60472800

CONTROL DATA CORPORATION

CDC® NETWORK PROCESSOR UNIT

2551-1
2551-2
2551-3
2551-4
2552-2

HARDWARE REFERENCE MANUAL

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary edition
9/77	
A	Initial release
(04-28-78)	
B	Revised to incorporate Engineering Change Orders 8050, 8062, 8130, 8039, 8240,
(12-15-78)	8216, and 8110. Editorial change only on page 1-5. Page 4-1 deleted.
C	Revised to incorporate latest configuration changes and requests on comment sheet.
(04-15-80)	This edition obsoletes the previous revisions.
D	Manual revised; includes Engineering Change Order 44344. Front Cover through vii,
(03-04-83)	ix, x, 1-1 through 1-17, 2-2, and 2-12 are revised. Pages 1-18 through 1-20 are
	added.
Publication No. 60472800	

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

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	PAGE	REV
Front Cover	-	3-12	C	4-35	C				
Title Page	-	3-13	C	4-36	C				
ii	D	3-14	C	4-37	C				
iii/iv	D	3-15	C	4-38	C				
v	D	3-16	C	4-39	C				
vi	D	3-17	C	4-40	C				
vii	D	3-18	C	4-41	C				
viii	C	3-19	C	4-42	C				
ix	D	3-20	C	4-43	C				
x	D	3-21	C	4-44	C				
1-1	D	3-22	C	4-45	C				
1-2	D	3-23	C	4-46	C				
1-3	D	3-24	C	4-47	C				
1-4	D	3-25	C	4-48	C				
1-5	D	3-26	C	4-49	C				
1-6	D	3-27	C	4-50	C				
1-7	D	3-28	C	4-51	C				
1-8	D	3-29	C	4-52	C				
1-9	D	3-30	C	4-53	C				
1-10	D	3-31	C	4-54	C				
1-11	D	3-32	C	4-55	C				
1-12	D	3-33	C	4-56	C				
1-13	D	3-34	C	4-57	C				
1-14	D	3-35	C	4-58	C				
1-15	D	3-36	C	4-59	C				
1-16	D	4-1	C	4-60	C				
1-17	D	4-2	C	4-61	C				
1-18	D	4-3	C	4-62	C				
1-19	D	4-4	C	4-63	C				
1-20	D	4-5	C	4-64	C				
2-1	C	4-6	C	4-65	C				
2-2	D	4-7	C	4-66	C				
2-3	C	4-8	C	5-1	C				
2-4	C	4-9	C	5-2	C				
2-5	C	4-10	C	5-3	C				
2-6	C	4-11	C	5-4	C				
2-7	C	4-12	C	5-5	C				
2-8	C	4-13	C	5-6	C				
2-9	C	4-14	C	5-7	C				
2-10	C	4-15	C	5-8	C				
2-11	C	4-16	C	A-1	C				
2-12	D	4-17	C	B-1	C				
2-13	C	4-18	C	B-2	C				
2-14	C	4-19	C	Comment					
2-15	C	4-20	C	Sheet	D				
2-16	C	4-21	C	Back Cover	-				
2-17	C	4-22	C						
2-18	C	4-23	C						
3-1	C	4-24	C						
3-2	C	4-25	C						
3-3	C	4-26	C						
3-4	C	4-27	C						
3-5	C	4-28	C						
3-6	C	4-29	C						
3-7	C	4-30	C						
3-8	C	4-31	C						
3-9	C	4-32	C						
3-10	C	4-33	C						
3-11	C	4-34	C						

PREFACE

This manual describes the physical, functional, operational, and programming characteristics of the CDC® 2551-1, 2551-2, 2551-3, 2551-4, and 2552-2 Network Processor Units (NPUs). The manual includes the various NPU configurations, which are available to the user by addition of standard product options and upgrade kits.

This manual is intended for use by customer, marketing, training, and programming personnel. For an in-depth analysis of the product structure, operation, and maintenance, refer to the NPU hardware maintenance manual.

The related publications listed below are available through CDC Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

Publication Title	Publication Number
Basic Microprogrammable Processor, Hardware Maintenance Manual	39451400
1714 Computer System, Reference Manual	60364900
3000L Communications Coupler DY161-A, Hardware Reference/Maintenance Manual	60470400
2552-1 6000 Coupler, Hardware Maintenance Manual	60471900
Cyclic Encoder DY180-A, Hardware Maintenance Manual	60472600
2550-101 Emulation Controlware, Reference Manual	60474000
2550-101 Installation Handbook	60474100
2551-1, 2551-2, 2551-3, 2551-4, 2552-2 Network Processor Unit Host Communications Processor 2550-2 Site Preparation Manual	74641200
Synchronous Communications Line Adapter DU138-A, DU139-A, DU140-A, Hardware Maintenance Manual	74700700
Asynchronous Communications Line Adapter DU137-A, Hardware Maintenance Manual	74700900
Communications Line Adapter, Synchronous Data Link Control DY162-A, Hardware Maintenance Manual	74873290

Communications Coupler DK106-B, Hardware Maintenance Manual	74873946
Network Processor Unit Equipment Cabinets, Hardware Maintenance Manual	74873971
System Autostart Module DY204-A, Hardware Maintenance Manual	74874450
Emulation Coupler DY159-A, XA226-A, Hardware Maintenance Manual	74879600
Tape Cassette Controller FA104-A, Hardware Maintenance Manual	96711900
255X Host Communications Processor/ Network Processor Unit, Reference Manual, CCP Version 1.0	60470000
Transform with Micromemory DE402-A/ 1700, Hardware Maintenance Manual	96728700
I/O TTY Controller CW212-A, Hardware Maintenance Manual	96728900
MP17 Communications Processor, Hardware Reference Manual	96768300
1700 Enhanced Processor with MOS Memory and Interface AA109-A/B, Hardware Maintenance Manual	96768600
AA109-A/B (AA135-A) Communications Controller Field Print Package	60475150

WARNING

This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A peripheral computing device pursuant to Subpart J of Part 15 of the FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

CONTENTS

APPENDIXES

FIGURES

TABLES

SYSTEM DESCRIPTION 1

This section contains a description of the 2551-1, 2551-2, 2551-3, 2551-4, and 2552-2 NPUs and the major features and components of the systems. The NPU is a medium-size digital data communications system that serves as a front-end to a host computer and interfaces with a variety of terminal devices. The NPU can serve a CDC® 6000, CYBER 170, CYBER 70, or a lower 3000 series host computer. Figures 1-1 and 1-2 are block diagrams of typical 2551-1, 2551-2, 2551-3, 2551-4, and 2552-2 NPU configurations.

NOTE

The 2552-2 NPU is no longer available as a CDC product. However, information is retained herein, for reference only.

MAJOR FEATURES

The 2551-1 NPU configuration contains the following features:

- Single-bay cabinet
- Single processor
- 32K words MOS memory (expandable to 128 words)
- Interfaces up to 32 communications lines
- Expandable to 2551-2 configuration that provides interface with 32 additional communications lines
- Forms a front-end to a single CDC host computer when connected by appropriate coupler

The 2551-2 configuration contains the following features:

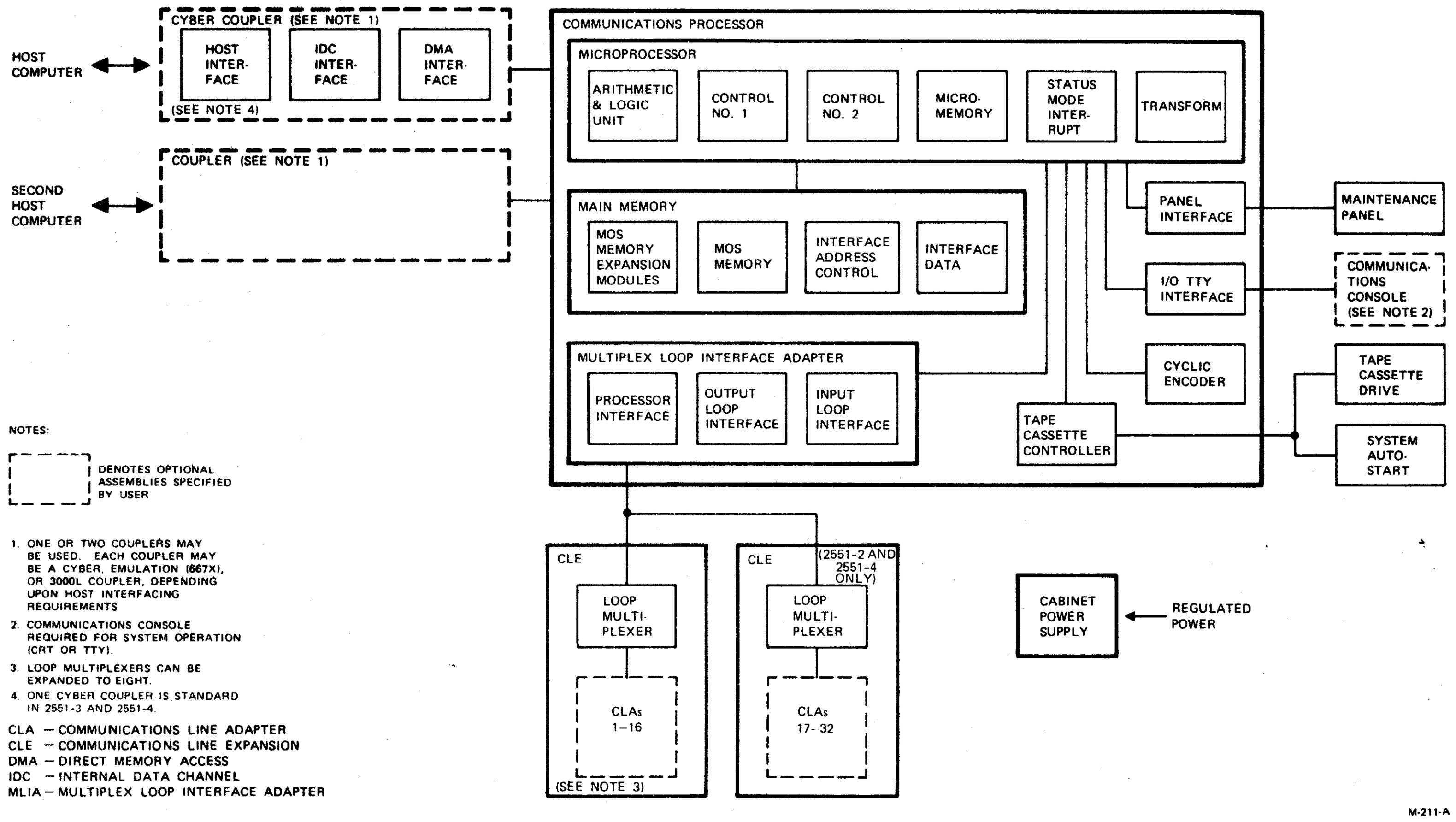
- Single-bay cabinet
- Single processor
- 32K words MOS memory (expandable to 128 words)
- Interfaces up to 64 communications lines
- Expandable to 128 lines with addition of a loop multiplexer expansion cabinet that contains two loop multiplexer expansion chassis and communications line adapters
- Forms a front-end to a single CDC host computer when connected by appropriate coupler.

The 2551-3 NPU configuration contains the following features:

- Single-bay cabinet
- Single processor
- CYBER Coupler
- 64K words MOS memory (expandable to 128 words)
- Interfaces up to 32 communications lines
- Expandable to 2551-4 configuration that provides interface with 32 additional communications lines
- Forms a front-end to a single CDC host computer

The 2551-4 configuration contains the following features:

- Single-bay cabinet
- Single processor
- CYBER Coupler
- 96K words MOS memory (expandable to 128 words)
- Interfaces up to 64 communications lines
- Expandable to 128 lines with addition of a loop multiplexer expansion cabinet that contains two loop multiplexer expansion chassis and communications line adapters
- Forms a front-end to a single CDC host computer



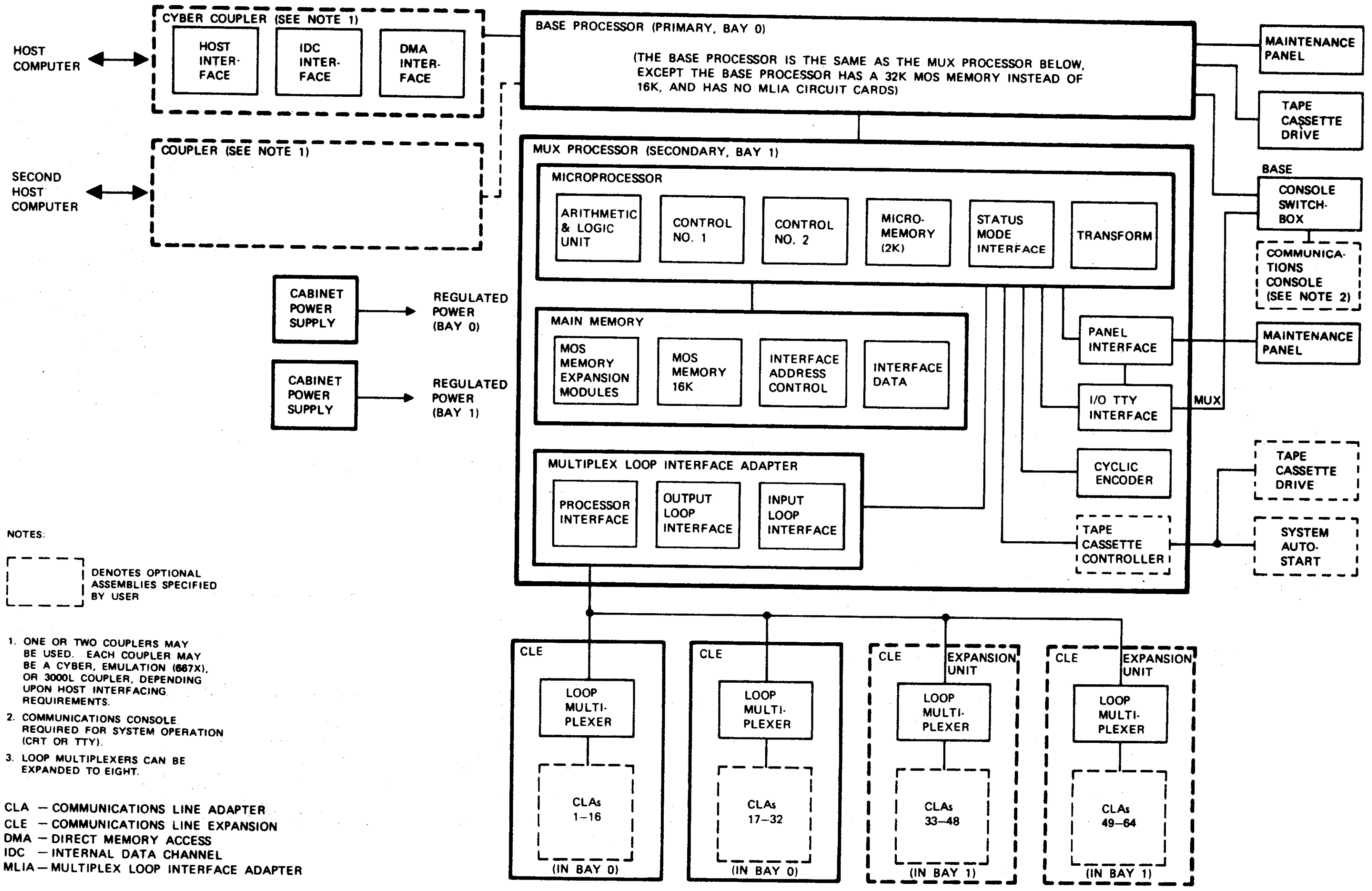
NOTES:

DENOTES OPTIONAL ASSEMBLIES SPECIFIED BY USER

1. ONE OR TWO COUPLERS MAY BE USED. EACH COUPLER MAY BE A CYBER, EMULATION (667X), OR 3000L COUPLER, DEPENDING UPON HOST INTERFACING REQUIREMENTS
2. COMMUNICATIONS CONSOLE REQUIRED FOR SYSTEM OPERATION (CRT OR TTY).
3. LOOP MULTIPLEXERS CAN BE EXPANDED TO EIGHT.
4. ONE CYBER COUPLER IS STANDARD IN 2551-3 AND 2551-4.

CLA	— COMMUNICATIONS LINE ADAPTER
CLE	— COMMUNICATIONS LINE EXPANSION
DMA	— DIRECT MEMORY ACCESS
IDC	— INTERNAL DATA CHANNEL
MLIA	— MULTIPLEX LOOP INTERFACE ADAPTER

Figure 1-1. 2551-1, 2551-2, 2551-3, 2551-4 Network Processor Unit Block Diagram



NOTES:

DENOTES OPTIONAL ASSEMBLIES SPECIFIED BY USER

1. ONE OR TWO COUPLERS MAY BE USED. EACH COUPLER MAY BE A CYBER, EMULATION (667X), OR 3000L COUPLER, DEPENDING UPON HOST INTERFACING REQUIREMENTS.
2. COMMUNICATIONS CONSOLE REQUIRED FOR SYSTEM OPERATION (CRT OR TTY).
3. LOOP MULTIPLEXERS CAN BE EXPANDED TO EIGHT.

CLA	— COMMUNICATIONS LINE ADAPTER
CLE	— COMMUNICATIONS LINE EXPANSION
DMA	— DIRECT MEMORY ACCESS
IDC	— INTERNAL DATA CHANNEL
MLIA	— MULTIPLEX LOOP INTERFACE ADAPTER

Figure 1-2. 2552-2 Network Processor Unit Block Diagram

The 2552-2 NPU configuration contains the following features:

- Double-bay cabinet
- Two processors
  - Base processor contains 32K word MOS memory (expandable to 128K words)
  - Multiplex processor contains 16K words MOS memory (expandable but usually not required)
- Interfaces up to 64 communications lines
- Expandable up to 128 lines with addition of two loop multiplexer expansion chassis and communications line adapters
- Forms front-end to a single CDC host computer with one host coupler, or two hosts with addition of a second host coupler

NETWORK COMMUNICATIONS SYSTEM CONCEPTS

The NPU operates as an integral part of a CDC network communications system (NCS). This system provides for the vast majority of communications (message handling) tasks to be handled at intermediate points between the message source and destination. This is accomplished by NPUs, also called nodes, which are designed to allow almost any combination of functions to be attended by hardware common to all nodes.

The NPU acts as an intermediate communications handler or communications node between a host computer and the user terminals. Figure 1-3 illustrates a single node system. The NPU is limited to two host computers. The host computer can have two or more NPU front-ends and can be extended by common trunk lines linked to remote nodes, which act as message concentrators from multiterminal interfaces to form a multinode system. See figure 1-4.

The network communications system concept allows open-ended linkage of the host computer and front-end nodes and a variety of geographically distributed remote NPUs. In the multinode system one host is designated to maintain circuits that control the entire network communications process.

PHYSICAL DESCRIPTION

The following is a brief physical description of the major equipments in each NPU configuration. For a detailed physical description, refer to the appropriate equipment maintenance manual. See preface.

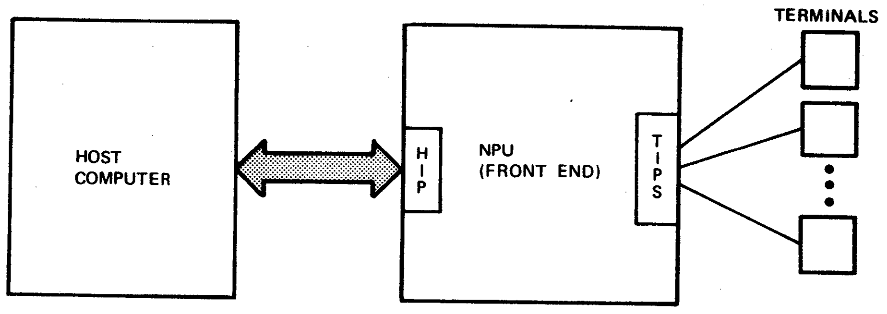


Figure 1-3. Single Node System, Block Diagram

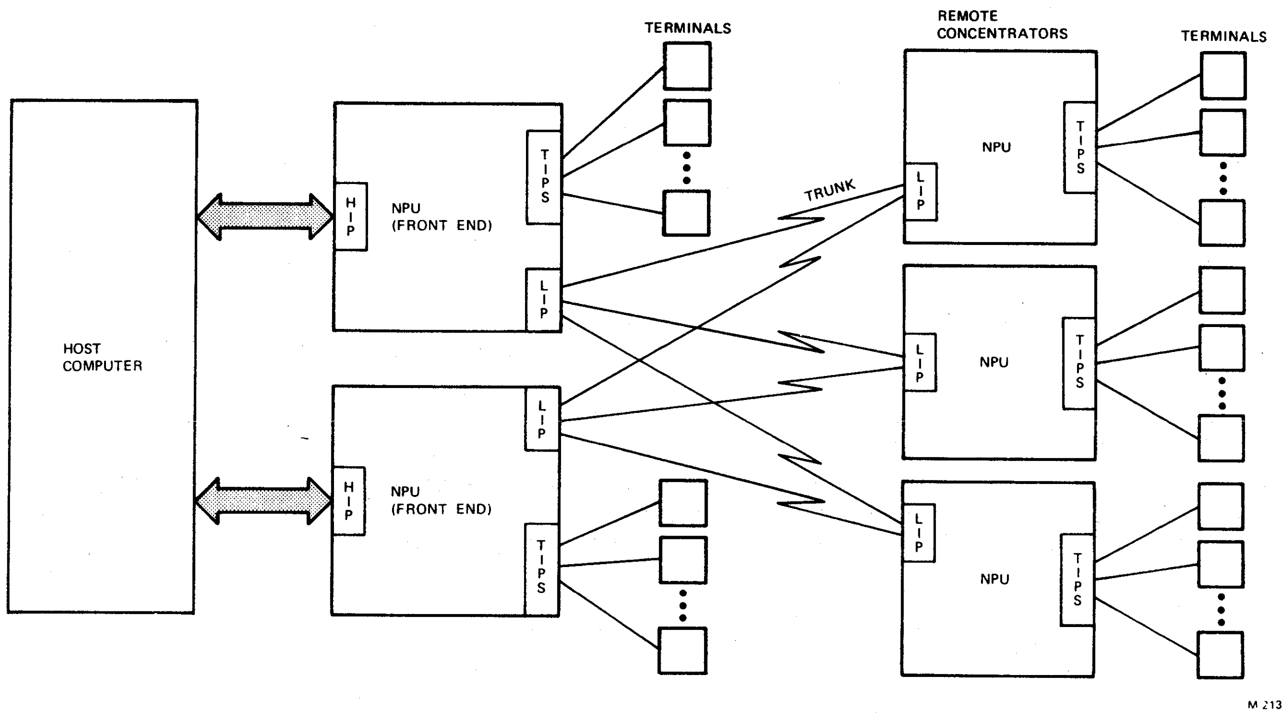


Figure 1-4. Multinode System (One Local and Three Remote Nodes), Block Diagram

2551-1 NPU EQUIPMENT

The 2551-1 NPU (figure 1-5) consists of NPU cabinet assembly DW117-A, processor AA135-A, and 32K MOS Memory AT241-B. The DW117-A contains the basic cabinet, loop multiplexer and CLA card cage, multiplex loop interface adapter (MLIA), cyclic encoder, power supply, and associated signal and power cabling. Also, a maintenance panel, power distribution box, cable enclosure assembly, and two air blower and filter assemblies are included with the cabinet. The processor is contained in the communications processor card cage, which also provides the card slots for the 32K MOS memory, cyclic encoder, and MLIA cards. One loop multiplexer card and from 1 to 16 CLA cards can be located in the loop multiplexer and CLA card cage.

2551-2 NPU EQUIPMENT

The 2551-2 NPU (figure 1-5) contains the same components as the 2551-1 and includes an additional loop multiplexer and CLA card cage. The NPU can service up to 64 CLAs in the system; thus, the 2551-2 NPU has double the line capacity of the 2551-1. The B version cabinet contains the same electromagnetic interference shielding as the 2551-1.

2551-3 NPU EQUIPMENT

The 2551-3 NPU (figure 1-5) consists of NPU cabinet assembly DW117-A, processor AA135-B, 64K MOS memory AT241-B, and CYBER coupler DK106-C. The DW117-A contains the basic cabinet, loop multiplexer and CLA card cage, multiplex loop interface adapter (MLIA), cyclic encoder, power supply, and associated signal and power cabling. Also, a maintenance panel, power distribution box, cable enclosure assembly, and two air blower and filter assemblies are included with the cabinet. The processor is contained in the communications processor card cage, which also provides the card slots in the 64K MOS memory, cyclic encoder, MLIA, and CYBER coupler cards. One loop multiplexer card and from 1 to 16 CLA cards can be located in the loop multiplexer and CLA card cage.

2551-4 NPU EQUIPMENT

The 2551-4 NPU (figure 1-5) contains the same components as the 2551-3 and includes an additional loop multiplexer and CLA card cage. The NPU can service up to 64 CLAs in the system; thus, the 2551-4 NPU has double the line capacity of the 2551-3. The cabinet has EMI shielding.

Communications Processor

The processor card cage used in 2551-1, 2551-2, 2551-3, and 2551-4 NPUs contains the following functional elements:

- Microprocessor
- Main memory
- Maintenance panel interface
- Cyclic encoder
- Input/output Teletype interface
- Tape cassette controller (optional)
- Multiplex loop interface adapter
- CYBER Coupler (2551-3 and 2551-4 only)

2552-2 NPU EQUIPMENT

The 2552-2 NPU contains two processor card cages mounted in a double-bay cabinet. See figures 1-2 and 1-6. The base processor card cage is located in bay 0 and the multiplex processor card cage is in bay 1. Each bay contains identical cabinet wiring, power supplies, air blower and filter assemblies, and tape cassette drives. The 2552-2 NPU contains two loop multiplexers and communications line adapter card cages located in bay 0. Two optional card cages can be installed in bay 1 to expand communications line capacity up to 128 lines.

Base Processor

The base processor card cage contains those elements required to perform the main data storage and processing functions. It includes the 32K-word memory. The base processor components are as follows:

- Microprocessor
- Main memory
- Maintenance panel interface
- Cyclic encoder
- Input/output TTY interface
- Tape cassette controller

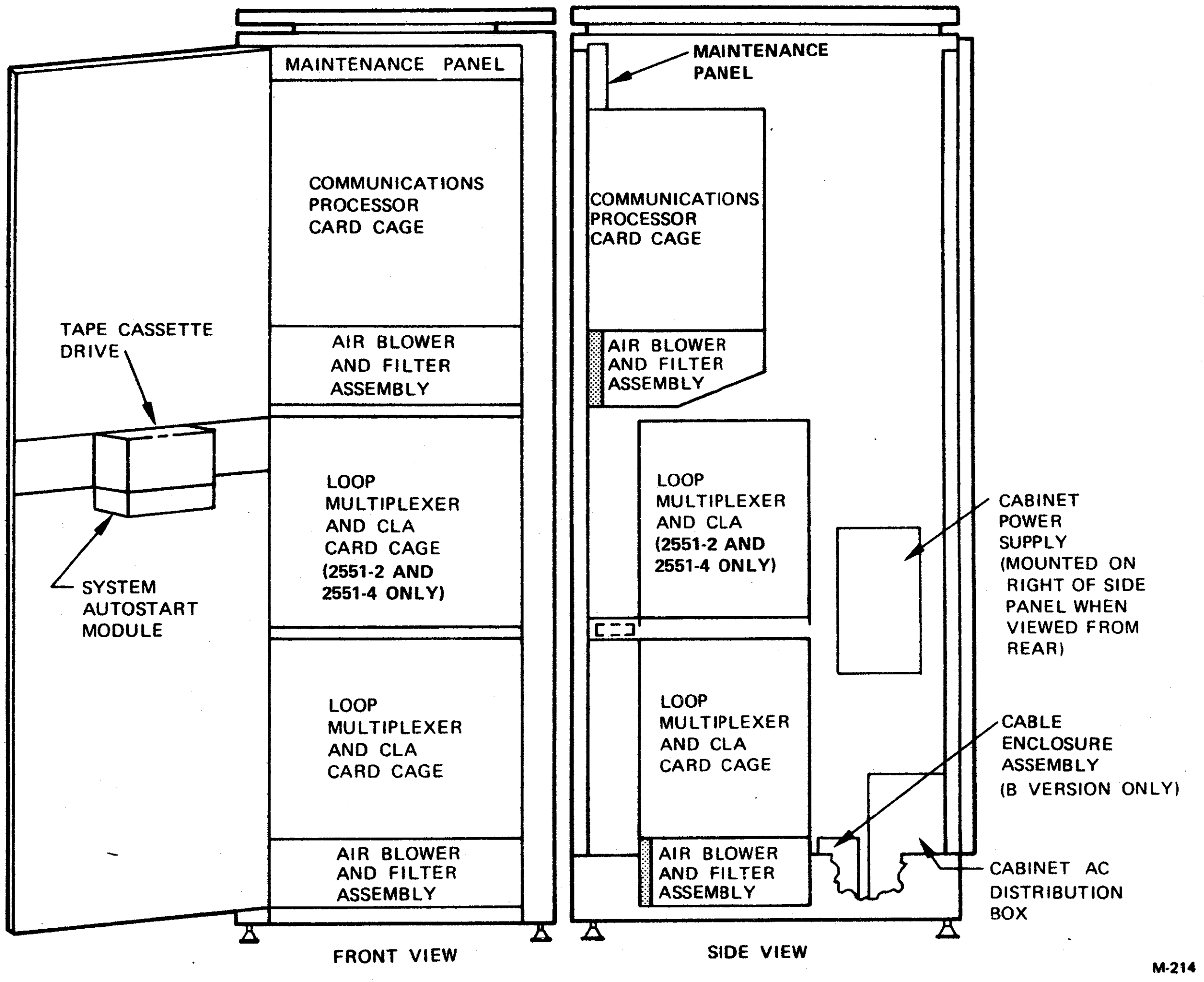

MAINTENANCE PANEL
COMMUNICATIONS PROCESSOR CARD CAGE
TAPE CASSETTE DRIVE
AIR BLOWER AND FILTER ASSEMBLY
LOOP MULTIPLEXER AND CLA CARD CAGE (2551-2 AND 2551-4 ONLY)
SYSTEM AUTOSTART MODULE
LOOP MULTIPLEXER AND CLA CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
FRONT VIEW
MAINTENANCE PANEL
COMMUNICATIONS PROCESSOR CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
LOOP MULTIPLEXER AND CLA (2551-2 AND 2551-4 ONLY)
CABINET POWER SUPPLY (MOUNTED ON RIGHT OF SIDE PANEL WHEN VIEWED FROM REAR)
LOOP MULTIPLEXER AND CLA CARD CAGE
CABLE ENCLOSURE ASSEMBLY (B VERSION ONLY)
AIR BLOWER AND FILTER ASSEMBLY
CABINET AC DISTRIBUTION BOX
SIDE VIEW
M-214

Figure 1-5. 2551-1, 2551-2, 2551-3, and 2551-4 Network Processor Unit Front and Side Views

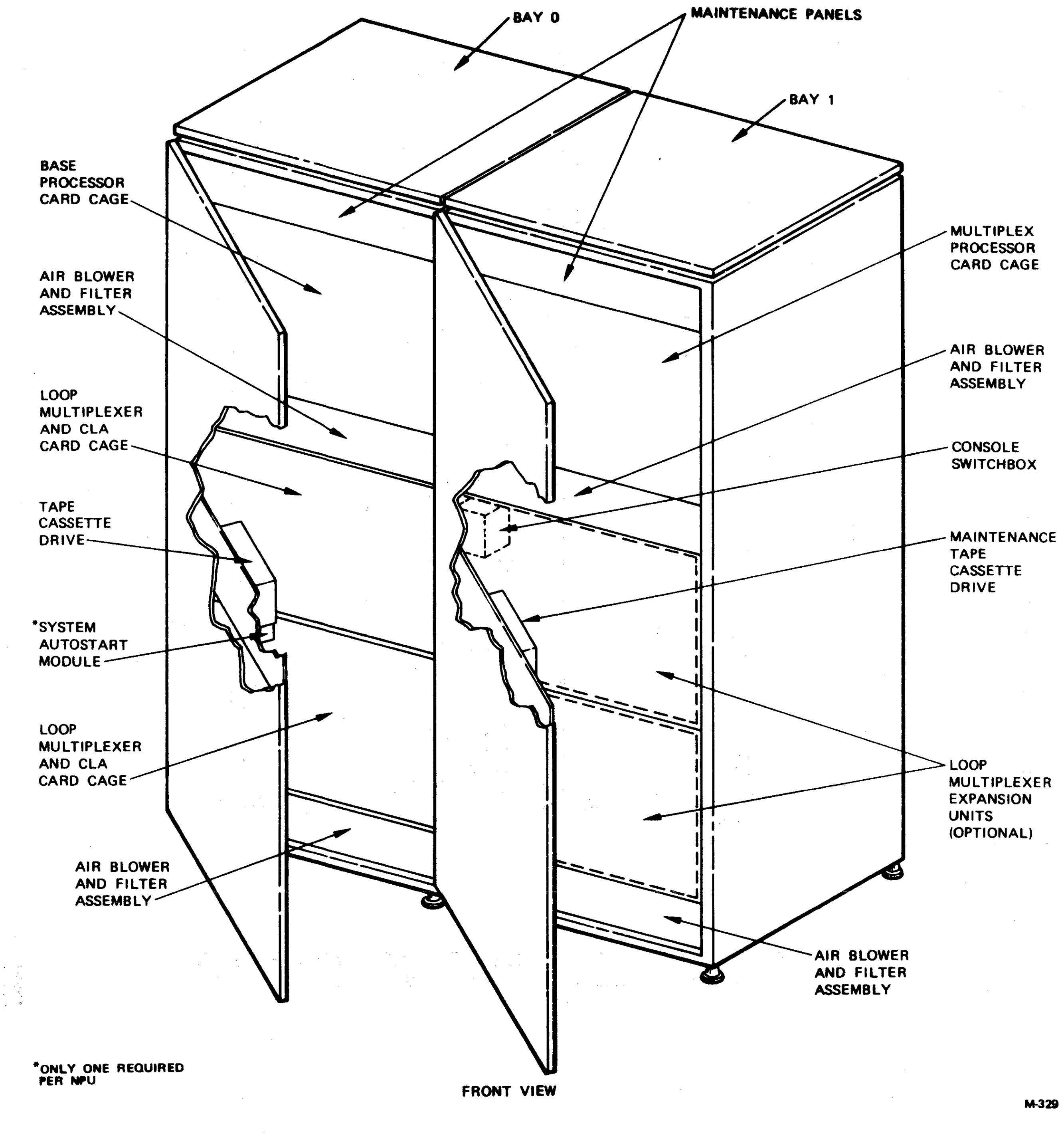

BAY 0
MAINTENANCE PANELS
BAY 1
BASE PROCESSOR CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
LOOP MULTIPLEXER AND CLA CARD CAGE
TAPE CASSETTE DRIVE
*SYSTEM AUTOSTART MODULE
LOOP MULTIPLEXER AND CLA CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
MULTIPLEX PROCESSOR CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
CONSOLE SWITCHBOX
MAINTENANCE TAPE CASSETTE DRIVE
LOOP MULTIPLEXER EXPANSION UNITS (OPTIONAL)
AIR BLOWER AND FILTER ASSEMBLY
*ONLY ONE REQUIRED PER NPU
FRONT VIEW
M-329

Figure 1-6. 2552-2 Network Processor Unit (Sheet 1 of 2)

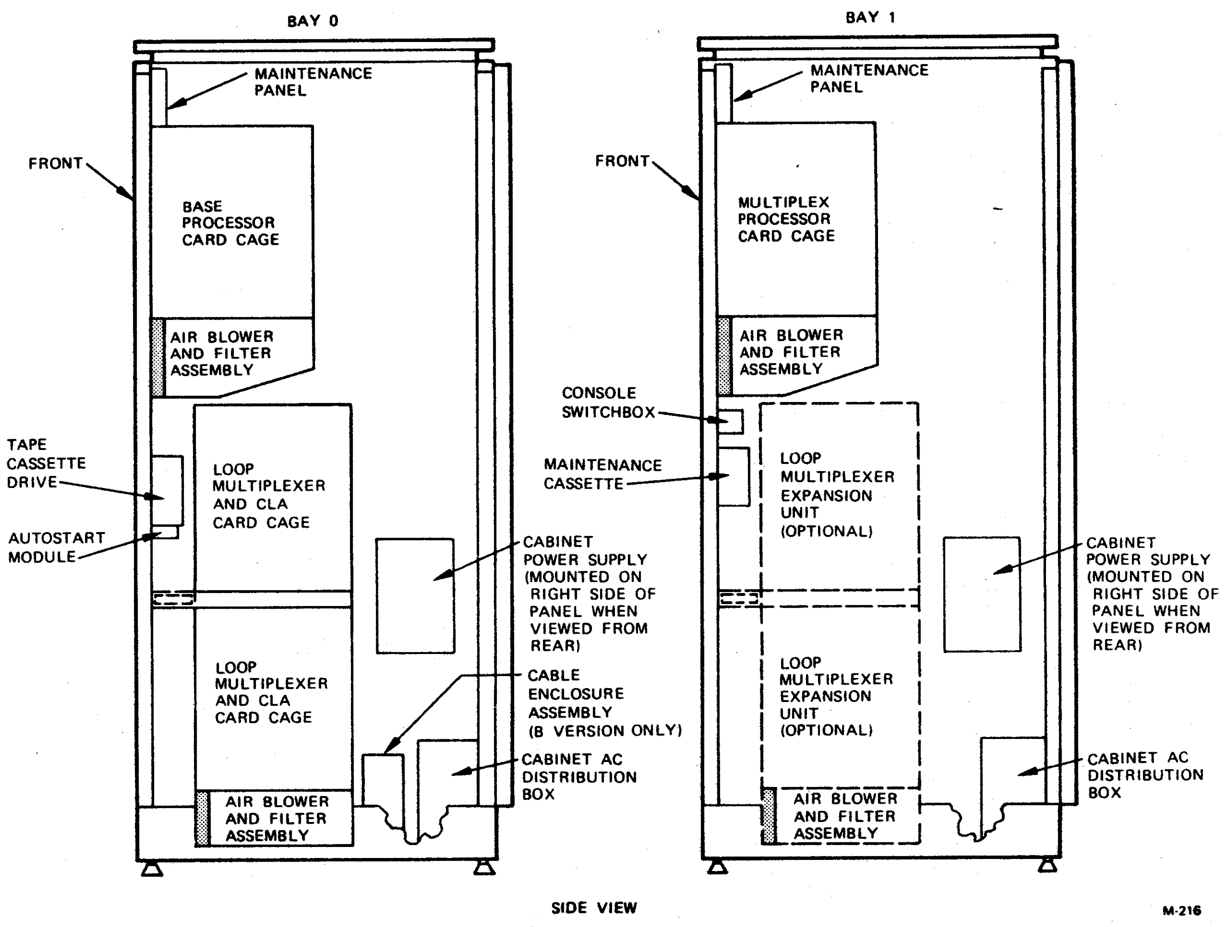

BAY 0
MAINTENANCE PANEL
FRONT
BASE PROCESSOR CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
TAPE CASSETTE DRIVE
AUTOSTART MODULE
LOOP MULTIPLEXER AND CLA CARD CAGE
LOOP MULTIPLEXER AND CLA CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
CABINET POWER SUPPLY (MOUNTED ON RIGHT SIDE OF PANEL WHEN VIEWED FROM REAR)
CABLE ENCLOSURE ASSEMBLY (B VERSION ONLY)
CABINET AC DISTRIBUTION BOX
BAY 1
MAINTENANCE PANEL
FRONT
MULTIPLEX PROCESSOR CARD CAGE
AIR BLOWER AND FILTER ASSEMBLY
CONSOLE SWITCHBOX
MAINTENANCE CASSETTE
LOOP MULTIPLEXER EXPANSION UNIT (OPTIONAL)
LOOP MULTIPLEXER EXPANSION UNIT (OPTIONAL)
AIR BLOWER AND FILTER ASSEMBLY
CABINET POWER SUPPLY (MOUNTED ON RIGHT SIDE OF PANEL WHEN VIEWED FROM REAR)
CABINET AC DISTRIBUTION BOX
SIDE VIEW
M-216

Figure 1-6. 2552-2 Network Processor Unit (Sheet 2 of 2)

Multiplex Processor

The multiplex processor card cage contains the elements required to perform the primary command processing functions and interfaces with the input and output loops. This processor has only a 16K-word memory and includes the multiplex loop interface adapter circuit cards. The multiplex processor components are as follows:

- Microprocessor
- Main memory
- Maintenance panel interface
- Cyclic encoder
- Input/output TTY interface
- Tape cassette controller
- Multiplex loop interface adapter

Cross-Control and Console Switch

The multiplex processor contains a cross-control module to generate interrupts and a console switch, which allows the NPU operator to select either the base or multiplex processor.

COMMUNICATIONS CONSOLE

A communications console is required to operate the NPU and may be either a teletype (TTY) or cathode-ray tube (CRT) conversational terminal. The console is not supplied with the standard NPU but is selected according to the user application. Among the CDC products, the user may select a 722-10, 752 Display Terminal, or a 1711-4, -5 (KSR TTY), or a 1713-4, -5 (AST TTY) terminal as the communications console.

COMMUNICATIONS COUPLER

The communications couplers used in conjunction with the 2551-1, 2551-2, 2551-3, 2551-4, and 2552-2 NPU configurations can be one or two of the following types:

- CYBER communications coupler - to interface with CYBER series 70/170 or 6000 series host computers.

- 6671/6676 emulation coupler - to interface with CYBER SERIES 70/170 or 6000 series host computers, permitting the NPU to emulate a 6671 or 6676 Data Set Controller.

- 3000L coupler - to interface with 3000L series host computers

MULTIPLEX SUBSYSTEM

The multiplex subsystem consists of the communications line adapters, the loop multiplexer, and the multiplex loop interface adapter.

Communications Line Adapters

Each communications line adapter circuit card contains two complete, independent circuits. The communications line adapter is installed in the loop multiplexer and communications line adapter card cage. The card cage accommodates up to 16 communications line adapter cards which can service a maximum of 32 communications lines.

Loop Multiplexer

The loop multiplexer is located in the loop multiplexer and communications line adapter card cage. Each loop multiplexer can accommodate 32 communications line adapters (16 communications line adapter cards) and can be series-connected to a maximum of eight loop multiplexers per loop for up to 254 communications line adapters.

Multiplex Loop Interface Adapter

The multiplex loop interface adapter consists of three circuit cards located in the communications processor card cage.

MAINTENANCE PANEL

A maintenance panel is located above the processor in each bay. See figures 1-5 and 1-6. Controls and indicators on the panel can be used by the operator to monitor the status of the system.

CASSETTE TAPE DRIVE (OPTIONAL)

A cassette tape drive is mounted on the door of each cabinet bay. See figures 1-5 and 1-6. The drive operates at a speed of 7.5 inches (190 mm) per second and recording density is 800 bits per inch (31.5 bits per mm). Rewind speed is approximately 50 inches (1270 mm) per second. The drive uses Phillips style cassettes.

SYSTEM AUTOSTART MODULE

The system autostart module is an assembly attached to the bottom of the cassette tape drive. See figures 1-5 and 1-6.

POWER SUPPLIES

The NPU contains a power supply which provides -5 V dc, +5 V dc, -12 V dc and +12 V dc outputs. In the 2551-1, 2551-2, 2551-3, and 2551-4 NPUs the power supply is mounted internally on the right rear cabinet wall. See figure 1-5. In the 2552-2 NPU a power supply is mounted on the right rear cabinet wall of each bay. See figure 1-6.

CABLING

Cabling for the NPU consists of signal cables and power cables. Refer to the 2550 series site preparation manual (see preface) for information on cabling.

AIR BLOWERS AND FILTER ASSEMBLIES

Two air blowers and filter assemblies are provided with the basic NPU, and additional units are provided to accommodate extra processor and communications line extension units in the 2552-2 NPU. Refer to the 2550 series site preparation manual (see preface) for information on blower and filter assemblies.

EXPANSION SYSTEMS

The NPU can be expanded for greater use by addition of optional standard products and upgrade kits, such as host couplers and communications line expansion units.

ELECTROMAGNETIC INTERFERENCE PROTECTION

Electromagnetic compatibility measures are included in the NPU design. However, the NPU should not be located in the area of influence by strong, electromagnetic interferences (EMI) such as radar, X-ray equipment, radio signals, and electrical equipment. EMI protection is provided in the NPU cabinets (B version) by special shielding on the top and sides of the cabinets. A cable enclosure assembly is included to ensure the CYBER cable shields are grounded to the cabinet. Additional provisions for grounding communications line adapter cable shields consists of copper pressure strips that contact the cable shields at the bottom entrance to the cabinets.

FUNCTIONAL DESCRIPTION

The major functional elements of the NPU are the processor and memory, the coupler, and the multiplex subsystem. See figure 1-7.

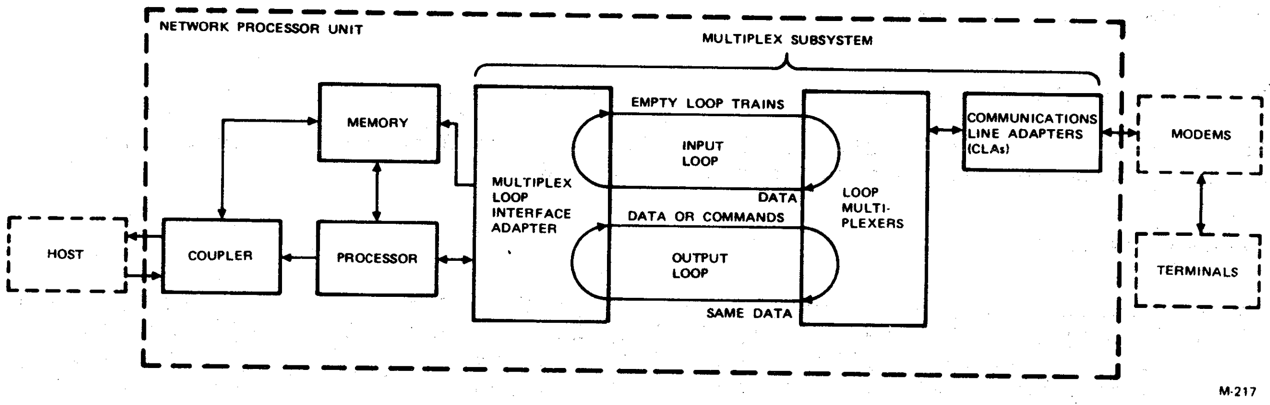


Figure 1-7. NPU Simplified Block Diagram

PROCESSOR

The processor, the primary element, contains firmware (microinstructions) and software (macroinstructions) to control the coupler and multiplex subsystem. Specific parameters of the NPU are listed in table 1-1.

NOTE

> Macroinstructions are compatible with the CDC 1700 instruction repertoire and are stored in the processor main memory. Microinstructions control operation of the processor and are stored in the processor micromemory.

The processor has a direct memory access channel to the host computer for both input and output and for input only from the multiplex subsystem. The internal data channel is controlled mostly by firmware microinstructions and provides a high throughput of 10K characters per second between the multiplex subsystem and the processor. This forms the main access path to and from the processor and the multiplex subsystem.

COUPLER

The processor is linked to the host computer through the coupler which provides data paths and synchronizing clocks between the computers. The coupler provides the means for direct memory-to-memory transfers between the host and the base processor at a 1-MHz character rate. It is controlled by both host and NPU software commands. The coupler also enables the host to start, clear, and stop the processor. In addition to single-word transfers, the coupler provides block data transfers in either direction (e.g., downline load, upline dump) plus the ability to store and retrieve data via the standard buffer format. The coupler selected must be compatible with the host computer.

MULTIPLEX SUBSYSTEM

The multiplex subsystem transfers digital information on input and output loops between the processor and the terminals. The input and output loops are two high-speed serial data transmission links. The multiplex feature allows transmission of messages from many communications lines over one circuit by sampling data (up to 8 bits per byte) on each line and assembling the data in a data train (loop cells).

Data transferred to and from the loops via the multiplex loop interface adapter are checked for accuracy by use of a cyclic redundancy checksum generated by the cyclic encoder. The cyclic redundancy checksum characters are used in error code generation and verification of both incoming and outgoing data to the terminals.

The multiplex loop interface adapter receives commands from the processor and accepts or provides parallel data to a circulating buffer in the memory. The input loop to the multiplex loop interface adapter carries input data and status as well as output data demand signals from the communications line adapters to the memory. The output loop carries output data and supervision commands to the communications line adapters.

TABLE 1-1. NPU FUNCTIONAL CHARACTERISTICS

<u>Main Memory</u>

Capacity: (standard)	2551-1 - 32K words (32 768) AT241-B
	2551-2 - 32K words (32 768) AT241-B
	2551-3 - (2) 32K words (32 768) AT241-B
	2551-4 - (3) 32K words (32 768) AT241-B
	2552-2 - 48K words (49 152) AT241-B and AT275-B

Maximum capacity can be extended to 131 072 (128K) words for all configurations by the addition of CDC 2554-32 (AT241-B) 32K MOS memory expansion modules. If a 2554-16 (AT275-B) module is fitted, as in the 2552-2 configuration, it must be replaced by a 2554-32 (AT241-B) module.

Word length: 16 bits (plus a parity bit and protect bit)

Read-Access Time: 550 nanoseconds

Memory Addressing Modes: 8

Memory Word and Region Protection

Memory Parity Protection

Direct Memory Access (4 users)

External CPU Access

Automatic Interfaced Refresh

<u>Micromemory (2K RAM)</u>

Capacity: 3K (3072) words (2K (2048) words read/write; 1K (1024) words read only emulator)

Word Length: 32 bits

Read-Access Time: 168 nanoseconds

<u>Micromemory (8K RAM)</u>

Capacity: 8K (8192) words (7K (7168) words read/write:
1K (1024 words read-only emulator)

Word Length: 32 bits

Read-Access Time: 168 nanoseconds

TABLE 1-1. NPU FUNCTIONAL CHARACTERISTICS (Contd)

Interrupts	
Macrointerrupts:	16 (corresponds to CDC 1700 interrupts)
Microinterrupts:	16 (internal)
Registers:	256 words
Files:	File 1, 256 words; File 2, 32 words

The loop multiplexer accepts input data in parallel from the communications line adapters and serializes the data. The loop multiplexer also assembles loop cells from the output loop and presents the data in parallel form to the communications line adapters.

The communications line adapters provide the interface between the loop multiplexer and the terminals with or without a modem. On input from the terminals, the communications line adapters are serial-to-parallel converters, assembling serial data at the signal rate of the terminal, and transferring the data to the loop multiplexer in 8-bit bytes. On output, the communications line adapters are parallel-to-serial converters, receiving the data characters in bytes of 8 bits from the loop multiplexer and outputting them serially at the signal rate of the terminal.

OPERATING PROGRAMS AND DIAGNOSTICS

NPU operating programs and diagnostic test programs are loaded from cassette tapes. The cassette tape controller serves as the access path for all data transfer between the tape drive and the processor. The cassette tape drive can be used as a standard peripheral or as a deadstart device.

When the NPU is program loaded and placed on-line, the NPU operation is initiated and controlled by the host. NPU operation is monitored by the host and by internal checks within the NPU. Processor operation can be interrupted at any time of error detection within the multiplex subsystem or on command from the host.

The system autostart module monitors the program status. If a program error is detected, a system master clear is issued, and the program tape is automatically reloaded. Then the system restarts and runs until another error is detected and the reload cycle automatically repeats. If no error is detected, the autostart reload operation remains inhibited.

The operator can access the NPU via the console (TTY or CRT) or by the maintenance panel to perform all common computer control panel functions. The panel interface card in the processor provides the interface between the operator panels and the processor main bus and internal processor control signals. Operations performed at the maintenance panel are local and those performed at the console are remote.

The 2552-2 NPU with dual processors provides better response time and greatly increased message handling capability to accommodate any user requirements. See figure 1-2. In the dual processor configuration, the processors share the NPU functions which allows different operations to be performed simultaneously. The base processor performs host signal interface operations, main data storage, and processing functions. The multiplex processor performs the primary command functions to control the multiplex subsystem operations.

MAINTENANCE

Maintenance and troubleshooting procedures for the NPU consist of a series of diagnostic test procedures which are performed on-line and off-line. On-line diagnostics are performed to determine whether the fault is with the vendor equipment, such as communications drives, modems, and input power, or with CDC equipment.

The objective of the off-line diagnostics is to allow fault isolation for all components. Fault detection is performed automatically by the diagnostic test program. Fault isolation is performed by the operator observing the results of the tests and determining, with the aid of charts, which circuit card is at fault.

COMMUNICATIONS SOFTWARE

The communications software is divided into three major groups as follows:

- Host Processor Network: Access Software
- 255X Network Processor Unit: Communications Control Software
- 255X Support Software

The following paragraphs briefly describe the various functions and relationships of the software groups. See figure 1-8.

HOST SOFTWARE

The host operating system software controls the operations of a network operating system. The network access software (also resident in the host) serves as the interface between the host operating system software and the communications control software resident in the NPU. The host applications programs process messages passed to them and also pass messages to the network to be distributed to terminals.

COMMUNICATIONS CONTROL SOFTWARE

The communications control software consists of base system software, network communications software, and interface programs. Available programs are as follows:

- Communications Control Program (CCP), NOS Version, used on CYBER 6000 and 70/170 computers.
- Communications Control INTERCOM (CCI), NOS/BE Version, used on CYBER 70/170 computer.
- Communications Control Master (CCM), used on 3000L computers.

For detailed information on the communications control software, refer to the applicable software reference manual.

Base System Software

The base system software, which includes the multiplex subsystem, comprises those elements of the communications control software that are required for

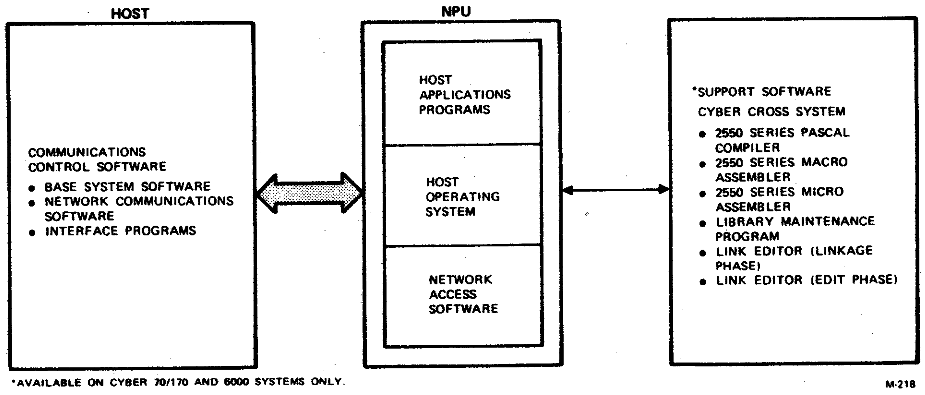

HOST
NPU
COMMUNICATIONS CONTROL SOFTWARE
• BASE SYSTEM SOFTWARE
• NETWORK COMMUNICATIONS SOFTWARE
• INTERFACE PROGRAMS
HOST APPLICATIONS PROGRAMS
HOST OPERATING SYSTEM
NETWORK ACCESS SOFTWARE
*SUPPORT SOFTWARE
CYBER CROSS SYSTEM
• 2550 SERIES PASCAL COMPILER
• 2550 SERIES MACRO ASSEMBLER
• 2550 SERIES MICRO ASSEMBLER
• LIBRARY MAINTENANCE PROGRAM
• LINK EDITOR (LINKAGE PHASE)
• LINK EDITOR (EDIT PHASE)
*AVAILABLE ON CYBER 70/170 AND 6000 SYSTEMS ONLY.
M-218

Figure 1-8. Communications Software Overview

all system applications. The base system software contains subroutines that provide:

- NPU resource management
- Memory space management
- Control of the local NPU console
- Facilities to add line-dependent or terminal dependent logic at various software levels.
- Multiplex subsystem elements operating with NPU hardware to establish data and control paths for information exchange between the communications lines and appropriate protocol handlers (terminal interface program) within the NPU.

Network Communications Software

The network communications software accomplishes the following:

- Network routing and service message processing to establish line and terminal configuration.
- Reports alarm conditions and error and traffic statistics.
- Provides common support modules for terminal interface programs.

Interface Programs

The interface programs provide the logical interaction necessary to transmit information to and from the NPU.

- Host Interface Program (HIP) accomplishes block data transfers with the host computer, monitors the host for failure and recovery, and reports host status to the NPU.
- Terminal Interface Programs (TIPs) provide the logic to control the line and terminal protocol and to ensure orderly transmission (in either direction) between the NPU and the connected terminal. These programs transform data between terminal protocols and block protocols.
- Link Interface Program (LIP) accomplishes data transfers between a local and a remote NPU. A special protocol (CDCCP) is used for these transfers. The LIPs are responsible for transforming data between CDCCP and block protocols at both ends of the trunk.

SUPPORT SOFTWARE

The CCP support software (CYBER CROSS System) includes a PASCAL compiler, two assemblers, and system generation programs for the development and maintenance of CCP software. The CYBER CROSS System executes only on the CYBER 70/170 series computer systems and executes under control of NOS as an application system. Refer to applicable documents for information on support software on the CYBER CROSS System.

FUNCTIONAL DESCRIPTION 2

This section contains a functional description of the NPU subsystems and components.

NPU MAIN DATA PATHS

The NPU receives input data from the modems and terminals, via the communications line adapters and loop multiplexer, and transforms the input data to a format compatible with the internal processing of the NPU. See figures 2-1 and 2-2. The multiplex loop interface adapter controls the data flow from the loop multiplexer and supplies data to the memory buffers under firmware and internal interrupt control. The cyclic encoder generates a cyclic redundancy checksum to verify accuracy of the data. The processor removes all protocol and language-dependent characters of the modified input data and provides storage and access to the data under firmware control. The communications coupler interfaces the host and serves as a transmittal device by moving NPU data in buffer format to the host memory. Conversely, the coupler transports data from the host to the NPU and stores it in the internal format used by the NPU processor. Data flow through the NPU from the communications line adapters in the direction of the host is referred to as inbound traffic. Data flow from the host through the NPU to the communications line adapters is outbound traffic.

The tape cassette is used for autoloading NPU microprograms and diagnostic test programs. The communications console allows operator remote control over the display of the NPU processing data and commands. The console normally is the primary NPU control, but may be used for maintenance or troubleshooting of the NPU. During normal operation the NPU operates unattended.

The maintenance panel gives the operator, programmer or maintenance personnel direct control over the display of NPU processing and commands from the NPU. The maintenance panel is used primarily for troubleshooting the NPU system or examining elements of the operating program.

The dual processor NPU functions the same as the single processor NPU except for expanded equipment and capabilities. In general, the main difference is that the single processor functions are divided between two processors for increased performance. In the following paragraphs the single processor system and the dual processor system are described separately.

BASIC EMULATION CONCEPTS

The 1700 series of computers, including the 1704 described in appendix A, consists of special-purpose hardware designed to perform the tasks indicated by the repertoire of instructions. The 255X system contains a totally different processor which responds to a series of instructions at the micro level. Microcode is a basic and flexible form of control, exercising and sequencing the flow between the various hardware elements and the operating

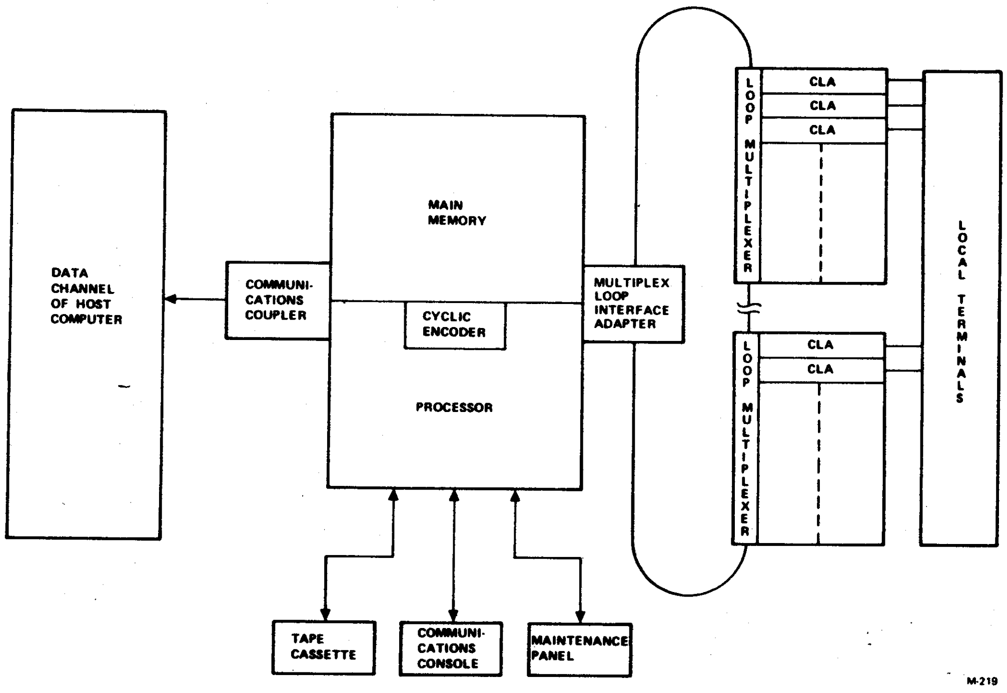


Figure 2-1. 2551-1, 2551-2, 2551-3, and 2551-4 NPU Major Functional Areas and Data Paths

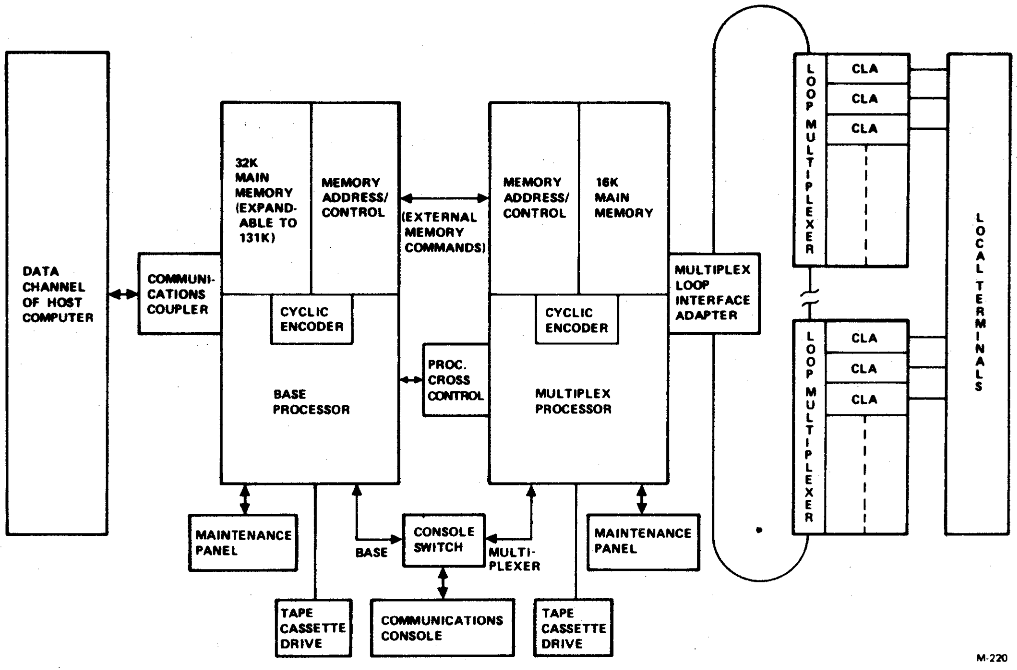


Figure 2-2. 2552-2 NPU Major Functional Areas and Data Paths

modes of these same elements. Instructions used by the microprocessor are stored as 32-bit words in the read/write and/or read-only portions of micromemory. Real-time operations are accomplished by executing microcode stored in the read/write portion of micromemory. These real-time operations are concerned primarily with time-critical tasks, such as character traffic on the communications lines.

Other, nonreal-time operations are written in 1700-style code stored in the processor main memory. These are executed through emulation. By this means, instructions are fetched from main memory one at a time, the various bit structures in an instruction are analyzed for content, converted into one or more microinstructions capable of performing the operation requested, and executed. The operation then continues in this manner, producing the same results as would be accomplished by the code executing in a 1700 series machine. To permit this emulation, a firmware instruction repertoire is included, stored in read/write memory. This emulation operation becomes practical, even though several microinstructions are required for each microinstruction (1700), because the microprocessor is significantly faster and simpler. Since the repertoire of instructions to be emulated is determined simply by the contents of a read-only memory, addition of a series of instruction set enhancements becomes practical.

INTERRUPT SYSTEM

While emulating the 1700 computer, the NPU microprocessor firmware (controlware) program actually accesses, under microprogram control, all 32 of the interrupts. As a byproduct of this processing, the 16 external interrupts are made available to the macroinstruction while the 16 internal interrupts are available only to the microinstruction, either within the emulation controlware package or the special communications firmware. It is more accurate to define the interrupts as macro- and microinterrupts rather than external and internal interrupts, respectively.

MACROINTERRUPTS

Macrointerrupts are associated with external I/O devices and perform traditional tasks such as flagging end-of-operation status and fault conditions. Microinterrupts, however, perform the important internal tasks of establishing priorities and queuing up tasks and in general providing communication and signaling between the various firmware/controlware routines. At the microinstruction level, there is no executive or monitor or scheduler; these functions must be performed by the microinterrupt priority and processing system.

MICROINTERRUPTS

In general, the microinterrupt system responds to three groups of interrupts: emulator interrupts, special firmware interrupts, and high-priority data such as multiplex loop interface adapter microinterrupts of which two are available at the microinterrupt level. Processing is typically split into low- and high-priority tasks so that time-critical tasks are handled immediately with subsequent processing set up for deferred processing in response to a lower priority microinterrupt.

Status/Mode and Interrupt Bits

The 1700 computers use interrupt registers in which pulse-type interrupts are stored until processed or cleared. Because of the emulation nature of interrupt handling, the NPU processor uses a different approach. The macrointerrupt and microinterrupt registers operate on a direct current rather than pulse basis. An external device must hold the interrupt line active until the interrupt has been serviced. The interrupt registers are used to prevent the interrupt status from changing while they are being interrogated. As such, the interrupt registers are clocked as a part of the reading process. The signals are stored in the status mode register. The bits of this register can be wired so that they can be set via external means and cleared under program control.

Microprocessing is organized so that certain response time standards are established for macrointerrupt response time. This is necessary since activating a macrointerrupt does not interrupt the processor but merely sets a bit in an interrupt register which can then be read and acted upon by the microprogramming. Only through the action of the firmware/controlware is the effect of the macrointerrupt felt. Because of this, two service standards are provided:

- Emulation controlware examines the state of the interrupt line through the mask during the read-next-instruction phase of each emulated macroinstruction. When an active interrupt is found (and the interrupt system enabled), normal macroprogram execution is delayed and macrointerrupt instruction routines are executed.

- Special firmware routines are organized so that every 20 microseconds (or less) the interrupt system is interrogated and, if an interrupt is found, processing is delayed.

Users can expect an interrupt latency time of no more than 20 microseconds (for the highest priority interrupt). This accommodates 56-kilobit communications lines.

Besides storing events, status/mode bits also serve another special function of driving special control lines for external I/O operation. Thus I/O timing, such as strobing and handshaking, is driven under firmware/controlware control rather than by dedicated hardware.

Automatic Data Transfer Mode

Devices capable of automatic data transfer mode operations require both macrointerrupts and microinterrupts. Such devices include the tape cassette and the optional card reader and line printer. Each time a character is available and required during automatic data transfer operations, the microinterrupt line is activated. (For circuit simplicity the macrointerrupt line can also be raised, but the microinterrupt will be seen first by the controlware and it is therefore the one upon which action will be taken.) Thus, character transfer is handled at the microlevel.

Multiplex Loop Interface Adapter Interrupts

Owing to their time-critical nature, two multiplex loop interface adapter interrupts are available as microinterrupts. However, these microinterrupts are not available to macroprograms such as MSMP17/ODS diagnostics. Thus, to provide for diagnosability, these two interrupts are parallel-wired as both macrointerrupts and microinterrupts. The third multiplex loop interface adapter is of routine priority and is available as a macrointerrupt only.

Real-Time-Clock Interrupts

Every 3.3 milliseconds a real-time-clock microinterrupt occurs. Via program control, this interrupt is counted down n times (n is a programmable limit). Only when n microinterrupts have occurred is a macrointerrupt generated. Thus, real-time-clock microinterrupts occur every 3.3 milliseconds and real-time-clock macrointerrupts occur every 3.3 divided by n milliseconds.

Interrupt Holding Registers

The interrupt holding registers are designated as I1 and I2. Register I1 provides internal microinterrupts; register I2 provides external macrointerrupts. Each register contains 16 bits, and the bits are numbered from left to right as follows:

Register	Bit Numbers
I1	I100 to I115
I2	I200 to I215

The priority levels of the interrupt bits are in numerical order. Bit I100 has the highest priority in register I1; bit I115 has the lowest priority. Similarly, bit I200 has the highest priority in register I2, and bit I215 has the lowest. The bit assignments for registers I1 and I2 are listed in table 2-1.

NPU COMPONENTS FUNCTIONAL DESCRIPTION

SINGLE PROCESSOR

The single processor provides buffer storage for input and output data, storage and execution of various firmware and software programs, and provides the hardware interfaces for both the host computer and the multiplexing subsystem. For a detailed functional description, refer to the Basic Microprogrammable Processor Hardware Maintenance Manual (see preface).

The functional elements of the single processor are as follows:

- Microprocessor
- Panel/Console Interface
- I/O Subsystem
- Cyclic Encoder
- Main Memory

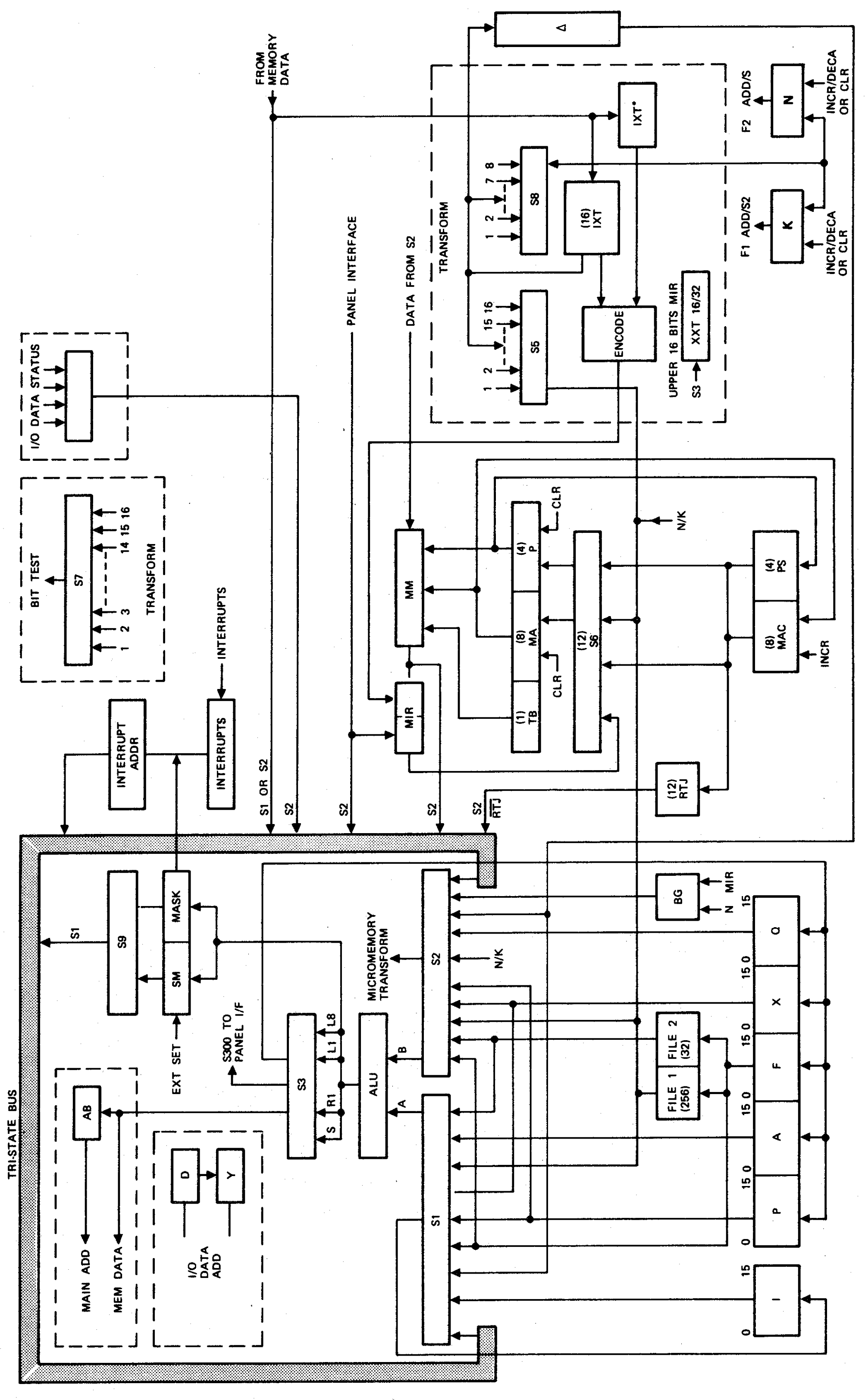
The main data paths for the single processor configuration are shown in figure 2-1.

TABLE 2-1. INTERRUPT BIT ASSIGNMENTS

Register Bit	Function
I100	Multiplex Loop Interface Adapter
I101	Communications Console (TTY or CRT)
I102	Multiplex Loop Interface Adapter
I103	(unassigned)
I104	2571/2570 Line Printer Controller
I105	(unassigned)
I106	(unassigned)
I107	Tape Cassette Controller
I108	Real-Time Clock
I109	(unassigned)
I110	(unassigned)
I111	2571/2572 Card Reader Controller
I112	1700 Emulator
I113	1700 Emulator
I114	1700 Emulator
I115	1700 Emulator
I200	Power Fail/Memory Parity
I201	Communications Console
I202	Multiplex Loop Interface Adapter
I203	Multiplex Subsystem Priority 3 (source is SM205)
I204	2571/2570 Line Printer Controller
I205	CYBER Coupler 2 (2558-1)
I206	CYBER Coupler
I207	Tape Cassette Controller
I208	Real-Time Clock
I209	1732/608-609 Mag Tape Controller (QSE)
I210	1740-1742 Line Printer Controller (QSE)
I211	2571/2572 Card Reader Controller
I212	Multiplex Loop Interface Adapter
I213	Multiplex Loop Interface Adapter
I214	(unassigned)
I215	Macro Breakpoint

Microprocessor

The microprocessor, the controlling element within the processor, is a small-scale, stored-program, parallel-mode digital computer. See figure 2-3. The microprocessor utilizes firmware (microinstructions) stored in a read-only memory (ROM) emulator to control the processing elements which permits execution of the main program (macroinstructions) residing in main memory. It also uses controlware (microinstructions) stored in a read-write memory (micromemory) to control the multiplexing subsystem and communications coupler. A direct memory access channel accepts both input and output data traffic from the host computer plus input traffic from the multiplex subsystem. An internal data channel serves as the main access path for the multiplexing subsystem and the various peripherals. The internal data channel handles data traffic unbuffered in both directions and operates under microinstruction emulator control.



NOTE: ALL ELEMENTS ENCLOSED IN DASHED LINES ARE PART OF THE MICROPROCESSOR.

Figure 2-3. Microprocessor Functional Block Diagram

The processor contains two separate read/write memory units which are identified as main memory and micromemory. Micromemory, a part of the microprocessor, has smaller capacity (typically 2K x 32), but faster access and cycle times. Each micromemory location contains a 64-bit double-word. The micromemory is a random access, semiconductor memory that contains an alterable random access memory nonviolatable and nonalterable read-only memory.

Main memory stores the macroinstruction program (and data); the emulator ROM and/or the micromemory stores the microinstruction program. Macroinstructions are read from main memory sequentially by a portion of the microprogram. The macroinstruction is then decoded into a series of microoperations by a combination hardware/controlware/firmware technique. The hardware portion of the macroinstruction decode is called a transform; the controlware/firmware portion is called an emulator (in the case of standard 1700-type instructions). The 1700 emulator consists of a series of fixed microinstructions contained in the read-only memory (ROM). The transform/emulator process causes the microprogram to form program branches, sets parameters, and performs arithmetic and logical operations.

A transformed/emulated enhanced 1700 language macroinstruction is performed as if it were one or more standard stored microinstruction(s). Numerous microinstructions may be required to execute one macroinstruction. Macroinstructions can be decoded either with or without the use of a transform feature; however, the use of transforms significantly reduces the number of microinstructions required and increases overall performance.

Arithmetic operations are performed by the arithmetic and logical unit shown in the left center portion of figure 2-3. The two inputs (A and B) are connected to a variety of input sources (e.g., I, P, A, F, X, and Q registers, Files 1 and 2, bit generator, N/K register, transform, tristate bus) via selectors S1 and S2, respectively. Arithmetic and logical unit functions include addition, subtraction, shifting left or right (arithmetic or logical) plus logical functions such as AND or NOT. Output of the arithmetic and logical unit (as is, shifted right one bit, shifted left one bit, or shifted left eight bits, depending on selector S3) then appears as inputs to the designated general purpose registers (P, A, F, X, Q). Files 1 and 2 are addressed by registers K and N, respectively.

Control is exercised through microinstructions from the micromemory. Addressing is controlled by a 4-bit page address, 8-bit memory address, and a 1-bit, upper/lower (T-bit) selector from the current microinstruction. Operating modes are established by setting/clearing bits in the status mode register. Bits in the status mode register are also used to control timing of the internal data channel operations. Of the 32 machine interrupts, 16 are designated macro- (external) interrupts and are available to the operating system program resident in main memory. The remaining interrupts are designated micro- (internal) interrupts and are available to the microprogram (firmware and/or controlware) for control of hardware elements and for intercommunication between firmware and controlware. Interrupts are selected via priority logic, read through an interrupt mask register set under program control.

ARITHMETIC AND LOGICAL UNIT

The arithmetic and logical unit contains the A, F, I, P, Q, and S registers and also provides S1, S2, and ALU look-ahead functions. The arithmetic and logical unit includes File 1, consisting of 16 bits in each of 256 locations, and File 2, which has 16 bits in each of 32 locations.

The arithmetic and logical unit provides the basic capabilities for arithmetic and logical operations within the microprocessor. The unit combines two input words: the A input (provided by S1) and the B input (provided by S2). These two inputs are combined according to the function code specified in the microinstruction; the result is delivered simultaneously to the output of the arithmetic and logical unit for use by any of the following:

- A specified destination register
- Status mode register
- Mask register
- Memory bus
- Maintenance panel

NOTE

All registers cannot provide inputs to both sides of the arithmetic and logical unit, nor receive outputs from the arithmetic and logical unit; see figure 2-3.

It is also possible for the microprocessor to ignore the output of the arithmetic and logical unit during a given operation. The results of arithmetic and logical unit operations, i.e., sign, zero, and magnitude, are available to the test-bit logic for instruction sequencing.

Arithmetic operations can be performed using either one's or two's complement arithmetic; the desired mode is selected via the microprogram, which controls the states of the applicable bits in the status mode register. The microprocessor operates in one's complement mode, but is converted to two's complement by the emulation firmware. This creates a special situation for processor access to main memory location of hexadecimal FFFF.

STATUS MODE INTERRUPT

The status mode interrupt function consists of two status mode registers, two interrupt holding registers, two mask registers, two priority encoders, plus associated selectors and other hardware elements. The two pairs of each results from separate elements for the processing of macro and micro-interrupts. Status mode registers contain 16 bits each. These bits are divided into operating mode, event occurrence, and timing and control functions.

Interrupt holding registers I1 and I2 provide internal microinterrupts and external macrointerrupts, respectively. Mask registers contain 16 bits each. These bits correspond to the interrupt lines with the same number (100 thru 115, 200 thru 215). The function of the two priority encoders is to look at interrupt lines (through the two mask registers) and produce a number equal to the highest priority interrupt currently active. This interrupt is then serviced by the processor. Microinterrupt processing takes priority over macrointerrupt processing.

Real-Time Clock - A real-time clock, part of the I/O subsystem, is designed to appear as a 1700 peripheral to the macrolevel software. A microlevel interrupt is generated 300 times per second (3.3 ms). Under program control, a macrolevel interrupt can be obtained at the same rate as the microlevel interrupt. Alternatively, a count value can be programmed to reduce the macrointerrupt rate by the count value. Two functions are available to the macrolevel program: Enable Limit Interrupt and Disable Limit Interrupt. Two status bits are also available to the macrolevel program: Limit Interrupt and Lost Count.

The limit interrupt being received by a macrolevel program is dependent on the appropriate mask register bit being set and the macrointerrupt enabled as with all other 1700 peripherals.

Microlevel code is capable of receiving a data interrupt from the real-time clock every 3-1/3 milliseconds (derived from a crystal oscillator). This interrupt is enabled anytime that the corresponding mask bit is set. Refer to information on real-time clock presented earlier in this section.

Parity - Each main memory word consists of 19 bits consisting of: 16 data word bits, 1 data word parity bit, 1 protect bit, and 1 protect parity bit.

The memory parity error function is used to indicate that either a word or protect parity error has occurred. A word parity error indicates that one of the memory word bits is wrong or the word parity bit is wrong. Word parity error has no effect on memory operation other than setting the parity error line; it may, however, place the validity of the memory word(s) in question. A protect parity error indicates that either the protect bit or the protect parity bit is wrong. It is only operative when the protect system is enabled by a switch on the maintenance panel.

TRANSFORM

The transform feature provides the microprogram with the capability of selecting any of several pattern of bits from the data transmission paths of the microprocessor. By this means, bit patterns from the enhanced micro-program are rearranged and regrouped to resemble more closely the instruction formats actually executed by the microprocessor. These bit patterns are used to form micromemory addresses which sequence the microprogram. The bit patterns are also used to set the contents of the K and N register, and the upper 16 bits of the microinterrupt register.

The 1700 transform provides the following functions: Selectors 5, 7, and 8; IXT and IXT*, Transform, Encode; and Transform Micromemory. This function contains memory protect violation logic, decimal correction logic, and a delta field generator for the lower 8 bits of each data word.

Panel/Console Interface

All communications between the microprocessor and the maintenance panel are routed through the panel interface. See figure 2-4. The communications console serves as the operator's primary point of interface with the system. Either a teletypewriter (TTY) or cathode ray tube (CRT) conversational display (with optional hard copy printer) can be used as the communications console, provided that the selected device has two-way, simultaneous, serial (full-duplex), ASCII, odd-parity characteristics. The communications console CRT interface is RS232-compatible; the TTY interface is a 20-milliampere current loop.

The communications console functions in either of two modes: panel or I/O. A LOCAL/REMOTE switch on the maintenance panel designates whether the panel or console is selected.

In the panel (local) mode, all communication between the console and the microprocessor is routed only through the I/O subsystem, and the console operates as a standard peripheral device. As such, it is under the program control of the macroprogram (operational).

In the I/O (remote) mode, the console essentially functions as a maintenance panel; in this case, input from the console is routed through the I/O TTY interface to the panel interface, and then to the microprocessor. Output from the microprocessor is routed via the panel interface, through the I/O TTY interface to the console.

The maintenance panel LOCAL switch position only can be used to develop microprocessor instructions, and it can direct execution of these instructions by the microprocessor. Instructions are transferred directly to the microinstruction registers. The panel provides a number of capabilities such as register display, macro- or microbreakpoint.

I/O Subsystem

The I/O subsystem includes a TTY interface and the I/O control for both CDC A/Q-type and NCR M05-type operations of the internal data channel. Either a TTY or CRT display can be attached provided that it functions as an asynchronous current-mode or RS232-compatible device. A device having similar functional characters but operating at TTL voltage levels, such as the tape cassette (deadstart portion), can also be connected. Four ASCII character rates up to 9600 bits per second can be accommodated.

In addition to the status mode bits that generate timing, operations of the internal data channel are controlled by two registers. The D register contains the data to be written on the internal data channel. The Y register, loaded through the D register, contains address information. Transfers can be either in or out, in A/Q or M05 format, under program control.

Cyclic Encoder

The cyclic encoder is a special algorithm hardware unit with the primary function of computing cyclic redundancy checksum and/or longitudinal redundancy checksum characters. Operating under microinstruction control, the encoder can use any of 16 preprogrammed polynomial of sixteenth degree or less as a generator to compute a checksum on any message character of from 1 to 8 bits in length.

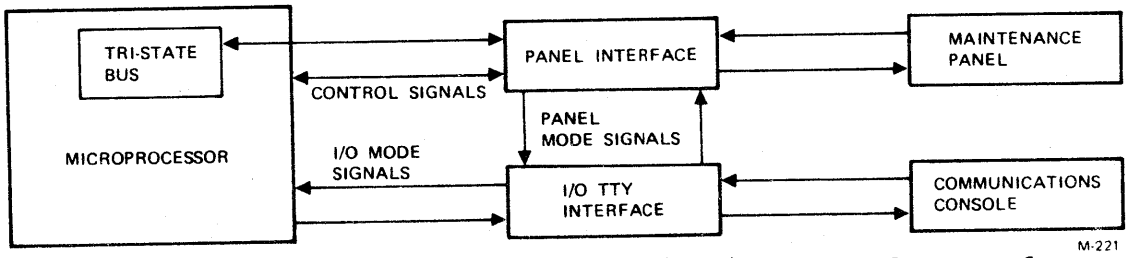


Figure 2-4. Maintenance Panel/Communications Console Interface Block Diagram

Cyclic redundancy checksum/longitudinal redundancy checksum polynomials are stored in read-only memory on the cyclic encoder. The cyclic encoder operates in a serial mode, computing a partial checksum for each message character it receives, one bit at a time.

After a checksum is computed for either an inbound or outbound message character, the results of the computation are accessible by the program until another message character is issued to the cyclic encoder. After this time a new checksum is available and the previous one is either destroyed (inbound) or stored (outbound) in the cyclic encoder but is no longer accessible by the program. Thus blocks of outbound data have the checksum precomputed on the entire block prior to output; input data characters accumulate partial checksums as the characters arrive, saving these partial checksums until the enter block is received and the final checksum becomes available.

Information is transferred to the cyclic encoder from the output of selector S1 in the microprocessor; and data is transferred out of the cyclic encoder via the input bus to selector S1. Transfer in both directions occurs under microinstruction (firmware) control. The cyclic encoder occupies the areas normally reserved for double precision arithmetic operation control and thus uses some double precision operations for its control.

The cyclic encoder computes a checksum on an 8-bit message character in less than 760 nanoseconds following execution of a load X* microinstruction. Computation time decreases by 61 nanoseconds per bit for message characters less than 8 bits.

Information is transferred to the cyclic encoder from selector S1 and stored in one of two registers designated A* or X*. Both registers are loadable only and cannot be used as utility registers because their outputs are not available from the card.

Main Memory

The single processor NPU has a 32K-word (2551-1 and 2551-2), 64K-word (2551-3), 96K-word (2551-4), main memory. The main memory modules comprise a one-bank system; that is, the modules are interconnected to preclude simultaneous read operations from two or more locations in main memory. However, there are two ports for main memory, which provide independent data and control paths to memory. These ports are identified as input/output (I/O) and direct memory access. The I/O port serves as the normal access path for the processor to main memory. The direct memory access port permits concurrent operation of the memory and processor.

Within main memory, user requests are coordinated, processed, and controlled by the memory management system. Six user requests can be accommodated: processor (lowest priority), direct memory access (4), and refresh (highest priority). The direct memory access is further subdivided by use of a 4-position scanner, thus enabling a total of six sources (including refresh) to access the memory.

Each of the 64 row addresses must be accessed once every two milliseconds (minimum) to maintain the data. These accesses are automatically inserted approximately once every 30 microseconds. Refresh thus constitutes less than two percent of the available memory cycle.

Memory can be accessed by the processor in either full-cycle or split-cycle mode. Selection of the mode to be used is determined by whether or not the protect system is needed.

DUAL PROCESSOR

Two processors are employed in the dual processor NPU, each performing half of the processing functions, thereby providing a significant performance increase. The base processor functions the same as the processor in the single processor NPU except that the multiplex loop interface function is absent. The multiplex processor is also the same basic unit except that the communications coupler function and protocol-dependent program functions are absent. Data communications between the processors occurs via common memory portions of the base processor main memory, accessible by either processor. Main memory in the base processor is 32K words, expandable to 128K. The multiplex processor requires only 16K of local memory, usually not expanded. It performs the multiplex interface function. The major functional areas and data paths for the two processors are shown in figure 2-2.

Operationally, the base processor interfaces with the host via the coupler and the multiplex processor interfaces with the communications lines via the multiplex subsystem. Control communication between the two processors is achieved through two interface devices: a console switch and a processor cross control module.

Console Switch

The console switch allows the system operator to communicate with either the base or the multiplex processor with one console.

Processor Cross Control

The processor cross control interconnections provide the following functions:

- Cross-interrupt - permits a status mode bit generated by either a base or multiplex processor to generate an interrupt in the other processor, which in effect alerts the other processor of a common operation or function to perform. The actual operation is determined by reading the contents of a communications area in common memory.

- Remote start - consists of a status bit (pulse) generated by the level transition of a base processor status mode bit. The status mode bit going off generates a pulse that stops and clears the multiplex processor. The status mode bit going on generates another pulse which starts the multiplex processor. Generation of these pulses in effect allows the multiplex processor to be controlled by the host computer in the same manner as the base processor (via communications coupler functions).

- Coupled master clear - consists of a Master Clear command generated by the base processor which clears both the base and multiplex processors. The master clear function of the multiplex processor operates conventionally in that a master clear is generated by depressing the MASTER CLEAR switch on the multiplex processor maintenance panel, by depressing the question mark (?) key on the operator's console, or through a power-on master clear.

MULTIPLEX SUBSYSTEM

The multiplex subsystem contains the hardware, firmware, and software elements which provide the data and control paths for information flow between communications lines and user program software in the processor. See figures 2-1 and 2-2. Many of the line/protocol-dependent functions that were performed by fixed hardware in previous systems are implemented in common alterable firmware and software, allowing reduced development and recurring hardware costs for each line and protocol that is added to the communications system.

The purpose of the multiplex subsystem is to gather data from the various terminal devices and deliver this data to the processor (multiplex processor in the dual processor NPU), and to receive data from the processor for distribution to the appropriate terminal devices. In performing this data movement (e.g., from terminal to processor) all unique signal characteristics not requiring knowledge of the meaning of the bits are stripped out. These signal characteristics include: data rate, synchronous or asynchronous transmission, parity or no parity, number of bits (5, 6, 7 or 8), and serial or parallel form. All information arrives in memory in bit-parallel, right-justified form. Control codes, error control, and other housekeeping bits are removed.

Similarly, information from the processor is formatted and conditioned sufficiently to permit transmission to any modem or local peripheral. Differences that require knowing the meaning of the bit, such as protocol and code type, are handled by software.

The heart of the multiplex subsystem comprises two high-speed digital data circuits called multiplex loops, as shown in figure 2-1. The input loop handles data enroute to the processor from the terminal devices. The output loop handles traffic from the processor to the terminals. The loops also resolve simultaneous input requests.

The multiplex subsystem is demand-driven. When action is required by a terminal, a demand service request is generated and transmitted to the processor via the input loop.

The primary functional elements of the mechanism are the communications line adapters that accomplish character assembly/disassembly and terminal interface; the input loop which transports data demands, data, and status from the communications line adapters to the processor; the output loop which transports data and control from the processor to the communications line adapters; a multiplex loop interface adapter which controls the movement of data demands, data, and supervision between the loops and the processor; and the loop mulitplexers which provide access to the loops by the communications line adapters.

Multiplex action on the input loop occurs when the multiplex loop interface adapter issues a loop-end control signal. As the signal propagates around the loop, each loop multiplexer in turn has an opportunity to put data and/or supervision from the communications line adapters on the loop. Loop multiplexers holding information from the communications line adapters monitor the loop for the loop end signal, place the information on the loop, and replace the loop end signal. The multiplex loop interface adapter transfers the information received on the input loop to buffer storage for processing by the processor.

The multiplex loop interface adapter takes addressed information from buffer storage under direction of the processor and transmits this information on the output loop. The loop multiplexer receives the addressed information blocks and presents the address to all communications line adapters. The communications line adapter recognizes that its address has been selected and allows the information to be transferred from the output loop to the communications line adapter.

Multiplex Loop Interface Adapter

The multiplex loop interface adapter provides the interface between the processor and the multiplex loop.

The multiplex loop interface adapter performs limited editing and checking functions on the input data and deposits the data directly into a large circular buffer in main memory via the direct memory access channel.

The multiplex loop interface adapter receives output data demand signals from the communications line adapters and forwards these to the processor as high-priority interrupts. The multiplex loop interface adapter then receives output data from the processor via the internal data channel and distributes the data to the loop multiplexers. The multiplex loop interface adapter provides first-in, first-out buffering for input data, output data demand signals, and output data.

Loop Multiplexer

The loop multiplexer is the interface between the communications line adapters and the input and output loops. The loop multiplexer accepts data in parallel from the preceding loop multiplexer or from the multiplex loop interface adapter. When not synchronized to the data stream, the loop multiplexer passes all information received to the next loop multiplexer or to the multiplex loop interface adapter. The clock signal is reconstructed to specification when passed sequentially to the next element in the system.

INPUT SECTION

The input section of the loop multiplexer receives contiguous loop cells from the preceding loop multiplexer or from the multiplex loop interface adapter. When not synchronized to the data stream, the loop multiplexer passes all information received to the next loop multiplexer or to the multiplex loop interface adapter. The clock signal is reconstructed to specification when passed sequentially to the next element in the system.

The loop multiplexer utilizes the unique combination of a loop end cell and a subsequent empty cell to acquire synchronization. When synchronized, the loop multiplexer accepts data from each communications line adapter that has information for the system; the loop multiplexer then outputs this information to the input loop, in a line frame. At each frame boundary, the loop multiplexer examines the input loop for an empty cell. If the empty cell is detected, the loop multiplexer places the next communications line adapter frame on the loop. If an empty cell is not detected, the loop multiplexer drops synchronization.

All communications line adapters plug into a common bus and control-line structure. Each communications line adapter has a unique address and may be inserted into any communications line adapter slot. Each communications line adapter can inform the loop multiplexer, via one of 32 lines, that it has information for the system. The loop multiplexer can inform each communications line adapter, via one of 32 control lines, that it can place information on the communications line adapter bus in response to strobe signals. When a communications line adapter has been selected, the loop multiplexer will accept information from that communications line adapter until either an information-end signal from that communications line adapter is received, or 16 cells have been processed. When one of the above occurs, the loop multiplexer deselects that communications line adapter, and selects another communications line adapter. When all communications line adapters requesting access to the loop have been serviced, the loop multiplexer drops synchronization.

OUTPUT SECTION

The output section of the loop multiplexer receives contiguous loop cells from the preceding loop multiplexer or from the multiplex loop interface adapter. The output section always passes all loop cells received to the next loop multiplexer or to the multiplex loop interface adapter. The clock signal is reconstructed to specification when passed sequentially to the next element in the system. The loop multiplexer uses empty cells to synchronize itself to the data stream. When synchronized, the loop multiplexer looks for frame boundaries and passes loop cells within the frame to the communications line adapters. The loop multiplexer checks the cyclic redundancy checksum character received with each frame.

When the loop multiplexer has accepted the address of a frame, it presents this address to all communications line adapters. The communications line adapter with the corresponding address then connects itself to the output bus. The selected communications line adapter accepts the information directed to it until disconnected by the loop multiplexer. If a cyclic redundancy checksum error is detected by the loop multiplexer, it is reported to the communications line adapter prior to termination.

COMMUNICATIONS LINE ADAPTER

The function of the communications line adapter is to provide the interface between the loop multiplexer and a terminal with or without a modem. On input, the communications line adapter serves as a serial-to-parallel converter assembling the serial data at the signaling rate of the terminal, and transferring the data to the loop multiplexer in an 8-bit byte. On output, the communications line adapter functions as a parallel-to-serial converter receiving the data characters in bytes of 8 bits from the loop multiplexer and outputting them serially at the signaling rate of the terminal.

In order to minimize the number of communications line adapter types required, the communications line adapter hardware is designed to accommodate most circuit characteristics under software/firmware control. Each circuit's individual characteristics are handled, for the most part, by associated software which detects the specifics and sets parameter registers within a particular communications line adapter to handle them. Software-selected elements include: circuit speed, code set, and mode of operation. For synchronous circuits these elements also contain the synchronization pattern(s), and for asynchronous circuits they contain different input and output speeds, parity detect and generation, and number of stop bits per character.

The only remaining circuit variables are whether the circuit is synchronous or asynchronous, and if synchronous, the signal level standard to which the circuit interfaces. The terms synchronous and asynchronous refer to the way in which the data is transmitted: synchronously with a clock, or without reference to a clock (but at a fixed rate and with synchronization elements appended).

A number of types of communications line adapters, which have differing functional and/or electrical interface characteristics, are available. Refer to the Hardware Maintenance manuals for the functional descriptions of the specific communications line adapter types selected. See preface.

TAPE CASSETTE

The tape cassette controller serves as the access path for all communications between the tape drive and the processor. The controller provides input to, and receives output from, the transport at a rate of 750 characters per second (8-bit characters). This yields a data transfer rate of 6 kHz which is compatible with the tape drive.

The cassette controller interfaces to the internal I/O, internal data channel of the processor. The controller may be status- or interrupt-driven at the macro level and data transfers may also take place in the auto-data transfer mode at the micro level.

The cassette controller deadstart interface provides the means for the tape cassette transport to begin reading data in response to an operator-initiated command at the maintenance panel or the communications console. Only the read function is performed in the deadstart mode of operation. The data is ASCII, asynchronous, binary, serial-by-bit, at the 9600 baud rate.

Signal transfer between the controller and the tape drive is provided by a cable which connects the backpanel directly behind the controller card to the transport.

The tape cassette drive can be used as a program-load or data-capture device. The tape drive is capable of both reading and writing on Philips-type, magnetic tape, digital data cassettes. The tape drive contains servomotors, servoamplifiers, read-write circuitry, read-after-write head, and the necessary control circuits for processing data and controlling the tape.

SYSTEM AUTOSTART MODULE

The system autostart module monitors the program status and reloads the deadstart program tape any time the interval between the received programmed reset pulses exceed 3 seconds. For further information refer to the system autostart module manual (see preface).

COMMUNICATIONS COUPLER

The primary purpose of the communications coupler is to provide a link between the host computer and the processor. The coupler provides the means for direct memory transfers between the host peripheral processing unit (PPU) and the processor at a 1-megahertz character rate. The coupler is controlled by both host and processor software commands. The coupler also enables the host to start, clear, and stop the processor.

As shown in figure 2-5, the communications coupler has three interfaces: the host PPU data channel, the NPU internal data channel, and the NPU direct memory access channel. Information can be transferred between any two interfaces, and between any interface and various data and address registers within the coupler.

Communications between processor and host is through two unidirectional, software-controlled, single-word register transfers. Although their software-controlled nature makes them available for any purpose (the coupler is basically a transparent device), standard usage is to send work requests, status requests, etc., from the host via the orderword register. The host fills this register; when full, the register interrupts the processor, which then reads the register and takes action as directed by the resident software. Information from the processor to the host uses the processor status register (another free-form register); when full, the register sets a bit in the coupler status word (which the host reads periodically). The host then reads the processor status word.

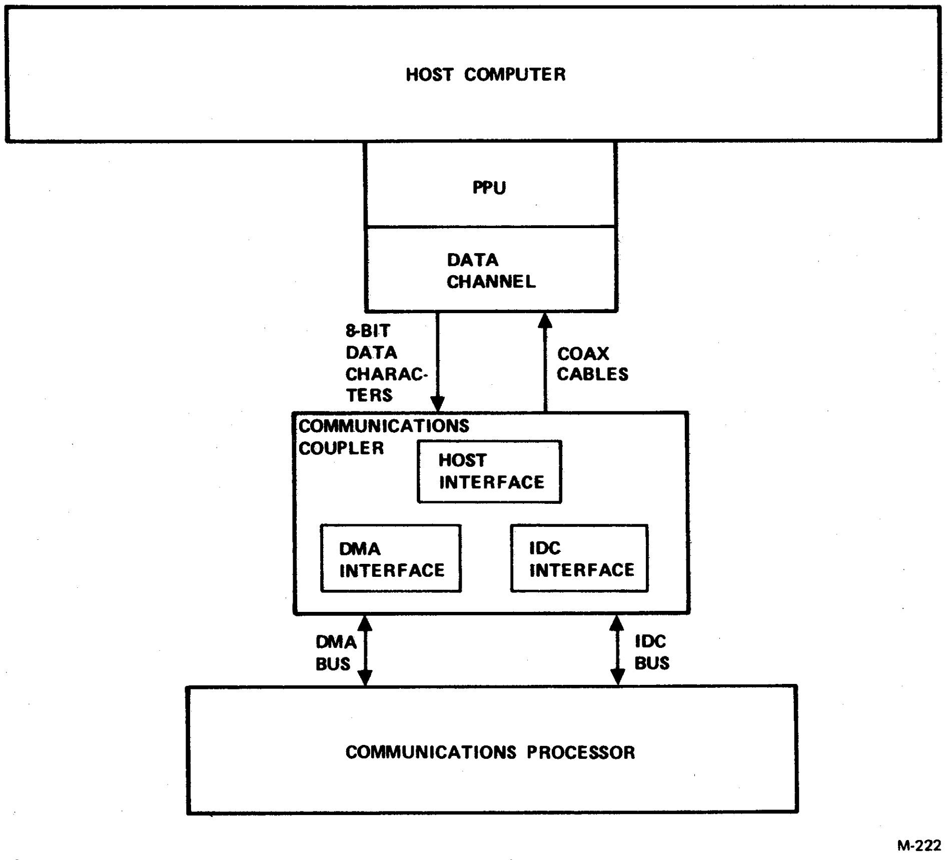


Figure 2-5. Communications Coupler Interfaces Block Diagram

OPERATING INSTRUCTIONS 3

This section describes the use of the function control register in internal communication, the control and indicator functions, and the start-up and operating procedures.

FUNCTION CONTROL REGISTER

The function control register is the basic means of communication between the microprocessor and the maintenance panel and communications console. It replaces the traditional front panel switches with a series of software-controlled switches and indicators capable of being set or read locally or remotely.

The communications console and maintenance panel use the function control register to provide the following basic functions:

- Master clear
- Halt memory operations
- Start memory operations
- Step mode
- Run mode
- Display memory
- Load memory
- Display register
- Load register
- Stop, when condition met

The function control register stores and defines the functions that are to be performed by the microprocessor. The contents of the function control register can be displayed at the console or the maintenance panel in response to an operator's command. The contents can then be changed by operator commands from the console or maintenance panel. The maintenance panel operation is local and the console is remote, but both provide the same level of control.

The function control register is a 32-bit register which is divided into 8 groups of 4 bits each as shown in table 3-1. Bit 00 is the most significant bit, and bit 31 is the least significant bit.

The machine status hexadecimal digits 6 and 7 (bits 24 thru 31) are set by the processor and indicate machine status such as overflow, macro running, micro running, main storage parity error, and protect fault. The machine mode digits 2 thru 5 (bits 8 thru 23) of the function control register are used to set such conditions as selective stop/on/off, and step/run mode. The display digits 0 and 1 (bits 0 thru 7) determine which individual registers of two groups of registers (identified in table 3-2) can be displayed or modified.

TABLE 3-1. FUNCTION CONTROL REGISTER BIT AND DIGIT FUNCTIONS

Bit Number	Bit Name	Digit Number	Digit Function
31 30 29 28	Overflow Protected Instruction Protect Fault Parity Error	(LSB) 7	Machine Status
27 26 25 24	Interrupt System Active Auto Restart Enabled Micro Running Macro Running	6	
23 22 21 20	Undefined Undefined Enable Auto Display Enable Console Echo	5	Machine Mode
19 18 17 16	Enable Micromemory Write† Multilevel Indirect Address Mode Undefined Suppress Console Transmit	4	
15 14 13 12	} Macrobreakpoint Breakpoint Interrupt (Breakpoint Stop if Clear) Micro Breakpoint, Step, Go, Stop (Macro if Clear)	3	
11 10 09 08	Step Selective Stop Selective Skip Protect Switch	2	
07 06 05 04	} Display 1 (K Function)	1	Display
03 02 01 00	} Display 0 (L Function)	0 (MSB)	

†A hard-wired function, not controllable by the function control register.
Legend: LSB - Least significant bit
MSB - Most significant bit

TABLE 3-2. FUNCTION CONTROL REGISTER DISPLAY CODE DEFINITIONS

Code		Display 1 (K Function)	Display 0 (L Function)
0	0 0 0 0	FCR	F2 (addressed by N)
1	0 0 0 1	P†	N (MSDs)††
2	0 0 1 0	I	K (LSDs)††
3	0 0 1 1		X
4	0 1 0 0	A	Q
5	0 1 0 1	MIR	F
6	0 1 1 0	BP/P-MA	F1 (addressed by K, enabled by SM111)
7	0 1 1 1	BP/P-MA (Display only)	MEM
8	1 0 0 0	SM1	
9	1 0 0 1	M1	$\overline{\text{RTJ}}$
A	1 0 1 0	SM2	
B	1 0 1 1	M2	
C	1 1 0 0		MM
D	1 1 0 1	A*	
E	1 1 1 0	X*	
F	1 1 1 1	Q*	

† Used to address main memory; automatically incremented after each memory reference

†† Combined contents of these two registers are used to address micromemory. K register is automatically incremented after each memory reference. N register does not automatically increment.

NOTES

Bits 14_{16} and 15_{16} of the function control register (enable console echo and enable auto display) are mutually exclusive; that is, the operator may select one or the other, but not both simultaneously.

Unassigned display codes are undefined.

Selection of BP or P-MA causes both BP and P-MA to be displayed. BP consists of the leftmost 16 bits and P-MA consists of the rightmost 16 bits. BP can be modified only if BP is selected; P-MA cannot be modified in either case.

Selection of N or K causes both N and K to be displayed. N is the left 8 bits and K is the right 8 bits. However, when N is selected, only the N register can be modified; when K is selected, only the K register can be modified.

The machine status and machine mode bits are described in table 3-3. Note that in the protected instruction (bit 30), a reply response is enabled for all commands if the program protect line indicates the instruction is protected. If the program protect line indicates that the instruction is unprotected, then for each function and data transfer command, a reject is generated; however, a status command is never rejected. A typical function control register setting is 442008C0 (A/Q reference) or 712008C0 (main memory reference).

CONTROLS AND INDICATORS

Controls and indicators for monitoring and controlling the NPU functions are used extensively throughout the NPU. Many are located on circuit cards and some monitor operation of external devices such as the tape cassette drive.

Switches on the tape cassette drive monitor whether the lid is closed, which side of cassette (A or B) is up, and the presence or absence of the write enable plug for whichever track is positioned under the read/write head.

Main power circuit breakers on the cabinet are used during starting and shutdown procedures in this section.

Multisection microswitches located on NPU circuit cards are preset at the factory for normal operation. Replacement cards may require switch setting changes.

The maintenance panel and the communications console contain the majority of the controls and indicators used to monitor and operate the NPU.

MAINTENANCE PANEL

The maintenance panel controls and indicators are shown in figure 3-1 and described in tables 3-4 and 3-5.

COMMUNICATIONS CONSOLE

The communications console is the operator's primary point of interface with the processor. Either a cathode ray tube (CRT) conversational display or a teletypewriter (TTY) device is used as the console. Figure 3-2 presents the controls and indicators (including keyboard) for a typical TTY, and figure 3-3 illustrates a typical CRT conversational display. For a complete description of the controls and indicators for these units, refer to the appropriate manual for the communications console selected for the system.

I/O TTY INTERFACE

The I/O TTY interface, illustrated in figure 3-4, contains a 4-element switch assembly. Each element is a 2-position switch. The switches are used to select the desired baud rate as shown in table 3-6. They are set at the time the NPU is installed and are not reset under operating conditions.

CYCLIC ENCODER

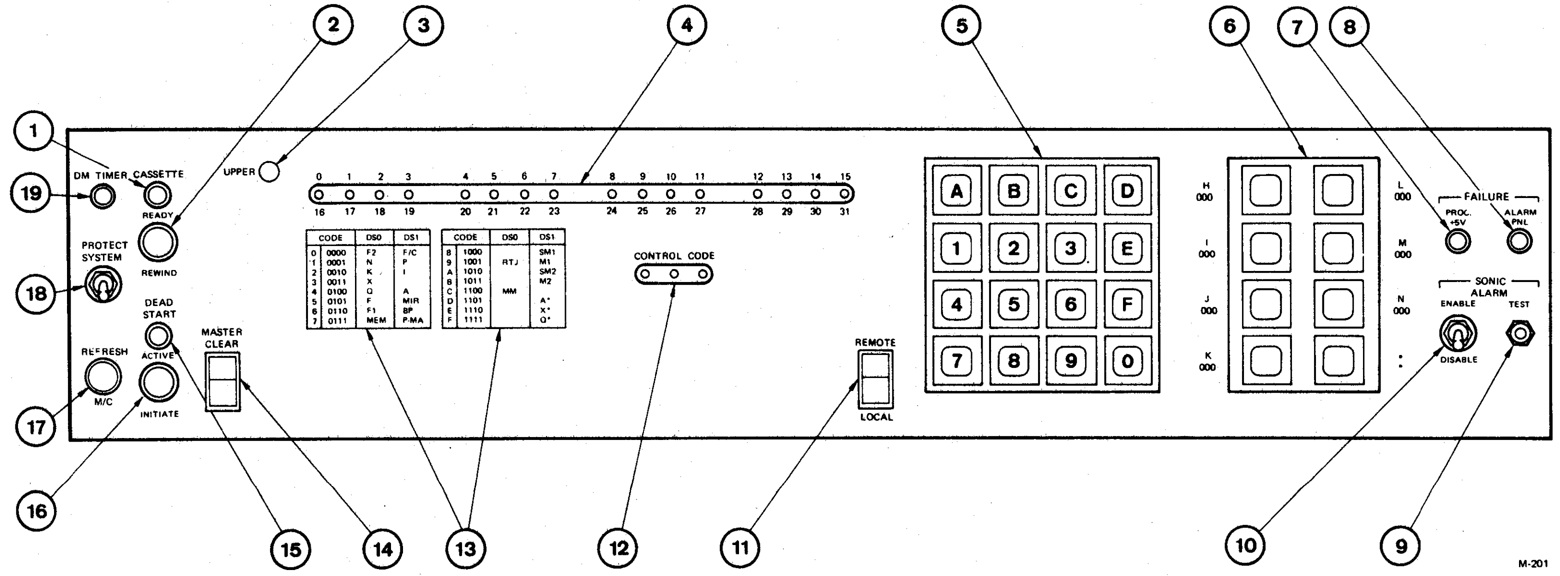
The cyclic encoder, illustrated in figure 3-5, contains a single, LED power indicator which indicates the presence of +5 volts when lit.

TABLE 3-3. FUNCTION CONTROL REGISTER BIT DESCRIPTIONS

Bit Number	Bit Description
31	Overflow. Set when an overflow occurs during execution of an arithmetic and logical unit.
30	Protected Instruction. Set when processor is executing a protected instruction (protect bit is set).
29	Protect Fault. Set when attempt is made to access a protected area in main memory from a nonprotected instruction (and protect system is enabled).
28	Parity Error. Set when main memory interface logic detects a parity error while reading data from main memory.
27	Interrupt System Active. Set when interrupt system is available for use (regardless of contents of interrupt mask register).
26	Auto-Restart Enabled. Set when automatic program restart is desired after a power failure.
25	Micro Running. Set when emulator function (microprogram) is operating in the microprocessor. When the emulator is halted, bit is not set.
24	Macro Running. Set when a macroprogram is being executed in microprocessor (bit 25 must also be set or macroprograms cannot be executed).
21	Enable Auto Display. Set to allow viewing of a given register (selected by digits 0 and 1) while the processor is running. Auto display mode is used with the maintenance panel to verify that certain registers are changing; it is not normally used with the console.
20	Enable Console Echo. Set to allow viewing of data being entered at communications console (as well as machine responses); usually used with teletypewriter input to provide a record of all messages; sometimes used with the CRT console and almost never with the maintenance panel itself.
18	Multilevel Indirect Address Mode. Set when a maximum of 32K words can be accessed in main memory and the most significant bit denotes indirect addressing; when not set, 65K words can be accessed. The 32K mode corresponds to CDC-1704 operation; the 65K mode corresponds to 1714, et al, operation and is the commonly used mode.
16	Suppress Console Transmit. Set when operational mode transmission from console to microprocessor is inhibited; when set, maintenance mode transmission by the console is permitted.

TABLE 3-3. FUNCTION CONTROL REGISTER BIT DESCRIPTIONS (Contd)

Bit Number	Bit Description
15, 14	Breakpoint. Used in combination to select three types of breakpoint: Bit 14 / Bit 15 0 0 Breakpoint not selected 1 1 Instruction reference breakpoint 1 0 Store operand breakpoint 1 1 All references breakpoint If bit 12 is set, these are all microbreakpoints or macrobreakpoints.
13	Breakpoint Interrupt. When set and a breakpoint is reached, program does not halt immediately, but halts upon reaching an interrupt instruction.
12	Microbreakpoint, Step, Go, Stop. When set, the processor is operating in the microbreakpoint, step, go, stop mode only; when not set, processor is operating in macromode for these functions.
11	Step. When set, program operates in a stepping mode; operator triggers the execution of each instruction individually.
10	Selective Stop. When set, halt instructions cause instruction execution to cease; when not set, halt instructions have no effect on program execution.
9	Selective Skip. When set, program executions for skip switch on are executed; when not set, program executions for skip switch not on are executed.
8	Protect Switch. When set, it is possible to write into protected areas in main memory, if protect system is not enabled, this switch has no effect.
NOTE:	All of these actions occur because of controlware operation and not because of any direct hardware control.



CODE		DS0	DS1
0	0000	F2	F/C
1	0001	N	P
2	0010	K	I
3	0011	X	
4	0100	Q	A
5	0101	F	MIR
6	0110	F1	BP
7	0111	MEM	P-MA

CODE		DS0	DS1
8	1000		SM1
9	1001	RTJ	M1
A	1010		SM2
B	1011		M2
C	1100	MM	
D	1101		A*
E	1110		X*
F	1111		Q*

Figure 3-1. Maintenance Panel Controls and Indicators

TABLE 3-4. MAINTENANCE PANEL CONTROLS AND INDICATORS

Key No. (Fig. 3-1)	Panel Name	Function
1	CASSETTE REWIND pushbutton switch	Actuates rewind mode of operation in drive.
2	CASSETTE READY indicator light	Lights when tape cassette drive is ready (cassette installed, door closed, transport not rewinding, and no cyclic redundancy checksum error). A cassette must be ready to initiate deadstart.
3	UPPER indicator light	Operates with the data display indicators. Lights when data display represents contents of the upper bits in a register. Primary use is display of function control register.
4	Data Display indicator lights (16)	Display contents of a selected register on panel; lights when bits 0 thru 15 (for 32-bit register) are being displayed; when not lit, bits 16 thru 31 are displayed.
5	Data Entry push-button switches (16)	Enter data in hexadecimal form.
6	Control Character pushbutton switches (8)	Enter control characters H, I, J, K, L, M, N, and the colon (:)
7	FAILURE PROC +5V indicator	Not used
8	FAILURE ALARM PNL indicator	Not used
9	SONIC ALARM TEST pushbutton switch	Not used
10	SONIC ALARM ENABLE/DISABLE toggle switch	Not used
11	REMOTE/LOCAL rocker switch	Selects either maintenance panel or communications console as the point of control for NPU. When set to LOCAL, panel is enabled; when switch is set to REMOTE, console is enabled.

TABLE 3-4. MAINTENANCE PANEL CONTROLS AND INDICATORS (Contd)

Key No. (Fig. 3-1)	Panel Name	Function
12	CONTROL CODE indicator light (3 LEDs)	Indicate last character entered via the control character switches in the control codes listed; a 1 denotes a lighted LED. Once entered, a control code remains set until replaced by another controlor code or master clear.
13	Function Control Definition Table decal	Provides operator assistance information of function control register and control character.
14	MASTER CLEAR rocker switch	Clears memory in processor and peripheral controller.
15	DEAD START ACTIVE indicator light	Indicates NPU ready to receive data from a card reader or tape cassette transport, and deadstart sequence in process.
16	DEAD START INITIATE pushbutton switch	Actuates data reading by a card reader or tape cassette transport; selection of device determined by the ready state of the peripherals. Only one deadstart peripheral can be ready when using this switch.
17	REFRESH MC pushbutton switch	Not used
18	PROTECT SYSTEM toggle switch	Normally in down position (system not protected); in up position, designated portions of memory are protected from overwrite during certain diagnostic routines.
19	DM TIMER indicator	Not used

TABLE 3-5. CONTROL CHARACTER CODE/FUNCTIONS

Character	Display	Function
H (Master Cleared State)	000	When used alone (followed by a colon terminator), halts program execution; affects either macroprogram or microprogram execution, depending on setting of bit 12 of function control register. When preceded by a 2-digit number (followed by a colon terminator), resets corresponding bit of function control register.
I	001	When used alone (followed by a colon terminator), initiates program execution; affects either macroprogram or microprogram execution, depending on the setting of bit 12 of function control register. When preceded by a 2-digit number (followed by a colon terminator), sets corresponding bit of function control register.
J	010	When used alone (followed by a colon terminator), changes state of UPPER indicator, permitting upper and lower portions of function control register, for example, to be displayed. When preceded by a 2-digit number (followed by a colon terminator), causes 4 bits of function control register referenced by first digit (0 thru 5) to be replaced by value of second digit.
K	011	When used alone (followed by a colon terminator), displays contents of register specified by Display 1 portion of function control register. When preceded by a 4- or 8-hex digit number (followed by a colon terminator), causes contents of register by the function control register Display 1 character to be replaced by the specified digits.
L	100	Operationally the same as K function except it is associated with Display 0.

TABLE 3-5. CONTROL CHARACTER CODE/FUNCTIONS (Contd)

Character	Display	Function
M	101	Not used.
N	110	Not used.
ERROR	111	Can be used as indicator that a program failed to load properly.
:	-	Terminates all entries (when using console, three terminating characters can be used: colon, G, or @). The @ character also switches control from panel mode to I/O mode.

NOTE

When main memory is displayed or entered, the register selected in Display 1 is the main memory address. The Display 1 selection must be the P register, which is incremented by 1 after the display.

When micromemory is displayed or entered, K register is the least significant 8 bits of the address, and N register provides for remaining bits. K register is incremented by 1 after the display.

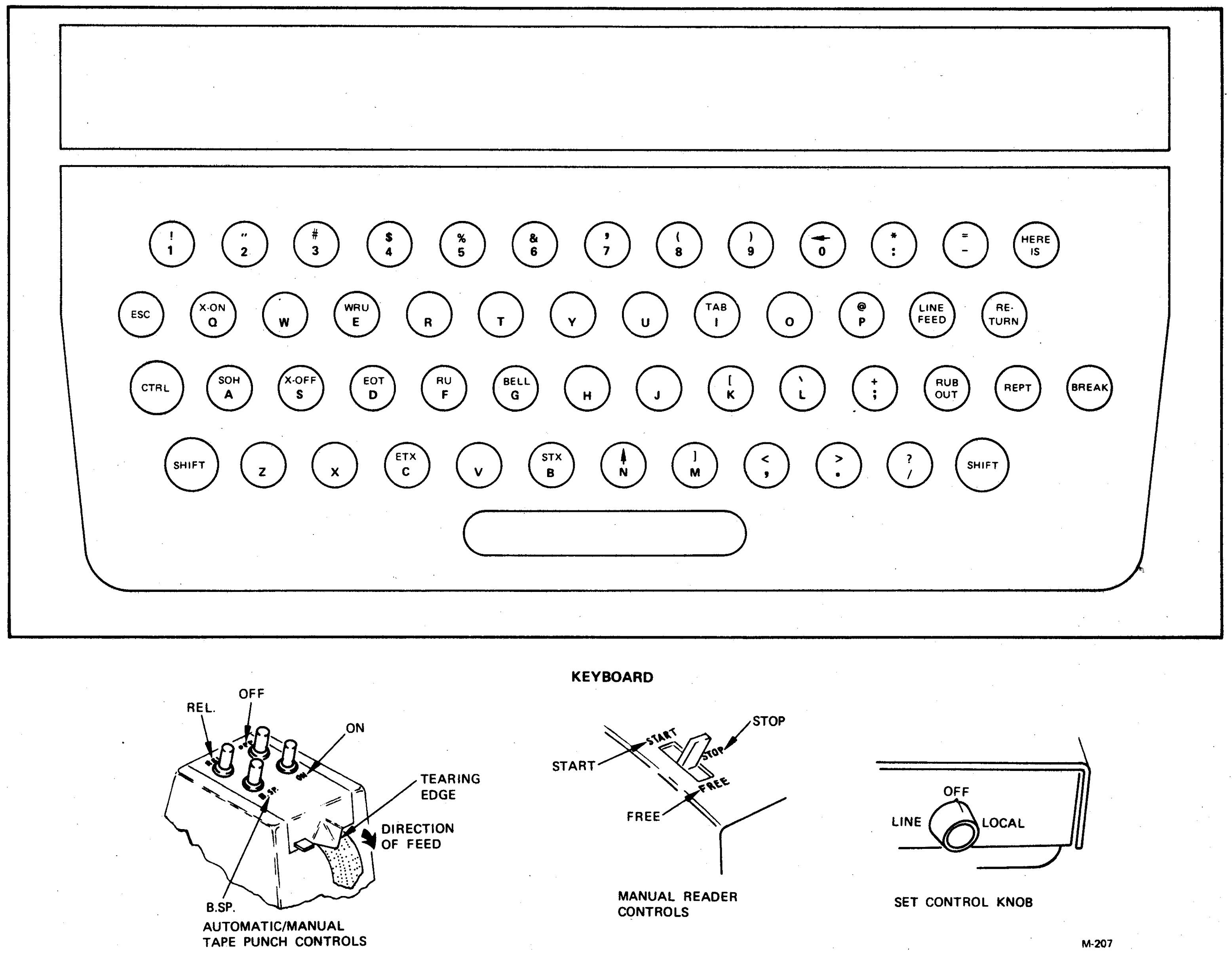


Figure 3-2. Typical Teletypewriter Keyboard and Controls

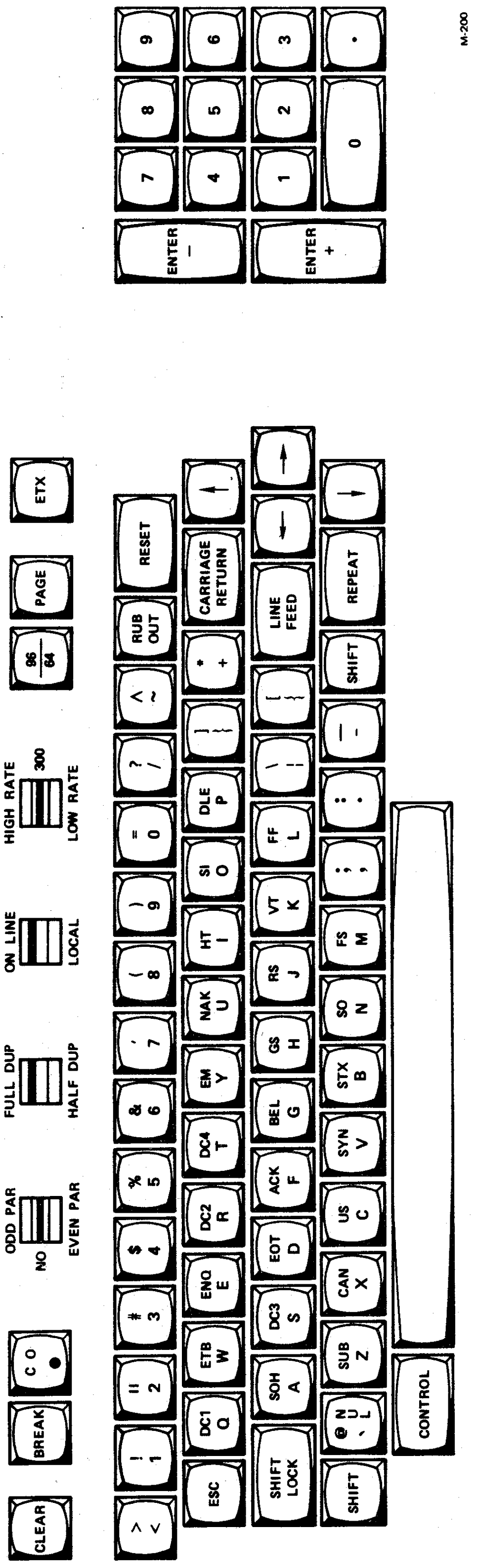


Figure 3-3. Typical Cathode Ray Tube 752 Conversational Display Keyboard and Controls

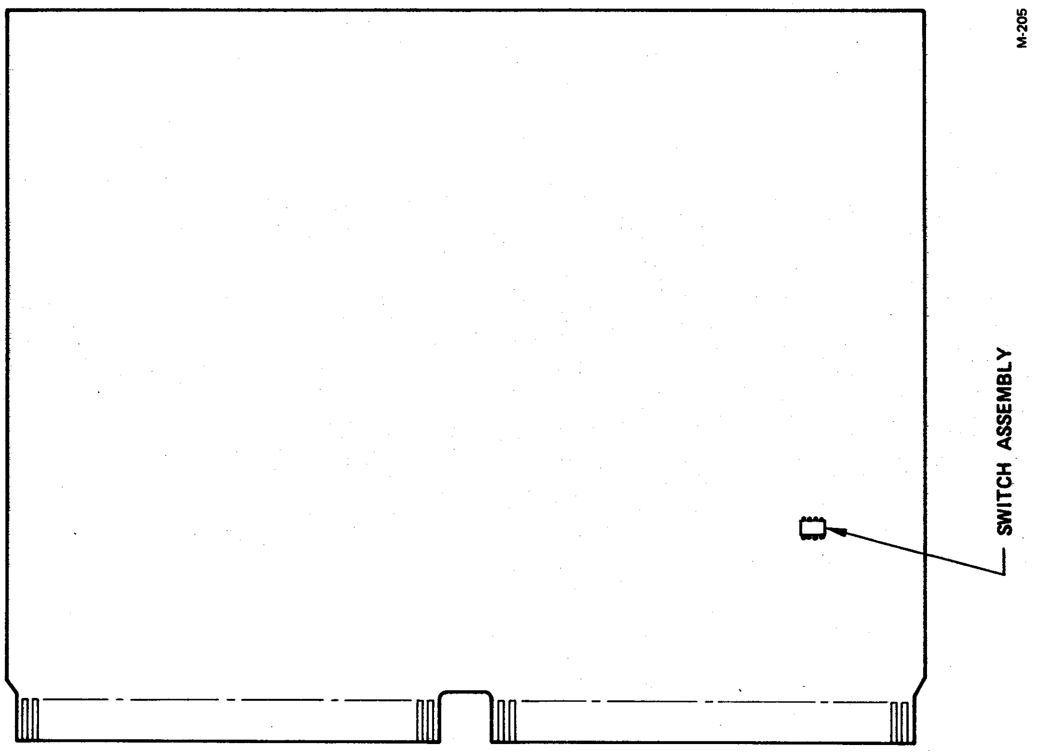


Figure 3-4. I/O TTY Interface Controls

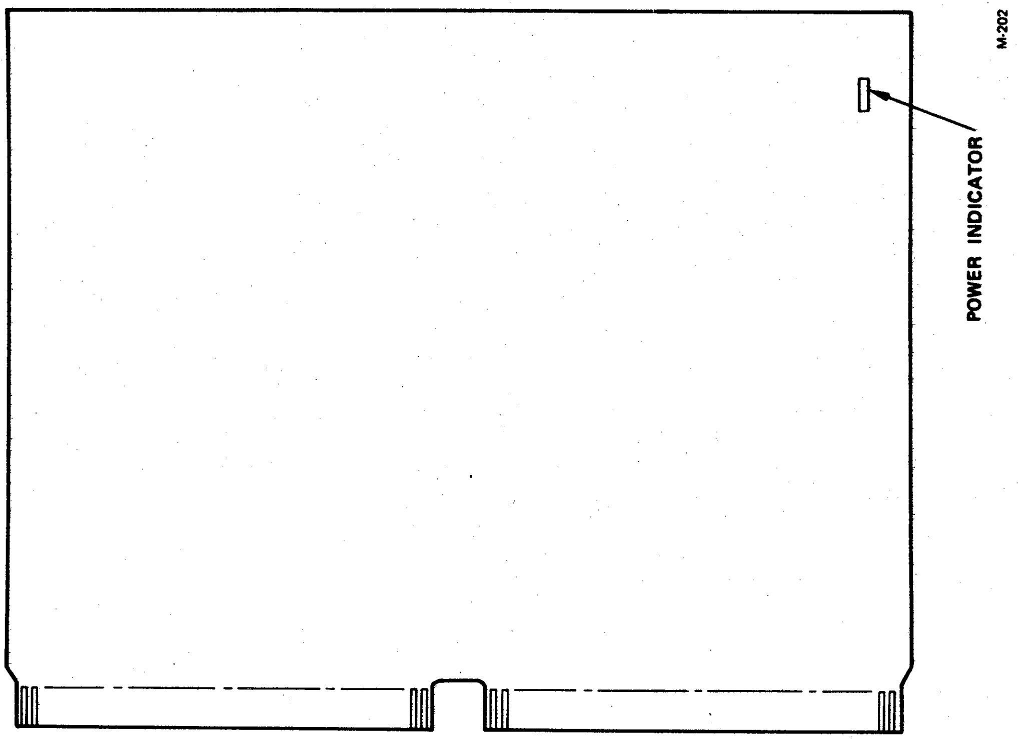


Figure 3-5. Cyclic Encoder Indicator

TABLE 3-6. SWITCH SETTINGS FOR I/O TTY INTERFACE

Deadstart †		Program Select ††		Baud Rate
1	2	3	4	
On	On	On	On	110
Off	Off	On	Off	300
Off	On	Off	On	1200
Off	Off	Off	Off	9600
†Input Device: Tape Cassette, 9600 ††I/O Console: TTY, 110; CRT, 110 to 9600				

LOOP MULTIPLEXER

The loop multiplexer controls and indicators, located on the circuit card handle, include a power switch and four LED indicators. See figure 3-6 and table 3-7. An identification label holder allows each user to identify the circuits connected to the communications line adapters by insertion of small labels.

MULTIPLEX LOOP INTERFACE ADAPTER

All three of the circuit cards which comprise the multiplex loop interface adapter contain LED indicators. The processor interface card and the input loop interface card each contain a single LED indicator that indicates the presence of +5 volts electrical power when lit. The output loop interface, illustrated in figure 3-7, contains the following five LED indicators (the numbered items below correspond to key numbers in figure 3-7):

1. Multiplex Loop Interface Adapter Running. When lit, this LED indicates that the multiplex loop interface adapter is not reset.

2. Output On. When lit, this LED indicates the presence of data and clock signals on the output loop.

3. Input On. When lit, this LED indicates the presence of data and clock signals on the input loop.

4. Input Busy. When lit, this LED indicates either the presence of loop batches on the input loop or the absence of a valid restart loop end following an error.

5. Power. When lit, this LED indicates the presence of +5 volts electrical power.

COMMUNICATIONS LINE ADAPTER

Each synchronous and asynchronous communications line adapter contains four LED indicators, two thumbwheel address switches, and an enable/disable toggle switch. Two independent communications line adapters are mounted on one circuit card; thus two sets of controls and indicators are mounted on the card handle, as shown in figure 3-8.

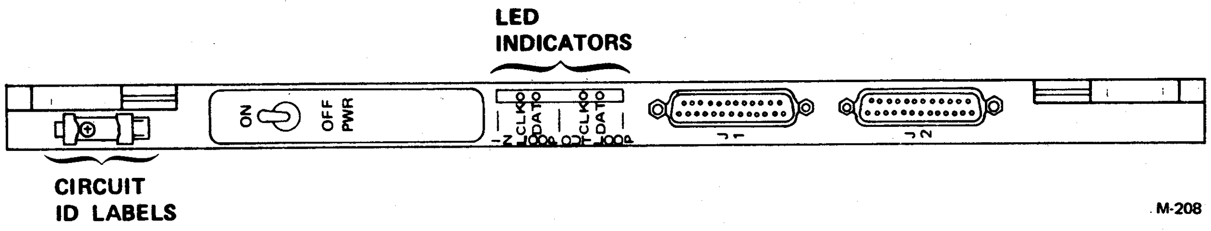


Figure 3-6. Loop Multiplexer Controls and Indicators

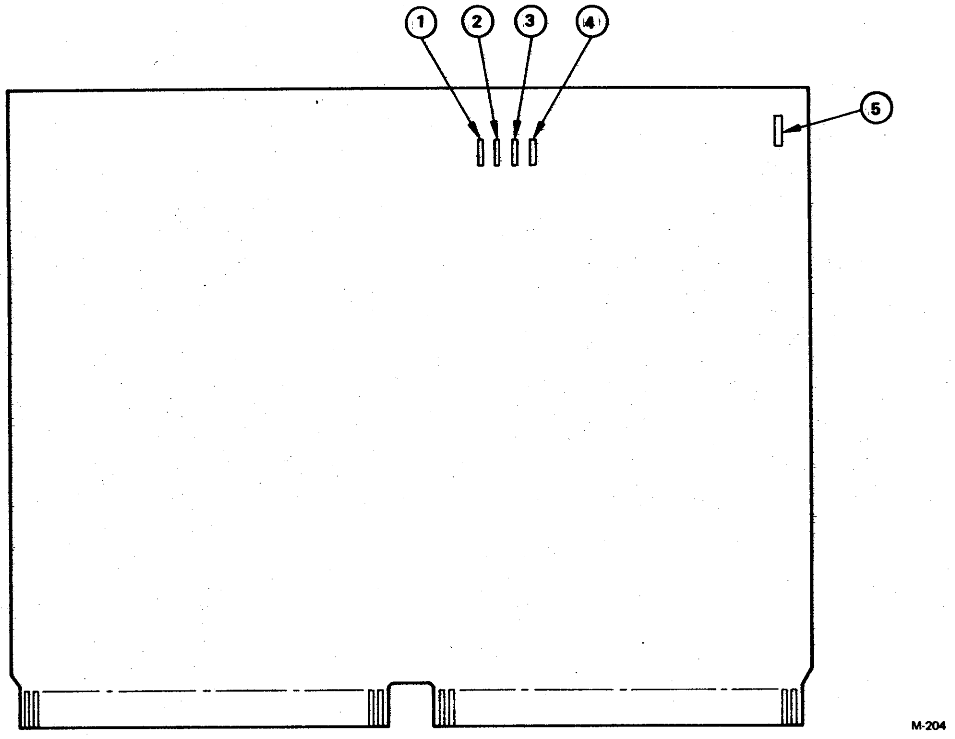


Figure 3-7. Multiplex Loop Interface Adapter Output
Loop Interface Indicators

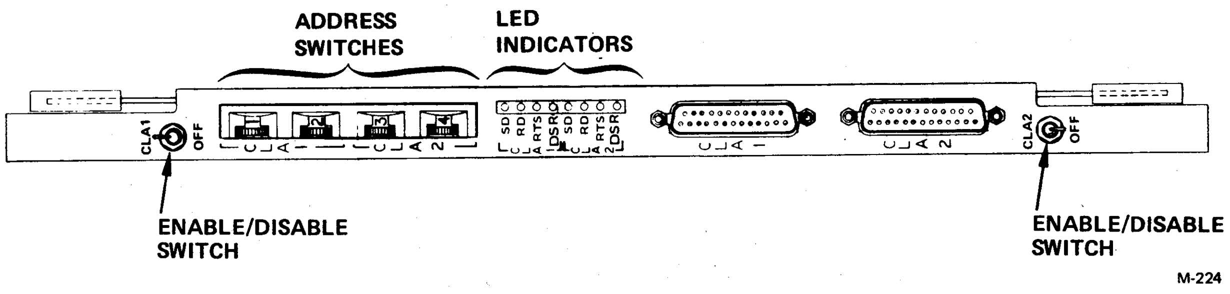


Figure 3-8. Communications Line Adapter Controls and Indicators

TABLE 3-7. LOOP MULTIPLEXER SWITCH AND INDICATORS

Panel Marking	Function
PWR ON/OFF toggle switch	Controls electrical power to the loop multiplexer.
IN LOOP CLK indicator light	When lit, indicates input loop clock signal is receiving bit clocking from multiplex loop interface adapter or from preceding loop multiplexer.
IN LOOP DATA indicator light	When lit, indicates input loop is receiving data cells from multiplex loop interface adapter or from preceding loop multiplexer.
OUT LOOP CLK indicator light	When lit, indicates output loop clock signal is receiving bit clocking from multiplex loop interface adapter or from preceding loop multiplexer.
OUT LOOP DATA indicator loop	When lit, indicates output loop is receiving data cells from multiplex loop interface adapter or from preceding loop multiplexer.

NOTE

The synchronous data link control communications line adapter is mounted one per circuit card. Therefore, only one set of controls and indicators appear on the card handle of this equipment.

Each address switch has 16 positions (hexadecimal); thus each pair of switches (CLA1 and CLA2) can be set to 256 different addresses. Each communications line adapter in the NPU system must be set to a unique address by means of the switches.

The names and functions of all communications line adapter (CLA) switches and indicators are listed in table 3-8.

COMMUNICATIONS COUPLER

The CYBER interface (synchronizer) in the communications coupler, illustrated in figure 3-9, contains a clock adjustment, a clock indicator, an on-line switch, and a dual-inline package (DIP) switch.

Clock Adjustment

A potentiometer can be used to adjust the phase of the clock signal provided by the host computer. This control is adjusted during NPU installation and normally needs no readjustment.

TABLE 3-8. COMMUNICATIONS LINE ADAPTER SWITCHES AND INDICATORS

Name	Function
Address switches	Switch setting displayed in hexadecimal code (00 to FF). Top switch is most significant digit; bottom switch is least significant digit.
Modem indicator SD	Blinking, indicates CLA sending data to modem
Modem indicator RD	Blinking, indicates CLA receiving data from modem
Modem indicator RTS	Lighted, indicates request-to-send signal from CLA is active
Modem indicator DCD (synchronous CLA only)	Lighted, indicates data-carrier-detect signal from modem is active
CLA1/OFF toggle switch	Enables or disables CLA ability to input to the system
Modem indicator DSR (asynchronous CLA only)	Lighted, indicates data-set-ready signal from modem is active

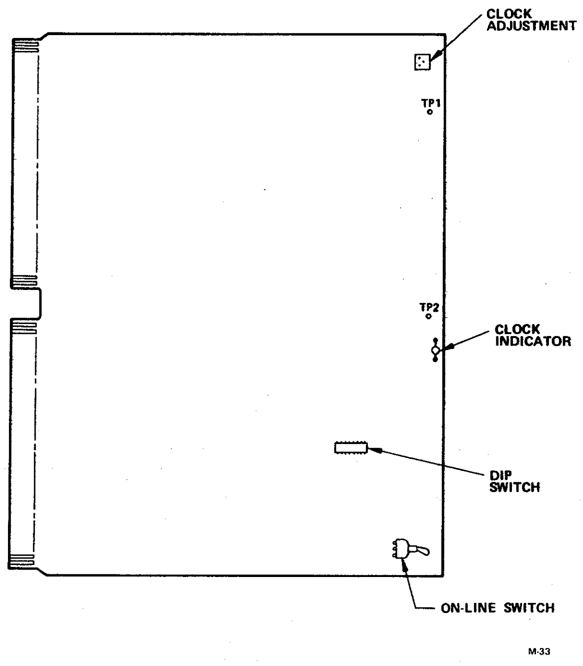


Figure 3-9. Communications Coupler CYBER Interface Controls and Indicators

Clock Indicator

This LED indicator goes on to indicate that the on-line clock signal from the host computer is available. The LED continues to indicate the availability of the on-line clock, even if the switch is set to the off position. When lit, this LED also indicates the presence of +5 volts electrical power.

On-Line Switch

This 2-position toggle switch is used to connect the NPU to the enabled host data channel. When the switch is set to the up position, the NPU is connected to the host channel. When the switch is in the down position, the communication links with the host data channel are disabled. The switch is in the on position except when off-line diagnostics are being run.

DIP Switch

This is an 8-element switch assembly, each element is a 2-position rocker switch. Switches 1, 2, and 3 are used to select an equipment code (address) for the communications coupler; switches 4, 5, 6 and 7 are not used; and switch 8 is used for parity. Table 3-9 lists the functions for these switches. This switch assembly is set at the time the NPU is installed and is not reset under normal conditions.

MICROMEMORY

The 2K RAM micromemory card (figure 3-10) contains a 4-element switch assembly. Each element is a 2-position switch. The switch is used for micromemory page selection as shown in table 3-10.

TABLE 3-9. SWITCH SETTINGS FOR COMMUNICATIONS COUPLER SYNCHRONIZER

1	2	3	Equipment Code	8	Parity
On	On	On	0	On	Channel Parity Disabled (6000, CYBER 70)
Off	On	On	1		
On	Off	On	2		
Off	Off	On	3		
On	On	Off	4	Off	Channel Parity Enabled (CYBER 170)
Off	On	Off	5		
On	Off	Off	6		
Off	Off	Off	7		
The standard equipment code assignment is 7.					

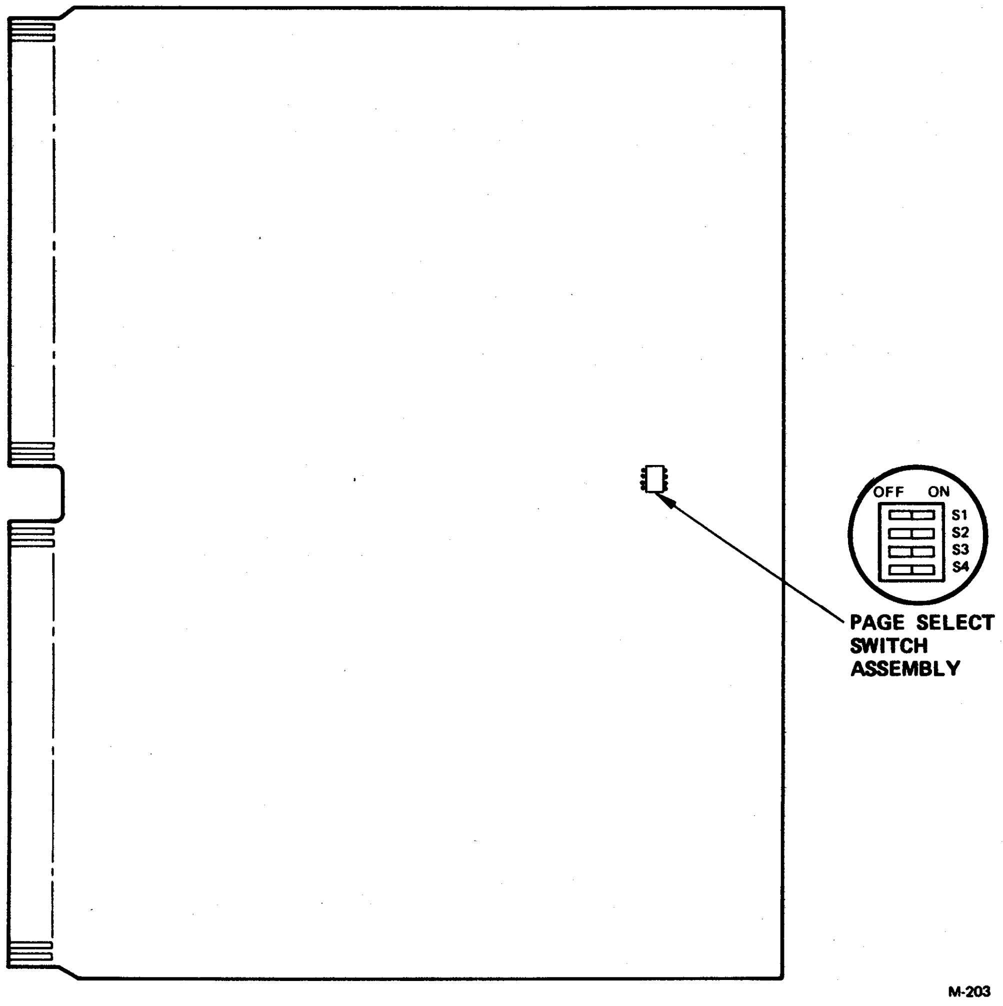


Figure 3-10. 2K RAM Micromemory Page Select Switches

TABLE 3-10. SWITCH SETTINGS FOR 2K RAM MICROMEMORY PAGE SELECT SWITCHES

Switch Number					Micromemory Pages Selected
(V1) (V2)	1 4 Top	2 3	3 2	4 1 Bottom	
	Off Off On On	Off On Off On	Off Off Off Off	N/A N/A N/A N/A	0-3 4-7 8-11 12-15
No other switch settings are used.					

The two configurations of the micromemory are identified as Version 1 (V1) and Version 2 (V2). For Version 1, the switches are numbered 1, 2, 3, 4 from top to bottom. For Version 2, the switches are number 4, 3, 2, 1 from top to bottom. (The terms top and bottom refer to the circuit card as normally installed.) The switches are set at the time the NPU is installed and are not reset under normal conditions.

The 8K RAM micromemory card (figure 3-11), contains two 4-element switch assemblies. For normal operation, all of the switches should be in the OFF position. In this mode all 8K of RAM can be accessed unconditionally. High 8K and low 8K can also be selected. See table 3-11.

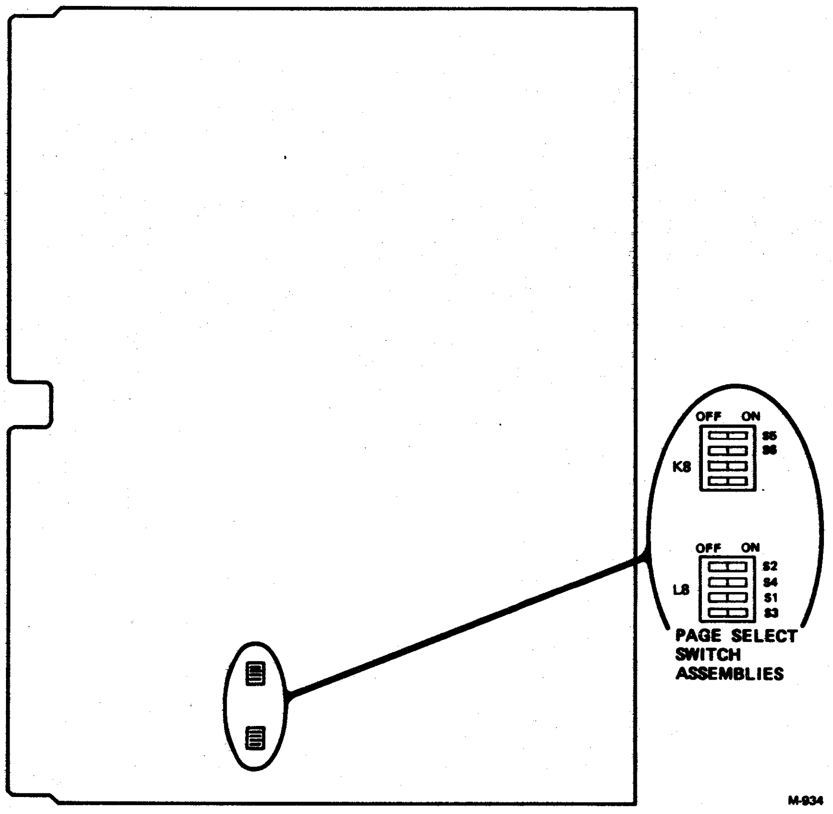


Figure 3-11. 8K RAM Micromemory Page Select Switches

TABLE 3-11. SWITCH SETTINGS FOR 8K RAM MICROMEMORY PAGE SELECT SWITCHES

Switch		Description
S1	S2	
Off	X	Unconditional 8K enable
On	Off	Active high 8K enable
On	On	Active low 8K enable

CASSETTE TAPE CONTROLLER

The cassette tape controller (figure 3-12) contains an equipment switch assembly and a control functions switch assembly.

Equipment Switch Assembly

This switch assembly consists of four segments; each segment is a 2-position switch. The switches are used to set the equipment code (address) for the cassette tape controller. The switches are set at the time the NPU is installed and are not reset during normal operating conditions. Standard equipment switch address is 7_{16}.

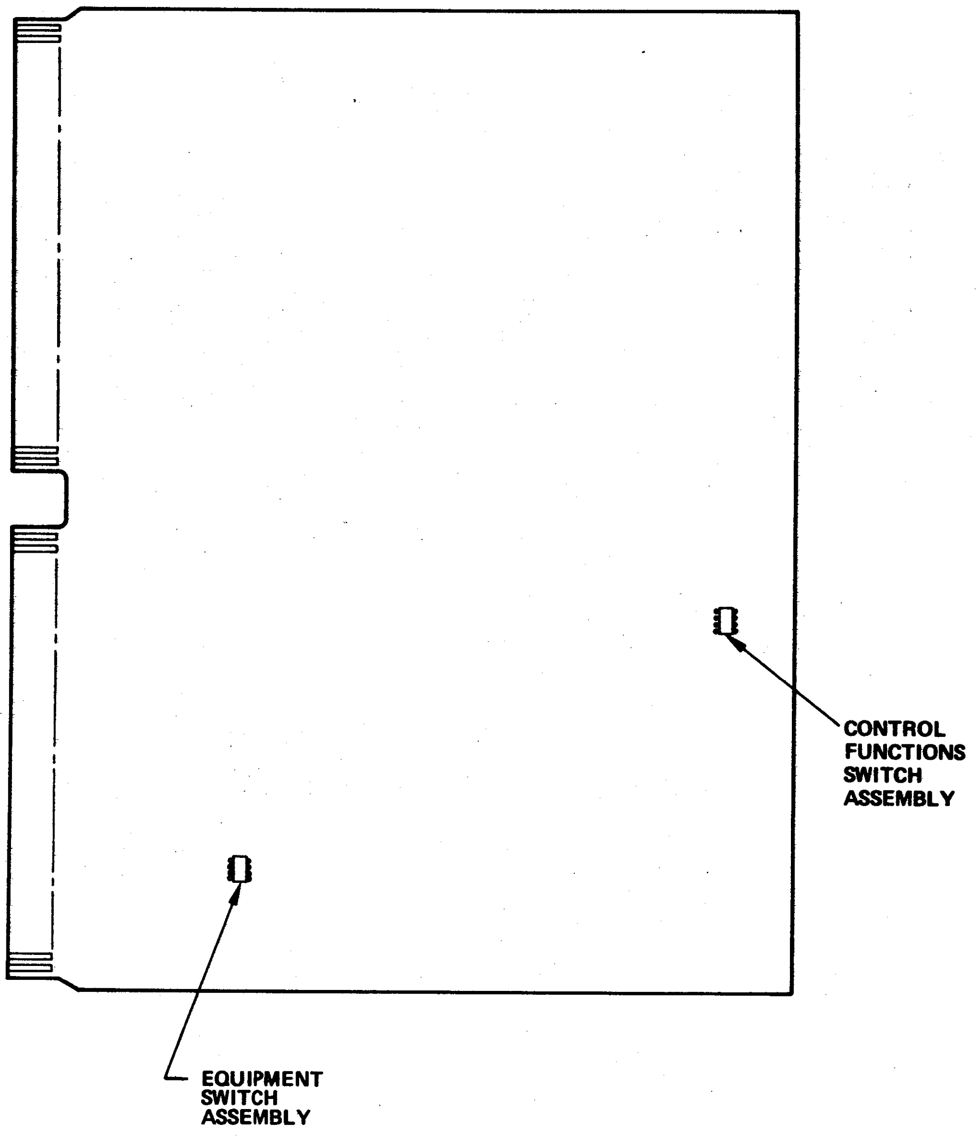


Figure 3-12. Cassette Tape Controller Controls

Control Functions Switch Assembly

This switch assembly consists of four segments: segments 1, 2, 3, and 4. Segments 2 and 4 are not used by the NPU. Each segment is a 2-position switch. The control functions provided by the switches are listed in table 3-12. The switches are set at the time the NPU is installed and are not reset during normal operating conditions.

AUTO-REWIND ENABLE SWITCH (Segment 1)

When this switch is on, the tape drive automatic-rewind feature is operative. Thus, each time a cassette is inserted and the lid is closed, the tape is automatically rewound to the beginning-of-tape hole. When this switch is off, the automatic-rewind feature is not operative. However, rewind function commands are always operative.

In the inhibited position, the switch facilitates editing operations and permits sequential loads of diagnostic overlays from more than one deadstart device.

PROTECT SWITCH (Segment 3)

When this switch is in unprotected mode (off position), the controller ignores the state of the program protect lines from the microprocessor and accepts all I/O instructions (function, status, and data transfer commands). An unprotected controller never rejects a function, status, or data transfer command.

When the switch is in protected mode (on position), the controller monitors the program protect line from the microprocessor. The controller's response to the I/O instruction is then determined by the state of the program protect line as follows:

1. If the program protect line indicates that the I/O instruction is unprotected, then for each function and data transfer command, the controller generates a reject. However, a status command is never rejected.

2. If the program protect line indicates that the I/O instruction is protected, then for all commands (function, status, and data transfers) a reply response is enabled by the protect condition. The controller does not return a reply until all other conditions within the controller indicate that it is permissible to do so.

TABLE 3-12. CONTROL FUNCTIONS SWITCH

Position	Segment 1	Segment 3
On	Auto-Rewind Enabled	Protected Mode
Off	Auto-Rewind Inhibited	Unprotected Mode

CASSETTE TAPE DRIVE

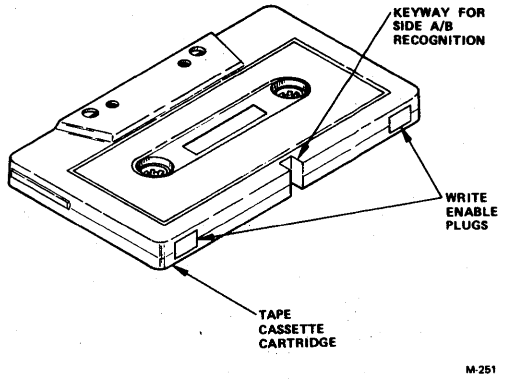
The cassette tape drive contains the following controls: a lid interlock switch, a side A/B switch, and a write enable switch. See figure 3-13.

Lid Interlock Switch

This is an interlock switch that is depressed by the drive lid when closed. When the lid is open, all drive functions are disabled. Opening the lid is the primary means for making the cassette Not Ready (refer to deadstart operation).

Side A/B Switch

This switch senses a notch in the tape cassette to determine which side (A or B) is up. Position of this switch is reported as a status bit.



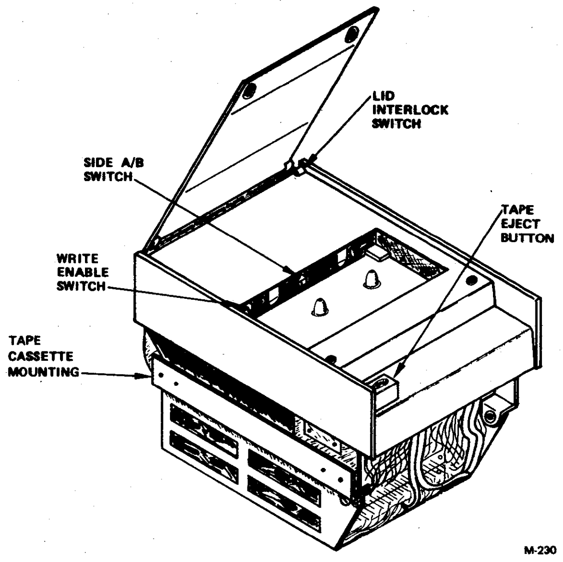


Figure 3-13. Cassette Tape Drive Controls

Write Enable Switch

This switch senses the presence or absence of the write enable plug (at the cassette) for whichever track is positioned under the read/write head.

CONSOLE SWITCHBOX

The console switchbox, shown in figure 1-2, allows the operator to select between the base and multiplex processors in the 2552-2 NPU.

SYSTEM AUTOSTART MODULE

The controls and indicators for the system autostart module are shown in figure 3-14. Table 3-13 describes the function of the controls and indicators.

START-UP PROCEDURE

The NPU power is normally on all the time, except for motor-driven peripherals which are turned on only as needed. The following procedure is to be used when the NPU has been completely shut off (for example, following an emergency shutdown):

1. Set main ac power circuit breakers at the rear of each cabinet bay to the on position.
2. Set PWR switch on each loop multiplexer to ON.
3. Observe illumination of all LED indicators on communications line adapters, loop multiplexers, and the maintenance panel.

SYSTEM AUTOSTART MODULE OPERATION

To intialize the NPU with the SAM-C tape, proceed as follows:

1. Set autostart module switch to DISABLE.
2. Press MASTER CLEAR switch on maintenance panel.
3. Insert SAM-C cassette, close lid, and wait for tape rewind and load point.
4. When CASSETTE READY indicator on maintenance panel lights, press DEADSTART switch. The tape then starts and the system loads.
5. Set autostart module switch to ENABLE when MONITOR RESET PULSE indicator starts flashing.
6. CASSETTE READY indicator on maintenance panel remains lit. Tape has rewound and returned to load point.

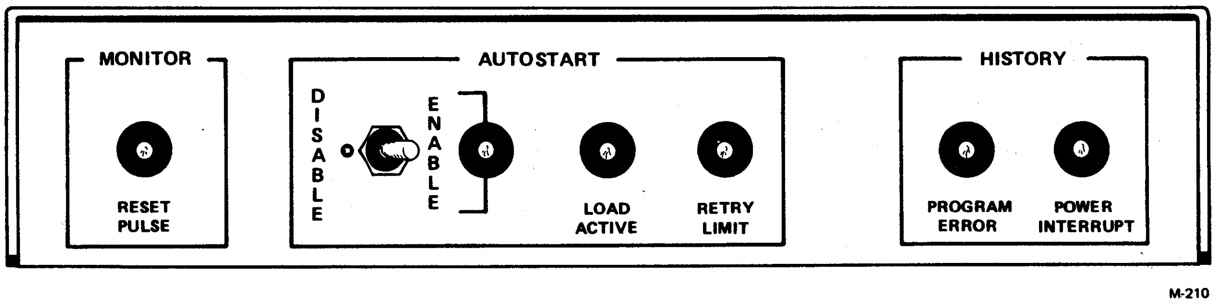


Figure 3-14. System Autostart Module Controls and Indicators

TABLE 3-13. SYSTEM AUTOSTART MODULE CONTROLS AND INDICATORS

Panel Name	Function
MONITOR RESET PULSE indicator	When lit, indicates programmed reset pulses are being received.
AUTOSTART DISABLE/ ENABLE toggle switch	In DISABLE position, disables autostart module; in ENABLE position, starts 3-second and 5-minute timers.
AUTOSTART ENABLE indicator	When lit, indicates autostart module is enabled, tape is in cassette, programmed reset pulses are being received and cassette is at load point.
AUTOSTART LOAD ACTIVE indicator	Indicates program is being loaded.
AUTOSTART RETRY LIMIT indicator	Indicates autoload module reloaded the program repeatedly, until retry limit expired and each reload was unsuccessful (no programmed pulses occurred).
HISTORY PROGRAM ERROR indicator	Indicates interval between reset pulses has exceeded 3 seconds.
HISTORY POWER INT indicator	Indicates system power was interrupted.

7. Autostart module indicators are as follows:

ENABLED	- On
LOAD ACTIVE	- Off
RETRY LIMIT	- Off
RESET PULSES	- Flashing
PROGRAM ERROR	- Off
POWER INT	- Off

NOTE

PROGRAM ERROR, RETRY LIMIT and POWER INT indicators are latched on. Set DISABLE/ENABLE switch to off then on momentarily to reset these indicators. When the autostart module is enabled and reload sequence begins, REWIND and INITIATE LOAD switches on maintenance panel are inhibited.

8. To check recovery from a power failure, set cabinet ac power circuit breaker to OFF momentarily. After recovery, set autostart module switch to off momentarily to reset POWER INT indicator.

9. To change the tape at any time, even during a load sequence, open the tape cassette lid.

10. Remove and install cassette tape and close lid.

11. Set autostart module switch to DISABLE momentarily to reset PROGRAM ERROR indicator.

OPERATING PROCEDURES

MASTER CLEAR

The NPU can be master-cleared by any of the following actions:

1. By depressing the question mark (?) key on the communications console.

2. By depressing the ESCAPE key on a teletypewriter, or if no ESCAPE key, by depressing SHIFT, CONTROL, and K simultaneously.

3. By depressing MASTER CLEAR switch on the maintenance panel.

4. By receipt of an external master clear signal from the host computer.

5. By a power-on master clear.

DISPLAY CONTENTS

To display the contents of the function control register at any time, enter on input device:

```
J77:
||└──end of message symbol
|└───dummy number (status portion of function control register, which
|    cannot be affected)
└────control function
```

Whenever a printout occurs, the initial symbol is the last control function entered. When the same control function is required, it need not be reentered. The stored value is changed by master clear (to 000) or error (to 111).

CLEAR BIT

To clear a single bit position in the function control register, enter:

```
H14:
||└──end of message symbol
|└───bit number (hexadecimal) to be cleared
└────control function
```

The updated function control register is displayed at the console.

SET BIT

To set a single bit position in the function control register, enter:

```
I14:
||└──end of message symbol
|└───bit number (hexadecimal) to be set
└────control function
```

The updated function control register is displayed at the console.

CHANGE CONTENTS (DIGIT MODE)

To change the contents of a function control register in digit (0_{16} thru 5) mode, enter:

```
J14:
|||└──end of message symbol
||└───value (0-F) hexadecimal
|└────specifies hexadecimal digit number
└─────control function
```

The above entry specifies a function control register change is to be made in the digit mode and updates function control register hexadecimal digit 1 to a value of 4_{16} (0100_2). To determine the meaning of the value, refer to table 3-2, where digit 1 is display 1, and a count code of 4 = A register. The response to the console is the updated function control register value (always displayed in 8 hexadecimal digits).

CHANGE CONTENTS (BIT MODE)

The operator cannot change the contents of function control register digits 6_{16} and 7, as they are readouts of machine status. The digits 0_{16} thru 5 can all be changed with the same command from the console, as follows:

a. ESCAPE key (or SHIFT/CTRL/K) (Panel Mode)

b. ? (Master Clear)

c. ESCAPE key (PANEL MODE)

d. To change digits 0_{16} thru 5 together, enter digits as follows:

```
K710008xx:
|||||||| |
|||||||| └─end of message symbol
||||||||
|||||||└───anything
|||||||
||||||└────enable console echo
||||||
|||└┴┴─────{default
|||         default
|||         default
||└────────P register
|└─────────main memory
└──────────control function
```

Default is the prior value of the function control register that is not to be changed (in this case, zeros). The response to the console is the updated function control register.

TOGGLE UPPER INDICATOR

To toggle the UPPER indicator on the maintenance panel (alternately display the upper and lower 16 bits of the function control register), enter the following:

```
J:
||
|└──────end of message symbol
|
└───────control function
```

DISPLAY REGISTER CONTENTS (DISPLAY 0)

To display the contents of any of the registers defined in display 0 (see table 3-2), use the following procedure (Assume the operator wants to display the contents of K register at the console.):

1. Define in the function control register the element to be displayed. The console input is:

```
J02:
││││
│││└─────end of message symbol
││└──────specifies K register
│└───────specifies display 0
└────────control function
```

   The above input specifies function control register change is in digit mode and defines K register in display 0. The response to the console is updated function control register.

2. Type a command from the console to dispaly 0:

```
L:
││
│└───────end of message symbol
└────────control function
```

   The response to the console is the present value of K register.

DISPLAY REGISTER CONTENTS (DISPLAY 1)

The operator can display the contents of any of the registers defined in display 1 (see table 3-2) by using the following procedure (Assume the operator wants to display the contents of P register at the console.):

1. Define in the function control register the element to be displayed. The console input is:

```
J11:
││││
│││└─────end of message symbol
││└──────specifies P register
│└───────specifies display 1
└────────control function
```

   The above input specifies function control register change is in the digit mode and defines P register in display 1. The response to the console is the updated function control register.

2. Type a command from the console to display 1:

```
K:
||
|└──────end of message symbol
|
└───────control function
```

The response to the console is the present value of P register.

LOAD A REGISTER (DISPLAY 0)

The operator can load a value into any register defined in display 0 (see table 3-2) by using the following procedure (Assume the operator wants to load $14FE_{16}$ into N register.):

1. Define the element to be loaded.

```
J01:
||||
|||└────end of message symbol
|||
||└─────specified N register
||
|└──────specifies display 0
|
└───────control function
```

2. Enter the update value.

```
L14FE:
| ‾|‾ |
|  |  └─end of message symbol
|  |
|  └────update value in hexadecimal
|
└───────control function
```

3. Verify that the new value has been transferred into desired register.

```
L:
||
|└──────end of message symbol
|
└───────control function
```

LOAD A REGISTER (DISPLAY 1)

The operator can load a value into any register defined in display 1 (see table 3-2 by using the following procedure (Assume the operator wants to load $14FE_{16}$ into the breakpoint register.):

1. Define the element to be loaded:

```
J16:
||||
|||└──end of message symbol
|||
||└───specifies breakpoint register
||
|└────specifies display 1
|
└─────control function
```

2. Enter the update value.

```
K14FE:
| |  |
| |  └──end of message symbol
| |
| └─────update value in hexadecimal
|
└───────control function
```

3. Verify that new value has been transferred into desired register.

```
K:
||
|└──────end of message symbol
|
└───────control function
```

TAPE CASSETTE AUTOLOAD

1. Master clear the NPU using any of the methods previously described.

2. Insert tape cassette in tape drive and close lid securely. Tape drive is enabled and tape moves to beginning-of-tape position.

3. Press ESCAPE key on communications console.

4. Press DEADSTART switch on maintenance panel. Tape loads and processor starts.

NOTE

When loading a program, observe the value displayed in the three CONTROL CODE indicators (figure 3-1). When all three indicators are lit simultaneously, an error is indicated. Frequently the final value of the CONTROL CODE indicators is the error condition (all on) if an error was sensed anywhere in the load operation. This applies regardless of the type of autoload device used.

HOST COMPUTER AUTOLOAD

The host computer autoload is a software function that provides a DOWNLINE LOAD command to the NPU. In order for this command to be received, the ON LINE/OFF LINE switch on the communications coupler CYBER interface must be set to ON LINE.

START PROCESSOR

The NPU can be started from the console by entering the following command:

```
I:
││
│└────end of message symbol
│
└─────control function
```

If bit $0C_{16}$ (12_{10}) of the function control register is 0, both microprograms and macroprograms begin running. If this bit is 1, only microprograms begin running.

STOP PROCESSOR

The NPU can be stopped from the console by entering the following command:

```
H:
││
│└────end of message symbol
│
└─────control function
```

If function control register bit $0C_{16}$ (12_{10}) is 0, a macro stop occurs. If this bit is 1, a micro stop occurs.

Halt the processor when inserting patches to programs being executed. If not halted, patches may not enter the program as expected.

PROCEDURE EXAMPLES

The following paragraphs contain typical procedural sequences used in operating the NPU.

Display Main Memory Location

The contents of a specific location within main memory can be interrogated and displayed by entering the following:

1. ESCAPE (Panel Mode)
2. ? (Master Clear)
3. ESCAPE (Panel Mode)
4. Load function control register to define P register and main memory

   K710000800:

   Verify function control register update.
5. Load P with desired memory location

   Kxxxx: $(\text{xxxx} \rightarrow \text{P})$

6. Display memory location

   L:

7. Response to console is memory contents pointed to by the P register; the P register is incremented.

8. Consecutive memory locations can be displayed each time the following is entered at the console:

```
  :
response
  :
response, etc.
```

Write Into Main Memory

Specific locations within main memory can be loaded by entering the following:

1. ESCAPE

2. ?

3. ESCAPE

4. K71000800:, set up function control register

5. Kxxxx:, (xxxx→P)

6. Lyyyy:, (yyyy→loc xxxx)

   The P register has automatically been incremented following the : from the console.

7. The next memory location (xxxx +1) can now be loaded with a new value by Lyyyy:, etc., without having to update P.

Display P Register in Repeat Mode

The P register can be displayed in the repeat mode by entering the following:

1. ESCAPE (Panel Mode)

2. ? (Master Clear)

3. ESCAPE (Panel Mode)

4. K71000400: (Select P register, repeat mode)

5. K: (Display P contents)

   Response is P register contents, repeatedly. This command is usually used with local (maintenance panel) operations rather than console mode when a TTY is used as the console device.

To display the P register locally (to determine, for example, if the program is running), enter the following:

1. LOCAL/REMOTE switch to local
2. K11: (selects P register)
3. J54: (sets repeat mode, resets console echo)

   Response is P register contents, repeatedly. To exit this mode, enter:

   J58: (sets console echo, resets repeat mode)

Operation in Step Mode

The processor can be placed in the step mode, and programs can be executed in single steps (individual instructions) by entering the following:

1. ESCAPE (Panel Mode)
2. ? (Master Clear)
3. ESCAPE (Panel Mode)
4. K71100800: (load function control register)
5. K1000: (1000 $\rightarrow$ P register)
6. K: (display P contents)
7. Repeat K: to execute individual instructions sequentially through the program.

Load and Execute Macroprogram

A program can be loaded into main memory and executed by entering the following:

1. ESCAPE (Panel Mode)
2. ? (Master Clear)
3. ESCAPE (Panel Mode)
4. K71000800: (load function control register; select P register, memory; console echo)
5. K1000: (0B00 Loc 1000)
6. L0B00: (0B00 Loc 1000)
7. L0B00: (0B00 Loc 1001)
8. L0B00: (0B00 Loc 1002)
9. L0B00: (0B00 Loc 1003)
10. L18FB (18FB Loc 1004)

11. K1000: (1000 P)

12. I:, execute program starting at location 1000

EMERGENCY-OFF PROCEDURE

1. Set the main ac power circuit breakers at the rear of each bay of the NPU to the off position.

2. Turn off all peripherals.

CHECKS AND ADJUSTMENTS

Operational checks are limited to observing the LED indicators on the circuit cards and the maintenance panel to ensure the presence of all required signals as described in controls and indicators portions of this section.

SHUTDOWN PROCEDURE

The following procedure is to be used should the operator desire to completely shut down the NPU.

1. Depress MASTER CLEAR switch on maintenance panel.

2. Set PWR switches on all loop multiplexers and communications line adapters to the off position.

3. Set main ac power circuit breakers at the rear of each cabinet bay of the NPU to the off position.

COMMUNICATIONS PROCESSOR INSTRUCTIONS 4

This section contains the format and instruction descriptions for macroinstructions and microinstructions for the communications processor. The macrolevel instructions set is compatible with the CDC 1700 instruction repertoire and are stored in the main memory. Microinstructions that control operation of the processor are stored in the micromemory. Information on macro and microinstruction timing is also included in this section.

MACROINSTRUCTION FORMATS AND INSTRUCTIONS

The macroinstruction format consists of 16-bit words numbered right to left as 0 to 15 with the leftmost bit (15) as the most significant bit.

MACROINSTRUCTION WORD

The basic macroinstruction formats and enhanced instruction formats are described below.

BASIC MACROINSTRUCTION FORMATS

1. Storage Reference Instructions
2. Register Reference Instructions
   - Interregister Instructions
   - Skip Instructions
   - Shift Instructions

NOTE

The latter three type instructions are subgroups of the Register Reference Instructions.

ENHANCED MACROINSTRUCTION FORMATS

1. Type 2 Storage Reference Instruction
2. Field Reference Instruction
3. Decrement and Repeat Instructions
4. Type 2 Interregister Instructions
5. Miscellaneous Instructions
   - Load Micromemory
   - Load Registers
   - Store Registers
   - Set/Sample Output or Input
   - Sample Position/Status
   - Define Microinterrupt

- Clear Breakpoint Interrupt
- Generate Character Parity Even
- Generate Character Parity Odd
- Scale Accumulator
- Load Upper Unprotected Bounds
- Load Lower Unprotected Bounds
- Execute Microsequence

6. Autodata Transfer Instructions

STORAGE REFERENCE INSTRUCTION FORMAT

The storage reference instruction format, shown in figure 4-1, contains three fields: instruction, address mode, and delta. The instruction field contains the operation code. The address mode field contains flags for indexing, indirect addressing, and relative addressing. The delta field is a signed 8-bit address modifier where the most significant bit is the sign bit, except where noted.

The following definitions apply to the description of addressing modes:

1. Instruction Address - The address of the instruction being executed; contents of P register.

2. Indirect Address - A storage location that contains an address rather than an operand.

3. Base Address - The operand address after all indirect addressing but before modification by the index registers. The base address is the effective address when no indexing is specified.

4. Effective Address - The final address of the operand. At certain times the effective address equals the operand for read-operand type instructions. See table 4-1.

5. Indexing - The computer has two index registers. Index register 1 is the Q register. Index register 2 is storage location $00FF_{16}$ (the I register). The base address can be modified by either or both of the index registers. If index register 1 flag is set, the contents of the

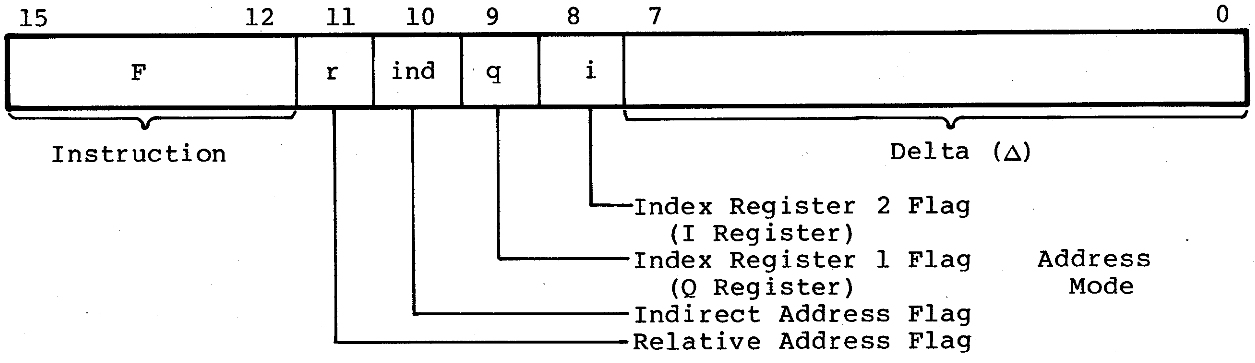


Figure 4-1. Storage Reference Instruction Format

TABLE 4-1. STORAGE REFERENCE INSTRUCTION ADDRESSING

Mode	Binary b11-b8	Hex	Delta	Effective Address	Address of Next Instruction
Absolute Constant	0000	0	≠0 =0	Δ P+1	P+1 P+2
Absolute Constant	0001	1	≠0 =0	Δ+(00FF) (P+1)+(00FF) †	P+1 P+2
Absolute Constant	0010	2	≠0 =0	Δ+(Q) (P+1)+(Q) †	P+1 P+2
Absolute Constant	0011	3	≠0 =0	Δ+(Q)+(00FF) (P+1)+(00FF) †	P+1 P+2
Indirect Storage ††	0100	4	≠0 =0	(Δ) (P+1)	P+1 P+2
Indirect Storage ††	0101	5	≠0 =0	(Δ)+(00FF) (P+1)+(00FF)	P+1 P+2
Indirect Storage ††	0110	6	≠0 =0	(Δ)+(Q) (P+1)+(Q)	P+1 P+2
Indirect Storage ††	0111	7	≠0 =0	(Δ)+(Q)+(00FF) (P+1)+(Q)+(00FF)	P+1 P+2
Relative 16-bit Relative	1000	8	≠0 =0	P+Δ P+1+(P+1)	P+1 P+2
Relative 16-bit Relative	1001	9	≠0 =0	P+Δ+(00FF) P+1+(P+1)+(00FF)	P+1 P+2
Relative 16-bit Relative	1010	A	≠0 =0	P+Δ+(Q) P+1+(P+1)+(Q)	P+1 P+2
Relative 16-bit Relative	1011	B	≠0 =0	P+Δ+(Q)+(00FF) P+1+(P+1)+(Q)+(00FF)	P+1 P+2
Relative Indirect 16-bit Relative Indirect ††	1100	C	≠0 =0	(P+Δ) (P+1+(P+1))	P+1 P+2
Relative Indirect 16-bit Relative Indirect ††	1101	D	≠0 =0	(P+Δ)+(00FF) (P+1+(P+1))+(00FF)	P+1 P+2

TABLE 4-1. STORAGE REFERENCE INSTRUCTION ADDRESSING (Contd)

Mode	Binary b11-b8	Hex	Delta	Effective Address	Address of Next Instruction
Relative Indirect 16-bit Relative Indirect[††]	1110	E	$\neq 0$ $=0$	$(P+\Delta)+(Q)$ $(P+1+(P+1))+(Q)$	P+1 P+2
Relative Indirect 16-bit Relative Indirect[††]	1111	F	$\neq 0$ $=0$	$(P+\Delta)+(Q)+(00FF)$ $(P+1+(P+1))+(Q)+(00FF)$	P+1 P+2

[†]Effective address = operand for read-operand type instruction.
[††]Multilevel only in 32K mode.

Note: The 255X NPU normally does not use 32K mode.

Q register are added to the base address to form the effective address. If the index register 2 flag is set, the contents of storage location 00FF are added to the base address to form the effective address. If both index register flags are set, the contents of Q are added to the base address; then the contents of $00FF_{16}$ are added to the result to form the effective address. Indexing occurs after completion of indirect addressing.

The computer uses the 16-bit one's complement adder during indexing operations. Consequently, the index register contents are treated as signed quantities (bit 15 is the sign bit).

The storage reference instructions have eight different types of addressing modes:

1. Absolute
2. Constant
3. Indirect
4. Storage
5. Relative
6. 16-bit Relative
7. Relative Indirect
8. 16-bit Relative Indirect

The eight types of addressing modes can be established using the storage reference instructions:

1. Absolute (address mode bits are 0, 1, 2, or 3). Both relative and indirect flags equal 0 and delta is not 0. The base address equals delta. Delta has no sign bit. The contents of the index registers, when specified, are added to the base address to form the effective address.

2. Constant (address mode bits are 0, 1, 2, or 3). Both relative and indirect flags are 0 and delta is 0. Where the address mode bits are 0, P + 1 is the effective address. Where the address mode bits are 1, 2, or 3, the contents of P + 1, plus the contents of one or both index registers form the effective address. The effective address is taken as the operand for read-operand type instructions.

3. Indirect (address mode bits are 5, 6, or 7). Relative address flag is 0, indirect flag is 1, and delta is not 0. The 8-bit value of delta is an indirect address. Delta is a magnitude quantity for this operation (no sign bit).

4. Storage (address mode bits are 4, 5, 6, or 7). Relative address flag equals 0, indirect flag is 1, and delta is 0. The contents of location P + 1 are an indirect address. When the base address is formed (indirect addressing complete), the contents of one or both index registers, if specified, are added to form the effective address.

5. Relative (address mode bits are 8, 9, A, or B). The relative flag is 0, and delta is not 0. The base address is equal to the instruction address, P, plus the value of delta with sign extended. The contents of the index registers, when specified, are added to the base address to form the effective address. The address referenced by this mode is located forward or backward relative to the P in the program.

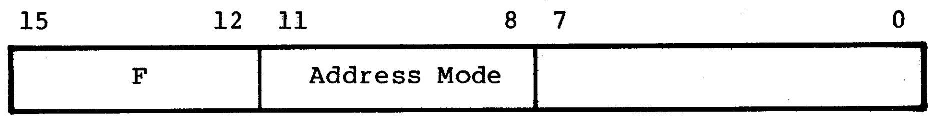
6. 16-bit Relative (address mode bits are 8, 9, A, or B). The relative address flag is 1, the indirect address flag is 0, and delta is 0. If no indexing is specified, the instruction address P + 1, plus the contents of location P + 1, form the base address = effective address. If indexing is specified, the contents of the specified index registers are added to the base address to form the effective address.

7. Relative Indirect (address mode bits are C, D, E, or F). Both relative and indirect flags are 1. If delta is not 0, the value of the instruction address, P, plus the value of delta with sign extending is an indirect address. When in 32K mode (multilevel indirect), bit 15 of the contents of this indirect address is 0, the contents of this indirect address is the base address. If bit 15 of the contents of the indirect address is set, the contents of the indirect address is another indirect address. Indirect addressing continues until bit 15 of the contents of the indirect address is 0. The contents of the index registers, when specified, are then added to the base address to form the effective address. When in the 65K mode (single level indirect), all 16 bits of the indirect address is the base address, regardless of whether bit 15 is set.

8. 16-bit Relative Indirect (address mode bits are C, D, E, or F). Both relative and indirect flags are 1. In 65K mode, the contents of P + 1 + (P + 1)[1] is the base address. [1]( ) denotes contents of. Then the contents of the index registers, when specified, are added to the base address to form the effective address. In 32K mode, P + 1 + (P + 1) = base address if bit 15 of (P + 1) is 0; if 1, then P + 1 + (P + 1) forms an indirect address and the indirection continues until bit 15 is 0.

Table 4-1 shows all the addressing possibilities for storage reference instructions that can be obtained through combinations of flag bits.

STORAGE REFERENCE INSTRUCTIONS



The storage reference instructions areas are as follows:

1. Unconditional Jump (F = 1) — JMP

   Effective address specifies the location of the next instruction.

2. Multiply Integer (F = 2) — MUI

   Multiply the contents of the storage location specified by the contents of the A register. The 32-bit product replaces the contents of Q and A with the most significant bits in the Q register. One's complement arithmetic is used.

3. Divide Integer (F = 3) — DVI

   Divide the combined contents of the Q and A registers with the contents of the effective address. The Q register contains the most significant bits before execution. The quotient is in the A register and the remainder in the Q register at the end of execution. The overflow indicator is set if the magnitude of the quotient is greater than the capacity of the A register. Once set, the overflow indicator remains set until a skip on overflow instruction is executed.

4. Store Q (F = 4) — STQ

   Store the contents of the Q register in the storage location specified by the effective address. The contents of Q are not changed.

5. Return Jump (F = 5) — RTJ

   Replace the contents of the storage location specified by the effective address with the address of the next consecutive instruction. The address stored in the effective address is either P + 1 or P + 2, depending on the addressing mode of RTJ. The contents of P are then replaced with the effective address + 1.

6. Store A (F =6) — STA

   Store the contents of the A register in the storage location specified by the effective address. The contents of A are not altered.

7. Store A, Parity to A
(F = 7)

SPA

Store the contents of the A register in the storage location specified by the effective address. Set the A register to 0001 if the parity bit of the word stored in the effective address is set. If the parity bit is not set, set the A register to 0000.

8. Add to A
(F = 8)

ADD

Add the contents of the storage location specified by the effective address to the contents of the A register. One's complement arithmetic is used. The overflow indicator is set if the magnitude of the sum is greater than the capacity of the A register. Once set, the overflow indicator remains set until a skip on overflow instruction is executed.

9. Subtract from A
(F = 9)

SUB

Subtract the contents of the storage location spcified by the effective address from the contents of the A register. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

10. AND with A
(F = A)

AND

Form the logical product, bit by bit, of the contents of the storage location specified by the effective address and the contents of the A register. The result replaces the contents of A.

11. Exclusive OR with A
(F = B)

EOR

Form the logical difference (exclusive OR), bit by bit, of the contents of the storage location specified by the effective address and the contents of the A register. The results replace the contents of A.

12. Load A
(F = C)

LDA

Load the A register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

13. Replace Add One in Storage
(F = D)

RAO

Add one to the contents of the storage location specified by the effective address. The contents of A and Q are not changed. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

14. Load Q (F = E) **LDQ**

Load the Q register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

15. Add to Q (F = F) **ADQ**

Add the contents of the storage location specified by the effective address to contents of the Q register. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

REGISTER REFERENCE INSTRUCTION FORMAT

Register reference instructions use the address mode field for the operation code. Register reference instructions are identified by zeros in the upper 4 bits of an instruction. See figure 4-2.

REGISTER REFERENCE INSTRUCTIONS

The register reference instructions are as follows:

1. Selective Stop (F1 = 0) $\Delta = 0$ **SLS**

   Stops the machine if this instruction is executed when the STOP switch is on. The instruction becomes a NOP when the switch is off.

2. Input to A (F1 = 2) **INP**

   Read one word from an external device into the A register. The word in the Q register selects the sending device. If the device sends a reply, the next instruction comes from P + 1. If the device sends an external reject, the next instruction comes from P + 1 +Δ, where Δ is an 8-bit signed number including sign. An internal reject causes the next instruction to come from P +Δ.

3. Output from A (F1 = 3) **OUT**

   Deliver one word from the A register to an external device. The word in the Q register selects the receiving device. If the device sends a reply, the next instruction comes from P + 1. If the device sends an external reject, the next instruction comes from P + 1 +Δ, where Δ is an 8-bit signed number including sign. An internal reject causes the next instruction to come from P +Δ.

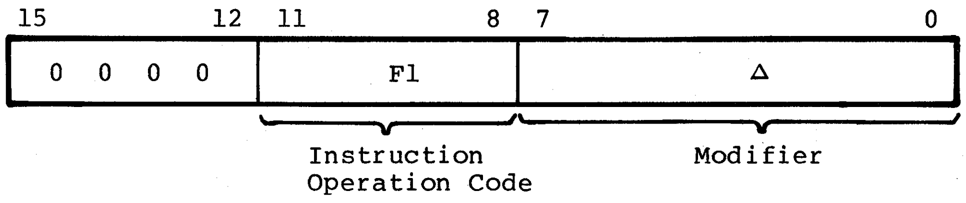


Figure 4-2. Register Reference Instruction Format

4. Increase A (F1 = 9) **INA**

   Replaces the contents of A with the sum of the initial contents of A and delta. Delta is treated as a signed number with the sign extended into the upper 8 bits. Operation of overflow is the same as in ADD.

5. Enter A (F1 = A) **ENA**

   Replaces the contents of the A register with 8-bit delta, sign extended.

6. No Operation (F1 = B) Δ = 0 **NOP**

7. Enter Q (F1 = C) **ENQ**

   Replaces the contents of Q with the 8-bit delta, sign extended.

8. Increase Q (F1 = D) **INQ**

   Replaces the contents of Q with the sum of the initial contents of Q and delta. Delta is treated as a signed number with the sign extended into the upper 8 bits. Operation of overflow is the same as in ADD.

The following instructions are legal only when the PROGRAM PROTECT switch[†] is off or the instructions themselves are protected. If an instruction is illegal, it becomes a Selective Stop and an interrupt on Program Protect Fault is possible (if selected).

Switch ON:	Selective Stop unless instruction is protected.
Switch OFF:	Normal execution (no program protection).

9. Enable Interrupt (F1 = 4) Δ = 0 **EIN**

   Activates the interrupt system. The interrupt system must be active and the mask bit set for an interrupt to be recognized.

[†] A software switch activated by setting a bit in the Function Control register. See table 3-3.

10. Inhibit Interrupt $\boxed{\text{IIN}}$
(Fl = 5) = 0

Deactivates interrupt system.

11. Set Program Protect $\boxed{\text{SPB}}$
(Fl = 6) = 0

Sets the program protect bit in the address specified by Q.

12. Clear Program Protect $\boxed{\text{CPB}}$
(Fl = 7) = 0

Clears the program protect bit in the address specified by Q.

13. Exit Interrupt State $\boxed{\text{EXI}}$
(Fl = E)

This instruction is used to exit from any interrupt state. Delta defines the interrupt state from which the exit is taken. This instruction reads the address, resets the overflow indicator, if necessary, activates the interrupt system and jumps to the return address.

14. Interregister Instructions
(Fl = 8)

These instructions cause data from certain combinations of origin registers to be sent through the adder to some combination of destination registers. Various operations, selected by the adder control lines, are performed on the data as it passes through the adder. See figure 4-3.

If M is the destination register and the instruction is not protected and the program protect switch is set, this is a nonprotected Selective Stop instruction. The program protect fault bit is set and interrupt occurs if selected.

The origin registers are considered as operands. There are two kinds defined as follows:

a. Operand 1 can be one of the following:

(1) $FFFF_{16}$ (bit 5 = 0)

(2) The contents of A (bit 5 = 1)

b. Operand 2 can be one of the following:

(1) $FFFF_{16}$ (bit 4 = 0 and bit 3 = 0)

(2) The contents of M (bit 4 = 0 and bit 3 = 1)

(3) The contents of Q (bit 4 = 1 and bit 3 = 0)

(4) The OR, bit-by-bit, of the contents of Q and M
(bit 4 = 1 and bit 3 = 1)

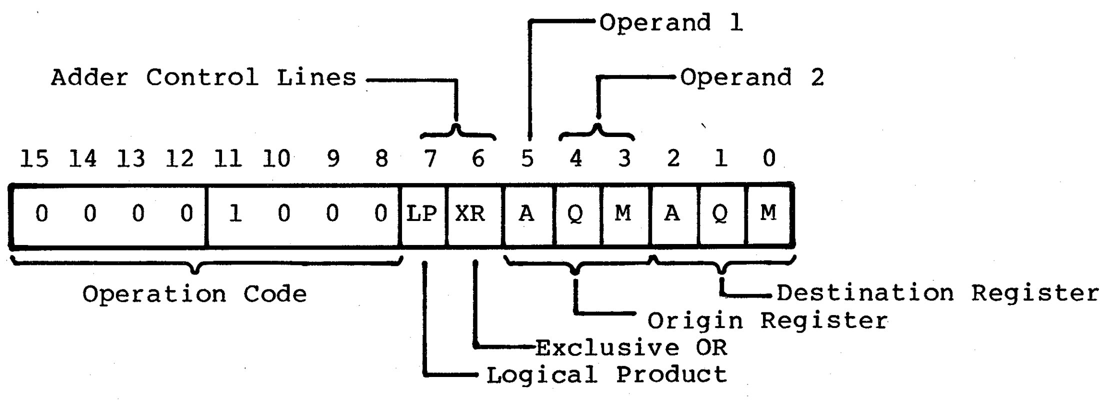


Figure 4-3. Interregister Instruction Format

The following operations are possible (see table 4-2 for examples of all possible 4-bit operands):

a. LP = 0 and XR = 0 - The data place in the destination register(s) is the arithmetic sum of operand 1 and operand 2. The overflow indicator operates the same as in ADD.

b. LP = 1 and XR = 0 - The data placed in the destination register(s) is the logical product-bit-by-bit, of operand 1 and operand 2.

c. LP = 0 and XR = 1 - The data placed in the destination register(s) is the exclusive OR, bit-by-bit, of operand 1 and operand 2.

TABLE 4-2. INTERREGISTER INSTRUCTION TRUTH TABLE

Operand 1	Operand 2	LP=Q LR=1	LP=1 XR=0	LP=1 XR=0	LP=0 XR=0
0 0 1 1	0 1 0 1	0 1 1 0 (Exclusive OR)	0 0 0 1 (Logical Product)	1 1 1 0 (Complement Logical Product)	Arithmetic Sum

NOTES: 1. Register transfers can be accomplished with LP = 0, XR = 0, and by making operand 1 or operand 2 equal to $FFFF_{16}$.

2. Magnitude comparisons without destroying either operand can be done with LP = 0, XR = 0, no destination register selected, and by testing the overflow indicator.

3. Complementing registers can be done with LP = 0, XR = 1, and making either operand 1 or operand 2 equal to $FFFF_{16}$.

d. LP = 1 and XR = 1 - The data in the destination registers is the complement of the logical product, bit-by-bit, of operand 1 and operand 2.

The Interregister instructions for the processor are as follows:

Mnemonic	Description	Bit 765434
SET	Set to Ones	10000
CLR	Clear to Zero	01000
TRA	Transfer A	00100
TRQ	Transfer Q	00010
TRM	Transfer M	00001
TRB	Transfer Q+M	00011
TCA	Transfer Complement A	01100
TCM	Transfer Complement M	01001
TCQ	Transfer Complement Q	01010
TCB	Transfer Complement Q+M	01011
AAM	Transfer Arithmetic Sum A,M	00101
AAB	Transfer Arithmetic Sum A,Q+M	00111
AAQ	Transfer Arithmetic Sum A,Q	00110
EAM	Transfer Exclusive OR A,M	01101
EAQ	Transfer Exclusive OR A,Q	01110
EAB	Transfer Exclusive OR A,Q+M	01111
LAM	Transfer Logical Product A,M	10101
LAQ	Transfer Logical Product A,Q	10110
LAB	Transfer Logical Product A,Q+M	10111
CAM	Transfer Complement Logical Product A,M	11101
CAQ	Transfer Complement Logical Product A,Q	11110
CAB	Transfer Complement Logical Product A,Q+M	11111

NOTE: The symbol + implies an OR.

15. Skip Instruction (F1 = 1)

Skip Instruction

Format for this instruction is shown in figure 4-4.

Skip instructions are identified when the address mode field is 1 and the instruction mode field is 0. When the skip condition is met, the contents of the skip count +1 are added to P to obtain the address of the next instruction (e.g., when the skip condition is not met, the address of the next instruction is P+1 (skip count ignored). The skip count does not have a sign bit.

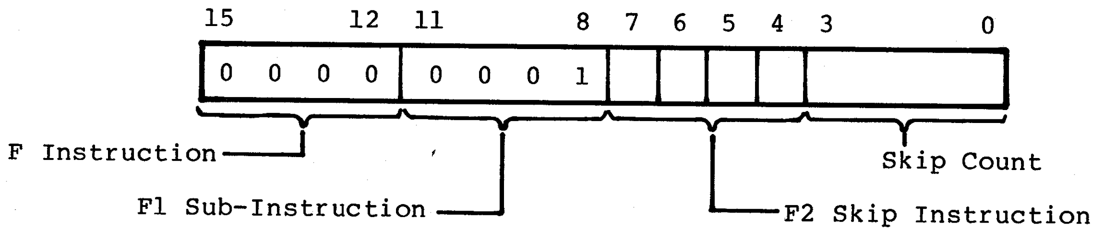


Figure 4-4. Skip Instruction Format

The skip instructions of the processor are as follows:

SAZ	Skip if A = +0 (F2 = 0) all bits zero
SAN	Skip if A $\neq$ +0 (F2 =1) not all bits zero
SAP	Skip if A = positive (F2 = 2)
SAM	Skip if A = negative (F2 = 3)
SQZ	Skip if Q = +0 (F2 = 4) all bits zero
SQN	Skip if Q $\neq$ +0 (F2 = 5) not all bits zero
SQP	Skip if Q = positive (F2 = 6)
SQM	Skip if Q = negative (F2 = 7)
SWG	Skip if SELECTIVE SKIP switch set (F2 = 8)
SWN	Skip if SELECTIVE SKIP switch not set (F2 = 9)
SOV	Skip on overflow (F2 = A)
SNO	Skip on no overflow (F2 = B)
SPE	Skip on storage parity error (F2 = C)
SNP	Skip on no storage parity error (F2 = D)
SPF	Skip on Program Protect Fault (F2 = E)
SNF	Skip on no program protect Fault (F2 = F)

The skip-on-overflow (SOV) instruction performs skip if an overflow condition occurred. It clears the overflow indicator.

The skip-on-storage-parity-error (SPE) instruction performs skip if a storage parity error occurred. It clears the storage parity error signal and indicator.

The program protect fault is set by:

1. A nonprotected instruction attempting to write into a protected address.

2. A protected instruction executed immediately following a nonprotected instruction, except when an interrupt has caused the instruction sequence.

3. Execution of any nonprotected instruction which attempts to alter the interrupt system.

The program protect fault is cleared when an SPF or SNF is executed. The program protect fault cannot be set if the program protect system is disabled.

16. Shifts

Shifts

This shifts A or Q, or QA, left or right for the number of places specified by the 5-bit shift correct. The sign is extended on right shifts. Left shifts are end around.

Format for the shift instruction is shown in figure 4-5.

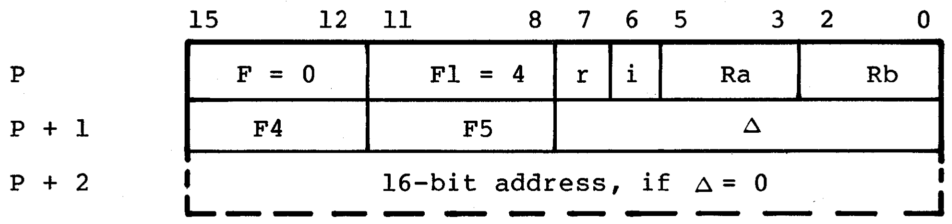
Shift Instructions

Mnemonic	Instruction Name
ARS	A Right Shift
ARS	Q Right Shift
LRS	Long Right Shift (QA)
ALS	A left Shift
LLS	Long Left Shift

ENHANCED INSTRUCTIONS

This section describes enhanced instruction formats.

Type 2 Storage Reference Instructions



The type 2 storage reference instruction is identified by a Δ ≠ 0 of the enable interrupt instruction (F = 0, Fl = 4). A type 2 instruction is made up of 2 (and sometimes 3) words. The r, i, Ra, and Δ subsections provide addressing modes; Rb specifies an operand.

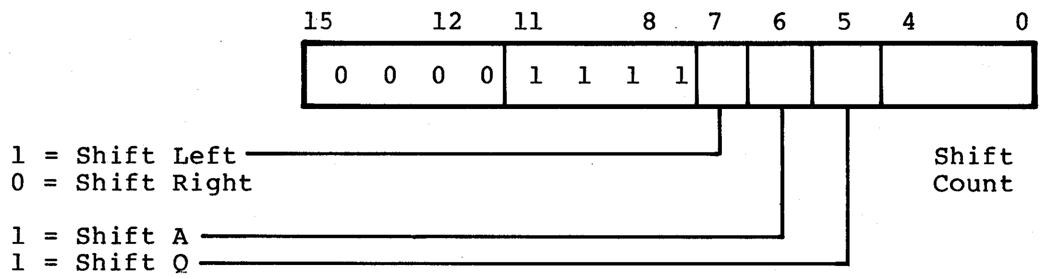


Figure 4-5. Shift Instruction Format

The F4 field determines the instruction (e.g., add, subtract, etc.); the F5 field determines the instruction mode and has the following definition:

F5 = 0 - word processing, register destination
F5 = 1 - word processing, memory destination
F5 = 2 - character processing, register destination
F5 = 3 - character processing, memory destination

NOTE

F5 is not used for subroutine jumps and subroutine exit. The register/memory destination bit of F5 is not used for compare instructions (see below).

The addressing mode fields contain four fields to determine the addressing:

1. Delta determines 8- or 16-bit addressing. If delta is zero, a third word is required to specify a 16-bit address.

2. Flag r is the relative address flag.

3. Flag i is the indirect address flag.

4. Register Ra is the index register.

The addressing modes are similar to type 1 storage instructions. Type 1 allows indexing by one or two registers (I and Q); while type 2 allows indexing by any one of seven registers (1, 2, 3, 4, Q, A, or I). Table 4-3 specifies the addressing modes, the effective address, and the address of the next instruction.

The addressing mode fields determine the effective address for operand A. Register Rb and the instruction mode field (F5) determine the address for operand B. Note that for character addressing, the effective address (operand A and register Rb) are combined to ascertain the actual character effective address (refer to character instructions). Operand B is always the A register for character addressing.

NOTE

For character addressing, a selection of absolute (r is zero), no indirect (i is zero), no index register (Ra is zero), and no character register (Rb = 0) results in an EIN instruction.

Unspecified combinations of F4, F5, and Rb are reserved for future expansion.

The following definitions apply to the description of addressing modes.

1. Instruction Address: The address of the instruction being executed, also called P.

2. Indirect Address: A storage address that contains an address rather than an operand. Note that there is no multilevel indirect addressing for type 2 storage reference instructions.

3. Base Address: The operand address after all indirect addressing but before modification by an index register. The base address is the effective address if no indexing is specified.

TABLE 4-3. TYPE 2 STORAGE ADDRESSING RELATIONSHIPS

Addressing Mode	Delta	r	i	Ra	Effective Address (EA)	Address of Next Instruction
8-bit Absolute	$\Delta \neq 0$	0	0	0	Δ	P+2
		0	0	1	Δ +(1)	P+2
		0	0	2	Δ +(2)	P+2
		0	0	3	Δ +(3)	P+2
		0	0	4	Δ +(4)	P+2
		0	0	5	Δ +(Q)	P+2
		0	0	6	Δ +(A)	P+2
		0	0	7	Δ +(I)	P+2
8-bit Absolute Indirect	$\Delta \neq 0$	0	1	0	(Δ)	P+2
		0	1	1	(Δ)+(1)	P+2
		0	1	2	(Δ)+(2)	P+2
		0	1	3	(Δ)+(3)	P+2
		0	1	4	(Δ)+(4)	P+2
		0	1	5	(Δ)+(Q)	P+2
		0	1	6	(Δ)+(A)	P+2
		0	1	7	(Δ)+(I)	P+2
8-bit Relative †	$\Delta \neq 0$	1	0	0	P+1+Δ	P+2
		1	0	1	P+1+Δ+(1)	P+2
		1	0	2	P+1+Δ+(2)	P+2
		1	0	3	P+1+Δ+(3)	P+2
		1	0	4	P+1+Δ+(4)	P+2
		1	0	5	P+1+Δ+(Q)	P+2
		1	0	6	P+1+Δ+(A)	P+2
		1	0	7	P+1+Δ+(I)	P+2
8-bit Relative Indirect †	$\Delta \neq 0$	1	1	0	(P+1+Δ)	P+2
		1	1	1	(P+1+Δ)+(1)	P+2
		1	1	2	(P+1+Δ)+(2)	P+2
		1	1	3	(P+1+Δ)+(3)	P+2
		1	1	4	(P+1+Δ)+(4)	P+2
		1	1	5	(P+1+Δ)+(Q)	P+2
		1	1	6	(P+1+Δ)+(A)	P+2
		1	1	7	(P+1+Δ)+(I)	P+2
Absolute Constant	$\Delta = 0$	0	0	0	P+2	P+3
		0	0	1	P+2+(1)	P+3
		0	0	2	P+2+(2)	P+3
		0	0	3	P+2+(3)	P+3
		0	0	4	P+2+(4)	P+3
		0	0	5	P+2+(Q)	P+3
		0	0	6	P+2+(A)	P+3
		0	0	7	P+2+(I)	P+3

TABLE 4-3. TYPE 2 STORAGE ADDRESSING RELATIONSHIP (Contd)

Addressing Mode	Delta	r	i	Ra	Effective Address (EA)	Address of Next Instruction
16-bit Absolute (Storage)	Δ = 0	0	1	0	(P+2)	P+3
		0	1	1	(P+2)+(1)	P+3
		0	1	2	(P+2)+(2)	P+3
		0	1	3	(P+2)+(3)	P+3
		0	1	4	(P+2)+(4)	P+3
		0	1	5	(P+2)+(Q)	P+3
		0	1	6	(P+2)+(A)	P+3
		0	1	7	(P+2)+(I)	P+3
16-bit Relative	Δ = 0	1	0	0	P+2+(P+2)	P+3
		1	0	1	P+2+(P+2)+(1)	P+3
		1	0	2	P+2+(P+2)+(2)	P+3
		1	0	3	P+2+(P+2)+(3)	P+3
		1	0	4	P+2+(P+2)+(4)	P+3
		1	0	5	P+2+(P+2)+(Q)	P+3
		1	0	6	P+2+(P+2)+(A)	P+3
		1	0	7	P+2+(P+2)+(I)	P+3
16-bit Relative Indirect	Δ = 0	1	1	0	(P+2+(P+2))	P+3
		1	1	1	(P+2+(P+2))+(1)	P+3
		1	1	2	(P+2+(P+2))+(2)	P+3
		1	1	3	(P+2+(P+2))+(3)	P+3
		1	1	4	(P+2+(P+2))+(4)	P+3
		1	1	5	(P+2+(P+2))+(Q)	P+3
		1	1	6	(P+2+(P+2))+(A)	P+3
		1	1	7	(P+2+(P+2))+(I)	P+3

†For these addressing modes, delta is sign extended.
NOTE: ( ) denotes contents of.

4. Effective Address: The final address of the operand.

5. Indexing: If specified, the contents of register Ra is added to the base address to form the effective address. Indexing occurs after completion of addressing.

   The computer uses the 16-bit one's complement adder during indexing operations. Consequently, the index register contents are treated as signed quantities (bit 15 is the sign bit).

6. Registers: Registers Ra and Rb are defined as follows:

Register	Value
None	0
1	1
2	2
3	3
4	4
Q	5
A	6
I	7

Type 2 storage reference instructions (table 4-3) have the following eight different types of addressing modes:

1. <u>8-Bit Absolute</u> - (r is zero, i is zero, and delta is not zero.) The base address equals delta. The sign bit of delta is not extended. The contents of index register Ra, when specified, are added to the base address to form the effective address.

2. <u>8-Bit Absolute Indirect</u> - (r is zero, i is one, and delta is not zero.) The 8-bit value of delta is an indirect address. The sign bit of delta is not extended. The contents of this address in low core (addresses 0001 to 00FF) is the base address. The contents of index register Ra, when specified, are added to the base address to form the effective address.

3. <u>8-Bit relative</u> - (r is one, i is zero, and delta is not zero.) The base address is equal to the instruction address plus one, P + 1, plus the value of delta with sign extended.

   The contents of index register Ra (when specified) are added to the base address to form the effective address.

   If no indexing takes place, the addresses that can be referenced in the 8-bit relative mode are restricted to the program area. Delta is 8 bits long, thus the computer references a location between P-$7E_{16}$ and P + 80_{16} inclusive.

4. <u>8-Bit Relative Indirect</u> - (r is one, i is one, and delta is not zero.) The address of the second word of the instruction, P + 1, plus the value of delta with sign extended is an indirect address. The contents of this address is the base address. The contents of index register Ra, when specified, is added to the base address to form the effective address.

5. <u>Absolute Constant</u> - (r is zero, i is zero, and delta is zero.) The address of the third word of instruction P + 2 is the base address. The contents of index register Ra, when specified, are added to the base address to form the effective address. Thus, when Ra is not specified, the contents of P + 2 is the value of the operand.

   Note that there is no immediate operand condition (i.e., indexing is specified and the instruction is a read-operand type) as there is for type 1 storage reference addressing.

6. <u>16-Bit Absolute</u> - (Storage) (r is zero, i is one, and delta is zero.) The base address equals the contents of P + 2. The contents of index register Ra, when specified, are added to the base address to form the effective address.

7. <u>16-Bit Relative</u> - (r is one, i is zero, and delta is zero.) The base address equals the contents of P + 2 plus P + 2. The contents of index register Ra, when specified, are added to the base address to form the effective address.

8. <u>16-Bit Relative Indirect</u> - (r is one, i is one, and delta is zero.) The address of the third word of the instruction, P + 2, plus the contents of the third word of the instruction is an indirect address. The contents of this address is the base address. The contents of the index register Ra, when specified, are added to the base address to form the effective address.

Instruction descriptions are as follows:

1. Subroutine/Jump Exit (F4 = 5, F5 = 0, Rb = 0) **SJE**

   The contents of P are replaced with the effective address. This instruction can be used as a jump or subroutine exit. For example, if delta is one and Ra has been set up by a previous subroutine jump (see below), control is returned following that subroutine jump.

   Note that subroutine jumps save the address of the instruction, rather than the next instruction, so the subroutine jump exit may be a two-word instruction (delta non-zero) rather than three.

   For example, the following program makes a subroutine jump at location 1000. Register A contains 1002 upon entry to the subroutine SUB. Upon completion, the SUB exits to location 1003.

```
1000      0446                     SJA+SUB
1001      5000
1002      2000                        .
1003                                  .
                                      .
2000      ....      SUB            . . .   . . .
                                      .
                                      .
                                      .
2020      0430                     SJE-1,A
2021      5001
```

   NOTE

   Since Rb is 0, a selection of absolute (r is 0), no indirect (i is 0), and no index register (Ra = 0) results in an EIN instruction.

2. Subroutine Jump **SJr**

   (F4 = 5, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
   (r = 1, 2, 3, 4, Q, A, or I)

   Load register r with the address of the last word of this instruction (i.e., P+1 for delta not zero; P+2 for delta zero). The contents of P are then replaced with the effective address.

3. Add Register **ARr**

   (F4 = 8, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
   (r = 1, 2, 3, 4, A, Q, or I)

   Add (using one's complement arithmetic) the contents of the storage location specified by the effective address to the contents of register r. Operation on overflow is the same as for the ADD instruction. The contents of storage are not altered.

4. Subtract Register $\boxed{\text{SBr}}$

(F4 = 9, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Subtract (using one's complement arithmetic) the contents of the storage location specified by the effective address from the contents of register r. Operation on overflow is the same as for the ADD instruction. The contents of storage are not altered.

5. AND Register $\boxed{\text{ANr}}$

(F4 = A, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Form the logical product (AND), bit-by-bit, of the contents of the storage location specified by the effective address and the contents of register r. The result replaces the contents of register r. The contents of storage are not altered.

6. AND Memory $\boxed{\text{AMr}}$

(F4 = A, F5 = 1, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Form the logical product (AND), bit-by-bit, of the contents of the storage location specified by the effective address and the contents of register r. The result replaces the contents of the storage location specified by the effective address. The original contents of the storage location (specified by the effective address) replace the contents of register A. The contents of register r are not altered unless r is the A register. Memory is locked until completion of the instruction. This instruction is useful for communication between processors via memory.

7. Load register $\boxed{\text{LRr}}$

(F4 = C, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Load register r with the contents of the storage location specified by the effective address. The contents of storage are not altered.

8. Store Register $\boxed{\text{SRr}}$

(F4 = C, F5 = 1, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Store the contents of register r in the storage location specified by the effective address. The contents of register r are not altered.

9. Load Character to A $\boxed{\text{LCA}}$
(F4 = C, F5 = 2)

Load bits 0 through 7 of register A with a character from the storage location specified by the sum of the effective address and bits 1 through 15 of register Rb. Register Rb bit 0 is zero specifies the left character (bits 8 through 15) of the storage location; bit 0 = 1 specifies the right character (bits 0 through 7). Bits 8 through 15 of register A are not altered.

10. Store Character from A **SCA**
(F4 = C, F5 = 3)

Store the contents of bits 0 through 7 of register A into a character of the storage location specified by the sum of the effective address and bits 1 through 15 of register Rb. Bit 0 is zero of register Rb specifies the left character (bits 8 through 15) of the storage location; bit 0 is one specifies the right character (bits 0 through 7). The contents of register A and other storage characters are not altered.

11. OR Register **ORr**

(4 = D, F5 = 0, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Form the logical sum (inclusive OR), bit-by-bit, of the contents of the storage location specified by the effective address and the contents of register r. The result replaces the contents of register r. The contents of storage are not altered.

12. OR Memory **OMr**

(F4 = D, F5 = 1, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

Form the logical sum (inclusive OR), bit-by-bit, of the contents of the storage location specified by the effective address and the contents of register r. The result replaces the contents of the storage location specified by the effective address. The original contents of the storage location (specified by the effective address) replaces the contents of register A. The contents of register r are not altered unless r is the A register. Memory is locked until completion of the instruction. This instruction is useful for communication between processors via memory.

13. Compare Register Equal **CrE**

(F4 = E, F5 = 5, Rb = 1, 2, 3, 4, 5, 6, or 7)
(r = 1, 2, 3, 4, Q, A, or I)

If the contents of register r and the contents of the storage location specified by the effective address are equal, bit-by-bit, skip one location; otherwise, execute next instruction. The contents of register r and storage are not altered.

14. Compare Character Equal **CCE**
(F4 = E, F5 = 2)

If the contents of bits 0 through 7 of register A and the character of the storage location specified by the sum of the effective address and bits 1 through 15 of register Rb are equal, bit-by-bit, skip one location; otherwise, execute the next instruction. Bit 0 is zero of register Rb specifies the left character (bits 8 through 15) of the storage location; bit 0 is one specifies the right character (bits 0 through 7). The contents of register A and storage are not altered. Each compare instruction assumes that a one-word instruction follows it.

Field Reference Instructions

These instructions are described in the following paragraphs.

	15 12	11 8	7	6	5 3	2 0
	F = 0	F1 = 5	r	i	Ra	F3a
P + 1	FLDSTR	FLDLTH-1	Δ			
P + 2		16-bit address, if Δ= 0 Δ				

Field reference instructions are identified when the F field is zero, the F1 field is five, and the r, i, Ra, and F3a fields are not all zero. (If these fields are all zero, the instruction is an IIN.)

Field reference instructions contain four parts: operation field (F3a), addressing mode fields (delta, r, i, and Ra), FLDSTR, and FLDLTH-1 fields. The F3a field determines the operation (e.g., load, store, etc.). The addressing mode fields are defined exactly as the type 2 storage reference instruction.

FLDSTR defines the starting bit of the field: FLDSTR = 0 means the field starts at bit 0; FLDSTR = 15 means the field starts at bit 15. FLDLTH-1 defines the length of the field minus one: if FLDLTH-1 is zero the field is one bit long; if FLDLTH-1 is 15, the field is 16 bits long. Note that if FLDLTH-1 is zero the field reference instructions become bit reference instructions.

A field starts at the bit specified by FLDSTR and includes FLDLTH contiguous bits to the right of that bit. No field can cross a word boundary (i.e., FLDSTR-FLDLTH-1 must be greater than or equal to zero); e.g., if FLDSTR is zero, the field length must be one bit long (FLDLTH-1 is zero).

Note that F3a = 0, F3a = 1, and FLDSTR-FLDLTH-1 1 are reserved for future expansion.

1. Skip if Field Zero (F3a = 2) **SFZ**

   If the contents of the specified field of the storge location identified in the effective address are zero (all bits are zero), skip one location; if the contents are not zero execute the next instruction.

2. Skip if Field Not Zero (F3a = 3) **SFN**

   If the contents of the specified field of the storage location field identified in the effective address are not zero (not all bits are zero), skip one location, if zero, execute the next instruction.

NOTE

Each skip field instruction assumes that a one-word instruction follows it.

3. Load Field (F3a = 4) **LFA**

   Load register A, right justified, with the contents of the specified field of the storage location field identified in the effective address. All other bits of register A are cleared to zero. The contents of storage are not altered.

4. Store Field (F3a = 5) **SFA**

   Store the contents of the field from register a, right justified, into the specified field of the storage location identified in the effective address. All other storage bits are unchanged. Memory is locked until completion of the instruction. The contents of A are not altered.

5. Clear Field (F3a = 6) **CLF**

   Clear the specified field of the storage location specified by the effective address to all zeros. All other storage bits are unchanged. Memory is locked until completion of the instruction.

6. Set Field (F3a = 7) **SEF**

   Set the specified field of the storage location identified in the effective address to all ones. All other storage bits are unchanged. Memory is locked until completion of the instruction.

Type 2 Skip Instruction

These skip instructions are in the following format:

15 12	11 8	7 4	3 0
F = 0	F1 = 0	F2	SK

Type 2 skip instructions are identified when the F and F1 fields are both zero, and the F2 and SK fields are not both zero. (If these fields are both zero, the instruction is an SLS.)

Type 2 skip instructions (similar to type 1 skips, e.g., SAZ) contain two parts: operation field (F2) and skip count (SK). The F2 field determines the operation (i.e., skip on register 1, 2, 3, or 4 if zero, nonzero, positive, or negative). The skip count specifies how many locations to skip if the skip condition is met.

When the skip condition is met, the skip count plus one is added to the P register to obtain the address of the next instruction (e.g., when the skip count is one, go to P + 2). When the skip condition is not met, the address of the next instruction is P + 1 (skip count ignored). The skip count does not have a sign bit.

Note from above that if F2 is 0 (SrZ), the skip count cannot be zero because the instruction would be an SLS.

Instruction descriptions are as follows:

1. Skip if Register Zero (F2 = 0, 4, 8, or C) (r = 4, 1, 2, or 3) SrZ SK

   Skip if register r is a positive zero (all bits are zero).

2. Skip if Register Nonzero (F2 = 1, 5, 9 or D) (r = 4, 1, 2, or 3) SrN SK

   Skip if register r is not a positive zero (not all bits are zero).

3. Skip if Register Positive (F2 = 2, 6, A or E) (r =4, 1, 2, or 3) SrP SK

   Skip if register r is positive (bit 15 is zero).

4. Skip if Register Negative (F2 = 3, 7, B, or F) (r = 4, 1, 2, or 3) SrM SK

   Skip if register r is negative (bit 15 is one).

Decrement and Repeat Instructions

These instructions are in the following format:

15 12	11 8	7 5	4	3 0
F = 0	Fl = 6	Ra	0	SK

Decrement and repeat instructions are specified when the F field is zero, the Fl field is equal to six, bit 4 is zero, and the Ra and SK fields are both not zero. (If these fields are both zero, the instruction is a SPB.)

Decrement and repeat instructions contain two parts: Register field (Ra) and skip count (SK). The register field specifies which register is to be decremented by one and checked for the skip condition. The skip count specifies how many locations to repeat (go backwards) if the skip condition is met.

When the skip condition is met, the skip count is subtracted from the P register to obtain the address of the next instruction (e.g., when the skip count is one, go to P-1). When the skip condition is not met, the address of the next instruction is P + 1. The skip count does not have a sign bit.

Note that Ra = 0 and bit 4 = 1 are reserved for future expansion.

Instruction description is as follows:

1. Decrement and Repeat if Positive $\boxed{\text{DrP SK}}$

   (Ra = 1, 2, 3, 4, 5, 6, or 7) (r = 1, 2, 3, 4, Q, A, or I)

   Decrement by one the contents of register r. Operation on overflow is the same as for the ADD instruction. Repeat (go backwards) SK locations if the contents of register r are positive (bit 15 is zero); if otherwise, execute the next instruction.

Type 2 Interregister Instructions

These instructions are in the following format.

15 12	11 8	7 5	4 3	2 0
F = 0	F1 = 7	Ra	F2a	Rb

Type 2 interregister instructions are identified when the F field is zero, the F1 field is seven, and the F2a, Ra, and Rb fields are not all zero. (If these fields are all zero, the instruction is CPB.)

Type 2 interregister instruction (similar to type 1 interregister, e.g., TRA Q) contains three parts: operation field (F2a) and two register fields (Ra and Rb). The F2a field determines the operation (e.g., transfer). The Ra and Rb fields specify two operands.

Note that F2a = 1, F2a = 2, F2a = 3, Ra = 0, and Rb = 0 are reserved for future expansion.

Instruction description is as follows:

1. Transfer Register XFr R
   (F2a = 0, Ra = 1, 2, 3, 4, 5, 6, or 7)
   (r = 1, 2, 3, 4, Q, A, or I)

   Transfer the contents of register r to register R.

   Note that R = 1, 2, 3, 4, Q, A, or I, implies that Rb = 1, 2, 3, 4, 5, 6, or 7.

Miscellaneous Instructions

Miscellaneous instructions are in the following format.

15 12	11 8	7 5	4	3 0
F = 0	F1 = B	Ra	0	F3

Miscellaneous instructions are identified when the F field is zero, the F1 field is a decimal 11 (hex B) and bit 4 is zero, and the Ra and F3 fields are not both zero. (If these fields are both zero, the instruction is an NOP.) All of the miscellaneous instructions are privileged instructions, i.e., they cannot be executed by an unprotected program and cause a program protect violation instead.

Miscellaneous instructions contain two parts: Operation field (F3) and register field (Ra).

If Ra is not zero, the F3 operation field can select up to 16 miscellaneous instructions with register Ra used to specify an operand. If Ra is zero, the F3 operation field can select up to 15 more miscellaneous instructions without any explicit operand specified.

Instruction descriptions are as follows:

1. Load Micromemory (F3 = 1, Ra = 0) LMM

   Load a block of 32-bit micromemory words into read/write micromemory from 16-bit main memory.

   Initially, the Q register contains the number of 32-bit micromemory words to be transferred (if Q is 0, no words are be transferred). Register 1 contains the starting address of micromemory.

15 13	12 9	8 1	0
0 0 0	Page	Microaddress	U/L

   a. Bits 13 thru 15 must be zero.
   b. Bits 9 thru 12 specify the micropage.
   c. Bits 1 thru 8 specify the micromemory address.
   d. Bit 0 specifies the upper (0) or lower (1) microinstruction.

   Register 2 contains the starting address of main memory.

   This instruction is interruptible after storing each 32-bit micromemory word, and when registers 1, 2, and Q are incremented/decremented to allow the restarting of the instruction after any interruption. Therefore, upon completion of the instruction, these registers do not contain their original values, but do contain the following:

   Q ← 0
   R1 ← (R1)i+(Q)i
   R2 ← (R2)i+2*(Q)i

   Where i denotes the initial value before execution.

2. Load Registers (F3 = 2, Ra = 0) LRG

   Register 1, 2, 3, 4, Q, A, I, M, and the OVERFLOW indicator are loaded with the contents of nine storage locations, beginning at a storage location specified by the contents of the contents of the next location, P + 1 as follows:

```
                 (((P+1))+1)  →  1
                 (((P+1))+2)  →  2
                 (((P+1))+3)  →  3
                 (((P+1))+4)  →  4
                 (((P+1))+5)  →  Q
                 (((P+1))+6)  →  A
                 (((P+1))+7)  →  I
                 (((P+1))+8)  →  M
     bit 15 of   (((P+1))+9)  →  OVERFLOW
```

The contents of the storage location specified by the contents of the next location are then decremented by a decimal 10, that is:

```
((P+1))-10  →  (P+1)
```

Any data stored in location ((P + 1)) or bits 0 through 14 of location ((P + 1)) + 0 can be extracted before the execution of the LRG instruction via the address (specified by the contents of the contents of the next location).

The contents of the nine storage locations are not altered and the next instruction is executed at location P + 2 (i.e., the LRG instruction is a two-word instruction).

3. Store Registers (F3 = 3, Ra = 0) $\boxed{\text{SRG}}$

The contents of the storage location specified by the contents of the next location P + 1, is first incremented by a decimal 10, that is:

```
((P+1))+10  →  (P+1)
```

Then the contents of registers 1, 2, 3, 4, Q, A, I, and M, and the OVERFLOW indicator are stored in nine storage locations, beginning at a storage location specified by the contents of the incremented address as follows:

```
                    (1)    →   ((P+1))+1
                    (2)    →   ((P+1))+2
                    (3)    →   ((P+1))+3
                    (4)    →   ((P+1))+4
                    (Q)    →   ((P+1))+5
                    (A)    →   ((P+1))+6
                    (I)    →   ((P+1))+7
                    (M)    →   ((P+1))+8
     (OVERFLOW) bit is 15 of   ((P+1))+9
```

After the storing is completed, the OVERFLOW indicator is cleared.

Note that location ((P+1)) and bits 0 thru 14 of location ((P+1)+9 can be used to store a program address, priority level, parameter address, or other data after the execution of the SRG instruction via the address (specified by the contents of the next location).

The contents of the registers are not altered and the next instruction is executed at location P+2 (i.e., the SRG instruction is a two-word instruction).

4. Set/Sample Output or Input (F3 = 4, Ra = 0) SIO

For output, one word from register A is set (output) to an external device. The word in register Q selects the receiving device.

For input, one word from an external device is sampled (input) to register A. The word in register Q selects the sending device.

This instruction permits transmission to/from MO5 peripheral devices. The Q register is defined as follows:

15 11	10	9 7	6 4	3 1	0
0 0 0 0 0	1	Port	Pos	Mode	0

a. Bits 11 thru 15 and bit 0 must be zero.

b. Bits 7 thru 10 contain the port number of the device, with bit 10 always a one. Port numbers are analogous to the AW I/O equipment numbers and thus cannot conflict with them.

c. Bits 4, 5, and 6 designate the device's position on the port. These bits can also be mode bits (see below) if some or all of the position bits are not required.

d. Bits 1, 2, and 3 contain the mode in which the device is to operate. Bit 3 is always the set/sample condition bit; if one, one data word is set (output); if zero, one data word is sampled (input).

5. Sample Position/Status (F3 = 5, Ra = 0) SPS

Sample (input) to the A register the position and the status of a MO5 device, which has caused a macrointerrupt. The word in the Q register selects the device. This instruction also provides for the clearing of the MO5 generated macrointerrupt. The Q register is defined as follows:

15 11	10	9 7	6 0
0 0 0 0 0	1	Port	0 0 0 0 0 0 0

a. Bits 0 thru 6 and 11 through 15 must be zero.

b. Bits 7 thru 10 contain the port number of the MO5 device, with bit 10 always a one. Port numbers are analogous to the AW I/O equipment numbers and thus cannot conflict with them.

Upon completion of the instruction, the A register contains the following:

15 12	11 8	7 5	4 2	1	0
0 0 0 0	Status	0 0 0	Pos	0	0

a. Bits 0 and 1, 5, 6, and 7, and 12 thru 15 are zero.

b. Bits 2, 3, and 4 contain the device's position.

c. Bits 8 thru 11 contain the least significant 4 bits of the data input lines. If these 4 bits of status information are insufficient, the device's controller can provide for the transfer of additional status bits via the SIO instruction.

6. Define Microinterrupt (F3 = 6, Ra = 0) DMI

Defines the use of one of the 12 available microinterrupts. (The use of microinterrupts 12 thru 15 are restricted for internal use.)

This instruction allows the enabling/disabling of a microinterrupt and defining it for Autodata Transfer (ADT) or special usage.

The Q register selects and enables/disables the microinterrupt, which is defined as follows:

15	14 4	3 0
X	0 0 0 0 0 0 0 0 0 0 0	Microinterrupt

a. Bit 15 enables/disables the microinterrupt; if one, the microinterrupt is enabled; if zero, the microinterrupt is disabled; the A register is not utilized. Initially, all 12 microinterrupts are disabled.

b. Bits 4 thru 14 must be zero.

c. Bits 0 thru 3 contain the microinterrupt number (0 thru 11). Note that 12 thru 15 are not available for use and, if used, the instruction is treated as an NOP.

The A register defines the microinterrupt for ADT or special usage and is defined as follows:

15	14 0
X	ADT Table or Page Micromemory Address

a. If bit 15 is a zero, ADT is selected and bits 0 thru 14 contain the ADT Table Address. It must be within the lower 32K of main memory.

Four types of ADT Tables are possible for a particular microinterrupt:

(1) Single AQ device
(2) Multi AQ devices
(3) Clock device
(4) Single or Multi MO5 devices

b. If bit 15 is a one, the special usage is selected and a jump is made to the upper microinstruction of the page micromemory in bits 0 thru 14. A section of micromemory is assumed to have been previously loaded, and it must process the microinterrupt properly and return control to the current macroinstruction address (P) by jumping to the lower microinstruction of micromemory address $3E_{16}$ in micropage zero. Registers P, A, Q, and all file 2 should not be altered, and return must be within 12.5 microseconds.

NOTE

The processor microfunction SUB- must not be used.

Bits 0 thru 15 of the A register are defined as follows for this special usage:

15 12	11 8	7 0
1 0 0 0	Page	Micromemory Address

a. Bit 15 must be one.
b. Bit 12 must be zero.
c. Bits 8 thru 11 specify the micropage.
d. Bits 0 thru 7 specify the micromemory address.

NOTE

Extreme caution should be exercised in using this option, since it provides an escape from the 1700 emulation being performed.

7. Clear Breakpoint Interrupt (F3 = 7, Ra = 0) $\boxed{\text{CBP}}$

Clear the processor macrobreakpoint interrupt. This interrupt occurs when the following conditions are true.

a. Macrobreakpoint (reference and/or storage) is externally selected.

b. Macrobreakpoint interrupt option is externally selected.

c. The MP recognizes a breakpoint condition and generates a macrobreakpoint interrupt because of b.

The macroprogrammer then has the responsibility of clearing (CBP instruction) and processing the interrupt.

8. Generate Character Parity Even (F3 = 8), Ra = 0) GPE

Set or clear bit 7 of the A register to cause the parity of bits 0 thru 7 to be even. The rest of the contents of the A register are not altered.

9. Generate Character Parity Odd (F3 = 9, Ra = 0) GPO

Set or clear bit 7 of the A register to cause the parity of bits 0 thru 7 to be odd. The rest of the contents of the A register are not altered.

10. Scale Accumulator (F3 = A, Ra = 0) ASC

The A register is shifted left (end-around) until bits 14 and 15 of the A register are different. Upon completion of the instruction, register contains the number of places that the A register was shifted. (This number can range from zero through 14_{10}.) If the A register is $\pm$ zero (0000 or FFFF), no shifting is done and register 1 contains minus zero (FFFF).

11. Load Upper Unprotected Bounds LUB R

(F3 = 0, Ra = 1, 2, 3, 4, 5, 6, or 7) (R = 1, 2, 3, 4, Q, A, or I)

Load the upper unprotected bounds register from the contents of register R.

12. Load Lower Unprotected Bounds LLB R

(F3 = 1, Ra = 1, 2, 3, 4, 5, 6, or 7) (R = 1, 2, 3, 4, Q, A, or I)

Load the lower unprotected bounds from the contents of register R.

13. Execute Microsequence EMS R

(F3 = 2, Ra = 1, 2, 3, 4, 5, 6, or 7) (R = 1, 2, 3, 4, Q, A, or I)

Transfer machine control to the upper microinstruction of the page/micromemory address in bits 0 thru 15 of register R. A section of micromemory is assumed to have been previously loaded and it should return control to the next macroinstruction address (P + 1) by jumping to the lower microinstruction of micromemory address $3E_{16}$ in micropage zero.

Registers P, A, Q, and all of file 2 should not be altered, and return must be within 12.5 microseconds (or the microsequence must be interruptible). Bits 0 thru 15 of register R are defined as follows:

15 12	11 8	7 0
0 0 0 0	Page	Micromemory Address

a. Bits 12 thru 15 must be zero.
b. Bits 8 thru 11 specify the micropage.
c. Bits 0 thru 7 specify the micromemory address.

NOTE

Extreme caution should be exercised in using this instruction, since it provides escape from 1700 emulation.

Autodata Transfer

Autodata Transfer (ADT) provides for pseudo-direct memory transfers of data blocks to or from a device. At the macrolevel, the transfer appears as direct memory/storage access (DMA) transfer; however, at the microlevel, the 1700 emulator processes each data interrupt and inputs or outputs the next datum in a singular fashion. Thus, ADT takes less time than input/out via the INP, OUT, or SIO instructions but more time than a true DMA transfer.

To accomplish ADT for a particular device, perform the following:

1. The device and its controller must adhere to the Autodata Transfer specifications.

2. The macroprogrammer must execute a DMI instruction. This command specifies where the block of data is, how long it is, the direction (input/output), and the device address.

3. The ADT operation is then initiated by an INP, OUT, or SIO instruction as specified by the particular device.

While the ADT operation is in progress, the emulator is executing instructions. After each instruction is executed, interrupts are checked. When the particular ADT microinterrupt becomes the highest active interrupt, the next datum is input or output. After the interruption, the next instruction is executed, except when another interrupt is active.

When the ADT operation is completed (or if there is an error), a macrointerrupt is generated. The macroprogrammer can than disable the ADT microinterrupt or initiate another ADT operation to or from the device.

The following defines the four types of ADT tables specified by DMI instructions:

1. ADT Table for a Single AQ Device

15	14	13	12	11	10 7	6 0	Word
0	0	W/C	0	R/W	Equip	Station Director	1
FWA-1 and CWA							2
LWA							3
Not used							4

The ADT table for this type consists of four words:

a. In word 1, bits 12, 14, and 15 must be zero.

b. In word 1, bit 13 is zero if a word operation, is one if a character operation. For word operation, data is transferred one word at a time. Normally, a total of (CWA - FWA + 1) words are transferred.

   For character operation, data is transferred one character (8 bits) at a time. On input, the first character is stored in the most significant half (bits 8 thru 15) of the current word address; the second character in the least significant half (bits 0 thru 7). Subsequent pairs of characters are input in the same fashion. For output, the first character is output from the most significant half (bits 8 thru 15) of the current word address; the second character from the least significant half (bits 0 thru 7). Likewise, subsequent pairs of characters are output in the same way. Normally, a total of 2 x (CWA - FWA + 1) characters are transferred.

c. In word 1, bit 11 is zero if a read ADT operation and one if a write ADT operation.

d. In word 1, bits 7 thru 10 contain the equipment number of the device.

e. In word 1, bits 0 thru 6 contain the station/director bits of the device to execute the ADT operation. The director bits should specify a data (not a status/function) transfer.

f. Word 2 is initially set to the first word address - 1 (FWA-1) of the data block to be transferred. This word is used as the current word address (CWA) as the ADT operation is in progress and points to the last data word read or stored. Each time a word (or two characters if a character operation) is transferred, CWA is incremented. Specifically, CWA can be used to ascertain whether all the ADT operation was completed (i.e., if CWA = LWA, all the data has been transferred).

g. Word 3 contains the last word address (LWA) of the data block to be transferred.

h. Word 4 is not currently used and should be zero. It is reserved for future use.

2. ADT Table for Multiple AQ Devices. See figure 4-6.

   The ADT table for this type consists of I*4+4 words, where I is the number of multiple AQ devices, up to 32, on one microinterrupt:

   a. In word 1, bit 15, bits 11 thru 13, and bit 0 must be zero.

   b. In word 1, bits 14 and 1 must be one.

   c. In word 1, bits 7 thru 10 contain the equipment number of the device.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
0	1	0	0	0	Equip				Max Station					1	0	1
T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0	2
T_{31}	T_{30}	T_{29}	T_{28}	T_{27}	T_{26}	T_{25}	T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	3
Not used																4
0	0	W/C	0	R/W	Equip				Station Director							5
FWA-1 and CWA																6
LWA																7
Not used																8
•																•
•																•
•																•
0	0	W/C	0	R/W	Equip				Station Director							I*4+1
FWA-1 and CWA																I*4+2
LWA																I*4+3
Not used																I*4+4

Figure 4-6. ADT Table for Multiple AQ Devices

d. In word 1, bits 2 thru 6 contain the maximum station (or channel) number. This is equivalent to the number of multiple AQ devices, minus one, on a microinterrupt. Note that the station numbers must be contiguous (i.e., 0 to I-1).

e. Words 2 and 3 contain termination bits for the 32 devices. Initially, they must be all zero. When a macrointerrupt occurs, one or more of these bits are set to one to indicate that one or more ADT operations have terminated. Thus, if T_7 is 1, the seventh device has terminated its ADT operation. Note that after receipt, the bit should be cleared via an instruction that locks memory (e.g., a CLF instruction).

f. Word 4 is not currently used and should be zero. It is reserved for future use.

g. Words 5, 6, 7, and 8 (and also words I*4+1, I*4+2, I*4+3, and I*4+4, where $2 \leq I \leq 32$) are defined the same as a single AQ device except bit 14 of the first word I*4+1 must be a one. (Refer to ADT Table for a Single AQ Device).

NOTE

The Standard 2550-2 does not contain any multiple AQ devices.

3. ADT Table for the Clock

15	14	13	12	11	10 … 7	6 … 0	Word
1	0	0	0	0	Equip	Station Director	1
Clock Counter							2
Clock Limit							3
Not used							4

The ADT table for this type consists of four words:

a. In word 1, bit 15 must be one.

b. In word 1, bits 11 through 14 must be zero.

c. In word 1, bits 7 through 10 contain the equipment of the clock, which is always one.

d. In word 1, bits 0 through 6 contain the station/director bits of the clock, which is always equal to 70_{16}. (Thus, word 1 should be $80F0_{16}$.)

e. Word 2 is initially set to zero. Whenever the clock has been enabled, the clock counter is incremented every 3-1/3 milliseconds.

f. Word 3 contains the clock limit, which is interpreted as a multiple of 3-1/2 milliseconds. When the clock counter equals the clock limit and the macroclock interrupt is enabled, the macroclock interrupt occurs. Thus, if the clock limit is five, the clock interrupt is 16-2/3 milliseconds, or 60 times a second. To continue the process, the clock counter should be reset to zero, or the limit counter should be incremented by its original value (e.g., five). In the later method, the clock counter can function as an elapsed time counter. Note that if the macroclock interrupt is not answered, the clock limit continues to be incremented.

g. Word 4 is not currently used and should be zero. It is reserved for future use.

MACROINSTRUCTION TIMING

The execution times for each macroinstruction operation are listed in table 4-4.

MICROPROGRAMMING USAGE

In the NPU, two microprograms are co-resident. A 1700 emulator resides in read-only memory and directs the execution of macrocode 1700-type instructions from main memory. Special microcode, loaded in read/write memory, supports portions of the multiplex subsystem and other time-critical functions. Communication between the routine is via internal interrupts.

TABLE 4-4. MACROINSTRUCTION EXECUTION TIMES

Mnemonic	Definition	Execution (Microsec)	Op Code			
		See Note				
AAB	Transfer Arithmetic Sum A, Q, M	1.2 - 2.2	0	8	3	8-F
AAM	Transfer Arithmetic Sum A, M	1.2 - 2.2	0	8	2	8-F
AAQ	Transfer Arithmetic Sum A, Q	1.1 - 1.5	0	8	3	0-7
ADD	ADD A	1.6	8	*	△	△
ADQ	ADD Q	1.6	F	*	△	△
ALS	A Left Shift	1.6+0.06N	0	F	C/D	0-F
AMr	AND Memory		0	4	*	*
			A	1	△	△
AND	AND with A	1.5	A	*	△	△
ANr	AND Register		0	4	*	*
			A	0	△	△
ARr	Add Register		0	4	*	*
			8	0	△	△
ARS	A Right Shift	1.6+0.06N	0	F	4/5	0-F
ASC	Accumulator Scale		0	B	0	A
CAB	Transfer Complement Logical Product A, Q + M	1.7 - 2.2	0	8	F	8-F
CAM	Transfer Complement Logical Product A, M	1.7 - 2.2	0	8	E	8-F
CAQ	Transfer Complement Logical Product A, Q	1.1 - 1.5	0	8	F	0-7
CBP	Clear Breakpoint Interrupt		0	B	0	7
CCE	Compare Character Equal		0	4	*	*
			E	2	△	△
CLF	Clear Field		0	5	+	6
			#	#	△	△
CLR	Clear to Zero	1.1 - 1.5	0	8	4	0-7
CPB	Clear Program Protect	1.6	0	7	0	0
CrE	Compare Register Equal		0	4	*	*
			E	0	△	△
DMI	Define Microinterrupt	1.3	0	B	0	6
DrP	Decrement and Repeat		0	6	2,4,6,8, A,C,S,E	
DVI	Divide Integer	9.4 - 10.6	3	*	△	△
EAB	Transfer Exclusive OR A, Q, M	1.7 - 2.2	0	8	7	8-F
EAM	Transfer Exclusive OR A, M	1.7 - 2.2	0	8	6	8-F
EAQ	Transfer Exclusive OR A, Q	1.7 - 2.2	0	8	7	0-7

TABLE 4-4. MACROINSTRUCTION EXECUTION TIMES (Contd)

Mnemonic	Definition	Execution (Microsec)	Op Code			
		See Note				
EIN	Enable Interrupt	1.4	0	4	0	0
EMS	Execute Microsequence		0	B	r,o	2
ENA	Enter A	0.9	0	A	△	△
ENQ	Enter Q	0.9	0	C	△	△
EOR	Exclusive OR with A	1.5	B	*	△	△
EXI	Exit Interrupt State	1.9	0	E	△	△
GPE	Generate Character Parity Even		0	B	0	8
GPO	Generate Character Parity Odd		0	B	0	9
IIN	Inhibit Interrupt	1.4	0	5	0	0
INA	Increase A	0.9	0	9	△	△
INP	Input to A	2.8 - 9.9	0	2	△	△
INQ	Increase Q	0.9	0	D	△	△
JMP	Jump	1.1	1	*	△	△
LAB	Transfer Logical Product A, Q+M	1.7 - 2.2	0	8	B	8-F
LAM	Transfer Logical Product A, M	1.1 - 1.5	0	8	A	8-F
LAQ	Transfer Logical Product A, Q	1.1 - 1.5	0	8	B	0-7
LCA	Load Character to A		0	4	*	*
			C	2	△	△
LDA	Load A	1.5	C	*	△	△
LDQ	Load Q	1.5	E	*	△	△
LFA	Load Field		0	5	*	4/0
			*	*	△	△
LLB	Load Lower Unprotected Bounds		0	B	4,o	1
LLS	Long Left Shift	2.2+0.06N	0	F	E/F	0-F
LMM	Load Micromemory		0	B	0	1
LRG	Load Register		0	B	0	2
LRr	Load Register		0	4	*	*
			C	0	△	△
LRS	Load Right Shift	2.2+0.06N	0	F	6/7	0-F
LUB	Load Upper Unprotected Bounds		0	B	r,o	0
MUI	Multiply Integer	5.4 - 7-4	2	*	△	△
NOP	No operation	1.1	0	F	0/1	0-F
OMr	OR Memory		0	4	*	*
			D	1	△	△

TABLE 4-4. MACROINSTRUCTION EXECUTION TIMES (Contd)

Mnemonic	Definition	Execution (Microsec)	Op Code			
		See Note				
ORr	OR register		0	4	*	*
			D	1	Δ	Δ
OUT	Output from A	2.4 - 9.5	0	3	Δ	Δ
QLS	Q Left Shift	1.9+0.06N	0	F	A/B	0-F
QRS	Q Right Shift	1.9+0.06N	0	F	2/3	0-F
RAO	Replace Add 1 in Storage	1.9	D	*	Δ	Δ
RTJ	Return Jump	1.6	5	*	Δ	Δ
SAM	Skip if A = -	1.1 / 1.4*	0	1	3	S
SAN	Skip if A ≠ 0	1.1 / 1.4*	0	1	1	S
SAP	Skip if A = +	1.1 / 1.4*	0	1	2	S
SAZ	Skip if A = +0	1.1 / 1.4*	0	1	0	S
SBr	Subtract Register		0	4	*	*
			9	0	Δ	Δ
SCA	Store character from A		0	4	*	*
			C	3	Δ	Δ
SEF	Set Field		0	5	*	7/F
			*	*	Δ	Δ
SET	Set to 1's	1.1 - 1.5	0	8	8	0-7
SFA	Store Field		0	5	*	5/D
			*	*	Δ	Δ
SFN	Skip if Field Not Zero	1.1 / 1.4*	0	5	*	3/B
			*	*	Δ	Δ
SFZ	Skip if Field Zero	1.1 / 1.4*	0	5	*	2/A
			*	*	Δ	Δ
SIO	Set/Sample Output or Input		0	B	0	4
SJE	Subroutine Jump Exit		0	4	*	*
			5	0	Δ	Δ
SJr	Subroutine Jump		0	4	*	*
			5	0	Δ	Δ
SLS	Select Stop	1.3 / 1.5**	0	0	0	0
SNF	Skip on No Program Protect Fault	1.1 / 1.4*	0	1	B	S
SNO	Skip on No Overflow	1.1 / 1.4*	0	1	F	S

TABLE 4-4. MACROINSTRUCTION EXECUTION TIMES (Contd)

Mnemonic	Definition	Execution (Microsec)	Op Code			
		See Note				
SNP	Skip on No Storage Parity Error	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	D	S
SOV	Skip on Overflow	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	A	S
SPA	Store A, Parity to A	2.1	7	*	△	△
SPB	Set Program Protect	1.6	0	6	0	0
SPE	Skip on Storage Parity Error	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	C	S
SPF	Skip on Program Protect Fault	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	E	S
SPS	Sample Position Status		0	B	0	5
SQM	Skip if Q = -	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	7	S
SQN	Skip if Q ≠ +0	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	5	S
SQP	Skip if Q = +	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	6	S
SQZ	Skip if Q = +0	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	4	S
SRG	Store Registers		0	B	0	3
SrM	Skip if Register Negative	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	0	3,7 B,F	S
SrN	Skip if Register Non Zero	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	0	1,5 9,D	S
SrP	Skip if Register Positive	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	0	2,6 A,E	S
SRr	Store Registers		0 C	4 1	* △	* △
SrZ	Skip if Register Zero	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	0	0,4 8,C	S
STA	Store A	1.6	6	*	△	△
STQ	Store Q	1.6	4	*	△	△
SUB	Subtract	1.6	9	*	△	△
SWN	Skip if Switch Not Set	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	9	S
SWS	Skip if Switch Set	$\left\{\begin{matrix}1.1\\1.4*\end{matrix}\right\}$	0	1	8	S
TCA	Transfer Complement A	1.1 - 1.5	0	8	6	0-7

TABLE 4-4. MACROINSTRUCTION EXECUTION TIMES (Contd)

Mnemonic	Definition	Execution (Microsec)	Op Code			
		See Note				
TCB	Transfer Complement Q + M	1.2 - 1.7	0	8	5	8-F
TCM	Transfer Complement M	1.2 - 1.7	0	8	4	8-F
TCQ	Transfer complement Q	1.1 - 1.5	0	8	5	0-7
TRA	Transfer A	1.1 - 1.5	0	8	A	0-7
TRB	Transfer Q + M	1.2 - 1.7	0	8	9	8-F
TRM	Transfer M	1.2 - 1.7	0	8	8	8-F
TRQ	Transfer Q	1.1 - 1.5	0	8	9	0-7
XFr	Transfer Register		0	7	1-7,0,1-7	

*If skip is taken
**If selective stop set

NOTE: Add 550 ns (approx) per memory reference for each instruction:

```
CPU = 750 ns Read                CPU/Exp Avg = 425 ns
      850 ns Write                 Setup      +155 ns
Avg = 800 ns                                   580 ns

Exp = 1145 ns Read
      1305 ns Write
Avg = 1225 ns
```

MICROINSTRUCTION FORMATS

Microinstruction formats consist of 32-bit instruction words which are divided into five main sections numbered left to right as bits 0 to 31. The total microinstruction word appears as follows:

0 1	2 15	16 18	19 23	24 31
M	ALU Control	T	S	C

Bits	Function
0,1	Mode (M) field specifies format of S and C field and sequencing mode to obtain next microinstruction pair.
2-15	ALU control field specifies ALU operation, sources of operands, and destination of result of operation.
16-18	Test (T) field specifies method of selecting which microinstruction of next microinstruction pair to execute.

19-23	Special (S) field specifies subformat selection and special operation.
24-31	Constant (C) or suboperation field specifies constants, or other codes.

The M field specifies one of three addressing modes to be used to obtain the next microinstruction pair from micromemory and specifies the format to be used in interpreting bits 19 through 31 of the microinstruction as follows:

M	Addressing Mode	Format for bits 19 to 31
00	Return	Format 1
01	Sequential	Format 1
10	Jump	Format 2
11	Sequential	Format 3

Figure 4-7 shows all microinstruction formats. Note that format 1 and format 2 microinstructions have two subformats, which are selected by the value of bit 19 (SF).

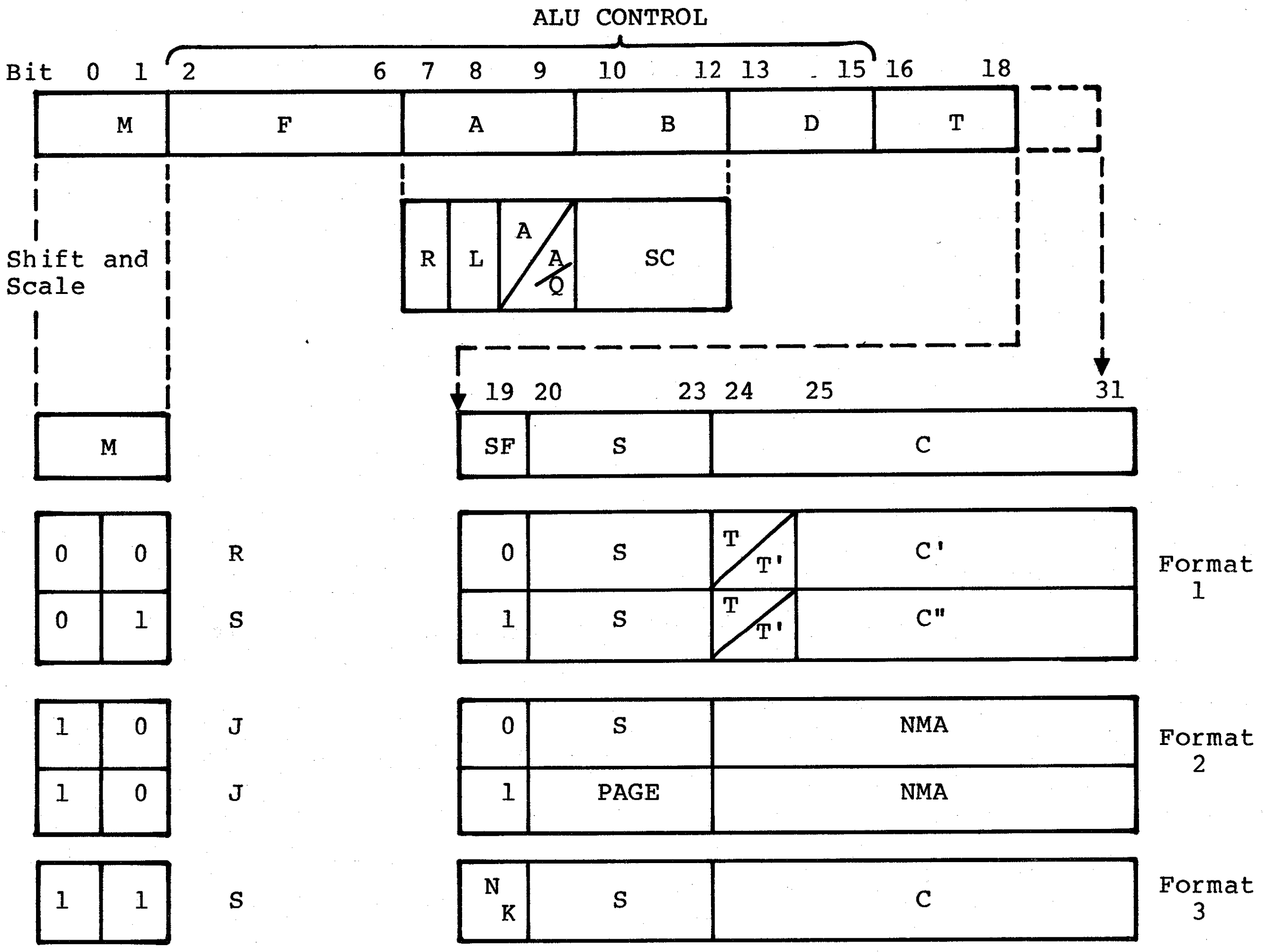
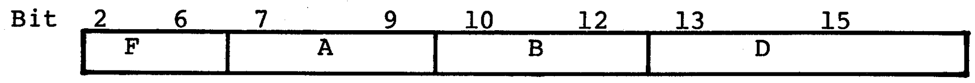
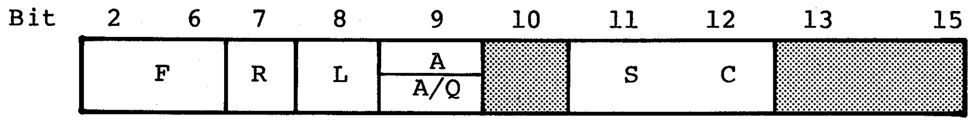


Figure 4-7. Microinstruction Formats

The ALU control fields specify the sources of two operands on which an arithmetic, logical shift, or scale operation is to be performed and specify the destination of the result of the operation. For arithmetic and logical operations, the ALU control fields consist of the ALU function (F), A source (A), B source (B), and destination (D) fields, shown as follows:



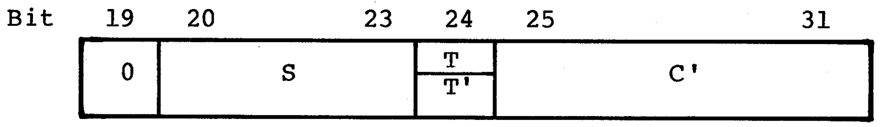
For shift and scale operations, the A and B fields are interpreted as follows:



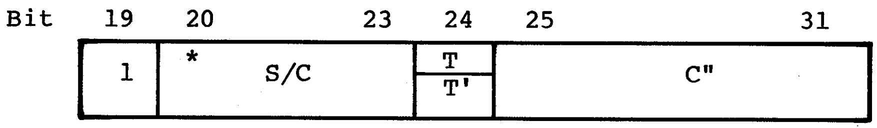
The F field specifies shift or scale operation. Bits 7 and 8 specify right or left shifting. Bit 9 specifies whether the A register alone or the A and Q registers together are to be shifted or scaled. Bit 10 is not used, and bits 11 and 12 specify the shift control code. The D field contains a no-operation code for shift and scale operations.

The T field is the conditional branch of the microinstruction and specifies which microinstruction, upper or lower, of the next microinstruction pair to execute. The test can be based on the result of the ALU operation of the current microinstruction or on some other condition.

The codings in the S and C fields depend upon the contents of the M field. The S and C fields are coded in three formats. Format 1 is specified when the M field contains 00 (return mode) or 01 (sequential mode) as follows:



Format 1



*If SM207 is set to a 1, the normal S field commands are disabled with C" codes TMA/j, TMAK/j, GITMAK/j, GITMAK/xt, and TK/j.

The S field specifies operations such as main memory read or write operations; alternate codings to be used in the A, B, and D fields; etc. The T/T' bit specifies that the code in the T field is to be interpreted as the normal T code (T/T'=0) or as the alternate T' code (T/T'=1). Subformat select bit 19 determines whether bits 25 through 31 are to be interpreted as C' codes or as C" codes. The C' code can contain the constant for driving the bit generator, additional information to control the main memory read or write, or other operation. The C" codes are associated with transforms.

Format 2 is specified when the M field contains 10 (jump mode) as follows:

Bit 19	20 23	24 31
0	S	NMA

Format 2

Bit 19	20 23	24 31
1	PAGE	NMA

If a jump is specified to a microinstruction pair within the same micro-instruction page, the subformat select bit is 0; bits 20 to 23 contain a special operation code as in format 1. Bits 24 through 31 contain the micromemory address of the next microinstruction address of the next microinstruction pair. The subformat select bit is 1 when a jump is specified to a different micromemory page; bits 20 through 31 contain the complete micromemory address of the next microinstruction pair.

Format 3 is specified when the M field is 11 (sequential mode), as follows:

Bit 19	20 23	24 31
N/K	S	C

Format 3

This format allows one special operation to be performed as specified by the S code and also causes the 8 bits of the C field to be transferred to the N register (bit 19 is 1) or to the K register (bit 19 is 0).

DETAILED MICROINSTRUCTION CODING

The following text describes the fields of the microinstruction words.

M Field (Bits 0 and 1)

The field defines the major operation taking place in the microinstruction and specifies the type of sequencing to be used to obtain the next instruction pair. The operations specified in the M field are listed in table 4-5.

F Field (Bits 2 through 6)

The F field specifies the logical or arithmetic operation to be performed by the ALU, or the shift or scale operations performed with the A and Q registers.

The split adder option allows the main ALU to be split into two independent adders. This split is activated by setting the adder split flag in the SM register. The split blocks the carry between the two portions of the adder. The upper portion of the adder always functions as a two's complement adder; the lower portion can function as a one's complement or as a two's complement adder, depending upon the state of the one's complement SM register flag

(32-bit processor only). In one's complement mode, the carryout of the lower portion is used as the end-around carry bit. In two's complement mode, both portions of the adder act as independent two's complement adders. The split adder has no effect on logical operations because no carry is involved in these operations.

Logical Operations

The logical operations perform bit-by-bit combinations of the A input and the B input for delivery to the destination. The logical operations are described in table 4-6.

Arthmetic Operations

The arithmetic operations (table 4-7) operate on single-precision (using the main ALU).

The second option allows capture of the overflow condition in the SM register (indicated by a T in the instruction mnemonic). If this is indicated, the overflow test is performed by checking the sign of the two inputs to the ALU and by setting a status/mode bit if the result is inconsistent. The SM overflow bit is set to 1 when the overflow occurs; it must be set to 0 by a microinstruction which sets the SM bit to 0. Two other optional SM bits allow checking for binary or decimal arithmetic overflow.

TABLE 4-5. M FIELD OPERATIONS

M Code	Mnemonic	Operation
00	R	Select next microinstruction pair from address contained in RTJ register. Use format 1 for special operations.
01	S	Select next microinstruction pair in current page from address contained in MAC (normally next sequential pair, unless suppressed by T field coding). Use format 1 for special operations.
10	J	A jump or page jump. Select next microinstruction pair from address specified by bits 24 to 31 of this microinstruction. Address is in current MM page if bit 19 is 0 or from page specified in bits 20 through 23 if bit 19 is 1. Use format 2 for special operations.
11	S	Transfer bits 24 through 31 of this microinstruction to N or K register as specified by bit 19 of this microinstruction. N register is specified if bit 19 is 1, and K register is specified if bit 19 is 0. Select next microinstruction pair in current page from address contained in MAC (normally next sequential microinstruction pair). Use format 3 for special operations.

TABLE 4-6. LOGICAL OPERATIONS

		A input = 0 0 1 1, B input = 0 1 0 1
F Code	Mnemonic	Bit Result
01100	ZERO	0 0 0 0
01110	A B	0 0 0 1
01101	A -B	0 0 1 0
01111	A	0 0 1 1
01000	-A B	0 1 0 0
01010	B	0 1 0 1
01001	EOR	0 1 1 0
01011	A+B	0 1 1 1
00100	-A -B	1 0 0 0
00110	-EOR	1 0 0 1
00101	-B	1 0 1 0
00111	A+-B	1 0 1 1
00000	-A	1 1 0 0
00010	-A+B	1 1 0 1
00001	-A+-B	1 1 1 0
00011	ONE	1 1 1 1

Shift Operations

The shift operations in the F field specify a shift of the A register or the A/Q register of the main MP organization only; no shift is possible in the double-precision registers from this command. The ALU is not used to perform the shift but performs some operation based on its decoding of the F field (which should be considered as unknown). The destination receives this meaningless output unless a NOP is chosen for the destination of the D field.

The type of shift is determined by the coding in bits 7 through 12 of the microinstruction, and the amount of the shift is determined by the number contained in the N register. The operation examines the register, and, if it is zero, the shift operation is terminated and the next microinstruction is executed. The T field codes that can be used with a shift are U, L, KZU, INTU, OVFL, K7L, and BTU. Other T codes cannot be used.

If the N register is not zero, a shift of one bit position is taken as specified, N is decremented by one, and the test for zero is repeated as above.

The shift conditions are:

- Shift A - Shift the A register only.
- Shift AQ - Shift the combined A and Q register. The Q register is considered to be the least significant bits of the combined A/Q register.
- Shift external (transform).
- Shift left or right.

- Enter 0 - Enter a zero in the vacated bit position at the end of the register.
- Enter 1 - Enter a one in the vacated bit position at the end of the register.
- Extend sign - Extend the sign (for a right shift only).
- End-around carry - Enter the bit coming off the end of the register into the vacated one position at the other end.

All shifts are performed with an F code of 11110. The type of shift is determined by bits 7 through 12 of the microinstruction. The shifts are defined in table 4-8.

TABLE 4-7. ARITHMETIC OPERATIONS

F Code	Appli-cable Notes	Mnemonic	Operation
10110		SUB	Subtract B input from A input.
11000		ADD	Add A and B inputs.
10101		SUBT	Subtract with an overflow test.
11001		ADDT	Add with an overflow test.
10110	1,3	SUB-	Perform A-B-1. See applicable notes.
	2,3	SUB-C	A-B with force carry in. See applicable notes.
11010	3	ADD+	Perform A + B with force carry in. See applicable notes.
10111	1,3	SUB-T	Perform SUB- with an overflow test. Set applicable notes.
	2,3	SUB-TC	Perform SUB-C with an overflow test. See applicable notes.
11011	3	ADD+T	Perform ADD+ with an overflow test. See applicable notes.

NOTES:

1. With this F code specified and when the system is in two's complement mode, this is the resultant arithmetic operation that occurs.
2. With this F code specified and when the system is in one's complement mode, this is the resultant arithmetic operation that occurs.
3. If split adder mode is selected, this operation is performed on the upper adder without a force carry.

TABLE 4-8. SHIFT OPERATIONS

Bit Code		Mnemonic	Operations
7 8 9	11 12		
1 0 0	0 0	AR0E	A is right shifted (N) bits, with 0 entered as most significant bit.
1 0 0	0 1	ARSE	A is right shifted (N) bits, with sign extension.
1 0 0	1 0	AREA	A is right shifted (N) bits, with end-around carry.
0 1 0	0 0	AL0E	A is left shifted (N) bits, with 0 entered as least significant bit.
0 1 0	0 1	AL1E	A is left shifted (N) bits, with 1 entered as least significant bit.
0 1 0	1 0	ALEA	A is left shifted (N) bits, with end-around carry.
1 0 1	0 0	AQR0E	A/Q is right shifted (N) bits, with 0 entered as most significant bit in A.
1 0 1	0 1	AQRSE	A/Q is right shifted (N) bits, with sign extension.
1 0 1	1 0	AQREA	A/Q is right shifted (N) bits, with end-around carry.
0 1 1	0 0	AQL0E	A/Q is left shifted (N) bits, with 0 entered at least significant bit in Q.
0 1 1	1 0	AQLEA	A/Q is left shifted (N) bits, with end-around carry.

Scale Operations

The scale operations are similar to the shift operations but the stopping of the shift is conditioned on bits 0 and 1 of A not being equal. (The scale point is normally between bits 0 and 1 of the A register. A design option allows the scale point to be specified between different bits in the A register if necessary for efficient floating-point emulation.) The maximum number of bits to scale is contained in the N register, and, on completion of the scale, N is decremented by the number of shifts which were necessary to scale the number.

The scale operation is performed as follows:

1. Examine N; if it is zero, exit the microinstruction.

2. Examine bits 0 and 1 of the A register; if they differ, exit the microinstruction.

3. Shift the A or A/Q register left by one bit position as specified in the instruction.

4. Decrement the N register by one count and go to step 2.

NOTE

If the number that is being scaled is all zeros or all ones (i.e., the number cannot be scaled), then when N=$FE (after passing through N+0), the scale operation is terminated. In order to avoid exiting the micro-instruction before the scale operation is actually completed, (N) should be at least equal to the number of bits in the word to be scaled.

The scale operation is coded the same in bits 7 through 12 of the microinstruction and allows the same left shift options as the shift command. The same comments on exiting the shift and the usable T field codes apply to the scale operation.

All scale operations are performed with an F code of 11111. The type of shift for the scale is determined by bits 7 through 12 of the instruction. The scales are given in table 4-9.

A Field (Bits 7 through 9)

The field specifies the input to S1 and thus to the A side of the ALU. The eight A codes are expanded by eight A' codes by placing 1010 or 0111 in the S field.

The eight A inputs to S1 and to the A side of the ALU are indicated when the S field is not 0111 or 1010; the eight A codes specify inputs from the files, the registers, or main memory, according to table 4-10.

The eight A' inputs to S1 and to the A side of the ALU are indicated if the S field is 0111 or 1010. The A' codes specify input from the SM registers or mask register. The A' codes are given in table 4-11.

TABLE 4-9. SCALE OPERATIONS

Bit Code		Mnemonic	Operations
7 8 9	11 12		
0 1 0	0 0	SL0E	A is scaled left, with 0 entered as least significant bit.
0 1 0	0 1	SL1E	A is scaled left, with 1 entered as least significant bit.
0 1 0	1 0	SLEA	A is scaled left, with end-around carry.
0 1 1	0 0	SDL0E	A/Q is scaled left, with 0 entered as least significant bit in Q.
0 1 1	1 0	SDLEA	A/Q is scaled left, with end-around carry.

TABLE 4-10. A INPUT OPERATIONS

A Code	Mnemonic	Operation
000[†]	F2	Use contents of file 2 register as A source input. Current value of N register is used to address register file 2. If value of N is changed in current microinstruction, its initial value is used to reference file register.
001	P	Use contents of P register as A source.
010	I	Use contents of I register as A source.
011	X	Use contents of X register as A source.
100	A	Use contents of A register as A source.
101	F	Use contents of F register as A source.
110[†]	F1	Use contents of file 1 register (when it is furnished) or external source as A source. Current value of K register is used to address register file 1. If value of K is changed in current microinstruction, initial value of K is used to reference file register. Note that file 1 is available only as an option.
111	MEM	Obtain data read from main memory and use it as A source. If main memory command was not given in the preceding microinstruction, all ones are input to A source if B' source is not selected. Restriction: This command is restricted to a microinstruction with type A, B, or C execution time.

[†]RESTRICTION: Value of addressing register (N or K) cannot have been modified by a C' increment or decrement command in preceding microinstruction. In addition, whichever file is read during the current microinstruction, this same file cannot have been written in the preceding microinstruction.

TABLE 4-11. A' INPUT OPERATIONS

A' Code	Mnemonic	Operation
000	SM1	Use contents of SM register 1 as A source.
001	M1	Use contents of interrupt mask register 1 as A source.
010	SM2	Use contents of SM register 2 as A source.
011	M2	Use contents of interrupt mask register 2 as A source input.
100	A*R8	Use contents of double-precision A* register, shifted right eight bits with end-around carry, as A source. A* register remains unshifted.
101	A*	Use contents of double-precision A* register as A source.
110	X*	Use contents of double-precision X* register as A source.
111	Q*	Use contents of double-precision Q* register as A source.
NOTE: Double-precision hardware has been replaced by the Cyclic Encoder algorithm card and thus the starred instructions refer to the Cyclic Encoder.		

B Field (Bits 10 through 12)

The B field specifies the input to S2 and thus to the B side of the ALU. The 11 possible B codes are expanded by the seven B' codes when the S field contains 1000. The N and K codes are controlled by bits 28 and 29 from the C field.

The eleven B inputs to S2 and thus to the B side of the ALU are indicated if the S field is not 1000. Code 001 of the B field is expanded by the use of bits 28 and 29 of the microinstruction for enabling the N or K register to S2. This use of bits 28 and 29 is independent of the other use of the C field, and thus, by judicious use of commands in the C field, the input for the N and K can be used in conjunction with commands or constants in the C field. The codes for B inputs are given in table 4-12.

The B' inputs are specified in the B field when the S field contains 1000. The codes and actions are given in table 4-13.

TABLE 4-12. B CODES

B Codes	28 29	Mnemonic	Operation
000[†]		F2	Use contents of file 2 register as B source. Value of N register, before instruction is executed, is used to address register file 2. If value of N is changed in current microinstruction its initial value is used to reference file register.
001	1 1	Zero	B source is all zeros.
001[††]	1 0	N	Use contents of N register as B source. Since N is an 8-bit register, this source uses N as the upper 8 bits and zeros as the lower bits.
001[††]	0 0	N, K	Use contents of N and K registers as B source. These registers are combined, with N register as the upper 8 bits of source and K as the lower 8 bits.
010		BG	Use contents of BG register as B source. This register has only one bit set to 1, and position of bit in BG register is specified either on value in the N register or by number in C field, depending on state of controlling SM register bit.
011		X	Use contents of X register as B source.
100		Q	Use contents of Q register as B source.
101		F	Use contents of F register as B source.
110[†]		F1	Similar to F2, but uses the contents of file 1 register (when it is furnished) addressed by (K) as B source. Note that file 1 is available only as an option.
111		MEM	Obtains data read from main memory and uses it as B source. If main memory Read command was not given in the preceding microinstruction, all one's are input to B source if A' source not selected. This command is restricted to a microinstruction with type A, B, or C execution time.

[†]Value of addressing register (N or K) cannot have been modified by a C' increment or decrement command in preceding microinstruction. In addition, whichever file is read during the current microinstruction, this same file cannot have been written in the preceding microinstruction.

[††]In a 32-bit processor, the 16 most significant bits are always zeros. It is the 16 least significant bits of the 32-bit B source that are controlled with these codes.

TABLE 4-13. B' CODES

B' Code	Mnemonic	Operation
000	OPEN	
001	CRTJ	Transfer the complement of the RTJ register to the twelve LSB's of S2. Transfer 1's to the four most significant bits of S2.
010	INRD	Input data/status from I/O channel.
011	INRS	Input to S2 I/O response signals.
100	MMU	Transfer upper 16 bits of data from micromemory to X register in 16-bit MP; 32-bit MP transfer total 32-bit word. F field must make a reference to B source. Address is specified by transform or NK. D field must be an NOP.
101[†]	MML	Same as above, but takes lower 16 bits of data in a 16-bit MP.
110 or 111	INTA	Use contents of interrupt address encoder as B source. The output of this encoder represents the complement of the interrupt address of the highest priority interrupt line active having its corresponding mask bit set. Restrictions: An INTU test command must be given in the preceding microinstruction.

[†]This code is nonoperative in a 32-bit processor.

D Field (Bits 13 through 15)

The D field specifies the destination of information from the main organization of the MP. There are four sources of this information as follows:

- An optionally shifted ALU output. This shifting occurs in the S3 shift network that connects the ALU output to the P, A, F, X, or Q registers.
- The output of the ALU
- The A source
- The B source

All D destinations except the I register are optionally shiftable by S3 when specified by a code in the C field or by the L8EA command in the S field, if the alternate codings, D' or DD", are not specified. The I destination differs from the others in that the output of the A source is the input to the I register. The codes and their operations are given in table 4-14.

TABLE 4-14. D CODE TRANSFERS

D Code	Mnemonic	Operation
000	NOP	Do not transfer data to any destination.
001	P	†Transfer output of S3 to P, AB†††
010	I	Transfer output of S1 to I, AB
011	Q	Transfer output of S3 to Q, AB
100	F1	††Transfer output of S3 to F register, AB, and write this data in file 1 (when it is furnished) at address specified by K at completion of this instruction.†††
101	A	Transfer output of S3 to A, AB†††
110	X	Transfer output of S3 to X, AB†††
111	F	Transfer output of S3 to F, AB†††

†If a D field command to load the AB is issued in the next microinstruction following the microinstruction with this command, the transfer AB is inhibited.

††The writing of data into the file 1 register takes place during the first part of the next microinstruction and takes advantage of the updated value of K from this microinstruction. Also, the next microinstruction must not specify a read of file 1.

†††AB is the main memory address buffer register.

The D' destinations are specified by the D field if the S field is set to 1001 or 1010. The codes and action are given in table 4-15.

The D" destinations specified by the D field if the S field is set to 1011. These destinations transfer data to the double-precision logic from S1. The codes and actions are given in table 4-16.

The DD" option allows the performance of an operation on A, X, or F; this changes the register but keeps a copy of the original register in a double-precision register. This is another way of getting data to the double-precision registers. The DD" option is specified when the S field contains 0001. Table 4-17 lists the DD" codes and their operations.

TABLE 4-15. D' CODE TRANSFERS

D' Code	Mnemonic	Operation
000	IOD	Transfer output of S3 to I/O data register.
001	IOA	Transfer output of S3 via the I/O data register to I/O address register. Destroys contents of I/O data register.
010	MMU	Transfer output of S2 to upper 16 bits of micromemory in 16-bit MP, or transfer output of S2 to 32-bit word in micromemory in 32-bit MP.
011	MML	Transfer output of S2 to lower 16 bits of micromemory location in 16 bit MP, or transfer output of S2 to 32-bit word in micromemory in 32-bit MP.
100	M1	Transfer output of ALU to mask register 1.
101	SM1	Transfer output of ALU to mask register 1.
110	M2	Transfer output of ALU to mask register 2.
111	SM2	Transfer output of ALU to SM register 2.

Note: Outputs to the mask and SM registers are direct from the ALU and are not shiftable.

TABLE 4-16. D" CODE TRANSFERS

D" Code	Mnemonic	Operation
000	NOP	Do not transfer data to any destination.
001	A*LHW	Transfer output of S1 to A* register, shifted left one-half word, with end-around carry.
010	X*LHW	Transfer output of S1 to X* register, shifted left one-half word, with end-around carry.
011	Q*LHW	Transfer output of S1 to Q* register, shifted left one-half word, with end-around carry.
100	NOP	Do not transfer data to any destination.
101	A*	Transfer output of S1 to A* register.
110	X*	Transfer output of S1 to X* register.
111	Q*	Transfer output of S1 to Q* register.

TABLE 4-17. DD" CODES

DD" Code	Mnemonic	Operation
101	AA*	Transfer output of S3 to A register, and transfer output of S1 to A* register.
110	XX*	Transfer output of S3 to X register, and transfer output of S1 to X* register.
111	FQ*	Transfer output of S3 to F register, and transfer output of S1 to Q* register.

T Field (Bits 16 through 18)

The purpose of the T field is to select the upper or lower microinstruction of the next microinstruction pair to execute. The selection of the next microinstruction can be a conditional selection or an unconditional selection, depending on the T field code. This field is available to all addressing modes in addition to I/O operations. A conditional selection can test the ALU output, the value of certain registers, certain internal conditions such as interrupts, and particular bits wired to the transform card. The only exception to these uses of the T field is when micromemory data is being read or written; in these cases, the T field is used as part of the micromemory data reference address and the upper instruction in the next sequential microinstruction pair is always selected.

The T field codes consist of two groups, T codes and T' codes. Similarly to the A, B, and D fields that are extended by using the S field, the T field is always extended in the following sense. Bit 24 of format 1 microinstructions is either zero or one if T or T', respectively, is specified. The T' codes are available only for the return and sequential addressing modes (format 1). The T/T' codes select the upper or lower portion of the next microinstruction pair as the next microinstruction to execute. The T codes are listed in table 4-18; the T' codes are listed in table 4-19.

TABLE 4-18. T ADDRESSING MODES

T Code	Mnemonic	Operation
000[†]	L	Execute lower microinstruction of this microinstruction pair as next microinstruction. This operation overrides M field addressing mode.
000	U	Execute upper microinstruction of next microinstruction pair.
010	L	Execute lower microinstruction of next microinstruction pair.
011[†]	KZU	If initial contents of K register is zero, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair. If decrement K command is included in same microinstruction, K contains all ones on satisfying zero test.
100	NZU	If initial contents of N register is zero, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of the next microinstruction pair. If decrement N command is included in same microinstruction, N contains all ones on satisfying zero test.
101	INTU	If there is an interrupt and its corresponding interrupt mask bit is set, execute upper microinstruction of next microinstruction pair; otherwise execute lower microinstruction of next microinstruction pair.
110	NU	If sign bit of ALU output is negative on completion of this microinstruction, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair.
111	ZL	If output of ALU is zero on completion of this instruction, execute lower microinstruction of next microinstruction pair; otherwise, execute upper microinstruction of next microinstruction pair.

[†]These T-field commands cannot be used in microinstruction with a C-field INCK or INCN command, respectively.

TABLE 4-19. T' ADDRESSING MODES

T' Code	Mnemonic	Operation
000	*L	Execute lower microinstruction of this microinstruction pair. This operation overrides M field addressing mode.
001	LQL	If, at start of this microinstruction, least significant bit of Q is 1, execute lower microinstruction of next microinstruction pair. Otherwise, execute upper microinstruction of next microinstruction pair.
010	K7L	If the LSB of K register is set, execute lower of next microinstruction pair. If clear, execute upper microinstruction of next microinstruction pair.
011	OVFL	If overflow exists, execute lower microinstruction of next microinstruction pair; if not, execute upper microinstruction of next microinstruction pair.
100	BTU	Bit test. Lower order 5 bits in C field of this microinstruction specify a setting of bit test selector. If bit at this position is 1, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair. Bit test is general-purpose testing facility that allows wiring any bit of organization available on machine's backpanel to bit test selector. This wiring is defined on transform card.
101	LQ*L	If, at start of this microinstruction, least significant bit of Q* is 1, execute lower microinstruction of next microinstruction pair; otherwise, execute upper microinstruction of next microinstruction pair.
110	COL	Carry-out lower. If, as result of arithmetic operation, there is carry-out of ALU, execute lower microinstruction of next microinstruction pair. Otherwise, execute upper microinstruction of next microinstruction pair. This instruction allows a test for carry-out of ALU during multiple-precision arithmetic.
111	Z*L	Same as ZL except ALU* is tested.

Subformat Select Bit (SF)

Bit 19 (SF) is used to select either variations in format 1 and format 2 decoding or the choice of addressing the N or K register in format 3.

S Field (Bits 20 through 23)

The S field of the microinstruction is used to specify a special command (including alternate codings in the A, B, and D fields), in addition to page or constant information (as required by the code in the C field). The S codes specify actions which take place at the same time as the ALU operation specified in the F, A, B, and D fields. The codes and operations are given in table 4-20.

C Field (Bits 24 through 31)

The C field is used to specify an additional special operation, an address for a jump, or a constant for setting the K or N register. Bit 24 in format 1 specifies the T field interpretation. The codes for this field are listed in table 4-21.

TABLE 4-20. S FIELD CODES

S Code	Mnemonic	Operation
000C	NOP	No operation for S field.
0001	DD	Alternate D field coding, DD".
0010	RPT	If N register is not zero, selection of next microinstruction pair is inhibited and current microinstruction pair is next microinstruction pair. N register is decremented by one. Normal T field selection applies. If N register is zero, normal next microinstruction pair is used.
0011	READ	Read main memory command. Read word from main memory at address contained in the address buffer (AB). Instruction execution is delayed until a resume is received from memory acknowledging command. Restrictions: AB must be loaded in a previous microinstruction (D field command). If D field command reloading AB is issued in same microinstruction as Read command, the new data is not loaded into AB until completion of read portion of cycle. A Read command can only be given in a microinstruction with type A, B, C, or D execution time. Memory data must be input to system in following microinstruction with type A, B, or C execution time only.

TABLE 4-20. S FIELD CODES (Contd)

S Code	Mnemonic	Operation
0100	WRITE	Write main memory. Transmit output of S3 to main memory as data to be written at address contained in address buffer (AB). Instruction execution is delayed until a resume is received from memory acknowledging command. Data is stored in memory at completion of this instruction. Restrictions: AB must be loaded in a previous micro-instruction (D field command). If D field command reloading AB is issued in same micro microinstruction as Write command, the new data is not loaded into AB until completion of Write portion of cycle.
0101	L8EA	Output of ALU via S3 is shifted left 8 bits, end-around.
0110	F2WR	Write data contained in F register into file 2 at address specified by contents of N register at beginning of current microinstruction. Actual writing takes place during first part of instruction.
0111	AP	Alternate A field coding, A'.
1000	BP	Alternate B field coding, B'.
1001	DP	Alternate D field coding, D'.
1010	APDP	Alternate A and D field coding, A'D'.
1011	DPP	Alternate D field coding, D".
1100	GATEI	Gate output of S1 to I register.
1101	HALT	If halt bit of SM register is 1, stop operation of MP on completion of this microinstruction. When start signal is received, continue with next micro-instruction specified by addressing mode and T field. If halt bit is 0, continue with microinstruction sequencing.
1110	RTJ	Transfer address of next sequential microinstruction pair to RTJ register. This is done regardless of actual addressing mode used in this instruction.
1111	CLRNP	Clear N register and page register.

TABLE 4-21. C CODE ACTIONS

C' Code	Mnemonic	Operation
		The following special operations, for format 1 have a zero in bit 19 (SF) to specify C' codes (bit 25 through 31). Bit 24 specifies the T field interpretation.
00xxxxx		xxxxx is a constant for use in driving the bit generator or in any other commands using lower 5 bits of instruction for control.
0100000 0100001 0100010†	WRCH/0 WRCH/1 WRCH/2	Write 8-bit character specified from output of S3 at memory address specified by output of AB. See Write command in S field for details of operation and restrictions. Character 0 is bit 0 through 7 (MSBs); character 1 is bits 8 through 15, etc. Character is not repositioned in Write command.
0100100	RMW	Read modify write - Perform a read of information from main memory in same manner as Read instruction in S field. Memory system performs a read cycle and lockup before performing write cycle. Memory must be forced to complete write cycle by issuing Write instruction, using same address, before it responds to any other read operations.
0100101	WRHW0	Write bit 0 through 15 (MSBs) from output of S3 at memory address specified by output of AB. Bits 16 through 31 in memory are unchanged. See Write command in S field for details of operation and restrictions.
0100111†	WRHW1	Write bits 16 through 31 (LSBs) from output of S3 at memory address specified by output of AB. Bits 0 through 15 in memory are unchanged. See Write command in S field for details of operation and restrictions.
0101000	WRPB	Write protect bit - See protect system discussion (Appendix A).
011xxxx	GATEIXT	General purpose strobe at T4 time. In MP17, this signal is used to gate IXT on 1700 transform module.
1000101	INCK	Increment number contained in K register by one.
1001101	INCN	Increment number contained in N register by one.
1000100	DECK	Decrement number contained in K register by one.
1001100	DECN	Decrement number contained in N register by one.
1000000	CLRK	Clear K register.
1001000	CLRN	Clear N register.

TABLE 4-21. C CODE ACTIONS (Contd)

C' Code	Mnemonic	Operation
101xxxx	SETF/j	xxxx is value of j, from 0 to 15. Set SM register flag j to 1.
110xxxx	CLRF/j	xxxx is value of j, from 0 to 15. Clear SM register flag j to 0.
1110000[††]	RQLXN	Destination register (P, A, F, or X) and Q register are considered as one double-length register with Q register as lower-order bits. Combined register is shifted left one bit position with complement of ALU sign bit entered into lowest bit position of Q register.
1110011[††]	RQR1E	Shift combined destination and Q register right one bit, and enter 1 in sign position of destination register. This command is used in multiply iteration.
1110010[††]	RQR0E	Shift combined destination and Q register right one bit, and enter 0 in sign position of destination register. This command is used in multiply iteration.
1110100[††]	RL0E	Shift destination register left one bit, entering 0 in lowest bit position of the register. This operation cannot be performed when Q is destination register.
1110101[††]	RL1E	Shift destination register left one bit, entering 1 in lowest bit position. This operation cannot be performed when Q is destination register.
1110110[††]	RR0E	Shift destination register right one bit, entering 0 in sign position of register. This operation cannot be performed when Q is destination register.
1110111[††]	RR1E	Shift destination register right one bit, entering 1 in sign position. This operation cannot be performed when Q is destination register.

The following special operations, for format 1, have a one in bit 19 (SF) to specify the C" codes (bits 25 through 31). Bit 24 specifies the T field interpretation. See discussion of transforms and transform module.

C" Code		
000xxxx[†††]	TMA/J	xxxx = j with value 0 through 15. Obtain next microinstruction pair from address specified by MA transform selector setting j.

TABLE 4-21. C CODE ACTIONS (Contd)

C" Code	Mnemonic	Operation
001xxxx[†††]	TMAK/j	xxx = j, 0 through 15. Obtain next instruction pair from address specified by MA transform selector setting j. Also, set K register to value specified by K transform selector setting j.
0100xxx[†††]	GITMAK/j	Gate output of main memory to IXT register (on transform module) and perform TMAK/j operation. Note that j = xxx with values of 0 through 7. This command must be executed in the microinstruction following a Read command. This command is applicable only if transform module is configured with IXT register.
0101xxx[†††]	GITMAK/xt	Gate output of main memory to IXT Register (on transform module) and perform a transform of the upper 16 bits of MIR and select one of eight transforms of MA from the decoded and encoded microinstruction loaded into IXT. This command must be executed in the microinstruction following a Read command. This command is applicable only if configuration includes the required specialized transform module hardware.
011xxxx[†††]	TK/j	xxxx = j, with values from 0 through 15. Set K register to value specified by K transform selector setting j.
100xxxx	TN/j	xxxx = j, from 0 through 15. Set N register to value specified by N transform selector setting j.

The following two codes are used if the optional program protect logic is included.

101xxx	SUB	Set upper bounds - transfer output of S3 to upper bounds register in main memory interface.
110xxxx	SLB	Set lower bounds - transfer output of S3 to lower bounds register in main memory interface.

TABLE 4-21. C CODE ACTIONS (Contd)

C Code	Mnemonic	Operation
The following format 3 codings are for setting the K and N register; this format has the M field bits 0 and 1 set to 11, while bit 19 selects the register.		
Value	K = value	When microinstruction bit 19 is 0, transfer C field value (bits 24 through 31) to K register and execute next sequential microinstruction pair.
Value	N = value	When microinstruction bit 19 is 1, transfer C field value (bits 24 through 31) to N register and execute next sequential microinstruction pair.
The following format 2 codings in the C field are used to perform a jump, specified if the M field (bit 0 and 1) is 10.		
Number	Number	Number is address of next instruction pair. If page jump is required (bit 19 is 1), S field contains page setting instead of S field code.

†NOTE: Commands applicable to 32-bit processor only.

††RESTRICTION: This operatin cannot be performed when S field command (L8EA) is selected.

†††RESTRICTION: If SM 207 is set to 1 , S field commands are disabled.

MICROINSTRUCTION TIMING

The basic microinstruction execution time is 168 nanoseconds. Some microinstructions have longer execution times to allow certain operations to be completed. The microinstructions have been grouped according to execution times as A, B, C, D, E, F, and G as shown in table 4-22.

The classification of microinstructions is shown in figure 4-8. This figure provides execution time by types for all legal combinations of microcommands that extend the basic cycle time.

Exceptions to the execution times listed in figure 4-8 are:

1. The combination of one's complement arithmetic with an ADD+ or ADD+T arithmetic operation is classified as a type B rather than a type C.

2. The $\underline{\text{MMU}}$ and $\underline{\text{MML}}$ B' commands are included under read/write micromemory operand commands; therefore, they are a type F rather than a type B.

3. A type G microinstruction followed by a microinstruction with a GITMAK command (C field) has an additional execution time of 185 nanoseconds (typical); thus the total execution time becomes 625 nanoseconds.

Analysis of a microprogram for execution time starts by classifying each of the microinstructions as type A, B, C, D, E, F, or G. This is done by using the microinstruction classification table or by examining the assembler output listing.

Each microinstruction with a main memory Read or Write command has a 440-nanosecond (typical) execution time regardless of the type of cycle (assuming no DMA activity on main memory interface). However, type E and F microinstructions cannot contain a main memory reference command. An additional execution time of 185 nanoseconds (typical) is required on all Read commands followed by a microinstruction containing a GITMAK command.

The total execution time required is then calculated from start of main memory command to next main memory command. If the total time, starting with a microinstruction with a Read or Write command and including all micro-instructions up to the following Read or Write command is less than the memory cycle time (600 nanoseconds), take 600 nanoseconds as the execution time for that sequence. If the total time is greater than 60 nanoseconds, the execution time for that sequence is the calculated time. See figure 4-9.

TABLE 4-22. MICROINSTRUCTION EXECUTION TIMES

Microinstruction Type	Execution Time in Nanoseconds[†]
A	168
B	224
C	280
D	336
E (shift or scale)	280 +56n (where n = number of shifts)
F (micromemory read/ write operand)	448
G (read/write main memory)	440 (typical)

[†]Execution times may vary ±4% over temperature range (0° to 70°C) and voltage ±5%.

READ/WRITE MAIN MEMORY	READ/WRITE MICROMEMORY OPERAND	SHIFT OR SCALE	ONE'S COMPLEMENT ARITHMETIC	ADD OR SUBTRACT	TMA, TMAK, GITMAK	A', B', NU, ZL, COL, Z*L	INSTRUCTION TIME
0	0	0	0	0	0	0	A
0	0	0	0	0	0	1	B
0	0	0	0	0	1	0	C
0	0	0	0	0	1	1	C
0	0	0	0	1	0	0	B
0	0	0	0	1	0	1	C
0	0	0	0	1	1	0	C
0	0	0	0	1	1	1	C
0	0	0	1	0	0	0	A
0	0	0	1	0	0	1	B
0	0	0	1	0	1	0	C
0	0	0	1	0	1	1	C
0	0	0	1	1	0	1	D
0	0	0	1	1	1	0	C
0	0	0	1	1	1	1	D
0	0	1	0	0	0	0	E
0	0	1	0	0	1	0	E
0	1	0	0	0	0	0	F
1	0	0	X	X	X	X	G

X = 0 OR 1 (DON'T CARE CONDITION)

Figure 4-8. Microinstruction Classification

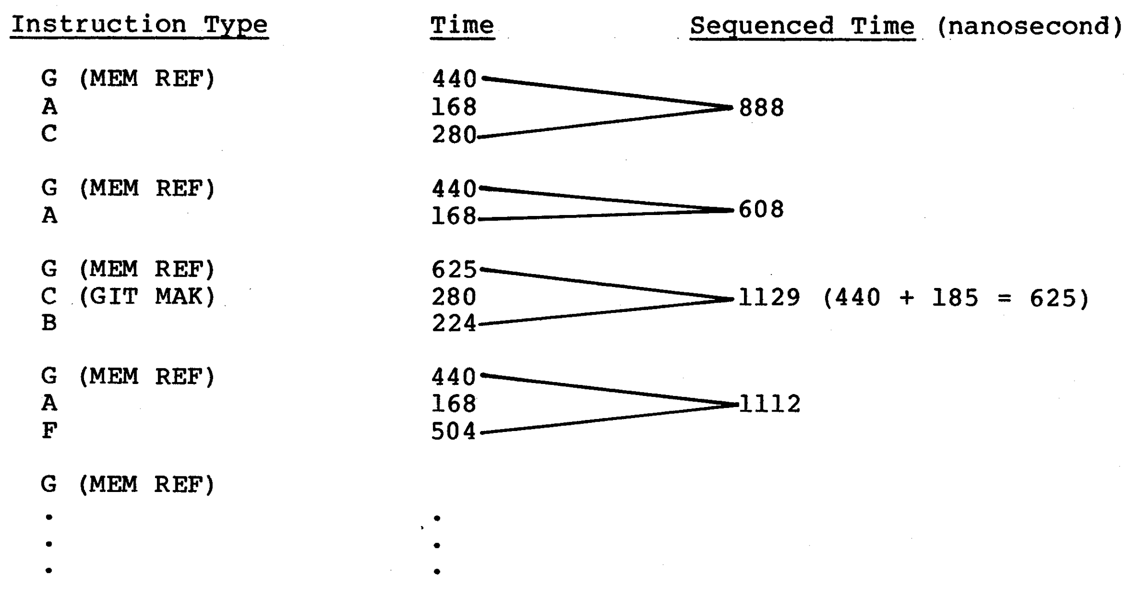


Figure 4-9. Calculation of Microprogram Sequence Execution Time

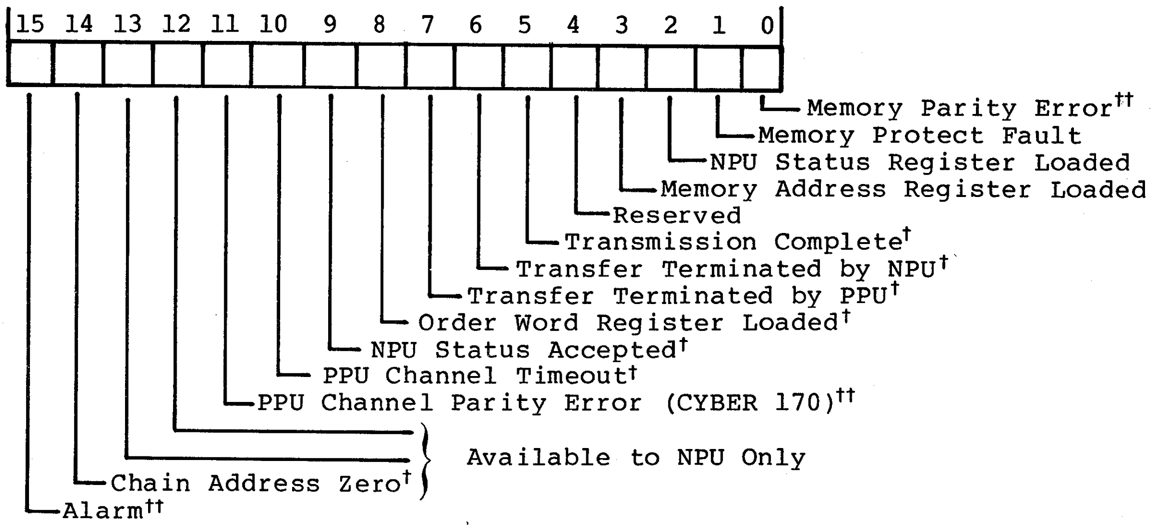
I/O PROGRAMMING FORMATS 5

This section contains I/O programming formats for communications couplers and peripheral I/O devices installed with the 255X NPU system. See figures 5-1 through 5-15.

11 10 9	8	7	6	5	4	3	2	1	0	
	1	0	0	0	0	X	X	X	X	Clear Coupler
Equip Code (=7)	0	1	0	0		X	X	X	X	Clear Processor
	0	0	1	0		X	X	X	X	Stop Processor
	0	0	0	1		X	X	X	X	Start Processor
	X	X	X	X		0	0	0	0	Input Memory Address Zero†
	X	X	X	X		0	0	0	1	Input Memory Address One†
	X	X	X	X		0	0	1	0	
	X	X	X	X		0	0	1	1	Input Data
	X	X	X	X		0	1	0	0	Input Processor Status
	X	X	X	X		0	1	0	1	Input Coupler Status
	X	X	X	X		0	1	1	0	Input Order Word†
	X	X	X	X		0	1	1	1	Input Program
	X	X	X	X		1	0	0	0	Output Memory Address 0 (upper byte)
	X	X	X	X		1	0	0	1	Output Memory Address 1 (lower byte)
	X	X	X	X		1	0	1	0	
	X	X	X	X		1	0	1	1	
	X	X	X	X		1	1	0	0	Output Data
	X	X	X	X		1	1	0	1	Output Program
	X	X	X	X		1	1	1	0	Output Order Word
	X	X	X	X		1	1	1	1	

†Hardware Maintenance feature

Figure 5-1. Communications Coupler PPU Function Codes



[††]Alarm Condition (all alarms generate NPU interrupt)
[†]NPU Interrupt Condition

NOTE: All nonalarm interrupt conditions except OWRL are cleared by input coupler status command, clear coupler function, and clear coupler command. OWRL interrupt condition is cleared by input OW command, clear coupler function and command, and Master Clear. Alarm interrupt conditions are cleared only by clear coupler functions and commands and Master Clear.

Figure 5-2. Communications Coupler Status Format

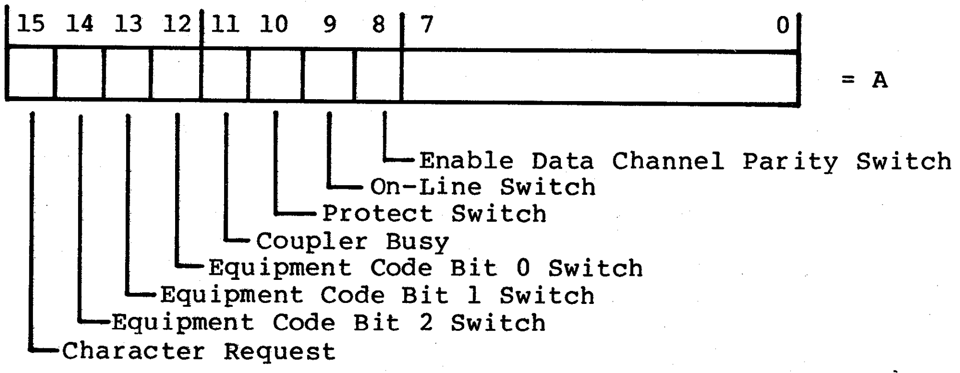

15 14 13 12 11 10 9 8 7 0
= A
Enable Data Channel Parity Switch
On-Line Switch
Protect Switch
Coupler Busy
Equipment Code Bit 0 Switch
Equipment Code Bit 1 Switch
Equipment Code Bit 2 Switch
Character Request

Figure 5-3. Communications Coupler Input Switch Status Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	E	E	E							

Equipment Address (bits 10–7)

First Coupler = 1100 ('C') Standard

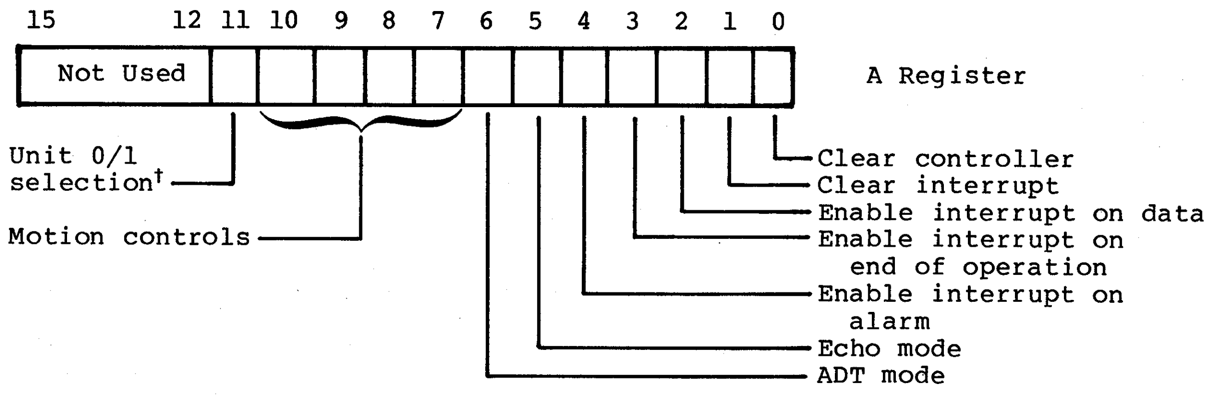
Second Coupler = 1101 ('D') (Optional)

Bits 1–0: Not Used

	6	5	4	3	2	
Sample (Read)	0	0	0	0	0	Input Memory Address Zero†
	0	0	1	0	0	Input Memory Address
	0	1	0	0	0	
	0	1	1	0	0	Input First/Present Character Displacement
	1	0	0	0	0	Input Processor Status†
	1	0	1	0	0	Input Coupler Status
	1	1	0	0	0	Input Order Word
	1	1	1	0	0	Input IO†
	0	0	0	0	1	Input Last Word From Data Channel†
	0	0	1	0	1	Input FDMAR0/FDMAR1†
	0	1	0	0	1	Input FDMAR0/Flag Mux†
	0	1	1	0	1	Input FDMAR1/Flag Mux/Flag Register†
	1	0	0	0	1	
	1	0	1	0	1	Input Switch Status
	1	1	0	0	1	
	1	1	1	0	1	Input Character†
Set (Write)	0	0	0	1	0	Output Memory Address Zero†
	0	0	1	1	0	
	0	1	0	1	0	
	0	1	1	1	0	Output FCD, PCD, LCD†
	1	0	0	1	0	Output Processor Status
	1	0	1	1	0	Output Buffer Length
	1	1	0	1	0	Output Order Word†
	1	1	1	1	0	
	0	0	0	1	1	Clear Coupler Terminate Transfer
	0	0	1	1	1	
	0	1	0	1	1	
	0	1	1	1	1	
	1	0	0	1	1	Output Test†
	1	0	1	1	1	Input Test†
	1	1	0	1	1	Output Memory Address
	1	1	1	1	1	Output Character†

†Hardware maintenance feature

Figure 5-4. Communications Coupler-Processor Set/Sample Instruction Format



[†]A manual switch on the controller card can override this function bit and force Unit 1 to always be selected. Standard NPU systems always have this switch on.

Figure 5-5. Cassette Tape Controller Control Functions Format

A Register Bits				Function
10	9	8	7	
1	0	0	0	Search Tape Mark (Reverse)
1	0	0	1	Search Tape Mark (Forward)
1	0	1	0	Write Tape Mark
1	0	1	1	Write One Record
1	1	0	0	Backspace One Record (or Tape Mark)
1	1	0	1	Rewind
1	1	1	0	Erase
1	1	1	1	Read One Record

Figure 5-6. Cassette Tape Controller Motion Controls

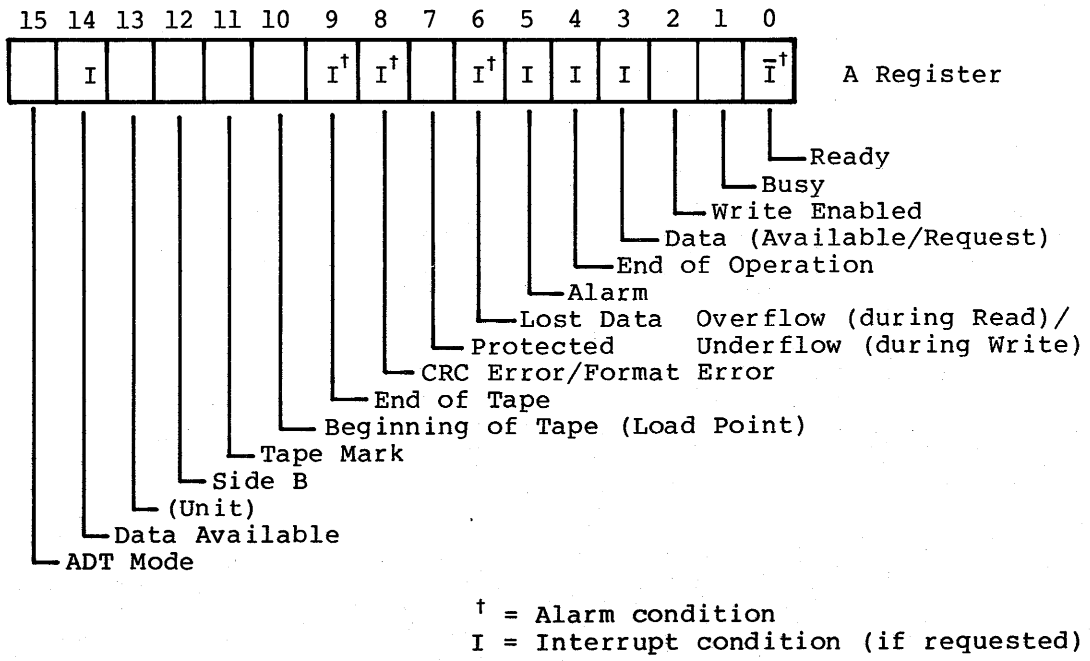


Figure 5-7. Cassette Tape Controller Status Response Format

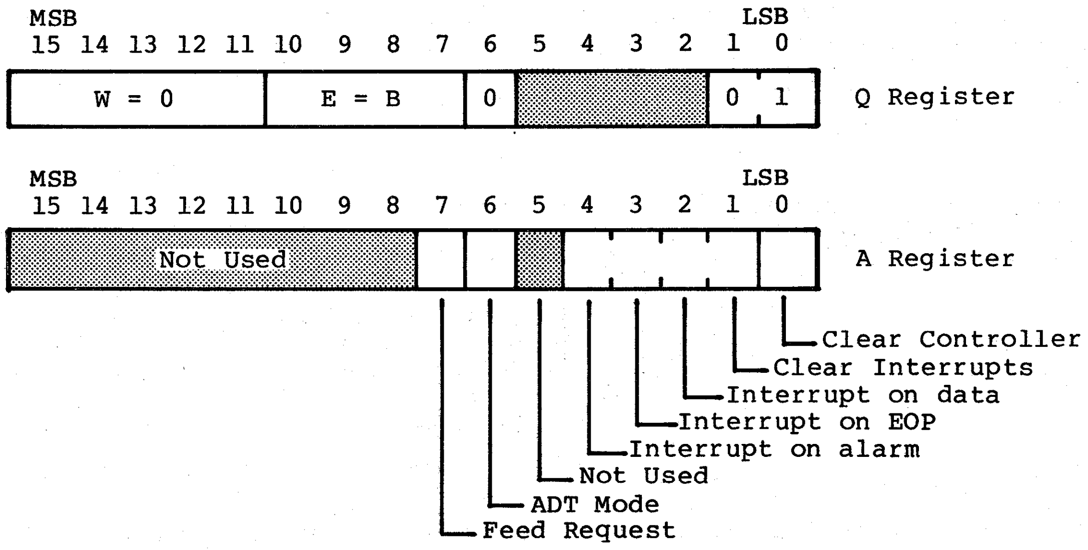


Figure 5-8. Card Reader Controller Director Function Format

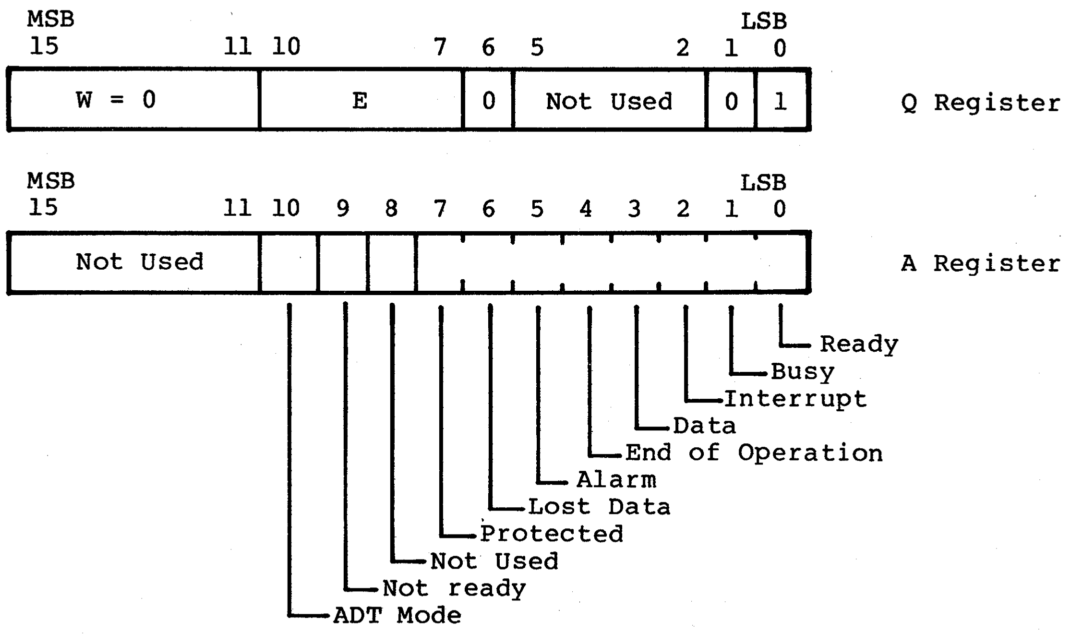


Figure 5-9. Card Reader Controller Director Status 1 Format

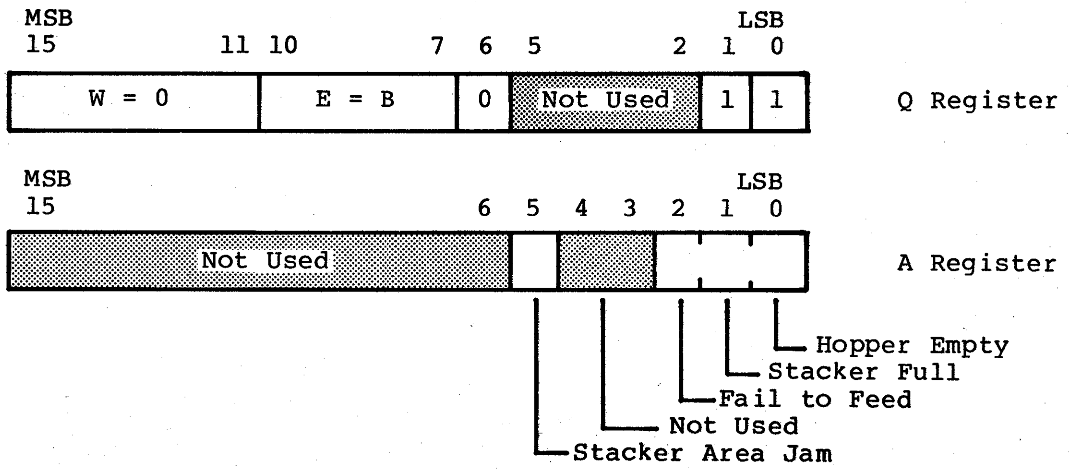


Figure 5-10. Card Reader Controller Director Status 2 Format

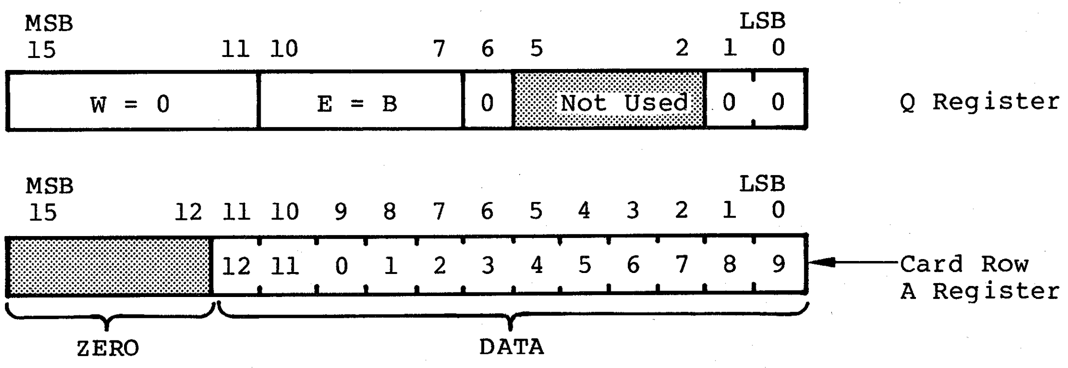


Figure 5-11. Card Reader Controller Data Transfer Command Format

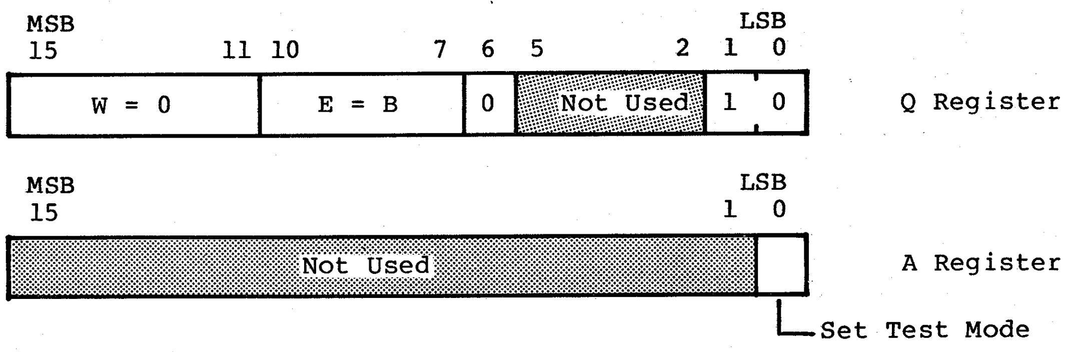


Figure 5-12. Peripheral Controller Self-Test Command Format

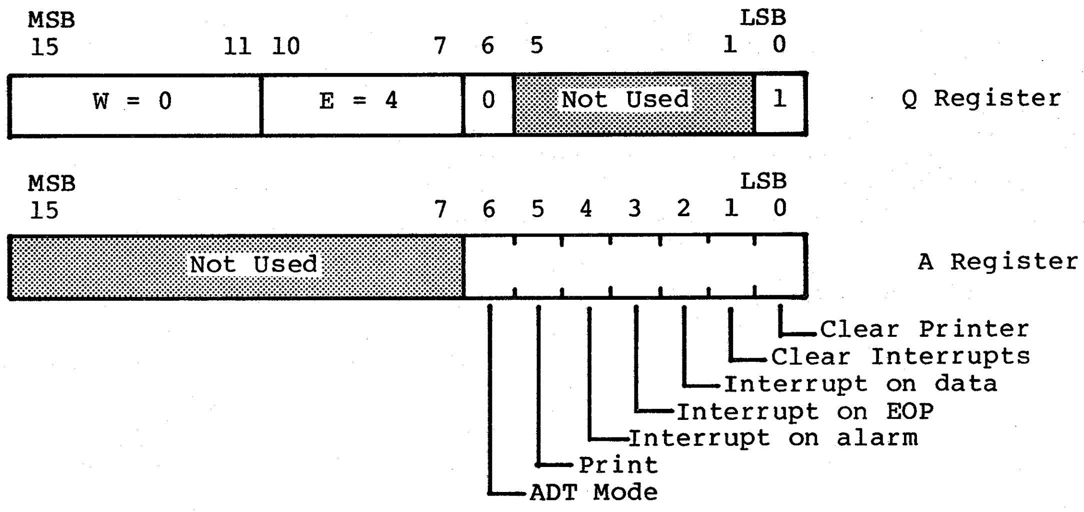


Figure 5-13. Line Printer Controller Director Function Format

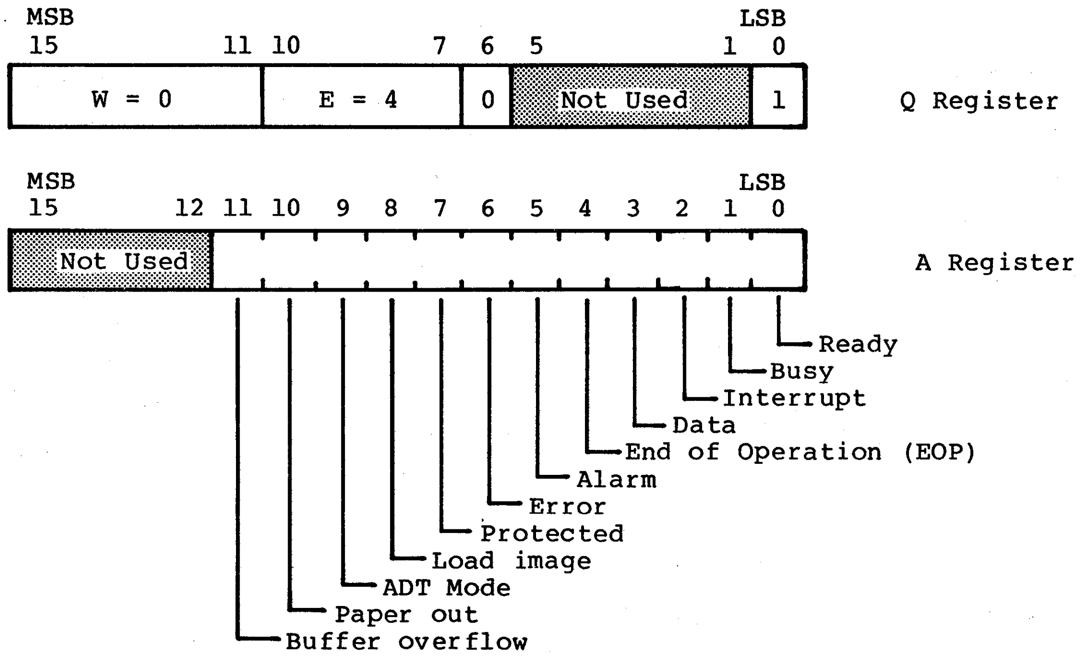


Figure 5-14. Line Printer Controller Director Status Response Format

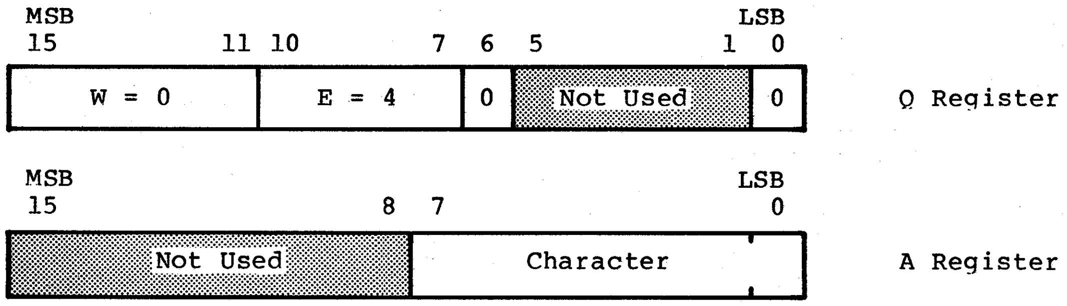


Figure 5-15. Line Printer Controller Data Transfer Command Format

HEXADECIMAL/DECIMAL CONVERSION A

TO CONVERT DECIMAL TO HEXADECIMAL

1. Find decimal number in body of table (Example: 157).
2. Scan horizontally to left to find first hexadecimal digit (e.g., 9
3. Scan vertically up (from 157 in table) to find second hexadecimal digit (in this case, D).
4. Thus decimal number 157 = hexadecimal number 9D.

TO CONVERT HEXADECIMAL TO DECIMAL

1. To find first hexadecimal digit in left-hand column (Example: D).
2. Find second hexadecimal digit in top row (Example: 9).
3. Scan horizontally to right (from D) and scan vertically downward (from 9) to find point of intersection in body of table (in this case, 217)
4. Thus hexadecimal number D9 = decimal number 217.

TABLE A-1. HEXADECIMAL/DECIMAL CONVERSION

First Hex Digit	Second Hexadecimal Digit															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	317	318	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	$\boxed{157}$	518	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	$\boxed{217}$	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

MNEMONICS LISTING B

MNEMONIC AND DESCRIPTION

ALU	Arithmetic and logical unit
A/Q	Type of IDC interface (CDC)
BG	Bit generator
BP	Breakpoint
CCITT	Consultive Committee, International Telephone & Telegraph
CIB	Circular input buffer
CLA	Communications line adapter
CLE	Communications line expansion module
CLR	Clear
CLRF	Clear flags
CRC	Cyclic redundancy checksum (created by cyclic encoder)
CRCS	Cyclic redundancy checksum (created by multiplex loop interface adapter)
CRT	Cathode ray tube
CTS	Clear to send
DMA	Direct memory access
DSA	Direct storage access
DSR	Data set ready
DTO	Data transfer overrun
DTR	Data terminal ready
DTU	Data transfer underrun
ECHO	Echoplex mode
EOB	End of loop batch
EOF	End of line frame
EXGO	External go
EXMC	External master clear
EXSTOP	External stop
FCD	First character displacement
FCO	First character outbound
FCR	Function control register
FES	Framing error status
FIFO	First in, first out
FNE	Flag, null empty
IAV	Input available
IDC	Internal data channel
IEN	Input end
IER	Input error
I/F	Interface
ILE	Input loop error
INS	Input select
INT	Interrupt
INTU	Interrupt test
I/O	Input/output
ION	Input section on
ISON	Input supervisor on
ISR	Input supervision report
IST	Input strobe
LB	End of loop batch flag
LCD	Last character displacement
LED	Light-emitting diode
LF	End of line frame flag
LFP	Last frame position
LIT	Loop input test
LOC	Location
LRC	Longitudinal redundancy checksum
LSB	Least significant bit
LSI	Large-scale integration
MA	Micromemory address
MAC	Memory address counter
MAE	Memory address error
MCA	Memory acknowledge
MCL	Master clear
MCR	Memory cycle request
MCT	Memory cycle timing
MDS	Memory data strobe
MIR	Microinstruction register
MLIA	Multiplex loop interface adapter
MML	Micromemory lower
MMU	Micromemory upper
Modem	Modulator/Demodulator
MO5	Type of IDC interface (NCR)
MOS	Metal oxide semiconductor
MPE	Memory parity error
MPP	Microprogrammed processor
MR	Master reset line
MSB	Most significant bit
MSI	Medium-scale integration
MW	Memory write
NCNA	Next character not available
NSYN	New synchronization
NWP	Next word position
ODD	Output data demand
OER	Output error
OLE	Output loop error
OM	Originate mode
OON	Output section on

OSL	Output select	SB	Stop bit
OST	Output strobe	SCL	Serial computation logic
PAD	Process additional data	SCR	Serial block receive
PC	Program controlled	SCT	Serial clock transmit
PCD	Present character displacement	SD	Send data
PES	Parity error status	SETF	Set flags
PG	Protective ground	SM	Status mode
PI	Parity inhibit	SMI	Status mode interrupt
PIO	Program input/output	SO	Storage register outbound
P/MA	Page/micromemory address	SQD	Signal quality detector
PPU	Peripheral processing unit	SRLSD	Secondary receive line signal detector
PROM	Programmable read-only memory		
PS	Page storage	SRTS	Secondary request to send
PSET	Parity set	SSEE	Test section and error number
RAM	Random access memory	SSI	Small-scale integration
RD	Receive data	SSTB	Strobe line
RDATA	Receive data (modem)	STERM	Terminate
RETL	Return address location	STJP	Stop jump
RI	Ring indicator	TB	Terminal busy
RLSD	Receive line signal detector	TDATA	Transmit data
RNI	Read next instruction	TMA	Transmit memory address
ROM	Read-only memory	TTL	Transistor/transistor logic
RSD	Restraint detector	TTY	Teletypewriter
RSYN	Resynchronize	U/L	Upper/lower
RTJ	Return jump	XT/MA	Transform/memory address
RTS	Request to send		
R/W	Read/write		

COMMENT SHEET

MANUAL TITLE: CDC Network Processor Unit 2551-1, 2551-2, 2551-3, 2551-4, 2552-2 Hardware Reference Manual

PUBLICATION NO.: 60472800 REVISION: D

NAME: ____________________

COMPANY: ____________________

STREET ADDRESS: ____________________

CITY: ____________ STATE: ____________ ZIP CODE: ____________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

FOLD FOLD

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD FOLD