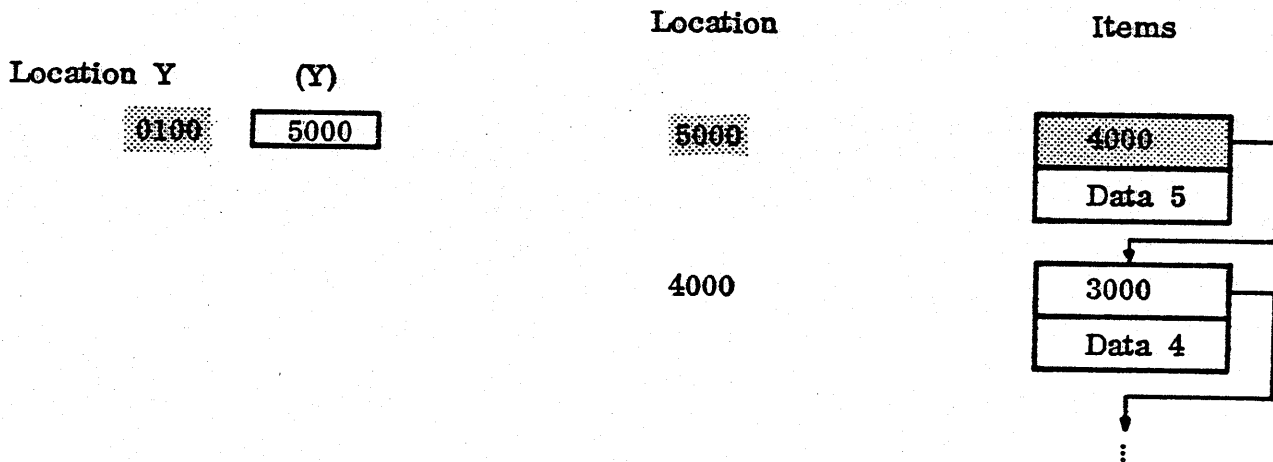


Figure 7-2. Example of an SPT Operation

Operation: (Y) → Ra, (Y) → Y

Stackpointer

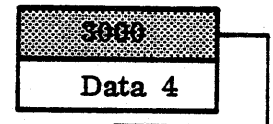
List of Linked Items

Location Y (Y)
0100 4000

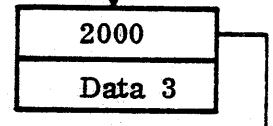
Location

Items

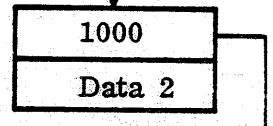
4000



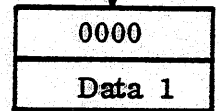
3000



2000



1000



To get top item in stack

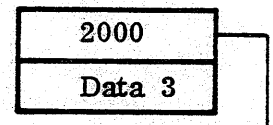
Execute SGT a,y,m

Location Y (Y)
0100 3000

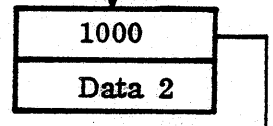
Location

Items

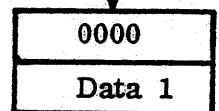
3000



2000



1000



Where:

- 1) Ra 4000 points to Data 4;
- 2) (Y) 3000 points to Data 3 (new top)

Figure 7-3. Example of an SGT Operation.

Queue Pointers

List of Linked Items

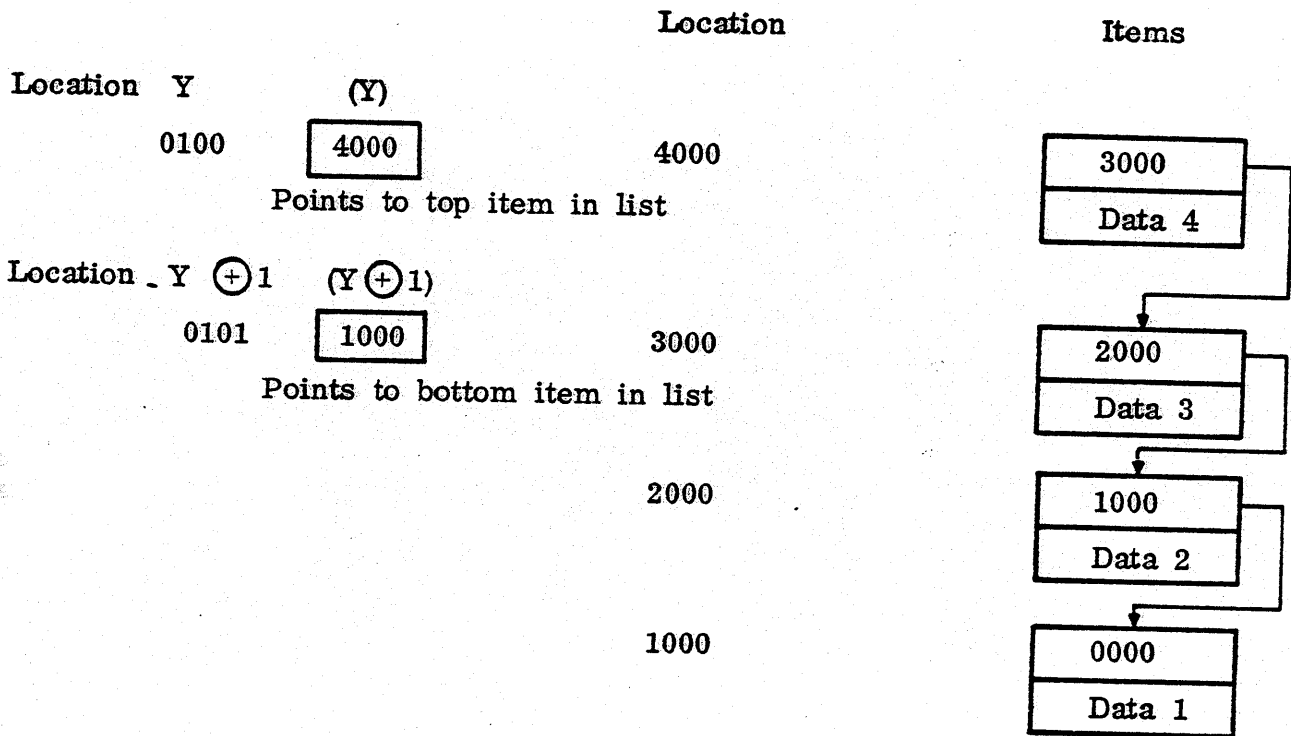


Figure 7-4. Example of a Queue

Queue Put Top (QPT a, y, m)

12	a	m	$(Y) \rightarrow (Ra), (Ra) \rightarrow Y, \text{ if } (Y) = 0$ then $(Ra) \rightarrow Y \oplus 1$
----	---	---	---

The QPT instruction adds a new item to the top of the queue. (Ra) specifies the address of the new item. (Y) is placed at (Ra) thus linking the new item to the item previously at the top of the queue. The address of the new item (Ra) is placed at Y, maintaining Y as the pointer to the top of the queue. If (Y) = 0, the queue was empty prior to executing the instruction. In this case, (Ra) is placed in $Y \oplus 1$, setting $Y \oplus 1$ to point to the bottom of the queue (in the case of a queue of one item, the top and bottom pointers point to the same address). $Y = y + (Rm)$ where y is the second 16 bits of the instruction, and (Rm) is the contents of the register specified by the m-field. Figure 7-5 is a representation of the action of this instruction.

Queue Put Bottom (QPB a, y, m)

16	a	m	$(Ra) \rightarrow (Y \oplus 1), (Ra) \rightarrow Y \oplus 1, 0 \rightarrow (Ra)$
----	---	---	--

The QPB instruction adds a new item to the bottom of the queue. (Ra) specifies the address of the new item. This address is placed at $(Y \oplus 1)$, the bottom address of the queue prior to executing the instruction. This links the item previously at the bottom of the queue to the new item. The address of the new item is then placed in $Y \oplus 1$, maintaining $Y \oplus 1$ as the pointer to the bottom of the queue. The 0 is then stored at (Ra), planting a null link in the new bottom item of the queue. $Y = y + (Rm)$ where y is the second 16 bits of the instruction, and (Rm) is the contents of the register specified by the m-field of the instruction. Figure 7-6 is a representation of the action of this instruction.

Queue Get Top (QGT a, y, m)

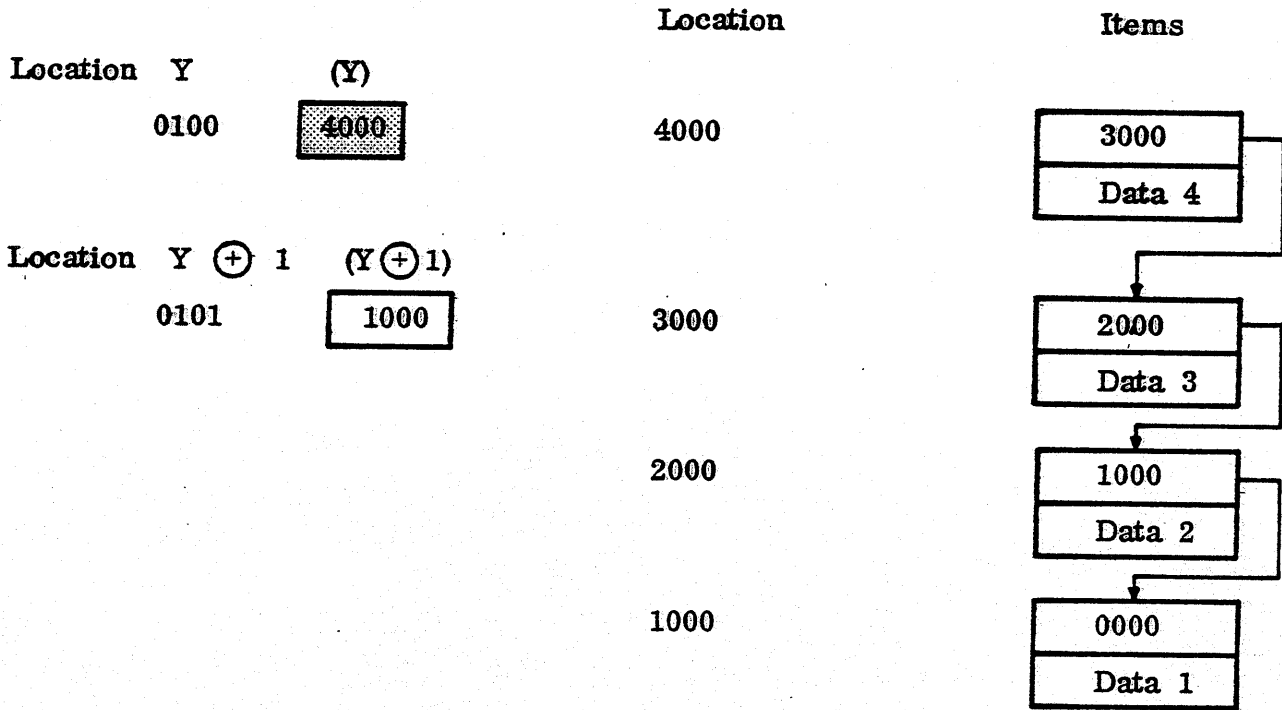
1E	a	m	$(Y) \rightarrow Ra; \text{ if } (Y) = 0 (P) + 2 \rightarrow P; \text{ if } (Y) \neq 0$ then $(P) + 3 \rightarrow P; ((Y)) \rightarrow Y. \text{ If } ((Y)) = 0, \text{ then}$ $Y \rightarrow Y \oplus 1$
----	---	---	---

The QGT instruction removes an item from the top of the queue. The top pointer contents (address of the top item) is placed in Ra, and if it is 0, the queue is empty and the instruction exits to the next sequential instruction located $P + 2$. If $(Y) \neq 0$, the queue was not empty and the address of the item pointed to by the link of the top item is placed in Y. This maintains Y as the queue top pointer. The contents of the top pointer is now tested, and if $((Y)) = 0$, the last item was removed from the queue and the pointers are initialized by placing the top pointer address Y in the bottom pointer $Y \oplus 1$. The next instruction executed is at $P + 3$.

Operation: $(Y) \rightarrow (Ra)$, $(Ra) \rightarrow Y$; if $(Y) = 0(Ra) \rightarrow Y \oplus 1$

Queue Pointers

List of Linked Items



To add item to the top of the queue

- 1) Establish item; 5000 → Link Address
5001 → Data 5
- 2) Load location of item in Ra; 5000 → Ra
- 3) Execute QPT a,y,m

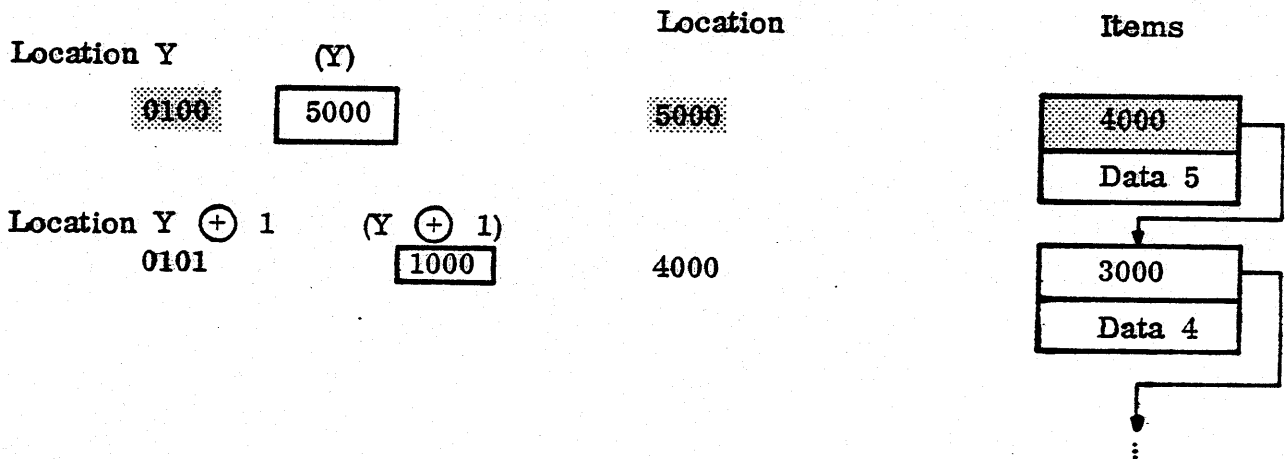
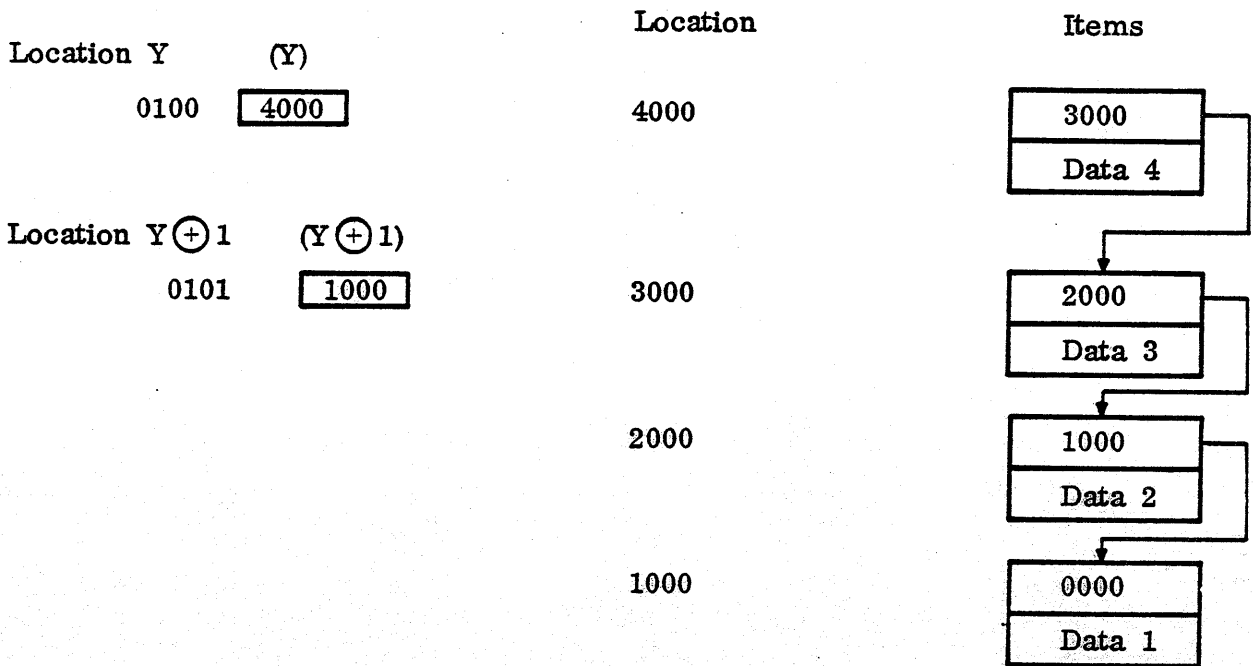


Figure 7-5. Example of a QPT Operation

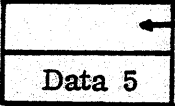
Operation: $(Ra) \rightarrow (Y \oplus 1)$, $(Ra) \rightarrow Y \oplus 1$, $O \rightarrow (Ra)$

Queue Pointers

List of Linked Items



To add item to the bottom of the queue

- 1) Establish item; 5000  Link Address
- 2) Load location of item in Ra; $5000 \rightarrow Ra$
- 3) Execute QPB a,y,m

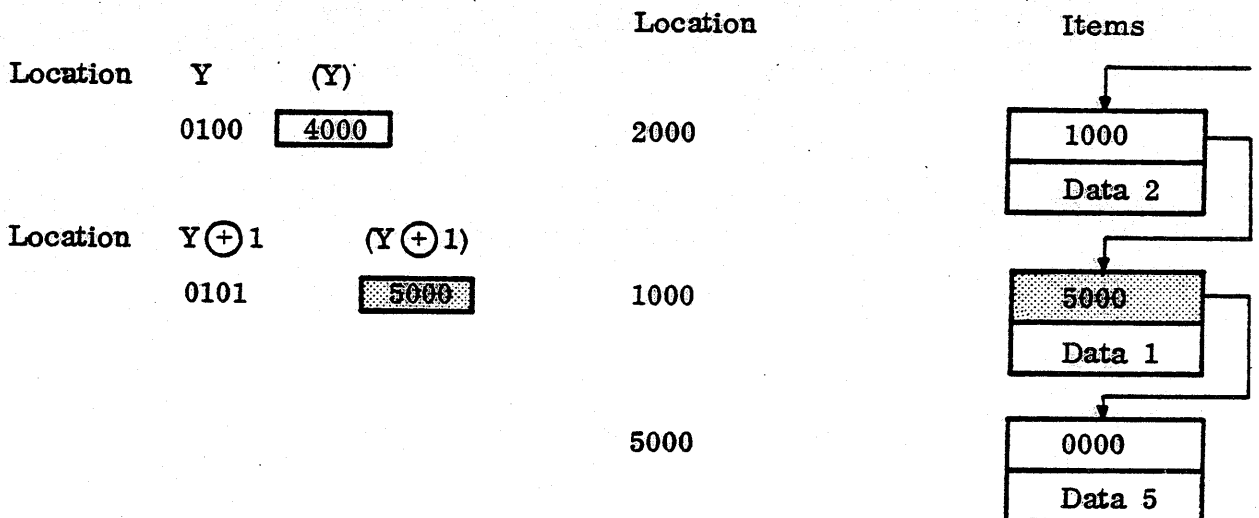
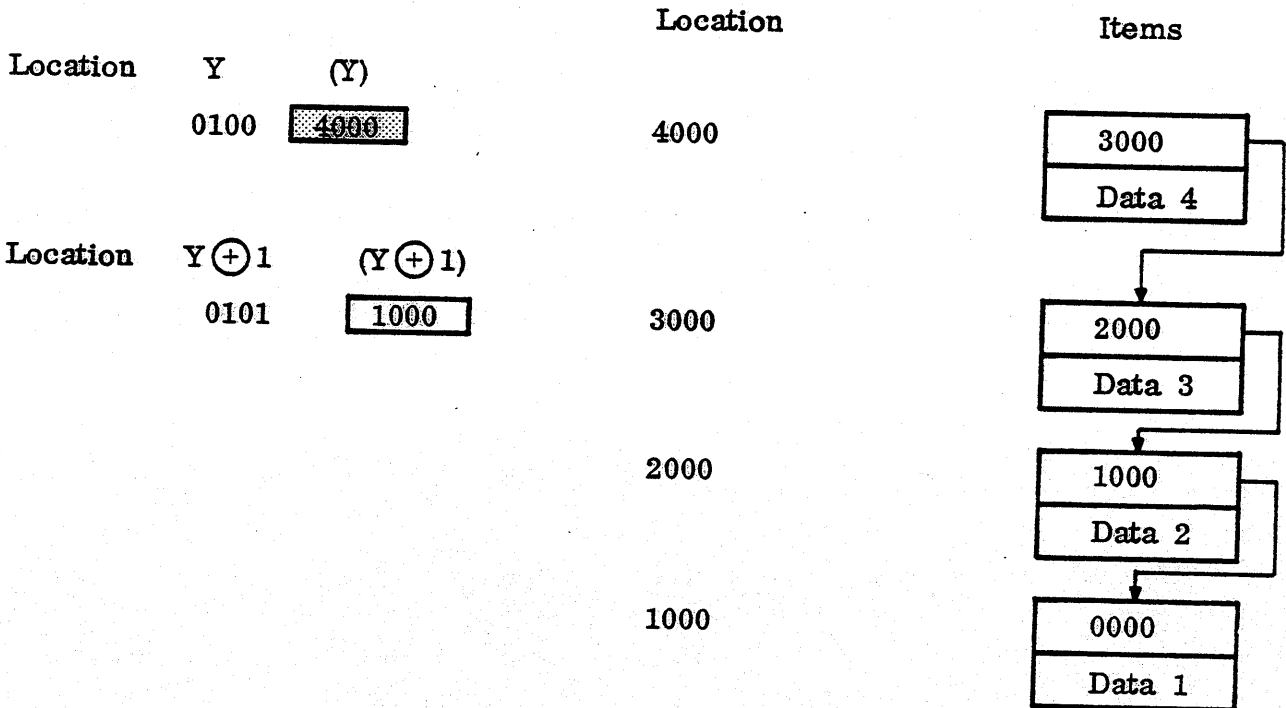


Figure 7-6. Example of a QPB Operation

Operation: $(Y) \rightarrow Ra,$ $(Y) \rightarrow Y$

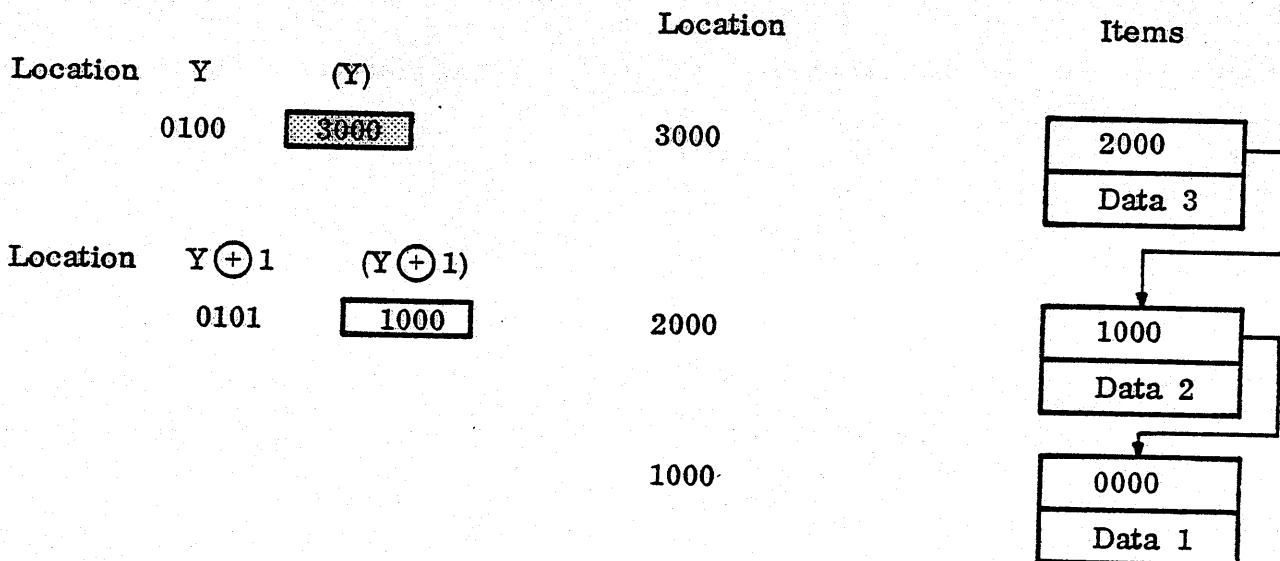
Queue Pointers

List of Linked Items



To get top item in list

Execute QGT a,y,m



Where:

- 1) Ra **4000** points to Data 4
- 2) (Y) **3000** points to Data 3 (new top)

Figure 7-7. Example of a QGT Operation

GENERAL

The instruction descriptions in this section consist of the following four parts.

- 1) Instruction name
- 2) Hexadecimal coding format
- 3) Mnemonic coding format recognized by the assembler
- 4) Instruction description

MNEMONIC CONVENTIONS

The mnemonic operation codes recognized by the assembler follow a set of conventions for prefixes and suffixes.

Conventions for mnemonic letters indicating the format type are:

- 1) RL - Format mnemonics are prefixed by L and local jumps.
- 2) RR - Format mnemonics are suffixed by R.
- 3) RI - Format mnemonics are suffixed by I except for local jumps.
- 4) RX - Format mnemonics are not suffixed.
- 5) RK - Format mnemonics are suffixed by K except for 32-bit jumps and instructions with no RX format equivalent (e.g., shifts). The mnemonics for 32-bit jump instructions are identical for RK and RX formats. To differentiate between the two, the programmer codes an asterisk (*) preceding the y operand for RX (indirect) jumps and omits the asterisk for RK (direct) jumps.

Examples

J	y,m	.	jump to Y (RK format)
J	*y,m	.	jump to (Y) (RX format)

The mnemonic description of the instruction operation code uses the following conventions.

Initial Letter
of MnemonicMeaning

A	Add, AND, Algebraic, Activity
B	Byte, Biased
C	Compare, Circular, Channel, Count

Initial Letter
of Mnemonic

Meaning

D	Divide, Decrement, Disable, Diagnostic
E	Executive, Enable
H	Halt
I	Increment, Input/Output, Initiate, Interrupt
J	Jump
L	Load, Logical, Local
M	Multiply, Masked
N	Negative, No Operation
O	OR, One's
P	Positive, Processor
R	Round, Remote, Reverse, Read
S	Store, Subtract, Set, Square
T	Two's
W	Write
X	Exclusive, Index
Z	Zero

DOUBLE PRECISION

The instruction repertoire includes several instructions that provide for operations involving two adjacent 16-bit registers or two adjacent registers and two sequential 16-bit memory locations. The rules concerning use of these instructions are as follows:

- 1) Register-to-Register - Ra and Rm must be even-numbered registers with the most significant 16 bits in Ra and Rm and the least significant bits in Ra $\oplus 1$ and Rm $\oplus 1$.
- 2) Register-to-Memory/Memory-to-Register - Ra must be even-numbered register containing the most significant 16 bits of the double-length value, and Y must be an even-numbered memory location containing the most significant 16 bits of the double-length value Y, $Y \oplus 1$.

Examples

ADR a,m . Ra and Rm are even-numbered registers

AD a,y,m . y + (Rm) is an even-numbered address

These rules apply to all instructions which operate upon 32-bit quantities. They do not apply to load/store multiple instructions.

Programmers are encouraged to use the assembly language EVEN directive to ensure that double-precision quantities are assigned such that operand Y containing the most significant 16 bits falls at an even-numbered address. If the ODD directive was used, the most and least significant bits would both end up at an odd-numbered address. As stated before, to keep the most significant bits (even-numbered register) assigned to an even address, the EVEN directive must be used for all double-precision quantities.

- 3) Multiply Instructions - Ra must be an even-numbered register, but it should be remembered that the multiplicand is assumed by the hardware to be in the odd-numbered register of the two-register pair.
- 4) Divide Instructions - Ra must be an even-numbered register.

REPertoire OF INSTRUCTIONS

The instruction repertoire is divided into instruction types, and within a type listed by operation code in alphanumeric order..

Instruction types are as follows:

- 1) Load instructions
- 2) Store instructions
- 3) Arithmetic instructions
- 4) Logical instructions
- 5) Compare instructions
- 6) Shift instructions
- 7) Unconditional jump instructions
- 8) Conditional jump instructions
- 9) Miscellaneous instructions
- 10) I/O instructions (see Section 10 - I/O Channel Operations)

LOAD INSTRUCTIONS

Byte Load (Indirect) (03 a m) (BL a,y,m)

This instruction loads the selected byte from memory address Y in bits 0 through 7 of Ra, clears bits 8 through 15, and sets the condition code. The various sources of byte selection are described in Section 3.

Load (Register) (04 a m) (LR a,m)

This instruction loads (Rm) into Ra and sets the condition code.

Load (Immediate) (05 a m) (LI a,m)

This instruction loads the contents of the memory address Y* into Ra and sets the condition code. The memory address is located in Rm.

Load (Constant) (06 a m) (LK a,y,m)

This instruction loads the operand Y into Ra and sets the condition code.

Load (Indirect) (07 a m) (L a,y,m)

This instruction loads the contents of memory address Y into Ra and sets the condition code.

Load Double (Immediate) (09 a m) (LDI a,m)

This instruction loads the contents of memory address Y* and $Y^* \oplus 1$, into Ra and $Ra \oplus 1$, respectively, and sets the condition code. The memory address is located in Rm.

Load Double (Indirect) (0B a m) (LD a,y,m)

This instruction loads the contents of memory addresses Y and $Y \oplus 1$ into Ra and $Ra \oplus 1$, respectively, and sets the condition code.

Load P Register (0C a 4) (LPR a)

This instruction loads (Ra) into P.

Load Multiple (0F a m) (LM a,y,m)

This instruction loads the contents of sequential memory addresses beginning at Y into sequential registers Ra through Rm. If a is greater than m, the registers loaded are Ra, Ra + 1, ..., R15, R0, ..., Rm. If a equals m, then only one register is loaded. In this instruction, Y equals y. No indexing or indirect addressing is performed. The beginning address may be either odd or even.

Byte Load and Index by 1 (Indirect) (13 a m) (BLX a,y,m)

This instruction loads the selected byte from memory address Y into bits 0 through 7 of Ra, clears bits 8 through 15, sets the condition code, and then increments (Rm) by 1 if $a \neq m$. The various sources of byte select are described in Section 3.

Load and Index by 1 (Immediate) (15 a m) (LXI a,m)

This instruction loads the contents of memory address Y* into Ra, sets the condition code, and then increments (Rm) by 1 if $a \neq m$. The memory address is located in Rm.

Load and Index by 1 (Indirect) (17 a m) (LX a,y,m)

This instruction loads the contents of memory address Y into Ra, sets the condition code, and then increments (Rm) by 1 if $a \neq m$.

Load Double and Index by 2 (Immediate) (19 a m) (LDXI a,m)

This instruction loads the contents of memory address Y* and $Y^* \oplus 1$ into Ra and $Ra \oplus 1$, respectively, sets the condition code, and then increments (Rm) by 2 if $a \neq m$. The memory address is located in Rm.

Load Double and Index by 2 (Indirect) (1B a m) (LDX a,y,m)

This instruction loads the contents of memory address Y and $Y \oplus 1$ into Ra and $Ra \oplus 1$, respectively, sets the condition code, and then increments (Rm) by 2 if a \neq m.

Load PSW (Immediate) (1D - m) (LPI m)

This instruction loads the contents of memory addresses Y^* , $Y^* + 1$, and $Y^* + 2$ into the program address register, status register 1, and status register 2, respectively. The memory address is located in Rm and may be either odd or even. The a-field is not used by the hardware. This is an executive mode instruction.

Load PSW (Indirect) (1F - m) (LP y,m)

This instruction loads the contents of memory address Y, Y + 1, and Y + 2 into the program address register, status register 1, and status register 2, respectively. Y may be either an odd or even address. The a field is not used by the hardware. This is an executive mode instruction. When location last jump is queued by the CSE, the Y portion of the instruction is displayed rather than the basic instruction itself.

Literal Load (CC a m) (LL a,m)

This instruction loads the 4-bit literal contained in the m-field of the instruction into bits 3 through 0 of Ra, clears bits 15 through 4, and sets the condition code.

Load Address Register (Register) (B0 a m) (LARR a,m)

This instruction loads (Rm) into the page register specified by bits 5 through 0 of (Ra). This is an executive mode instruction.

Load Address Register (Immediate) (B1 a m) (LARI a,m)

This instruction loads the contents of the memory address Y^* into the page address register specified by bits 0 through 5 of (Ra). This is an executive mode instruction. The memory address is located in Rm.

Load Address Register Multiple (B3 a m) (LARM a,y,m)

This instruction loads the contents of the sequential memory addresses beginning at Y into the sequential page address registers beginning at the register specified by bits 5 through 0 of (Ra) and continuing until the number of executions equals one plus the count in bits 13 through 8 of (Ra). A count of 0's loads one page address register. This is an executive mode instruction. If the beginning page register address and the count combine so as to load beyond the last page register, the addressing will go end around on the page registers.

NOTE:

In a dual system (CPU and IOP), the CPU instructions LARR, LARI, and LARM must not be included in the program until approximately 25 microseconds (25 instructions) after power up, or a memory error will occur.

STORE INSTRUCTIONS

Byte Store (Indirect) (23 a m) (BS a,y,m)

This instruction stores bits 7 through 0 of (Ra) into the selected byte at memory address Y. The value that is used as the byte selector is described in Section 3.

Store (Immediate) (25 a m) (SI a,m)

This instruction stores (Ra) at memory address Y*. The memory address is located in Rm.

Store (Indirect) (27 a m) (S a,y,m)

This instruction stores (Ra) at memory address Y.

Store Double (Immediate) (29 a m) (SDI a,m)

This instruction stores (Ra) and $(Ra \oplus 1)$ at memory addresses Y* and $Y* \oplus 1$, respectively. The memory address is located in Rm.

Store Double (Indirect) (2B a m) (SD a,y,m)

This instruction stores (Ra) and $(Ra \oplus 1)$ at memory addresses Y and $Y \oplus 1$, respectively.

Store Multiple (2F a m) (SM a,y,m)

This instruction stores, in sequential memory addresses beginning at Y, the contents of sequential registers beginning at Ra and ending with Rm. If a is greater than m, the registers stored shall be Ra, Ra + 1, ..., R15, R0, ..., Rm. In this instruction Y equals y. No indexing or indirect addressing is performed. The beginning address may be either odd or even.

Byte Store and Index by 1 (Indirect) (33 a m) (BSX a,y,m)

This instruction stores bits 7 through 0 of Ra in the selected byte at memory address Y and then increments (Rm) by 1. The various sources of byte select are described in Section 3.

Store and Index by 1 (Immediate) (35 a m) (SXI a,m)

This instruction stores (Ra) at memory address Y* and then increments (Rm) by 1. The memory address is located in Rm.

Store and Index by 1 (Indirect) (37 a m) (SX a,y,m)

This instruction stores (Ra) at memory address Y and then increments (Rm) by 1.

Store Double and Index by 2 (Immediate) (39 a m) (SDXI a,m)

This instruction stores (Ra) and (Ra ⊕ 1) at memory addresses Y* and Y* ⊕ 1, respectively, and then increments (Rm) by 2. The memory address is located in Rm.

Store Double and Index by 2 (Indirect) (3B a m) (SDX a,y,m)

This instruction stores (Ra) and (Ra ⊕ 1) at memory addresses Y and Y ⊕ 1, respectively, and then increments (Rm) by 2.

Store 0's (Immediate) (3D - m) (SZI m)

This instruction stores all 0's at memory address Y*. The a-field is not used by the hardware. The memory address is located in Rm.

Store 0's (Indirect) (3F - m) (SZ y,m)

This instruction stores all 0's at memory address Y. The a-field is not used by the hardware.

Store Address Register (Register) (B4 a m) (SARR a,m)

This instruction stores the contents of the page address register specified by bits 5 through 0 of (Ra) into Rm.

Store Address Register (Immediate) (B5 a m) (SARI a,m)

This instruction stores the contents of the page address register specified by bits 5 through 0 of (Ra) at memory address Y*. The memory address is located in Rm.

Store Address Register Multiple (Indirect) (B7 a m) (SARM a,y,m)

This instruction stores the contents of sequential page address registers beginning at the address word specified by bits 5 through 0 of (Ra) into sequential memory locations beginning at Y and continuing until the number of executions equals the count in bits 13 through 8 of (Ra). A count of zero stores one page register. If the beginning page register address and the count combine so as to store beyond the last page register, the addressing will go end around on the page registers.

ARITHMETIC INSTRUCTIONS

Subtract (Register) (40 a m) (SUR a,m)

This instruction subtracts (Rm) from (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Subtract (Immediate) (41 a m) (SUI a,m)

This instruction subtracts the contents of memory address Y* from (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry. The memory address is located in Rm.

Subtract (Constant) (42 a m) (SUK a,y,m)

This instruction subtracts the operand Y from (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Subtract (Indirect) (43 a m) (SU a,y,m)

This instruction subtracts the contents of memory address Y from (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Subtract Double (Register) (44 a m) (SUDR a,m)

The instruction subtracts the double-length (Rm, Rm ⊕ 1) from the double-length (Ra, Ra ⊕ 1), stores the result into Ra and Ra ⊕ 1, and then sets the condition code, overflow, and carry.

Subtract Double (Immediate) (45 a m) (SUDI a,m)

This instruction subtracts the double-length contents of memory addresses Y*, Y* ⊕ 1 from the double-length (Ra, Ra ⊕ 1), stores the result into Ra and Ra ⊕ 1, and then sets the condition code, overflow, and carry. The memory address is located in Rm.

Subtract Double (Indirect) (47 a m) (SUD a,y,m)

This instruction subtracts the double-length contents of memory addresses Y, Y ⊕ 1 from the double-length (Ra, Ra ⊕ 1), stores the result into Ra and Ra ⊕ 1, and then sets the condition code, overflow, and carry.

Literal Subtract (C8 a m) (LSU a,m)

This instruction subtracts the 4-bit unsigned literal contained in the m-field of the instruction from (Ra), stores the result into Ra, and sets the condition code, overflow, and carry. The literal is right justified, zero filled before the subtraction.

Literal Subtract Double (C9 a m) (LSUD a,m)

This instruction subtracts the 4-bit unsigned literal contained in the m-field of the instruction from the double-length contents of Ra, Ra ⊕ 1, stores the result into Ra, Ra ⊕ 1, and sets the condition code, overflow, and carry. The literal is right justified, zero filled before the subtraction.

Byte Subtract (Indirect) (D3 a m) (BSU a,y,m)

This instruction subtracts the selected byte of memory address Y from (Ra), stores the result into Ra, and sets the condition code, overflow, and carry. The selected byte is right justified and sign extended before the subtraction. The various sources of byte selection are described in Section 3.

Floating-Point Subtract (Register) (A0 a m) (FSUR a,m)

This instruction subtracts (R_m , $R_m \oplus 1$) from (R_a , $R_a \oplus 1$), stores the normalized result in R_a , $R_a + 1$ with the residue (when selected via SR1) in $R_a + 2$, $R_a + 3$, and then sets the condition code, overflow, and carry.

If residue is selected (SR1, bit 6, set), the residue will be returned to the $R_a + 2$, $R_a + 3$ registers. The residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit subtraction result.
- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

The arithmetic fault interrupt will be generated if it is enabled (SR1, bit 7, set) and if a result exponent overflow or underflow occurs. An exponent overflow (SR1, bit 10, set) will occur if:

- 1) The exponent of the normalized result is too large to represent.
- 2) One of the operands is machine infinity.

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for subtract instructions.

An exponent underflow (SR1, bits 10 and 11, set) will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Subtract (Immediate) (A1 a m) (FSUI a,m)

This instruction subtracts the contents of memory addresses Y^* , $Y^* \oplus 1$ from (R_a , $R_a \oplus 1$), stores the result in R_a , $R_a + 1$ with the residue (when selected via SR1) in $R_a + 2$, $R_a + 3$, and then sets the condition code, overflow, and carry. The memory address is located in R_m .

If residue is selected (SR1, bit 6, set), the residue will be returned to the $R_a + 2$, $R_a + 3$ registers. The residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit subtraction result.

- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

The arithmetic fault interrupt will be generated if it is enabled (SR1, bit 7, set) and if a result exponent overflow or underflow occurs. An exponent overflow (SR1, bit 10, set) will occur if:

- 1) The exponent of the normalized result is too large to represent.
- 2) One of the operands is machine infinity.

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for subtract instructions.

An exponent underflow (SR1, bits 10 and 11, set) will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Subtract (Indirect) (A3 a m) (FSU a,y,m)

This instruction subtracts the contents of memory addresses Y, $Y \oplus 1$ from $(Ra, Ra \oplus 1)$, stores the result in $Ra, Ra + 1$ with the residue (when selected via SR1) in $Ra + 2, Ra + 3$, and then sets the condition code, overflow, and carry.

If residue is selected (SR1, bit 6, set), the residue will be returned to the $Ra + 2, Ra + 3$ registers. The residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit subtraction result.
- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

The arithmetic fault interrupt will be generated if it is enabled (SR1, bit 7, set) and if a result exponent overflow or underflow occurs. An exponent overflow (SR1, bit 10, set) will occur if:

- 1) The exponent of the normalized result is too large to represent.
- 2) One of the operands is machine infinity.

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for subtract instructions.

An exponent underflow (SR1, bits 10 and 11, set) will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Add (Register) (48 a m) (AR a,m)

This instruction adds (Rm) to (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Add (Immediate) (49 a m) (AI a,m)

This instruction adds the contents of memory address Y* into (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry. The memory address is located in Rm.

Add (Constant) (4A a m) (AK a,y,m)

This instruction adds the operand Y to (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Add (Indirect) (4B a m) (A a,y,m)

This instruction adds the contents of memory address Y to (Ra), stores the result into Ra, and then sets the condition code, overflow, and carry.

Add Double (Register) (4C a m) (ADR a,m)

This instruction adds the double-length (Rm, Rm \oplus 1) to the double-length (Ra, Ra \oplus 1), stores the result into Ra and Ra \oplus 1, and then sets the condition code, overflow, and carry.

Add Double (Immediate) (4D a m) (ADI a,m)

This instruction adds the double-length contents of memory addresses Y*, Y* \oplus 1 to the double-length (Ra, Ra \oplus 1), stores the result into Ra and Ra \oplus 1, and then sets the condition code, overflow, and carry. The memory address Y* is located in Rm.

Add Double (Indirect) (4F a m) (AD a,y,m)

This instruction adds the double-length contents of memory address Y, Y \oplus 1 to the double-length (Ra, Ra \oplus 1), stores the result into Ra and Ra \oplus 1, and then sets the condition code, overflow, and carry.

Literal Add (CA a m) (LA a,m)

This instruction adds the 4-bit unsigned literal contained in the m-field of the instruction to (Ra), stores the result into Ra, and sets the condition

code, overflow, and carry. The literal is right justified and zero filled before the addition.

Literal Add Double (CB a m) (LAD a,m)

This instruction adds the 4-bit unsigned literal contained in the m-field of the instruction to the double length contents of Ra, Ra ⊕ 1, stores the result into Ra, Ra ⊕ 1, and sets the condition code, overflow, and carry. The literal is right justified and zero filled before the addition.

Byte Add (D7 a m) (BA a,y,m)

This instruction adds the selected byte of memory address Y to (Ra), stores the result into Ra, and sets the condition code, overflow, and carry. The selected byte is right justified and sign extended before the addition. The various sources of byte selection are described in Section 3.

Floating-Point Add (Register) (A4 a m) (FAR a,m)

This instruction adds (Rm, Rm ⊕ 1) to (Ra, Ra ⊕ 1), stores the normalized result into Ra, Ra + 1 with the residue (when selected via SRL) in Ra + 2, Ra + 3, and then sets the condition code, overflow, and carry.

If selected, residue will be returned to the Ra + 2, Ra + 3 registers. Residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit addition result.
- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized result is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for the add instructions.

Exponent underflow will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Add (Immediate) (A5 a m) (FAI a,m)

This instruction adds the contents of memory addresses Y^* , $Y^* \oplus 1$ to $(Ra, Ra \oplus 1)$, stores the normalized result into $Ra, Ra + 1$ with the residue in $Ra + 2, Ra + 3$, and then sets the condition code, overflow, and carry. The memory address is located in Rm .

If selected, residue will be returned to the $Ra + 2, Ra + 3$ registers. Residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit addition result.
- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized result is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for the add instructions.

Exponent underflow will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Add (Indirect) (A7 a m) (FA a,y,m)

This instruction adds the contents of memory addresses $Y, Y \oplus 1$, to $(Ra, Ra \oplus 1)$, stores the normalized result into $Ra, Ra + 1$ with the residue in $Ra + 2, Ra + 3$, and then sets the condition code, overflow, and carry.

If selected, residue will be returned to the $Ra + 2, Ra + 3$ registers. Residue will be a floating-point word as follows:

- 1) If the exponent difference of the two normalized inputs is 5 or less, the residue will have an exponent of 6 less than the result's exponent, and the mantissa will represent the 24 least significant bits of the normalized 48-bit addition result.
- 2) If the exponent difference of the two normalized inputs is greater than 5, the residue will be the normalized input with the smaller exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized result is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs. The machine infinity sign will be as defined in Table 3-1 for the add instructions.

Exponent underflow will occur if the exponent of the normalized result is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Multiply (Register) (58 a m) (MR a,m)

This instruction multiplies the integer (Rm) by the integer (Ra ⊕ 1), stores the double-length result most significant bits into Ra, least significant bits into Ra ⊕ 1, and then sets the condition code.

Multiply (Immediate) (59 a m) (MI a,m)

This instruction multiplies the integer contents of memory address Y* by the integer (Ra ⊕ 1), stores the double-length result most significant bits into Ra, least significant bits into Ra ⊕ 1, and then sets the condition code. The memory address is located in Rm.

Multiply (Constant) (5A a m) (MK a,y,m)

This instruction multiplies the integer operand Y by the integer (Ra ⊕ 1), stores the double-length result most significant bits into Ra, least significant bits into Ra ⊕ 1, and then sets the condition code.

Multiply (Indirect) (5B a m) (M a,y,m)

This instruction multiplies the integer contents of memory address Y by the integer (Ra ⊕ 1), stores the double-length result most significant bits in Ra, least significant bits into Ra ⊕ 1, and then sets the condition code.

Multiply Double (Register) (B8 a m) (MDR a,m)

This instruction multiplies the double-length integer (Rm, Rm ⊕ 1) by the double-length integer (Ra, Ra ⊕ 1), stores the result in Ra, Ra + 1, Ra + 2, Ra + 3, and then sets the condition code (most significant bits into Ra and least significant bits into Ra + 3). If Ra = 14, the 64-bit product is stored end around in the register stack.

Multiply Double (Immediate) (B9 a m) (MDI a,m)

This instruction multiplies the double-length integer contents of memory addresses Y^* , $Y^* + 1$ by the double-length integer $(Ra, Ra \oplus 1)$, stores the result into $Ra, Ra \oplus 1, Ra + 2, Ra + 3$, and then sets the condition code (most significant bits to Ra and least significant bits to $Ra + 3$). If $Ra = 14$, the 64-bit product is stored end around in the register stack. The memory address is located in Rm .

Multiply Double (Indirect) (BB a m) (MD a,y,m)

This instruction multiplies the double-length integer contents of memory addresses $Y, Y \oplus 1$ by the double-length integer $(Ra, Ra \oplus 1)$, stores the result into $Ra, Ra + 1, Ra + 2, Ra + 3$, and then sets the condition code (most significant bits to Ra , least significant bits to $Ra + 3$). If $Ra = 14$, the 64-bit product is stored end around in the register stack.

Literal Multiply (CE a m) (LMUL a,m)

This instruction multiplies the 4-bit unsigned literal contained in the m -field of the instruction by the integer $(Ra \oplus 1)$, stores the double-length most significant bits into Ra , least significant bits to $Ra \oplus 1$, and then sets the condition code. The 4-bit literal is right justified and zero filled before the multiply.

Floating-Point Multiply (Register) (A8 a m) (FMR a,m)

This instruction multiplies the double-length $(Rm, Rm \oplus 1)$ by the double-length $(Ra, Ra + 1)$, stores the normalized result into $Ra, Ra + 1$, and if residue selected via SRL , residue into $Ra + 2, Ra + 3$; then sets the condition code, overflow, and carry.

If selected, residue will be returned to the $Ra + 2, Ra + 3$ registers. Residue will be a floating-point word; the mantissa represents the 24 least significant bits of the normalized, 48-bit product.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized product is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs, except for the case of machine infinity times zero. Machine infinity times zero will generate a result and residue of machine zero with an overflow indication. The machine infinity sign will be the exclusive OR of the two operands as defined in Table 3-1 for the multiply instruction. Exponent underflow will occur if the exponent of the normalized product is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Multiply (Immediate) (A9 a m) (FMI a,m)

This instruction multiplies the double-length contents of memory addresses Y^* , $Y^* \oplus 1$ by the double-length $(Ra, Ra \oplus 1)$, stores the normalized result into Ra , $Ra + 1$, and if residue selected via $SR1$, residue into $Ra + 2$, $Ra + 3$; then sets the condition code, overflow, and carry. The memory address is located in Rm .

If selected, residue will be returned to the $Ra + 2$, $Ra + 3$ registers. Residue will be a floating-point word; the mantissa represents the 24 least significant bits of the normalized, 48-bit product.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized product is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs, except for the case of machine infinity times zero. Machine infinity times zero will generate a result and residue of machine zero with an overflow indication. The machine infinity sign will be the exclusive OR of the two operands as defined in Table 3-1 for the multiply instruction. Exponent underflow will occur if the exponent of the normalized product is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Multiply (Indirect) (AB a m) (FM a,y,m)

This instruction multiplies the double-length contents of memory addresses Y , $Y \oplus 1$ by the double-length $(Ra, Ra \oplus 1)$, stores the normalized result into Ra , $Ra + 1$, and if residue selected via $SR1$, residue into $Ra + 2$, $Ra + 3$; then sets the condition code, overflow, and carry.

If selected, residue will be returned to the $Ra + 2$, $Ra + 3$ registers. Residue will be a floating-point word; the mantissa represents the 24 least significant bits of the normalized, 48-bit product.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if:

- 1) The exponent of the normalized product is too large to represent
- 2) One of the operands is machine infinity

The result and residue will be machine infinity if exponent overflow occurs, except for the case of machine infinity times zero. Machine infinity times zero will generate a result and residue of machine zero with an overflow indication. The machine infinity sign will be the exclusive OR of the two operands as defined in Table 3-1 for the multiply instruction. Exponent

underflow will occur if the exponent of the ormalized product is too small to represent. The result and residue will be machine zero if exponent underflow occurs.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underfow will not be set.

Divide (Register) (5C a m) (DR a,m)

This instruction divides the double-length integer (Ra, Ra ⊕ 1) by integer (Rm), stores the quotient into Ra ⊕ 1 and the remainder into Ra, and then sets the condition code and overflow. Condition code represents quotient. Remainder has the sign of the dividend (Ra, Ra ⊕ 1). Overflow is set in SR1 if quotient is too large to represent in Ra ⊕ 1.

Divide (Immediate) (5D a m) (DI a,m)

This instruction divides the double-length integer (Ra, Ra ⊕ 1) by the integer contents of memory address Y*, stores the quotient into Ra ⊕ 1 and the remainder into Ra, and then sets the condition code and overflow. Condition code represents quotient. Remainder has the sign of the dividend (Ra, Ra ⊕ 1). The memory address is located in Rm. Overflow is set in SR1 if quotient is too large to represent in Ra ⊕ 1.

Divide (Constant) (5E a m) (DK a,y,m)

This instruction divides the double-length integer (Ra, Ra ⊕ 1) by the integer operand Y, stores the quotient into Ra ⊕ 1 and the remainder into Ra, and then sets the condition code and overflow. Condition code represents quotient. Remainder has sign of the dividend (Ra, Ra ⊕ 1). Overflow is set in SR1 if quotient is too large to represent in Ra ⊕ 1.

Divide (5F a m) (D a,y,m)

This instruction divides the double-length integer (Ra, Ra ⊕ 1) by the integer contents of memory address Y, stores the quotient into Ra ⊕ 1 and the remainder into Ra, and then sets the condition code and overflow. Condition code represents quotient. Remainder has sign of the dividend (Ra, Ra ⊕ 1). Overflow is set in SR1 if quotient is too large to represent in Ra ⊕ 1.

Divide Double (Register) (BC a m) (DDR a,m)

This instruction divides the double-length integer (Ra, Ra + 1, Ra + 2, Ra + 3) by the double-length integer (Rm, Rm ⊕ 1), stores the quotient into Ra + 2, Ra + 3 with the remainder into Ra, Ra + 1, and then sets the condition code to represent quotient. Remainder has sign of the dividend. Overflow is set in SR1 if quotient is too large to represent in Ra + 2, Ra + 3. If $Ra \geq 13$, wraparound addressing is done on the register stack.

Divide Double (Immediate) (BD a m) (DDI a,m)

This instruction divides the double-length integer (Ra, Ra + 1, Ra + 2, Ra + 3) by the double-length integer contents of memory addresses Y*,

$Y^* \oplus 1$, stores the quotient into $Ra + 2$, $Ra + 3$ with the remainder into Ra , $Ra + 1$, and then sets the condition code to represent quotient. Remainder has sign of the dividend. Overflow is set in $SR1$ if quotient is too large to represent in $Ra + 2$, $Ra + 3$. If $Ra \geq 13$, wraparound addressing is done on the register stack. The memory address is located in Rm .

Divide Double Integer (BF a m) (DD a,y,m)

This instruction divides the double-length integer (Ra , $Ra + 1$, $Ra + 2$, $Ra + 3$) by the double-length integer contents of memory addresses Y , $Y \oplus 1$, stores the quotient into $Ra + 2$, $Ra + 3$ with the remainder into Ra , $Ra + 1$, and then sets the condition code to represent quotient. Remainder has sign of the dividend. Overflow is set in $SR1$ if quotient is too large to represent in $Ra + 2$, $Ra + 3$. If $Ra \geq 13$, wraparound addressing is done on the register stack.

Literal Divide (CF a m) (LDIV a,m)

This instruction divides the double-length integer (Ra , $Ra \oplus 1$) by the 4-bit unsigned literal contained in the m -field of the instruction, stores the quotient into $Ra \oplus 1$ and the remainder into Ra , and then sets the condition code to represent quotient. Remainder has sign of the dividend. Overflow is set in $SR1$ if quotient is too large to represent in $Ra \oplus 1$. The literal is right justified and zero filled before the dividend operation.

Floating-Point Divide (Register) (AC a m) (FDR a,m)

This instruction divides the double-length (Ra , $Ra \oplus 1$) by the double-length (Rm , $Rm \oplus 1$), stores the quotient into Ra , $Ra + 1$ and, if residue selected via $SR1$, the remainder in $Ra + 2$, $Ra + 3$, and then sets the condition code, overflow, and carry.

If selected, residue will be returned to the $Ra + 2$, $Ra + 3$ registers. The floating-point residue will be a floating-point word representing the remainder of the division operation. The exponent of the residue will be as follows:

- 1) If the mantissa of the normalized divisor is greater than the mantissa of the normalized dividend, the exponent will be 6 less than the quotient's exponent.
- 2) If the mantissa of the normalized divisor is less than or equal to the mantissa of the normalized dividend, then the exponent will be 5 less than the quotient's exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if the exponent of the normalized result is too large to represent. In this case, the result and residue will be machine infinity. Exponent underflow will occur if the exponent of normalized result is too small to represent. In this case, the result and residue will be machine zero. Refer to Table 3-1 for special divide cases and results.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Divide (Immediate) (AD a m) (FDI a,m)

This instruction divides the double-length (Ra, Ra ⊕ 1) by the double-length contents of memory addresses Y*, Y* ⊕ 1, stores the quotient into Ra, Ra + 1 and, if residue selected via SRL, the remainder in Ra + 2, Ra + 3, and then sets the condition code, overflow, and carry.

If selected, residue will be returned to the Ra + 2, Ra + 3 registers. The floating-point residue will be a floating-point word representing the remainder of the division operation. The exponent of the residue will be as follows:

- 1) If the mantissa of the normalized divisor is greater than the mantissa of the normalized dividend, the exponent will be 6 less than the quotient's exponent.
- 2) If the mantissa of the normalized divisor is less than or equal to the mantissa of the normalized dividend, then the exponent will be 5 less than the quotient's exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if the exponent of the normalized result is too large to represent. In this case, the result and residue will be machine infinity. Exponent underflow will occur if the exponent of normalized result is too small to represent. In this case, the result and residue will be machine zero. Refer to Table 3-1 for special divide cases and results.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Floating-Point Divide (Indirect) (AF a m) (FD a,y,m)

This instruction divides the double-length (Ra, Ra ⊕ 1) by the double-length contents of memory addresses Y, Y ⊕ 1, stores the quotient into Ra, Ra + 1 and, if residue selected via SRL, the remainder in Ra + 2, Ra + 3, and then sets the condition code, overflow, and carry.

If selected, residue will be returned to the Ra + 2, Ra + 3 registers. The floating-point residue will be a floating-point word representing the remainder of the division operation. The exponent of the residue will be as follows:

- 1) If the mantissa of the normalized divisor is greater than the mantissa of the normalized dividend, the exponent will be 6 less than the quotient's exponent.
- 2) If the mantissa of the normalized divisor is less than or equal to the mantissa of the normalized dividend, then the exponent will be 5 less than the quotient's exponent.

If enabled, the arithmetic fault interrupt will be generated if result exponent overflow or underflow occurs. Exponent overflow will occur if the exponent of the normalized result is too large to represent. In this case, the result and residue will be machine infinity. Exponent underflow will occur if the exponent of normalized result is too small to represent. In this case, the result and residue will be machine zero. Refer to Table 3-1 for special divide cases and results.

A residue of machine zero will be generated if the residue exponent is too small to represent, but exponent underflow will not be set.

Round Ra (08 a 2) (RR a)

This instruction adds bit 15 of Ra \oplus 1 to (Ra) and stores the result into Ra. The condition code is set in accordance with the resulting quantity in Ra.

Increment Register by 1 (08 a 8) (IROR a)

This instruction increments (Ra) by 1, stores the result into Ra, and then sets the condition code, overflow, and carry.

Decrement Register by 1 (08 a 9) (DROR a)

This instruction decrements (Ra) by 1, stores the result into Ra, and then sets the condition code, overflow, and carry.

Increment Register by 2 (08 a A) (IRTR a)

This instruction increments (Ra) by 2, stores the result into Ra, and then sets the condition code, overflow, and carry.

Decrement Register by 2 (08 a B) (DRTR a)

This instruction decrements (Ra) by 2, stores the result into Ra, and then sets the condition code, overflow, and carry.

Make Positive (08 a 0) (PR a)

If (Ra) is negative, this instruction generates the 2's complement of (Ra), stores the result into Ra, and sets the condition code. When the maximum negative number (1000000000000000) is complemented, the overflow bit is set in SR1. If (Ra) is positive, (Ra) is not changed.

Make Negative (08 a 1) (NR a)

If (Ra) is positive, this instruction generates the 2's complement of (Ra), stores the result into Ra, and sets the condition code. If (Ra) is negative, (Ra) is not changed. If (Ra) is zero, the carry bit is set in SR1.

Two's Complement Single (08 a 4) (TCR a)

This instruction generates the 2's complement of (Ra), stores the result into Ra, and sets the condition code. When the maximum negative number is

complemented, the overflow bit is set in SR1. When zero is complemented, the carry bit is set in SR1.

Two's Complement Double (08 a 5) (TCDR a)

This instruction performs the 2's complement of double-length (Ra, Ra ⊕ 1), stores the result into Ra, Ra + 1, and sets the condition code. When the maximum negative number is complemented, bit is set in SR1. When zero is complemented, the carry bit in SR1 is set.

One's Complement Single (08 a 6) (OCR a)

This instruction performs the bit-by-bit complement of (Ra), stores the result into Ra, and sets the condition code.

LOGICAL INSTRUCTIONS

AND (Register) (60 a m) (ANDR a,m)

This instruction performs the bit-by-bit logical AND of (Ra) and (Rm), stores the result into Ra, and sets the condition code.

	1	0
1	1	0
0	0	0

AND

AND (Immediate) (61 a m) (ANDI a,m)

This instruction performs the bit-by-bit logical AND of (Ra) and the contents of memory address Y*, stores the result into Ra, and sets the condition code. The memory address is located in Rm.

AND (Constant) (62 a m) (ANDK a,y,m)

This instruction performs the bit-by-bit logical AND of (Ra) and the operand Y, stores the result into Ra, and sets the condition code.

AND (Indirect) (63 a m) (AND a,y,m)

This instruction performs the bit-by-bit logical AND of (Ra) and the contents of memory address Y, stores the result into Ra, and sets the condition code.

OR (Register) (64 a m) (ORR a,m)

This instruction performs the bit-by-bit logical OR of (Ra) and (Rm), stores the result into Ra, and sets the condition code.

	1	0
1	1	1
0	1	0

OR

OR (Immediate) (65 a m) (ORI a,m)

This instruction performs the bit-by-bit logical OR of (Ra) and the contents of memory address Y*, stores the result into Ra, and sets the condition code. The memory address is located in Rm.

OR (Constant) (66 a m) (ORK a,y,m)

This instruction performs the bit-by-bit logical OR of (Ra) and the operand Y, stores the result into Ra, and sets the condition code.

OR (Indirect) (67 a m) (OR a,y,m)

This instruction performs the bit-by-bit logical OR of (Ra) and the contents of memory address Y, stores the result in Ra, and sets the condition code.

Exclusive OR (Register) (68 a m) (XORR a,m)

This instruction performs the bit-by-bit exclusive OR of (Ra) and (Rm), stores the result into Ra, and sets the condition code.

	1	0
1	0	1
0	1	0

Exclusive OR

Exclusive OR (Immediate) (69 a m) (XORI a,m)

This instruction performs the bit-by-bit exclusive OR of (Ra) and the contents of memory address Y*, stores the result into Ra, and sets the condition code. The memory address is located in Rm.

Exclusive OR (Constant) (6A a m) (XORK a,y,m)

This instruction performs the bit-by-bit exclusive OR of (Ra) and the operand Y, stores the result in Ra, and sets the condition code.

Exclusive OR (Indirect) (6B a m) (XOR a,y,m)

This instruction performs the bit-by-bit exclusive OR of (Ra) and the contents of memory address Y, stores the result in Ra, and sets the condition code.

COMPARE INSTRUCTIONS

Compare (Register) (50 a m) (CR a,m)

This instruction arithmetically compares (Ra) to (Rm) and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged.

Compare Bit (1C a m) (CBR a,m)

This instruction tests against zero the bit in Ra corresponding to the value of m and sets the condition code. Condition code bit 8 applies to any bit tested. Bit 9 is valid only when bit 15 of Ra is tested; otherwise, it is zero. If the bit of Ra tested is zero, SRI bit 8 = 0; if the bit of Ra tested is a one, SRI bit 8 = 1.

Compare (Immediate) (51 a m) (CI a,m)

This instruction arithmetically compares (Ra) to the contents of memory address Y* and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged. Memory address is located in Rm.

Compare (Constant) (52 a m) (CK a,y,m)

This instruction arithmetically compares (Ra) to the operand Y and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged.

Compare (Indirect) (53 a m) (C a,y,m)

This instruction arithmetically compares (Ra) to the contents of memory address Y and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged.

Compare Double (Register) (54 a m) (CDR a,m)

This instruction arithmetically compares the double-length (Ra, Ra ⊕ 1) to the double-length (Rm, Rm ⊕ 1) and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged.

Compare Double (Immediate) (55 a m) (CDI a,m)

This instruction arithmetically compares the double-length (Ra, Ra ⊕ 1) to the double-length contents of memory addresses Y*, Y* ⊕ 1 and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged. The memory address is located in Rm.

Compare Double (Indirect) (57 a m) (CD a,y,m)

This instruction arithmetically compares the double-length (Ra, Ra ⊕ 1) to the double-length contents of memory addresses Y, Y ⊕ 1 and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged.

Literal Compare (CD a m) (LC a,m)

This instruction arithmetically compares the 4-bit unsigned literal contained in the m-field of the instruction with (Ra) and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged. The literal is right justified and zero filled before the compare operation.

Byte Compare (Indirect) (DB a m) (BC a,y,m)

This instruction arithmetically compares (Ra) to the selected byte of memory address Y and sets the condition code. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged. Selected byte is right justified and sign extended before the compare operations. The various sources of byte selection are described in Section 3.

Byte Compare and Index by 1 (DF a m) (BCX a,y,m)

This instruction arithmetically compares (Ra) to the selected byte of memory address Y, sets the condition code, and increments the contents of Rm by 1. Overflow and carry are set as a result of the arithmetic operation, but of no meaning. Original operands are unchanged. Selected byte is right justified and sign extended before the compare operation. Byte selection is described in Section 3.

Compare Masked (Register) (70 a m) (CMR a,m)

This instruction arithmetically compares the result of the logical AND of (Ra) and (Ra ⊕ 1) to the result of the logical AND of (Rm) and (Ra ⊕ 1) and sets the condition code.

Compare Masked (Immediate) (71 a m) (CMI a,m)

This instruction arithmetically compares the result of the logical AND of (Ra) and (Ra ⊕ 1) to the result of the logical AND of contents of memory address Y* and (Ra ⊕ 1) and sets the condition code. Memory address is located in Rm.

Compare Masked (Constant) (72 a m) (CMK a,y,m)

This instruction arithmetically compares the result of the logical AND of (Ra) and (Ra ⊕ 1) to the result of the logical AND of the operand Y and (Ra ⊕ 1) and sets the condition code.

Compare Masked (Indirect) (73 a m) (CM a,y,m)

This instruction arithmetically compares the result of the logical AND of (Ra) and (Ra ⊕ 1) to the result of the logical AND of contents of memory address Y and (Ra ⊕ 1) and sets the condition code.

SHIFT INSTRUCTIONS

Logical Right Shift (Register) (20 a m) (LRSR a,m)

This instruction shifts (Ra) right n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm.

Logical Right Shift (Constant) (22 a m) (LRS a,y,m)

This instruction shifts (Ra) right n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of operand Y.

Logical Right Double Shift (Register) (28 a m) (LRDR a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) right n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm.

Logical Right Double Shift (Constant) (2A a m) (LRD a,y,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) right n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of the operand Y.

Literal Logical Right Shift (C0 a m) (LLRS a,m)

This instruction right shifts (Ra) n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Literal Logical Right Double Shift (C2 a m) (LLRD a,m)

This instruction right shifts the double-length (Ra, Ra ⊕ 1) n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Algebraic Right Shift (Register) (24 a m) (ARSR a,m)

This instruction shifts (Ra) right n places with sign extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm.

Algebraic Right Shift (Constant) (26 a m) (ARS a,y,m)

This instruction shifts (Ra) right n places with sign extended to fill and sets the condition code; n is the value in bits 0 through 5 of the operand Y.

Algebraic Right Double Shift (Register) (2C a m) (ARDR a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) right n places with Ra sign extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm.

Algebraic Right Double Shift (Constant) (2E a m) (ARD a,y,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) right n places with Ra sign extended to fill and sets the condition code; n is the value of bits 0 through 5 of the operand Y.

Literal Algebraic Right Shift (C1 a m) (LARS a,m)

This instruction right shifts (Ra) n places with sign extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Literal Algebraic Right Double Shift (C3 a m) (LARD a,m)

This instruction right shifts the double-length (Ra, Ra ⊕ 1) n places with Ra sign extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Algebraic Left Shift (Register) (30 a m) (ALSR a,m)

This instruction shifts (Ra) left n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm. Overflow bit in SR1 set if a bit of magnitude shifts into or through the sign bit.

Algebraic Left Shift (Constant) (32 a m) (ALS a,y,m)

This instruction shifts (Ra) left n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of the operand Y.

Algebraic Left Double Shift (Register) (38 a m) (ALDR a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of Rm.

Algebraic Left Double Shift (3A a m) (ALD a,y,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 5 of the operand Y.

Literal Algebraic Left Shift (C4 a m) (LALS a,m)

This instruction left shifts (Ra) n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Literal Algebraic Left Double Shift (C6 a m) (LALD a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left n places with 0's extended to fill and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Circular Left Shift (Register) (34 a m) (CLSR a,m)

This instruction shifts (Ra) left circular n places and sets the condition code; n is the value of bits 0 through 5 of Rm.

Circular Left Shift (Constant) (36 a m) (CLS a,y,m)

This instruction shifts (Ra) left circular n places and sets the condition code; n is the value of bits 0 through 5 of the operand Y.

Circular Left Double Shift (Register) (3C a m) (CLDR a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left circular n places, with bit 15 of Ra transferred to bit 0 of Ra ⊕ 1 in each shift and sets the condition code; n is the value of bits 0 through 5 of Rm.

Circular Left Double Shift (Constant) (3E a m) (CLD a,y,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left circular n places with bit 15 of Ra transferred to bit 0 of Ra ⊕ 1 in each shift and sets the condition code; n is the value in bits 0 through 5 of the operand Y.

Literal Circular Left Shift (C5 a m) (LCLS a,m)

This instruction shifts (Ra) left circular n places and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

Literal Circular Left Double Shift (C7 a m) (LCLD a,m)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left circular n places and sets the condition code; n is the value in bits 0 through 3 of the instruction m-field.

JUMP INSTRUCTIONS (UNCONDITIONAL)

Jump (Register) (80 8 m) (JR m)

This instruction unconditionally jumps to the instruction located at the address specified by (Rm).

Jump (Constant) (82 8 m) (J y,m)

This instruction unconditionally jumps to the instruction located at the address specified by the operand Y.

Jump (Indirect) (83 8 m) (J *y,m)

This instruction unconditionally jumps to the instruction located at the address specified by the contents of memory address Y.

Local Jump (Indirect) (85 d) (LJI d)

This instruction jumps to the instruction located at the address specified by the contents of (P) + d.

The 16-bit operand d is formed by extending bit d₇ through bit positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location.

Local Jump (81 d) (LJ d)

This instruction jumps to the instruction located at memory address (P) + d.

The 16-bit operand d is formed by extending bit d₇ through bit positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location. In the IOP, if d = 0, the instruction causes a wait without referencing memory.

Local Jump, Link Memory (8D d) (LJLM d)

This instruction stores (P) + 1 at memory address (P) + d and jumps to the instruction located at memory address (P) + d + 1.

The 16-bit operand d is formed by extending bit d₇ through bit positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location.

Jump, Link Register (Register) (88 a m) (JLRR a,m)

This instruction stores (P) + 1 in Ra and jumps to the instruction located at the address specified by (Rm).

Jump, Link Register (Constant) (8A a m) (JLR a,y,m)

This instruction stores (P) + 2 in Ra and jumps to the instruction located at the address specified by the operand Y.

Jump, Link Register (Indirect) (8B a m) (JLR a,*y,m)

This instruction stores (P) + 2 in Ra and jumps to the instruction located at the address specified by the contents of address Y.

Jump, Link Memory (Constant) (8E - m) (JLM y,m)

This instruction stores (P) + 2 at memory address Y and jumps to the instruction located at memory address Y + 1.

Jump, Link Memory (Indirect) (8F - m) (JLM *y,m)

This instruction stores (P) + 2 at the address specified by the contents of address Y and jumps to the instruction located at the address specified by the contents of address Y + 1.

JUMP INSTRUCTIONS (CONDITIONAL)

Jump Equal (Register) (80 0 m) (JER m)

If the condition code represents equal, this instruction jumps to the instruction at the address specified by (Rm). If the condition code is not 00 or 10, the next instruction is executed.

Jump Equal (Constant) (82 0 m) (JE y,m)

If the condition code represents equal, this instruction jumps to the instruction located at the address specified by the operand Y. If the condition code is not 00 or 10, the next instruction is executed.

Jump Equal (Indirect) (83 0 m) (JE *y,m)

If the condition code represents equal, this instruction jumps to the instruction located at the address specified by the contents of memory address Y. If the condition code is not 00 or 10, the next instruction is executed.

Local Jump Equal (91 d) (LJE d)

If the condition code represents equal, this instruction jumps to the instruction located at memory address (P) + d. If the condition code is not 00 or 10, the next instruction is executed.

The 16-bit operand d is formed by extending bit d₇ through bits positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location. In the IOP, if d = 0, and the jump is taken, the instruction causes a wait without referencing memory.

Jump Not Equal (Register) (80 1 m) (JNER m)

If the condition code represents not equal, this instruction jumps to the instruction located at the address specified by (Rm). If the condition code is not 01 or 11, the next instruction is executed.

Jump Not Equal (Constant) (82 1 m) (JNE y,m)

If the condition code represents not equal, this instruction jumps to the instruction located at the address specified by the operand Y. If the condition code is not 01 or 11, the next instruction is executed.

Jump Not Equal (Indirect) (83 1 m) (JNE *y,m)

If the condition code represents not equal, this instruction jumps to the instruction located at the address specified by the contents of memory address Y. If the condition code is not 01 or 11, the next instruction is executed.

Local Jump Not Equal (95 d) (LJNE d)

If the condition code represents not equal, this instruction jumps to the instruction located at memory address (P) + d. If the condition code is not 01 or 11, the next instruction is executed.

The 16-bit operand d is formed by extending bit d₇ through bits positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location.

Jump Less (Register) (80 3 m) (JLSR m)

If the condition code represents less than, this instruction jumps to the instruction at the address specified by (Rm). If the condition code is not 10 or 11, the next instruction is executed.

Jump Less (Constant) (82 3 m) (JLS y,m)

If the condition code represents less than, this instruction jumps to the instruction located at the address specified by the operand Y. If the condition code is not 10 or 11, the next instruction is executed.

Jump Less (Indirect) (83 3 m) (JLS *y,m)

If the condition code represents less than, this instruction jumps to the instruction located at the address specified by the contents of memory address Y. If the condition code is not 10 or 11, the next instruction is executed.

Local Jump Less (9D d) (LJLS d)

If the condition code represents less than, this instruction jumps to the instruction located at memory address (P) + d. If the condition code is not 10 or 11, the next instruction is executed.

The 16-bit operand d is formed by extending bit d₇ through bit positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location.

Jump Greater or Equal (Register) (80 2 m) (JGER m)

If the condition code represents greater than or equal, this instruction jumps to the instruction at the address specified by (Rm). If the condition code is not 00 or 01, the next instruction is executed.

Jump Greater or Equal (Constant) (82 2 m) (JGE y,m)

If the condition code represents greater than or equal, this instruction jumps to the instruction located at the address specified by the operand Y. If the condition code is not 00 or 01, the next instruction is executed.

Jump Greater or Equal (Indirect) (83 2 m) (JGE *y,m)

If the condition code represents greater than or equal, this instruction jumps to the instruction located at the address specified by the contents of memory address Y. If the condition code is not 00 or 01, the next instruction is executed.

Local Jump Greater or Equal (99 d) (LJGE d)

If the condition code represents greater than or equal, this instruction jumps to the instruction located at memory address (P) + d. If the condition code is not 00 or 01, the next instruction is executed.

The 16-bit operand d is formed by extending bit d₇ through bit positions 8 through 15. The operand d is then algebraically added to (P). When d₇ is 0, the jump is forward, up to 127 words. When d₇ is 1, the jump is backward, up to 128 words. For backward jumps, d must contain the 2's complement of the length of the jump. For example, d = FF₁₆ causes a backward jump of one location.

Jump Overflow (Register) (80 4 m) (JOR m)

If bit 10 of status register 1 is set, this instruction jumps to the instruction at the address specified by (Rm). If bit 10 is not set, the next instruction is executed.

Jump Overflow (Constant) (82 4 m) (JO y,m)

If bit 10 of status register 1 is set, this instruction jumps to the instruction located at the address specified by the operand Y. If bit 10 is not set, the next instruction is executed.

Jump Overflow (Indirect) (83 4 m) (JO *y,m)

If bit 10 of status register 1 is set, this instruction jumps to the instruction located at the address specified by the contents of memory address Y. If bit 10 is not set, the next instruction is executed.

Jump Carry (Register) (80 5 m) (JCR m)

This instruction jumps to the instruction at the address specified by (Rm) if bit 11 of status register 1 is set. If bit 11 is not set, the next instruction is executed.

Jump Carry (Constant) (82 5 m) (JC y,m)

This instruction jumps to the instruction located at the address specified by the operand Y if bit 11 of status register 1 is set. If bit 11 is not set, the next instruction is executed.

Jump Carry (Indirect) (83 5 m) (JC *y,m)

This instruction jumps to the instruction located at the address specified by the contents of memory address Y if bit 11 of status register 1 is set. If bit 11 is not set, the next instruction is executed.

POWER-OUT-OF-TOLERANCE JUMP INSTRUCTIONS.

The three power-out-of-tolerance jump instructions cause one of the following:

- 1) The instruction that is executed before power fault or generation of thermal overload results in a no operation, and the next instruction is executed.
- 2) The instruction executed after a power fault or thermal overload from same chassis, by the CPU or IOP in a dual configuration, results in a jump to a quite bus firmware routine until power drops. These instructions should be used as the last instruction of a power or thermal interrupt program to ensure that memory is not changed during the power down sequence. In a dual CPU/IOP configuration, the IOP must be in controller mode or the IOP must also execute jump power out of tolerance instructions to ensure no memory data loss.
- 3) The instruction executed in a dual CPU/IOP configuration with an extension chassis, after the power fault or thermal overload generated by the PCM in the extension chassis, results in the following:

- a) IOP in controller mode. CPU recognizes the interrupt. CPU software executes a power-out-of-tolerance jump instruction. The jump is taken and program execution continues at (Rm), Y, or (Y).

NOTE:

If the IOP is now put in processor mode, it must execute a power-out-of-tolerance jump instruction which causes the IOP's firmware to map I/O before the IOP executes command instructions.

- b) IOP in processor mode. They both recognize interrupt. Execution in the CPU is the same as in a) if the jump is taken. To ensure no loss of data in memory in the extension chassis as power goes down, do not reference that memory via the CPU.

Execution of the power-out-of-tolerance jump instruction by the IOP results in a jump to a quite bus firmware routine until power drops in the extension chassis. The firmware then regenerates the list of available channels including only those contained in the CPU chassis, reenables the I/O and channel interrupt system, and jumps to the software address (Rm), Y, or (Y). During the firmware channel list generation, control memory locations 8, 9, and A are cleared for all channels.

- 4) The instruction executed by a CPU-only configuration with an extension chassis, after the power fault or thermal overload generated by the PCM in the extension unit, results in a quite bus firmware routine until power drops in the extension unit. The firmware then regenerates the list of available channels, including only those contained in the CPU chassis, reenables the I/O and channel interrupt system, and jumps to software address (Rm), Y, or (Y).

Jump Power Out of Tolerance (Register) (80 6 m) (JPTR m)

This instruction jumps to the firmware quite bus routine or to the instruction at the address specified by (Rm) if power is out of tolerance. If not, the next instruction is executed.

Jump Power Out of Tolerance (Constant) (82 6 m) (JPT y,m)

This instruction jumps to the firmware quite bus routine or to the instruction located at the address specified by the operand Y if power is out of tolerance. If not, the next instruction is executed.

Jump Power Out of Tolerance (Indirect) (83 6 m) (JPT *y,m)

This instruction jumps to the firmware quite bus routine or to the instruction located at the address specified by the contents of memory address if power is out of tolerance. If not, the next instruction is executed.

Jump Bootstrap 2 Selected (Register) (80 7 m) (JBR m)

This instruction jumps to the instruction located at the address specified by (Rm) if bootstrap 2 is selected. If not, the next instruction is executed.

Jump Bootstrap 2 Selected (Constant) (82 7 m) (JB y,m)

This instruction jumps to the instruction located at the address specified by the operand Y if bootstrap 2 is selected. If not, the next instruction is executed.

Jump Bootstrap 2 Selected (Indirect) (83 7 m) (JB *y,m)

This instruction jumps to the instruction located at the address specified by the contents of memory address Y if bootstrap 2 is selected. If not, the next instruction is executed.

Index Jump (Register)(84 a m) (XJR a,m)

This instruction tests (Ra). If (Ra) does not equal zero, (Ra) is decremented by 1, and the instruction jumps to the instruction located at the address specified by (Rm). If (Ra) equals zero, the next instruction is executed.

Index Jump (Constant) (86 a m) (XJ a,y,m)

This instruction tests (Ra). If (Ra) does not equal zero, (Ra) is decremented by 1, and the instruction jumps to the instruction located at the address specified by the operand Y. If (Ra) equals zero, the next instruction is executed.

Index Jump (Indirect) (87 a m) (XJ a,*y,m)

This instruction tests (Ra). If (Ra) does not equal zero, (Ra) is decremented by 1, and the instruction jumps to the instruction located at the address specified by the contents of memory address Y. If (Ra) equals zero, the next instruction is executed. This instruction causes a false protect fault when executed by the IOP if the address (Y) is in an execute protected page of memory.

Jump Zero (Register) (90 a m) (JZR a,m)

This instruction tests (Ra). If (Ra) equals zero, the instruction jumps to the instruction located at the address specified by (Rm). If (Ra) does not equal zero, the next instruction is executed.

Jump Zero (Constant) (92 a m) (JZ a,y,m)

This instruction tests (Ra). If (Ra) equals zero, the instruction jumps to the instruction located at the address specified by the operand Y. If (Ra) does not equal zero, the next instruction is executed.

Jump Zero (Indirect) (93 a m) (JZ a,*y,m)

This instruction tests (Ra). If (Ra) equals zero, the instruction jumps to the instruction located at the address specified by the contents of memory address Y. If (Ra) does not equal zero, the next instruction is executed.

Jump Not Zero (Register) (94 a m) (JNZR a,m)

This instruction tests (Ra). If (Ra) does not equal zero, the instruction jumps to the instruction located at the address specified by (Rm). If (Ra) equals zero, the next instruction is executed.

Jump Not Zero (Constant) (96 a m) (JNZ a,y,m)

This instruction tests (Ra). If (Ra) does not equal zero, the instruction jumps to the instruction located at the address specified by the operand Y. If (Ra) equals zero, the next instruction is executed.

Jump Not Zero (Indirect) (97 a m) (JNZ a,*y,m)

This instruction tests (Ra). If (Ra) does not equal zero, the instruction jumps to the instruction located at the address specified by the contents of memory address Y. If (Ra) equals zero, the next instruction is executed.

Jump Positive (Register) (98 a m) (JPR a,m)

This instruction tests (Ra). If (Ra) is equal to or greater than zero, the instruction jumps to the instruction located at the address specified by (Rm). If (Ra) is less than zero, the next instruction is executed.

Jump Positive (Constant) (9A a m) (JP a,y,m)

This instruction tests (Ra). If (Ra) is equal to or greater than zero, the instruction jumps to the instruction located at the address specified by the operand Y. If (Ra) is less than zero, the next instruction is executed.

Jump Positive (Indirect) (9B a m) (JP a,*y,m)

This instruction tests (Ra). If (Ra) is equal to or greater than zero, the instruction jumps to the instruction located at the address specified by the contents of memory address Y. If (Ra) is less than zero, the next instruction is executed.

Jump Negative (Register) (9C a m) (JNR a,m)

This instruction tests (Ra). If (Ra) is less than zero, the instruction jumps to the instruction located at the address specified by (Rm). If (Ra) is equal to or greater than zero, the next instruction is executed.

Jump Negative (Constant) (9E a m) (JN a,y,m)

This instruction tests (Ra). If (Ra) is less than zero, the instruction jumps to the instruction located at the address specified by the operand Y. If (Ra) is equal to or greater than zero, the next instruction is executed.

Jump Negative (Indirect) (9F a m) (JN a,*y,m)

This instruction tests (Ra). If (Ra) is less than zero, the instruction jumps to the instruction located at address specified by the contents of memory address Y. If (Ra) is equal to or greater than zero, the next instruction is executed.

AN/AYK-14(V) stop instructions are of three types: Jump After Stop, Jump After Stop If Stop Key 1 Set, and Jump After Stop If Stop Key 2 Set. These instructions are implemented in RR, RK, and RX formats. The jump destination is loaded into the program address register and a stop occurs if the relevant conditions are obtained.

NOTE:

The stop portion of the instruction will not occur if a L/V or CCU is not connected.

Jump After Stop (Register) (80 9 m) (JSR m)

The jump destination is loaded and a check is made to see if computer support equipment is connected. If it is, the AN/AYK-14(V) software execution stops and, on restart, jumps to the address specified (Rm). If no computer support equipment is connected, the jump is executed without the stop. This instruction is an executive mode instruction. In dual CPU/IOP systems, this instruction causes the IOP to enter controller mode if executed in the IOP.

Jump After Stop (Constant) (82 9 m) (JS y,m)

The jump destination is loaded and a check is made to see if computer support equipment is connected. If it is, the AN/AYK-14(V) software execution stops and, on restart, jumps to the address specified by the operand Y. If no computer support equipment is connected, the jump is executed without the stop. This instruction is an executive mode instruction. In dual CPU/IOP systems, this instruction causes the IOP to enter controller mode if executed in the IOP.

Jump After Stop (Indirect) (83 9 m) (JS *y,m)

The jump destination is loaded and a check is made to see if computer support equipment is connected. If it is, the AN/AYK-14(V) software execution stops and, on restart, jumps to the address specified by the contents of memory address Y. If no computer support equipment is connected, the jump is executed without the stop. This instruction is an executive mode instruction. In dual CPU/IOP systems, this instruction causes the IOP to enter controller mode if executed in the IOP.

Jump After Stop if Key 1, 2 Set (Register) (80 A m; 80 B m)
(JKSR1 m; JKSR2 m)

The jump destination is loaded and a check is made to see if computer support equipment is connected and if Stop Key 1 or 2 is set. If both conditions are met, AN/AYK-14(V) software execution stops and, on restart,

jumps to the address specified by (Rm). If either condition is not met, the jump is executed without the stop. These instructions are executive mode instructions.

Stop keys 1 and 2 are set via request from the CCU (see CCU User's Manual).

Jump After Stop if Key 1, 2 Set (Constant) (82 A m; 82 B m) (JKS1 y,m; JKS2 y,m)

The jump destination is loaded and a check is made to see if computer support equipment is connected and if Stop Key 1 or 2 is set. If both conditions are met, AN/AYK-14(V) software execution stops and, on restart, jumps to the address specified by the operand Y. If either condition is not met, the jump is executed without the stop. These instructions are executive mode instructions.

Stop keys 1 and 2 are set via a request from the CCU (see CCU User's Manual).

Jump After Stop if Key 1, 2 Set (Indirect) (83 A m; 83 B m) (JKS1 *y,m; JKS2 *y,m)

The jump destination is loaded and a check is made to see if computer support equipment is connected and if Stop Key 1 or 2 is set. If both conditions are met, AN/AYK-14(V) software execution stops and, on restart, jumps to the address specified by the contents of memory address Y. If either condition is not met, the jump is executed without the stop. These instructions are executive mode instructions.

Stop keys 1 and 2 are set via a request from the CCU (see CCU User's Manual).

MISCELLANEOUS INSTRUCTIONS

Executive Return (0C a 0) (ER a)

This instruction generates the executive return interrupt and stores the contents of the program address register plus one (P) + 1 into general register Ra. If locked out, this interrupt is lost.

Set Bit (14,a m) (SBR a,m)

This instruction sets the bit in Ra corresponding to the value in the m-field and sets the condition code.

Zero Bit (18 a m) (ZBR a,m)

This instruction clears the bit in Ra corresponding to the value in the m-field and sets the condition code.

Input/Output Command (74 - -) (IOCR)

This instruction causes execution of the I/O command instruction in the command cells (main memory address 0060₁₆ and 0061₁₆) and then clears bits 14 and 15 of command cell 0060₁₆. This is an executive mode instruction.

In the IOP, the command cells are 0062₁₆ and 0063₁₆.

Biased Fetch (Immediate) (75 - m) (BFI m)

This instruction sets the condition codes based on the original value of memory at address Y*, and then sets the most significant two bits at the memory location leaving the remaining bits unchanged. When executed, the biased fetch queues on the two most significant bits of the operand word. This instruction does not prevent another processor from accessing the same memory location (memory access protect) at the same time as the CPU with instructions other than stack, queue, and biased fetch.

Biased Fetch (Indirect) (77 - m) (BF y,m)

This instruction sets the condition codes based on the original value of memory at address Y, then sets the most significant two bits at the memory location leaving the remaining bits unchanged. When executed, the biased fetch queues on the two most significant bits of the operand word. This instruction does not prevent another processor from accessing the same memory location (memory access protect) at the same time as the CPU with instructions other than stack, queue, and biased fetch.

Remote Execute (76 - m) (REX y,m)

This instruction causes the instruction at the address specified by the contents of memory address Y to be executed and, upon completion of that instruction, the program continues with the next sequential instruction if the remote instruction did not result in a modification to the P register.

Initiate Processor Interrupt (08 a 3) (IPI a)

This instruction generates one of the processor interrupts as specified by the least significant bit (LSB) of the contents of the register specified by the a field as follows:

LSB (a) = 0	Processor interrupt 0
LSB (a) = 1	Processor interrupt 1

When the IPI is executed by a CPU, the interrupt will interrupt the IOP processing if an IOP is in the system and in processor mode; otherwise, it interrupts the CPU. When the IPI is executed by an IOP, it will interrupt a CPU if a CPU is in the system, or else the IOP is interrupted. This is an executive mode instruction.

IPL Failed (08 - C) (IPLF)

This instruction sets the IPL failed discrete, indicating an initial program load failure by the bootstrap loader. This is an executive mode instruction.

NOTE:

In a dual system (CPU and IOP), the CPU instructions IOCR and IPI must not be included in the program until approximately 15 microseconds (15 instructions) after power up, or a master clear or an IOP event test error will occur.

Diagnostic Jump (08 a D) (DJ a)

This instruction enables or disables a hardware fault warning interrupt as follows:

Bit 15 of R15 ₁₀ = 1 and	
Bit 0 of R15 ₁₀ = 0	Enable Hardware Fault Warning Interrupt
= 1	Disable Hardware Fault Warning Interrupt

Bit 15 of R15₁₀ = 0
Bits 11 through 0 of R15₁₀ contain a micromemory jump address.

This is an executive mode instruction.

Masked Substitute (Register) (6C a m) (MSR a,m)

For each bit set in (Ra ⊕ 1), the value of the corresponding bit in (Rm) is transferred to the corresponding bit in Ra. For each bit not set in (Ra ⊕ 1), the corresponding bit in Ra is unaltered. The condition code is set.

Masked Substitute (Immediate) (6D a m) (MSI a,m)

For each bit set in (Ra ⊕ 1), the value of the corresponding bit in the contents of memory address Y* is transferred to the corresponding bit in Ra. For each bit not set in (Ra ⊕ 1), the corresponding bit in Ra is unaltered. The condition code is set.

Masked Substitute (Constant) (6E a m) (MSK a,y,m)

For each bit set in (Ra ⊕ 1), the value of the corresponding bit in the operand Y is transferred to the corresponding bit in Ra. For each bit not set in (Ra ⊕ 1), the corresponding bit in Ra is unaltered. The condition code is set.

Masked Substitute (Indirect) (6F a m) (MS a,y,m)

For each bit set in (Ra ⊕ 1), the value of the corresponding bit in the contents of memory at address Y is transferred to the corresponding bit in Ra. For each bit not set in (Ra ⊕ 1), the corresponding bit in Ra is unaltered. The condition code is set.

Reverse Register (10 a 1) (RVR a)

This instruction reverses the order of the 16 bits contained in the general register specified by the a field (bit 15 to bit 0, bit 0 to bit 15, and so forth). The condition code is set.

Count One's (10 a 2) (CNT a)

This instruction counts the number of set bits in (Ra) and stores the count into Ra + 1.

Scale Factor (10 a 3) (SFR a)

This instruction shifts the double-length (Ra, Ra ⊕ 1) left algebraically, with 0's extended to fill, until bit 15 of Ra does not equal bit 14 of Ra and stores the shift count into Ra + 2. If (Ra, Ra ⊕ 1) is all 0's or all 1's, the shift count becomes 31.

INTERRUPT PROCESSING

AN/AYK-14(V) interrupts are divided into three classes (Table 9-1) which correspond to their function. Class I interrupts indicate hardware failures or hardware functions; Class II interrupts indicate software failures or software functions; Class III interrupts indicate I/O failures or I/O functions. In case of conflicts (two events active during one check for active events), Class I interrupts have priority over Class II interrupts, which in turn have priority over Class III interrupts. Within each class as well, a priority scheme is established.

Status register 1, bits 3 through 1 specify interrupt lockouts by class. If bit 3 is clear, Class I interrupts are locked out; if bit 2 is clear, Class II interrupts are locked out, and if bit 1 is clear, Class III interrupts are locked out. Certain interrupts such as power fault and thermal overload cannot be locked out. Table 9-1 shows the effect of lockouts on the various interrupts. Interrupts which are queued and pending are honored when the lockout is lifted; multiple pending interrupts are handled according to the priority scheme.

When the IOP is in controller mode, the IOP passes all Class III interrupts to the CPU. Class I and III interrupts are handled directly by the CPU.

When in the dual processor mode, the IOP handles all interrupts indicating hardware faults associated with the modules it is using and software interrupts within the program it executes.

Hardware fault warning interrupts are queued one level as follows:

- CPU - 13 times, 14th time no lockout
- IOP - 6 times, 7th time no lockout

Therefore, when in the dual processor mode, both processors will act upon all interrupts associated with their responsibilities.

The firmware processing sequence for an interrupt requires the following steps:

- 1) On determination that an interrupt is to be generated, the firmware checks to see whether the interrupt is locked out.
- 2) If the lockout bit for the specified class of interrupt is set, the firmware determines appropriate action to be taken depending on the interrupt (refer to Table 9-1).
- 3) If no lockout is set or the interrupt cannot be locked out, the processing routine is entered at the completion of a software instruction. The current program status word (PSW), consisting of

TABLE 9-1. INTERRUPT LOCKOUT EFFECTS

Class	Priority In Class	Binary Code	Interrupt	Lockout Effect
I (Hardware)	1	00000	Power fault interrupt (input power)	No lockout
	2	00010	Memory timeout (memory module)	Lost
	3	00100	Memory parity	Lost
	4	00110	Hardware fault warning	Queued 1 level, 13 times; 14th no lockout (IOP is 6 times, 7th no lockout)
	5	01010	I/O failure (I/O module not in configuration)	Lost
	6	01100	Thermal overload	No lockout
	7	01110	Hardware fault (BIT indicator set)	Queued 1 level
II (Software)	1	00000	CP instruction fault	No lockout
	2	00010	I/O instruction fault	No lockout
	3	00100	Floating point over/underflow	Lost
	4	00110	Executive return	Lost
	5	10000	Executive mode instruction fault	No lockout
	6	11000	Memory protect fault	Lost
	7	01000	RTC overflow	Queued 1 level
	8	01010	Monitor clock	Queued 1 level
	9	10110	System reset	No lockout
	10	01100	Processor interrupt 0	Queued 1 level
	11	01110	Processor interrupt 1	Queued 1 level
III (I/O)	1	110	I/O channel abnormal (ERI)	Queued 1 level (per channel)
	2	000	External interrupt (EII)	Queued 1 level (per channel)
	3	100	Output chain interrupt (OCI)	Queued 1 level (per channel)
	4	010	Input chain interrupt (ICI)	Queued 1 level (per channel)

the current program counter and status registers, as well as the real-time clock values, are saved. The locations in which these values are saved are shown in Table 9-2. If the interrupt is locked out, it is queued or lost, and the instruction execution is continued.

- 4) New values for the program counter and status registers are then loaded from the locations indicated in Table 9-2. The program counter reload values are computed from a base address, to which is added an index, depending on the binary code of the interrupt (see Table 9-1).
- 5) The firmware automatically clears bit 15 of status register when it is loaded in step 4, setting the AN/AYK-14(V) into executive mode.
- 6) The processor then begins execution from the new program counter value. The new status register 1 value determines the lockouts in effect at the beginning of the interrupt software.

NOTE:

Acknowledgement of a second interrupt of the same class (i.e, using the same locations for storing the PSW) causes the original program counter value to be lost. To prevent this, as far as possible, the lockout for the class of interrupts of the received interrupt should be locked out during the interrupt processing routine. Interrupts which occur while the exchange of PSWs is taking place are not recognized until after interrupt processing by firmware is complete.

CLASS I INTERRUPTS

POWER FAULT PRIORITY 1 BINARY CODE 00000

The PCM generates the power fault interrupt when an output voltage out of tolerance state or an input power failure is detected. Either condition causes power to the AN/AYK-14(V) to go down approximately 300 microseconds after the interrupt is generated. This interrupt cannot be locked out. When the PCM output power is out of tolerance and the input power is good, the bit indicator is set by the PCM by generating the hardware fault interrupt as well as the power fault interrupt. The users program will be vectored to the power fault interrupt because of its higher priority. This interrupt causes input/output data transfers and the I/O channel interrupt system to be cleared and disabled. The firmware available channels list is also cleared.

MEMORY TIME-OUT PRIORITY 2 BINARY CODE 00010

The memory time-out interrupt is generated by the MCM when CMM or SMM fails to provide a resume signal within 2 microseconds after a memory request. This interrupt is generated if the software addresses memory locations not provided in the hardware configuration. The memory time-out interrupt can also be caused by a MCM control circuitry failure. If Class I interrupts are locked out, this interrupt is lost. The firmware sets a bit pattern in

TABLE 9-2. ASSIGNED MEMORY ADDRESSES

Function	CPU			IOP		
	I	II	III	I	II	III
	Hex	Hex	Hex	Hex	Hex	Hex
Store P	58	50	48	68	70	48
Store SR1	59	51	49	69	71	49
Store SR2	5A	52	4A	6A	72	4A
Store RTC lower	5B	53	4B	6B	73	4B
Preload	5C	54	4C	6C	74	4C
SR1 reload	5D	55	4D	6D	75	4D
SR2 reload	5E	56	4E	6E	76	4E
Store RTC upper	5F	57	4F			
I/O command cell	60-61 ₁₆			62-63 ₁₆		
Auto start entrance	7F ₁₆					
External interrupt word storage	80-8F ₁₆			80-8F ₁₆		
Bootstrap ROM	0-3F ₁₆ and C0-3FF ₁₆			0-3F ₁₆ and C0-3FF ₁₆		

the lower 8 bits of status register 2 upon generating this interrupt. Bits 7 through 4 give the memory bank code of the failing module. This code is the upper 4 bits of the 19-bit absolute memory address computed by the MCM. Bit 3 of status register 2 indicates whether the odd (=1) or even (=0) memory bus is used. Bits 2 through 0 of status register 2 are used to distinguish types of memory failure. For memory time-out interrupts, bit 0 is set. If a memory protect fault occurs together with the memory time-out, bit 1 of status register 2 is also set. In the case of multiple memory failures of different types, only the highest priority interrupt is generated. The 3 lowest order bits of status register 2 must be examined to determine if other failures occurred (Figure 4-2). This interrupt may be lost if it occurs while the firmware is processing the hardware fault warning, I/O failure, or hardware fault interrupt.

MEMORY PARITY PRIORITY 3 BINARY CODE 00100

The MCM generates the memory parity error interrupt when a parity error is detected. If Class I interrupts are locked out, this interrupt is lost. Receipt of this interrupt indicates that no memory time-out occurred (time-out has higher priority) but a protect fault is possible. The firmware indicates the memory bank code and bus code as for the memory time-out interrupt; bit 1 of status register 2 is set for memory parity errors. This interrupt may be lost if it occurs while the firmware is processing the hardware fault warning, I/O failure, or hardware fault interrupt.

HARDWARE FAULT WARNING PRIORITY 4 BINARY CODE 00110

The hardware fault warning interrupt is generated by the PSM when the BIT counter increments ever 2^{21} microseconds (approximately 2.097) seconds after:

- 1) Power-up
- 2) Execution of Reset Bit Timer instruction.
- 3) Execution of Set Bit Indicator instruction.

This interrupt indicates a possible hardware fault. The software is expected typically to reset the BIT timer before 2^{21} microseconds elapse (on the order of one million instructions). The interval prior to generation of the hardware fault warning can be extended by lockout of Class I interrupts. In this case, up to 13 interrupts are queued in a one-level queue. The fourteenth interrupt, however, is generated and received by the software (14 interrupt times provide an interval of approximately 29.36 seconds prior to generation of the interrupt). The BIT counter is incremented every 2^{21} microseconds regardless of lockout conditions. The execution of the Diagnostic Jump instruction with bits 15 and 0 of R15 set, performs the same inhibiting function as the Class I lockout, but does not affect the other Class I interrupts. While the software execution is stopped and CCU is connected, the generation of the interrupt is inhibited.

I/O FAILURE PRIORITY 5 BINARY CODE 01010

The I/O failure interrupt is generated by the GPM when the software references an I/O channel not provided in the hardware configuration. If Class I interrupts are locked out, the I/O failure interrupt is lost. When an I/O failure interrupt is generated, the firmware places the logical channel number of the referenced channel in bits 7 through 4 of status register 2.

THERMAL OVERLOAD PRIORITY 6 BINARY CODE 01100

The PCM generates the thermal overload interrupt when a temperature out of tolerance condition is detected. Such a condition shall cause power to go off approximately 300 microseconds after the interrupt is generated. This interrupt cannot be locked out. This interrupt causes input/output data transfers and the I/O channel interrupt system to be cleared and disabled. The firmware available channels list is also cleared.

HARDWARE FAULT PRIORITY 7 BINARY CODE 01110

The hardware fault interrupt is generated when the BIT indicator is set under the following conditions:

- 1) PCM output power out of tolerance
- 2) Execution of SET BIT INDICATOR (SBT) instruction
- 3) BIT counter increments to 15 (all 1's) (7 in the EOP)
- 4) CPUBUS or IOBUS time-out (firmware generated)
- 5) Event system failure

The output power out of tolerance condition also generates the power fault interrupt. The hardware fault warning interrupt is generated each time the BIT counter increments. The hardware fault warning interrupt may be inhibited (see preceding description); in which case, 13 (6 in IOP) interrupts are successively queued one level. The fourteenth (seventh in IOP) interrupt is not locked out. The system then has approximately 2.097 seconds to reset the BIT counter before the BIT indicator is set and hardware fault interrupt generated.

If a CPUBUS or IOBUS time-out occurs, the GPM is not able to access memory or I/O modules. The time-out limit is 30 to 40 microseconds.

The hardware fault interrupt is placed in a one-level queue if Class I interrupts are locked out.

NOTE:

The BIT indicator is set regardless of lockout conditions.

The hardware fault event clears any hardware fault warning events that may be active.

CLASS II INTERRUPTS

CLASS INSTRUCTION FAULT PRIORITY 1 BINARY CODE 00000

The CP instruction fault interrupt is generated when the processor attempts to execute an instruction not in the repertoire, including attempts to execute I/O instructions in the processor without an IOCR instruction. The value of the program counter saved by the interrupt processing routine will be the address plus one of the illegal instruction or the address plus two of illegal instructions 46_{16} and $4E_{16}$ causing the interrupt. The CP instruction fault interrupt cannot be locked out.

I/O INSTRUCTION FAULT PRIORITY 2 BINARY CODE 00010

The I/O instruction fault is generated when the processor attempts to execute an instruction using an IOCR or in a chain which is not in the I/O instruction repertoire. The chain in question is halted when the I/O instruction fault is a chain violation (channel operation is not otherwise affected) and (CAP) equals the address of the instruction causing the interrupt plus one. The I/O instruction fault interrupt cannot be locked out. The value of the program counter saved will be the address plus one of the IOCR if the instruction in the command cell is not a command instruction.

FLOATING-POINT OVER/UNDERFLOW PRIORITY 3 BINARY CODE 00100

The floating-point over/underflow interrupt is generated when a floating-point arithmetic instruction produces a characteristic result which is too large or too small to be accurately represented and bit 7 of SR1 is zero. The section on floating-point arithmetic supplies details. The program counter value saved is the address plus one of the floating-point instruction which caused the interrupt. The floating-point over/underflow interrupt is lost if Class II interrupts are locked out or it is disabled by SR1 bit 7.

EXECUTIVE RETURN PRIORITY 4 BINARY CODE 00110

The executive return interrupt is generated upon execution of the Executive Return instruction (OC a 0). The program counter value saved is the address plus one of the executive return instruction, and P+1 is stored in Ra. The executive return interrupt is lost, and the Executive Return instruction is a no operation if Class II interrupts are locked out.

EXECUTIVE MODE FAULT PRIORITY 5 BINARY CODE 10000

The executive mode fault interrupt is generated when the CPU attempts to execute an executive mode instruction when not in the executive mode (the executive mode instructions are indicated in the repertoire). The program counter value saved is the address of the instruction causing the interrupt. The executive mode instruction fault interrupt cannot be locked out.

MEMORY PROTECT FAULT PRIORITY 6 BINARY CODE 11000

The memory protect fault interrupt is generated by the MCM when an attempt is made to reference a protected page of memory (see section on paging).

Bits 7 through 4 of status register 2 provide the memory bank code, and bit 3 of status register 2 provides the odd or even bus indicator for the location referenced. Bit 2 of status register 2 is set for the memory protect fault interrupt. Receipt of the interrupt implies that no memory time-out or parity errors occurred on the memory reference. If a memory protect fault occurs with one of the high priority memory faults, the high-est priority interrupt involved is generated but bits 2 through 0 indicate all detected failures. The memory protect fault interrupt is lost if Class II interrupts are locked out. This interrupt may also be lost if it occurs while the firmware is processing the RTC overflow, monitor clock, system reset, processor interrupt 0, or processor interrupt 1 interrupt.

REAL-TIME CLOCK OVERFLOW PRIORITY 7 BINARY CODE 01000

The real-time clock overflow interrupt is generated each time the RTC passes a multiple of 2^{16} (i.e., each time the upper 16 bits of the clock are incremented). The section on AN/AYK-14(V) clocks provides instructions for controlling the RTC and its interrupt. The RTC interrupt is placed in a one-level queue if Class II interrupts are locked out.

MONITOR CLOCK PRIORITY 8 BINARY CODE 01010

The monitor clock interrupt is generated when the monitor clock counts down to 0. The section on AN/AYK-14(V) clocks provides instruction which control the monitor clock and its interrupt. The monitor clock interrupt is placed in a one-level queue if Class II interrupts are locked out.

SYSTEM RESET PRIORITY 9 BINARY CODE 10110

The system reset interrupt is generated when the external system reset signal goes active. The system reset interrupt cannot be locked out.

PROCESSOR INTERRUPTS 0 and 1 PRIORITIES 10, 11 BINARY CODES 01100, 01110

The processor interrupts 0 and 1 are generated by the execution of the Initiate Processor Interrupt (IPI) instruction. If an IOP is included in the system, these interrupts provide communication between the IOP and CPU. If the IPI instruction executed in the CPU and the IOP is in processor mode, the IOP will receive the interrupt. If the IPI instruction is executed in the IOP, the CPU will receive the interrupt. If no IOP is present, these interrupts merely interrupt the CPU. The processor interrupts are placed in one-level queues if Class II interrupts are locked out. The Initiate Processor Interrupt instruction is an executive mode instruction.

CLASS III INTERRUPTS

I/O CHANNEL ABNORMAL PRIORITY 1 BINARY CODE 110

The I/O channel abnormal (ERI) interrupt is generated when an I/O channel detects an abnormal or error condition. Descriptions of the I/O modules supply details on conditions generating this interrupt. The I/O channel abnormal interrupts for a channel are placed in a one-level queue if Class III interrupts are locked out.

EXTERNAL PRIORITY 2 BINARY CODE 000

The external (EII) interrupt is generated by certain I/O modules upon receipt of an external command word, function word, or interrupt signal. The I/O module descriptions supply details. The EII interrupts for a given channel are placed in a one-level queue if Class III interrupts are locked out.

OUTPUT CHAIN INTERRUPT PRIORITY 3 BINARY CODE 100

The output chain interrupt (OCI) is generated by certain I/O modules upon execution of the Interrupt Processor instruction in an output chain program. The I/O module descriptions supply details. Output chain interrupts for a given channel are placed in a one-level queue if Class III interrupts are locked out.

INPUT CHAIN INTERRUPT PRIORITY 4 BINARY CODE 010

The input chain interrupt (ICI) is generated by certain I/O modules upon execution of the Interrupt Processor instruction in an input chain program. The I/O module descriptions supply details. Input chain interrupts for a given channel are placed in a one-level queue if Class III interrupts are locked out.

GENERAL

All I/O is based on the type of devices to be used in the system and the interface required to service them. The following 10 I/O channels are presently available on the AN/AYK-14(V) computer.

- Serial interface module (1553A)
- Discrete interface module
- NTDS parallel interface module (Slow)
- NTDS parallel interface module (Fast)
- NTDS parallel interface module (ANEW)
- NTDS serial interface module
- RS-232-C interface module
- PROTEUS interface module
- PIC/POC/SOC module
- Discrete input/output module

The AN/AYK-14(V) can accommodate up to seven of these I/O channel modules to provide communication between the computer and peripheral equipment and/or other computers.

Each I/O channel is assigned a unique 4-bit logical channel number and a unique 4-bit channel type code. The 4-bit logical channel number is used by the software to address a given channel and the 4-bit channel type code is used by the firmware in resolving I/O channel access conflicts. Any combination of the I/O channel modules may be intermixed within the physical constraints of a given AN/AYK-14(V) computer configuration.

All I/O channel types utilize the following three functional requirements.

- A 16-word by 16-bit control memory block
- I/O channel chain programs
- Processor to I/O channel communications.

CONTROL MEMORY

The 16-word by 16-bit control memory associated with each I/O channel contains the parameters which direct and control the operation of the I/O channel. Chain addresses, data buffer addresses, data buffer word counts, data word length definitions, channel mode definitions, and parity bit control are examples of the type of information contained in the control memory parameters. The format and definition of each location in a control memory is dependent on the I/O channel type and is discussed in the paragraphs for the I/O channel types. There is one restriction to the use of 16 words of control memory even though all of the I/O channel types incorporate 16 words. The IOP supports only the first 12 words, therefore, control memory locations C, D, E, and F (hexadecimal) are illegal in any CPU and IOP combined configuration or any IOP standalone configuration. Location B (hexadecimal) is reserved and used as a one-word buffer register by all I/O

channel types for output information transfer activity (i.e., contains the contents of the next main memory address to be output).

The I/O instructions are divided into command instructions executable via an input/output command instruction (IOCR) in the program and chaining instructions executable in a channel chain program. The IOCR instruction, encountered during software program execution, causes the execution of the I/O command instruction located at the command cells in main memory, addresses 0060₁₆ and 0061₁₆ (0062 and 0063 in IOP). I/O command instructions are illegal instructions during software program execution unless encountered via the IOCR instruction.

The main program, through the use of the legal IOCR instruction, can start and stop an I/O chain program, monitor and modify channel operations, and control memory locations.

Figure 10-1 shows I/O chain program initiation using the IOCR instruction. After initiation by the processor, subsequent I/O channel operations on the selected channel are controlled by the I/O chain program instructions in main memory.

NOTE:

Once initiated by the processor, execution of the I/O chain program on the selected I/O channel is independent of the processor software. Several I/O channels may be executing I/O chain programs at the same time.

When an I/O channel has an I/O chain program in execution, the following two forms of activity may be in progress.

- I/O information transfer activity
- I/O chaining activity

Under certain conditions, various I/O channel types may have both forms of activity in progress at the same time.

I/O chaining activity exists on an I/O channel when sequential I/O chain program instructions are being executed on that channel. These I/O chain instructions normally provide for transferring parameters between the main memory and the I/O channel control memory and for initiation of transfer of blocks or buffers of data and/or control words on the interface lines. When an I/O chain instruction that initiates a transfer on the interface lines is encountered, the chaining activity is halted and transfer activity begins. Normally, upon completion of the transfer activity, I/O chaining activity is reinitiated. See Figure 10-2. When a chain is running (no I/O being done), all lower priority chains are suspended.

Channel priority for chain execution is the same as channel priority for any I/O activity where channel 0 is the highest priority and channel F₁₆ is the lowest priority. Chaining is always performed from the highest priority channel first to the lowest priority channel.

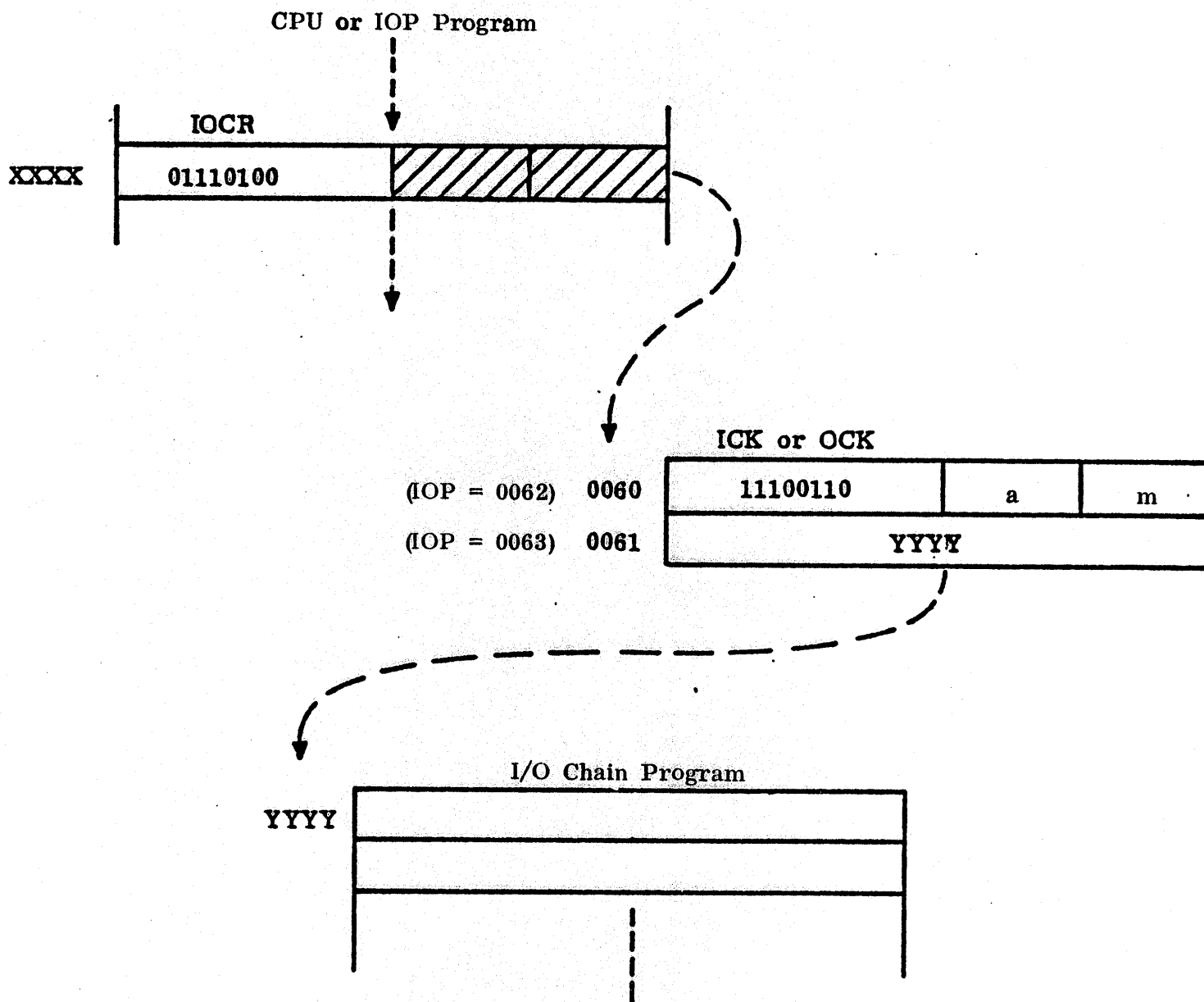


Figure 10-1. I/O Chain Program Initiation

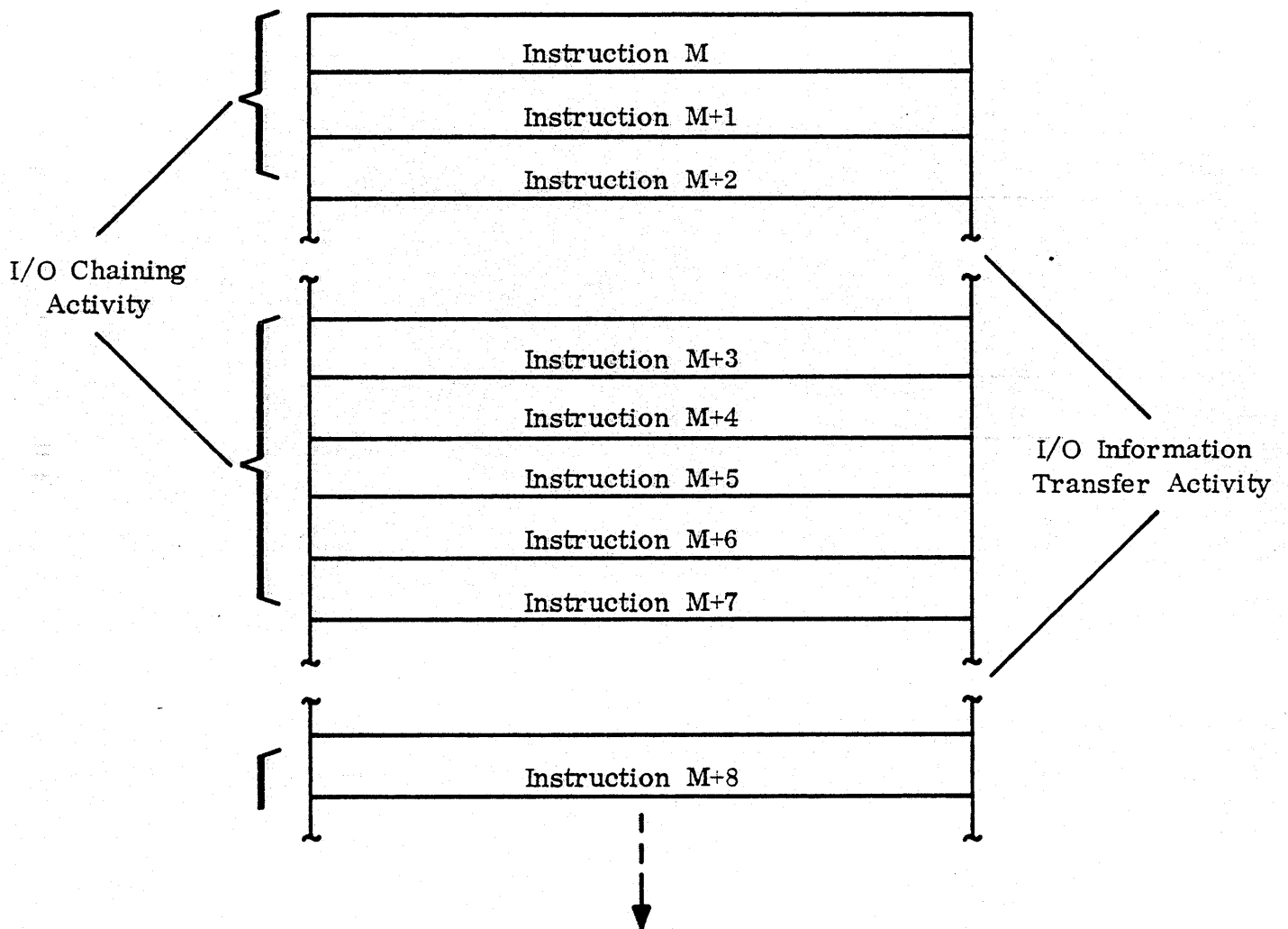


Figure 10-2. I/O Chain Program Operation

In a CPU only configuration, the PSM module regulates chain instruction execution so as to prevent the CPU from processing only chain instructions. One chain instruction is executed every 16 microseconds regardless of how many chain programs have been initiated.

There are I/O chain instructions that provide branching or jumping within the I/O chain program and halting of the chain program. Certain I/O channel types allow for the execution of two I/O chain programs at the same time: one I/O chain program performs I/O information input transfers while the second I/O program performs I/O information output transfers.

NOTE:

All I/O information transfers are subject to any existing main memory page read/write protection as defined by the CPU program.

PROCESSOR TO I/O CHANNEL COMMUNICATION

The AN/AYK-14(V) processor software may initiate I/O channel operation via the processor IOCR instruction which executes the appropriate I/O command instruction in address 0060₁₆ (0062₁₆ in IOP) of main memory. Several I/O command instructions may be executed out of sequence via the processor IOCR instruction to perform additional communications between the processor software and the I/O channel. The IOCR executable I/O command instructions are as follows:

- 1) Channel Control (EO a m) (ACR m;CCR a,m) Perform the operation specified by the m-field:

<u>m</u>	<u>Operation</u>
0000	Master clear all channels
0001	I/O instruction fault interrupt
0010	I/O instruction fault interrupt
0011	I/O instruction fault interrupt
0100	Set EIE on all channels (enable external interrupts)
0101	Clear EIE on all channels (disable external interrupts)
0110	Enable Class III priority 2,3, and 4 interrupts on all channels
0111	Disable Class III priority 2,3, and 4 interrupts on all channels
1000	Master clear the channel specified by the a-field
1001	I/O instruction fault interrupt
1010	I/O instruction fault interrupt
1011	I/O instruction fault interrupt
1100	Set EIE on the channel specified by the a-field (enable external interrupts)
1101	Clear EIE on the channel specified by the a-field (disable external interrupts)
1110	Enable Class III priority 2,3, and 4 interrupts on the channel specified by the a-field
1111	Disable Class III priority 2,3, and 4 interrupts on the channel specified by the a-field

NOTE

The a-field is not used if the instruction is executed in a chain program.

- 2) Initiate Chain (E6 a m) (ICK a,y; OCK a,y) Load the control memory location specified by the m-field for the I/O channel specified by the a-field with the value y and then initiate I/O channel operation as specified by the m-field as follows:

<u>m-field</u>	<u>Operation</u>
2	Initiate I/O channel input chain program as specified by the CAP in control memory location 2.
6	Initiate I/O channel output chain program as specified by the CAP in control memory location 6.

NOTE

For the SIM I/O channel type, control memory location 6 is used as CAP regardless of bit 2 in the m-field. If m=2 or 6, the control memory is loaded but the chain program is not initiated.

- 3) Write Control Memory (E7 a m) (WIM a,y,m) Load the control memory location specified by the m-field for the I/O channel specified by the a-field with the contents of main memory address y.
- 4) Read Control Memory (EB a m) (RIM a,y,m) - Store the contents of control memory location specified by the m-field for the I/O channel specified by the a-field at main memory address y.
- 5) Set/Clear Discretes (F8 a m) (SICR a,m) - Set or clear the discrete as specified by the m-field on the I/O channel specified by the a-field. The unique m-field specified operation is identical to that defined when an operation code of F8 is executed by the I/O chain program for a given I/O channel type. If the operation code F8 is a NOP in an I/O chain program for the given I/O channel type, then this instruction is also a NOP for that I/O channel type.
- 6) Store Status (FB a m) (SST a,y,m) - Store the channel status as specified by the m-field for the I/O channel specified by the a-field in main memory location y. The m-field specification is identical to that defined when an operation code of FB is executed by the I/O chain program for a given I/O channel type.
- 7) Start IOP (FC-m) (SIOP m,y) - Perform the operation specified by the m-field as follows:

<u>m-field</u>	<u>Operation</u>
XX00	Clear bit 12 of the IOP status register 1 and start the IOP executing instructions as a processor at starting address y in main memory.

- XX01 Set bit 12 of the IOP status register 1, y P, and start IOP.
- XX10 Clear bit 12 of the IOP status register 1.
- XX11 Set bit 12 of the IOP status register 1.

NOTE:

This I/O command instruction, with the m-field equal to 000X binary, functionally disconnects the IOP and all I/O channels from the CPU and initiates IOP to execute its own processor program using a defined subset of CPU instructions. If no IOP exists in a given configuration when this instruction is executed by the CPU via the IOCR instruction, it becomes a no operation. All Class III interrupts will be serviced by the IOP.

- 8) Exchange Control Memory (FE a m) (XIM a,y,m) - Store the contents of control memory location specified by the m-field for the I/O channel specified by the a-field in main memory location y. Load that control memory location specified by the m-field for the I/O channel specified by the a-field with the contents of main memory address y + 1.

NOTE:

The m-field = 2 or 6 causes an I/O instruction fault interrupt. CM2 and CM6 are CAP locations in control memory, thus the XIM command instruction will cause the chain program to jump, if it is active.

The capabilities previously listed enable processor software to start, stop, monitor, and modify I/O channel operation.

Communication from I/O channel to processor software programs is performed primarily via the interrupt system in the AN/AYK-14(V) computer. I/O channel interrupts are Class III, lowest priority, interrupts of the three classes within the AN/AYK-14(V) computer. There are four Class III interrupts for each of the 16 possible I/O channels. Refer to Section 9, Interrupts, for a description of each of these interrupts. Recognition of interrupts is based on a priority system using the firmware priority number for each channel as follows:

- Priority 1 ERI error or abnormal
- Priority 2 EII external
- Priority 3 OCI output chain
- Priority 4 ICI input chain

SERIAL INTERFACE MODULE

The serial interface module (SIM) is a serial 1553A I/O channel type in the AN/AYK-14(V) computer. General characteristics of the SIM are:

- Program selectable bus controller (BC) and remote terminal (RT) operating modes
- Dual bus interfaces for redundancy

- Comprehensive SYNC detection, Manchester code error detection, and parity maintenance
- Bus bypassed self-test mode under software control
- Asynchronous operation, half-duplex
- Error insertion under software control for system test purposes

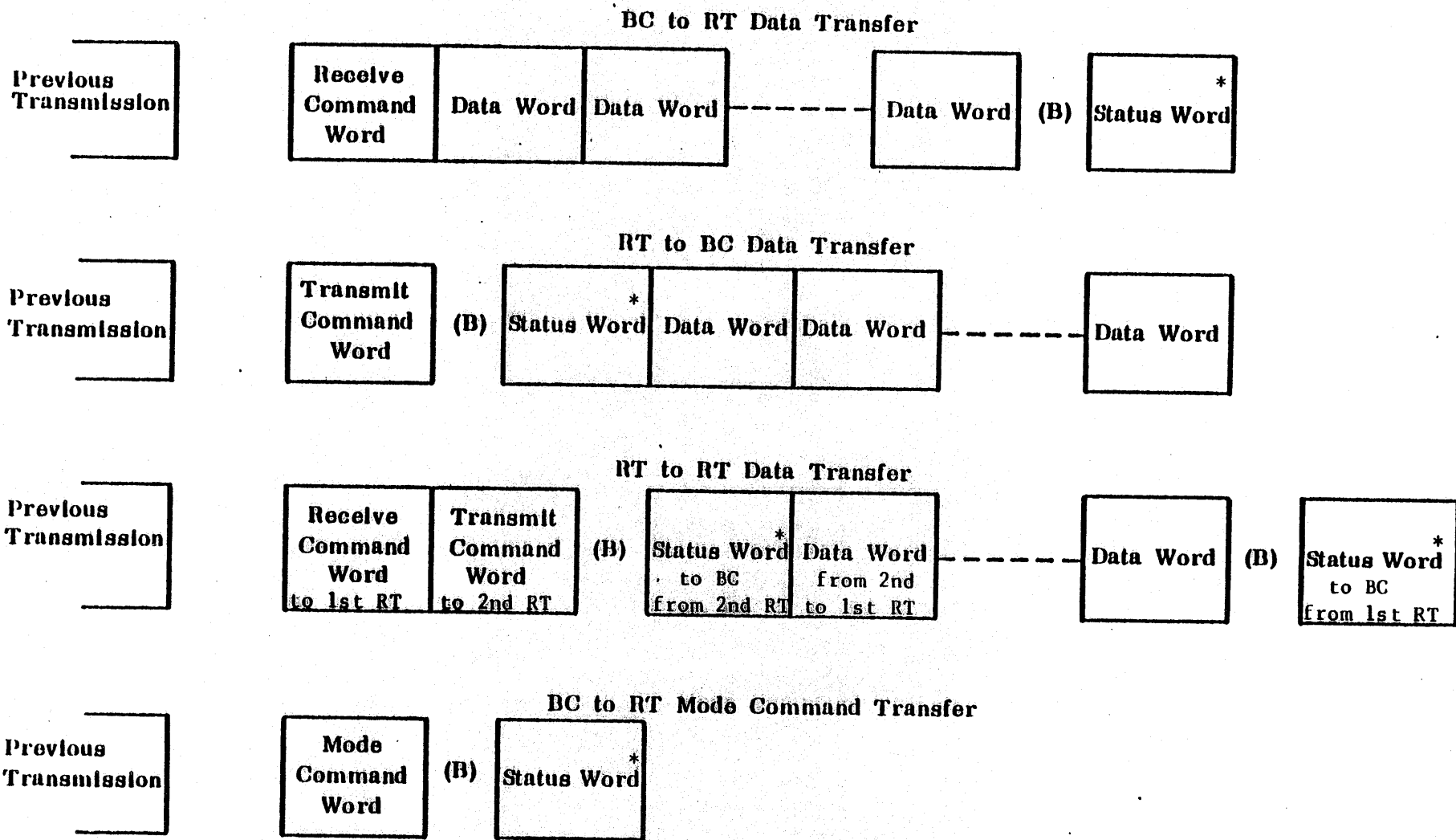
Sole control of information transmission on the bus resides with the operating BC which initiates all transmissions to or from RTs. Each bus can have up to 32 RTs connected to it. The same 32 RTs may be connected to both buses or 32 different RTs can be connected to each bus. All information on the bus is comprised of messages which are formed by appropriate sequences of the following three types of words:

- Command
- Data
- Status

MESSAGE FORMATS

The four message formats which may exist on a bus are shown in Figure 10-3 and are as follows:

- 1) BC to RT data transfer - The BC issues a receive command word followed by a specified number of data words. There are no interword gaps between command and data words. The addressed RT, after message validation and interword gap time, issues a status word response to the BC.
- 2) RT to BC data transfer - The BC issues a transmit command word. The addressed RT, after command word validation and interword gap time, transmits a status word followed by the specified number of data words to the BC. There are no interword gaps between the status word and the data words.
- 3) RT to RT message format sequence - The BC issues a receive command word to one RT and then a transmit command word to the other RT. The RT addressed to transmit, after command word validation and interword gap time, transmits a status word to the BC followed by the specified number of data words to the RT addressed to receive. There are no interword gaps between this status word and data words. The RT addressed to receive, after data validation and interword gap time, transmits a status word to the BC. The BC receives both status words and the data words are transferred directly from RT to RT.
- 4) BC to RT mode command word transfer - The BC issues a mode command word, with a function or mode control code in the data word count field. The addressed RT, after command word validation and interword gap time, transmits a status word response back to the BC.



*Status word shown in Figure 10-4.
 Gap (B) is between 2 and 5 microseconds.

Figure 10-3. SIM I/O Channel Type Message Formats

WORD FORMATS

Figure 10-4 shows the three word formats and their relationship to bus bit times.

NOTE:

All three word formats begin with a 3-bit time SYNC pattern. Command and status words start with a positive SYNC pattern and data words start with a negative SYNC pattern.

There are 16 information bits and an odd parity bit following the 3 SYNC bits. All information fields of all words are transmitted most significant bit first. Bits or field of bits are defined as follows:

- 1) Bits 15 through 11 (bus bit times 4 through 8) of the command and status words define one of 32 possible RT units connected to the selected bus.
- 2) Bit 10 (bus bit time 9) of the command word is the T/R bit (transmit/receive) and, for a nonzero subaddress field, specifies the following:

T/R = 0, Receive command word
T/R = 1, Transmit command word
- 3) Bits 9 through 5 (bus bit times 10 through 14) of the command word define a message subaddress. If this field of bits is 0's, the command word is termed a mode command word.
- 4) Bits 4 through 0 (bus bit times 15 through 19) of the command word define the number of data words to be transferred in the designated message.

NOTE:

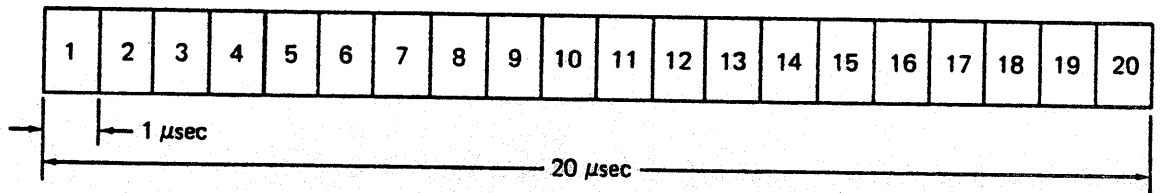
As many as 32 data words may be transferred in a single message. For a mode command word, this field is interpreted as a mode code.

- 5) Bit 10 (bus bit time 9) of the status word, when set, defines a message error prior to status word responses in a message sequence.
- 6) Bits 9 through 1 (bus bit times 10 through 18) of the status word define status codes as determined by system usage and software.
- 7) Bit 0 (bus bit time 19) is user-defined by the system equipment and AN/AYK-14(V) computer software.

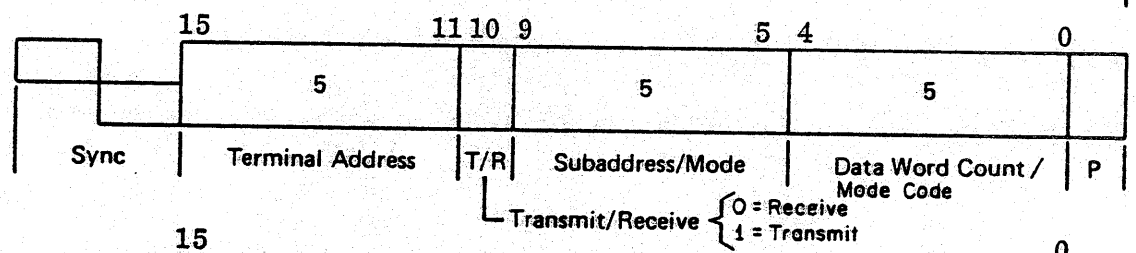
CONTROL MEMORY DEFINITION

Control memory format and usage for the SIM I/O channel type is shown in Figure 10-5. The specific function and format of certain of the control memory locations is different for different modes (BC or RT) of operation.

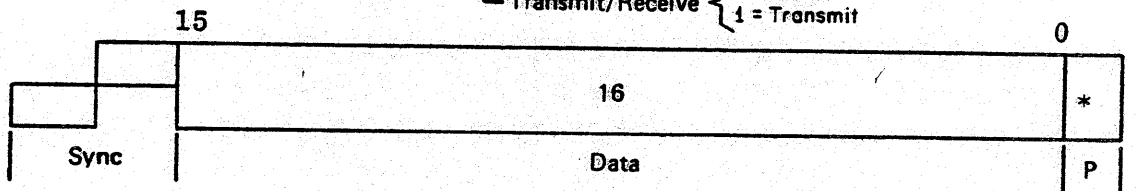
Bit Times:



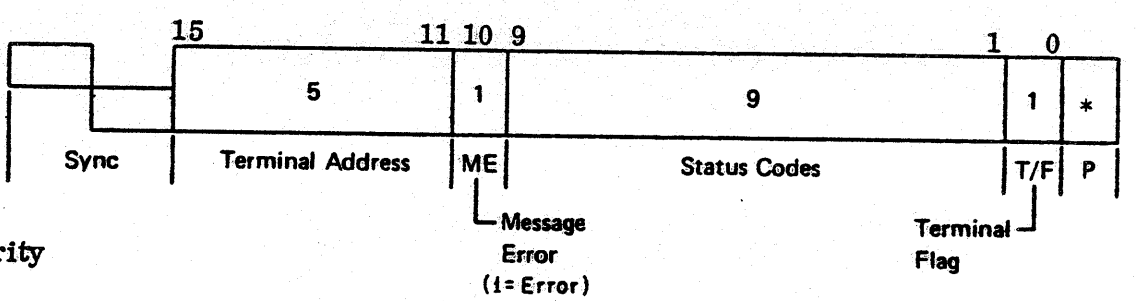
Command Word:



Data Word:



Status Word:



*Odd Parity

Figure 10-4. SIM Channel Type Word Formats

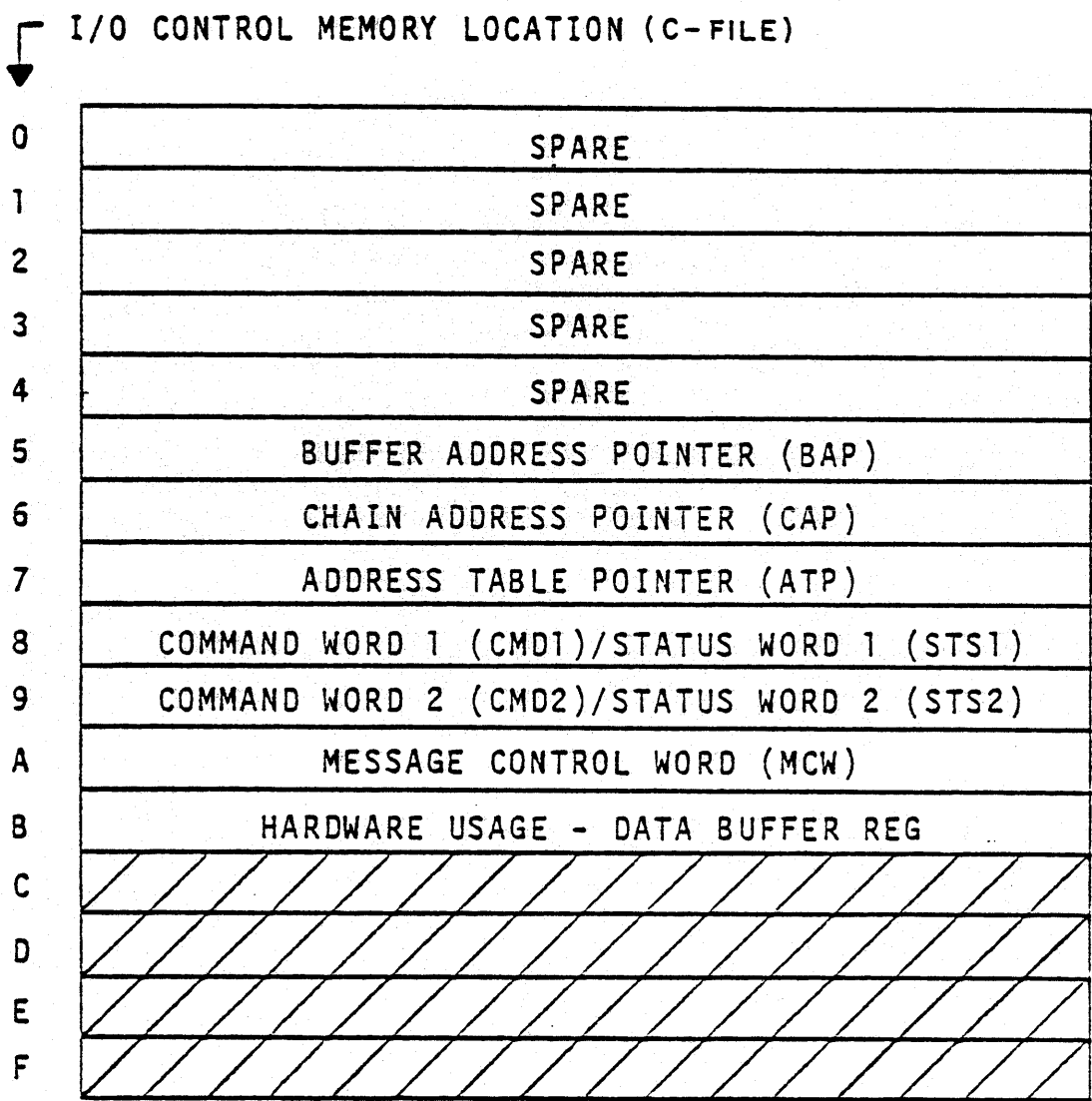


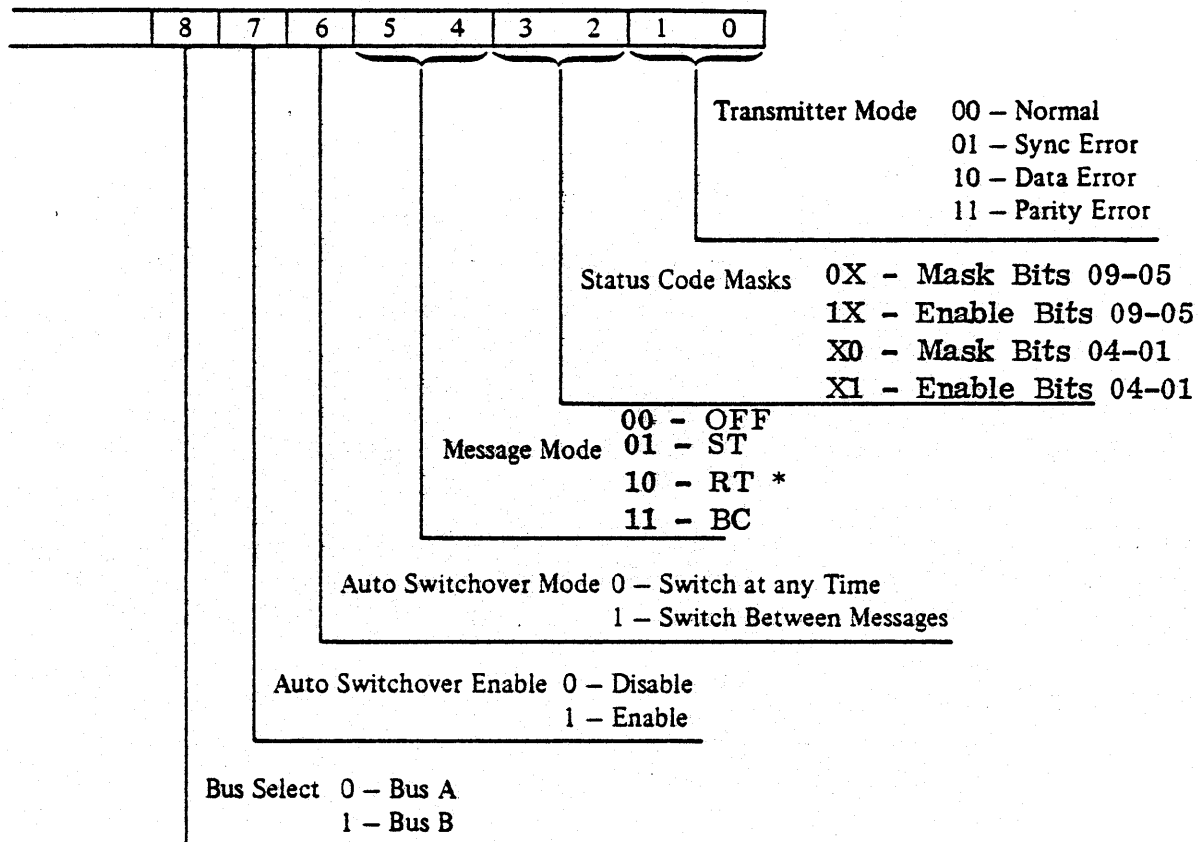
Figure 10-5. SIM Control Memory Map

However, location 5 always contains the address of the main memory location where the next sequential data word is to be input or output during a message sequence. BAP is incremented by 1 for each data word transferred during I/O information transfer activity. Location 6 always contains the address of the main memory location where the next I/O chain instruction is located and is updated during I/O chaining activity. In other words, CAP is the chain program address register. Location A is always the MCW which defines the mode and control of the SIM. The format of the MCW is shown in Figure 10-6 and the bits or fields of bits defined as follows:

- 1) Bit 8 - Bus Select - Selects the prime 1553A bus of the redundant pair. This is the only bus used during BC mode. In the RT mode, bits 7 and 6 determine if this selection is used, or not, at any given instant in time.
- 2) Bit 7 - Auto Switchover Enable - Enables switching of bus usage based on bit 6 in the RT mode. This bit is not used in the BC mode.
- 3) Bit 6 - Auto Switchover Mode - There are two modes of bus usage switchover when bit 7 is set and the SIM is operating in the RT mode. The two modes are as follows:
 - Bit 6 = 0 Switch to the nonprime bus when a valid command word SYNC pattern is detected on the nonprime bus at any time once. This would utilize a redundant bus scheme allowing the BC to switch the RTs from the prime bus to the nonprime bus anytime during a message.
 - Bit 6 = 1 Switch to a nonused bus when a valid command word SYNC pattern is detected on the nonused bus between messages. This allows the BC to address RTs on one bus or the other.
- 4) Bits 5 and 5 - Message Mode - These two bits define the SIM mode of operation.
- 5) Bits 3 and 2 - Status Code Mask - Enables or disables hardware monitoring of bits in the status code field of the status word received used in the BC mode only. These bits are used in conjunction with generation of an EII interrupt.
- 6) Bits 1 and 0 - Transmitter Mode - Enables normal or error insertion operation for system or bus network test purposes and can be used regardless of mode or message state within a message sequence. Errors may be inserted in a word or all words of a message to ensure proper error detection.

Locations 8 and 9 contain command words transferred when in BC mode and status words transferred when in RT mode.

Location 7 is used in RT mode only and contains a memory address which is modified by the subaddress field of the received command word. The resulting address contents are then loaded into control memory location 5.



*Bus Monitor (BM)
is a submode of
RT.

Figure 10-6. SIM MCW Format

SIM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on a SIM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (E0 a m) (ACR m; CCR a, m) - The same as listed under Processor to I/O Channel Communication paragraph with the following exceptions:

a-field is not used for the CCR instruction

m-field equal to 1100 - no operation } Cannot enable/disable
external interrupts

m-field equal to 1101 - no operation }

- 2) Initiate Message (E2 a m) IM a,y,m) - Load control memory location specified by the m-field with the value y then perform the operation specified by the a-field as follows:

<u>a</u>	<u>Mode</u>	<u>Operation</u>
00XX	BC	Initiate message using CMD1 and BAP in control memory locations 8 and 5, respectively.
00XX	RT	Initiate message response on the bus using STS1, STS2, and ATP in control memory locations 8, 9, and 7, respectively.
01XX	BC	Initiate RT to RT message sequence using CMD1 and CMD2 in control memory locations 8 and 9, respectively.
01XX	RT	Initiate monitor of all messages on the bus using BAP in control memory location 5.
1XXX		No operation

NOTE:

When the initiate message instruction with an a-field of 0XXX is encountered, the I/O chaining activity on the associated I/O channel is halted and I/O information transfer activity is initiated.

- 3) Initiate Transfer (E3 a 0) (IO a,y) - Load control memory locations 0 and 1 (a-field equal to XX00) or 4 and 5 (a-field not equal to XX00) with the contents of main memory addresses y and y + 1, respectively. Perform y + 1 the operation specified by the a-field as defined in initiate message. The associated channel is the one executing the I/O chain program.

NOTE:

When the initiate transfer instruction with an a-field 0XXX is encountered, the I/O chaining activity on the associated I/O channel is halted and I/O information transfer activity is initiated.

- 4) Load Control Memory (E6 0 m) (LCMK, m,y) - Load the control memory location specified by the m-field with the value y. The a-field is not used and the associated I/O channel is the one executing the I/O chain program.
- 5) Load Control Memory (E7 0 m) (LCM m,y) - Load the control memory location specified by the m-field with the contents of main memory address y. The a-field is not used and the associated I/O channel is the one executing the I/O chain program.
- 6) Store Control Memory (EB 0 m) (SCM m,y) - Store the contents of control memory location specified by the m-field at main memory address y. The a-field is not used and the associated I/O channel is the one executing the I/O chain program.
- 7) Halt/Interrupt (EC a 0) (HCR; IPR) - Perform the operation specified by the field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Halt I/O chaining activity
XXX1	Generate Class III, priority 3, OCI interrupt

- 8) Set/Clear Flag (EF a 0) (ZF y; SF y) - Set or clear the two most significant bits (flag) of the main memory value at address y as specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Clear flag
XXX1	Set flag

The m-field is not used and the associated I/O channel is the one executing the I/O chain program.

- 9) Conditional Jump (F2 oo) (SJMC 0,y) - This is an unconditional jump instruction for the SIM I/O channel type. Load control memory location 6 (CAP) with the value y.
- 10) Store Status (FB a m) (CSST y,m) - Store the channel status as specified by the m-field in main memory y as follows:

<u>m</u>	<u>Mode</u>	<u>Status Word Value</u>
1X1X	BC	First status word received during last message sequence on the bus.

<u>m</u>	<u>Mode</u>	<u>Status Word Value</u>
	RT	Not assigned
0X1X	BC	Second status word received during last RT-to-RT message sequence on the bus.
	RT	Not assigned
1X0X	-	Hardware status word 0 (Figure 10-7)
0X0X	-	Hardware status word 1 (Figure 10-8)

The a-field is not used and the associated I/O channel is the one executing the I/O chain program.

- 11) Bit Jump (FD 0m) BJ m,y - This a conditional jump instruction. If the bit in control memory location 3 specified by the m-field is a logic 1, then the value y is loaded into control memory location 6. If the bit is a logic 0, the next I/O chain instruction in sequence is executed.
- 12) Exchange Control Memory (FE 0m) (XCM m,y) - Store the contents of control memory location specified by the m-field in main memory location y. Then load the control memory location specified by the m-field with the contents of main memory location y + 1.

NOTE:

This becomes a branch or jump instruction if the m-field equals 6. The old contents of control memory location 6 is saved so that a return is possible when the entered routine is exited.

SIM INTERRUPT HANDLING

The SIM I/O channel type is capable of generating the EII, ERI, and OCI Class III interrupts. The SIM hardware shall generate an ERI interrupt for either of the following when in BC mode only.

- 1) Expiration of the monitoring times during a message sequence, such as B gap too long or total message length too long, causes the SIM to halt the message sequence, generate an ERI interrupt, and not restart the chaining activity.
- 2) Detection of any message abnormalities, such as improper sync character, wrong parity, or wrong word length, during a message sequence causes the same as stated in 1) above.

If the ERI is locked out at the processor by SR1, bit 11 of hardware status word 0 is set and stays set until the interrupt is processed.

The SIM hardware shall generate an EII interrupt in BC mode during a message sequence provided Class III, priorities 2, 3, and 4 are enabled for the

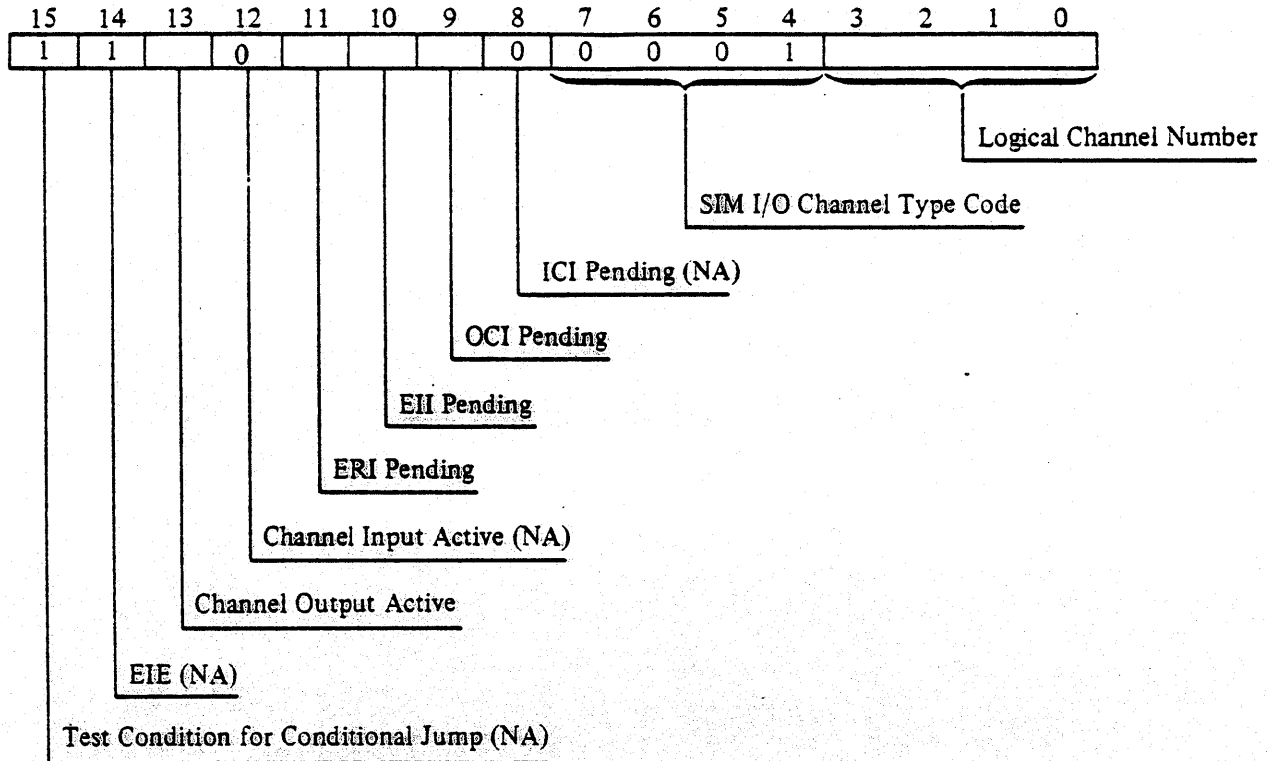


Figure 10-7. SIM Hardware Status Word 0 Format

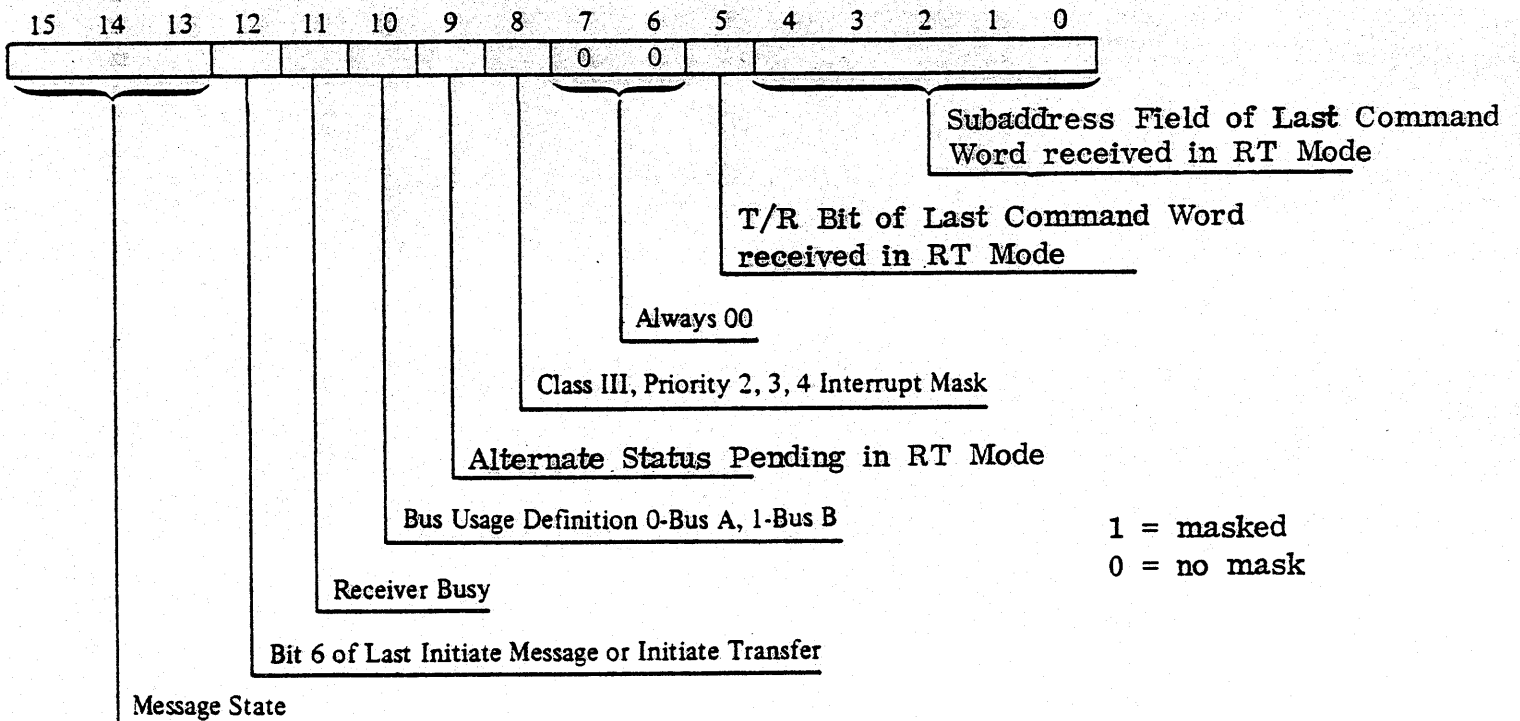


Figure 10-8. SIM Hardware Status Word 1 Format

channel when a status word is received over the bus which has an ME bit, unmasked status code bit, or TF bit set.

The interrupt word stored in main memory (80₁₆ plus channel) is the first status word received during the last message sequence on the bus. Channel operation is halted at the end of the message sequence and chaining activity is not restarted. If RT-RT is specified, the first RT status word is stored into 80₁₆ plus channel while the second RT's status word must be accessed via the SST/CSST instruction.

If not enabled, the status word will be stored in memory, the interrupt will be held pending at the channel level, and bit 10 of hardware status word 0 is set. When the enable condition is enabled, the interrupt will be passed on to the processor and, if not locked out by SRI, will be processed clearing bit 10 of hardware status word 0. If locked out by SRI, the interrupt will stay pending until processed.

The SIM I/O channel type shall generate an OCI interrupt only under I/O chain program control via the interrupt processor instruction (operation code EC).

No Class III ICI interrupts are generated by the SIM I/O channel type, as all chain programs are considered to be output chain programs.

SIM PROGRAMMING CONSIDERATIONS IN BC MODE

For a BC to RT message sequence, the programmer, via either command instructions or chain instructions, must load CMA with MCW, CM8 with CMD1 (the receive command word to be sent to the RT), and CM5 with BAP. In the chain program, start the message sequence with an initiate message instruction (IM a,y,m) with a-field equal to 00XX. This stops the chaining activity and starts the SIM hardware and processor firmware performing the following actual sequence of operations.

- 1) The receive command word (CMD1) in CM8 is transmitted.
- 2) The data words starting at BAP are transmitted.
- 3) The status word from the RT is received and examined.
- 4) If in the status word the ME bit, TF bit, and unmasked status bits are all zero, chaining activity is reinitiated.
- 5) If in the status word the ME bit, TF bit, or unmasked status bits are set, the status word is stored in memory, and the EII interrupt is generated based on the conditions of the enables. The chain activity is not reinitiated.
- 6) If during the message sequence an error is encountered, the ERI interrupt is generated, the sequence operation stops, and chaining is not reinitiated.

For a RT to BC message sequence the programmer must load CMA with MCW, CM8 with CMD1 (the transmit command word to be sent to the RT), and CM5 with

BAP. In the chain program, start the message sequence with an initiate message instruction (IM a,y,m) with a-field equal to 00XX. This stops the chaining activity and starts the SIM hardware and processor firmware performing the following actual sequence of operations.

- 1) The transmit command word (CMD1) in CM8 is transmitted.
- 2) The status word from the RT is received and examined.
- 3) If in the status word a bit is set, it is stored in memory, and the EII interrupt is generated based on the conditions of the enables.
- 4) The data words are received, examined, and stored in memory using BAP.
- 5) When all data words are received, the sequence operation stops, and chaining activity is reinitiated if the EII interrupt was not generated by a bad status word.
- 6) If during the message sequence the wrong number of words was received or a data word with errors was received, the sequence operation is stopped at that point, the ERI interrupt is generated, and chaining activity is not reinitiated. If the sequence terminates because of an error in a data word, that data word will be the last word stored in memory.

For an RT to RT message sequence, the programmer must load CMA with the MCW, CM8 with CMD1 (the receive command word to be sent to the first RT), and CM9 with CMD2 (the transmit command word to be sent to the second RT). In the chain program, start the message sequence with an initiate message instruction (IM a,y,m) with a-field equal to 01XX. This stops the chaining activity and starts the SIM hardware and processor firmware performing the following actual sequence of operations.

- 1) The receive command word in CM8 (CMD1) is transmitted to RT1.
- 2) The transmit command word in CM9 (CMD2) is transmitted to RT2.
- 3) The status word from RT2 is received and examined.
- 4) If in the status word a bit is set, it is stored in memory, and the EII interrupt is generated based on the conditions of the enables.
- 5) The status word from RT1 is received and examined after the data has been transmitted between the RTs.
- 6) If in the status word a bit is set, it is stored in memory, and the EII interrupt is generated based on the conditions of the enables.
- 7) Sequence of operations stop, and the chaining activity is not reinitiated if the EII interrupt was generated from either or both status words.

- 8) If during the message sequence an error occurs (status word not received, improper gap B times, or errors in the status words), the sequence of operation stops at that point, the ERI interrupt is generated, and chain activity is not reinitiated.

For a mode command word message sequence the programmer must load CMA with the MCW and CM8 with CMD1 (the mode command word to be sent to the RT). In the chain program, start the message sequence with an initiate message instruction (IM a,y,m) with a-field equal to 00XX. This stops chaining activity and starts the SIM hardware and processor firmware performing the following actual sequence of operations.

- 1) The mode command word in CM8 (CMD1) is transmitted to the RT.
- 2) The status word from the RT is received and examined.
- 3) If in the status word a bit is set, it is stored in memory, and the EII interrupt is generated based on the conditions of the enables.
- 4) Sequence of operations stop, and the chaining activity is reinitiated if the EII interrupt was not generated.
- 5) If during the message sequence an error occurs the sequence of operation stops at that point, the ERI interrupt is generated, and the chaining activity is not reinitiated.

SIM PROGRAMMING CONSIDERATIONS IN RT MODE

In RT mode, the programmer sets up the SIM and allows the BC to determine the type of message sequence and initiate it. The programmer must load CMA with the MCW, CM8 with STS1 (the status word to be sent to the BC), and the CM7 with the ATP before the SIM can be enabled to respond to the BC. In the chain program, the SIM is enabled by executing an initiate message instruction (IM a,y,m) with a-field equal to 00XX. This instruction stops chaining activity and starts the SIM hardware and the processor firmware to perform the following actual sequence operations.

The SIM monitors traffic on the bus, performing switchover if enabled and occurs, comparing the command word RT addresses against the RT addresses in CM8. The SIM continues to monitor the bus until the channel is master cleared, a new MCW is placed in CMA, or a RT address match occurs. When there is a match, bits 5 through 10 of the command word are examined to determine the type of message sequence to perform.

Transmit Command Word Message

If the command word is a transmit command word, bits 5 through 10 of the word are right justified and zero filled to a 16-bit word and then added to ATP from CM7. The contents of memory at the resulting address is loaded into CM5 (BAP). The SIM then transmits the status word from CM8 to the BC followed by the data words from memory using BAP. The SIM hardware then returns to monitoring the bus for another command word.

Mode Command Word Message

If the received command word is a mode command word (bits 5 through 9 are zero), it is stored in main memory at the EII location for the channel and the SIM BC generates the EII interrupt based on the condition of the enables. The SIM then transmits the status word from CM8 to the BC, reinitiates the chaining activity, and returns to monitoring the bus for another command word.

Receive Command Word Message

If the command word is a receive command word, CM5 (BAP) is loaded in the same manner as for a transmit command word. The data words are received, examined for errors, and placed in memory using BAP. When all data words are received and stored, the status word from CM8 is transmitted to the BC, and the SIM begins monitoring the bus for another command word. If during the sequence a data word with an error is received, that data word is the last word stored in memory. The SIM continues to monitor the bus and upon completion of the data transfer, transmits the status word from CM8 (with the ME bit set) and returns to monitoring the bus.

DISCRETE INTERFACE MODULE

The discrete interface module (DIM) has the following general characteristics:

- Eight external interrupts with individual mask bits and program selectable priority
- 32 bidirectional discretets (DIO) program selectable in groups of four as input or output discretets
- 16 input discretets (DID)
- 16 switch closure input discretets (DIS)
- Parallel software interface where individual input and output discretets are not set or cleared via software by individual bit addressing but rather via 16-bit word addressing.

WORD FORMATS

There are four input words (word 0 through word 3) and two output words (word 0 and word 1) in a DIM I/O type interface and one 8-bit interrupt word. Figure 10-9 shows these words for input or output applications.

There are bidirectional discretets (DIO) 31 through DIO 00, input discretets (DID) 15 through DID 00, and switch closure input discretets (DIS) 15 through DIS 00. These 64 signals are grouped into four 16-bit words as shown in Figure 10-9.

There are two 16-bit output words that share the 32 (DIO 31 through DIO 00) signals with input words 0 and 1. These output discretets may be enabled in groups of 4 bits to drive the related DIO signals or disabled to allow an external driver to feed them. Regardless of whether a given group of four DIO signals are programmed as outputs, they may also be read as inputs.

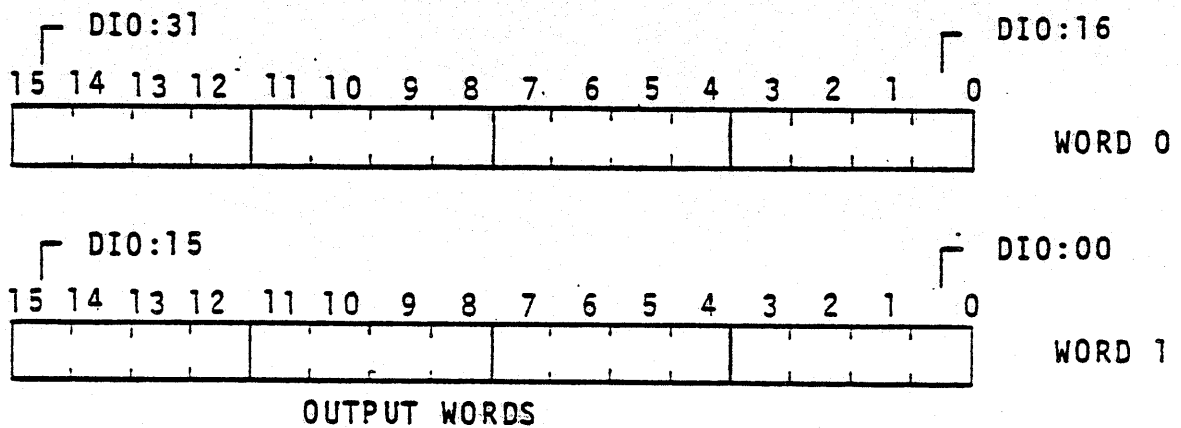
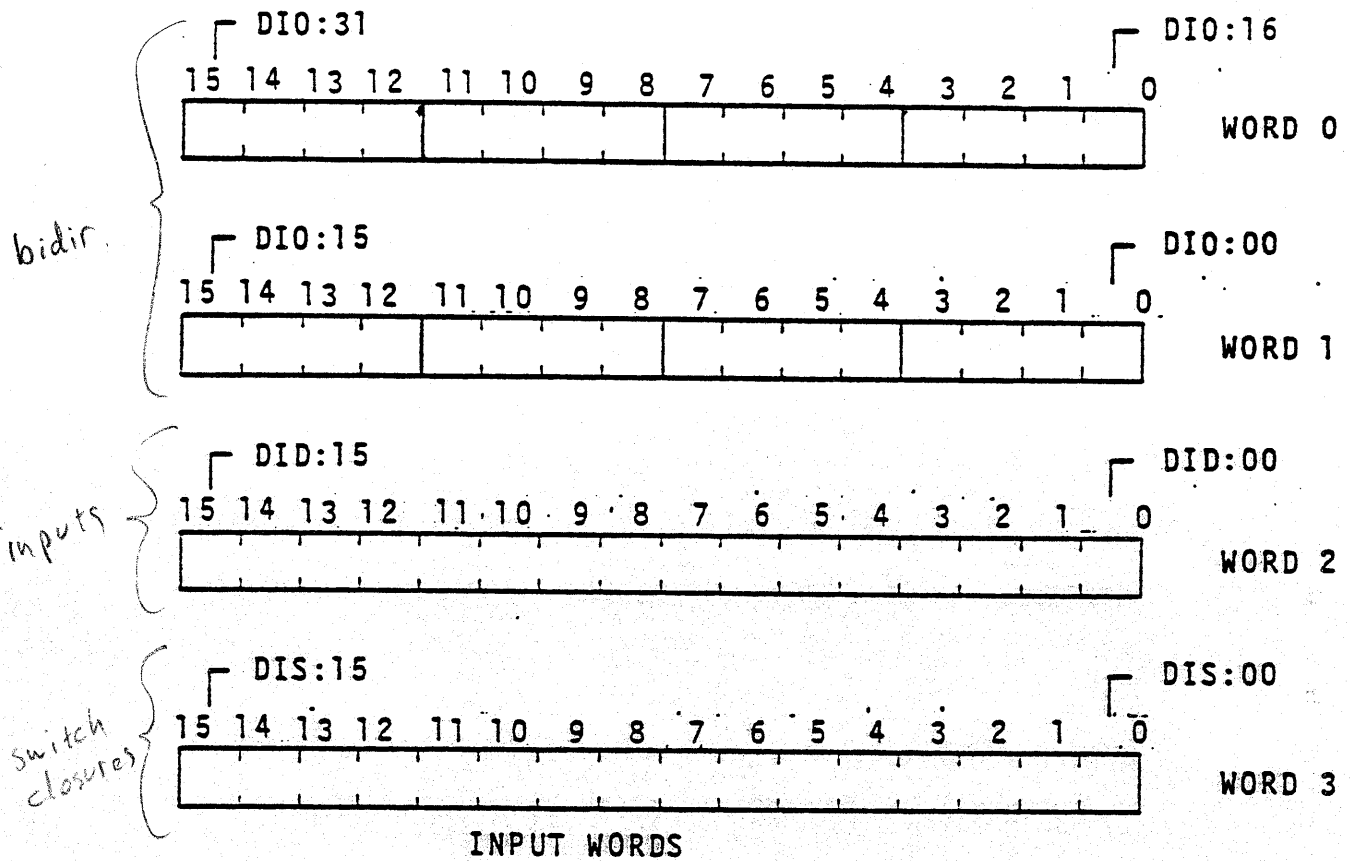


Figure 10-9. DIM Word Formats

CONTROL MEMORY DEFINITION

Control memory format and usage for the DIM I/O channel type is shown in Figure 10-10. The specific function and format of certain of the control memory locations are associated with only input or output activities. Locations 0, 1, and 2 are associated with input activity, and locations 4, 5, and 6 are associated with output activity.

Control memory locations 0 and 4 are the buffer control words (BCW) (Figure 10-11) and define the word count for input and output information transfer activity.

Bits 15 and 14 are termed transfer mode (TM) and are either 00 (abort for input, word for output) or 10 (word for input or output). No other combination will work on a DIM channel.

Bits 13 and 12 are not used.

Bits 11 through 0 are the buffer transfer count with a BTC of zero causing 4096 word transfers. Input transfers consist of only four possible words and output transfers consist of only two possible words. Counts greater than this will repeatedly handle the same words until the BTC equals zero. Control memory locations 8 and 9 are termed the mask/priority words (MPWs) 0 and 1 and define the priorities and mask bits for the interrupt signals.

Control memory location A is termed the discrete select word (DSW) and, as shown in Figure 10-12, uses only the lower eight bits.

The DSW selects which bidirectional discretetes (DIO) shall be utilized for input (logic 0) or output (logic 1). This provides for selecting up to 32 input or output discretetes, in groups of four, under program control.

DIM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on a DIM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (EO a m) (ACR m; CRR a,m) - The same as listed under Processor to I/O channel Communication paragraph except that the a-field is not used in the CCR instructions.
- 2) Initiate Message (E2 a m) (IM a,y,m) - Load control memory location specified by the m-field with the value y then perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
00X0	Initiate input data transfer sequence starting with input word 0 using the BCW and BAP in control memory locations 0 and 1.
01X0	Initiate input data transfer sequence starting with input word 1 using the BCW and BAP in control memory locations 0 and 1.

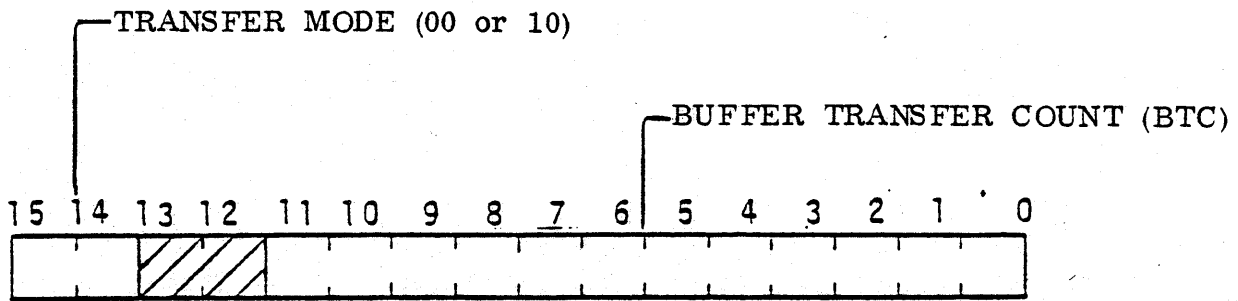


Figure 10-11. DIM BCW Format

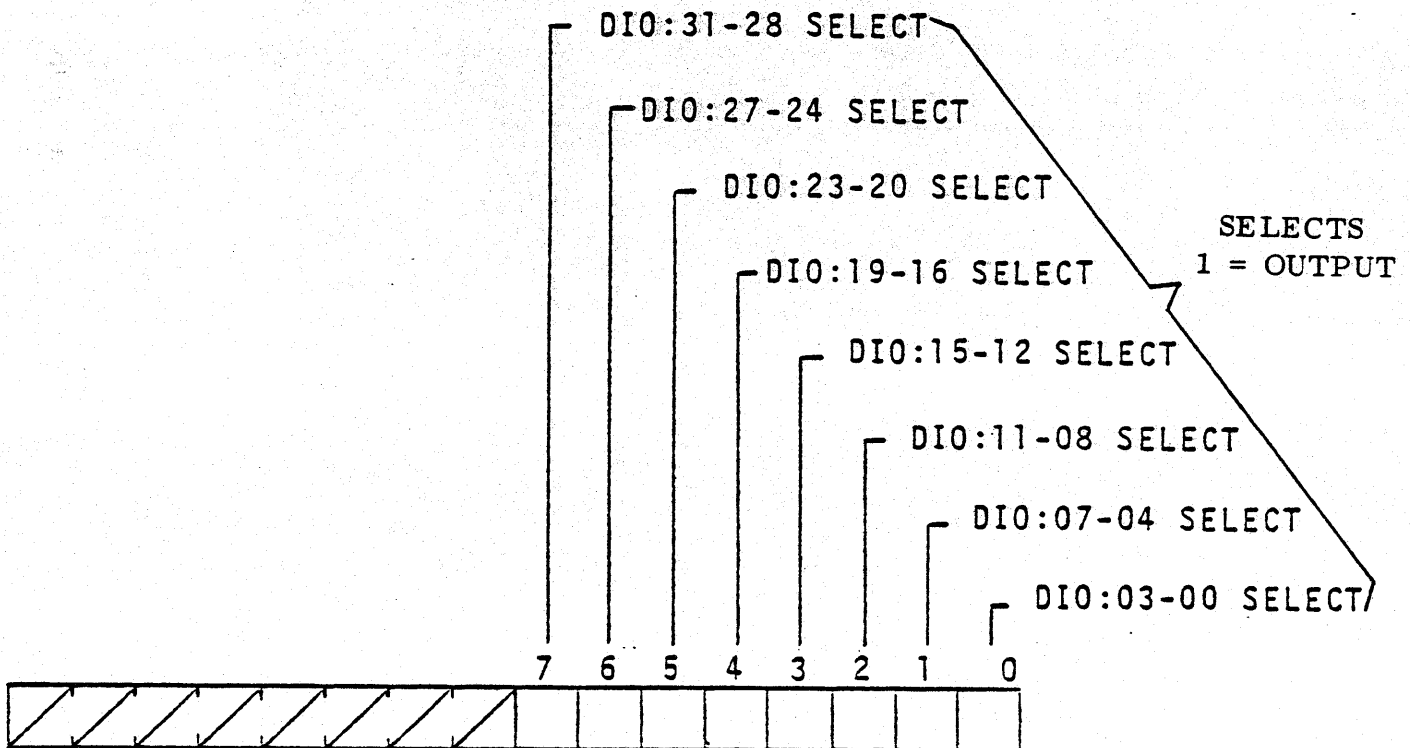


Figure 10-12. DIM DSW Format

<u>a</u>	<u>Operation</u>
10X0	Initiate input data transfer sequence starting with input word 2 using the BCW and BAP in control memory locations 0 and 1.
11X0	Initiate input data transfer sequence starting with input word 3 using the BCW and BAP in control memory locations 0 and 1.
XX01	Initiate output data transfer sequence starting with output word 0 using the BCW and BAP in control memory locations 4 and 5.
XX11	Initiate output data transfer sequence starting with output word 1 using the BCW and BAP in control memory locations 4 and 5.

3) Initiate Transfer (E3 a 0) (IO a,y) - Load control memory location 0 and 1 (a-field equal to XX00) or 4 and 5 (a-field not equal to XX00) with the contents of main memory addresses y and y + 1 respectively. Perform the operation specified by the a field as defined in 2) above.

4) Load Control Memory (E6 0 m) (LCMK m,y) - The same as listed under SIM I/O Channel Instructions paragraph.

5) Load Control Memory (E7 0 m) (LCM m,y) - The same as listed under SIM I/O Channel Instructions paragraph.

6) Store Control Memory (EB 0 m) (SCM m,y) - The same as listed under SIM I/O Channel Instructions paragraph.

7) Halt/Interrupt (EC a 0) (HCR; IPR) - Perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Halt I/O chaining activity.
XXX1	Generate Class III, priority 3, OCI interrupt if output chain or Class III, priority 4, ICI interrupt if input chain.

The m-field is not used and the associated I/O channel is the one executing the chain program.

8) Set/Clear Flag (EF a 0) (ZF y) - The same as listed under SIM I/O Channel Instructions paragraph.

9) Conditional Jump (F2 00) (SJMC 0, y) - This is an unconditional jump instruction for the DIM I/O channel type. Load control memory location 2 if input chain or control memory location 6 if output chain with the value y.

- 10) Serial Interface Control (F8 0 m) (CSIR m) - Set the interrupt active discrete or clear all eight interrupt active discrettes as specified by the m-field as follows:

<u>m</u>	<u>Operation</u>
0XXX	Clear all eight interrupt active discrettes
1000	Set interrupt 0 (INT0) active discrete
1001	Set interrupt 1 (INT1) active discrete
1010	Set interrupt 2 (INT2) active discrete
1011	Set interrupt 3 (INT3) active discrete
1100	Set interrupt 4 (INT4) active discrete
1101	Set interrupt 5 (INT5) active discrete
1110	Set interrupt 6 (INT6) active discrete
1111	Set interrupt 7 (INT7) active discrete

The a-field is not used and the associated I/O channel is the one executing the I/O channel program.

- 11) Store Status (FB 0 m) (CSST y,m) - Store the channel status as specified by the m-field in main memory location y as follows:

<u>m</u>	<u>Status Word Value</u>
X000	Channel hardware status word 0 (Figure 10-13)
X001	} Not assigned
X010	
X011	
X100	Input word 0
X101	Input word 1
X110	Input word 2
X111	Input word 3

} (Figure 10-9)

The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.

- 12) Bit Jump (FD 0 m) (BJ m,y) - This is a conditional jump instruction. If the bit in control memory location 3 specified by the m-field is a logic 1, then the value y is loaded into control memory location 2 if input chain or control memory location 6 if output chain. If the bit is a logic 0, the next I/O chain instruction in sequence is executed.
- 13) Exchange Control Memory (FE 0 m) (XCM m,y) - Store the contents of control memory location specified by the m-field in main memory location y. Then load the control memory location specified by the m field with the contents of main memory location y + 1.

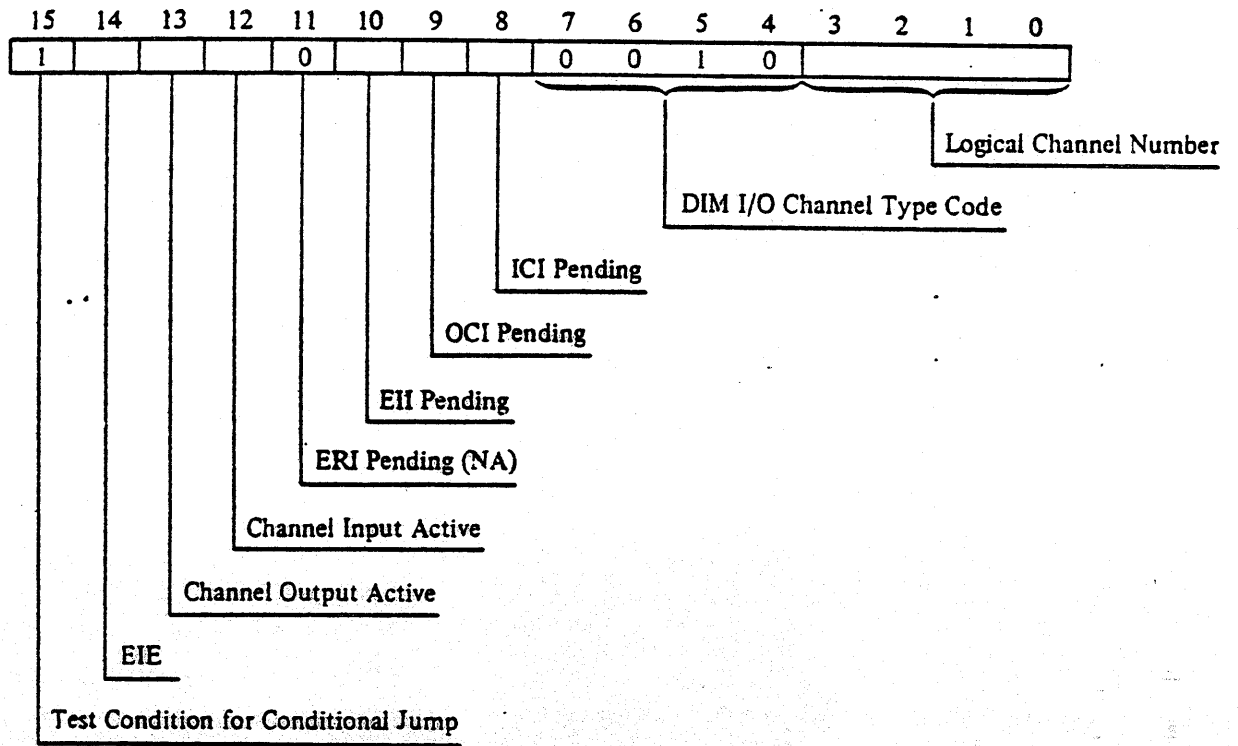


Figure 10-13. DIM I/O Channel Hardware Status Word 0 Format

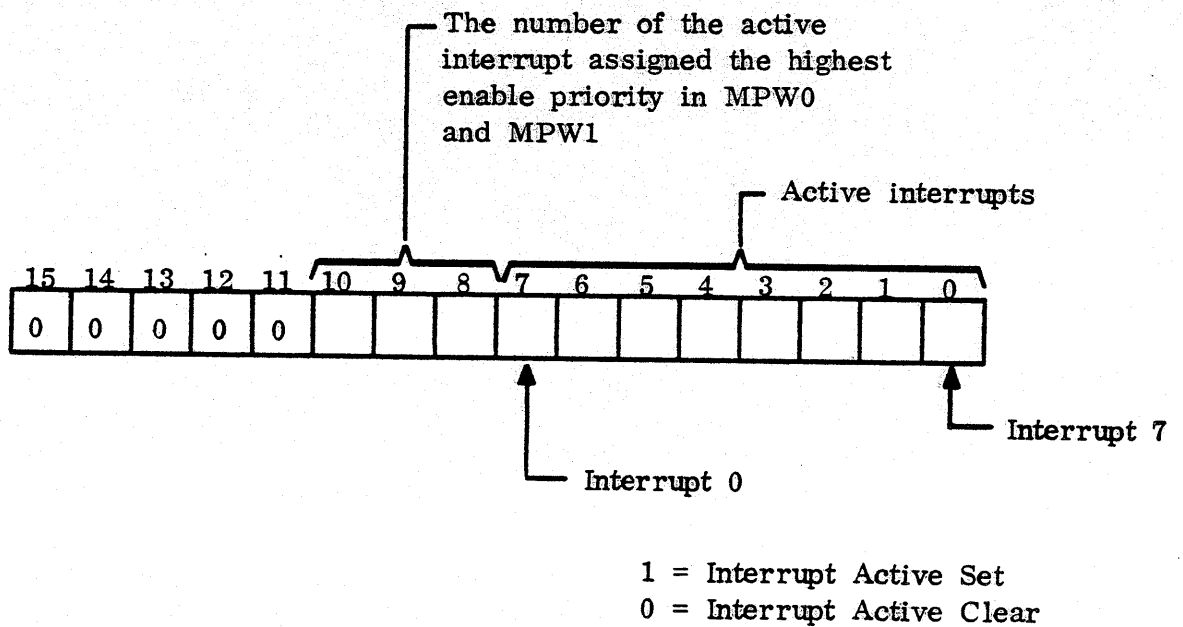


Figure 10-14. DIM Hardware Status Word 1 Format

NOTE:

This becomes a branch or jump instruction if the m-field equals 2 or 6. The old contents of control memory location 2 or 6 is saved so that a return is possible when the entered routine is exited.

MESSAGE FORMATS

Input and output discrettes are transferred between the DIM I/O channel type and main memory as 16-bit words instead of discrete individually addressed bits. These transfers are initiated by an initiate message (operation code E2 hexadecimal) or initiate transfer (operation code E3 hexadecimal) instruction. The initiate message and initiate transfer instructions can be executed at either an input or output chain program. Both chain programs can be active at the same time. An output transfer can be initiated by either instruction in an input chain, and an input transfer can be initiated by either instruction in an output chain. If during the execution of an input chain program an output transfer is initiated, the chain activity stops, the output transfer takes place, and upon completion the input chain activity is resumed. The same is true for input transfers started during output chain programs.

- 1) Input Data Transfers - The four input discrete words may be read into main memory, as previously stated. The transfer of these words is controlled by the BCW and BAP in control memory locations 0 and 1. An initial BTC = 0 specified 4096 words, BTC = 1 specifies 1 word, etc. During input data transfer, the BAP value is incremented by 1 and the BTC is decremented by 1 for each discrete word input and the chaining activity is reinitiated when BTC decrements to 0.

NOTE:

If TM = 0, words are not transferred. Initial BTC values greater than 4 repeat sets of the four possible input words that are defined until BTC = 0.

In addition to this buffering scheme, individual words may be read as status via the store status (operation code FB hexadecimal) instruction.

DIM INTERRUPT HANDLING

The DIM I/O channel type is capable of generating the EII, OCI, and ICI Class III interrupts.

The DIM I/O channel type shall generate the OCI and ICI interrupts only under I/O chain program control via the interrupt processor (IPR) chain instruction. Execution of the IPR instruction in an output chain program results in the OCI interrupt. Conversely, if the IPR instruction was executed in an input chain program, it results in the ICI interrupt.

SR1:1 must be set, Class III priorities 2, 3, and 4 must be enabled (CCR instruction), EIE must be enabled (CSIR instruction), and the MPWs must be set up before external interrupts can be recognized.

I/O chaining has priority over external interrupts.

The DIM hardware generates the EII interrupt in response to the eight external DIM interrupts. The eight interrupts are masked and assigned priority by the two control memory locations 8 and 9 labeled mask priority word zero (MPW0) and one (MPW1).

These two words, MPW0 and MPW1 (Figure 10-15), contain eight 4-bit groups each of which corresponds to an interrupt priority level. Bits 15 through 12 of MPW0 are associated with the highest priority (0) and bits 3 through 0 of MPW1 are associated with the lowest priority (7). The most significant bit (bits 3, 7, 11, and 15) is the mask bit and the other 3 bits define the number of the interrupt signal (INT0 through INT7) for each 4-bit group. Any one of the eight priority levels can be assigned any one of the eight interrupt signals; therefore, it is possible to have the same interrupt signal in more than one interrupt level.

The DIM hardware shall generate an EII interrupt if the following conditions exist.

- 1) The EIE must be set in the DIM channel (ACR/CCR 4 or CCR a 12).
- 2) The external interrupt that becomes active must be assigned a priority level by MPW0 or MPW1 in control memory.
- 3) The assigned priority for the active interrupt must have its mask bit set.

If these conditions are met, hardware status word one is stored in memory at the EII location for the DIM channel. Status word 1 can also be read and stored into memory using the store status instruction (CSST y,m). The DIM hardware status word (Figure 10-14) indicates the highest priority interrupt active in bits 10 through 8 and all active interrupts (enabled or not enabled) in bits 7 through 0.

- 4) If the Class III priorities 2, 3, and 4 are enabled on the DIM channel, the EII interrupt is generated and transmitted to the processor. If the Class III priorities 2, 3, and 4 are not enabled, the EII interrupt pending bit is set in hardware status word 0, and the EII interrupt is held pending at the channel level.

If during the time that an EII interrupt is pending at the channel level as locked out at SRI another interrupt of a higher priority becomes active with its mask bit set, the DIM hardware will cause another hardware status word 1 to be stored in the EII interrupt location. The status word will contain the number of the new higher priority interrupt in bits 10 through 8. During this time, an interrupt of lower priority or one with its mask bit cleared will not cause a new status word 1 to be stored in memory.

While processing the EII interrupt, the firmware clears the EIE in the DIM and clears bits 10 through 8 along with the associated interrupt active bit in status word 1. This prevents the status word 1 in memory from being altered and another EII interrupt from being generated until enabled by the software.

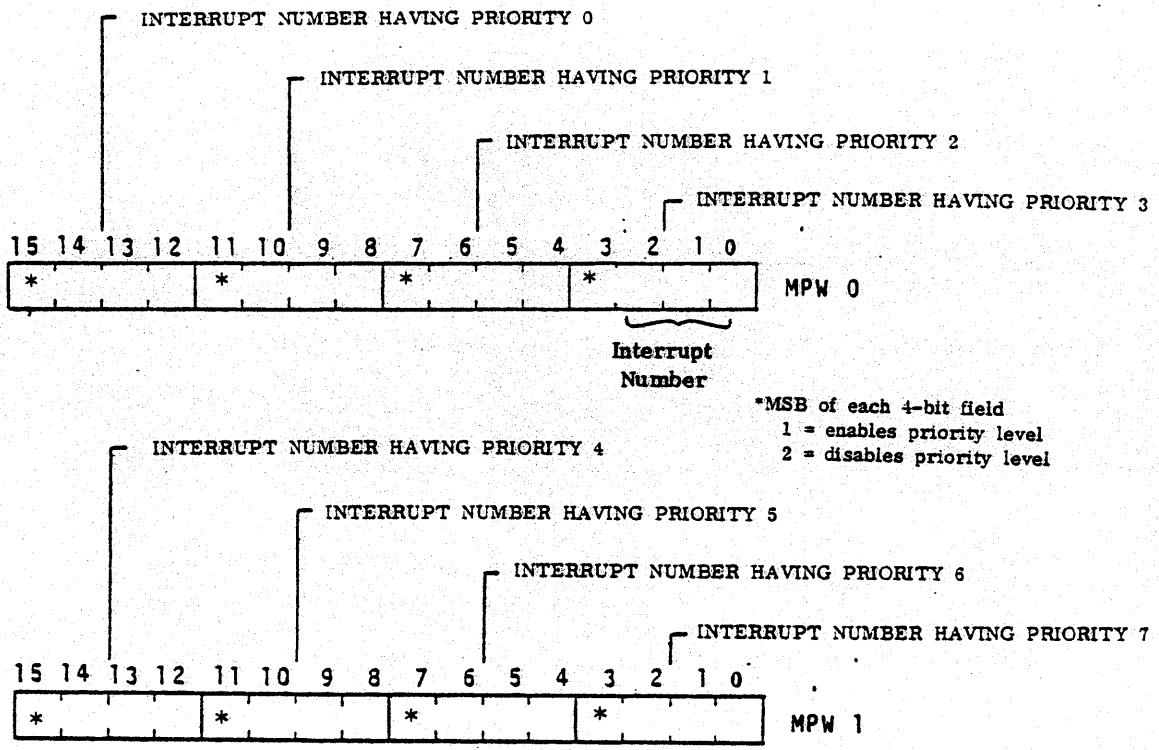


Figure 10-15. DIM MPW Formats

NTDS INTERFACE MODULES

The four NTDS interface modules (NIM) are defined as follows:

- NIM - type A (NTDS slow - 16-bit parallel)
- NIM - type B (NTDS fast - 16-bit parallel)
- NIM - type C (NTDS ANEW - 16-bit parallel)
- NIM - serial (NTDS serial)

NIM types A, B, and C have the following general characteristics.

- Accommodate one 16-bit input channel and one 16-bit output channel to allow full duplex operation.
- Configure to a 32-bit channel using two modules of the same type.

NIM type serial has the following general characteristic.

- Accommodates one input and one output (16-bit or 32-bit) channel to allow full duplex operation in 16-bit or 32-bit mode.

The NIM I/O channel types interface with the general processor module (GPM) via the IOBUS and the EVENTBUS. The IOBUS is for data and instruction communications, and the EVENTBUS is for interrupts and control.

The NIM modes of operation selectable under software are:

- Computer to computer (16 or 32 bit)
- Computer to peripheral (16 or 32 bit)
- Computer to peripheral, externally specified addressing in 32 bit
- Computer to peripheral, dual channel 32 bit

Data on the NIM channels can be either a 8-bit byte, 16-bit word, or 32-bit double word based on the type of channel and selectable via software. The data, as well as the external function words, command words, and external interrupt words have no parity bits attached. Interrupt, function, and command words are 16 bits in byte and word mode and are 32 bits in the 32-bit mode.

CONTROL MEMORY DEFINITION

Control memory format and usage for the NIM I/O channel types is shown in Figure 10-16. Control memory locations 0, 1, and 2 are associated with input chaining and data transfer activity. Control memory locations 4, 5, and 6 are associated with output chaining and data transfer activity.

Control memory locations 0 and 4 are termed the buffer control word (BCW) (Figure 10-17) and define the word count, transfer mode (TM), and the byte pointers for information transfer.

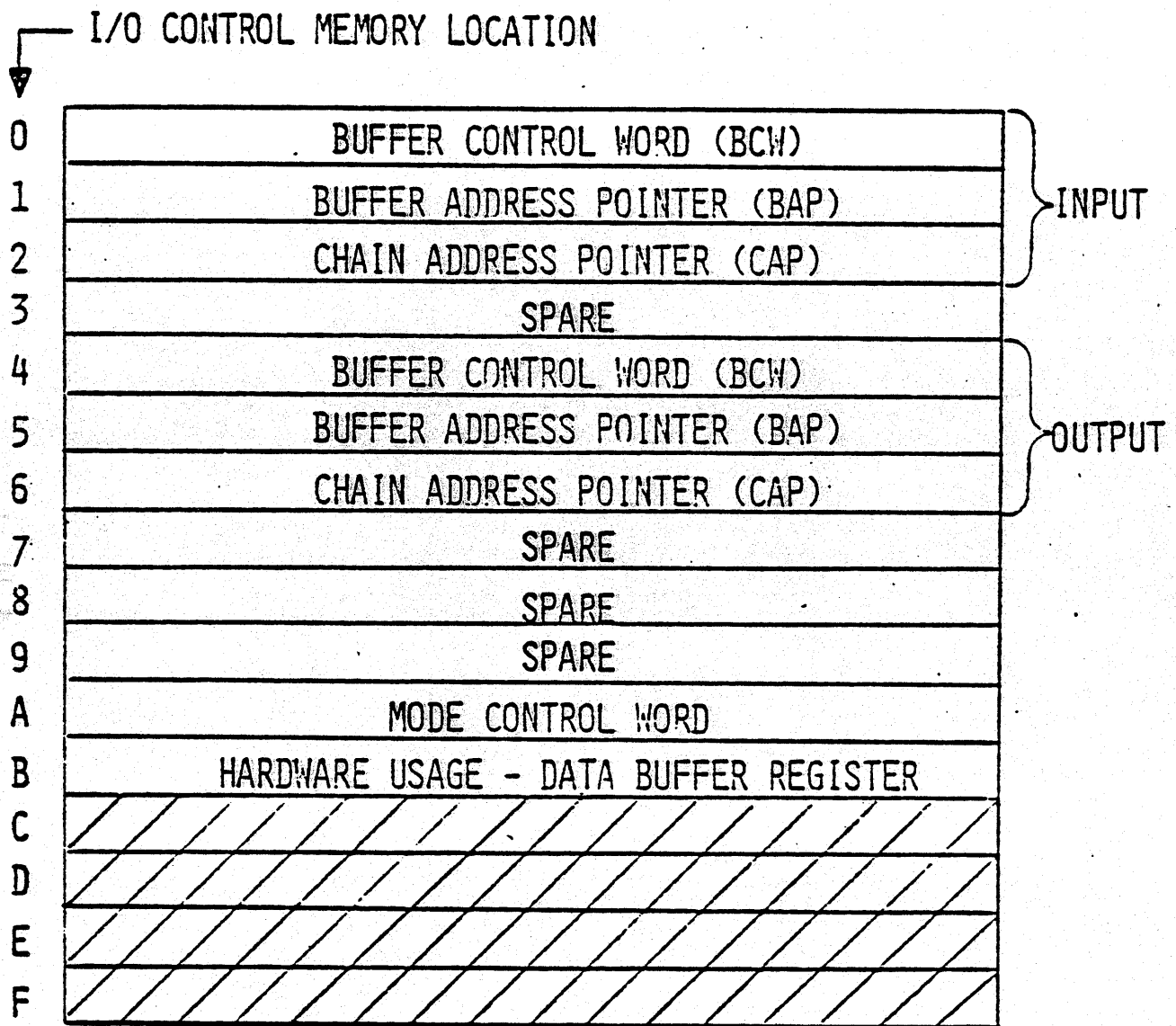


Figure 10-16. NIM Control Memory Map

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM	*	B	BUFFER TRANSFER COUNT												

TM = 00 ABORT TRANSFER (INPUT TRANSFER ONLY)
 01 BYTE TRANSFER (8 BITS)
 10 SINGLE-LENGTH TRANSFER (16 BITS)
 11 DUAL CHANNEL (32 BIT DOUBLE-LENGTH TRANSFER)

* NOT USED

B = BYTE POINTER, 0 = UPPER BYTE, 1 = LOWER BYTE

BUFFER TRANSFER COUNT = NUMBER OF BYTES, SINGLE
 WORDS, OR DOUBLE WORDS
 TO BE TRANSFERRED

Figure 10-17. NIM BCW Format

Bits 15 and 14 are termed the transfer mode (TM) and must be compatible with the mode control word in control memory location A and specify the word length to be transferred.

Bit 13 is not used.

Bit 12 is termed the byte pointer and specifies which half word (upper or lower) of the memory location will be used for the next transfer as follows:

0 = transfer upper byte (bits 15 through 8)

1 = transfer lower byte (bits 7 through 0).

This bit is toggled after each byte transfer and is interpreted only during byte transfers. The byte information is always transferred on data lines 7 through 0.

Bits 11 through 0 are termed the buffer transfer count (BTC) and specify the number of bytes, single-length words, or double-length words to be transferred during the selected operation. An initial count of zero specifies the maximum number of transfers (4096) and the contents of the BTC are decremented by one for each transfer. In byte mode, the BAP is incremented after both bytes of a memory location have been transferred.

Control memory location A is termed the mode control word (MCW) (Figures 10-18 and 10-19) and defines the mode of operation in which the channel is to operate. Only bits 3 through 0 are used with bit 3 being the mode enable bit. If bit 3 is clear, the NIM hardware default mode is enabled as shown in Figures 10-18 and 10-19.

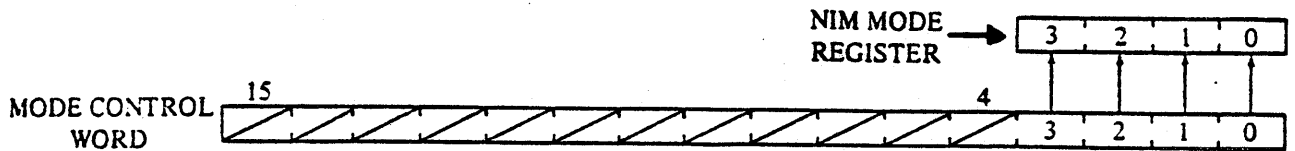
MESSAGE FORMATS (16-BIT CHANNEL)

The four types of sequences that a NIM parallel channel can perform are:

- Accept an external interrupt or command word
- Transfer an external function or command word
- Input data
- Output data

The interaction is controlled between the NIM and the peripheral device or computer via discrete signal lines following a well-defined channel protocol.

- 1) External Interrupt - A peripheral device or computer transfers an external interrupt word or command word to the NIM. The word is stored in memory, and the EII interrupt is generated.
- 2) External Function - The NIM transfers an external function word to a peripheral device or command word to a computer. The I/O chain program must initiate an external function transfer by executing an initiate transfer (operation code E3 hexadecimal with a-field equal to 2) instruction. When the peripheral device or computer is ready to accept an external function or command word, the NIM I/O channel type will transfer the data from memory. Peripheral devices which do not have an external function request line can be forced by an



MODE REG				MODE OF OPERATION
3	2	1	0	
0	0	0	0	HARDWARE DEFAULT MODE (COMPUTER TO PERIPHERAL 16 BIT)
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	COMPUTER TO PERIPHERAL - 16 BIT
1	0	0	1	COMPUTER TO COMPUTER - 16 BIT
1	0	1	0	UNDEFINED
1	0	1	1	TEST MODE
1	1	0	0	COMPUTER TO PERIPHERAL - 32 BIT
1	1	0	1	COMPUTER TO COMPUTER - 32 BIT
1	1	1	0	EXTERNALLY SPECIFIED ADDRESSING
1	1	1	1	UNDEFINED

Figure 10-18. NIM Parallel MCW Format

MODE REG				MODE OF OPERATION
3	2	1	0	
0	0	0	0	HARDWARE DEFAULT MODE (OFF)
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	OFF
1	0	0	1	CP TO CP, LOOP TEST, 16-BIT
1	0	1	0	CP TO PERIPHERAL, 16-BIT
1	0	1	1	CP TO CP, 16-BIT
1	1	0	0	CP TO PERIPHERAL, 32-BIT
1	1	0	1	CP TO CP, 32-BIT
1	1	1	0	CP TO PERIPHERAL, DUAL CHANNEL, 32-BIT
1	1	1	1	CP TO CP, DUAL CHANNEL, 32-BIT

Figure 10-19. NIM Serial MCW Format

initiate transfer (operation code E3 hexadecimal with a-field equal to 3) instruction.

Any number of external function words or command can be sent as determined by the BTC in the BCW.

- 3) Input Data - The NIM may input data from a peripheral device or a computer. The I/O chain program must initiate an input data transfer by executing an initiate transfer (operation code E3 hexadecimal with a-field equal to 0) instruction. The transfer will cause the number of words defined in the BCW to be stored in memory starting at BAP. The initiation of an input transfer may be executed from either an input or output chain. At correct termination of the transfer, the I/O chain program will resume execution.
- 4) Output Data - The NIM may output data to a peripheral device or computer. The I/O chain program must initiate an output data transfer by executing an initiate transfer (operation code E3 hexadecimal with a-field equal to 1) instruction. The transfer will cause the number of words defined in the BCW to be read from memory starting at the BAP. The initiation of an output data transfer may be executed from either an output or input chain. At correct termination of the transfer, the I/O chain program will resume execution.

MESSAGE FORMATS (32-BIT CHANNEL)

The five types of sequences which a NIM 32-bit parallel channel can perform are:

- Accept an external interrupt or command word
- Transfer an external function or command word
- Input data
- Output data
- Externally specified addressing (ESA) in computer-to-peripheral mode only.

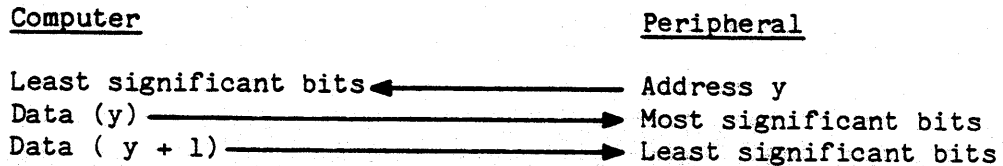
Requirements of a 32-bit channel are as follows:

- Both NIMs must be of the same type
- Both NIMs must be in a physical channel pair (0 and 1, 2 and 3, 4 and 5, or 6 and 7)
- Channel operation is controlled by programming at the software level of the odd-numbered channel
- Control memory for the odd-numbered channel is used by the chain program.
- The MCW for the even-numbered channel must also be set to the current mode of operation
- Most significant 16 bits (even channel) uses BAP and the least significant 16 bits (odd channel) uses BAP ⊕ 1 from the odd channels control memory.
- TM field of the BCW must be set to 11.

The external interrupt, external function, command word, and data transfer sequences function the same as on a 16-bit channel with the 32-bit interrupts and command words stored in the appropriate memory locations.

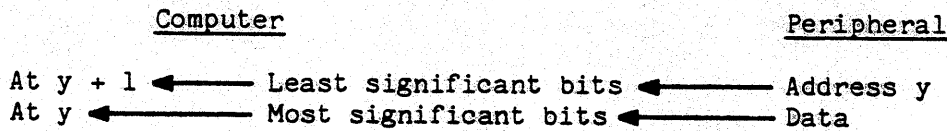
In the ESA mode, the NIM pair transfers data to and from the peripheral device on a word-by-word basis. The external interrupt and external function sequences are programmed and performed the same as when in computer-to-peripheral 32-bit mode. The data transfers are enabled by the execution of an initiate equal to 0000 or 0001 in the chain program. At that point, chaining activity stops and the NIMs are waiting on the peripheral device.

a = 0001 (Output data)



BAP and BTC are not used and chaining activity is not reinitiated.

a = 0000 (Input data)



BAP and BTC are not used and chaining activity is not reinitiated. These transfers are terminated via the execution of a channel control (CCR a,8) instruction from the command cell.

NIM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on a NIM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (E0 9m) (ACR m; CCR a,m) - The same as listed under Processor to I/O Channel Communication paragraph except for the CCR instruction the a-field is not used and the associated I/O channel is the one executing the I/O channel program.
- 2) Initiate Transfer (E3 a 0) (IO a,y) - If the a-field is equal to XX00, load control memory locations 0(BCW) and 1(BAP) with the contents of main memory addresses y and y + 1. If the a-field is not equal to XX00, load control memory locations 4(BCW) and 5(BAP) with the contents of main memory addresses y and y + 1, then perform the operation as specified by the a field as follows:

<u>a</u>	<u>Operation</u>
0000	Initiate input data transfer using the BCW and BAP in control memory locations 0 and 1.

- 0001 Initiate output data transfer using the BCW and BAP in control memory locations 4 and 5.
 - 0010 Initiate external function transfer using the BCW and BAP in control memory locations 4 and 5.
 - 0011 Initiate external function transfer with force using the BCW and BAP in control memory locations 4 and 5.
- 01XX }
 10XX } No operation
 11XX }

The associated I/O channel is the one executing the I/O chain program.

- 3) Load Control Memory (E6 0 m) (LCMK m,y) - The same as listed under SIM I/O Channel Instructions paragraph.
- 4) Load Control Memory (E7 0 m) (LCM m,y) - The same as listed under SIM I/O Channel Instructions paragraph.
- 5) Store Control Memory (EB 0 m) (SCM m,y) - The same as listed under SIM I/O Channel Instruction paragraph.
- 6) Halt/Interrupt (EC a 0) (HCR; IPR) - The same as listed under DIM I/O Channel Instruction paragraph.
- 7) Set/Clear Flag (EF a 0) (ZF y; SF y) - The same as listed under SIM I/O Channel Instructions paragraph.
- 8) Conditional Jump (F2 00) (SJMC 0,y) - The same as listed under DIM I/O Channel Instructions paragraph.
- 9) Store Status (FB 0 m) (CSST y,m) - Store the channel status as specified by the m-field in main memory location y as follows:

<u>m</u>	<u>Status Word Value</u>
0XXX	Mode status word (Figures 10-20 and 10-21)
1XXX	Channel hardware status word 0 (Figure 10-22)
- 10) Bit Jump (FD 0 m) (BJ m,y) - The same as listed under DIM I/O Channel Instructions paragraph.
- 11) Exchange Control Memory (FE 0 m) (XCM m,y) - The same as listed under DIM I/O Channel Instructions paragraph.

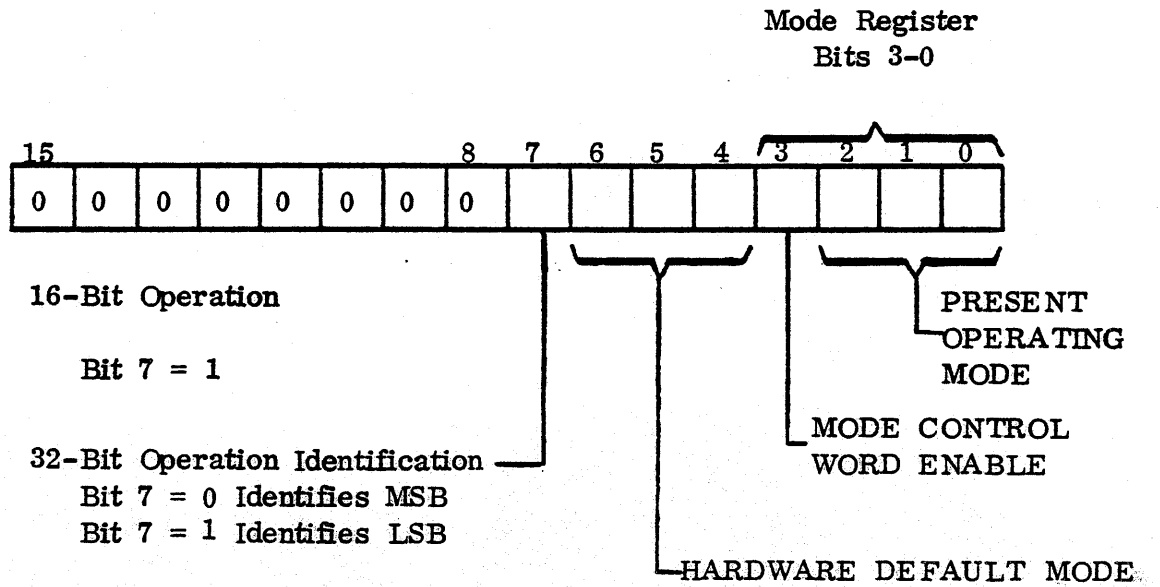


Figure 10-20. NIM Parallel Mode Status Word

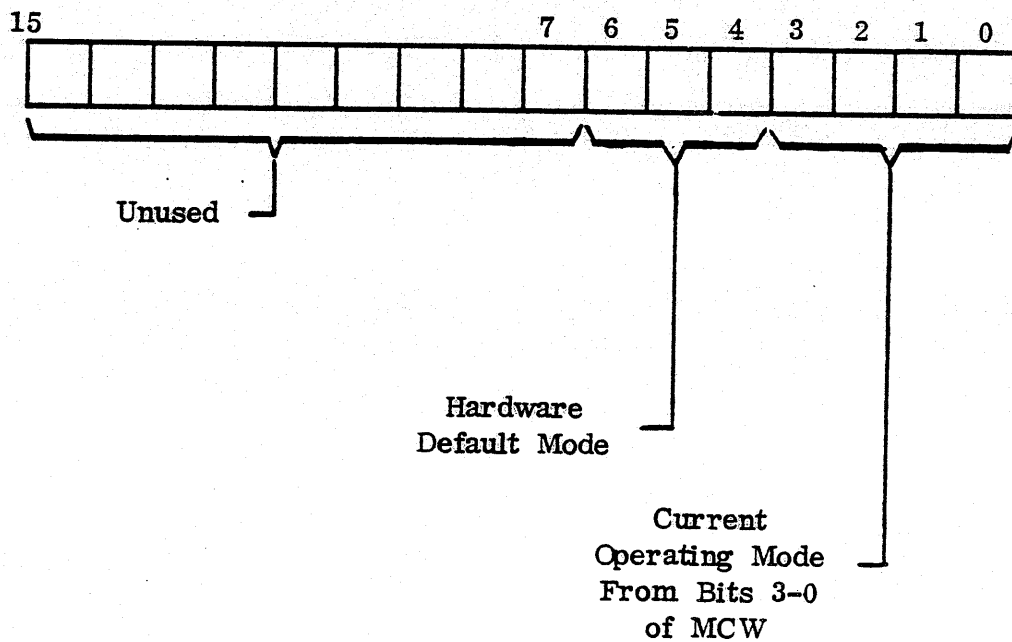


Figure 10-21. NIM Serial Mode Status Word

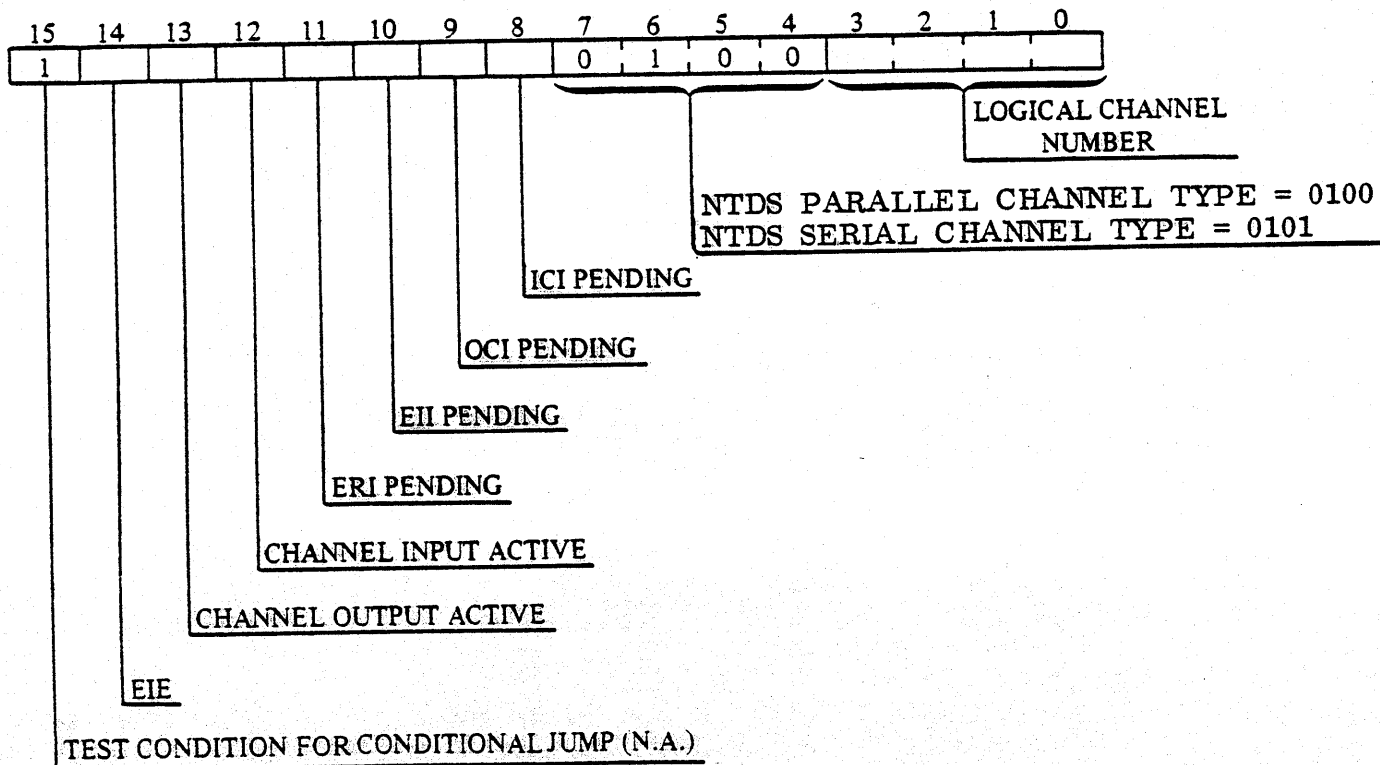


Figure 10-22. NIM Hardware Status Word 0

NIM INTERRUPT HANDLING

The NIM I/O channel types are capable of generating all four of the Class III interrupts: ERI, EII, OCI and ICI.

A parallel NIM shall generate an ERI interrupt whenever time-out occurs on an intercomputer operation. If the receiving computer fails to accept an output word within the time-out limits (300 to 400 milliseconds), the transmitting NIM will generate the ERI interrupt. A serial NIM shall generate an ERI interrupt whenever the peripheral device or computer does not respond within 20 microseconds. The serial NIM generates the ERI stops the data transfer operation and does not reinitiate chaining activity. The ERI interrupt will remain active until recognized and processed by the computer. The interrupt software program will have to resolve the no response condition. This interrupt cannot be masked at the channel level.

The NIM hardware shall generate an EII interrupt any time the external interrupts are enabled (EIE set) in response to receiving an interrupt word from a peripheral device or a command word from another computer. A peripheral device or computer shall not attempt to send the NIM channel a word and create the EII interrupt unless external interrupts are enabled.

The NIM hardware and processor firmware will perform the following events in response to an ERI active and a command word when in computer-to-computer mode or an interrupt word when in computer to peripheral mode.

- Store the command interrupt word in memory (80_{16} plus channel), most significant 16 bits even channel and least significant bits odd channel when in the 32-bit mode.
- Acknowledge receipt of the word by activating IDA signal.
- Disable external interrupts, EIE signal drops.
- Hold interrupt pending at channel level and set bit 10 in hardware status word 0 if Class III, priorities 2, 3, and 4 are disabled.
- Forward EII interrupt to processor if Class III, priorities 2, 3, and 4 are enabled.
- Drop IDA signal when EIR drops.

The NIM I/O channel types shall generate OCI and ICI interrupts under I/O chain program control via the interrupt processor (operation code EC hexadecimal) instruction. If executed in an output chain program, an OCI is generated, and if executed in an input chain program, an ICI is generated.

PROGRAMMING CONSIDERATIONS (16-BIT MODE)

To enable the NIM to handle an external interrupt word from a peripheral device or a command word from another computer requires the programmer to enable external interrupts (EIE set) on the channel. This can be done with a CCR a, 12 instruction via the command cell. The Class III priorities 2, 3, and 4 must also be enabled on the channel, or the EII interrupt is held pending at the channel level.

After the above conditions are met and the NIM is sent an interrupt or command word, the word will be placed in memory, the EIE cleared on the channel, and the EII interrupt generated. The NIM will no longer respond to external interrupt or command words until the EIE is set again via software.

External Function Sequence

Function words (16 bit) are sent to the peripheral device. The BTC in BCW determines the number of function words to be sent while BAP points to the memory location of the first word. The sequence is initiated in a chain program by execution of an IO a,y,m instruction with the a-field equal to 0010 or 0011. The sequence can be initiated from an input or output chain program, and upon completion of the transfer operation, the output chain activity is initiated. Also, regardless of which chain the transfer is initiated from, the BCW in control memory location 4 and the BAP in control memory location 5 are used during the transfer operation.

Once the sequence is initiated, chaining activity stops and the function words from memory are sent until the BTC equals 0. If the a-field of the IO instruction is 0011, a forced condition exists. If the a-field of the IO instruction is 0010, function words from memory will be sent until BTC equals 0, or the peripheral device is no longer ready to accept them. Termination because BTC equals 0 causes the output chain program to be initiated. If the peripheral device goes not ready during a transfer operation with an IO instruction a-field equal to 0010, the transfer stops and chaining activity is not reinitiated. Software will not be notified by interrupts or firmware.

Command Word Sequence

Command words (16 bits) are sent to another computer. The sequence is initiated in a chain program by execution of an IO a,y,m instruction with an a-field equal to 0010. The sequence can be initiated from either an input or output chain program, and upon completion, the output chain activity is initiated. The BCW in control memory location 4 and the BAP in control memory location 5 are always used during the command word transfer operation.

Once the sequence is initiated, chaining activity stops and the command words from memory are sent until BTC equals 0, or the receiving computer fails to respond. Termination via the BTC equals 0 causes the output chain activity to be started. Termination via no computer response causes the ERI interrupt to be generated, and chaining activity is not initiated.

Data Transfer Sequence

Bytes or words are transferred through the NIM to or from a peripheral device or another computer. The sequence is initiated from a chain program by execution of an IO a,y,m instruction. An input data transfer always uses BCW in control memory location 0 and BAP in control memory location 1, while an output data transfer always uses BCW in control memory location 4 and BAP in control memory location 5. This is true regardless of initiating the data transfer from the same chain (input transfer from an input chain) or different chain (input transfer from an output chain). Upon correct completion of the data transfer sequence, the associated chain activity is started (output transfer, output chain; input transfer, input chain).

Once the sequence is initiated, chaining activity stops and the data transfer takes place until BTC equals 0, and the peripheral device or the computer stops responding. If a peripheral device stops responding during the data transfer, the NIM stops and waits. No interrupts are generated and chaining is not started. The software will have to remove the NIM from this state by initiating another operation.

If a computer fails to respond within 400 milliseconds after every word sent during an output data sequence, the ERI interrupt is generated.

PROGRAMMING CONSIDERATIONS (32-BIT MODE)

A parallel NIM channel in 32-bit mode is enabled to handle external interrupt and command words the same as the 16-bit mode. That is, enable external interrupts (EIE set) and enable Class III priorities 2, 3, and 4 on the odd channel of the pair. The 32-bit external interrupt word or command word will be stored with the most significant bits in memory at 80_{16} plus the even channel number and the least significant bits in memory at 80_{16} plus the odd channel number.

The external function, command word, and data transfer sequences are initiated the same and terminate the same, when BTC equals 0 or for an error condition. The only difference is that the most significant bits are addressed by BAP and the least significant bits are addressed by BAP \oplus 1.

PROGRAMMING CONSIDERATIONS (SERIAL CHANNEL)

In 16-bit mode all channel words are 16 bits in length. The TM field of the BCW must be 10 (for word), and the BTC is set to the number of 16-bit words.

In 32-bit dual channel mode, all channel words are 32 bits in length. The TM field of the BCW must be 11 (for double word), and the BTC is set to the number of 32-bit words. The NIM serial channel must be physically located in an odd-numbered slot, and the next physical even-numbered slot must be left empty. The external interrupt words and command words are stored in memory the same as for 32-bit mode (parallel NIM).

In 32-bit mode all data words are 32 bits. The TM field of the BCW must be 10 (for word), and the BTC is set to the number of 16-bit words. The serial NIM will send 16-bit external functions (least significant bits) and command words from memory at BAP, storing only the least significant bits of accepted command and external interrupt words in memory at 80_{16} plus the channel number. The difference between this mode and 32-bit mode does not matter.

The serial NIM handles external interrupts and command words identical to the parallel NIM. If external interrupts are enabled (CCR a, 12) and Class III priorities 2, 3, and 4 are enabled (CCR a, 14), the external word (or portion) is stored in memory, the EII interrupt is generated, and external interrupts are disabled.

External Function Sequence

Function words are sent to a peripheral device in much the same order as for a parallel NIM. All the rules about sequence initiation, with or without force and termination when the BTC equals 0 are the same. The sequence can be initiated from either chain program, always uses control memory location 4 and control memory location 5 and will always start the output chain upon correct termination.

Once initiated, if the peripheral device does not respond within 20 microseconds, the NIM will generate the ERI interrupt, and output chaining activity is not started. If the sequence is not an external function with force, the NIM will wait until the peripheral device becomes ready before sending the external function word. The NIM will not time out on a ready condition.

Command Word Sequence

The command word sequence functions the same as the external function word sequence except that a command word sequence cannot be done to another computer with force (a-field equal to 0011).

Data Transfer Sequence

Single 16-bit words or double 32-bit words are transferred following all the same sequence initiation, terminating when BTC equals 0, and using the same control memory locations and chain program initiation rules. Most significant bits of the 32-bit word correspond to BAP+1 and least significant bits correspond to BAP with least significant bits transmitted first and received first.

The differences are with respect to the ERI interrupt. The serial NIM will generate the ERI interrupt whenever the peripheral device or the computer does not respond within 20 microseconds. If either device goes not ready, it must continue to communicate with the computer while not ready.

RS-232-C INTERFACE MODULE

The RS-232-C interface module (RIM) has the following general characteristics:

- One full duplex channel
- Asynchronous or synchronous
- Programmable baud rate and parity selection
- Programmable character length - 5,6,7 or 8 bits
- Internal loop test and echo test capabilities
- Programmable number of stop bits - 1 or 2 bits.

All of the clock and control signals as well as the data movement is controlled by the RIM

DATA FORMATS

Figure 10-23 shows the data formats for both sync and async operation. For sync operation, the number of bits and parity are selectable by software. For async, the number of bits, parity, and stop bits are selectable by software. The start bit is always a low and the stop bits are always high. The number of data bits per character is selectable from 5 through 8. Parity can be either odd, even, or no parity.

MESSAGE FORMATS

All messages that the RIM handles consist of either an output data transfer or an input data transfer. There are no special message formats for interrupt or function words. The meaning of the data sent or received is determined by the software. Some of the control communications on the channel are handled by the RIM hardware and firmware where others are the responsibility of the programmer.

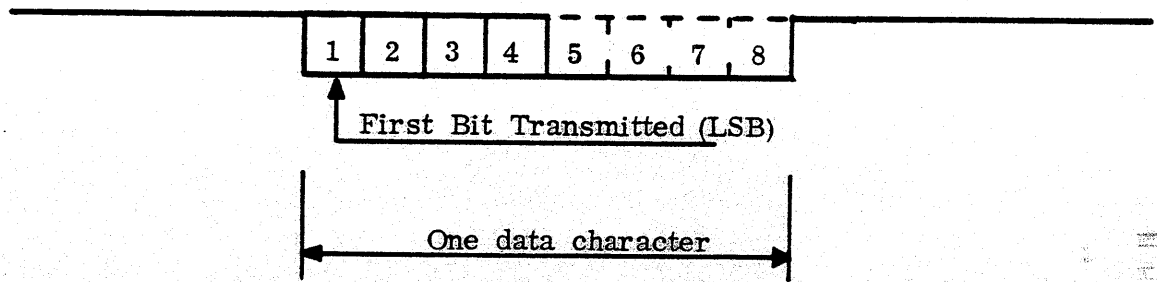
CONTROL MEMORY DEFINITION

Control memory format and usage for the RIM I/O channel type is shown in Figure 10-24. Control memory locations 0, 1, and 2 are associated with input activity and control memory locations 4, 5, and 6 are associated with output activity.

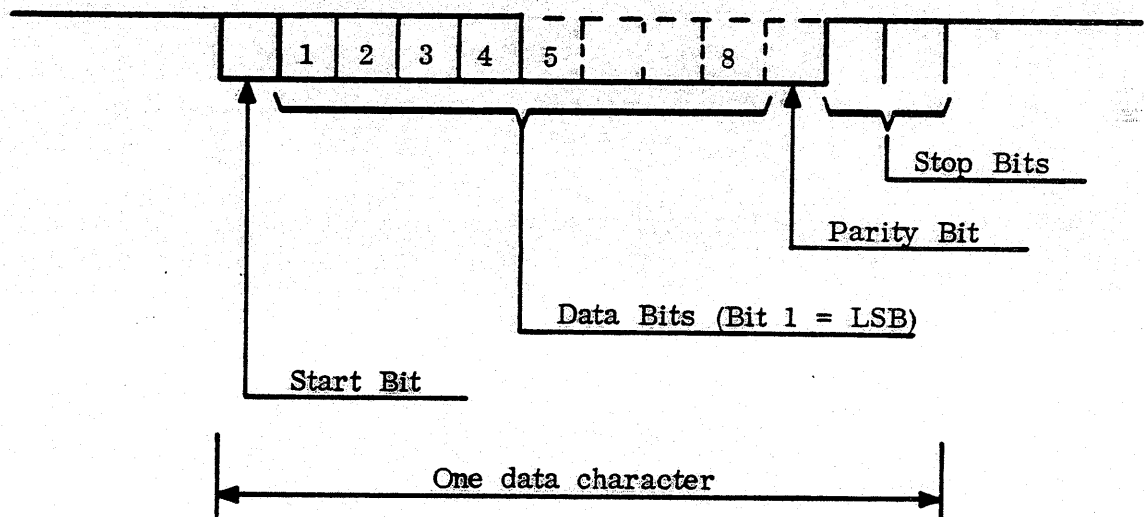
Control memory locations 0 and 4 are termed the buffer control word (BCW) (Figure 10-25) and define the word count, transfer mode (TM), and the byte pointer for I/O information activity.

Bits 15 and 14 are termed the transfer mode (TM) and are either 01 (byte transfer) or 10 (word transfer). In word transfer mode, one byte (bits 7 through 0) of each memory location is used.

Bit 13 is not used.



SYNCHRONOUS FORMAT



ASYNCHRONOUS FORMAT

Figure 10-23. RIM Sync/Async Data Formats

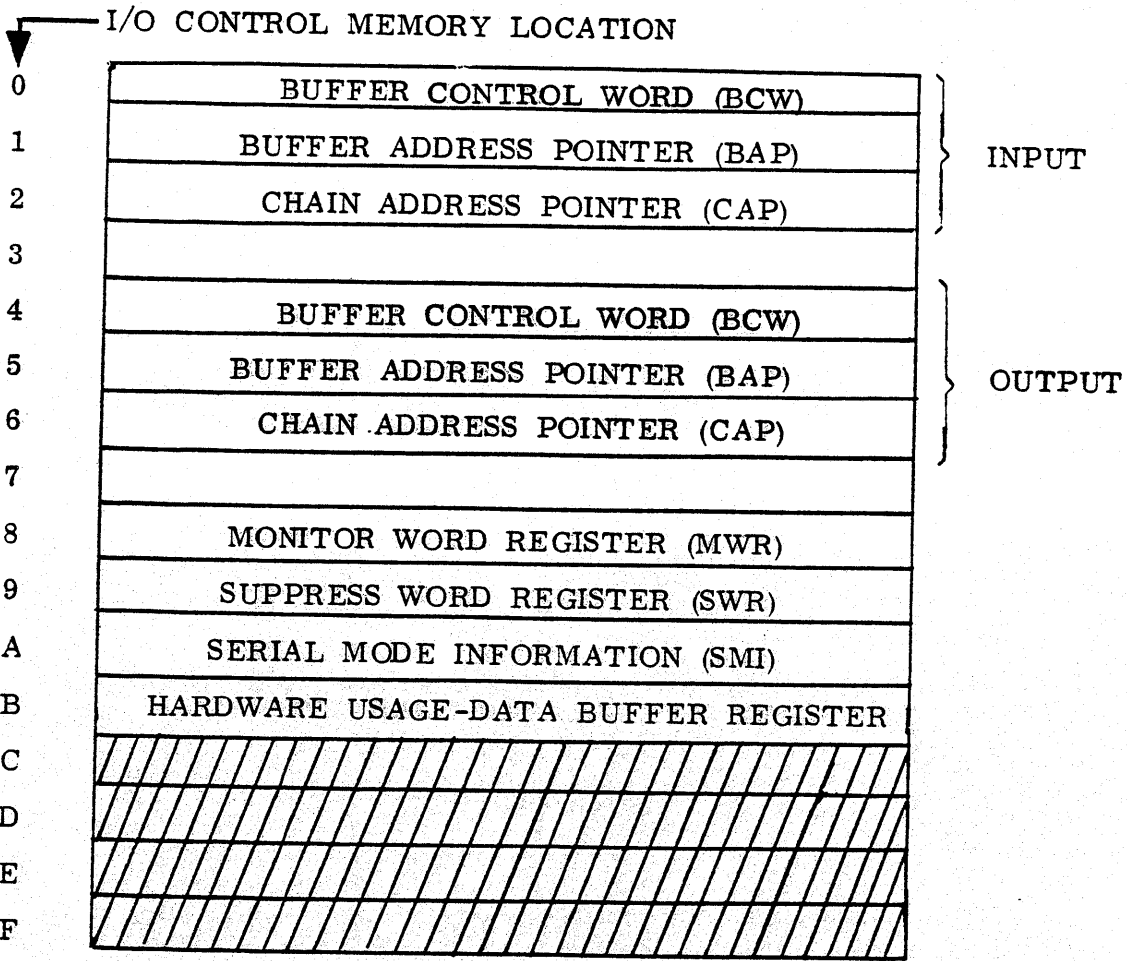


Figure 10-24. RIM Control Memory Map

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM	*	B	BUFFER TRANSFER COUNT												

Figure 10-25. RIM BCW Format

Bit 12 is termed the byte pointer and specifies which half word (upper or lower) of the memory location will be used for the next transfer as follows:

0 = transfer upper byte (bits 15 through 9)

1 = transfer lower byte (bits 7 through 0)

This bit is toggled after each byte transfer and is interpreted only during byte transfer.

Bits 11 through 0 are termed the buffer transfer count (BTC) and specify the number of bytes to be transferred during the selected operation. The contents of the BTC are decremented by one for each transfer.

Control memory location 8 is termed the monitor word register (MWR) and holds a character in the least significant bits that are compared by the RIM hardware with each incoming character, when enabled. If they compare, the character is stored in main memory, the monitor flag is set, the input transfer is terminated, and the input chaining activity is reinitiated.

Control memory location 9 is termed the suppress word register (SWR) and holds a character in the least significant bits that are compared by the RIM hardware with each incoming character, when enabled. If they compare, the suppress flag is set, and the character is not stored in main memory.

The monitor and suppress flags are in the RIM hardware and are testable via the chain conditional jump instruction. They are cleared upon input of the next character.

Control memory location A is termed the serial mode information (SMI) (Figure 10-26) and is used to select all the features of the data characters and baud rates.

Bits 15 through 11 are not used.

Bits 10 through 7 are termed the AN/AYK-14 asynchronous clock speed selection bits (see Table 10-1) and are used to select the baud rate.

Bits 6 and 5 are termed the AN/UYK-20 mode asynchronous clock speed selection bits (see Table 10-1) and are used to select one of the four speeds already selected by jumper wires on the I/O connector.

Bits 6 and 5 00 = lowest speed of the four

Bits 6 and 5 11 = highest speed of the four

Bit 4 is used to select one stop bit or two stop bits.

Bit 3 is used to enable or disable parity checking and generation.

Bit 2 is used to select odd/even parity.

Bits 1 and 0 are used to specify 5-, 6-, 7-, or 8-bit characters.

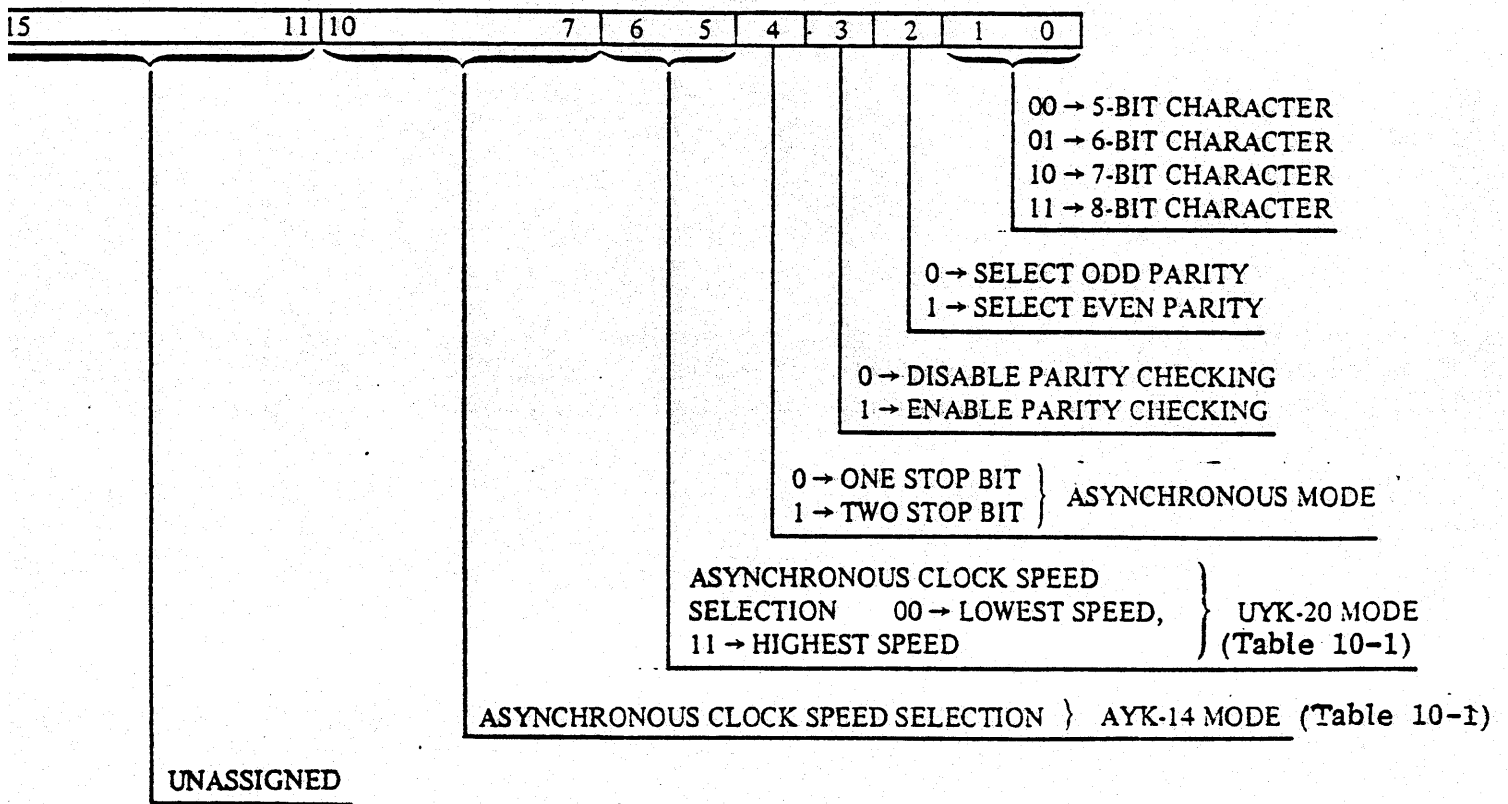


Figure 10-26. SMI Word Format

TABLE 10-1. RIM CHANNEL SPEED SELECTION

<u>AYK-14 PROGRAMMABLE BAUD RATE</u>			
<u>SERIAL MODE BITS</u>			
<u>10</u>	<u>9</u>	<u>8</u>	<u>7</u>
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

<u>BAUD RATE</u>	<u>UYK-20</u>
	<u>AVAILABLE BAUD RATES</u>
-	75
-	110
50	150
75	300
134.5	600
200	1200
2400	1800
9600	2400
4800	4800
1800	9600
1200	
2400	
300	
150	
110	

RIM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on a RIM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (E0 a m) (ACR m; CCR a,m) - The same as listed under Processor to I/O Channel Communication paragraph except for CCR instructions the a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 2) Initiate Transfer (E3 a 0) (IO a,y) - If the a-field is equal to XX00, load control memory locations 0 (BWC) and 1 (BAP) with the contents of main memory addresses y and y + 1, then initiate input word transfer using control memory location 0 (BWC) as buffer word count. If the a-field is equal to XX01, load control memory locations 4 (BWC) and 5 (BAP) with the contents of main memory addresses y and y + 1, then initiate output word transfer using control memory location 4 (BWC) as buffer word count. The associated I/O channel is the one executing the I/O chain program.
- 3) Load Control Memory (E6 0 m) (LCMK m,y) - Load the control memory location specified by the m-field with the value y.
- 4) Load Control Memory (E7 0 m) (LCM m,y) - Load the control memory location specified by the m-field with the contents of main memory address y. The associated I/O channel is the one executing the I/O chain program.
- 5) Store Control Memory (EB 0 m) (SCM m,y) - Store the contents of the memory location specified by the m-field at main memory address y. The a-field is not used, and the associated I/O channel is the one executing the I/O chain program.
- 6) Halt/Interrupt (EC a 0) (HCR; IPR) - Perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Halt I/O chaining activity
XXX1	Generate Class III, priority 4, ICI interrupt if input chain active or Class III, priority 3, OCI interrupt if output chain active.

The m-field is not used.

- 7) Set/Clear Flag (EF a 0) (ZF y; SF y) - The same as listed under SIM I/O Channel Instructions paragraph.

- 8) Conditional Jump (F2 a 0) (SJMC a,y) - This instruction causes a jump to y if one of the following a-field conditions is met:

<u>a</u>	<u>Operation</u>
XX00	Unconditional jump
XX01	Jump if suppress flag not set
XX10	Jump if monitor flag set

The suppress or monitor flag is cleared upon input of the next character.

- 9) Search for Sync (F4 0 m) (SFSC m) - This instruction conditions the next activated input buffer to perform operations as specified by the m-field as follows:

<u>m</u>	<u>Interface</u>	<u>Operation</u>
0000	Synchronous	Disable search for sync, disable monitor mask, disable suppress mask, and enable next chain instruction.
0001	Synchronous	Hold sync active and enable next chain instruction.
0010	Asynchronous	Enable suppress mask and enable next chain instruction.
0011	Synchronous	Enable suppress mask and enable next chain instruction.
0100	Asynchronous	Enable monitor mask and enable next chain instruction.
0101	Synchronous	Enable monitor mask and enable next chain instruction.
0110	Asynchronous	Enable monitor mask, enable suppress mask, and enable next chain instruction.
0111	Synchronous	Enable monitor mask, enable suppress mask, and enable next chain instruction.
1001	Synchronous	Enable search for sync and disable chaining.
1011	Synchronous	Enable search for sync, enable suppress mask, and disable chaining.

The a-field is not used, and the associated I/O channel is the one executing the I/O chain program.

NOTE:

When operating in multiple I/O configurations (e.g. three SIMs and one RIM), the second SYNC word may be lost.

- 10) Serial Interface Control (F8 0 m) (CSIR m) - Set or clear the discrettes as specified by the m-field. The a-field is not used, and the associated I/O channel is the one executing the I/O chain program.

<u>m</u>	<u>Function</u>	<u>Discrete</u>
0000	Set	Loop test (internal)
0001	Clear	Loop test (internal)
0010	} Not Used	
0011		
0100		
0101		
0110	Clear	Enable ring indicator (internal)
0111	Set	Enable ring indicator (internal)
1000	Clear	Request to send
1001	Set	Request to send
1010	Clear	New sync
1011	Set	New sync
1100	Clear	Data terminal ready
1101	Set	Data terminal ready
1110	Clear	Loop test (external)
1111	Set	Loop test (external)

- 11) Store Status (FB 0 m) (CSST y,m) - Store the channel status as specified by the m-field at main memory address y. The a-field is not used and the associated I/O channel is the one executing the I/O chain program.

<u>m</u>	<u>Status Word Value</u>
1XXX	Hardware status word 0 (Figure 10-27)
0XXX	Hardware status word 1 (Figure 10-28)

RIM INTERRUPT HANDLING

The RIM I/O channel type is capable of generating the EII, OCI, and ICI Class III interrupts.

The RIM hardware shall generate an EII interrupt when:

- 1) The received signal line detector signal goes off. The data communication equipment will turn off this signal when it is receiving a poor quality incoming signal from the other communication equipment.
- 2) The ring indicator signal goes on and the ring indicator has been enabled via an CSIR instruction with an m-field equal to 0111.

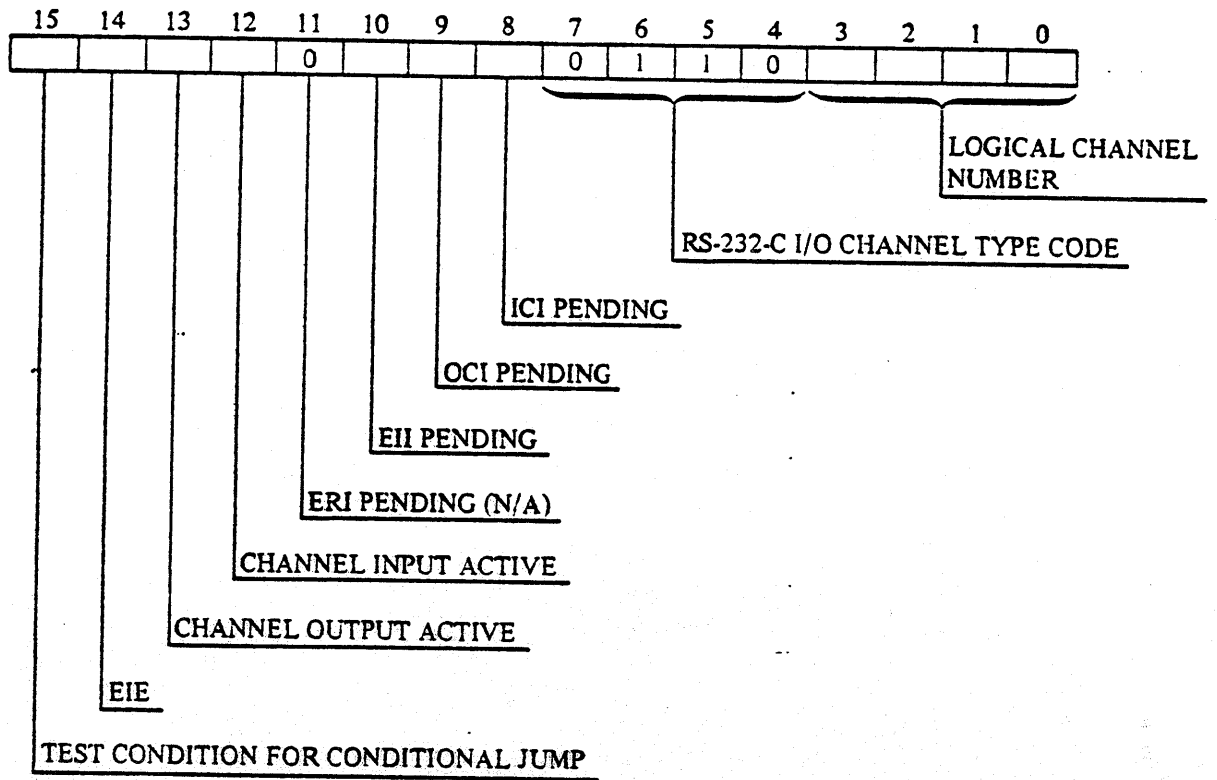


Figure 10-27. Hardware Status Word 0 Format

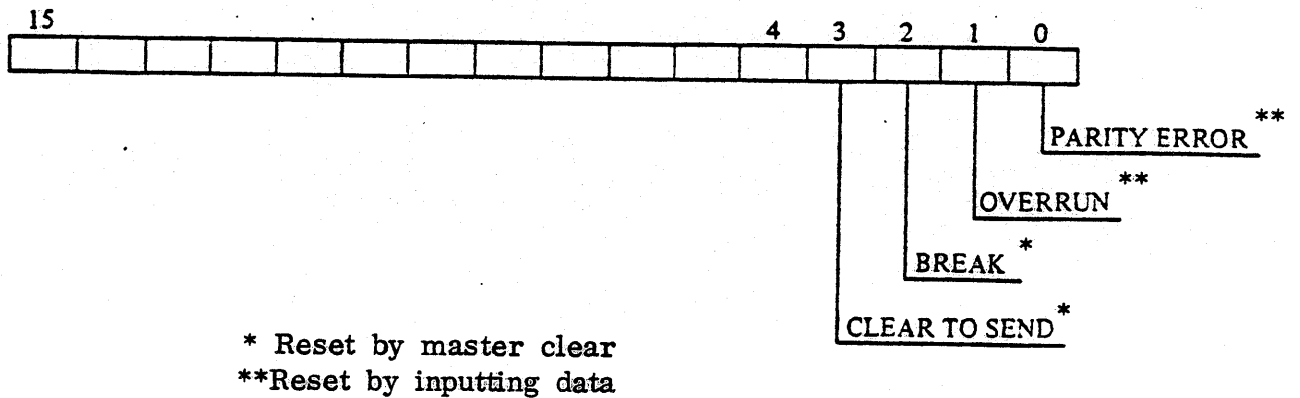


Figure 10-28. Hardware Status Word 1 Format

When either of these conditions occur during an input transfer where external interrupts on the RIM have been enabled (EIE set) and Class III, priority 2, 3, 4 enabled, the RIM interrupt word (Figure 10-29) is stored in memory at 80_{16} plus the channel number, the EII interrupt is generated and forwarded to the processor, and the input chaining activity is not reinitiated. If either of the enables are not enabled, the interrupt word is not stored and the EII interrupt is held pending at the channel level.

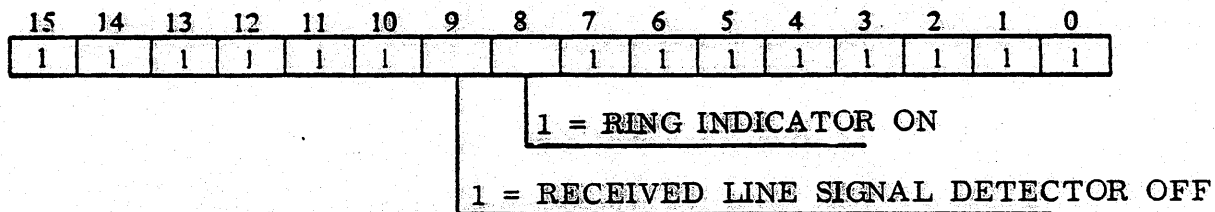


Figure 10-29. RIM Interrupt Word Format

The RIM I/O channel type shall generate OCI and ICI interrupts under I/O chain program control via the halt/interrupt (operation code EC hexadecimal) instruction. If executed in an output chain program, an OCI is generated, and if executed in, an input chain program an ICI is generated.

PROGRAMMING CONSIDERATIONS (ASYNC)

The RIM will not generate an EII interrupt during an output data transfer. The activation of the control signals is a software function (Figure 10-30). This can be done either from the command cell with an CSIR instruction or in a chain program with the CSIR instruction.

Clear to send signal from the communications equipment appears in hardware status word 1. Once it has been determined, via the program clear to send signal being active, initiate the output data transfer with an IO a,y,m instruction a-field equal to XX01. Chaining activity stops and the RIM hardware and processor firmware output the correct number of bytes. When BTC equals 0, the output chaining activity is reinitiated.

An output data transfer can be initiated from either an input or output chain program. Control memory locations 4 and 5 are used during the data transfer and upon termination of the transfer, the output chain activity is started in either case. If there is concern about the clear to send signal dropping during the data transfer, check it after the transfer before the chain program starts.

An input data transfer is programmed as shown in Figure 10-31. If the suppress and monitor features are to be used they must be loaded and enabled. Initiate the transfer with an IO a,y,m instruction a-field equal to XX00. The chaining activity is stopped, and the input data transfer is started.

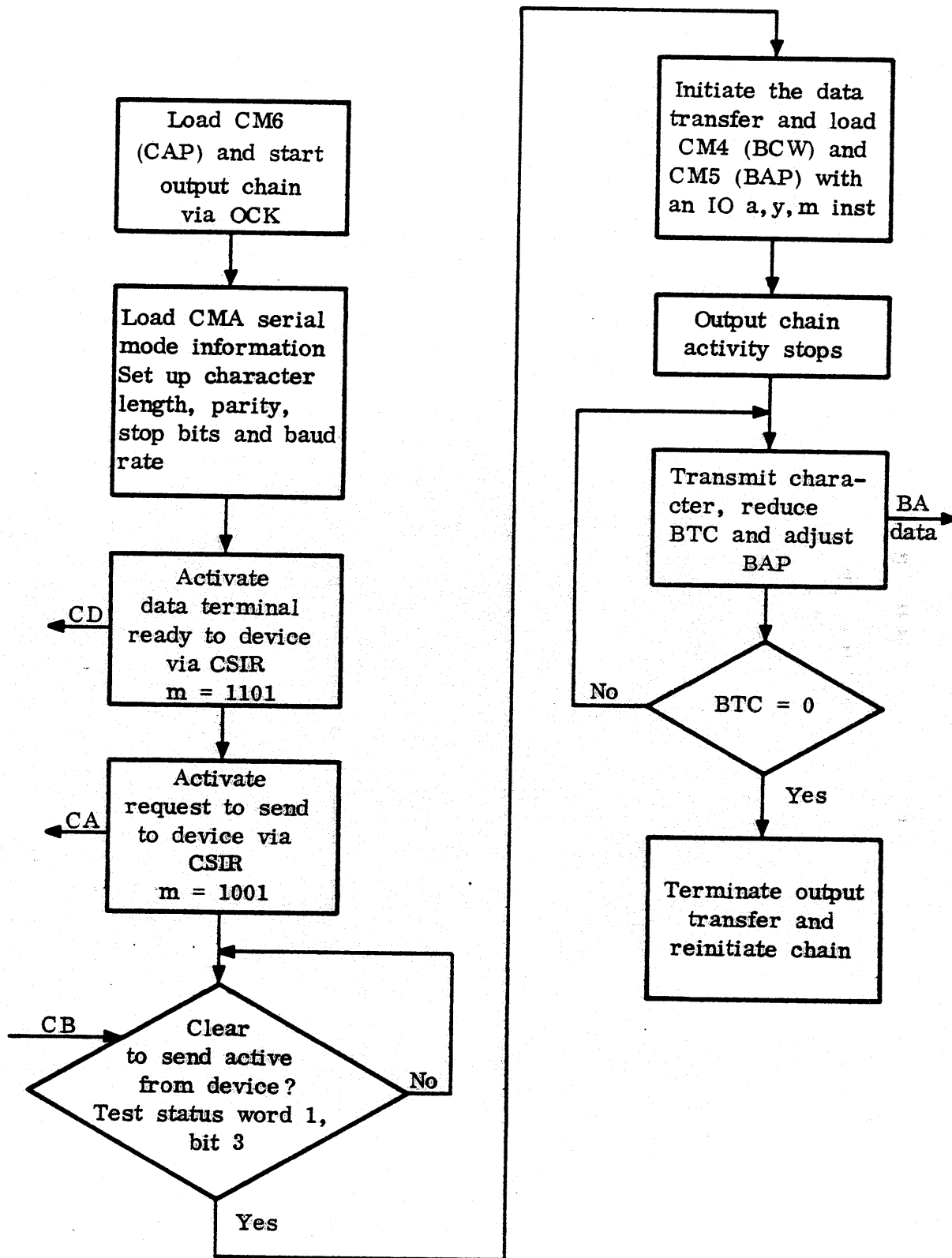


Figure 10-30. RIM Async Output Data Transfer

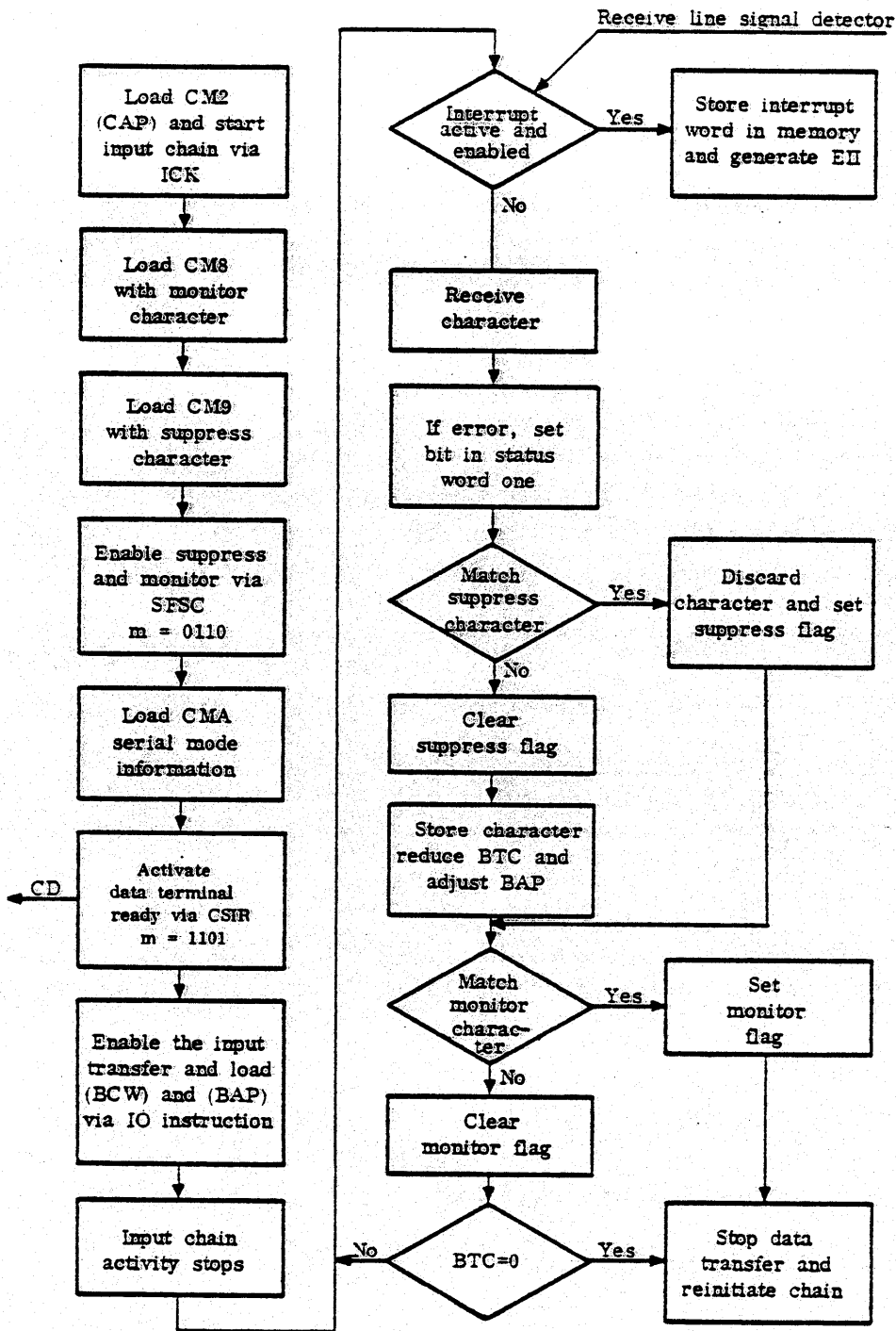


Figure 10-31. RIM Async Input Data Transfer

If during the transfer the receive line signal detector signal goes off, the data transfer stops, and the EII interrupt is generated based on the enable conditions.

If the monitor and/or suppress features are enabled, the RIM hardware will perform the appropriate functions on the incoming data. Normal input data transfer termination occurs when BTC is equal to 0. The input data transfer can be initiated from either an input or output chain program. Control memory locations 0 and 1 are used, and upon termination, the input chain activity is started.

The RIM will respond to the ring indicator going active, store the interrupt word in memory, and generate the EII interrupt if enabled via the proper CCR instruction.

PROGRAMMING CONSIDERATIONS (SYNC)

In synchronous mode the baud rate is determined by the device connected to the RIM; therefore, the serial mode information word in control memory location A only uses bits 0 and 1 to select character size and bit 3 to disable parity. Control memory location usage is the same as for asynchronous. The hardware status words are the same also. In hardware status word 1, the only meaningful bit is the overrun bit.

The EII interrupt will be generated for the same reasons following the state of enables as described in the Interrupt paragraph.

The CSIR instruction, with m-field equal to 1010 or 1011, is used to set or clear the new sync control signal to the communications equipment and is used as the system dictates.

The SFSC instruction is used to place the RIM in sync with the communications equipment and enable the suppress and monitor functions during the input data transfer.

Input Data Transfer

Once the interrupts are enabled and the serial information word is set up in control memory, load the suppress register with the character to be compared with the incoming data by the RIM hardware for a sync condition. The character is placed in the suppress register right justified with the unused bits and the parity bit set to 1's. Next, load the monitor register with the character used to terminate on for the input data transfer, unused bits must be 1's. To place the RIM in sync with the communications equipment, execute a SFSC instruction with the m-field equal to 1001. The chaining activity stops, and the RIM hardware compares the incoming data with the character in the suppress register. When the RIM hardware detects a match, the chaining activity is started again. Two successive matches must occur to ensure that it is in sync. The RIM hardware is still doing the comparison and controlling the suppress flag. Testing the suppress flag must be included in the chain program. If the second match does not occur, the suppress flag will not be set, and the SJMC instruction will be executed to jump back to the SFSC instruction and start the search for sync again. If the second match did occur, the suppress flag is set, and the SJMC instruction will continue on to the next chain instruction.

At this point, enable the suppress and monitor registers to be used as in an async operation during the input data transfer with an SFSC instruction m-field equal to 0111. Initiate the input transfer with an IO a,y,m instruction a-field equal to XX00. The input data transfer will now operate the same as an async (Figure 10-31) and terminate under the same conditions.

Output Data Transfer

Load the serial information word into control memory location A to select the number of bits in the character. Load control memory locations 4 and 5 and initiate the output data transfer with an IO a,y,m instruction a-field equal to XX01. This stops the chaining activity and starts the data transfer. The data transfer will terminate and output chain activity will start when BTC equals 0. The first two characters of the output buffer data are the sync characters.

Wraparound Mode

The RIM I/O channel type has a built-in control logic wraparound mode of testing. The transmitters and receivers are not tested when this function is executed. The wraparound mode (internal loop test) is enabled and disabled via the SICR instruction.

PROTEUS INTERFACE MODULE

The PROTEUS interface module (PIM) has the following general characteristics.

- Accommodates two PROTEUS digital channels (PDC)
- Input and output chain programs can operate simultaneously allowing full duplex operation
- Provides clocked differential nonreturn-to-zero (NRZ) encoded data
- Serial data transfer rate of 125K words/second
- Provides channel error correction capability
- Provides channel failure indications

The PDC that is defined as the source transmits control frames, data information, and a sampling clock to the external sink and receives control frames and sample clock back.

The PDC that is defined as the sink receives control frames, sample clock and data information from an external source and transmits control frames and sample clock back.

CHANNEL FORMATS

All communication on a PIM type channel is via two formats: control frames and data words.

The control frame (Figure 10-32) contains a length bit, a 4-bit identifier, and an odd parity bit. Parity does not include the length bit which is always a one for control frames. The control frame identifier binary codes have a different meaning dependent upon whether sent by a source or sink PDC.

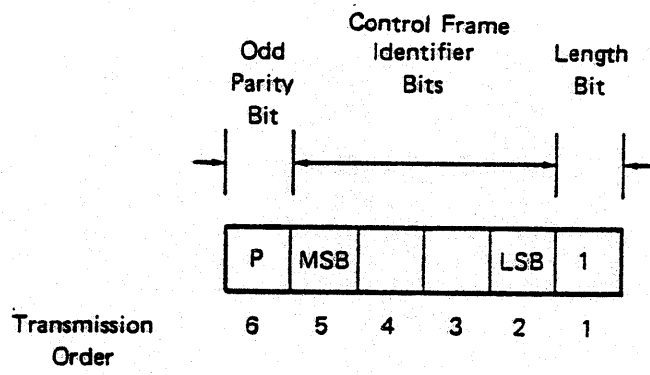


Figure 10-32. PIM Control Frame Format

- 1) Sequence Initialization and Control - The following control frames are used to start a sequence or control the data movement.

<u>Binary Code</u>	<u>Source to Sink</u>	<u>Sink to Source</u>
1110	System reset and load (SRL)	SRL acknowledge (SRLA)
1111	Channel reset (CRS)	CRS acknowledge (CRSA)
0000	Control interrupt word available (CIWA)	Request control interrupt word (RCIW)
0001	Normal interrupt word available (NIWA)	Request normal interrupt word (RNIW)
0010	Control word available (CWA)	Request control word (RCW)
0011	Data word available (DWA)	Request data word (RDW)

- 2) Synchronization - This control frame is used in word sequences as a means to delay a word transfer and still maintain proper transmission interval rates.

<u>Binary Code</u>	<u>Source to Sink</u>	<u>Sink to Source</u>
0101	WAIT	Repeat, sink busy (RSB)

- 3) Completion - This control frame is used to terminate word sequences in a normal manner.

<u>Binary Code</u>	<u>Source to Sink</u>	<u>Sink to Source</u>
0110	STOP	ENDS

- 4) Error Recovery - This control frame is used in any sequence to allow the channel to retransmit a reception.

<u>Binary Code</u>	<u>Source to Sink</u>	<u>Sink to Source</u>
0100	Repeat, error detected (RED)	Repeat, error detected (RED)

- 5) Sequence Violation - This control frame is used to notify a source or sink of a sequence violation which will terminate the active sequence. Sequence violations are defined as follows:

a) Three successive REDs or error receptions.

- b) An illegal time-out, no response within 3.2 microseconds of a transmission.
- c) An illegal sequence reception which uses a control frame in a sequence that does not follow the proper channel protocol.
- d) An illegal sink initialization in which the sink detects a control frame for a sequence which is not implemented in the sink.

<u>Binary Code</u>	<u>Source to Sink</u>	<u>Sink to Source</u>
0111	Sequence error (SEQE)	Sequence error (SEQE)

All other control frame binary codes are undefined and will be detected as sequence errors.

The data word, Figure 10-33, contains a length bit, 32 data bits, and an odd parity bit. Parity does not include the length bit which is always a zero for data words.

CHANNEL SEQUENCES

PIM channel communication is divided into three different sequences. Each sequence is a series of control frames and/or data words sent and received in a well-defined yet flexible protocol. All sequences start with a source-to-sink control frame that defines the type of sequence. The sink responds with an acknowledge control frame. The source and sink communication during the sequence continues on an alternating basis in half duplex until the sequence completes with the sink sending the final control frame.

The three types of sequences are as follows:

- 1) Immediate Sequence - This sequence consists of one control frame sent by the source and an acknowledge control frame sent by the sink. Two immediate sequences exist, one is an SRL back from the source and an SRLA back from the sink, and the second is a CRS back from the source and a CRSA back from the sink. An immediate sequence is used to inform the sink to take immediate action as defined by the control frame. A sink must respond to an immediate control frame anytime it is active regardless of its state with respect to input chain activity.
- 2) Word Sequences - This sequence consists of control frames and from 1 to 2048 32-bit words sent from source to sink. The purpose of the word sequences is to present a number of words to the sink which the sink shall buffer. There are two word sequences, one is control word and the second is data word.
- 3) Interrupt Word Sequences - This sequence consists of the proper control frames and one 32-bit interrupt word sent from the source to the sink. The purpose of an interrupt word sequence is to present a word to the sink which it must interrupt immediately upon completion of the sequence. The two interrupt word sequences are: first, a control interrupt word consisting of CIWA from the source,

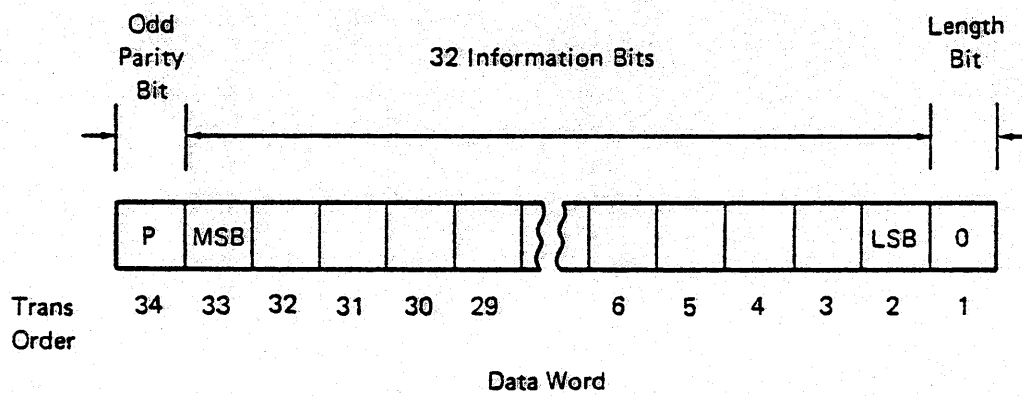


Figure 10-33. PIM Data Word Format

RCIW from the sink, CIW from the source, and ENDS from the sink; second, a normal interrupt word consisting of NIWA from the source, RNIW from the sink, NIW from the source, and ENDS from the sink.

CONTROL MEMORY DEFINITION

Control memory format and usage for the PIM I/O channel type is shown in Figure 10-34. Control memory locations 0, 1, and 2 are associated with the sink. Control memory locations 4, 5, and 6 are associated with the source.

Control memory locations 0 and 4 are termed the buffer control words (BCW). They have the same format as for the other channel types. Bits 15 and 14 are the transfer mode (TM) and must be 10 (word transfer) on the PIM channel type. Bits 11 through 0 are the buffer transfer count (BTC), the number of 16-bit words to be transferred, and is decremented by one for each 16-bit word transfer even though the words on the channel are 32-bits. For proper operation, the BTC must be set to an even number.

Control memory location 8 is termed the sink mode control word (Figure 10-35) and for normal sink operations it is all 0's.

Bit 7 enables internal wraparound between source and sink allowing for testing of the PIM when set.

Bit 6 disables sink transmitters when set.

Bit 5 generates sink parity error when set.

Bit 4 generates sink length error when set.

Bit 3 through 0 are not used.

Control memory location 9 is termed the source mode control word (Figure 10-36), and for normal source operations, bits 3 through 0 are used, bits 6 through 4 are zero's.

Bits 15 through 7 are not used.

Bit 6 disables source transmitters when set.

Bit 5 generates source parity error when set.

Bit 4 generates source length error when set.

Bits 3 through 0 are termed the source sequence control frame and are used to select the sequence to be activated by the source as follows:

<u>Bits</u>	<u>Source-to-Sink</u>	<u>Sequence</u>
3 0	<u>Transmission</u>	
0000	Control interrupt word available (CIWA)	} Interrupt word
0001	Normal interrupt word available (NIWA)	

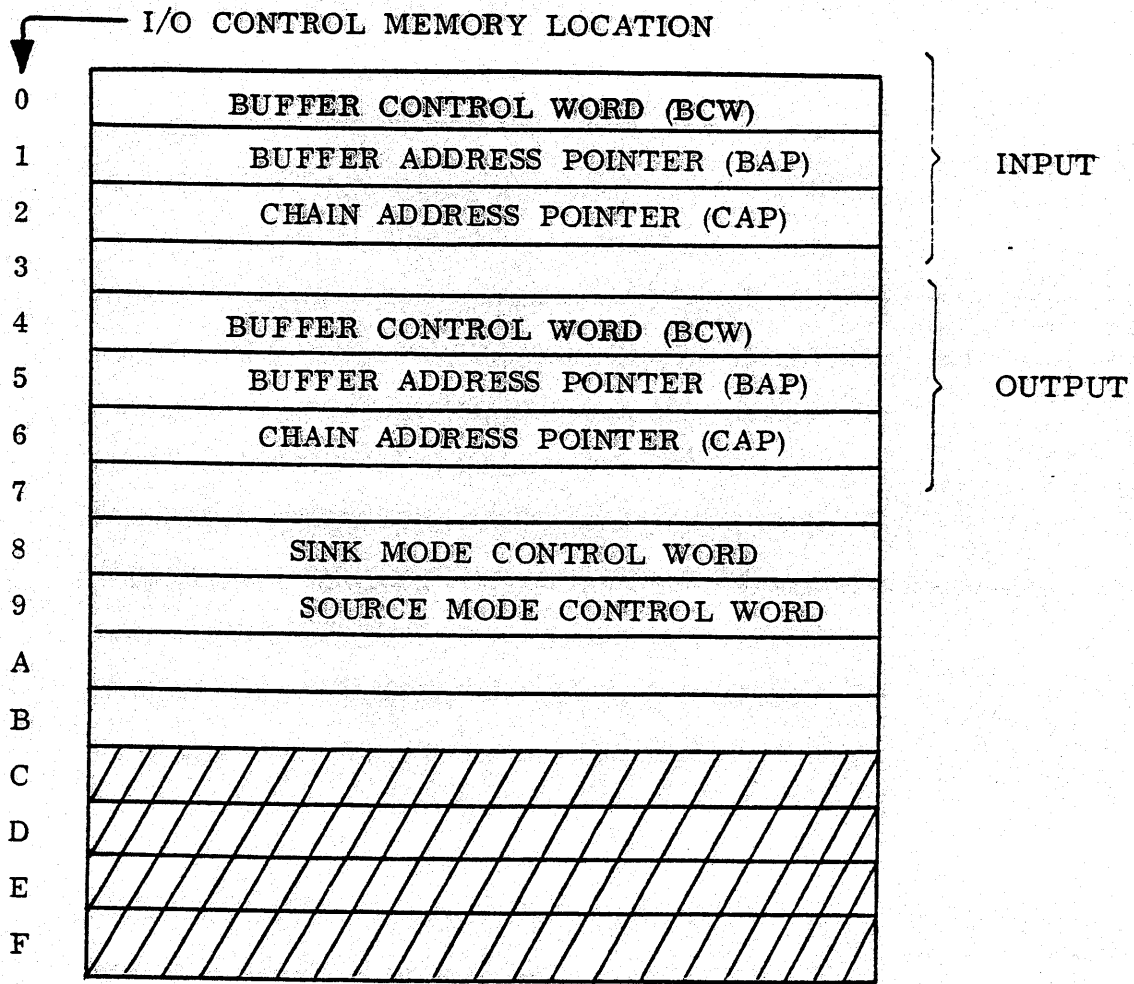


Figure 10-34. PIM Control Memory Map

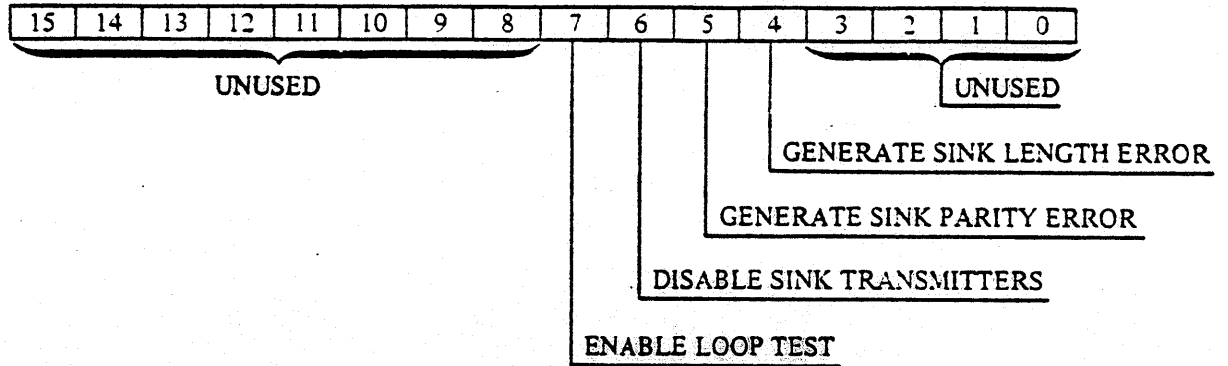


Figure 10-35. PIM Sink Mode Control Word Format

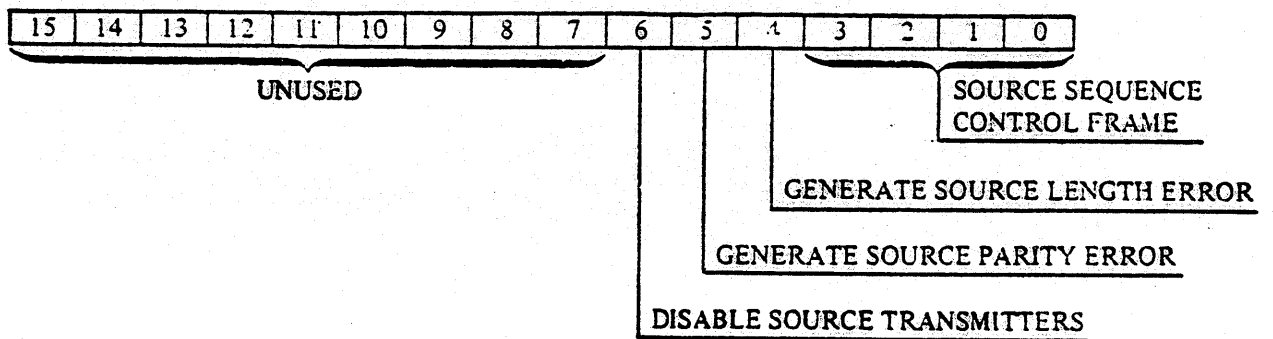


Figure 10-36. PIM Source Mode Control Word Format

<u>Bits</u>	<u>Source-to-Sink Transmission</u>	<u>Sequence</u>
3 0		
0010	Control word available (CWA)	} Word
0011	Data word available (DWA)	
1110	System reset and load (SRL)	} Immediate
1111	Channel reset (CRS)	

PIM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on a PIM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (E0 a m) (ACR m; CCR a,m) - The same as listed under Processor to I/O Channel Communication paragraph except that the a-field is not used in CCR instructions in a chain program, and the associated I/O channel is the one executing the I/O chain program.
- 2) Initiate Message (E2 a m) (IM a,y,m) - Load the control memory location specified by the m-field with the value y, and perform the operation as specified by the a field as follows:

Operation

a = XX00 Enable input word transfer (sink)
a = XX00 Initiate output word transfer (source)

The associated I/O channel is the one executing the I/O chain program.

- 3) Initiate Transfer (E3 a 0) (IO a,y) - Load control memory locations 0 and 1 (a-field equal to XX00) or 4 and 5 (a-field not equal to XX00) with the contents of main memory address y and y + 1, then perform the operation specified by the a field as follows:

Operation

a = XX00 Enable input word transfer
a = XX00 Initiate output word transfer

The associated I/O channel is the one executing the I/O chain program.

- 4) Load Control Memory (E6 0 m) (LCMK m,y) - Load the control memory location specified by the m-field with the value y. The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.

- 5) Load Control Memory (E7 0 m) (LCM m,y) - Load the control memory location specified by the m-field with the contents at main memory y. The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 6) Store Control Memory (EB 0 m) (SCM m,y) - Store the contents of the control memory location specified by the m-field at main memory address y. The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 7) Halt/Interrupt (EC a 0) (HCR; IPR) - The same as listed under DIM I/O Channel Instructions paragraph.
- 8) Set/Clear Flag (EF a 0) (ZF y; SF y) - The same as listed under SIM I/O Channel Instructions paragraph.
- 9) Conditional Jump (F2- a 0) (SJMCa,y) - Load control memory location 2 (if input chain) or control memory location 6 (if output chain) with the contents of y, if bit 15 of status word 0 is set. The a-field selects the test condition which drives bit 15 as follows:

<u>a</u>	<u>Operation</u>
XX00	Unconditional jump
XX01	Jump if the sink BTC is not equal to 0 after an input transfer.
XX10	Jump if the input transfer is not complete
XX11	Jump if the source receives an ENDS before its output buffer is empty.

The associated I/O channel is the one executing the I/O chain program.

- 10) Bit Jump (FD 0 m) (BJ m,y) - This is a conditional jump instruction. If the bit in control memory location 3 specified by the m-field is a logic 1, then the value y is loaded into control memory location 2 for input chain or control location 6 if output chain. If the bit is a logic 0, the next I/O chain instruction in sequence is executed.
- 11) Exchange Control Memory (FE 0 m) (XCM m,y) - Store the contents of control memory location specified by the m-field in main memory location y. Then load the control memory location specified by the m-field with the contents of main memory location y ⊕ 1.

NOTE:

This becomes a branch or jump instruction if the m-field equals 2 or 6. The old contents of control memory location 2 or 6 is saved so that a return is possible when the entered routine is exited.

- 12) Set/Clear Discretes (F8 0 m) (CSIR m) - Initiate a sink or a source activity as specified by the m-field as follows:

<u>m</u>	<u>Operation</u>
XX00	Halt sink
XX01	Halt source
XX10	Initiate SRL or CRS (source only)
XX11	Clear interrupt halt (sink only)

The a-field is not used and the associated I/O channel is the one executing the I/O chain program.

- 13) Store Status (FB 0 m) CSST y,m) - Store the channel status as specified by the m-field in main memory location y as follows:

<u>m</u>	<u>Status Word Value</u>
X000	Hardware status word 0 (Figure 10-37)
X001	Hardware status word 1 (Figure 10-38)
X010	Hardware status word 2 (Figure 10-39)
X100	Interrupt word (MS 16 bits)
X101	Interrupt word (LS 16 bits)

The a-field is not used and the associated I/O channel is the one executing the I/O chain program.

PIM INTERRUPT HANDLING

The PIM I/O channel type is capable of generating the ERI, EII, OCI, and ICI Class III interrupts.

The PIM hardware shall generate an ERI interrupt whenever the source or sink detects a sequence error during a sequence or receives a SEQE control frame. The PIM hardware status word 2 (Figure 10-39) will indicate the type of error that caused the interrupt.

If the ERI interrupt is locked at the processor by SRI, bit 11 of hardware status word 0 is set and will not be cleared until the interrupt is processed.

The PIM hardware shall generate an EII interrupt provided external interrupts are enabled and Class III, priorities 2, 3, and 4 are enabled for the channel when the sink receives either a CRS or SRL immediate sequence control frame. Bit 12 or 13 of status word 1 will be set and the word stored in memory if the EII was generated.

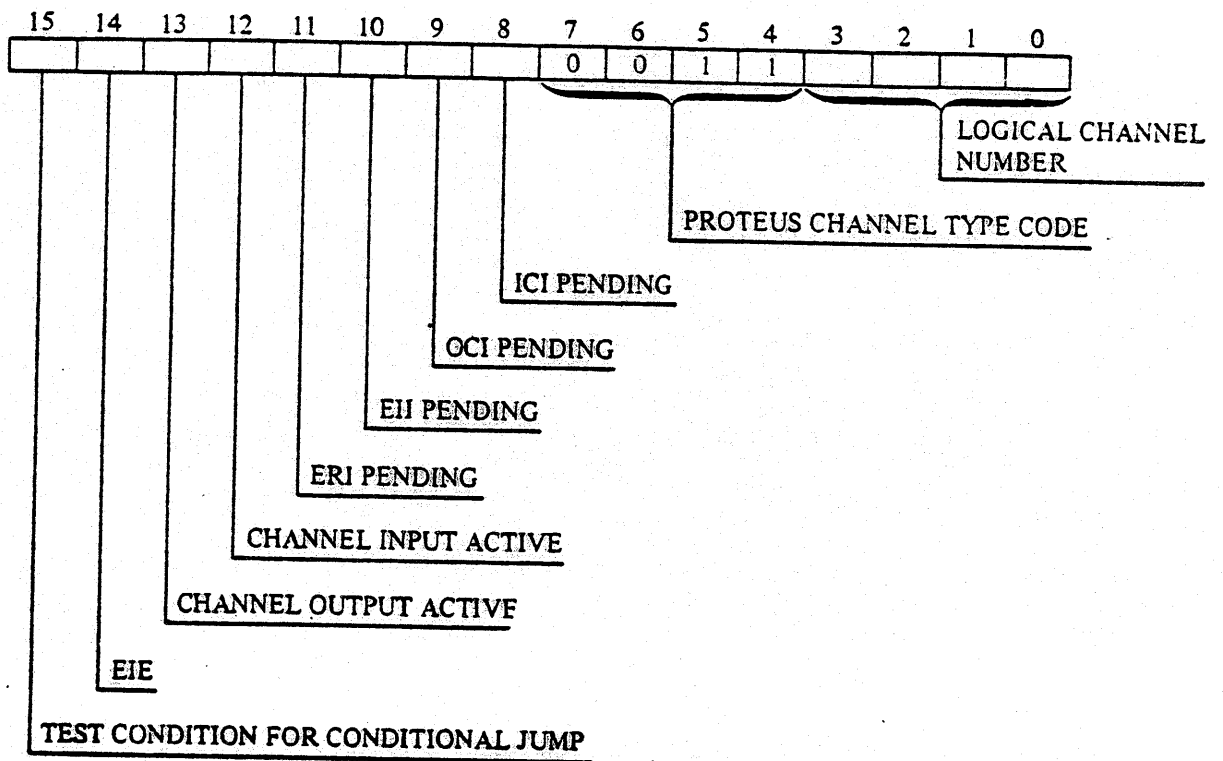


Figure 10-37. PIM Channel Hardware Status Word 0 Format

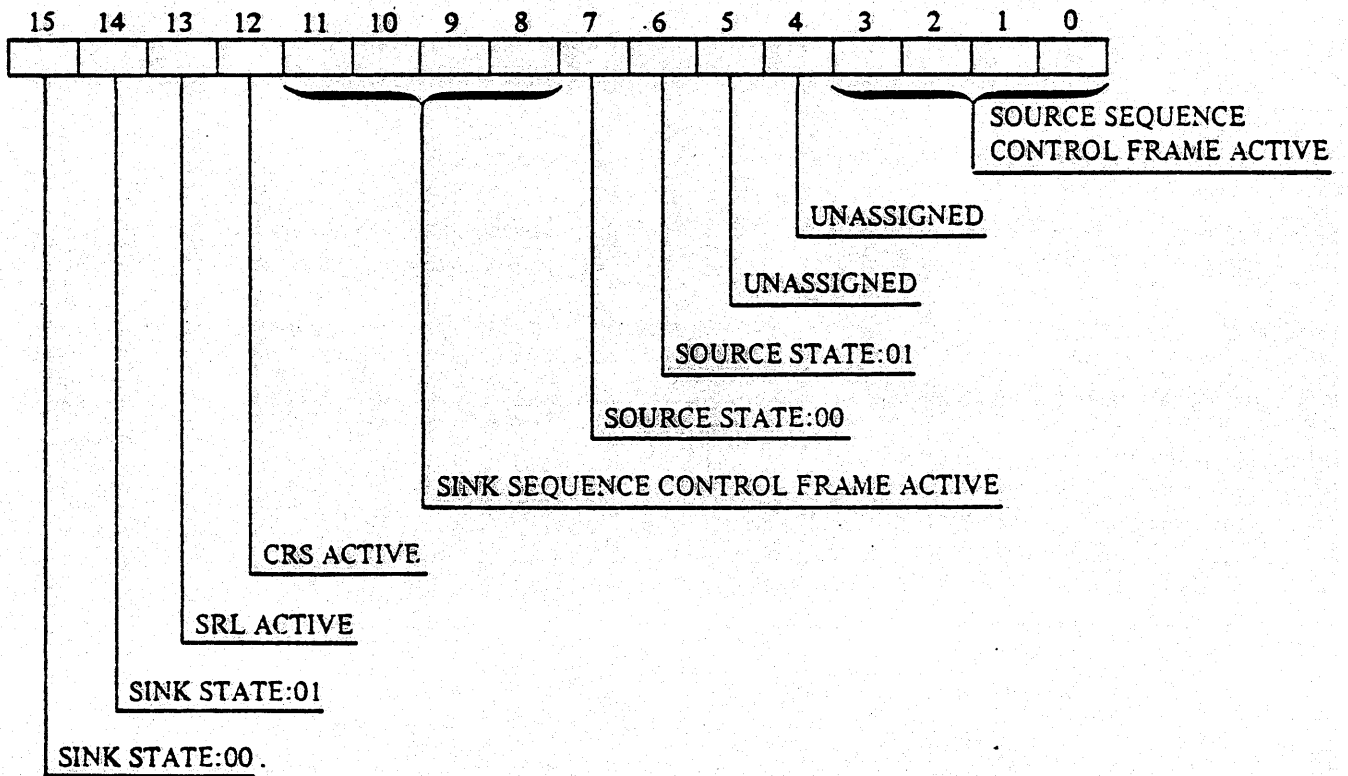


Figure 10-38. PIM Channel Hardware Status Word 1 Format

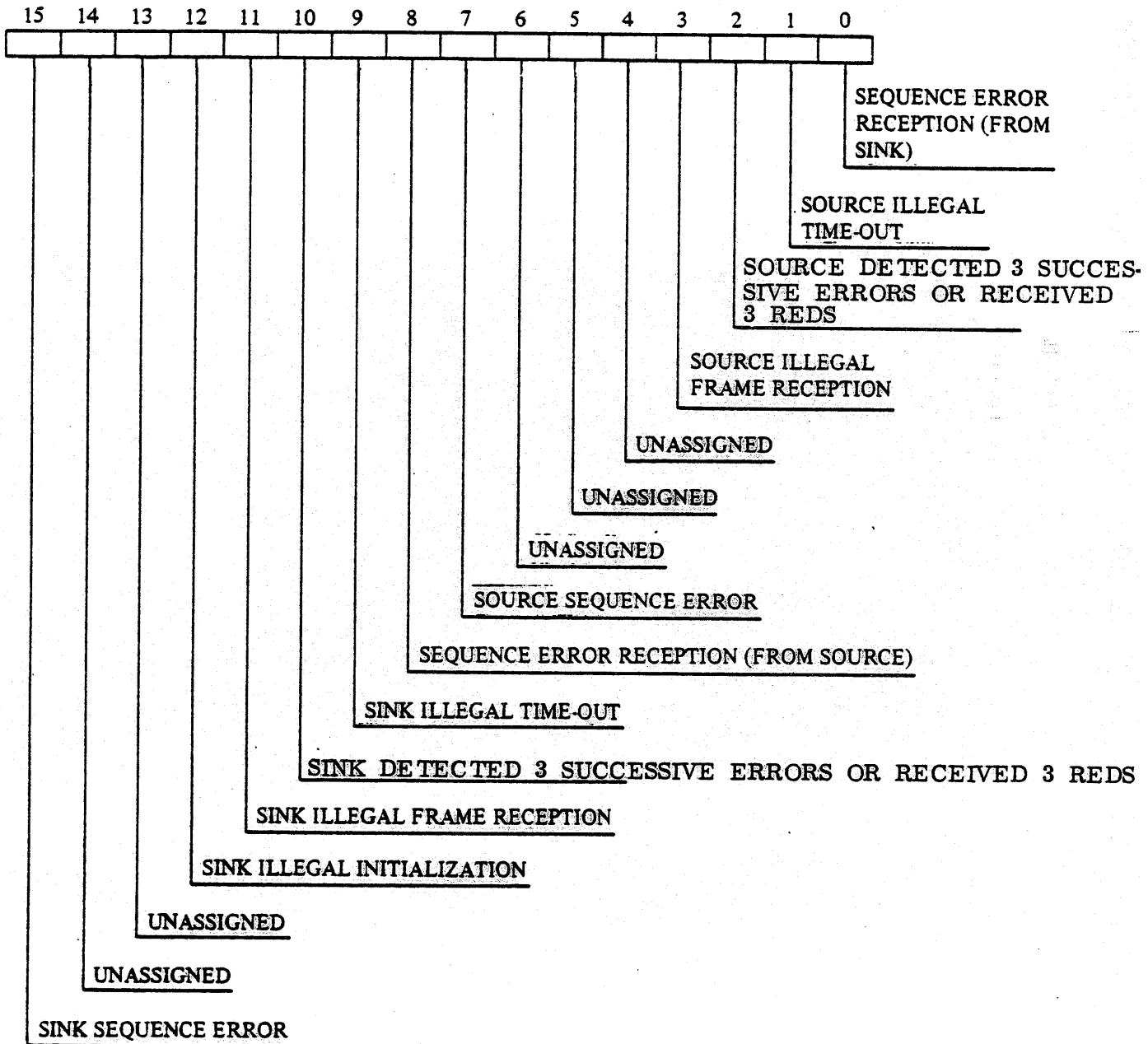


Figure 10-39. PIM Channel Hardware Status Word 2 Format

The PIM hardware shall respond to an immediate sequence control frame and set the correct bit in status word 1 even when the EII interrupt is disabled. When the enabling conditions are enabled, status word 1 is stored in memory, and the EII interrupt is generated.

When the PIM sink receives either interrupt word sequence control frame (CIWA or NIWA) and external interrupts are enabled, it places the control frame received in status word 1, accepts the 32-bit interrupt word, and responds to the source. If Class III, priorities 2, 3, and 4 are disabled the sink sets EII pending bit in hardware status word 0 and holds the interrupt pending at the channel level. If Class III, priorities 2, 3, 4 are enabled the hardware status word 1 is stored in main memory, the EII interrupt is generated, and the input chain activity, if active, is stopped.

The interrupt handling firmware disables the external interrupts on the channel. This way the sink cannot generate another EII interrupt until allowed to do so by the software.

The PIM shall generate an OCI interrupt when an IPR instruction is executed in the output chain program.

The PIM shall generate an ICI interrupt when an IPR instruction is executed in the input chain program.

PROGRAMMING CONSIDERATIONS

The PIM I/O channel type allows an input chain and an output chain to operate simultaneously and independently of each other.

Source sequences:

- 1) Immediate Sequence - Bits 3 through 0 of the source mode control word in the control memory location 9 are loaded with the immediate sequence control frame. Execution of set/clear discrettes (CSIR) instruction in the output chain with the m-field equal to XX10 (binary) initiates the sequence and stops chaining activity. The sequence will terminate when the source receives a sink response (SRLA or CRSA), and the output chain activity will continue. A sequence or protocol violation will set ERI, status word 2 will contain the type of violation, and the output chain activity will not be reinitiated.
- 2) Word Sequence - Bits 3 through 0 of the source mode control word in control memory location 9 are loaded with the current control frame. Execution of initiate transfer (IO) instruction in the output chain program with the m-field not equal to XX00 initiates the sequence and stops the chain activity. If a control or normal interrupt sequence is initiated, one 32-bit word will be transmitted to the external sink and the sequence terminates as follows:
 - a) A normal termination will occur if the sink responds to the 32-bit interrupt word with ENDS. Output chain activity is reinitiated.

- b) An abnormal termination will occur if a protocol or data error occurs, status word 2 will indicate the error, ERI will be generated, and output chain activity is not reinitiated.

If a control or data word sequence is initiated, 32-bit words will be transmitted to the external sink until one of the following terminations occur:

- a) A normal termination will occur if the buffer transfer count was zero when an ENDS was received from the external sink. This would indicate equal word counts in the source and sink controllers.
- b) A normal termination will occur if the buffer transfer count was zero and the external sink requested another word transmission. In this case the source would transmit a STOP and receive an ENDS.
- c) A normal termination will occur if the buffer transfer count was not equal to zero and the external sink sends an ENDS. In this case the conditional jump instruction test condition, a-field equal to XX11, is a logical 1.

In all three cases, the output chain activity will be reinitiated.

- d) Execution of set/clear discrettes (CSIR) instruction with the m-field equal to XX01, halts the source word sequence activity. The source will transmit a STOP to the sink, and the output chain activity will not be reinitiated.
- e) An abnormal termination will occur during a word sequence if protocol or data error occurs, status word 2 will contain the type of error, ERI will be generated, and output chain activity is not reinitiated.

Sink sequences:

- 1) Immediate Sequence - When an immediate sequence control frame (SRL or CRS) is detected, the sink transmits an acknowledge (SRLA or CRSA), sets status bit 13 or 12 of status word 1, generates EII if EIE is set, and Class III interrupts enabled. The status word 1 is stored in main memory and input chain activity halted if active. If both sequences are recognized before software recognizes the EII interrupt, both status bits will be set. Recognition of EII clears EIE and prevents generation of further EIIs. Clearing the immediate sequence status bits and setting EIE can be accomplished by executing a CCR a,12 instruction.
- 2) Word Sequence - When a control or normal interrupt word control frame (CIWA or NIWA)n is detected, the sink transmits an acknowledge (RCIW or RNIW), accepts the interrupt data word, generates EII (if Class III interrupts enabled and EIE is set), and halts the input chain activity if active. When the EII is generated, status word 1 has been stored in main memory by the firmware. The 32-bit

interrupt word is read by executing store status (CSST or SST) instruction with the m-field equal to X100 for the 16 MSB and X101 for the 16 LSB. In order for the sink to generate another EII in response to an interrupt control frame or immediate control frame, the EIE must be set again since recognition of the EII clears EIE.

Control word or data word sequences require that the sink has been enabled before it will activate the sequence.

If a control word (CWA) or data word (DWA) sequence control frame is detected and the sink is not enabled (i.e., an input transfer for IO instruction executed in an input chain program), the sink will respond with a RSB. If in the input chain an initiate transfer instruction with the m-field equal to XX00 has been executed, the sink responds with a request control word (RCW) or a request data word (RDW). The data transfer will continue until one of the following conditions occur:

- a) If the source sends a STOP, the sink responds with an ENDS, and the input chain activity is reinitiated.
- b) If the input buffer transfer count is zero, the sink will respond with an ENDS at the next transmission interval, and the input chain activity is reinitiated.
- c) Execution of set/clear discrettes (CSIR) command instruction with m-field equal to XX00 halts the sink word sequence activity. The sink will respond with an ENDS at the next transmission interval, and the input chain activity will not be reinitiated.

The sink allows blocks of data to be received from the source, terminating the source transmission when the sink buffer transfer count equals zero. After this has happened, one word will be accepted into the input data buffer with further data transmissions acknowledged with a RSB response from the sink. If a control or normal interrupt word is received and EIE is set, the word is accepted, and the data word presently in the input buffer will be lost.

After a word sequence has terminated and prior to the next initiate transfer instruction, the chain program may store status word 1 in main memory so software can detect which word sequence was active.

If a protocol or data error occurs during a word sequence, ERI will be generated, and the input chain activity is not reinitiated. Status word 2 will contain the type of error.

PIC/POC/SOC MODULE

The PIC/POC/SOC module (PPSM) has the following general characteristics.

PIC/POC

- Full duplex, 32-bit parallel input and 32-bit parallel output channels
- Request-acknowledge type control logic (external device controls the data rate)
- Internal and external wraparound test capacity
- External halt available for POC (an EII is generated by this signal).

SOC

- Serial output with a bit stream of multiples of 16 bits in length
- Data and control signals are differential or single ended
- Data rate of one megabit per second or 200 kilobits per second, selectable under program control
- Internal test capability
- External suspend line available to regulate data transmission.

All input to the AN/AYK-14(V) is via the parallel input channel (PIC), and output is via the parallel output channel (POC) or the serial output channel (SOC). I/O transfers are initiated by standard AN/AYK-14(V) I/O chains.

MESSAGE FORMATS

The PIC/POC 32-bit words are transferred to or from the AN/AYK-14(V) in two 16-bit halves, with the 16 most significant bits first.

The SOC bit streams outputted can consist of up to 4096 16-bit words. The total number of words to be transmitted is specified by the buffer transfer count (BTC) in the buffer control word (BCW).

The control of data transfers to and from the PIC/POC and to the SOC is accomplished through an exchange of commands and data at the controller - IOBUS interface. These commands operate in conjunction with the generation of events from the module to the controller. The IOBUS provides the communication path for commands and data from the controller to the module as well as a path for status and data from the module to the controller.

The command and function control word (format shown in Figure 10-40) provides the means to establish the type of communications to be performed as well as operations to be executed.

CONTROL MEMORY DEFINITION

Control memory format and usage for the PPSM I/O channel type is shown in Figure 10-41 for PIC/POC and in Figure 10-42 for SOC.

PIC/POC control memory locations 0, 1, and 2 are associated with the PIC (input activity) and locations 4, 5, and 6 with the POC (output activity).

PIC/POC control memory locations 0 and 4 are the input and output buffer control word, (BCW) and define the transmission mode and buffer transfer count (BTC) for data input and output transfers. The format for the BCWs is depicted in Figure 10-43. The format for the mode control word (MCW) is shown in Figure 10-44. Note that 32-bit (double-length transfer) mode should be used with the PIC/POC. Thus, the buffer transfer count is equal to the number of 32-bit words to transfer. A count of zero implies the maximum transfer count of 4096.

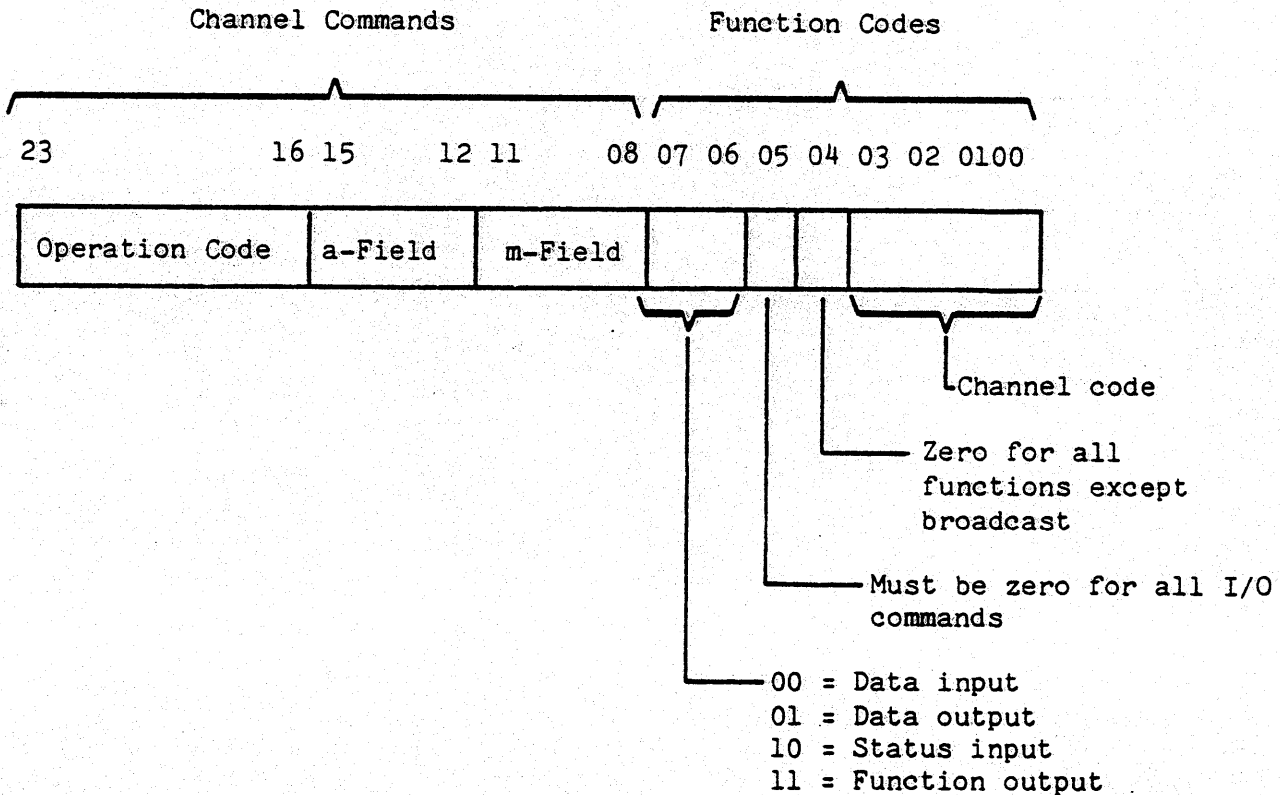


Figure 10-40. Command and Function Code Word Format

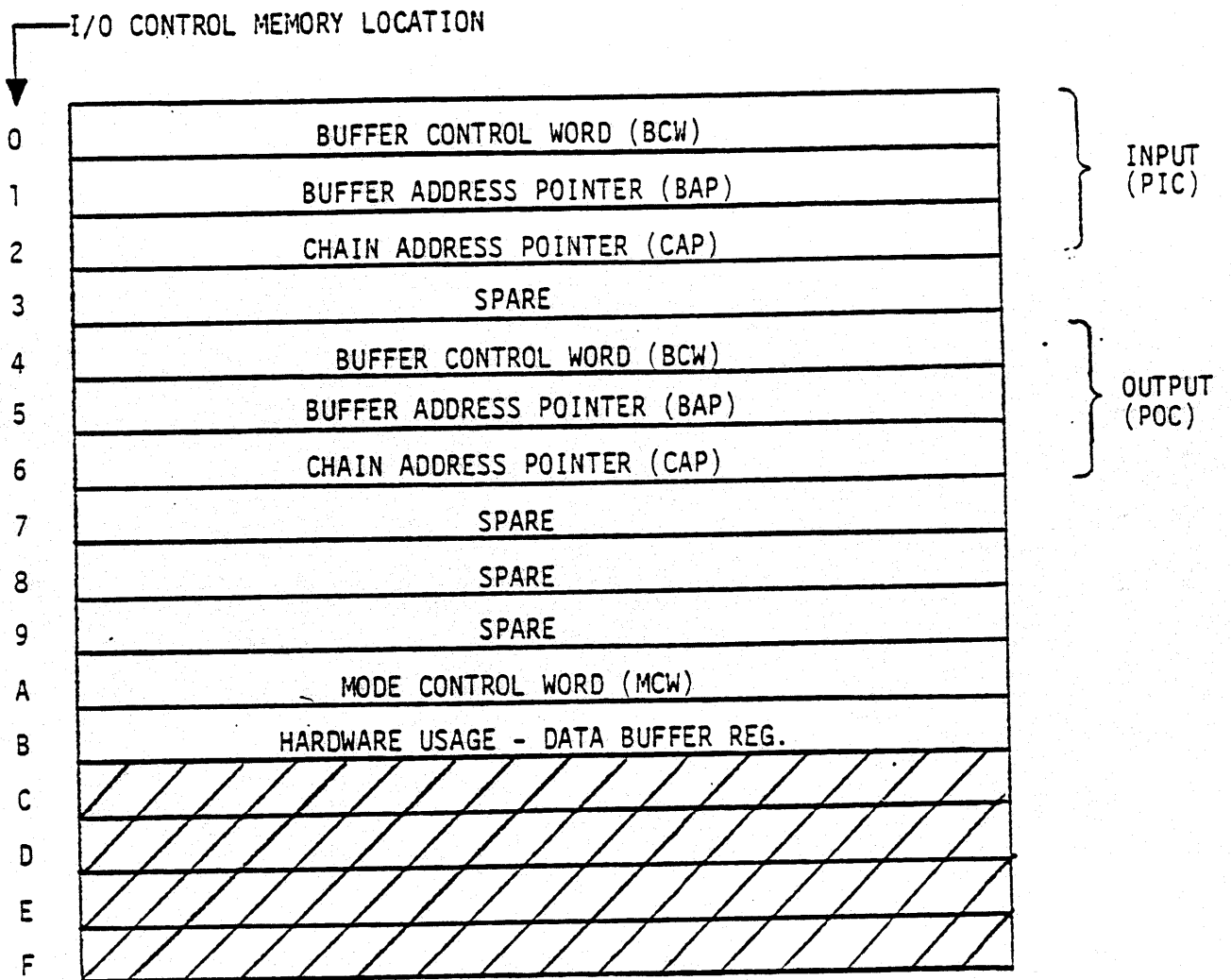


Figure 10-41. PIC/POC Control Memory Definition Map

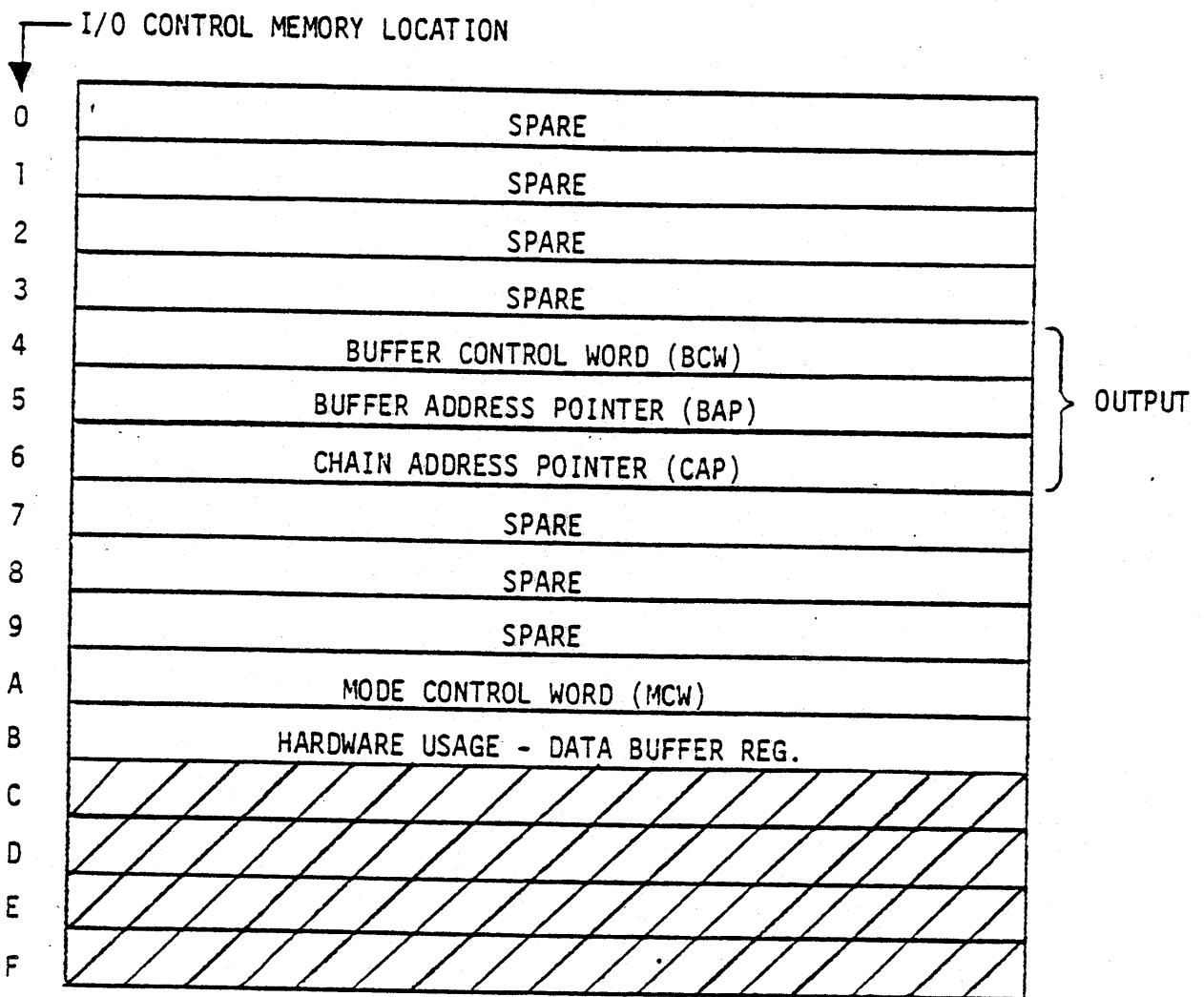


Figure 10-42. SOC Control Memory Definition Map

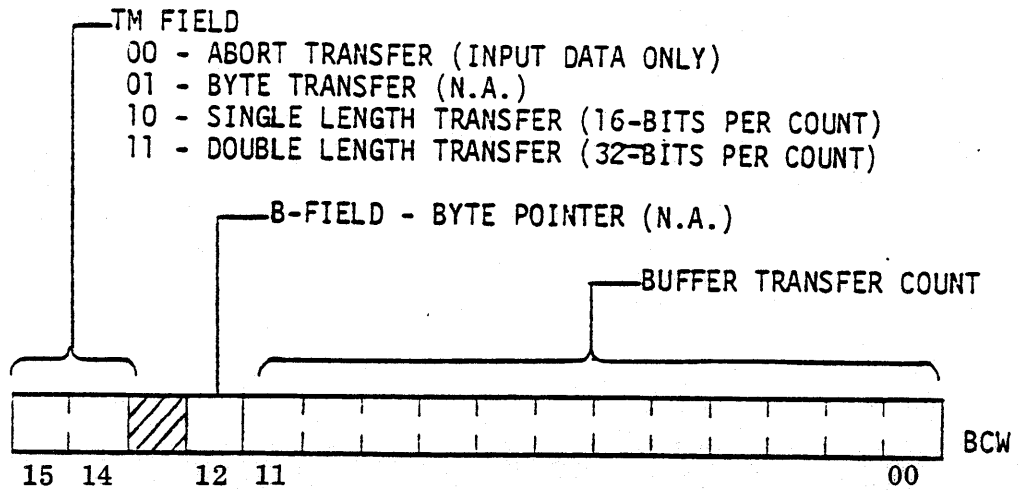


Figure 10-43. PIC/POC Buffer Control Word

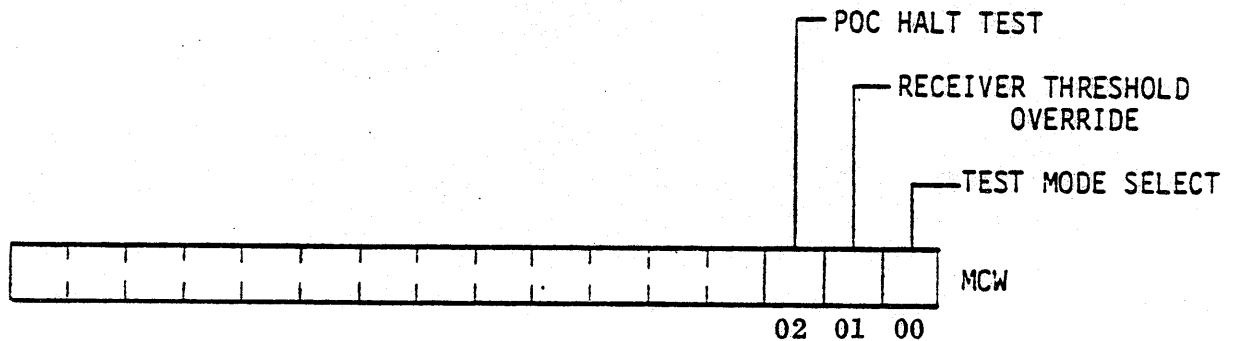


Figure 10-44. PIC/POC Buffer Control Word

SOC control memory location 4 is the buffer control word (BCW) and defines the transmission mode and buffer transfer count (BTC) for SOC data transfers. Transmission mode 2 (16-bit mode) must be used with the SOC I/O channel type. The buffer transfer count is the number of words to be output with 16 bits per word. A count of zero implies the maximum transfer count of 4096. The BCW format is shown in Figure 10-45. The format for the mode control word is shown in Figure 10-46. Control memory location A is the SOC mode control word (MCW). This location defines the data bit rate and various test mode bits.

PPSM I/O CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on the PPSM I/O channel type. Any that are not included are legal but perform a no operation.

- 1) Channel Control (E0 m) (ACR m; CCR m) - The same as listed under Processor to I/O Channel Communication paragraph with the following exceptions.

The a-field is not used and the associated I/O channel is the one executing the I/O channel program.

m-field equal to	0001	}	Illegal
	0010		
	0011		
	1001		
	1010		
	1011		

- 2) Initiate Message (E2 a m) (IM a,y,m) - Load control memory location specified by the m-field with the value y; then, perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
0000	Initiate input (PIC) data transfer using the BCW and BAP in control memory locations 0 and 1, respectively.
0001	Initiate output (POC or SOC) data transfer using the BCW and BAP in control memory locations 4 and 5, respectively.

The associated I/O channel is the one executing the I/O channel program.

- 3) Initiate Transfer (E3 a 0) (IO a,y) - Load control memory locations 0 and 1 (PIC, a-field equal to XX00) or 4 and 5 (POC or SOC, a-field not equal to XX00) with the contents of main memory addresses y and y + 1 respectively. Perform the operation specified by the a-field as defined in the Initiate Message instruction. The associated I/O channel is the one executing the I/O channel program.

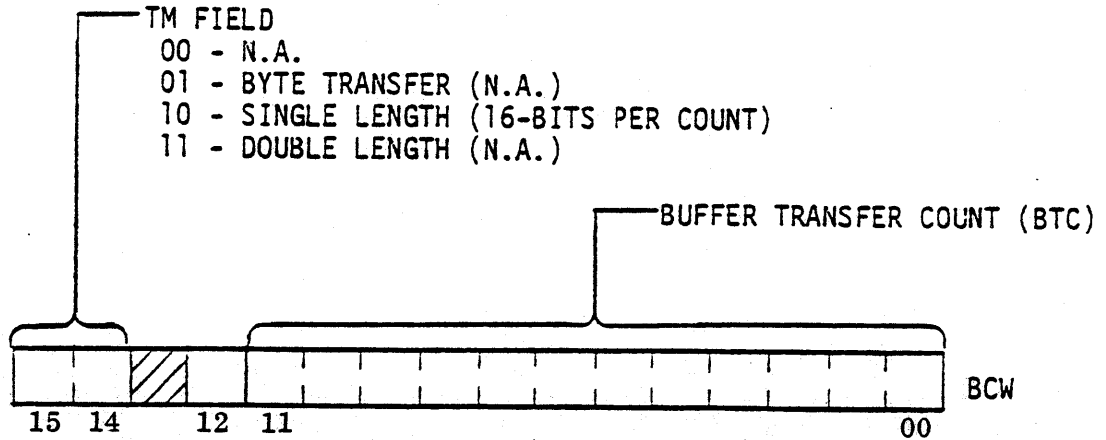


Figure 10-45. SOC Buffer Control Word

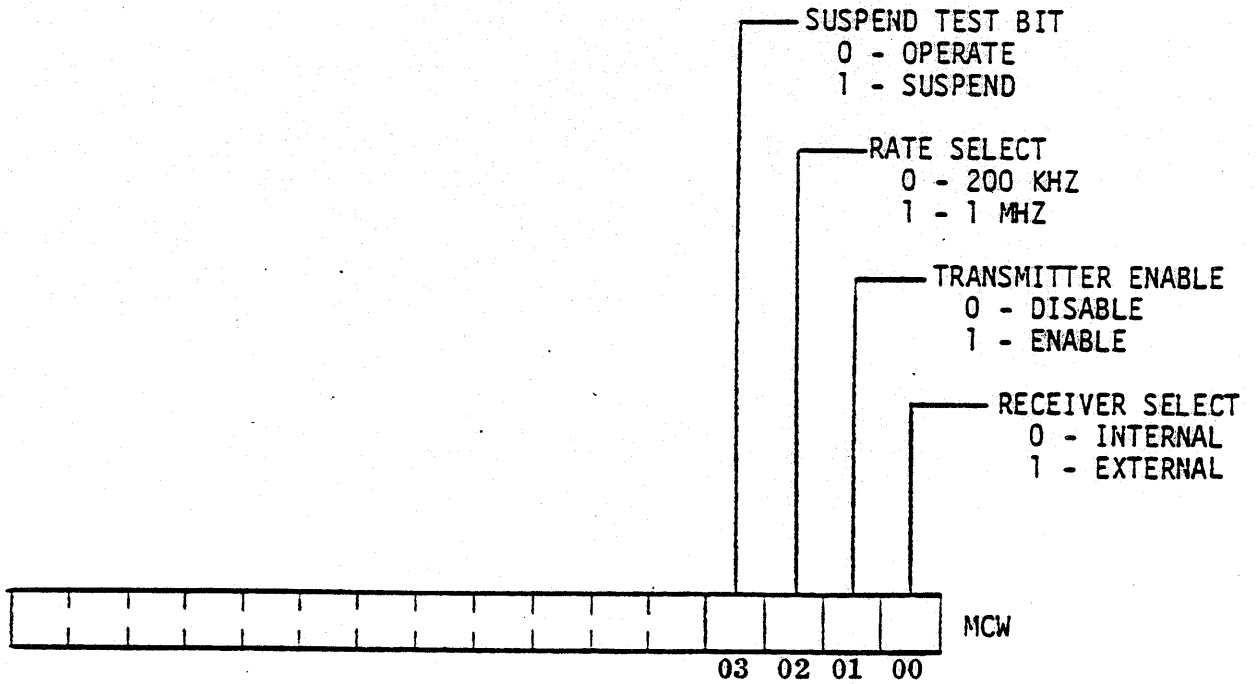


Figure 10-46. SOC Mode Control Word

- 4) Load Control Memory (E6 0 m) (LCMK m,y) - Load the control memory location specified by the m-field with the value y. The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 5) Load Control Memory (E7 0 m) (LCM m,y) - Load the control memory location specified by the m-field with the contents of main memory address y. The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 6) Store Control Memory (EB 0 m) (SCM m,y) - Store the contents of control memory location specified by the m-field at main memory address y. The a-field is not used and the associated I/O channel is the one executing the I/O channel program.
- 7) Halt/Interrupt (EC a 0) (HCR a; IPRa) - Perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Halt I/O chaining activity.
XXX1	Generate Class III, priority 3, OCI for output chain and ICI for input chain.

The m-field is not used, and the associated I/O channel is the one executing the I/O channel program.

- 8) Set/Clear Flag (EF a 0) (ZFa,y; SFa,y) - Set or clear the two most significant bits (flag) of the main memory value at address y as specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	Clear flag
XXX1	Set flag

The m-field is not used and the associated I/O channel is the one executing the I/O channel program.

- 9) SOC Conditional Jump (F2 0 0) (SIMC 0,y) - This is an unconditional jump for the SOC I/O channel type. When executed, this instruction loads control memory location 6 (CAP) with the value y.
- The a-field is not used, and the associated I/O channel is the one executing the I/O channel program.
- 10) PIC/POC Conditional Jump (F2 a 0) (SJMC a,y) - Load control memory location 6 (CAP) with the value y if bit 15 (MSB) of status word zero is set (logic 1). This instruction can be used in an input chain (PIC), but since bit 15 (MSB) of status word zero is cleared only by a POC data transfer, its use in an input chain may not be meaningful. If the a-field is even, this is an unconditional jump.

The m-field is not used, and the associated I/O channel is the one executing the I/O channel activity.

- 11) Store Status (FB 0 m) (CSST m,y) - Store the channel status as specified by the m-field in main memory location y as follows:

<u>m</u>	<u>Status Word Value</u>
XX00	PIC/POC or SOC channel status 0 (Figure 10-47)
XX01	PIC/POC or SOC channel status 1 (Figure 10-48)
XX1X	SOC input register. This register contains the most recent word shifted out. It is cleared when read as status.

The a-field is not used, and the associated I/O channel is the one executing the I/O channel activity.

- 12) Bit Jump (FD 0 m) (BJ m,y) - This is a conditional jump instruction. If the bit in control memory location 3 specified by the m-field is a logic 1, then the value y is loaded into control memory location 2 (input chain PIC) or control memory location 6 (output chain POC or SOC). If the bit is a logic 0, the next I/O channel instruction in sequence is executed.
- 13) Exchange Control Memory (FE 0 m) (CXCM m,y) - Store the contents of control memory location specified by the m-field in main memory location y. Then load the control memory location specified by the m-field with the contents of main memory location y + 1.

NOTE:

This becomes a branch or jump instruction if the m-field equals 2 (PIC) or 6 (POC or SOC). The old contents of control memory location 2 or 6 are saved so that a return is possible when the entered routine is exited.

PIC/POC INTERRUPT HANDLING

The PIC/POC can generate three Class III interrupt types, including EII, OCI, and ICI. All three can be masked by the single Class III mask bit associated with the given channel. This bit is set or cleared by execution of the appropriate activity control channel/ control (ACR/CCR) instruction, from either a chain program or via the command cell (that is, execution of an IOCR instruction). When the associated Class III mask bit is cleared, EII, OCI, and ICI interrupts are disabled but queued one level (i.e., will remain pending until the mask bit is set). When a particular interrupt is generated and enabled, recognition of that interrupt by the AN/AYK-14(V) processor depends on the status of other higher priority interrupts and on bit 1 of status word 1 in the processor.

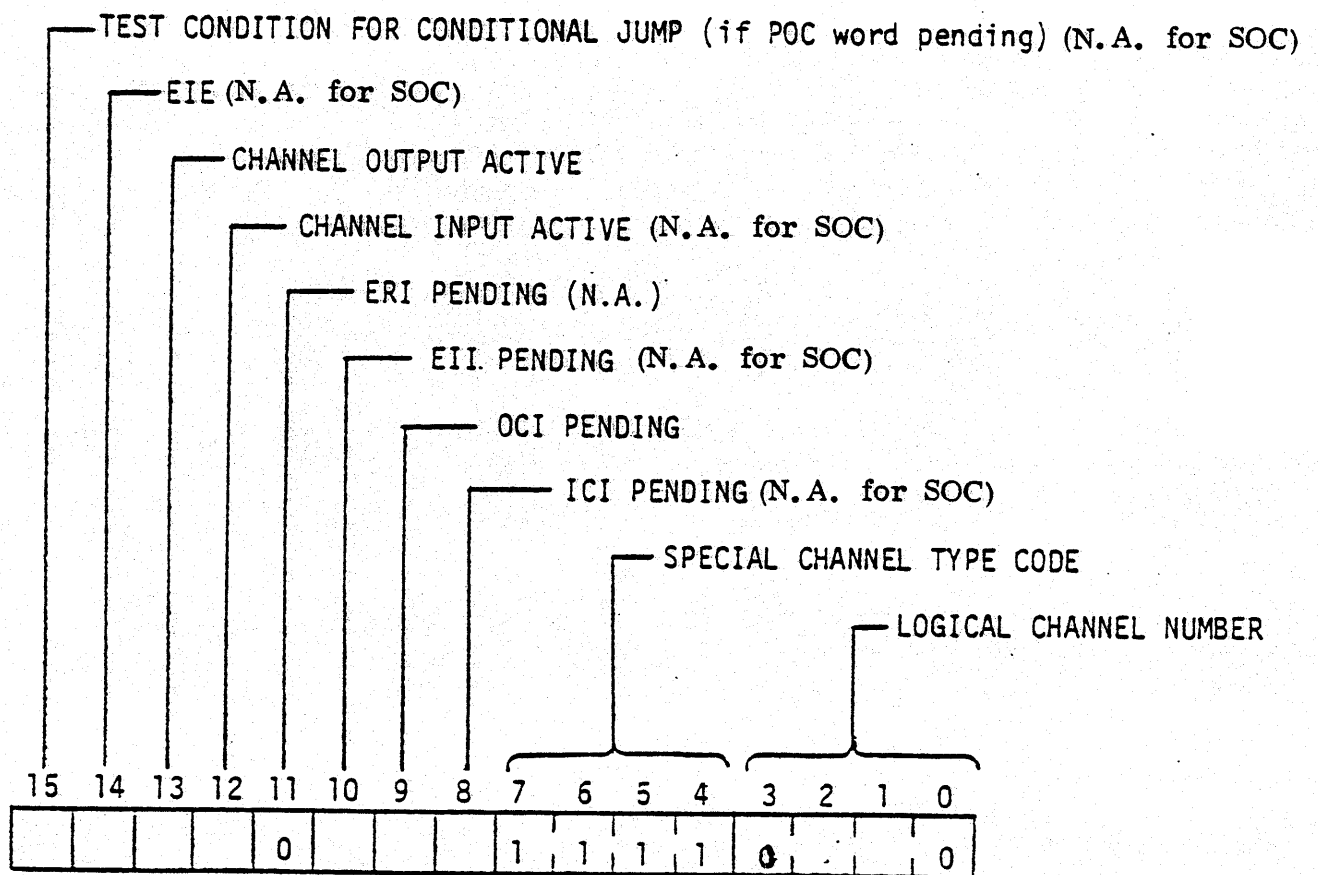


Figure 10-47. PIC/POC or SOC I/O Channel Status Word 0 Format

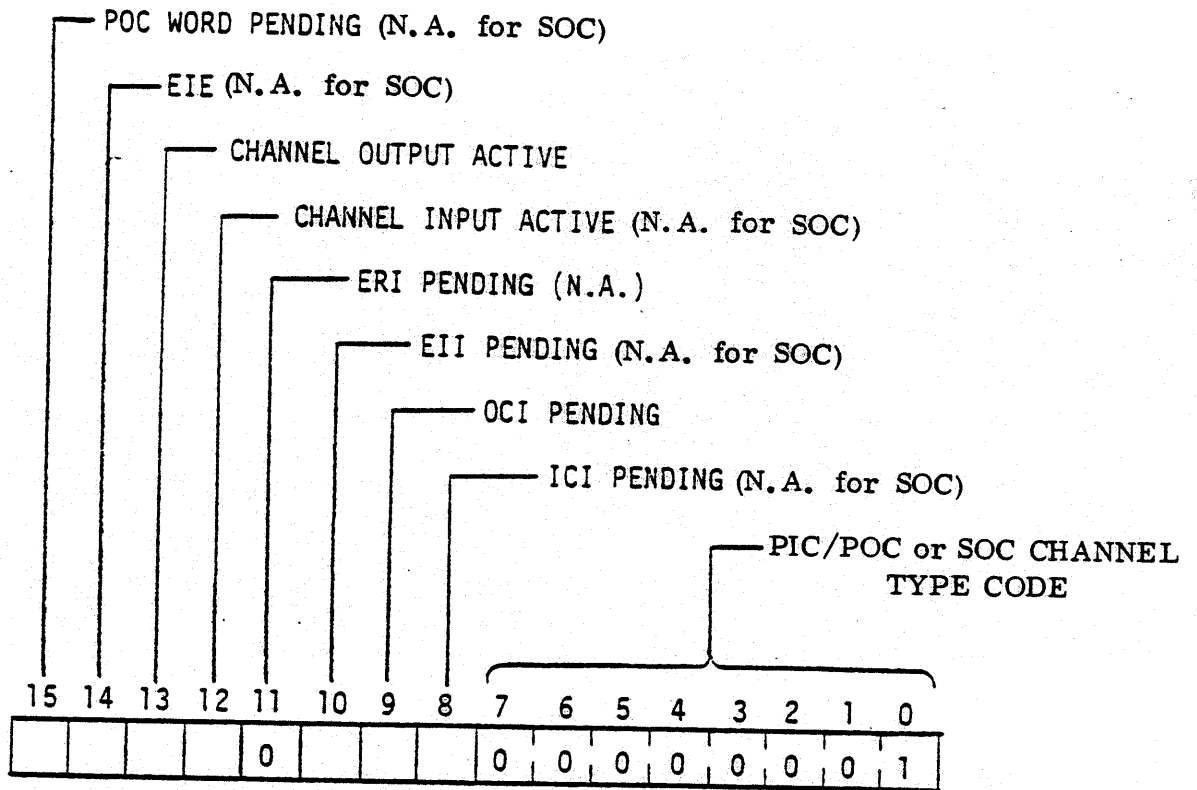


Figure 10-48. PIC/POC or SOC I/O Channel Status Word 1 Format

PIC/POC Class III interrupts are generated by the following conditions.

OCI - Execution of IPR instruction in an output (POC) chain

ICI - Execution of an IPR instruction in an input (PIC) chain

EII - If the POC channel external halt signal is set during or at the start of an output (POC) data transfer. Note that EIE must be set for an EII to be generated

OCI and ICI are masked only by the Class III mask, whereas the EII is also disabled if the EIE is clear. If disabled by the EIE, the external interrupt will not be generated and is not queued whether or not the Class III mask bit is set. When the POC EII is generated, the PIC/POC EIE is automatically cleared until reset under software control.

SOC INTERRUPT HANDLING

The SOC can generate the OCI Class III interrupt type. It can be masked by the single Class III mask bit associated with the given channel. This bit is set or cleared by execution of the appropriate activity control/channel control (ACC/CCR) instruction, from either a chain program or via the command cell (i.e., execution of an IOCR instruction). When the associated Class III mask bit is cleared, the SOC OCI type interrupt is disabled but queued one level (i.e., will remain pending until the mask bit is set). The SOC OCI type interrupt is generated by execution of an IPR instruction in an output (SOC) chain. When the interrupt is generated and enabled, its recognition by the AN/AYK-14(V) processor depends on the status of other higher priority interrupts, and on bit 1 of status word 1 in the processor.

DISCRETE INPUT/OUTPUT MODULE

The discrete input/output module (DIOM) has the following general characteristics.

- 144 output discretetes
- 48 input discretetes (used as inputs or interrupts, not both)
- All 48 inputs are individually prioritized in a fixed sequence for use as interrupts
- Interrupts are individually masked
- All outputs are internally wraparound testable, including the transmitters
- Internal testing of input scan logic, external testing of input receivers.
- Full duplex I/O channel capable of initiating both input and output chain which may be active simultaneously.

WORD FORMATS

The DIOM I/O channel provides the following three basic types of capabilities.

- Output discretetes
- Input discretetes
- Interrupt discretetes

Tables 10-2 and 10-3 define each discrete. Definitions are given for 16 different input discrete words and 13 different output discrete words. Output discrete words 0 through 8 define all 144 output discrettes. Input discrete words 0 through 8 provide capability to read the actual values of the output discrettes for internal wraparound testing. Input discrete words 9 through 11 define the 48 inputs when they are used directly. Interrupt discrete words 12 through 14 are the logic 1 captured values of the 48 inputs when the inputs are used as interrupts. The corresponding bits in interrupt mask words 9 through 11 define a mask bit for each of 48 possible interrupts. The highest priority interrupt is interrupt discrete 00. Interrupt discrete word 15 and interrupt mask word 12 are useful in internal wrap-around testing to provide capability to more fully test the DIOM logic.

CONTROL MEMORY DEFINITION

The DIOM control memory is diagrammed in Figure 10-49. Locations 0, 1, 2, and 8 are associated with input data transfers while locations 4, 5, 6, 9, and B are associated with output data transfer.

Control memory locations 0 and 4 are termed the buffer control word (BCW) which define control parameters and word count information. The BCW is shown in Figure 10-50. The first two bits are the transfer mode (TM) field which must be 00 or 10. Bits 13 and 12 are unused. Bits 11 through 0 are the buffer transfer count (BTC) or word count. Thirteen unique words are defined for output while sixteen are defined for input in Tables 10-2 and 10-3. If more than the defined number of words are transferred, the words will end around until the BTC reaches zero. Typically, BTCs greater than the defined number of words are not useful.

Control memory locations 1 and 5 are termed the BAP and must be initialized to the starting address of a data buffer. While a data transfer is in process, the buffer address pointer (BAP) contains the next address to be transferred between an I/O channel and memory.

Control memory locations 2 and 6 are termed the chain address pointer (CAP) and must be initialized to indicate the starting address of an I/O chain program. While a chain program is in process, the CAP contains the address of the next I/O instruction to be executed.

Control memory location 3 is associated with the bit jump I/O instruction.

Control memory location 8 is termed the input control word (ICW) and its lower 4 bits may specify which of 16 possible words to begin for input data transfers. See Table 10-2 and Figure 10-51.

Control memory location 9 is termed the output control word (OCW) and its lower 4 bits may specify which of 13 possible words to begin for output data transfers. See Table 10-3 and Figure 10-52.

Control memory location A is termed the message control word (MCW) and the lowest bit, when set to a logic 1, places the DIOM inputs into internal wraparound mode. See Figure 10-53.

TABLE 10-2. INPUT WORD DEFINITIONS

Word No.	ICW Code	IM/IO a-Field	Definition
0	0000	--	Output Discretes 000-015
1	0001	--	Output Discretes 016-031
2	0010	--	Output Discretes 032-047
3	0011	--	Output Discretes 048-063
4	0100	--	Output Discretes 064-079
5	0101	--	Output Discretes 080-095
6	0110	--	Output Discretes 096-111
7	0111	--	Output Discretes 112-127
8	1000	--	Output Discretes 128-143
9	1001	0100	Input Discretes 00-15
10	1010	1000	Input Discretes 16-31
11	1011	1100	Input Discretes 32-47
12	1100	--	Interrupt Discretes 00-15
13	1101	--	Interrupt Discretes 16-31
14	1110	--	Interrupt Discretes 32-47
15	1111	--	Interrupt Discretes 48-63

TABLE 10-3. OUTPUT WORD DEFINITIONS

Word No.	OCW Code	IM/IO a-Field	Definition
0	0000	0011	Output Discretes 000-015
1	0001	0111	Output Discretes 016-031
2	0010	1011	Output Discretes 032-047
3	0011	1111	Output Discretes 048-063
4	0100	0010	Output Discretes 064-079
5	0101	0110	Output Discretes 080-095
6	0110	1010	Output Discretes 096-111
7	0111	1110	Output Discretes 112-127
8	10XX	X001	Output Discretes 128-143
9	1100	--	Interrupt Masks 00-15
10	1101	--	Interrupt Masks 16-31
11	1110	--	Interrupt Masks 32-47
12	1111	--	Interrupt Masks 48-63

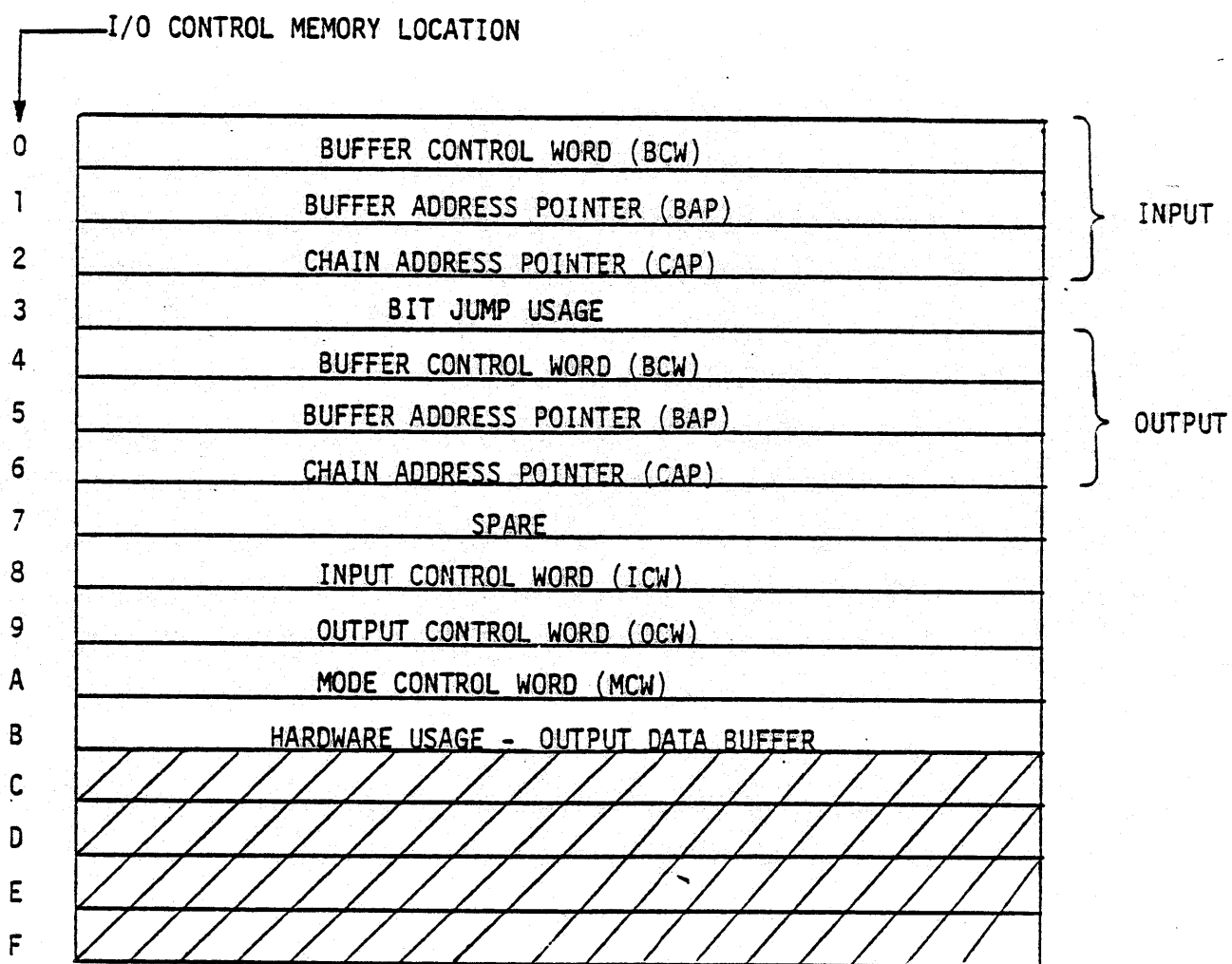


Figure 10-49. DIOM Control Memory Definition Map

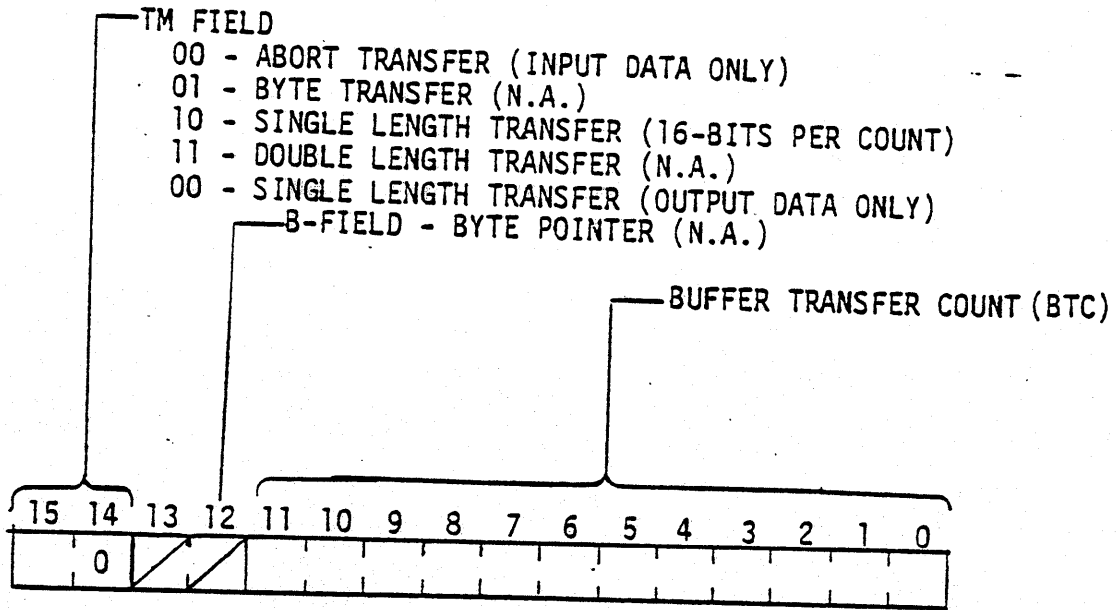


Figure 10-50. DIOM Buffer Control Word (BCW)

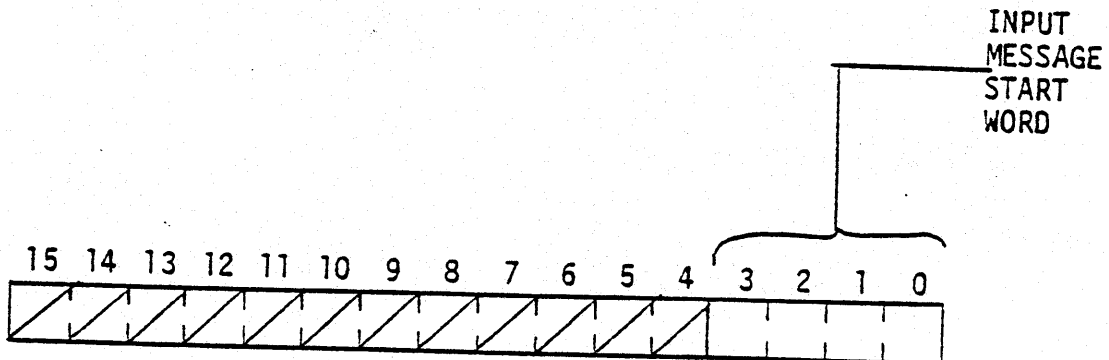


Figure 10-51. DIOM Input Control Word (ICW)

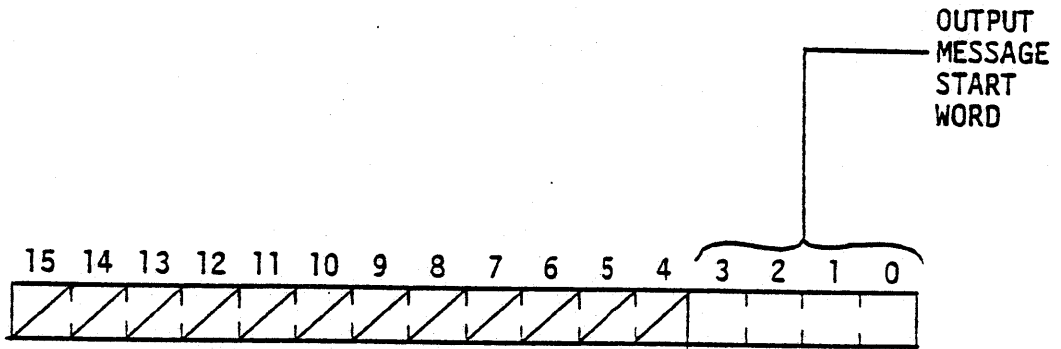


Figure 10-52. DIOM Output Control Word (OCW)

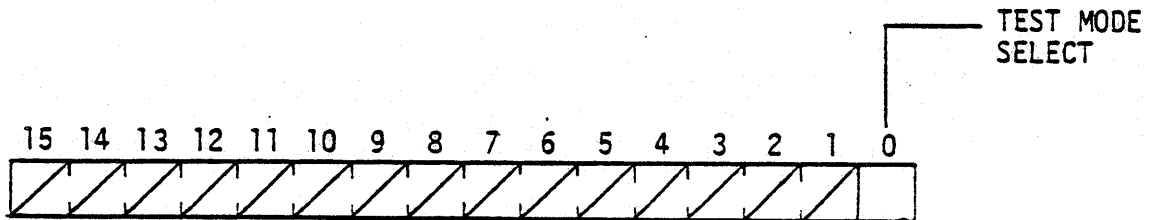


Figure 10-53. DIOM Mode Control Word (MCW)

Control memory location B is used as a lookahead output data word. This location is used dynamically within an output data transfer which, unlike other locations, requires no initialization. While an output data transfer is in process, the contents of buffer address pointer (BAP) is loaded into control memory location B.

DIOM CHANNEL INSTRUCTIONS

The following I/O chain instructions provide operations on the DIOM I/O channel type. Any that are not included are legal but performs a no operation.

- 1) Channel Control (E0 a m) (ACRm; CCR a,m) - The same as listed under Processor to I/O Channel Communication paragraph.
- 2) Initiate Message (E2 a m) (IM a,y,m) - Load the control memory location specified by the m-field with operand y; then, perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XX00	Initiate an input data transfer using the BCW and BAP in control memory locations 0 and 1, respectively. The first word is specified by the a-field. See Table 10-2.
0000	Contents of the lower 4 bits of control memory 8.
0100	Input Word 9
1000	Input Word 10
1100	Input Word 11
<u>XX00</u>	Initiate an output data transfer using BCW and BAP on control memory locations 4 and 5, respectively. The first word is specified by the a-field. See Table 10-3.
0011	Output Word 0
0111	Output Word 1
1011	Output Word 2
1111	Output Word 3
0010	Output Word 4
0110	Output Word 5
1010	Output Word 6
1110	Output Word 7

<u>a</u>	<u>Operation</u>
X001	Output Word 8
X101	Contents of the lower 4 bits of control memory 9.

- 3) Initiate Transfer (E3 a 0) (IO a,y) - Load control memory locations 0 and 1 (a-field equal to XX00) or 4 and 5 (a-field not equal to XX00) with the contents of main memory addresses y and y + 1 respectively. Perform the operation as specified by the a-field as defined in 2).
- 4) Initiate Chain (E6 a m) (ICK a,y,m; OCK a,y,m) - Load the control memory location specified by the m-field with the operand y for the I/O channel specified by the a-field. Then initiate an operation on the specified I/O channel as defined by the m-field as follows:

<u>m</u>	<u>Operation</u>
0010	Initiate an input chain program using the CAP in control memory location 2.
0110	Initiate an output chain program using the CAP in control memory location 6.
0X10	No operation

NOTE:

The CAP starting address for the chain program will be initialized to the value y.

- 5) Load Control Memory (E6 0 m) (LCMK m,y) - Load the control memory location specified by the m-field with the operand y.
- 6) Write Control Memory (E7 a m) (WIM a,y,m) - Load the control memory location specified by the m-field with the contents of the memory address y for the I/O channel specified by the a-field.
- 7) Load Control Memory (E7 0 m) (LCM m,y) - Load the control memory location specified by the m-field with the contents of memory address y.
- 8) Read Control Memory (EB a m) (RIM a,y,m) - Store the contents of the control memory location specified by the m-field at memory address y for the channel specified by the a field.
- 9) Store Control Memory (EB 0 m) (SCM m,y) - Store the contents of the control memory location specified by the m-field at memory address y.

- 10) Halt/Interrupt (EC a 0) (HCR;IPR) - Perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	HCR - Halt I/O chain program.
XXX1	IPR - Generate a Class III, ICI interrupt if executed within an input chain program.
XXX1	IPR - Generate a Class III, OCI interrupt if executed within an output chain program.

- 11) Set/Clear Flag (EF a 0) (SF y ; ZF y) - Perform the operation specified by the a-field as follows:

<u>a</u>	<u>Operation</u>
XXX0	ZF - Clear the 2 most significant bits (flag) of memory address y.
XXX1	SF - Set the 2 most significant bits (flag) of memory address y.

- 12) Conditional Jump (F2 a 0) (SIMC a,y) - Load the control memory location 2 (input) or control memory location 6 (output) with the operand y. The a-field may be any value. This instruction is an unconditional jump for the DIOM I/O channel type.

- 13) Search (F4 0 m) (SFSC m) - Perform the operation specified by the m-field as follows:

<u>m</u>	<u>Operation</u>
XXXX	No operation

- 14) Serial Interface Control (F8 0 m) (CSIR m) - Perform the operation specified by the m-field as follows:

<u>m</u>	<u>Operation</u>
XX00	Initialize the capture value file by: 1) clearing the capture value file and 2) setting each capture bit to a logic 1 whose corresponding input is at a logic 1 level.
XX01	Clear the capture value file.
XX10	Clear the capture bits to a logic 0 whose corresponding mask bit is at a logic 0.
XX11	NOP

- 15) Store Status (FB a m) (SST a,y,m) (CSST y,m) - Store the channel status as specified by the m-field in memory address y as follows:

<u>m</u>	<u>Operation</u>
XX00	Channel Status 0 (Figure 10-54)
XX01	Channel Status 1 (Figure 10-55)
XX1X	Interrupt Status (Figure 10-56)

- 16) Bit Jump (FD 0 m) (BJ m,y) - Load the control memory location 2 (input) or control memory location 6 (output) with operand y when the bit specified by the m-field is a logic 1. When the specified bit is a logic 0, no operation.
- 17) Exchange Control Memory (FE a m) (XIM a,y,m) - Store the contents of control memory location specified by the m-field at memory address y for the I/O channel specified by the a-field when the m-field is equal to 0X10. Then load the specified control memory location with the contents of memory address $y \oplus 1$. If the m-field is not equal to 0X10, an I/O instruction fault is generated.
- 18) Exchange Control Memory (FE 0 m) (XCM m,y) - Store the contents of control memory location specified by the m-field at memory address y and then load the specified control memory location with the contents of memory address $y \oplus 1$.

DIOM INTERRUPT HANDLING

The DIOM I/O channel type is capable of generating the EII, OCI, and ICI Class III interrupts.

The Class III mask, when set, enables all three types of interrupts. The ACR/CCR I/O channel instruction (Op Code E0) sets or clears this mask bit. This mask bit is found in interrupt word bit 6 (Figure 10-56). After a disable Class III mask is executed, all Class III interrupts will be blocked until re-enabled. No interrupts are lost because of the disable Class III I/O channel instruction.

Forty-eight possible external interrupts may cause the DIOM to generate an EII interrupt. An external interrupt is defined as an inactive-to-active transition of an input signal that has a minimum active duration of 2.5 microseconds. Therefore, an interrupt may be an active level or minimum of one active 2.5-microsecond pulse. All the external interrupts are assigned a fixed priority, interrupt 0 being the highest priority and interrupt 47 being the lowest. Table 10-2 defines three input words that contain the unmasked captured value of each individual interrupt. These interrupts may be input by setting up an appropriate I/O chain program for inputting the desired word numbers 12, 13, and/or 14.

For an external interrupt to generate an EII, several parameters must be set up. For each external interrupt, a corresponding individual mask must be set. Three mask words are defined in Table 10-3. They are output words 9,

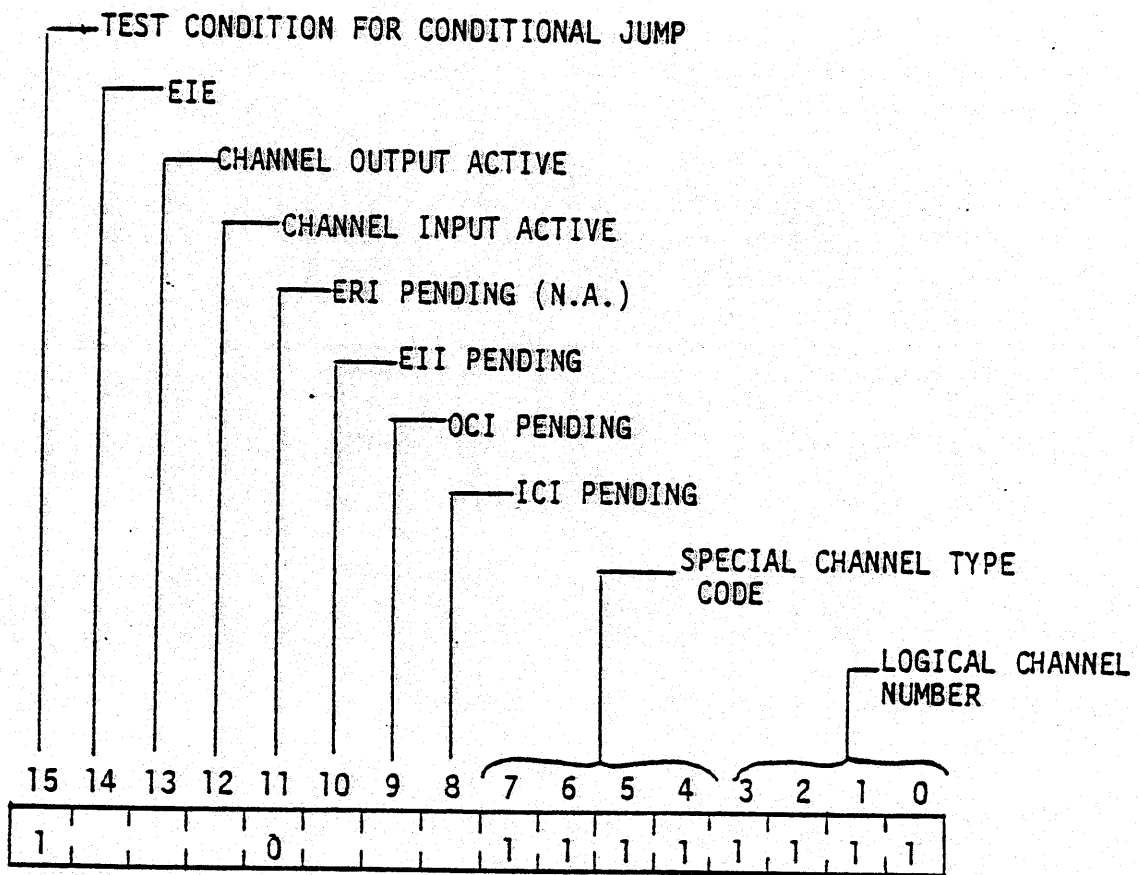


Figure 10-54. DIOM Status Word 0 Format

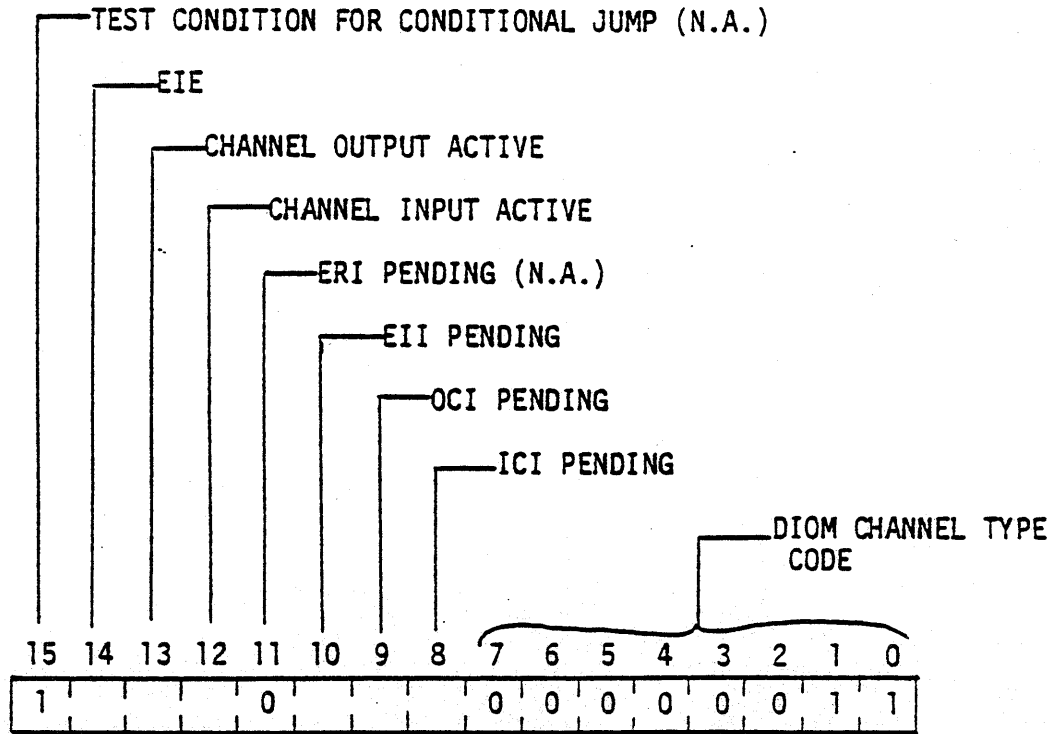


Figure 10-55. DIOM Status Word 1 Format

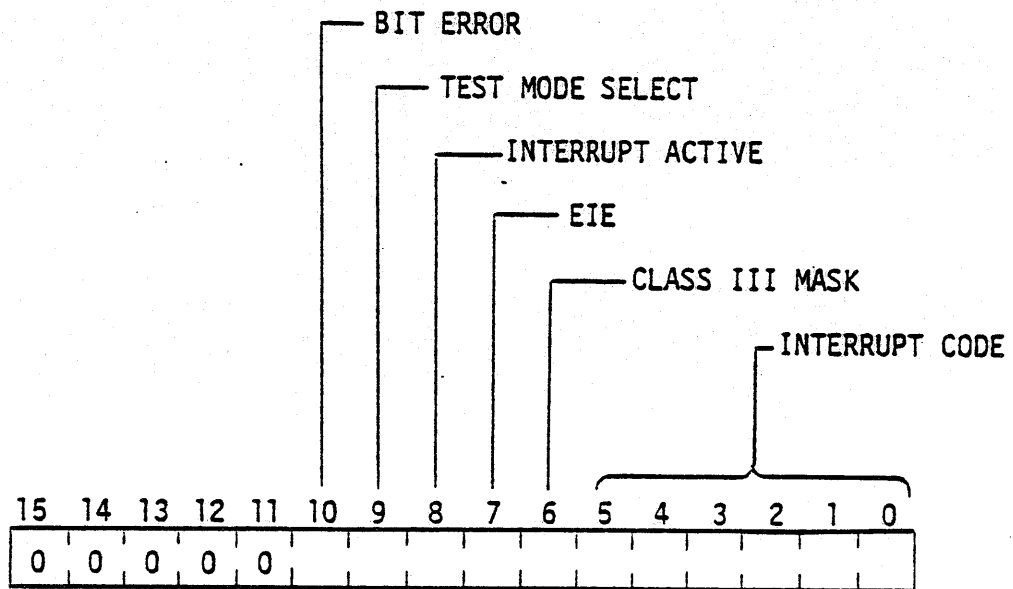


Figure 10-56. DIOM Interrupt Word Format

10, and 11 and may be output via an appropriate I/O chain program. Secondly, the EIE must be set by the appropriate I/O channel instruction (Op Code E0). When set, the EIE causes the DIOM to constantly scan for the highest priority masked external interrupt and then store it in memory address 8F hexadecimal with the interrupt word (Figure 10-56). Table 10-4 defines the interrupt codes. If, in addition to the EIE set, the Class III mask and bit 1 of processor status word 0 are set, then the EII interrupt will be generated. When the EII interrupt is generated, the DIOM automatically clears EIE and clears the capture bit associated with the interrupt in memory address 8F hexadecimal. Therefore, the interrupt routine shall not concern itself with clearing that capture value bit.

If either the EIE or Class III mask is clear, the EII interrupt will be blocked. Both Class III mask and EIE bits are found in interrupt word bits 6 and 7, respectively (Figure 10-56). Bit 8, when set, indicates that the interrupt code is an active interrupt. EII Pending is bit 10 of channel status 0 or 1 (Figures 10-54 and 10-55). EII Pending in the DIOM means that an active interrupt code has been input to memory address 8F hexadecimal. Three special interrupt management operations may be executed via the SICR/CSIR I/O channel instruction (Op Code F8).

On a power up or master clear I/O channel instruction, the capture value file initially cleared, and each corresponding capture bit set to a logic 1 if the level of the external interrupt is a logic 1. On a power up situation, the interrupt active signal (bit 8 of the interrupt word) has no meaning until the mask file is initialized.

The OCI and ICI interrupts are generated via the IPR I/O channel instruction (Op Code EC). The Class III mask and bit 1 of processor status word 0 must both be enabled for the interrupt to occur.

DIOM PROGRAMMING CONSIDERATIONS

The DIOM may set up chain programs to do input and output data transfers via IM or IO I/O channel instructions (Op Codes E2 and E3). These and other instructions are used to specify the initial values of the BCW and BAP.

Buffers of data transfers may be initiated starting with any word by setting up the ICW or OCW, and then using an IM or IO instruction. If some instruction time is desired to be saved, any output or direct input word may be transferred without setting up the ICW or OCW. It is the intent to provide the programmer with some option in initiating data transfers with minimal cost of hardware. If only one word is desired to output or input, only one word needs to be output or input by specifying the starting word with the a-field of the IM or IO instruction.

Input Data Transfers

There are 16 input words defined in Table 10-2. These may be read into memory with an IM or IO instruction as stated previously. The transfer of these words is controlled by the BCW (Figure 10-50) and BAP in control memory locations 0 and 1, respectively. During an input data transfer with a TM field of 10, the BAP value is incremented by 1 and the BTC of the BCW is decremented by 1 for each word input. When the BTC = 0, the chain program

TABLE 10-4. INTERRUPT WORD CODES

External Interrupt	Interrupt Word Bits 5-0	
	Decimal	Binary
00	000000	00
01	000001	01
02	000010	02
03	000011	03
04	000100	04
05	000101	05
06	000110	06
07	000111	07
08	001000	08
09	001001	09
10	001010	0A
11	001011	0B
12	001100	0C
13	001101	0D
14	001110	0E
15	001111	0F
16	010000	10
17	010001	11
18	010010	12
19	010011	13
20	010100	14
21	010101	15
22	010110	16
23	010111	17
24	011000	18
25	011001	19
26	011010	1A
27	011011	1B
28	011100	1C
29	011101	1D
30	011110	1E
31	011111	1F
32	100000	20
33	100001	21
34	100010	22
35	100011	23
36	100100	24
37	100101	25
38	100110	26
39	100111	27
40	101000	28
41	101001	29
42	101010	2A
43	101011	2B
44	101100	2C
45	101101	2D
46	101110	2E
47	101111	2F
*48	110000	30

TABLE 10-4. INTERRUPT WORD CODES (Cont.)

External Interrupt	Interrupt Word Bits 5-0		
	Decimal	Binary	Hexidecimal
*49		110001	31
*50		110010	32
*51		110011	33
*52		110100	34
*53		110101	35
*54		110110	36
*55		110111	37
*56		111000	38
*57		111001	39
*58		111010	3A
*59		111011	3B
*60		111100	3C
*61		111101	3D
*62		111110	3E
*63		111111	3F

*For Internal Wraparound only

continues its execution of I/O chain instructions. If the TM field equals 00, only the BTC is decremented, inputs are discarded and not stored in memory. Note that BTC values greater than 16 repeat sets of 16 possible input words.

Output Data Transfers

There are 13 output words defined in Table 10-3. These are also output from memory with an IM or IO instruction. These transfers are controlled by the BCW (Figure 10-50) and BAP in control memory locations 4 and 5, respectively. During an output data transfer, the BAP value is incremented by 1 and the BTC of the BCW is decremented by 1 for each word output. Again, when the BTC = 0, the chain program continues. Note that BTC values greater than 13 repeat sets of 13 possible output words.

Chain and Chain Transfers

The DIOM is a full duplex channel and may initiate input and output chains that may be active simultaneously. After initiation of an input data transfer from an input chain and on completion of that input data transfer, the DIOM continues to execute input chain instructions. This is also true of output chain programs. If, however, an output data transfer is initiated from an input chain, the DIOM halts the input chain and loads the output CAP with the input CAP. Upon completion of that output data transfer, the DIOM continues by executing output chain instructions. This same situation works for input data transfers initiated within output chains.

APPENDIX A
INSTRUCTION REPERTOIRE

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC	
#00	2	a	m	02	a	m	SPT a,y,m	Stack Put Top	(Y) → (Ra), (Ra) → Y			-NA-	
#00	3	a	m	03	a	m	BL a,y,m	Byte load	(Y) byte → Ra	0	0	X	
#01	0	a	m	04	a	m	LR a,m	Load (Register)	(Rm) → Ra	0	0	X	
#01	1	a	m	05	a	m	LI a,m	Load (Indirect)	(Y*) → Ra	0	0	X	
#01	2	a	m	06	a	m	LK a,y,m	Load (Constant)	Y → Ra	0	0	X	
#01	3	a	m	07	a	m	L a,y,m	Load	(Y) → Ra	0	0	X	
#02	0	a	00	08	a	0	PR a	Make positive	If (Ra) < 0, (Ra)' → Ra	X	X	X	
#02	0	a	01	08	a	1	NR a	Make negative	If (Ra) > 0, (Ra)' → Ra	X	0	X	
	02	0	a	02	08	a	2	RR a	Round	(Ra) + (Ra ⊕ 1):15 → Ra	X	X	X
#02	0	a	03	08	a	3	IPI a	°Initiate Processor Int	Set processor interrupt a			-NA-	
#02	0	a	04	08	a	4	TCR a	Two's Complement	(Ra)' → Ra	X	X	X	
	02	0	a	05	08	a	5	TCDR a	Two's Complement Double	(Ra, Ra ⊕ 1)' → Ra, Ra ⊕ 1	X	X	X
#02	0	a	06	08	a	6	OCR a	One's Complement	(Ra) bit-by-bit complement → Ra	0	0	X	
#02	0	a	07	08	a	7	SCI a	Support Channel Input	Support channel input → Ra	0	0	X	
#02	0	a	10	08	a	8	IROR a	Increase Ra by 1	(Ra) + 1 → Ra	X	X	X	
#02	0	a	11	08	a	9	DROR a	Decrease Ra by 1	(Ra) - 1 → Ra	X	X	X	
#02	0	a	12	08	a	A	IRTR a	Increase Ra by 2	(Ra) + 2 → Ra	X	X	X	
#02	0	a	13	08	a	B	DRTR a	Decrease Ra by 2	(Ra) - 2 → Ra	X	X	X	
#02	0	-	14	08	-	C	IPLF	°IPL Failed	Set IPLF discrete			-NA-	
#02	0	a	15	08	a	D	DJ a	°Diagnostic Jump	R15 → uP			-NA-	
#02	0	-	16	08	-	E	RBT	°Reset Bit Timer	0 → Bit Timer			-NA-	
#02	0	-	17	08	-	F	SBT	°Set BIT Indicator	Set BIT indicator			-NA-	
	02	1	a	09	a	m	LDI a,m	Load Double (Indirect)	(Y*, Y* ⊕ 1) → Ra, Ra ⊕ 1	0	0	X	
	02	3	a	0B	a	m	LD a,y,m	Load Double	(Y, Y ⊕ 1) → Ra, Ra ⊕ 1	0	0	X	
#03	0	a	00	0C	a	0	ER a	Executive Return	Generate interrupt; P+1 → Ra	0	0	X	

14122000

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

A-2

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#03	0	a	01	0C	a	1	SSOR a	Store SR1	(SR1) → Ra	0	0	X
#03	0	a	02	0C	a	2	SSTR a	Store SR2	(SR2) → Ra	0	0	X
#03	0	a	03	0C	a	3	SCR a	Store Clock	(RTC register):15-0 → Ra	0	0	X
03	0	a	04	0C	a	4	LPR a	Load P	(Ra) → P			
#03	0	a	05	0C	a	5	LSOR a	◦Load SR1	(Ra) → SR1		-NA-	
#03	0	a	06	0C	a	6	LSTR a	◦Load SR2	(Ra) → SR2		-NA-	
#03	0	a	07	0C	a	7	LCR a	◦Load RTC lower	(Ra) → RTC register:15-0		-NA-	
#03	0	-	10	0C	-	8	ECR	◦Enable Clock and Interrupt	Enable RTC register and interrupt		-NA-	
#03	0	-	11	0C	-	9	DCR	◦Disable Clock	Disable RTC register		-NA-	
03	0	a	12	0C	a	A	LEM a	◦Load and Enable Monitor Clock	(Ra) → Monitor clock register; enable countdown		-NA-	
03	0	-	13	0C	-	B	DM	◦Disable Monitor Clock	Disable monitor clock register		-NA-	
03	0	a	14	0C	a	C	LCRD a	◦Load Double and Enable Clock	(Ra, Ra ⊕ 1) → RTC; enable count up		-NA-	
#03	0	a	15	0C	a	D	SCRD a	Store Clock Double	(RTC register) → Ra, Ra ⊕ 1	0	0	X
#03	0	-	16	0C	-	E	ECIR	◦Enable Clock Interrupt	Enable RTC overflow interrupt		-NA-	
#03	0	-	17	0C	-	F	DCIR	◦Disable Clock Interrupt	Disable RTC overflow interrupt		-NA-	
#03	1	a	m	0D	a	m	SCIO a,m	Support Channel I/O	(Ra) → support channel buffer m → I/O code, Set channel busy		-NA-	
03	3	a	m	0F	a	m	LM a,y,m	Load multiple	(Y ... Y+m-a) → Ra ... Rm		-NA-	
04	0	a	01	10	a	1	RVR a	Reverse Register	Reverse (Ra)	0	0	X
04	0	a	02	10	a	2	CNT a	Count Ones	Number of binary ones in Ra → Ra ⊕ 1		-NA-	
04	0	a	03	10	a	3	SFR a	Scale Factor	Shift (Ra, Ra ⊕ 1) left until (Ra):15 ≠ (Ra):14; shift count → Ra ⊕ 1+1 ①		-NA-	
04	0	a	04	10	a	4	SMC a	Store Monitor Clock	(Mon) → Ra		-NA-	
#04	2	a	m	12	a	m	QPT a,y,m	Queue Put Top	(Y) → (Ra), (Ra) → Y; if (Y) was = 0 then (Ra) → Y ⊕ 1		-NA-	

14122000

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

14122000

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#04	3	a	m	13	a	m	BLX a,y,m	Byte Load and Index by 1	(Y) byte → Ra; (Rm)+1 → Rm	0	0	X
#05	0	a	m	14	a	m	SBR a,m	Set Bit	1 → (Ra):m	0	0	X
#05	1	a	m	15	a	m	LXI a,m	Load and Index by 1 (Indirect)	(Y*) → Ra; (Rm)+1 → Rm	0	0	X
#05	2	a	m	16	a	m	QPB a,y,m	Queue Put Bottom	(Ra) → (Y ⊕ 1), (Ra) → Y ⊕ 1, 0 → (Ra)		-NA-	
#05	3	a	m	17	a	m	LX a,y,m	Load and Index by 1	(Y) → Ra; (Rm)+1 → Rm	0	0	X
#06	0	a	m	18	a	m	ZBR a,m	Zero Bit	0 → (Ra):m	0	0	X
06	1	a	m	19	a	m	LDXI a,m	Load Double Index by 2 (Indirect)	(Y*, Y* ⊕ 1) → Ra, Ra ⊕ 1; (Rm)+2 → Rm	0	0	X
#06	2	a	m	1A	a	m	SGT a,y,m	Stack Get Top	(Y) → Ra, if (Y) ≠ 0 then ((Y)) → Y and P+3 → P, if (Y) ≠ 0 then P+2 → P		-NA-	
06	3	a	m	1B	a	m	LDX a,y,m	Load Double, Index by 2	(Y, Y ⊕ 1) → Ra, Ra ⊕ 1; (Rm)+2 → Rm	0	0	X
#07	0	a	m	1C	a	m	CBR a,m	Compare Bit	Compare bit m of Ra with zero	0	0	X
#07	1	-	m	1D	-	m	LPI m	°Load PSW (Indirect)	(Y*, Y*+1, Y*+2) → P, SR1, SR2		-NA-	
#07	2	a	m	1E	a	m	QGT a,y,m	Queue Get Top	(Y) → Ra; if (Y) = 0 then P+2 → P; if (Y) ≠ 0 then P+3 → P, ((Y)) → Y; if ((Y)) = 0 then Y → Y ⊕ 1		-NA-	
#07	3	-	m	1F	-	m	LP y,m	°Load PSW	(Y, Y+1, Y+2) → P, SR1, SR2		-NA-	
#10	0	a	m	20	a	m	LRSR a,m	Logical Right Shift (Register)	Shift (Ra) right (Rm):5-0 places, zero fill	0	0	X
#10	2	a	m	22	a	m	LRS a,y,m	Logical Right Shift	Shift (Ra) right Y:5-0 places, zero fill	0	0	X
#10	3	a	m	23	a	m	BS a,y,m	Byte Store	(Ra):7-0 → Y byte		-NA-	
#11	0	a	m	24	a	m	ARSR a,m	Algebraic Right Shift (Register)	Shift (Ra) right (Rm):5-0 places, sign fill	0	0	X
#11	1	a	m	25	a	m	SI a,m	Store (Indirect)	(Ra) → y*		-NA-	
#11	2	a	m	26	a	m	ARS a,y,m	Algebraic Right Shift	Shift (Ra) right Y:5-0 places, sign fill	0	0	X
# IOP Instructions						°Executive Mode Instructions			① Count = 31 for all zeros or all ones			

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#11	3	a	m	27	a	m	S a,y,m	Store	(Ra) → Y		-NA-	
12	0	a	m	28	a	m	LRDR a,m	Logical Right Double Shift (Register)	Shift (Ra, Ra ⊕ 1) right (Rm):5-0 places, zero fill	0	0	X
12	1	a	m	29	a	m	SDI a,m	Store Double (Indirect)	(Ra, Ra ⊕ 1) → Y*, Y* ⊕ 1		-NA-	
12	2	a	m	2A	a	m	LRD a,y,m	Logical Right Double Shift	Shift (Ra, Ra ⊕ 1) right Y:5-0 places, zero fill	0	0	X
12	3	a	m	2B	a	m	SD a,y,m	Store Double	(Ra, Ra ⊕ 1) → Y, Y ⊕ 1		-NA-	
13	0	a	m	2C	a	m	ARDR a,m	Algebraic Right Double Shift (Register)	Shift (Ra, Ra ⊕ 1) right (Rm):5-0 places, sign fill	0	0	X
13	2	a	m	2E	a	m	ARD a,y,m	Algebraic Right Double Shift	Shift (Ra, Ra ⊕ 1) right Y:5-0 places, sign fill	0	0	X
13	3	a	m	2F	a	m	SM a,y,m	Store Multiple	(Ra ... Rm) → Y ... Y+m-a		-NA-	
#14	0	a	m	30	a	m	ALSR a,m	Algebraic Left Shift (Register)	Shift (Ra) left (Rm):5-0 places, zero fill	0	X	X
#14	2	a	m	32	a	m	ALS a,y,m	Algebraic Left Shift	Shift (Ra) left Y:5-0 places, zero fill	0	X	X
#14	3	a	m	33	a	m	BSX a,y,m	Byte Store, Index by 1	(Ra):7-0 → Y byte; (Rm)+1 → Rm		-NA-	
#15	0	a	m	34	a	m	CLSR a,m	Circular Left Shift (Register)	Shift (Ra) circularly left (Rm):5-0 places	0	0	X
#15	1	a	m	35	a	m	SXI a,m	Store, Index by 1 (Indirect)	(Ra) → Y*; (Rm)+1 → Rm		-NA-	
#15	2	a	m	36	a	m	CLS a,y,m	Circular Left Shift	Shift (Ra) circularly left Y:5-0 places	0	0	X
#15	3	a	m	37	a	m	SX a,y,m	Store, Index by 1	(Ra) → Y; (Rm)+1 → Rm		-NA-	
16	0	a	m	38	a	m	ALDR a,m	Algebraic Left Double Shift (Register)	Shift (Ra, Ra ⊕ 1) left (Rm):5-0 places, zero fill	0	X	X
16	1	a	m	39	a	m	SDXI a,m	Store Double, Index by 2 (Indirect)	(Ra, Ra ⊕ 1) → Y*, Y* ⊕ 1; (Rm)+2 → Rm		-NA-	
16	2	a	m	3A	a	m	ALD a,y,m	Algebraic Left Double Shift	Shift (Ra, Ra ⊕ 1) left Y:5-0 places, zero fill	0	X	X

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format	Hexadecimal Format	Coding Format	Instruction	Operation	C	OV	CC
16 3 a m	3B a m	SDX a,y,m	Store Double, Index by 2	$(Ra, Ra \oplus 1) \rightarrow Y, Y \oplus 1; (Rm)+2 \rightarrow Rm$		-NA-	
17 0 a m	3C a m	CLDR a,m	Circular Left Double Shift (Register)	Shift $(Ra, Ra \oplus 1)$ circularly left $(Rm):5-0$ places	0	0	X
17 1 - m	3D - m	SZI m	Store Zeros (Indirect)	$0 \rightarrow Y^*$		-NA-	
17 2 a m	3E a m	CLD a,y,m	Circular Left Double Shift	Shift $(Ra, Ra \oplus 1)$ circularly left $Y:5-0$ places	0	0	X
17 3 - m	3F - m	SZ y,m	Store Zeros	$0 \rightarrow Y$		-NA-	
#20 0 a m	40 a m	SUR a,m	Subtract (Register)	$(Ra) - (Rm) \rightarrow Ra$	X	X	X
#20 1 a m	41 a m	SUI a,m	Subtract (Indirect)	$(Ra) - (Y^*) \rightarrow Ra$	X	X	X
#20 2 a m	42 a m	SUK a,y,m	Subtract (Constant)	$(Ra) - Y \rightarrow Ra$	X	X	X
#20 3 a m	43 a m	SU a,y,m	Subtract	$(Ra) - (Y) \rightarrow Ra$	X	X	X
21 0 a m	44 a m	SUDR a,m	Subtract Double (Register)	$(Ra, Ra \oplus 1) - (Rm, Rm \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
21 1 a m	45 a m	SUDI a,m	Subtract Double (Indirect)	$(Ra, Ra \oplus 1) - (Y^*, Y^* \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
21 3 a m	47 a m	SUD a,y,m	Subtract Double	$(Ra, Ra \oplus 1) - (Y, Y \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
#22 0 a m	48 a m	AR a,m	Add (Register)	$(Ra) + (Rm) \rightarrow Ra$	X	X	X
#22 1 a m	49 a m	AI a,m	Add (Indirect)	$(Ra) + (Y^*) \rightarrow Ra$	X	X	X
#22 2 a m	4A a m	AK a,y,m	Add (Constant)	$(Ra) + Y \rightarrow Ra$	X	X	X
#22 3 a m	4B a m	A a,y,m	Add	$(Ra) + (Y) \rightarrow Ra$	X	X	X
23 0 a m	4C a m	ADR a,m	Add Double (Register)	$(Ra, Ra \oplus 1) + (Rm, Rm \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
23 1 a m	4D a m	ADI a,m	Add Double (Indirect)	$(Ra, Ra \oplus 1) + (Y^*, Y^* \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
23 3 a m	4F a m	AD a,y,m	Add Double	$(Ra, Ra \oplus 1) + (Y, Y \oplus 1) \rightarrow Ra, Ra \oplus 1$	X	X	X
#24 0 a m	50 a m	CR a,m	Compare (Register)	$(Ra) - (Rm)$	X	X	X

14122000

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#24	1	a	m	51	a	m	CI a,m	Compare (Indirect)	$(Ra) - (Y^*)$	X	X	X
#24	2	a	m	52	a	m	CK a,y,m	Compare (Constant)	$(Ra) - Y$	X	X	X
#24	3	a	m	53	a	m	C a,y,m	Compare	$(Ra) - (Y)$	X	X	X
25	0	a	m	54	a	m	CDR a,m	Compare Double (Register)	$(Ra, Ra \oplus 1) - (Rm, Rm \oplus 1)$	X	X	X
25	1	a	m	55	a	m	CDI a,m	Compare Double (Indirect)	$(Ra, Ra \oplus 1) - (Y^*, Y^* \oplus 1)$	X	X	X
25	3	a	m	57	a	m	CD a,y,m	Compare Double	$(Ra, Ra \oplus 1) - (Y, Y \oplus 1)$	X	X	X
#26	0	a	m	58	a	m	MR a,m	Multiply (Register)	$(Ra \oplus 1) \cdot (Rm) \rightarrow Ra, Ra \oplus 1$	0	0	X
#26	1	a	m	59	a	m	MI a,m	Multiply (Indirect)	$(Ra \oplus 1) \cdot (Y^*) \rightarrow Ra, Ra \oplus 1$	0	0	X
#26	2	a	m	5A	a	m	MK a,y,m	Multiply (Constant)	$(Ra \oplus 1) \cdot Y \rightarrow Ra, Ra \oplus 1$	0	0	X
#26	3	a	m	5B	a	m	M a,y,m	Multiply	$(Ra \oplus 1) \cdot (Y) \rightarrow Ra, Ra \oplus 1$	0	0	X
#27	0	a	m	5C	a	m	DR a,m	Divide (Register)	$(Ra, Ra \oplus 1) / (Rm) \rightarrow Ra \oplus 1; remainder \rightarrow Ra$	0	X	X
#27	1	a	m	5D	a	m	DI a,m	Divide (Indirect)	$(Ra, Ra \oplus 1) / (Y^*) \rightarrow Ra \oplus 1; remainder \rightarrow Ra$	0	X	X
#27	2	a	m	5E	a	m	DK a,y,m	Divide (Constant)	$(Ra, Ra \oplus 1) / Y \rightarrow Ra \oplus 1; remainder \rightarrow Ra$	0	X	X
#27	3	a	m	5F	a	m	D a,y,m	Divide	$(Ra, Ra \oplus 1) / (Y) \rightarrow Ra \oplus 1; remainder \rightarrow Ra$	0	X	X
#30	0	a	m	60	a	m	ANDR a,m	AND (Register)	$(Ra) \odot (Rm) \rightarrow Ra$	0	0	X
#30	1	a	m	61	a	m	ANDI a,m	AND (Indirect)	$(Ra) \odot (Y^*) \rightarrow Ra$	0	0	X
#30	2	a	m	62	a	m	ANDK a,y,m	AND (Constant)	$(Ra) \odot Y \rightarrow Ra$	0	0	X
#30	3	a	m	63	a	m	AND a,y,m	AND	$(Ra) \odot (Y) \rightarrow Ra$	0	0	X
#31	0	a	m	64	a	m	ORR a,m	OR (Register)	$(Ra) \oplus (Rm) \rightarrow Ra$	0	0	X
#31	1	a	m	65	a	m	ORI a,m	OR (Indirect)	$(Ra) \oplus (Y^*) \rightarrow Ra$	0	0	X
# IOP Instructions												

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#31	2	a	m	66	a	m	ORK a,y,m	OR (Constant)	$(Ra) \oplus Y \rightarrow Ra$	0	0	X
#31	3	a	m	67	a	m	OR a,y,m	OR	$(Ra) \oplus (Y) \rightarrow Ra$	0	0	X
32	0	a	m	68	a	m	XORR a,m	Exclusive OR (Register)	$(Ra) \oplus (Rm) \rightarrow Ra$	0	0	X
32	1	a	m	69	a	m	XORI a,m	Exclusive OR (Indirect)	$(Ra) \oplus (Y^*) \rightarrow Ra$	0	0	X
32	2	a	m	6A	a	m	XORK a,y,m	Exclusive OR (Constant)	$(Ra) \oplus Y \rightarrow Ra$	0	0	X
32	3	a	m	6B	a	m	XOR a,y,m	Exclusive OR	$(Ra) \oplus (Y) \rightarrow Ra$	0	0	X
33	0	a	m	6C	a	m	MSR a,m	Masked Substitute (Register)	If $(Ra \oplus 1):n = 1; (Rm):n \rightarrow Ra:n$	0	0	X
33	1	a	m	6D	a	m	MSI a,m	Masked Substitute (Indirect)	If $(Ra \oplus 1):n = 1; (Y^*):n \rightarrow Ra:n$	0	0	X
33	2	a	m	6E	a	m	MSK a,y,m	Masked Substitute (Constant)	If $(Ra \oplus 1):n = 1; Y:n \rightarrow Ra:n$	0	0	X
33	3	a	m	6F	a	m	MS a,y,m	Masked Substitute	If $(Ra \oplus 1):n = 1; (Y):n \rightarrow Ra:n$	0	0	X
#34	0	a	m	70	a	m	CMR a,m	Compare Masked (Register)	$[(Ra) \odot (Ra \oplus 1)] - [(Rm) \odot (Ra \oplus 1)]$	0	0	X
#34	1	a	m	71	a	m	CMI a,m	Compare Masked (Indirect)	$[(Ra) \odot (Ra \oplus 1)] - [(Y^*) \odot (Ra \oplus 1)]$	0	0	X
#34	2	a	m	72	a	m	CMK a,y,m	Compare Masked (Constant)	$[(Ra) \odot (Ra \oplus 1)] - [Y \odot (Ra \oplus 1)]$	0	0	X
#34	3	a	m	73	a	m	CM a,y,m	Compare Masked	$[(Ra) \odot (Ra \oplus 1)] - [(Y) \odot (Ra \oplus 1)]$	0	0	X
#35	0	-	-	74	-	-	IOCR	Input/Output Command	Execute (C Cell); $0 \rightarrow C \text{ Cell: } 15-14$		-NA-	
35	1	-	m	75	-	m	BFI m	Biased Fetch (Indirect)	$(Y^*):15 \rightarrow CC; 1 \rightarrow (Y^*):15,14$	0	0	X
35	2	-	m	76	-	m	REX y,m	Remote Execute	Execute (Y); $P + 2 \rightarrow P$ unless jump		-NA-	
35	3	-	m	77	-	m	BF y,m	Biased Fetch	$(Y):15 \rightarrow CC; 1 \rightarrow (Y):15,14$	0	0	X
#40	0	00	m	80	0	m	JER m	Jump Equal	If CC indicates = or 0; $(Rm) \rightarrow P$		-NA-	
#40	0	01	m	80	1	m	JNER m	Jump Not Equal	If CC indicates \neq or not 0; $(Rm) \rightarrow P$		-NA-	
#40	0	02	m	80	2	m	JGER m	Jump Greater or Equal	If CC indicates \geq or +; $(Rm) \rightarrow P$		-NA-	
#40	0	03	m	80	3	m	JLSR m	Jump Less	If CC indicates < or -; $(Rm) \rightarrow P$		-NA-	
#40	0	04	m	80	4	m	JOR m	Jump Overflow	If overflow set: $(Rm) \rightarrow P$		-NA-	
#40	0	05	m	80	5	m	JCR m	Jump Carry	If carry set: $(Rm) \rightarrow P$		-NA-	
40	0	06	m	80	6	m	JPTR m	Jump Power out of Tolerance	If power out of tolerance: $(Rm) \rightarrow P$		-NA-	

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format				Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
40	0	07	m	80	7	m	JBR m	Jump Bootstrap 2 Selected	If bootstrap 2 selected: (Rm) → P			-NA-
#40	0	10	m	80	8	m	JR m	Jump	(Rm) → P			-NA-
#40	0	11	m	80	9	m	JSR m	° Jump after Stop	Stop; (Rm) → P			-NA-
40	0	12	m	80	A	m	JKSR 1,m	° Jump. If Key Set – Stop, then Jump (Register)	If key 1 set, stop; (Rm) → P			-NA-
40	0	13	m	80	B	m	JKSR 2,m	° Jump. If Key Set – Stop, then Jump (Register)	If key 2 set, stop; (Rm) → P			-NA-
#40	0	14	m	80	C	m	JSCR m	Jump Support Channel Busy – RR	If support channel busy; (Rm) → P			-NA-
#40	1		d	81		d	LJ x \bar{d}	Local Jump	P + x \bar{d} → P			-NA-
#40	2	00	m	82	0	m	JE y,m	Jump Equal	If CC indicates = or 0; Y → P			-NA-
#40	2	01	m	82	1	m	JNE y,m	Jump Not Equal	If CC indicates ≠ or not 0; Y → P			-NA-
#40	2	02	m	82	2	m	JGE y,m	Jump Greater than or Equal	If CC indicates ≥ or +; Y → P			-NA-
#40	2	03	m	82	3	m	JLS y,m	Jump Less	If CC indicates < or -; Y → P			-NA-
#40	2	04	m	82	4	m	JO y,m	Jump on Overflow	If overflow set; Y → P			-NA-
#40	2	05	m	82	5	m	JC y,m	Jump on Carry	If carry set, Y → P			-NA-
40	2	06	m	82	6	m	JPT y,m	Jump if Power out of Tolerance	If power out of tolerance; Y → P			-NA-
40	2	07	m	82	7	m	JB y,m	Jump if Bootstrap 2 Selected	If bootstrap 2 selected; Y → P			-NA-
#40	2	10	m	82	8	m	J y,m	Jump	Y → P			-NA-
#40	2	11	m	82	9	m	JS y,m	° Jump after Stop	Stop; Y → P			-NA-
40	2	12	m	82	A	m	JKS 1,y,m	° Jump. If Key Set – Stop, then Jump	If key 1 set, stop; Y → P			-NA-
40	2	13	m	82	B	m	JKS 2,y,m	° Jump. If Key Set – Stop, then Jump	If key 2 set, stop; Y → P			-NA-
#40	2	14	m	82	C	m	JSC y,m	Jump Support Channel Busy – RK	If support channel busy; Y → P			-NA-
#40	3	00	m	83	0	m	JE *y,m	Jump Equal	If CC indicates = or 0; (Y) → P			-NA-
#40	3	01	m	83	1	m	JNE *y,m	Jump Not Equal	If CC indicates ≠ or not 0; (Y) → P			-NA-
#40	3	02	m	83	2	m	JGE *y,m	Jump Greater or Equal	If CC indicates ≥ or +; (Y) → P			-NA-

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#40	3	03	m	83	3	m	JLS *y,m	Jump Less	If CC indicates < or -; (Y) → P		-NA-
#40	3	04	m	83	4	m	JO *y,m	Jump on Overflow	If overflow set; (Y) → P		-NA-
#40	3	05	m	83	5	m	JC *y,m	Jump on Carry	If carry set; (Y) → P		-NA-
40	3	06	m	83	6	m	JPT *y,m	Jump if Power out of Tolerance	If power out of tolerance; (Y) → P		-NA-
40	3	07	m	83	7	m	JB *y,m	Jump if Bootstrap 2 Selected	If bootstrap 2 selected; (Y) → P		-NA-
#40	3	10	m	83	8	m	J *y,m	Jump	(Y) → P		-NA-
#40	3	11	m	83	9	m	JS *y,m	°Jump after Stop	Stop; (Y) → P		-NA-
40	3	12	m	83	A	m	JKS 1,*y,m	°Jump. If Key Set – Stop, then Jump	If key 1 set, stop; (Y) → P		-NA-
40	3	13	m	83	B	m	JKS 2,*y,m	°Jump. If Key Set – Stop, then Jump	If key 2 set, stop; (Y) → P		-NA-
#40	3	14	m	83	C	m	JSC *y,m	Jump Support Channel Busy – RX	If support channel busy; (Y) → P		-NA-
#41	0	a	m	84	a	m	XJR a,m	Index Jump Register	If (Ra) ≠ 0; (Ra) - 1 → Ra, (Rm) → P		-NA-
#41	1		d	85		d	LJI x d	Local Jump (Indirect)	[P + xd] → P		-NA-
#41	2	a	m	86	a	m	XJ a,y,m	Index Jump	If (Ra) ≠ 0; (Ra) - 1 → Ra; Y → P		-NA-
#41	3	a	m	87	a	m	XJ a,*y,m	Index Jump	If (Ra) ≠ 0; (Ra) - 1 → Ra; (Y) → P		-NA-
#42	0	a	m	88	a	m	JLRR a,m	Jump, Link Register (Register)	(P) + 1 → Ra; (Rm) → P		-NA-
#42	2	a	m	8A	a	m	JLR a,y,m	Jump, Link Register	(P) + 2 → Ra; Y → P		-NA-
#42	3	a	m	8B	a	m	JLR a,*y,m	Jump, Link Register	(P) + 2 → Ra; (Y) → P		-NA-
#43	1		d	8D		d	LJLM x d	Local Jump, Link Memory	(P) + 1 → P + xD; P + xd + 1 → P		-NA-
#43	2	-	m	8E	-	m	JLM y,m	Jump, Link Memory	(P) + 2 → Y; Y + 1 → P		-NA-
#43	3	-	m	8F	-	m	JLM *y,m	Jump, Link Memory	(P) + 2 → (Y); (Y) + 1 → P		-NA-
# IOP Instructions							°Executive Mode Instructions				

14122000

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

A-10

	Octal Format			Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
#44	0	a	m	90	a	m	JZR a,m	Jump Zero (Register)	If (Ra) = 0; (Rm) → P			-NA-
#44	1		d	91		d	LJE x d	Local Jump Equal	If CC indicates = or 0; (P) + x d → P			-NA-
#44	2	a	m	92	a	m	JZ a,y,m	Jump Zero	If (Ra) = 0; Y → P			-NA-
#44	3	a	m	93	a	m	JZ a,*y,m	Jump Zero	If (Ra) = 0; (Y) → P			-NA-
#45	0	a	m	94	a	m	JNZR a,m	Jump Not Zero (Register)	If (Ra) ≠ 0; (Rm) → P			-NA-
#45	1		d	95		d	LJNE x d	Local Jump Not Equal	If CC indicates ≠ or not 0; (P) + x d → P			-NA-
#45	2	a	m	96	a	m	JNZ a,y,m	Jump Not Zero	If (Ra) ≠ 0; Y → P			-NA-
#45	3	a	m	97	a	m	JNZ a,*y,m	Jump Not Zero	If (Ra) ≠ 0; (Y) → P			-NA-
#46	0	a	m	98	a	m	JPR a,m	Jump Positive (Register)	If (Ra) ≥ 0; (Rm) → P			-NA-
#46	1		d	99		d	LJGE x d	Local Jump Greater or Equal	If CC indicates ≥ or +; (P) + x d → P			-NA-
#46	2	a	m	9A	a	m	JP a,y,m	Jump Positive	If (Ra) ≥ 0; Y → P			-NA-
#46	3	a	m	9B	a	m	JP a,*y,m	Jump Positive	If (Ra) ≥ 0; (Y) → P			-NA-
#47	0	a	m	9C	a	m	JNR a,m	Jump Negative (Register)	If (Ra) < 0; (Rm) → P			-NA-
#47	1		d	9D		d	LJLS x d	Local Jump Less	If CC indicates < or -; (P) + x d → P			-NA-
#47	2	a	m	9E	a	m	JN a,y,m	Jump Negative	If (Ra) < 0; Y → P			-NA-
#47	3	a	m	9F	a	m	JN a,*y,m	Jump Negative	If (Ra) < 0; (Y) → P			-NA-
50	0	a	m	A0	a	m	FSUR a,m	Floating Point Subtract (Register)	(Ra, Ra ⊕ 1) - (Rm, Rm ⊕ 1) → Ra, Ra+1; Res. → Ra+2, Ra+3		X	X
50	1	a	m	A1	a	m	FSUI a,m	Floating Point Subtract (Indirect)	(Ra, Ra ⊕ 1) - (Y*, Y* ⊕ 1) → Ra, Ra+1; Res. → Ra+2, Ra+3		X	X
50	3	a	m	A3	a	m	FSU a,y,m	Floating Point Subtract	(Ra, Ra ⊕ 1) - (Y, Y ⊕ 1) → Ra, Ra+1; Res. → Ra+2, Ra+3		X	X
51	0	a	m	A4	a	m	FAR a,m	Floating Point Add (Register)	(Ra, Ra ⊕ 1) + (Rm, Rm ⊕ 1) → Ra, Ra+1; Res. → Ra+2, Ra+3		X	X
51	1	a	m	A5	a	m	FAI a,m	Floating Point Add (Indirect)	(Ra, Ra ⊕ 1) + (Y*, Y* ⊕ 1) → Ra, Ra+1; Res. → Ra+2, Ra+3		X	X

14122000

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format		Hexadecimal Format		Coding Format	Instruction	Operation	C	OV	CC	
51	3	a	m	A7	a m	FA a,y,m	Floating Point Add	$(Ra, Ra \oplus 1) + (Y, Y \oplus 1) \rightarrow Ra, Ra+1; Res. \rightarrow Ra+2, Ra+3$	X	X
52	0	a	m	A8	a m	FMR a,m	Floating Point Multiply (Register)	$(Ra, Ra \oplus 1) \cdot (Rm, Rm \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	X	X
52	1	a	m	A9	a m	FMI a,m	Floating Point Multiply (Indirect)	$(Ra, Ra \oplus 1) \cdot (Y^*, Y^* \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	X	X
52	3	a	m	AB	a m	FM a,y,m	Floating Point Multiply	$(Ra, Ra \oplus 1) \cdot (Y, Y \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	X	X
53	0	a	m	AC	a m	FDR a,m	Floating Point Divide (Register)	$(Ra, Ra \oplus 1)/(Rm, Rm \oplus 1) \rightarrow Ra, Ra+1; Rem. \rightarrow Ra+2, Ra+3$	X	X
53	1	a	m	AD	a m	FDI a,m	Floating Point Divide (Indirect)	$(Ra, Ra \oplus 1)/(Y^*, Y^* \oplus 1) \rightarrow Ra, Ra+1; Rem. \rightarrow Ra+2, Ra+3$	X	X
53	3	a	m	AF	a m	FD a,y,m	Floating Point Divide	$(Ra, Ra \oplus 1)/(Y, Y \oplus 1) \rightarrow Ra, Ra+1; Rem. \rightarrow Ra+2, Ra+3$	X	X
54	0	a	m	B0	a m	LARR a,m	Load Address Register (Register)	$(Rm) \rightarrow ARr$	-NA-	
54	1	a	m	B1	a m	LARI a,m	Load Address Register (Indirect)	$(Y^*) \rightarrow ARr$	-NA-	
54	3	a	m	B3	a m	LARM a,y,m	Load Address Register Multiple	$(Y, \dots Y + u) \rightarrow ARr \dots ARr+u$	-NA-	
55	0	a	m	B4	a m	SARR a,m	Store Address Register (Register)	$(ARr) \rightarrow Rm$	-NA-	
55	1	a	m	B5	a m	SARI a,m	Store Address Register (Indirect)	$(ARr) \rightarrow Y^*$	-NA-	
55	3	a	m	B7	a m	SARM a,y,m	Store Address Register Multiple	$(ARr, \dots ARr+u) \rightarrow Y, \dots Y+u$	-NA-	
56	0	a	m	B8	a m	MDR a,m	Multiply Double (Register)	$(Ra, Ra \oplus 1) \cdot (Rm, Rm \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	0	0 X
56	1	a	m	B9	a m	MDI a,m	Multiply Double (Indirect)	$(Ra, Ra \oplus 1) \cdot (Y^*, Y^* \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	0	0 X
56	3	a	m	BB	a m	MD a,y,m	Multiply Double	$(Ra, Ra \oplus 1) \cdot (Y, Y \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$	0	0 X

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

	Octal Format		Hexadecimal Format		Coding Format	Instruction	Operation	C	OV	CC
	57	0 a m	BC a m		DDR a,m	Divide Double (Register)	$(Ra, Ra+1, Ra+2, Ra+3)/(Rm, Rm \oplus 1) \rightarrow Ra+2, Ra+3;$ Rem. $\rightarrow Ra, Ra+1$	0	X	X
	57	1 a m	BD a m		DDI a,m	Divide Double (Indirect)	$(Ra, Ra+1, Ra+2, Ra+3)/(Y^*, Y^* \oplus 1) \rightarrow Ra+2, Ra+3;$ Rem. $\rightarrow Ra, Ra+1$	0	X	X
	57	3 a m	BF a m		DD a,y,m	Divide Double	$(Ra, Ra+1, Ra+2, Ra+3)/(Y, Y \oplus 1) \rightarrow Ra+2, Ra+3;$ Rem. $\rightarrow Ra, Ra+1$	0	X	X
#60	0 a m	C0 a m			LLRS a,m	Literal Logical Right Shift	Shift (Ra) right m places, zero fill	0	0	X
#60	1 a m	C1 a m			LARS a,m	Literal Algebraic Right Shift	Shift (Ra) right m places, sign fill	0	0	X
60	2 a m	C2 a m			LLRD a,m	Literal Logical Right Double Shift	Shift (Ra, Ra \oplus 1) right m places, zero fill	0	0	X
60	3 a m	C3 a m			LARD a,m	Literal Algebraic Right Double Shift	Shift (Ra, Ra \oplus 1) right m places, zero fill	0	0	X
#61	0 a m	C4 a m			LALS a,m	Literal Algebraic Left Shift	Shift (Ra) left m places, zero fill	0	X	X
#61	1 a m	C5 a m			LCLS a,m	Literal Circular Left Shift	Shift (Ra) left circular m places	0	0	X
61	2 a m	C6 a m			LALD a,m	Literal Algebraic Left Double Shift	Shift (Ra, Ra \oplus 1) left m places, zero fill	0	X	X
61	3 a m	C7 a m			LCLD a,m	Literal Circular Left Double Shift	Shift (Ra, Ra \oplus 1) left circular m places	0	0	X
#62	0 a m	C8 a m			LSU a,m	Literal Subtract	$(Ra) - m \rightarrow Ra$	X	X	X
62	1 a m	C9 a m			LSUD a,m	Literal Subtract Double	$(Ra, Ra \oplus 1) - m \rightarrow Ra, Ra \oplus 1$	X	X	X
#62	2 a m	CA a m			LA a,m	Literal Add	$(Ra) + m \rightarrow Ra$	X	X	X
62	3 a m	CB a m			LAD a,m	Literal Add Double	$(Ra, Ra \oplus 1) + m \rightarrow Ra, Ra \oplus 1$	X	X	X
#63	0 a m	CC a m			LL a,m	Literal Load	$m \rightarrow Ra$	0	0	X
#63	1 a m	CD a m			LC a,m	Literal Compare	$(Ra) - m$	X	X	X
# IOP Instructions						^o Executive Mode Instructions				

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format				Hexadecimal Format			Coding Format	Instruction	Operation	C	OV	CC
63	2	a	m	CE	a	m	LMUL a,m	Literal Multiply	$(Ra \oplus 1) \cdot m \rightarrow Ra, Ra \oplus 1$	0	0	X
63	3	a	m	CF	a	m	LDIV a,m	Literal Divide	$(Ra, Ra \oplus 1)/m \rightarrow Ra \oplus 1; remainder \rightarrow Ra$	0	X	X
64	3	a	m	D3	a	m	BSU a,y,m	Byte Subtract	$(Ra) - (Y) \text{ byte} \rightarrow Ra$	X	X	X
65	3	a	m	D7	a	m	BA a,y,m	Byte Add	$(Ra) + (Y) \text{ byte} \rightarrow Ra$	X	X	X
66	3	a	m	DB	a	m	BC a,y,m	Byte Compare	$(Ra) - (Y) \text{ byte}$	X	X	X
67	3	a	m	DF	a	m	BCX a,y,m	Byte Compare and Index by 1	$(Ra) - (Y) \text{ byte}; (Rm) + 1 \rightarrow Rm$	X	X	X

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

A-14

Octal Format		Hexadecimal Format		Coding Format		Instruction	Operation	C	OV	CC
COMMAND/CHAIN INSTRUCTION										
#70	0 00 00	E0 0 0	ACR	Channel Control	Master clear all channels					
#70	0 00 04	E0 0 4	ACR 4 CCR 0,4	Channel Control	Enable external interrupts, all channels					
#70	0 00 05	E0 0 5	ACR 5 CCR 0,5	Channel Control	Disable external interrupts, all channels					
#70	0 00 06	E0 0 6	ACR 6 CCR 0,6	Channel Control	Enable Class III Interrupts, Priorities 2, 3, 4					
#70	0 00 07	E0 0 7	ACR 7 CCR 0,7	Channel Control	Disable Class III Interrupts, Priorities 2, 3, 4					
#70	0 a 10	E0 a 8	CCR a,8	Channel Control	Master clear channel a					
#70	0 a 14	E0 a C	CCR a,12	Channel Control	Enable channel a external interrupts					
#70	0 a 15	E0 a D	CCR a,13	Channel Control	Disable channel a external interrupts					
#70	0 a 16	E0 a E	CCR a,14	Channel Control	Enable channel a, Class III, priorities 2, 3, 4					
#70	0 a 17	E0 a F	CCR a,15	Channel Control	Disable channel a, Class III, priorities 2, 3, 4					
COMMAND INSTRUCTION										
#71	2 a 02	E6 a 2	ICK a,y	Initiate Input Chain	Y → Channel a Chain Pointer; initiate input chain					
#71	2 a 06	E6 a 6	OCK a,y	Initiate Output Chain	Y → Channel a Chain Pointer; initiate output chain					
#71	3 a m	E7 a m	WIM a,y,m	Write Control Memory	(Y) → Channel a CMm					
#72	3 a m	EB a m	RIM a,y,m	Read Control Memory	Channel a (CMm) → Y					
#76	0 a m	F8 a m	SICR a,m	Set and Clear Discretes	Set or clear channel a discrete function					
#76	3 a m	FB a m	SST a,y,m	Store Status	Channel a Status bits per m → Y					
#77	0 - m	FC - m	SIOP m,y	Start IOP ③	m:0 → IOP SR1:12, Y → IOP P if m = 0 or 1					
#77	2 a m	FE a m	XIM a,y,m	Exchange Control Memory ②	Channel a (CMm) → Y; (Y ⊕ 1) → Channel a CMm					

TABLE A-1. INSTRUCTION REPERTOIRE BY OP-CODE (Cont.)

Octal Format	Hexadecimal Format	Coding Format	Instruction	Operation	C	OV	CC
CHAIN INSTRUCTION							
#70	2 a m	E2 a m	IM a,y,m	Initiate Message	Y → CMm; Initiate message activity		
#70	3 a 00	E3 a 0	IO a,y	IO Function a	(Y, Y ⊕ 1) → BCW, BAP; initiate transfer		
#71	2 00 m	E6 0 m	LCMK m,y	Load Control Memory	Y → CMm		
#71	3 00 m	E7 0 m	LCM m,y	Load Control Memory	(Y) → CMm		
#72	3 00 m	EB 0 m	SCM m,y	Store Control Memory	(CMm) → Y		
#73	0 00 00	EC 0 0	HCR	Halt Chain	Halt chaining, a even		
#73	0 01 00	EC 1 0	IPR	Interrupt Processor	Generate chain interrupt, a odd		
#73	3 00 00	EF 0 0	ZF y	Zero Flag	0 → Y15,14, a even		
#73	3 01 00	EF 1 0	SF y	Set Flag	1 → Y15,14, a odd		
#74	2 00 00	F2 0 0	SJMC 0,y	Serial Jump on Met Condition	Unconditional Y → CAP; clear flag		
#74	2 01 00	F2 1 0	SJMC 1,y	Serial Jump on Met Condition	If suppress flag not set, Y → CAP; clear flag		
#74	2 02 00	F2 2 0	SJMC 2,y	Serial Jump on Met Condition	If monitor flag set, Y → CAP; clear flag		
#75	0 00 m	F4 0 m	SFSC m	Search For Sync	Perform function(s) assigned to m-bits		
#76	0 00 m	F8 0 m	CSIR m	Serial Interface Control	Set or clear discrete function		
#76	3 - m	FB - m	CSST y,m	Store Status	Status bits per m → Y		
#77	1 - m	FD - m	BJ m,y	Bit Jump	(Y) → CAP if (CM3):m = 1		
#77	2 - m	FE - m	XCM m,y	Exchange Control Memory	(CMm) → Y; (Y ⊕ 1) → CMm		
# IOP Instructions			② m = 2 or 6	③ no operation unless IOP			

14122000

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC

MNEMONIC	INSTRUCTION CODE		FORMAT	INSTRUCTION
	HEX	o f a m		
ARITHMETIC				
A	a,y,m	48 a m	22 3 a m	RX ADD, (Ra)+(Y) → Ra; Set CC, OV, C
AD	a,y,m	4F a m	23 3 a m	RX ADD DOUBLE, (Ra,Ra⊕1)+(Y,Y⊕1) → Ra,Ra⊕1; Set CC, OV, C
ADI	a,m	4D a m	23 1 a m	RI-2 ADD DOUBLE, (Ra,Ra⊕1)+(Y*,Y*⊕1) → Ra, Ra⊕1; Set CC, OV, C
ADR	a,m	4C a m	23 0 a m	RR ADD DOUBLE, (Ra, Ra⊕1)+(Rm,Rm⊕1) → Ra,Ra⊕1; Set CC, OV, C
AI	a,m	49 a m	22 1 a m	RI-2 ADD, (Ra)+(Y*) → Ra; Set CC, OV, C
AK	a,y,m	4A a m	22 2 a m	RK ADD, (Ra)+Y → Ra; Set CC, OV, C
AR	a,m	48 a m	22 0 a m	RR ADD, (Ra)+(Rm) → Ra; Set CC, OV, C
BA	a,y,m	D7 a m	65 3 a m	RX BYTE ADD, (Ra)+(Y)byte → Ra; Set CC, OV, C
BSU	a,y,m	D3 a m	64 3 a m	RX BYTE SUBTRACT, (Ra)-(Y)byte → Ra; Set CC, OV, C
D	a,y,m	5F a m	27 3 a m	RX DIVIDE, (Ra, Ra ⊕ 1)/(Y) → Ra ⊕ 1; Remainder → Ra; Set CC, OV
DD	a,y,m	8F a m	57 3 a m	RX DIVIDE DOUBLE, (Ra,Ra+1,Ra+2,Ra+3)/(Y,Y ⊕ 1) → Ra+2, Ra+3; Rem → Ra,Ra+1; Set CC, OV
DDI	a,m	8D a m	57 1 a m	RI-2 DIVIDE DOUBLE, (Ra,Ra+1,Ra+2,Ra+3)/(Y*,Y* ⊕ 1) → Ra+2, Ra+3; Rem → Ra,Ra+1; Set CC, OV
DDR	a,m	8C a m	57 0 a m	RR DIVIDE DOUBLE, (Ra,Ra+1,Ra+2,Ra+3)/(Rm,Rm ⊕ 1) → Ra+2,Ra+3; Rem → Ra,Ra+1; Set CC, OV
DI	a,m	5D a m	27 1 a m	RI-2 DIVIDE, (Ra,Ra ⊕ 1)/(Y*) → Ra ⊕ 1; Remainder → Ra; Set CC, OV
DK	a,y,m	5E a m	27 2 a m	RK DIVIDE, (Ra,Ra ⊕ 1)/Y → Ra ⊕ 1; Remainder → Ra; Set CC, OV
DR	a,m	5C a m	27 0 a m	RR DIVIDE, (Ra,Ra ⊕ 1)/(Rm) → Ra ⊕ 1; Remainder → Ra; Set CC, OV
DROR	a	08 a 9	02 0 a 11	RR DECREMENT RA BY 1, (Ra)-1 → Ra; Set CC, OV, C
DRTR	a	08 a 8	02 0 a 13	RR DECREMENT RA BY 2, (Ra)-2 → Ra; Set CC, OV, C
FA	a,y,m	A7 a m	51 3 a m	RX FLOATING POINT ADD, (Ra,Ra ⊕ 1)+(Y,Y ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FAL	a,m	A5 a m	51 1 a m	RI-2 FLOATING POINT ADD, (Ra,Ra ⊕ 1)+(Y*,Y* ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FAR	a,m	A4 a m	51 0 a m	RR FLOATING POINT ADD, (Ra,Ra ⊕ 1)+(Rm,Rm ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FD	a,y,m	AF a m	53 3 a m	RX FLOATING POINT DIVIDE, (Ra,Ra ⊕ 1)/(Y,Y ⊕ 1) → Ra,Ra+1; Rem → Ra+2,Ra+3; Set CC, OV
FDI	a,m	AD a m	53 1 a m	RI-2 FLOATING POINT DIVIDE, (Ra,Ra ⊕ 1)/(Y*,Y* ⊕ 1) → Ra, Ra+1; Rem → Ra+2,Ra+3; Set CC, OV
FDR	a,m	AC a m	53 0 a m	RR FLOATING POINT DIVIDE, (Ra,Ra ⊕ 1)/(Rm,Rm ⊕ 1) → Ra, Ra+1; Rem → Ra+2,Ra+3; Set CC, OV
FM	a,y,m	AB a m	52 3 a m	RX FLOATING POINT MULTIPLY, (Ra,Ra ⊕ 1)*(Y,Y ⊕ 1) → Ra, Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FMI	a,m	A9 a m	52 1 a m	RI-2 FLOATING POINT MULTIPLY, (Ra,Ra ⊕ 1)*(Y*,Y* ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FMR	a,m	A8 a m	52 0 a m	RR FLOATING POINT MULTIPLY, (Ra,Ra ⊕ 1)*(Rm,Rm ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FSU	a,y,m	A3 a m	50 3 a m	RX FLOATING POINT SUBTRACT, (Ra,Ra ⊕ 1)-(Y,Y ⊕ 1) → Ra,Ra+1; Res → Ra+2,Ra+3; Set CC, OV
FSUI	a,m	A1 a m	50 1 a m	RI-2 FLOATING POINT SUBTRACT, (Ra,Ra ⊕ 1)-(Y*,Y* ⊕ 1) → Ra,Ra+1, Res → Ra+2,Ra+3; Set CC, OV
FSUR	a,m	A0 a m	50 0 a m	RR FLOATING POINT SUBTRACT, (Ra,Ra ⊕ 1)-(Rm,Rm ⊕ 1) → Ra,Ra+1, Res → Ra+2,Ra+3; Set CC, OV
IROR	a	08 a 8	02 0 a 10	RR INCREMENT RA BY 1, (Ra)+1 → Ra; Set CC, OV, C
IRTR	a	08 a A	02 0 a 12	RR INCREMENT RA BY 2, (Ra)+2 → Ra; Set CC, OV, C
LA	a,m	CA a m	52 2 a m	RL LITERAL ADD, (Ra)+m → Ra; Set CC, OV, C
LAD	a,m	C2 a m	62 3 a m	RL LITERAL ADD DOUBLE, (Ra,Ra ⊕ 1)+m → Ra,Ra ⊕ 1; Set CC, OV, C
LDIV	a,m	CF a m	63 3 a m	RL LITERAL DIVIDE, (Ra,Ra ⊕ 1)/m → Ra ⊕ 1, Rem → Ra; Set CC, OV
LMUL	a,m	CE a m	63 2 a m	RL LITERAL MULTIPLY, (Ra ⊕ 1)*m → Ra,Ra ⊕ 1; Set CC
LSU	a,m	C2 a m	62 0 a m	RL LITERAL SUBTRACT, (Ra)-m → Ra; Set CC, OV, CC
LSUD	a,m	C9 a m	62 1 a m	RL LITERAL SUBTRACT DOUBLE, (Ra,Ra ⊕ 1)-m → Ra,Ra⊕1; Set CC, OV, C
M	a,y,m	5B a m	26 3 a m	RX MULTIPLY, (Ra ⊕ 1)*(Y) → Ra,Ra ⊕ 1; Set CC
MD	a,y,m	8B a m	56 3 a m	RX MULTIPLY DOUBLE, (Ra,Ra ⊕ 1)*(Y,Y ⊕ 1) → Ra,Ra+1,Ra+2,Ra+3; Set CC
MDI	a,m	89 a m	56 1 a m	RI-2 MULTIPLY DOUBLE, (Ra,Ra ⊕ 1)*(Y*,Y* ⊕ 1) → Ra,Ra+1,Ra+2,Ra+3; Set CC

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MNEMONIC	INSTRUCTION CODE		FORMAT	INSTRUCTION
	HEX	o f a m		
ARITHMETIC (CONT)				
MDR	a,m	88 a m	56 0 a m	RR MULTIPLY DOUBLE, $(Ra, Ra \oplus 1) * (Rm, Rm \oplus 1) \rightarrow Ra, Ra+1, Ra+2, Ra+3$; Set CC
MI	a,m	59 a m	26 1 a m	RI-2 MULTIPLY, $(Ra \oplus 1) * (Y^*) \rightarrow Ra, Ra \oplus 1$; Set CC
MK	a,y,m	5A a m	26 2 a m	RK MULTIPLY, $(Ra \oplus 1) * Y \rightarrow Ra, Ra \oplus 1$; Set CC
MR	a,m	58 a m	26 0 a m	RR MULTIPLY, $(Ra \oplus 1) * (Rm) \rightarrow Ra, Ra \oplus 1$; Set CC
NR	a	08 a 1	02 0 a 01	RR MAKE NEGATIVE, if $(Ra) > 0$, $(Ra)' \rightarrow Ra$; if $(Ra) \leq 0$, (Ra) Unchanged; Set CC and C
OCR	a	08 a 6	02 0 a 06	RR ONE'S COMPLEMENT SINGLE, Bit by Bit Comp of $(Ra) \rightarrow Ra$; Set CC
PR	a	08 a 0	02 0 a 00	RR MAKE POSITIVE, if $(Ra) < 0$, $(Ra)' \rightarrow Ra$; if $(Ra) \geq 0$, (Ra) Unchanged; Set CC, OV, C
RR	a	08 a 2	02 0 a 02	RR ROUND RA, if $(Ra) \geq 0$, $(Ra) + (Ra+1) : 15 \rightarrow Ra$; if $(Ra) < 0$, $(Ra) - (Ra+1) : 15 \rightarrow Ra$; Set CC, OV, C
SU	a,y,m	43 a m	20 3 a m	RX SUBTRACT, $(Ra) - (Y) \rightarrow Ra$; Set CC, OV, C
SUD	a,y,m	47 a m	21 3 a m	RX SUBTRACT DOUBLE, $(Ra, Ra \oplus 1) - (Y, Y \oplus 1) \rightarrow Ra, Ra \oplus 1$; Set CC, OV, C
SUDI	a,m	45 a m	21 1 a m	RI-2 SUBTRACT DOUBLE, $(Ra, Ra \oplus 1) - (Y^*, Y^* \oplus 1) \rightarrow Ra, Ra \oplus 1$; Set CC, OV, C
SUDR	a,m	44 a m	21 0 a m	RR SUBTRACT DOUBLE, $(Ra, Ra \oplus 1) - (Rm, Rm \oplus 1) \rightarrow Ra, Ra \oplus 1$; Set CC, OV, C
SUI	a,m	41 a m	20 1 a m	RI-2 SUBTRACT, $(Ra) - (Y^*) \rightarrow Ra$; Set CC, OV, C
SUK	a,y,m	42 a m	20 2 a m	RK SUBTRACT, $(Ra) - Y \rightarrow Ra$; Set CC, OV, C
SUR	a,m	40 a m	20 0 a m	RR SUBTRACT, $(Ra) - (Rm) \rightarrow Ra$; Set CC, OV, C
TCDR	a	08 a 5	02 0 a 05	RR TWO'S COMPLEMENT DOUBLE, $(Ra, Ra \oplus 1)' \rightarrow Ra, Ra \oplus 1$; Set CC, OV, C
TCR	a	08 a 4	02 0 a 04	RR TWO'S COMPLEMENT SINGLE, $(Ra)' \rightarrow Ra$; Set CC, OV, C
LOGICAL				
AND	a,y,m	63 a m	30 3 a m	RX AND, $(Ra) \odot (Y) \rightarrow Ra$; Set CC
ANDI	a,m	61 a m	30 1 a m	RI-2 AND, $(Ra) \odot (Y^*) \rightarrow Ra$; Set CC
ANDK	a,y,m	62 a m	30 2 a m	RK AND, $(Ra) \odot Y \rightarrow Ra$; Set CC
ANDR	a,m	60 a m	30 0 a m	RR AND, $(Ra) \odot (Rm) \rightarrow Ra$; Set CC
OR	a,y,m	67 a m	31 3 a m	RX OR, $(Ra) \oplus (Y) \rightarrow Ra$; Set CC
ORI	a,m	65 a m	31 1 a m	RI-2 OR, $(Ra) \oplus (Y^*) \rightarrow Ra$; Set CC
ORK	a,y,m	66 a m	31 2 a m	RK OR, $(Ra) \oplus Y \rightarrow Ra$; Set CC
ORR	a,m	64 a m	31 0 a m	RR OR, $(Ra) \oplus (Rm) \rightarrow Ra$; Set CC
XOR	a,y,m	68 a m	32 3 a m	RX EXCLUSIVE OR, $(Ra) \oplus (Y) \rightarrow Ra$; Set CC
XORI	a,m	69 a m	32 1 a m	RI-2 EXCLUSIVE OR, $(Ra) \oplus (Y^*) \rightarrow Ra$; Set CC
XORK	a,y,m	6A a m	32 2 a m	RK EXCLUSIVE OR, $(Ra) \oplus Y \rightarrow Ra$; Set CC
XORR	a,m	68 a m	32 0 a m	RR EXCLUSIVE OR, $(Ra) \oplus (Rm) \rightarrow Ra$; Set CC
COMPARE				
BC	a,y,m	DB a m	66 3 a m	RX BYTE COMPARE, $(Ra) : (Y)$ byte; Set CC, OV, C
BCX	a,y,m	DF a m	67 3 a m	RX BYTE COMPARE AND INDEX BY 1, $(Ra) : (Y)$ byte; $(Rm)+1 \rightarrow Rm$; Set CC, OV, C
C	a,y,m	53 a m	24 3 a m	RX COMPARE, $(Ra) : (Y)$; Set CC, OV, C
CBR	a,m	1C a m	07 0 a m	RR COMPARE BIT, Bit position m of (Ra) Tested; Set CC
CD	a,y,m	57 a m	25 3 a m	RX COMPARE DOUBLE, $(Ra, Ra \oplus 1) : (Y, Y \oplus 1)$; Set CC, OV, C
CDI	a,m	55 a m	25 1 a m	RI-2 COMPARE DOUBLE, $(Ra, Ra \oplus 1) : (Y^*, Y^* \oplus 1)$; Set CC, OV, C
CDR	a,m	54 a m	25 0 a m	RR COMPARE DOUBLE, $(Ra, Ra \oplus 1) : (Rm, Rm \oplus 1)$; Set CC, OV, C
CI	a,m	51 a m	24 1 a m	RI-2 COMPARE, $(Ra) : (Y^*)$; Set CC, OV, C
CK	a,y,m	52 a m	24 2 a m	RK COMPARE, $(Ra) : Y$; Set CC, OV, C
CM	a,y,m	73 a m	34 3 a m	RX COMPARE MASKED, $[(Ra) \odot (Ra \oplus 1)] : [(Y) \odot (Ra \oplus 1)]$; Set CC
CMI	a,m	71 a m	34 1 a m	RI-2 COMPARE MASKED, $[(Ra) \odot (Ra \oplus 1)] : [(Y^*) \odot (Ra \oplus 1)]$; Set CC
CMK	a,y,m	72 a m	34 2 a m	RK COMPARE MASKED, $[(Ra) \odot (Ra \oplus 1)] : [(Y) \odot (Ra \oplus 1)]$; Set CC
CMR	a,m	70 a m	34 0 a m	RR COMPARE MASKED, $[(Ra) \odot (Ra \oplus 1)] : [(Rm) \odot (Ra \oplus 1)]$; Set CC
CR	a,m	50 a m	24 0 a m	RR COMPARE, $(Ra) : (Rm)$; Set CC, OV, C
LC	a,m	CD a m	63 1 a m	RL LITERAL COMPARE, $(Ra) : m$; Set CC, OV, C

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MNEMONIC	INSTRUCTION CODE		FORMAT	INSTRUCTION	
	HEX	o f a m			
JUMPS					
J	y,m	82 8 m	40 2 10 m	RK	JUMP, Y → P
J	*y,m	83 8 m	40 3 10 m	RX	JUMP, (Y) → P
JB	y,m	82 7 m	40 2 07 m	RK	JUMP BOOTSTRAP 2 SELECTED, If Bootstrap 2 Selected, Y → P
JB	*y,m	83 7 m	40 3 07 m	RX	JUMP BOOTSTRAP 2 SELECTED, If Bootstrap 2 Selected, (Y) → P
JBR	m	80 7 m	40 0 07 m	RR	JUMP BOOTSTRAP 2 SELECTED, If Bootstrap 2 Selected, (Rm) → P
JC	y,m	82 5 m	40 2 05 m	RK	JUMP CARRY, If Carry Set, Y → P
JC	*y,m	83 5 m	40 3 05 m	RX	JUMP CARRY, If Carry Set, (Y) → P
JCR	m	80 5 m	40 0 05 m	RR	JUMP CARRY, If Carry Set, (Rm) → P
JE	y,m	82 0 m	40 2 00 m	RK	JUMP EQUAL, If (CC) indicates =, Y → P
JE	*y,m	83 0 m	40 3 00 m	RX	JUMP EQUAL, If (CC) indicates =, (Y) → P
JER	m	80 0 m	40 0 00 m	RR	JUMP EQUAL, If (CC) indicates =, (Rm) → P
JGE	y,m	82 2 m	40 2 02 m	RK	JUMP GREATER OR EQUAL, If (CC) indicates ≥, Y → P
JGE	*y,m	83 2 m	40 3 02 m	RX	JUMP GREATER OR EQUAL, If (CC) indicates ≥, (Y) → P
JGER	m	80 2 m	40 0 02 m	RR	JUMP GREATER OR EQUAL, If (CC) indicates ≥, (Rm) → P
JLM	y,m	8E - m	43 2 - m	RX	JUMP, LINK MEMORY, (P)+2 → Y; Y+1 → P
JLM	*y,m	8F - m	43 3 - m	RX	JUMP, LINK MEMORY, (P)+2 → (Y); (Y)+1 → P
JLR	a,y,m	8A a m	42 2 a m	RK	JUMP, LINK REGISTER, (P)+2 → Ra; Y → P
JLR	a,*y,m	8B a m	42 3 a m	RX	JUMP, LINK REGISTER, (P)+2 → Ra; (Y) → P
JLRR	a,m	88 a m	42 0 a m	RR	JUMP, LINK REGISTER, (P)+1 → Ra; (Rm) → P
JLS	y,m	82 3 m	40 2 03 m	RK	JUMP LESS, If (CC) indicates <, Y → P
JLS	*y,m	83 3 m	40 3 03 m	RX	JUMP LESS, If (CC) indicates <, (Y) → P
JLSR	m	80 3 m	40 0 03 m	RR	JUMP LESS, If (CC) indicates <, (Rm) → P
JN	a,y,m	9E a m	47 2 a m	RK	JUMP NEGATIVE, If (Ra) < 0, Y → P
JN	a,*y,m	9F a m	47 3 a m	RX	JUMP NEGATIVE, If (Ra) < 0, (Y) → P
JNE	y,m	82 1 m	40 2 01 m	RK	JUMP NOT EQUAL, If (CC) indicates ≠, Y → P
JNE	*y,m	83 1 m	40 3 01 m	RX	JUMP NOT EQUAL, If (CC) indicates ≠, (Y) → P
JNER	m	80 1 m	40 0 01 m	RR	JUMP NOT EQUAL, If (CC) indicates ≠, (Rm) → P
JNR	a,m	9C a m	47 0 a m	RR	JUMP NEGATIVE, If (Ra) < 0, (Rm) → P
JNZ	a,y,m	96 a m	45 2 a m	RK	JUMP NOT ZERO, If (Ra) ≠ 0, Y → P
JNZ	a,*y,m	97 a m	45 3 a m	RX	JUMP NOT ZERO, If (Ra) ≠ 0, (Y) → P
JNZR	a,m	94 a m	45 0 a m	RR	JUMP NOT ZERO, If (Ra) ≠ 0, (Rm) → P
JO	y,m	82 4 m	40 2 04 m	RK	JUMP OVERFLOW, If Overflow Set, Y → P
JO	*y,m	83 4 m	40 3 04 m	RX	JUMP OVERFLOW, If Overflow Set, (Y) → P
JOR	m	80 4 m	40 0 04 m	RR	JUMP OVERFLOW, If Overflow Set, (Rm) → P
JP	a,y,m	9A a m	46 2 a m	RK	JUMP POSITIVE, If (Ra) ≥ 0, Y → P
JP	a,*y,m	9B a m	46 3 a m	RX	JUMP POSITIVE, If (Ra) ≥ 0, (Y) → P
JPR	a,m	98 a m	46 0 a m	RR	JUMP POSITIVE, If (Ra) ≥ 0, (Rm) → P
JPT	y,m	82 6 m	40 2 06 m	RK	JUMP POWER OUT OF TOLERANCE, If Power Out of Tolerance, Y → P
JPT	*y,m	83 6 m	40 3 06 m	RX	JUMP POWER OUT OF TOLERANCE, If Power Out of Tolerance, (Y) → P
JPTR	m	80 6 m	40 0 06 m	RR	JUMP POWER OUT OF TOLERANCE, If Power Out of Tolerance, (Rm) → P
JR	m	80 8 m	40 0 10 m	RR	JUMP (Rm) → P
JZ	a,y,m	92 a m	44 2 a m	RK	JUMP ZERO, if (Ra) = 0, Y → P
JZ	a,*y,m	93 a m	44 3 a m	RX	JUMP ZERO, If (Ra) = 0, (Y) → P
JZR	a,m	90 a m	44 0 a m	RR	JUMP ZERO, If (Ra) = 0, (Rm) → P
LJ	d	81 d	40 1 d	RI-1	LOCAL JUMP, (P)+d → P
LJE	d	91 d	44 1 d	RI-1	LOCAL JUMP EQUAL, If (CC) indicates =, (P)+d → P
LJGE	d	99 d	46 1 d	RI-1	LOCAL JUMP GREATER OR EQUAL, If (CC) indicates ≥, (P)+d → P

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MNEMONIC	INSTRUCTION CODE			FORMAT	INSTRUCTION
	HEX	a	f		
JUMPS (CONT)					
LJI	d	85 d	41 1 d	RI-1	LOCAL JUMP INDIRECT, ((P)+d) → P
LJLM	d	8D d	43 1 d	RI-1	LOCAL JUMP, LINK MEMORY, (P)+1 → P+d; (P)+d+1 → P
LJLS	d	9D d	47 1 d	RI-1	LOCAL JUMP LESS, If (CC) indicates <, (P)+d → P
LINE	d	95 d	45 1 d	RI-1	LOCAL JUMP NOT EQUAL, If (CC) indicates ≠, (P)+d → P
XJ	a,y,m	86 a m	41 2 a m	RK	INDEX JUMP, If (Ra) ≠ 0, (Ra)-1 → Ra, Y → P
XJ	a,*y,m	87 a m	41 3 a m	RX	INDEX JUMP, If (Ra) ≠ 0, (Ra)-1 → Ra, (Y) → P
XJR	a,m	84 a m	41 0 a m	RR	INDEX JUMP, If (Ra) ≠ 0, (Ra)-1 → Ra, (Rm) → P
SHIFTS					
ALD	a,y,m	3A a m	16 2 a m	RK	ALGEBRAIC LEFT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Left Y:5-0 Places, Zero Fill; Set CC, OV; Sign Not Retained
ALDR	a,m	38 a m	16 0 a m	RR	ALGEBRAIC LEFT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Left (Rm): 5-0 Places, Zero Fill; Set CC, OV; Sign Not Retained
ALS	a,y,m	32 a m	14 2 a m	RK	ALGEBRAIC LEFT SINGLE SHIFT, Shift (Ra) Left Y:5-0 Places, Zero Fill; Set CC, OV; Sign Not Retained
ALSR	a,m	30 a m	14 0 a m	RR	ALGEBRAIC LEFT SINGLE SHIFT, Shift (Ra) Left (Rm):5-0 Places, Zero Fill; Set CC, OV; Sign Not Retained
ARD	a,y,m	2E a m	13 2 a m	RK	ALGEBRAIC RIGHT DOUBLE SHIFT, Shift (Ra, Ra ⊕ 1) Right Y:5-0 Places, Sign Fill; Set CC
ARDR	a,m	2C a m	13 0 a m	RR	ALGEBRAIC RIGHT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Right (Rm): 5-0 Places, Sign Fill; Set CC
ARS	a,y,m	26 a m	11 2 a m	RK	ALGEBRAIC RIGHT SINGLE SHIFT, Shift (Ra) Right Y:5-0 Places, Sign Fill; Set CC
ARSR	a,m	24 a m	11 0 a m	RR	ALGEBRAIC RIGHT SINGLE SHIFT, Shift (Ra) Right (Rm):5-0 Places, Sign Fill; Set CC
CLD	a,y,m	3E a m	17 2 a m	RK	CIRCULAR LEFT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Left Circularly Y:5-0 Places; Set CC; Sign Not Retained
CLDR	a,m	3C a m	17 0 a m	RR	CIRCULAR LEFT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Left Circularly (Rm):5-0 Places; Set CC; Sign Not Retained
CLS	a,y,m	36 a m	15 2 a m	RK	CIRCULAR LEFT SINGLE SHIFT, Shift (Ra) Left Circularly Y:5-0 Places; Set CC; Sign Not Retained
CLSR	a,m	34 a m	15 0 a m	RR	CIRCULAR LEFT SINGLE SHIFT, Shift (Ra) Left Circularly (Rm):5-0 Places; Set CC; Sign Not Retained
LALD	a,m	C6 a m	61 2 a m	RL	LITERAL ALGEBRAIC LEFT DOUBLE SHIFT, Left Shift (Ra,Ra ⊕ 1) n Places, Zero Fill; Set CC, OV Sign Not Retained
LALS	a,m	C4 a m	61 0 a m	RL	LITERAL ALGEBRAIC LEFT SINGLE SHIFT, Left Shift (Ra) n Places, Zero Fill; Set CC, OV; Sign Not Retained
LARD	a,m	C3 a m	60 3 a m	RL	LITERAL ALGEBRAIC RIGHT DOUBLE SHIFT, Right Shift (Ra,Ra ⊕ 1) n Places, Sign Fill; Set CC
LARS	a,m	C1 a m	60 1 a m	RL	LITERAL ALGEBRAIC RIGHT SINGLE SHIFT, Right Shift (Ra) n Places, Sign Fill; Set CC
LCLD	a,m	C7 a m	61 3 a m	RL	LITERAL CIRCULAR LEFT DOUBLE SHIFT, Circular Left Shift (Ra, Ra ⊕ 1) n Places; Set CC; Sign Not Retained
LCLS	a,m	C5 a m	61 1 a m	RL	LITERAL CIRCULAR LEFT SINGLE SHIFT, Circular Left Shift (Ra) n Places; Set CC; Sign Not Retained
LLRD	a,m	C2 a m	60 2 a m	RL	LITERAL LOGICAL RIGHT DOUBLE SHIFT, Right Shift (Ra,Ra ⊕ 1) n Places, Zero Fill; Set CC; Sign Not Retained
LLRS	a,m	C0 a m	60 0 a m	RL	LITERAL LOGICAL RIGHT SINGLE SHIFT, Right Shift (Ra) n Places, Zero Fill; Set CC; Sign Not Retained
LRD	a,y,m	2A a m	12 2 a m	RK	LOGICAL RIGHT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Right Y:5-0 Places, Zero Fill; Set CC; Sign Not Retained
LRDR	a,m	28 a m	12 0 a m	RR	LOGICAL RIGHT DOUBLE SHIFT, Shift (Ra,Ra ⊕ 1) Right (Rm):5-0 Places, Zero Fill; Set CC; Sign Not Retained
LRS	a,y,m	22 a m	10 2 a m	RK	LOGICAL RIGHT SINGLE SHIFT, Shift (Ra) Right Y:5-0 Places, Zero Fill; Set CC; Sign Not Retained
LRSR	a,m	20 a m	10 0 a m	RR	LOGICAL RIGHT SINGLE SHIFT, Shift (Ra) Right (Rm):5-0 Places, Zero Fill; Set CC, Sign Not Retained

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MINEMONIC	INSTRUCTION CODE		FORMAT	INSTRUCTION	
	HEX	o f a m			
LOADS/STORES					
BL	a,y,m	03 a m	00 3 a m	RX	BYTE LOAD, Selected (Y) byte → Ra Bits 0-7; Set CC
BLX	a,y,m	13 a m	04 3 a m	RX	BYTE LOAD AND INDEX BY 1, Selected (Y) byte → Ra; Bits 0-7; (Rm)+1 → Rm if a ≠ m; Set CC
BS	a,y,m	23 a m	10 3 a m	RX	BYTE STORE, (Ra) Bits 0-7 → Selected Y byte
BSX	a,y,m	33 a m	14 3 a m	RX	BYTE STORE AND INDEX BY 1, (Ra) Bits 7-0 → Selected Ybyte; (Rm)+1 → Rm
L	a,y,m	07 a m	01 3 a m	RX	LOAD, (Y) → Ra; Set CC
LD	a,y,m	0B a m	02 3 a m	RX	LOAD DOUBLE, (Y, Y ⊕ 1) → Ra, Ra ⊕ 1; Set CC
LDI	a,m	09 a m	02 1 a m	RI-2	LOAD DOUBLE, (Y*, Y* ⊕ 1) → Ra, Ra ⊕ 1; Set CC
LDX	a,y,m	1B a m	06 3 a m	RX	LOAD DOUBLE AND INDEX BY 2, (Y, Y ⊕ 1) → Ra, Ra ⊕ 1; (Rm)+2 → Rm if a ≠ m; Set CC
LDXI	a,m	19 a m	06 1 a m	RI-2	LOAD DOUBLE AND INDEX BY 2, (Y*, Y* ⊕ 1) → Ra, Ra ⊕ 1; (Rm)+2 → Rm if a ≠ m; Set CC
LI	a,m	05 a m	01 1 a m	RI-2	LOAD, (Y*) → Ra; Set CC
LK	a,y,m	06 a m	01 2 a m	RK	LOAD, Y → Ra; Set CC
LL	a,m	CC a m	63 0 a m	RL	LITERAL LOAD, m → Ra; Set CC
LM	a,y,m	0F a m	03 3 a m	RX	LOAD MULTIPLE, (Y ... Y+m-a) → Ra ... Rm
LPR	a	0C a 4	03 0 a 04	RR	LOAD P REGISTER, (Ra) → P
LR	a,m	04 a m	01 0 a m	RR	LOAD, (Rm) → Ra; Set CC
LX	a,y,m	17 a m	05 3 a m	RX	LOAD AND INDEX BY 1, (Y) → Ra; (Rm)+1 → Rm if a ≠ m; Set CC
LXI	a,m	15 a m	05 1 a m	RI-2	LOAD AND INDEX BY 1, (Y*) → Ra; (Rm)+1 → Rm if a ≠ m; Set CC
S	a,y,m	27 a m	11 3 a m	RX	STORE, (Ra) → Y
SARI	a,m	85 a m	55 1 a m	RI-2	STORE ADDRESS REGISTER, (ARr) → Y*, (Ra): 5-0 Designate ARr
SARM	a,y,m	87 a m	55 3 a m	RX	STORE ADDRESS REGISTER MULTIPLE, (ARr ... ARr+u) → Y ... Y+u; (Ra): 5-0 = Word Designator, (Ra): 13-8 = Count
SARR	a,m	84 a m	55 0 a m	RR	STORE ADDRESS REGISTER, (ARr) → Rm; (Ra): 5-0 Designate ARr
SCR	a	0C a 3	03 0 a 03	RR	STORE REAL TIME CLOCK LOWER, (RTC Register); 15-0 → Ra; Set CC
SCRD	a	0C a D	03 0 a 15	RR	STORE REAL CLOCK DOUBLE, (RTC Register) → Ra, Ra ⊕ 1; Set CC
SD	a,y,m	2B a m	12 3 a m	RX	STORE DOUBLE, (Ra, Ra ⊕ 1) → Y, Y ⊕ 1
SDI	a,m	29 a m	12 1 a m	RI-2	STORE DOUBLE, (Ra, Ra ⊕ 1) → Y*, Y* ⊕ 1
SDX	a,y,m	3B a m	16 3 a m	RX	STORE DOUBLE AND INDEX BY 2, (Ra, Ra ⊕ 1) → Y, Y ⊕ 1; (Rm)+2 → Rm
SDXI	a,m	39 a m	16 1 a m	RI-2	STORE DOUBLE AND INDEX BY 2, (Ra, Ra ⊕ 1) → Y*, Y* ⊕ 1; (Rm)+2 → Rm
SI	a,m	25 a m	11 1 a m	RI-2	STORE, (Ra) → Y*
SM	a,y,m	2F a m	13 3 a m	RX	STORE MULTIPLE, (Ra ... Rm) → Y ... Y+m-a
SMC	a	10 a 4	04 0 a 04	RR	STORE MONITOR CLOCK, (Mon) → Ra
SSOR	a	0C a 1	03 0 a 01	RR	STORE STATUS REGISTER 1, (SR1) → Ra; Set CC
SSTR	a	0C a 2	03 0 a 02	RR	STORE STATUS REGISTER 2, (SR2) → Ra; Set CC
SX	a,y,m	37 a m	15 3 a m	RX	STORE AND INDEX BY 1, (Ra) → Y; (Rm)+1 → Rm
SXI	a,m	35 a m	15 1 a m	RI-2	STORE AND INDEX BY 1, (Ra) → Y*; (Rm)+1 → Rm
SZ	y,m	3F - m	17 3 - m	RX	STORE ZEROS, 0 → Y
SZI	m	3D - m	17 1 - m	RX	STORE ZEROS, 0 → Y*
SUPPORT CHANNEL					
JSC	y,m	82 C m	40 2 14 m	RK	JUMP SUPPORT CHANNEL BUSY, If Support Channel Busy; Y → P
JSC	*y,m	83 C m	40 3 14 m	RX	JUMP SUPPORT CHANNEL BUSY, If Support Channel Busy; (Y) → P
JSCR	m	80 C m	40 0 14 m	RR	JUMP SUPPORT CHANNEL BUSY, If Support Channel Busy (Rm) → P
SCI	a	08 a 7	02 0 a 07	RR	SUPPORT CHANNEL INPUT, Support Channel Input → Ra; Set CC
SCIO	a,m	0D a m	03 1 a m	RR	SUPPORT CHANNEL I/O, (Ra) → Support Channel Buffer, m → I/O Code, Set Channel Busy

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MINEMONIC	INSTRUCTION CODE		FORMAT	INSTRUCTION	
	HEX	o f a m			
STACK AND QUEUE					
QGT	a,y,m	1E a m	07 2 a m	HYB	QUEUE GET TOP, (Y) → Ra; if (Y) = 0 then P+2 → P; If (Y) ≠ 0 then P+3 → P, ((Y)) → Y; If ((Y)) = 0 then Y → Y ⊕ 1
QPB	a,y,m	16 a m	05 2 a m	HYB	QUEUE PUT BOTTOM, (Ra) → (Y ⊕ 1), (Ra) → Y ⊕ 1, 0 → (Ra)
QPT	a,y,m	12 a m	04 2 a m	HYB	QUEUE PUT TOP, (Y) → (Ra), (Ra) → Y; If (Y) = 0 then (Ra) → Y ⊕ 1
SGT	a,y,m	1A a m	06 2 a m	HYB	STACK GET TOP, (Y) → Ra; (Y) ≠ 0 then ((Y)) → Y and P+3 → P, (Y) = 0 then P+2 → P
SPT	a,y,m	02 a m	00 2 a m	HYB	STACK PUT TOP, (Y) → (Ra), (Ra) → Y
MISCELLANEOUS					
BF	y,m	77 - m	35 3 - m	RX	BIASED FETCH, (Y): 15 → CC:9; 1 → (Y):15,14; Set CC
BFI	m	75 - m	35 1 - m	RI-2	BIASED FETCH, (Y*):15 → CC:9; 1 → (Y*):15, 14; Set CC
CNT	a	10 a 2	04 0 a 02	RR	COUNT ONES, Number of 1's in Ra → Ra ⊕ 1
ER	a	0C a 0	03 0 a 00	RR	EXECUTIVE RETURN, Generate Interrupt (P)+1 → Ra
MS	a,y,m	6F a m	33 3 a m	RX	MASKED SUBSTITUTE, If (Ra ⊕ 1):n = 1, (Y): n → Ra: n; Set CC
MSI	a,m	6D a m	33 1 a m	RI-2	MASKED SUBSTITUTE, If (Ra ⊕ 1):n=1, (Y*):n → Ra:n; Set CC
MSK	a,y,m	6E a m	33 2 a m	RK	MASKED SUBSTITUTE, If (Ra ⊕ 1):n=1; Y:n → Ra:n; Set CC
MSR	a,m	6C a m	33 0 a m	RR	MASKED SUBSTITUTE, If (Ra ⊕ 1):n=1; (Rm):n → Ra:n, Set CC
NOP	d=1	81 d	40 1 d	RI-1	NO OPERATION, (Software) (P)+1 → P
NOPD	d=1	81 d	40 1 d	RI-1	NO OPERATION DOUBLE, (P) + 1 → P, (P) + 1 → P
	d=1	81 d	40 1 d	RI-1	
REX	y,m	76 - m	35 2 - m	RK	REMOTE EXECUTE, Execute (Y); (P)+2 → P Unless Jump
RVR	a	10 a 1	04 0 a 01	RR	REVERSE REGISTER, Reverse Order of Bits in Ra, Set CC
SBR	a,m	14 a m	05 0 a m	RR	SET BIT, 1 → Bit Position m of (Ra); Set CC
SFR	a	10 a 3	04 0 a 03	RR	SCALE FACTOR, Shift (Ra, Ra ⊕ 1) Left Until (Ra):15 ≠ (Ra):14; Shift Count → Ra ⊕ 1 + 1; Count = 31 for all Zeros or all Ones
ZBR	a,m	18 a m	06 0 a m	RR	ZERO BIT, 0 → Bit Position m of (Ra); Set CC

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MNEMONIC	INSTRUCTION CODE				FORMAT	INSTRUCTION
	HEX		o	f a m		
COMMAND/CHAIN INSTRUCTIONS						
ACR		EO	0 0	70 0 00 00	RR	CHANNEL CONTROL, Master clear all channels
ACR 4		EO	0 4	70 0 00 04	RR	CHANNEL CONTROL, Enable external interrupts, all channels
CCR 0,4						
ACR 5		EO	0 5	70 0 00 05	RR	CHANNEL CONTROL, Disable external interrupts, all channels
CCR 0,5						
ACR 6		EO	0 6	70 0 00 06	RR	CHANNEL CONTROL, Enable Class III Interrupts, Priorities 2,3,4
CCR 0,6						
ACR 7		EO	0 7	70 0 00 07	RR	CHANNEL CONTROL, Disable Class III Interrupts, Priorities 2,3,4
CCR 0,7						
CCR a,8		EO	a 8	70 0 a 10	RR	CHANNEL CONTROL, Master clear channel a
CCR a,12		EO	a C	70 0 a 14	RR	CHANNEL CONTROL, Enable channel a external interrupts
CCR a,13		EO	a D	70 0 a 15	RR	CHANNEL CONTROL, Disable channel a external interrupts
CCR a,14		EO	a E	70 0 a 16	RR	CHANNEL CONTROL, Enable channel a, Class III Interrupts, priorities 2,3,4
CCR a,15		EO	a F	70 0 a 17	RR	CHANNEL CONTROL, Disable channel a Class III interrupts, priorities 2,3,4
COMMAND INSTRUCTION						
ICK a,y		E6	a 2	71 2 a 02	RK	INITIATE INPUT CHAIN, Y → Channel a Chain Pointer; Initiate Input Chain
OCC a,y		E6	a 6	71 2 a 06	RK	INITIATE OUTPUT CHAIN, Y → Channel a Chain Pointer; Initiate Output Chain
RIM a,y,m		EB	a m	72 3 a m	RX	READ CONTROL MEMORY, Channel a (CMm) → Y
SICR a,m		FB	a m	76 0 a m	RR	SET AND CLEAR DISCRETES, Set or clear channel a discrete function per m designator
SIOP m,y		FC	- m	77 0 - m	RK	START IOP, m:0 → IOP SR1:12, Y → IOP P if m = 0 or 1
SST a,y,m		FB	a m	76 3 a m	RX	STORE STATUS, Channel a Status bits per m → Y
WIM a,y,m		E7	a m	71 3 a m	RX	WRITE CONTROL MEMORY, (Y) → Channel a CMm
XIM a,y,m		FE	a m	77 2 a m	RX	EXCHANGE CONTROL MEMORY, m=2or6, Channel a (CMm) → Y; (Y ⊕ 1) → Channel a CMm
CHAIN INSTRUCTION						
BJ m,y		FD	- m	77 1 - m	RK	BIT JUMP, (Y) → CAP if (CM3): m = 1
CSIR m		F8	0 m	76 0 00 m	RR	SERIAL INTERFACE CONTROL, Set or clear discrete function per m designator
CSST y,m		FB	- m	76 3 - m	RX	STORE STATUS, Status bits per m → Y
HCR		EC	0 0	73 0 00 00	RR	HALT CHAIN, Halt chaining, a even
IM a,y,m		E2	a m	70 2 a m	RK	INITIATE MESSAGE, Y → CMm; Initiate message activity
IO a,y		E3	a 0	70 3 a 00	RX	IO FUNCTION a, (Y, Y ⊕ 1) → BCW, BAP; initiate transfer
IPR		EC	1 0	73 0 01 00	RR	INTERRUPT PROCESSOR, Generate chain interrupt, a odd
LCM m,y		E7	0 m	71 3 00 m	RX	LOAD CONTROL MEMORY, (Y) → CMm
LCMK m,y		E8	0 m	71 2 00 m	RK	LOAD CONTROL MEMORY, Y → CMm
SCM m,y		EB	0 m	72 3 00 m	RX	STORE CONTROL MEMORY, (CMm) → Y
SF y		EF	1 0	73 3 01 00	RX	SET FLAG, 1 → Y:15, 14, a odd
SFSC m		F4	0 m	75 0 00 m	RR	SEARCH FOR SYNC, Perform function(s) assigned to m-bits
SJMC 0,y		F2	0 0	74 2 00 00	RK	SERIAL JUMP ON MET CONDITION, Unconditional Y → CAP; clear flag
SJMC 1,y		F2	1 0	74 2 01 00	RK	SERIAL JUMP ON MET CONDITION, If suppress flag not set, Y → CAP; clear flag
SJMC 2,y		F2	2 0	74 2 02 00	RK	SERIAL JUMP ON MET CONDITION, If monitor flag set, Y → CAP; clear flag
XCM m,y		FE	- m	77 2 - m	RX	EXCHANGE CONTROL MEMORY, (CMm) → Y; (Y ⊕ 1) → CMm
ZF y		EF	0 0	73 3 00 00	RX	ZERO FLAG, 0 → Y:15, 14, a even

TABLE A-2. INSTRUCTION REPERTOIRE BY MNEMONIC (Cont.)

MNEMONIC:	INSTRUCTION CODE			FORMAT	INSTRUCTION
	HEX	o	f s m		
EXECUTIVE MODE					
CLOCKS					
DCIR	OC - F	03 0 -	17	RR	DISABLE RTC OVERFLOW INTERRUPT
DCR	OC - 9	03 0 -	11	RR	DISABLE REAL TIME CLOCK REGISTER
DM	OC - 8	03 0 -	13	RR	DISABLE MONITOR CLOCK REGISTER AND MONITOR CLOCK INTERRUPT
ECIR	OC - E	03 0 -	16	RR	ENABLE RTC OVERFLOW INTERRUPT
ECR	OC - 8	03 0 -	10	RR	ENABLE REAL TIME CLOCK REGISTER AND INTERRUPT
LCR a	OC a 7	03 0 a	07	RR	LOAD REAL TIME CLOCK LOWER, (Ra) → RTC Register Lower: 15 - 0
LCRD a	OC a C	03 0 a	14	RR	LOAD REAL TIME CLOCK DOUBLE AND ENABLE COUNT, (Ra,Ra ⊕ 1) → RTC Register, Enable Count Up
LEM a	OC a A	03 0 a	12	RR	LOAD AND ENABLE MONITOR CLOCK, (Ra) → Monitor Clock Register; Enable Count Down
RBT	08 - E	02 0 -	16	RR	RESET BIT TIMER, 0 → Bit Timer
JUMPS					
DJ a	08 a D	02 0 a	15	RR	DIAGNOSTIC JUMP, (R13) → μp
JKS 1 y,m	82 A m	40 2 12	m	RK	JUMP AFTER STOP KEY 1 SET, If Key 1 Set, Stop; Y → P
JKS 1 *y,m	83 A m	40 3 12	m	RX	JUMP AFTER STOP KEY 1 SET, If Key 1 Set, Stop; (Y) → P
JKS 2 y,m	82 B m	40 2 13	m	RK	JUMP AFTER STOP KEY 2 SET, If Key 2 Set, Stop; Y → P
JKS 2 *y,m	83 B m	40 3 13	m	RX	JUMP AFTER STOP KEY 2 SET, If Key 2 Set, Stop; (Y) → P
JKSR 1 m	80 A m	40 0 12	m	RR	JUMP AFTER STOP KEY 1 SET, If Key 1 Set, Stop; (Rm) → P
JKSR 2 m	80 B m	40 0 13	m	RR	JUMP AFTER STOP KEY 2 SET, If Key 2 Set, Stop; (Rm) → P
JS y,m	82 9 m	40 2 11	m	RK	JUMP AFTER STOP, Stop; Upon Restart, Y → P
JS *y,m	83 9 m	40 3 11	m	RX	JUMP AFTER STOP, Stop; Upon Restart, (Y) → P
JSR m	80 9 m	40 0 11	m	RR	JUMP AFTER STOP, Stop; Upon Restart, (Rm) → P
LOADS/STORES					
LARI a,m	B1 a m	54 1 a	m	RI-2	LOAD ADDRESS REGISTER, (Y*) → ARr; (Ra):5-0 Designate ARr
LARM a,y,m	83 a m	54 3 a	m	RX	LOAD ADDRESS REGISTER MULTIPLE, (Y ... Y+u) → ARr ... ARr+u; (Ra):5-0 = Word Designator, (Ra):13-8 = Count
LARR a,m	80 a m	54 0 a	m	RR	LOAD ADDRESS REGISTER, (Rm) → ARr; (Ra):5-0 Designates ARr
LP y,m	1F - m	07 3 -	m	RX	LOAD PSW, (Y,Y+1,Y+2) → P-Register, Status Register 1 and Status Register 2, Respectively
LPI m	ID - m	07 1 -	m	RI-2	LOAD PSW, (Y*,Y*+1,Y*+2) → P-Register, Status Register 1, and Status Register 2, Respectively
LSOR a	OC a 5	03 0 a	05	RR	LOAD STATUS REGISTER 1, (Ra) → SR1
LSTR a	OC a 6	03 0 a	06	RR	LOAD STATUS REGISTER 2, (Ra) → SR2
MISCELLANEOUS					
IOCR	74 - -	35 0 -	-	RR	INPUT/OUTPUT COMMAND, Execute (C Cell); 0 → C Cell:15,14
IPI a	08 a 3	02 0 a	03	RR	INITIATE PROCESSOR INTERRUPT, Set Processor Interrupt a
IPLF	08 - C	02 0 -	14	RR	INITIAL PROGRAM LOAD FAILED, Set IPLF Discrete
SBT	08 - F	02 0 -	17	RR	SET BIT INDICATOR

APPENDIX B

GENERAL REFERENCE TABLES

TABLE B-2. HEXADECIMAL CONVERSION TABLES

The following tables aid in converting hexadecimal values to decimal values, or the reverse.

Direct Conversion Table

This table provides direct conversion of decimal and hexadecimal numbers in these ranges:

HEXADECIMAL	DECIMAL
000 to FFF	0000 to 4095

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

HEXADECIMAL	DECIMAL
1000	4096
2000	8192
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	53248
E000	57344
F000	61440

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00_	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01_	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02_	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03_	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04_	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05_	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06_	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07_	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08_	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09_	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A_	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B_	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C_	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D_	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E_	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F_	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10_	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11_	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12_	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13_	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14_	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15_	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16_	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17_	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18_	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19_	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A_	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B_	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C_	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D_	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E_	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F_	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

TABLE B-2. HEXADECIMAL CONVERSION TABLES (Cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20_	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21_	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22_	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23_	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24_	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25_	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26_	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27_	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28_	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29_	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A_	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B_	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C_	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D_	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E_	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F_	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30_	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31_	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32_	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33_	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34_	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35_	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36_	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37_	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38_	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39_	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A_	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B_	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C_	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D_	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E_	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F_	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40_	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41_	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42_	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43_	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44_	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45_	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46_	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47_	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48_	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49_	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A_	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B_	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C_	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D_	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E_	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F_	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50_	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51_	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52_	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53_	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54_	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55_	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56_	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57_	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58_	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59_	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A_	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B_	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C_	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D_	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E_	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F_	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

TABLE B-2. HEXADECIMAL CONVERSION TABLES (Cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60_	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61_	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62_	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63_	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64_	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65_	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66_	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67_	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68_	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69_	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A_	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B_	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C_	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D_	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E_	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F_	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70_	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71_	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72_	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73_	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74_	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75_	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76_	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77_	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78_	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79_	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A_	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B_	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C_	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D_	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E_	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F_	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80_	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81_	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82_	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83_	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84_	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85_	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86_	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87_	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88_	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89_	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A_	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B_	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C_	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D_	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E_	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F_	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90_	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91_	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92_	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93_	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94_	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95_	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96_	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97_	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98_	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99_	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A_	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B_	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C_	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D_	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E_	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F_	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

TABLE B-2. HEXADECIMAL CONVERSION TABLES (Cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0_	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1_	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2_	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3_	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4_	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5_	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6_	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7_	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8_	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9_	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA_	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB_	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC_	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD_	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE_	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF_	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0_	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1_	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2_	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3_	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4_	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5_	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6_	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7_	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8_	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9_	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA_	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB_	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC_	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD_	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE_	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF_	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0_	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1_	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2_	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3_	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4_	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5_	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6_	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7_	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8_	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9_	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA_	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB_	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC_	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD_	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE_	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF_	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0_	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1_	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2_	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3_	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4_	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5_	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6_	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7_	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8_	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9_	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA_	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB_	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC_	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD_	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE_	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF_	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

TABLE B-2. HEXADECIMAL CONVERSION TABLES (Cont.)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F0_	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1_	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2_	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3_	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4_	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5_	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6_	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7_	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
F8_	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9_	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA_	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB_	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC_	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED_	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE_	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF_	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0_	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1_	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2_	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3_	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4_	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5_	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6_	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7_	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8_	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9_	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA_	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB_	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC_	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD_	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE_	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF_	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

TABLE B-2. HEXADECIMAL CONVERSION TABLES (Cont.)

Hexadecimal and Decimal Fraction Conversion Table

BYTE				HALFWORD									
0123		4567		0123				4567					
Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal Equivalent	Hex	Decimal Equivalent	Hex	Decimal Equivalent
.0	.0000	.00	.0000	0000	.000	.0000	0000	0000	.0000	.0000	0000	0000	0000
.1	.0625	.01	.0039	0625	.001	.0002	4414	0625	.0001	.0000	1525	8789	0625
.2	.1250	.02	.0078	1250	.002	.0004	8828	1250	.0002	.0000	3051	7578	1250
.3	.1875	.03	.0117	1875	.003	.0007	3242	1875	.0003	.0000	4577	6367	1875
.4	.2500	.04	.0156	2500	.004	.0009	7656	2500	.0004	.0000	6103	5156	2500
.5	.3125	.05	.0195	3125	.005	.0012	2070	3125	.0005	.0000	7629	3945	3125
.6	.3750	.06	.0234	3750	.006	.0014	6484	3750	.0006	.0000	9155	2734	3750
.7	.4375	.07	.0273	4375	.007	.0017	0898	4375	.0007	.0001	0681	1523	4375
.8	.5000	.08	.0312	5000	.008	.0019	5312	5000	.0008	.0001	2207	0312	5000
.9	.5625	.09	.0351	5625	.009	.0021	9726	5625	.0009	.0001	3732	9101	5625
.A	.6250	.0A	.0390	6250	.00A	.0024	4140	6250	.000A	.0001	5258	7890	6250
.B	.6875	.0B	.0429	6875	.00B	.0026	8554	6875	.000B	.0001	6784	6679	6875
.C	.7500	.0C	.0468	7500	.00C	.0029	2968	7500	.000C	.0001	8310	5468	7500
.D	.8125	.0D	.0507	8125	.00D	.0031	7382	8125	.000D	.0001	9836	4257	8125
.E	.8750	.0E	.0546	8750	.00E	.0034	1796	8750	.000E	.0002	1362	3046	8750
.F	.9375	.0F	.0585	9375	.00F	.0036	6210	9375	.000F	.0002	2888	1835	9375
1		2		3				4					

TO CONVERT .ABC HEXADECIMAL TO DECIMAL

- Find .A in position 1 .6250
- Find .0B in position 2 .0429 6875
- Find .00C in position 3 .0029 2968 7500
- .ABC Hex is equal to .6708 9843 7500

To convert fractions beyond the capacity of table, use techniques below:

HEXADECIMAL FRACTION TO DECIMAL

Convert the hexadecimal fraction to its decimal equivalent using the same technique as for integer numbers. Divide the results by 16ⁿ (n is the number of fraction positions).

Example: .8A7 = .540771₁₀

$$8A7_{16} = 2215_{10}$$

$$16^3 = 4096 \quad \frac{2215}{4096} = .540771$$

TO CONVERT .13 DECIMAL TO HEXADECIMAL

- Find .1250 next lowest to subtract

$$\begin{array}{r} .1300 \\ -.1250 \\ \hline \end{array} = .2_{Hex}$$
- Find .0039 0625 next lowest to

$$\begin{array}{r} .0050 \ 0000 \\ -.0039 \ 0625 \\ \hline \end{array} = .01$$
- Find .0009 7656 2500

$$\begin{array}{r} .0010 \ 9375 \ 0000 \\ -.0009 \ 7656 \ 2500 \\ \hline \end{array} = .004$$
- Find .0001 0681 1523 4375

$$\begin{array}{r} .0001 \ 1718 \ 7500 \ 0000 \\ -.0001 \ 0681 \ 1523 \ 4375 \\ \hline \end{array} = .0007$$
- .13 Decimal is approximately equal to $\frac{.0000 \ 1037 \ 5776 \ 5625}{.0000 \ 1037 \ 5776 \ 5625} = .2147_{Hex}$

DECIMAL FRACTION TO HEXADECIMAL

Collect integer parts of product in the order of calculation.

Example: .5408₁₀ = .8A7₁₆

$$\begin{array}{r} .5408 \\ \times 16 \\ \hline 8 \leftarrow [8] .6528 \\ \times 16 \\ \hline A \leftarrow [10] .4448 \\ \times 16 \\ \hline 7 \leftarrow [7] .1168 \end{array}$$

TABLE B-3. ASCII CODE MATRIX

<div style="display: flex; align-items: center;"> <div style="writing-mode: vertical-rl; transform: rotate(180deg); margin-right: 5px;">Bits</div> <div style="display: flex; flex-direction: column; align-items: center;"> <div style="margin-bottom: 5px;">→ b₇</div> <div style="margin-bottom: 5px;">→ b₆</div> <div style="margin-bottom: 5px;">→ b₅</div> </div> </div>					0	0	0	0	1	1	1	1
					b ₄	b ₃	b ₂	b ₁	0	1	2	3
					0	1	2	3	4	5	6	7
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	8	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(8	H	X	h	x
1	0	0	1	9	HT	EM)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	/
1	1	0	1	13	CR	GS	-	=	M]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

APPENDIX C

PSEUDO-OPS, COMMANDS, AND REQUESTS

TABLE C-1. MACRO 20/14 PSEUDO-OPS

Assembly Start	- *ULTRA
Assembly End	- END
Source Library Input	- COPY
Symbol Definitions Statements	
External Modifier	- ENTRY, LINK
Address Counter Name Declaration	- AC
Literal Label Declaration	- LIT
Equate	- EQU
Assign Symbol	- SET
Define Format Symbol	- FORM
Address Counter Control Statements	
Modify Address Counter	- RES, BSS
Set Address Counter	- ORIG
Align Address Counter	- EVEN, ODD
MACRO Definition Statements	
MACRO Entry	- MACRO, NAME
Unconditional Branch	- GO, EXIT
MACRO End	- END
Conditional Assembly Statements	
Conditional/Multiple Statement Assembly	- DO
Conditional Block Assembly	- IFx/ELSEC/END
Output Control Statements	
Output Disposition	- OPTIONS
Source Statement Output	- PUNCH
Listing Control Statements	
Generate Message	- NOTE
Identify Assembly Output	- TITLE
Start New Page	- EJECT
Space Listing	- SPACE
Print Assembly Source	- LIST, NLST
Print Unassembled Sources	- LISTC, NLSTC
Print MACRO Expansion	- LISTM
Print Cross Reference	- LISTX

TABLE C-2. AN/AYM-18 FIELD PACKAGE COMMANDS

AL	AUTOLOAD
TM	TAPE TO MEMORY
GO	START EXECUTION
LVL	L/V LINK TEST
LVC	L/V CATALOG/CHECKSUM TAPE
ST	AN/AYK-14(V) SELF-TEST
EN	SUPPORT CHANNEL (SRA)
TX	TRANSMITTER
RX	RECEIVER
PTR	REWIND CASSETTE
MT	MEMORY TO TAPE
CS	CLEAR SCRATCH AREA OF CASSETTE
CN	COMPUTER NUMBER
LVW	L/V WRITE TEST
ES1	ENABLE STOP KEY 1
ES2	ENABLE STOP KEY 2
DS1	DISABLE STOP KEY 1
DS2	DISABLE STOP KEY 2
B1	DISABLE BOOT 2
B2	ENABLE BOOT 2

TABLE C-3. AN/AYM-18 LAB PACKAGE COMMANDS

RD	GENERAL REGISTER DISPLAY
S1	STATUS REGISTER 1 AND CONTENTS
S2	STATUS REGISTER 2 AND CONTENTS
P	P REGISTER AND CONTENTS
PG	PAGE REGISTER AND CONTENTS
CH	CHANNEL DISPLAY
LLJ	LOCATION LAST JUMP
MD	MEMORY DISPLAY
BP	ENTERS AND ENABLES/DISABLES BREAKPOINT P
BO	ENTERS AND ENABLES/DISABLES BREAKPOINT OPERAND
S	STOP/STEP
MC	MASTER CLEAR
LVR	DISPLAY L/V CODE
LVM	MODIFY L/V CODE
LVT	COPY L/V PROGRAM TO TAPE
X	CHANGE
ENTER KEY	+
BACKSPACE KEY	-
PTA	ADVANCE TAPE ONE FILE
SD	SOFTWARE/ERROR DISPLAY
FD	FILE DISPLAY

TABLE C-4. CCU REQUESTS

Computer Number Definition

ST-n.	START
UL-n.	MICROMEMORY LOAD
U7-n.	MICROMEMORY LOAD, 7-TRACK
UV-n.	MICROMEMORY CHECKSUM VERIFY

Display Mode Requests

SM-.	SOFTWARE MODE
HO-.	HEX/OCTAL
RD-.	REGISTER DISPLAY
MD-a.	MEMORY DISPLAY
RM-.	ROLL MEMORY
PD-a.	PAGE DISPLAY
CH-a.	CHANNEL CONTROL MEMORY DISPLAY
FD-a.	FILE DISPLAY
BD-a.	BINARY DISPLAY
CD-a.	C FILE DISPLAY

AN/AYK-14 Tape and Printer Requests

AL-.	AN/AYK-14 MEMORY LOAD
A7-.	AN/AYK-14 MEMORY LOAD, 7-TRACK
VL-.	VERIFY AN/AYK-14 MEMORY
MT-a, b.	MEMORY TO TAPE
PM-a, b.	PRINT MEMORY

AN/AYK-14 Operation Requests

MC-.	MASTER CLEAR
Control R	RUN
Control S	STOP/STEP
AS-.	AUTO STEP ENABLE/DISABLE
RJ-.	RUN UNTIL JUMP
P -a.	ENTER INTO P
AD-a.	ENTER INTO AD
+	AD+1
-	AD-1
S1-d.	ENTER INTO S1
S2-d.	ENTER INTO S2
CR-r, d.	CHANGE REGISTER
CP-a, d.	CHANGE PAGE
CB-n	CHANGE BLOCK
CM-a, d ₀ , . . . , d _n .	CHANGE MEMORY

TABLE C-4. CCU REQUESTS (Cont.)

FM-a, b, d.	FILL MEMORY
EP-a.	EVEN PARITY
OP-a	ODD PARITY
BP-a	ENTER INTO BP
BO-a.	ENTER INTO BO
EP-P.	ENABLE BREAKPOINT ON P
DB-P.	DISABLE BREAKPOINT ON P
EB-O.	ENABLE BREAKPOINT ON OPERAND ADDRESS
DB-O.	DISABLE BREAKPOINT ON OPERAND ADDRESS
ES-1	ENABLE STOP KEY 1
DS-1	DISABLE STOP KEY 1
ES-2	ENABLE STOP KEY 2
DS-2	DISABLE STOP KEY 2
B1-.	BOOT 1 SELECT
B2-.	BOOT 2 SELECT
DI-R.	DISABLE REAL-TIME CLOCK INTERRUPT
EX-i.	EXECUTE INSTRUCTION
PE-.	SEARCH FOR PARITY ERROR

Support Channel Requests

EN-d.	ENTER SUPPORT CHANNEL DATA
Control E	CLEAR SUPPORT CHANNEL BUSY

Tape Requests

PT-L.	POSITION TAPE TO LOAD POINT
P7-L.	POSITION 7-TRACK TAPE TO LOAD POINT
PT-F.	ADVANCE FILE MARK
P7-F.	ADVANCE FILE MARK, 7-TRACK
PT-A.	ADVANCE RECORD
P7-A.	ADVANCE RECORD, 7-TRACK
PT-B.	BACKSPACE RECORD
P7-B.	BACKSPACE RECORD, 7-TRACK
PT-RD.	DISPLAY TAPE RECORD
P7-RD.	DISPLAY TAPE RECORD, 7-TRACK
PT-RP.	PRINT TAPE RECORD
P7-RP.	PRINT TAPE RECORD, 7-TRACK
PT-AF.	ADVANCE FILE
PT-BF.	BACKSPACE FILE

Line Printer Requests

JM-.	JOURNAL MODE
PS-.	PRINT SCREEN

TABLE C-4. CCU REQUESTS (Cont.)

Loader/Verifier Requests (AN/AYM-18)

TX	READ 9-TRACK, XMIT ON CSC LINK (2EOF)
TXF	READ 9-TRACK, XMIT ON CSC LINK (CORE FILL)
RX	READ CSC LINK, WRITE 9-TRACK
LVC	READ, CHECKSUM, CATALOG 9-TRACK TAPE, PROVIDE CRT OR PRINTER OUTPUT

8080 Requests

D8-a.	DISPLAY 8080 MEMORY
P8-a, b.	PRINT 8080 MEMORY
F8-a, d.	CHANGE 8080 MEMORY
S8-a, b.	8080 MEMORY DUMP
T8-.	CCU BIT
T8-C.	CCU CRT/KEYBOARD TEST

Firmware Requests

Control U	MICROMODE
UC-a, d ₀ ...d _n .	MICROMEMORY CHANGE
Control G	GO
GO-a.	GO FROM ADDRESS
GM-.	GO MODE
Control H	HALT
Control A	STEP
UB-a.	ENTER MICROBREAKPOINT
Control B	MICROBREAKPOINT ENABLE/DISABLE
Control I	INCREMENT MICROBREAKPOINT
Control D	DECREMENT MICROBREAKPOINT
Control F	FORCE MICROBREAKPOINT ENABLE/DISABLE
CF-a, d.	CHANGE FILE
CC-a, d.	CHANGE C FILE
ID-a.	MICROMEMORY DISPLAY
UP-a, b.	MICROMEMORY PRINT
UT-a, b.	MICROMEMORY TO TAPE
L8-d.	MESSAGE TO AN/AYK-14

COMMENT SHEET

MANUAL TITLE AN/AYK-14(V) Instruction Set Programmer's Reference Manual

PUBLICATION NO. 14122000 REVISION _____

FROM: NAME: _____
BUSINESS ADDRESS: _____

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

PRINTED IN U.S.A.

A3419 REV. 11/69

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

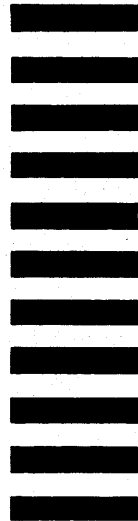
FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241
MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION



CUT ALONG LINE

FOLD

FOLD