Burroughs

# B 700 SYSTEMS

## B 720 Systems

## REFERENCE MANUAL

# Burroughs

# B 700 SYSTEMS

# B 720 Systems

## REFERENCE MANUAL

**Burroughs Corporation**
Detroit, Michigan 48232

Printed in U.S. America

4-75

1082484

AA615386



Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Technical Information Organization, Burroughs Corporation, 200 West Lancaster Ave., Wayne, Pa. 19087.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS



# LIST OF TABLES

# INTRODUCTION

The Burroughs B 720-Series Computer System is a small-scale, but powerful, system which incorporates the latest state-of-the art data processing techniques, such as dynamic interpreter, interpretive structure, and programmable data communications processor.

This manual provides reference data concerning the major functional, operational, and programming characteristics and capabilities of the B 720 System. The intent of the presentation is to provide system-level product information to sales personnel, prospective customers, and users of the system, while also providing sufficient systems and programming reference data to aid programmers in understanding and applying the capabilities of the System.

Section 1, Equipment System Description, introduces the B 720 System and its physical/functional configuration. Basic operational characteristics and capabilities are presented for each major equipment unit and subsystem.

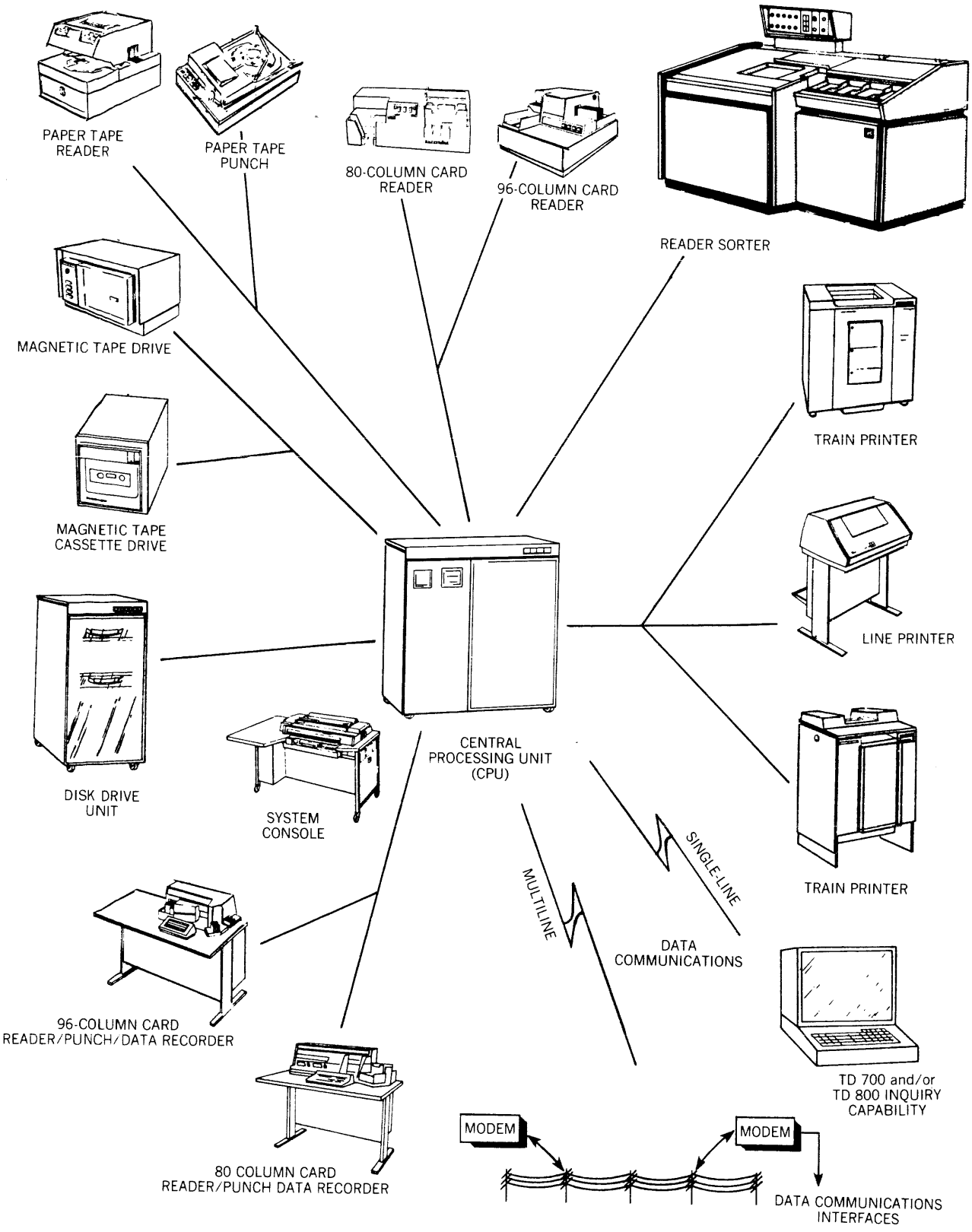
Section 2, Programming System Description, describes the B 720 programming system structure and the function/application of each major programming element. Application data, such as memory allocation and word/data structure, are provided for reference purposes.

Section 3, System Processor Description, provides a general functional and operational description of the B 720 Central Processing Unit.

Section 4, Data Communications Subsystems, provides a general functional and operational description of the two types of data communications subsystems: the programmable communications processor subsystem, and the single-line subsystem (file inquiry application).

Section 5, Reader Sorter Subsystem, provides a general functional and operational description of the B 720 application of the reader sorter capability in processing encoded documents.

Appendices are provided for quick reference to such items as terms and abbreviations, instruction codes, registers, and flags.


PAPER TAPE READER
PAPER TAPE PUNCH
80-COLUMN CARD READER
96-COLUMN CARD READER
READER SORTER
MAGNETIC TAPE DRIVE
TRAIN PRINTER
MAGNETIC TAPE CASSETTE DRIVE
LINE PRINTER
DISK DRIVE UNIT
CENTRAL PROCESSING UNIT (CPU)
SYSTEM CONSOLE
TRAIN PRINTER
SINGLE-LINE
MULTILINE
DATA COMMUNICATIONS
96-COLUMN CARD READER/PUNCH/DATA RECORDER
TD 700 and/or TD 800 INQUIRY CAPABILITY
80 COLUMN CARD READER/PUNCH DATA RECORDER
MODEM
MODEM
DATA COMMUNICATIONS INTERFACES

# SECTION 1

# SYSTEM DESCRIPTION

## GENERAL

The Burroughs B 720 Computer System represents a significant expansion, in terms of capabilities and flexibility, to the B 700-Series of small-scale, powerful computer systems. In addition to retaining most of the existing B 720 System capabilities, the B 720 provides the following major enhancements and capabilities:

- Main memory capacity of up to 96K, 8-bit bytes (expansible from 16KB to 96KB in 8KB increments).
- Expanded I/O device or subsystem handling (up to 11 I/O ports; modular I/O expansion capability).
- Programmable data communications subsystem (communications processor) with enhanced communications capabilities.
- Direct-memory access for disk and data communications subsystems.
- Integral photoelectric memory loader.
- Direct entry system console with electronic keyboard.
- MICR/OCR (character recognition) subsystem capability.

Figure 1-1 illustrates a typical console-based system.

## SYSTEM EQUIPMENT CONFIGURATION AND CHARACTERISTICS

The B 720 is a modular system in terms of functional equipment units or subsystems and can be configured within the minimum and maximum restrictions to fulfill a customer's requirements. Figure 1-2 is a functional interface representation of the general system configuration. Table 1-1 lists the general characteristics and capabilities of the Central Processor and each type of peripheral device that can be used in the system. The basic (or minimum requirement System consists of the following:

a. B 720 Central Processing Unit (CPU) with minimum main-memory capacity of 32K-bytes, photoelectric (paper tape) memory loader, and power supply group.

b. B 9343-22 Console (26 inch form-feed, 64 characters) or B 9343-42 Console (26 inch form-feed, 94 characters).

c. B 346 (64-character) or B 346-1 (94-character) Console Control.

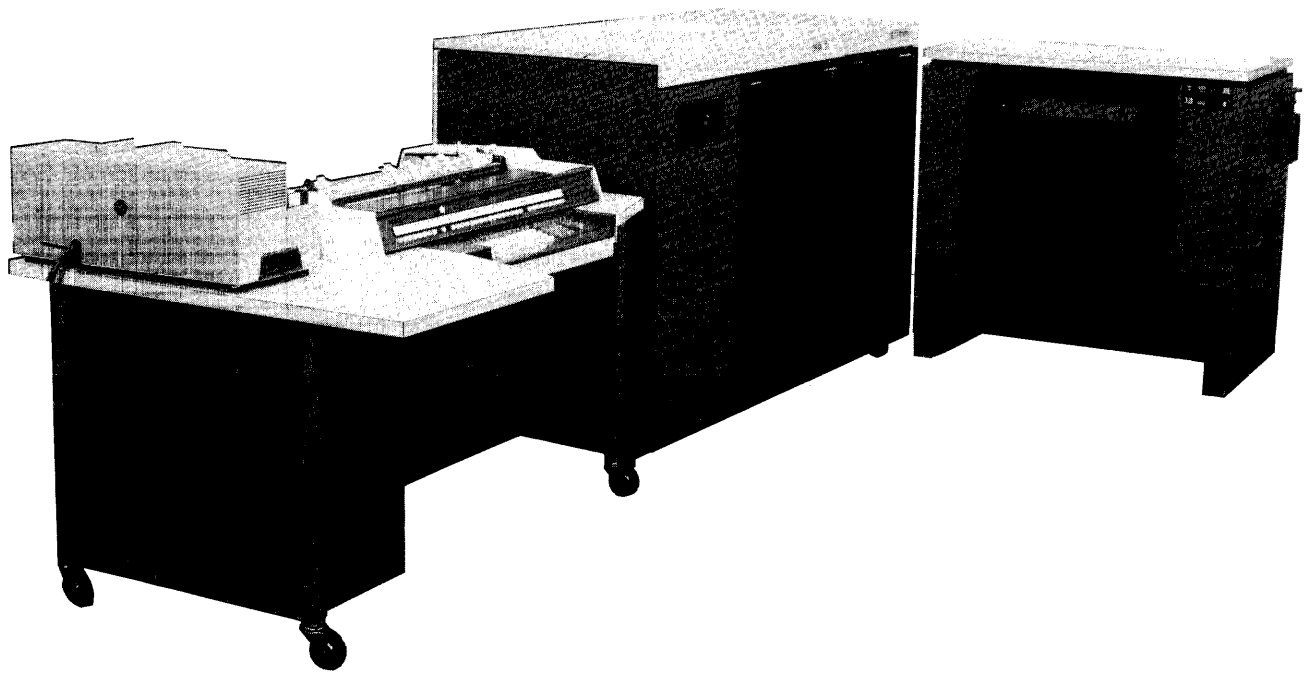


**Figure 1-1. Typical B 720 Computer System**

d. B 9480-11/12 (single/dual drive, 100 TPI) or B 9481-11/12 (single/dual drive, 200 TPI) Disk Cartridge Drive.

e. B 489-2 Cartridge Disk Drive Control.

The main-memory capacity can be expanded to 96 K-bytes in 8 K-byte increments (total of 12 B 31-2 memory modules). Up to 11 I/O control ports (Device Dependent Ports, DDP's) are available in the processor to accommodate basic and optional peripheral devices or interfaces. An additional port is reserved for communications processor expansion.

Three of the processor IOC ports (DDP1, DDP4, and DDP12) are dedicated to the data communications subsystem and basic system controls. Three additional controls may be installed in interchangeable ports in the basic system. Up to five additional controls may be installed by use of the optional I/O expansion module, which provides five interchangeable ports. Table 1-2 lists the applicability of the basic system I/O controls (IOC's) and those I/O controls available to interface the various optional peripheral devices.

## PERIPHERAL DEVICE SUBSYSTEMS

As shown in figures 1-2 and 1-3 (peripheral device subsystem configurator), the minimum system configuration incorporates two dedicated peripheral device subsystems: the B 9343-22 or B 9343-42 Console interfaced through the B 346 or B 346-1 Console I/O Control, and the A 9480-11/12 or A 9481-11/12 Disk Cartridge Drive Unit interfaced through the B 489-2 Cartridge Disk Drive I/O Control. The integral processor memory loader device is interfaced directly with the processor main memory and does not require a dedicated port.

In addition to the basic or required peripheral subsystems, the following optional peripheral device subsystems are available for use with the system through their respective I/O controls and the processor interchangeable ports:

a. Punched card reader subsystems (80-column and/or 96-column).

b. Punched Paper Tape Reader/Punch Subsystem.

c. Punched Paper Tape (PPT)/Edge-Punched Card (EPC) subsystems.

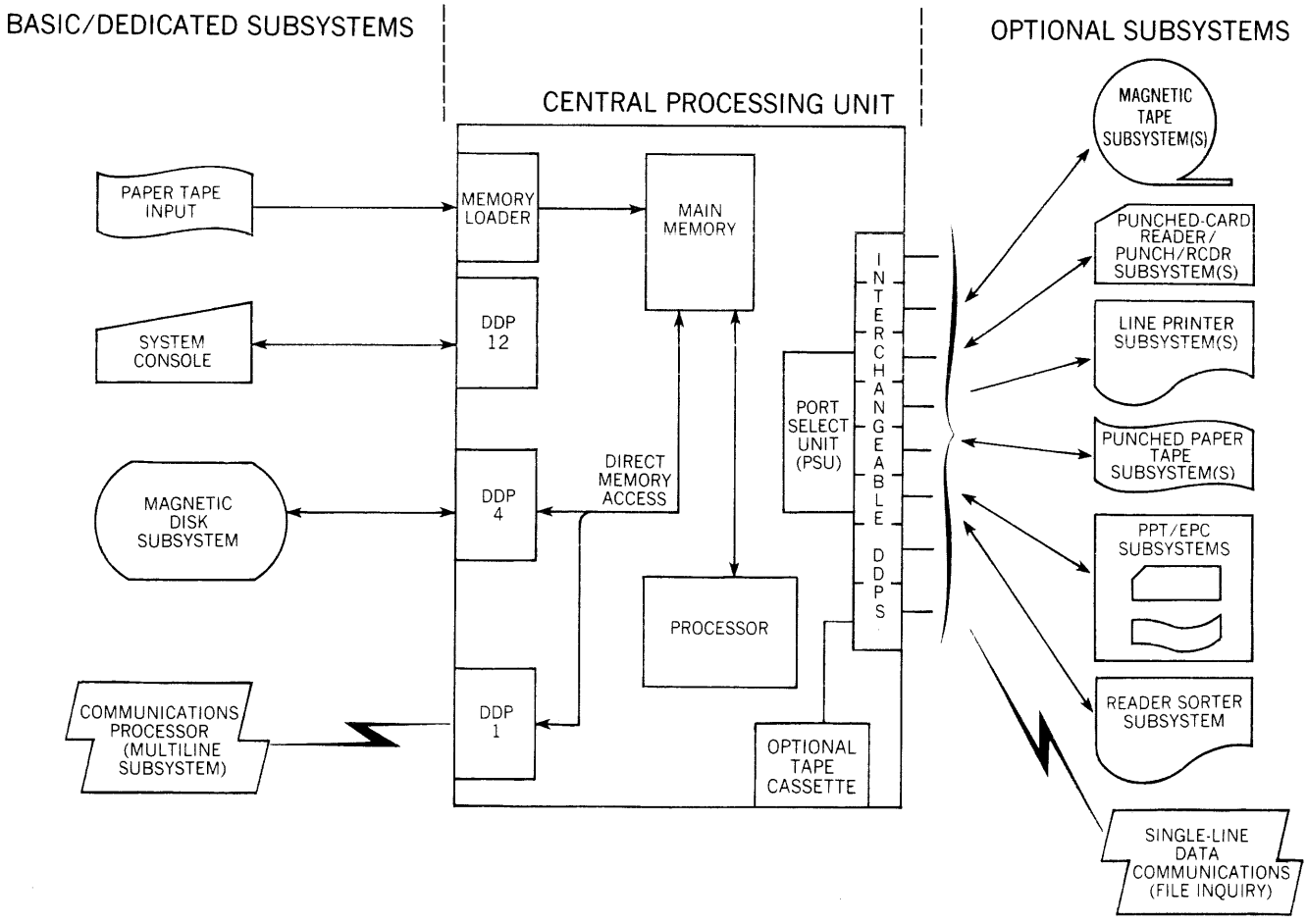



**Figure 1-2. B 720 System Interface Configuration**

**Table 1-1. B 720 Equipment Characteristics and Capabilities**

| Equipment | Characteristics and Capabilities |
|---|---|
| B 720 Central Processing Unit (CPU) | Main (semiconductor memory of up to 96K 8-bit bytes (48K-words).<br>Memory expansion from 32K-bytes (min.) to 96K-bytes (max.) in 8K-byte increments.<br>Operating rate of 1 MHz.<br>I/O interface of up to 11 I/O ports (DDP'S); five interchangeable ports are provided in basic processor.<br>Up to eight interchangeable (common) I/O ports (three in basic processor, five in I/O expansion module)<br>Integral photoelectric memory loader; optional integral magnetic tape cassette. |
| B 9343-22 or B 9343-42 Console | Electronic Keyboard.<br>64 characters (B 9343-22) or 94 characters (Katakana, B 9343-42).<br>26-in. forms transport (handler).<br>Optional 64-character sets. |
| A 9480-11 or A 9480-12 Disk Cartridge Drive | Single (A 9480-11) or dual (A 9480-12) drive capability.<br>Replaceable disk cartridges with:<br>14-in. diameter coated disk.<br>200 tracks (plus 3 spares) per surface.<br>100 tracks per inch (TPI) recording.<br>32 segments per track (180 bytes per segment).<br>1,152M-bytes per surface; total of 2.304M-bytes per disk.<br>4.6M-byte total recording capacity (two disks).<br>Data access time of 14 msec. (min) to 125 msec. (max).<br>1.55M-bits per sec. data transfer rate.<br>1500 ±20 rpm disk rotation speed.<br>60 msec. average seek time.<br>20 msec. average latency. |
| A 9481-11 or A 9481-12 Disk Cartridge Drive | Single (A 9481-11) or dual (A 9481-12) drive capability.<br>Replaceable disk cartridges with:<br>14-in. diameter coated disk.<br>400 tracks (plus 6 spares) per surface.<br>200 tracks per inch (TPI) recording.<br>32 segments per track (180 bytes/segment)<br>2.304M-bytes per surface; total of 4.608M-bytes per disk).<br>9.216M-byte total recording capacity (two disks).<br>Data access of 14 msec. (min.) to 125 msec. (max).<br>1.55M-bits per sec data transfer rate.<br>1500 ±20 rpm disk rotation speed.<br>60 msec. average seek time.<br>20 msec. average latency. |

**Table 1-1. B 720 Equipment Characteristics and Capabilities (Cont)**

| Equipment | Characteristics and Capabilities |
|---|---|
| A 9247-2/3 or A 9247-12/13 Line Printer | 400 or 750 line-per-minute printing rate.<br>120 print position (132 print-position option).<br>Standard 48-character set (optional character sets available).<br>20-IPS paper slew rate.<br>Two-channel or 12-channel (optional) format control tape.<br>Self-aligning forms. |
| A 9249-2/3 Line Printer | 85, 160, or 250 line-per-minute printing rate.<br>132 print positions; 10 characters per inch.<br>Standard 48-character set (64 or 94 character set optional).<br>Two-channel or 12-channel (optional) format control tape.<br>Self-contained 132-character buffer. |
| A 9114-1 Card Reader | 80-column card media.<br>200-card per minute reading rate.<br>350-card input hopper capacity.<br>350-card output stacker capacity.<br>Table-top mounting. |
| A 9119-1 Card Reader | 96-column card media.<br>300-card per minute reading rate.<br>600-card input hopper capacity.<br>600-card output stacker capacity.<br>Table-top mounting. |
| A 9490-21 or A 9490-25 Magnetic Tape Cassette Drive Unit | Optional integral (A 9490-21) or external (A 9490-25) installation.<br>Dual-gap read-write head.<br>Spindle-controlled, 10 or 30 IPS reel drive.<br>Photoelectrically controlled tape speed.<br>Dual-track recording (one data track; one clock track).<br>Data is NRZI encoded at 800 BPI.<br>100 characters per inch (bit serial).<br>280-ft. cassette recording capacity.<br>10-IPS nominal read/write speed.<br>30-IPS nominal fast forward or rewind speed. |
| A 9491-2 Magnetic Tape Unit | Table-top mounting.<br>Nine-track tape.<br>Dual-gap read-write heads.<br>12.5 IPS read/write speed.<br>50-IPS rewind speed.<br>Data is NRZI encoded at 800 BPI.<br>600-foot tape reel capacity.<br>10,000 characters per second transfer rate. |

**Table 1-1. B 720 Equipment Characteristics and Capabilities (Cont)**

| Equipment | Characteristics and Capabilities |
|---|---|
| A 9418-2 Card Reader/Punch/Data Recorder | 80-column card media.<br>200-card per minute reading rate.<br>45-card per minute card punching and printing rates.<br>600-card primary input hopper capacity.<br>400-card secondary input hopper capacity.<br>Two 400-card capacity output stackers. |
| A 9419-2/6 Card Reader/Punch/Data Recorder | 96-column card media.<br>300-card per minute reading rate.<br>60-card per minute card punching rate.<br>60-card per minute card printing rate.<br>600-card primary input hopper capacity.<br>400-card secondary input hopper capacity.<br>Two 400-card capacity output stackers. |
| A 9222-1 Paper Tape Punch | Punches tape, or edge-punches cards.<br>Reads 5, 6, 7, or 8 channel coded tape or cards at 40 code-per-second rate.<br>Photoelectric sensing.<br>Handles reeled, strip, or fanfold 11/16 in. or 1-in. tape.<br>Handles individual, fanfold, or Mylar-reinforced edge-punched cards.<br>Supply and rewind mechanism for roll tape. |
| A 9122-1 Paper Tape Reader | Punched paper tape or edge-punched card media.<br>Punches 5, 6, 7, or 8 channel codes at 40 code-per-second rate.<br>Tape-supply holder with 8-inch roll capacity.<br>Power-driven takeup reel.<br>Handles card widths from 3 to 5 inches.<br>250-card supply stacker. |
| A 9135 or B 9136 Reader Sorter | Magnetic-ink or optical character recognition capabilities (MICR/OCR).<br>Up to 16 document pockets in 4-pocket increments.<br>Handles 900 documents per minute.<br>Mis-sort and double-document detection.<br>17.5-inch input hopper capacity.<br>3.5-inch capacity stackers. |

d. Line Printer Subsystems.
e. Reader Sorter (Character Recognition) Subsystem.
g. Magnetic-tape unit/cassette subsystems.

A mixture of devices, or multiples of types, may be used in the system; however each separate device interface requires a separate corresponding device I/O control. (Refer to Table 1-2.)

Although the interchangeable ports can be allocated to any device subsystem, certain ports are normally allocated to certain devices because of I/O service priority classifications. Also, certain devices (such as the Reader-Sorter are restricted to installation in an I/O expansion port).

In typical operational applications, a device subsystem is used and controlled individually (such as magnetic tape or reader-sorter), or configured with other related device subsystems to form low-speed, medium-speed, and/or high-speed subsystems. For example, a low-speed subsystem might consist of the following peripheral devices with associated I/O controls:

a. A 9114-1 Card Reader (80-column card media, 200 CPM rate).
b. A 9249-1 Line Printer (90 LPM rate).
c. A 9122-1 Paper Tape Reader (40 code-per-second rate).
d. A 9222-1 Paper Tape Punch (40 code-per-second rate).

A medium-speed subsystem might consist of the A 9418-2 and A 9419-2 Card Reader/Punch/Data Recorder devices and their controls. These devices are 80-column/200 CPM and 96-column/300 CPM card devices, respectively.

A high-speed subsystem might consist of the A 9247-2/12 Line Printer (400 LPM rate) and the A 9119-1 Card Reader (96-column card media, 300 CPM rate) and their associated controls.

The following paragraphs provide general descriptions of each type of peripheral subsystem.

## CONSOLE SUBSYSTEM

The basic or minimum system includes a B 9343-22 (64-character) or B 9343-42 (Katakana 94-character) console which serves as the system operating console and a direct-entry peripheral. The selected console is interfaced with the processor through the B 346 console I/O control (64-character) or the B 346-1 Console I/O Control (94-character) though dedicated port 12 (highest service priority).

Each type of console has an electronic keyboard, a 26-inch front-feed forms transport (handler), and a set of program-select keys. Optional 64-character sets are available in addition to the standard 64-character set supplied with the B 9343-22 console.

Only one console subsystem is used in the system.

## MAGNETIC DISK SUBSYSTEM

The basic or minimum system includes an A 9480-11/12 or A 9481-11/12 Disk Cartridge drive unit interfaced with the processor through a B 489-2 Cartridge Disk Drive Control. Each model is capable of handling one or two disk cartridges (single drive is 11; dual drive is 12). A total magnetic disk storage capacity of 9,216,000 bytes is obtained when the B 9481-12 unit is used. The B 720 may use a maximum of four disk units and a total of eight drives (spindles) through a multiplexer.

Only one disk control, dedicated to processor port 4, can be installed in the system.

## MAGNETIC TAPE SUBSYSTEM

Magnetic tape storage capabilities can be added to the system in a number of optional forms, involving the use of the A 9490-21 or A 9490-25 Magnetic Tape Cassette Unit and/or the A 9491-2 Magnetic Tape Unit. The A 9490-21 is available for installation as an integral part of the processor cabinet, while the A 9490-25 is the external desk-top version. Either or both can be used; however each requires a B 392-1 (internal) and B 392 Magnetic Tape I/O control in a selected interchangeable port (DDP).

The A 9490-21/25 has a dual-gap read-write head, dual-track (one data, one clock) recording capability, and cassette drive speeds of 10 IPS (forward) and 30 IPS (rewind).

The B 9491-2 Magnetic Tape Unit is an external reel-type, table-top mounted device that provides a nine-track recording capability, 12.5-IPS forward speed, and 50-IPS rewind speed. This unit is interfaced through a B 391 Magnetic Tape I/O control installed in an interchangeable port. A 10,000 character-per-second data transfer rate is possible with the A 9491-2 unit.

## PUNCHED-CARD READER SUBSYSTEMS

Either or both 80-column and 96-column punched-card reader subsystems are available for use in the system. The A 9114-1 Card Reader is an 80-column device with a 200 card-per-minute rate and a 350-card hopper/stacker capacity. This table-top device is interfaced through a B 111 Card Reader Control in an interchangeable DDP and is normally used in low-speed subsystem applications.

The A 9119-1 Card Reader provides the 96-column and handling capability and has a 300 card-per-minute rate and 600-card hopper and stacker. This table-top device is interfaced through a B 311 Card Reader Control in an interchangeable DDP and is normally used in high-speed subsystem applications.

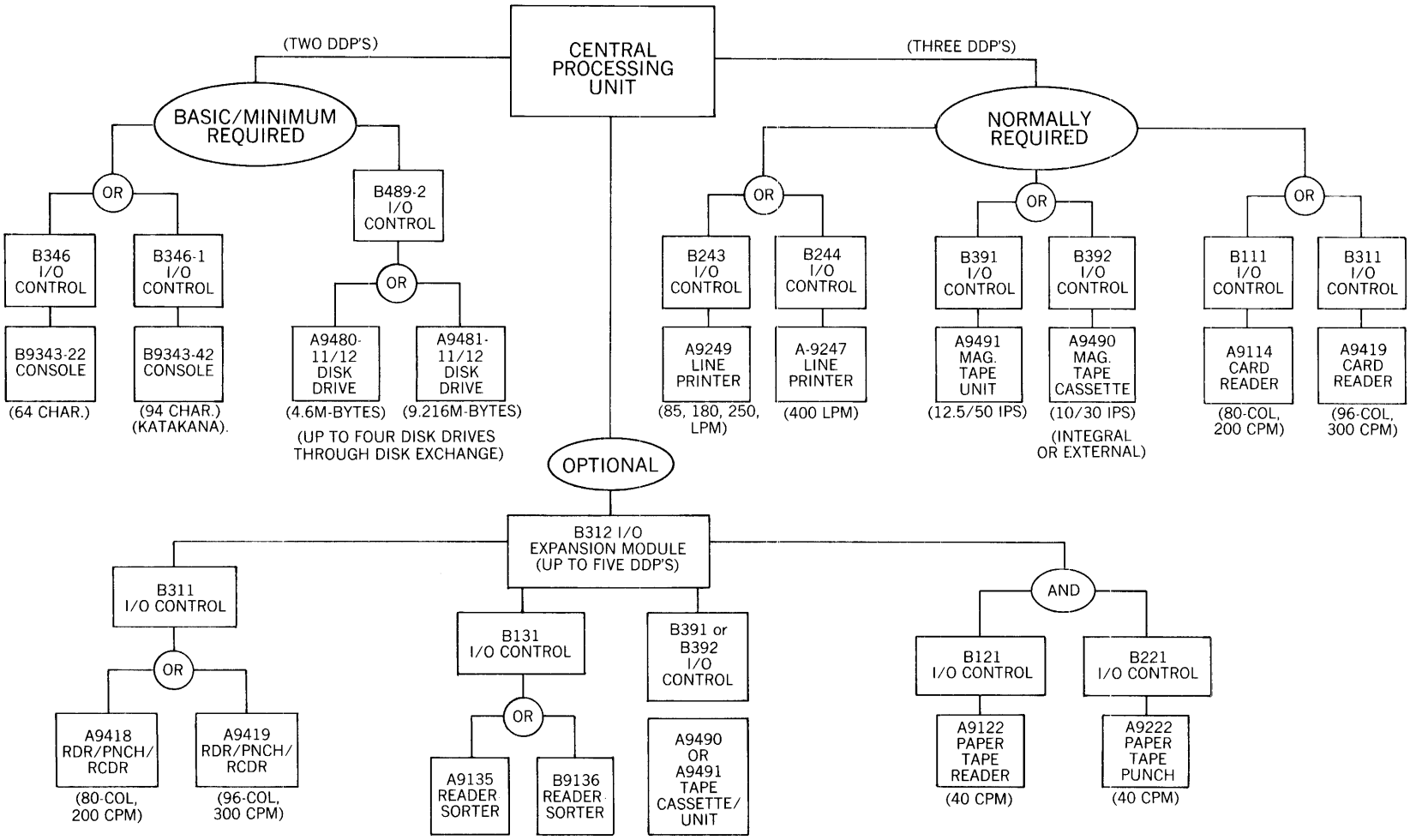



**Figure 1-3. Peripheral Device Subsystem Configurator**

**Table 1-2. B 720 I/O Controls**

| I/O Control | Peripheral Interface | Notes |
|---|---|---|
| B 346 Console Control | B 9343-22 Console | Required for 64-character electronic keyboard; dedicated to port 12 of processor. |
| B 346-1 Console Control | B 9343-42 Console | Required for 94-character (Katakana) electronic keyboard; dedicated to port 12 of processor. |
| B 489-2 Cartridge Disk Drive | B 9480-11/12 or B 9481-11/12 Disk Cartridge Drive | Dedicated to processor port 4. |
| B 351-1 Single-Line Control (SLC) | Single-Line data communications interface (direct or through data set/ modem) | Occupies two interchangeable processor ports (DDP's). |
| B 352 Communications Processor | Communications Processor interface (direct of through data sets/modems). | Dedicated to processor port 1; must be installed in modular I/O expansion rack. (Refer to description of Data Communications Subsystem.) |
| B 111 Card Reader Control | A 9114-1 Card Reader | For 80-column card media; installed in interchangeable processor port (DDP). |
| B 243 Line Printer Control | A 9249 Line Printers (1, 2, 3) | Installed in interchangeable processor port (DDP). |
| B 244 and B 244-X Line Printer Control | A 9247 Line Printers (2, 3, 12, 13) | Installed in interchangeable processor port (DDP). Optional EBCDIC and KATAKANA code sets.) |
| B 391 Magnetic Tape Unit | A 9491-2 Magnetic Tape Unit | Installed in interchangeable processor port (DDP). |
| B 392 Magnetic Tape Cassette Control | A 9490-25 Magnetic Tape Cassette Unit | Used for external tape unit installation; installed in interchangeable processor port (DDP). |
| B 392-1 Magnetic Tape Cassette Control | A 9490-21 Magnetic Tape Cassette Unit | Used for processor cabinet (integral) installation of tape unit; installed in interchangeable processor port (DDP). |
| B 131 Reader Sorter Control | A 9135 or B 9136 Reader Sorter | Installed B 312 I/O expansion module; restricted from installation in basic system interchangeable DDP's. |
| B 121-1 Paper Tape Reader Control | A 9122-1 Paper Tape Reader | Installed in interchangeable processor port (DDP). |
| B 221 Paper Tape Punch Control | A 9222-1 Paper Tape Punch | Installed in interchangeable processor port (DDP). |
| B 311 Card Reader/Punch/ Recorder Control | A 9418 RDR/PNCH/RCDR (80-col), A 9419 RDR/PNCH/ RCDR (96 col), or A 9119 Card Reader (96 col) | Handles 80 or 90 column card formats; installed in interchangeable processor port (DDP) |

## PUNCHED PAPER TAPE/EDGE-PUNCHED CARD (PPT/EPC) SUBSYSTEMS

Punched paper tape and edge-punched card reading and punching capabilities are provided by the A 9122-1 Paper Tape Reader and A 9222-1 Paper Tape Punch, respectively. These units are small, compact, desk-top or free-standing units that read and punch, respectively, 5, 6, 7, or 8 channel coded tapes or cards at a 40 code-per-second rate.

The A 9122-1 is interfaced with the processor by the B 121-1 Paper Tape Reader I/O Control, while the A 9122-1 is interfaced by the B 221 Paper Tape Punch I/O Control. Each type of control can be installed in an interchangeable processor DDP. Either, both, or multiples of these subsystems can be used in low-speed subsystem applications.

The A 9122-1 Paper Tape Reader has a photoelectric sensing mechanism and can handle reeled, strip, or fanfold punched paper tape. The A 9122-1 also handles individual, fanfold, or Mylar reinformed edge-punched cards. Supply and rewind mechanisms are provided for roll tape.

The A 9222-1 Paper Tape Punch has a tape-supply holder with an 8-inch roll capacity, a power-driven tapeup reel, and a 250-card supply stacker. Card widths from 3 to 5 inches can be handled.

## PUNCHED-CARD READER PUNCH RECORDER SUBSYSTEMS

Combination punched-card reading, punching, and recording capabilities in medium-speed subsystem applications are provided by the A 9418-2 and A 9419-2/6 Card Reader/Punch/Data Recorder Units. The A 9418-2 handles standard 80-column cards and is interfaced with the processor through the B 311 Reader/Punch/Recorder I/O control. The A 9418-2 can read punched cards at a 200 card-per-minute rate and/or punch and print cards at a 45 card-per-minute rate. A 600-card primary input hopper, a 400-card secondary input hopper, and two 400-card output stackers are provided on the A 9418-2.

The A 9419-2/6 (2/6 pocket) Card Reader/Punch/Data Recorder handles 96-column cards and also is interfaced with the processor by the B 311 I/O control. The A 9419 has a reading rate of 300 cards-per-minute and punching/printing rates of 60 cards per minute. A 600-card primary input hopper, a 400-card secondary input hopper, and two 400-card output stackers are provided on the A 9419.

## LINE PRINTER SUBSYSTEMS

Output printing capabilities of various speeds and formats are provided by the A 9247 and A 9249 Line Printers available for use on B 720 Systems. The A 9247 Line Printer is interfaced through a B 244 Line Printer Control, while the A 9249 Line Printer is interfaced through a B 243 Line Printer Control. Both types of controls are installed in interchangeable I/O ports.

Normally, one line printer subsystem is used in the B 721 System. The A 9249 Line Printer is used in low-speed subsystem applications; however, if it is the only line printer used in the system, it is configured as a high-speed device. The A 9247 is normally used in high-speed subsystem applications with the A 9119 Card Reader.

The A 9247-2/3 and A 9247-12/13 Line Printers have 400 and 750 line-per-minute printing rates, respectively, and provide 120 print positions. (A 132 print-position option is available.) The A 9247 is provided with a standard 48-character set, but optional character sets are available. A standard two-channel (or optional 12-channel) format control tape is provided. The B 244 I/O control interfaces those printers using ASCII code. B 244-1 through B 244-6 controls provide interfaces with EBCDIC and KATAKANA variations.

The A 9249 Line Printer has 85, 160, and 250 line-per-minute rates and 132 character positions. A standard 48-character set is provided; optional 64 and 96 character sets are available. A standard two-channel (or optional 12-channel) format control tape is provided. The A 9249 has a self-contained 132-character buffer.

## READER SORTER SUBSYSTEM

The Reader Sorter subsystem is a character-recognition subsystem designed for sorting and processing magnetic ink or optically encoded documents (MICR or OCR). This subsystem consists of the A 9135 or B 9136 Reader-Sorter interfaced through the B 131 Reader-Sorter Control.

The B 131 Reader-Sorter Control must be installed in an interchangeable port provided by the B 312 I/O Expansion Module. That is, an I/O Expansion Module, with the required cabling and cooling options, must be included in the CPU when installing a reader-sorter subsystem.

The Reader-Sorter has both magnetic ink and optical character-recognition capabilities and can handle 900 documents per minute. Up to 16 document sorting pockets may be installed on the Reader-Sorter (in four-pocket increments). Mis-sort and double-document detection capabilities are provided, along with a 17.5-inch capacity input hopper and 3.5-inch capacity stackers.

Only one Reader-Sorter Subsystem may be used on the system. A complete system includes a reader-sorter subsystem, processor and console, disk, and line printers. Section 5 provides a general description of the Reader-Sorter Subsystem. Refer to the B 700 Reader-Sorter Subsystem Reference Manual, form 1082500, for a complete description of the system.

## DATA COMMUNICATIONS SUBSYSTEMS

There are two types of data communications subsystems systems available for B 720 Systems: a programmable communications processor subsystem, and a single-line data communications subsystem. Figure 1-4 is a data communications subsystem configurator.

## COMMUNICATIONS PROCESSOR SUBSYSTEM

An optional programmable communications processor is available for servicing up to four communications lines in the half-duplex or full-duplex modes. This subsystem is allocated an entire I/O module in the CPU I/O expansion rack and consists of basic (required) and optional elements. Only one communications processor may be used in the system, and it is dedicated to I/O port 1 (DDP1).

The major programmable controlling and processing element in the subsystem is the B 352 Communications Processor. The B 352 consists of an I/O expansion rack module wired and configured to accept the following required or optional items:

a. Up to four B 651 Line Adapters (one for each communications line to be serviced).

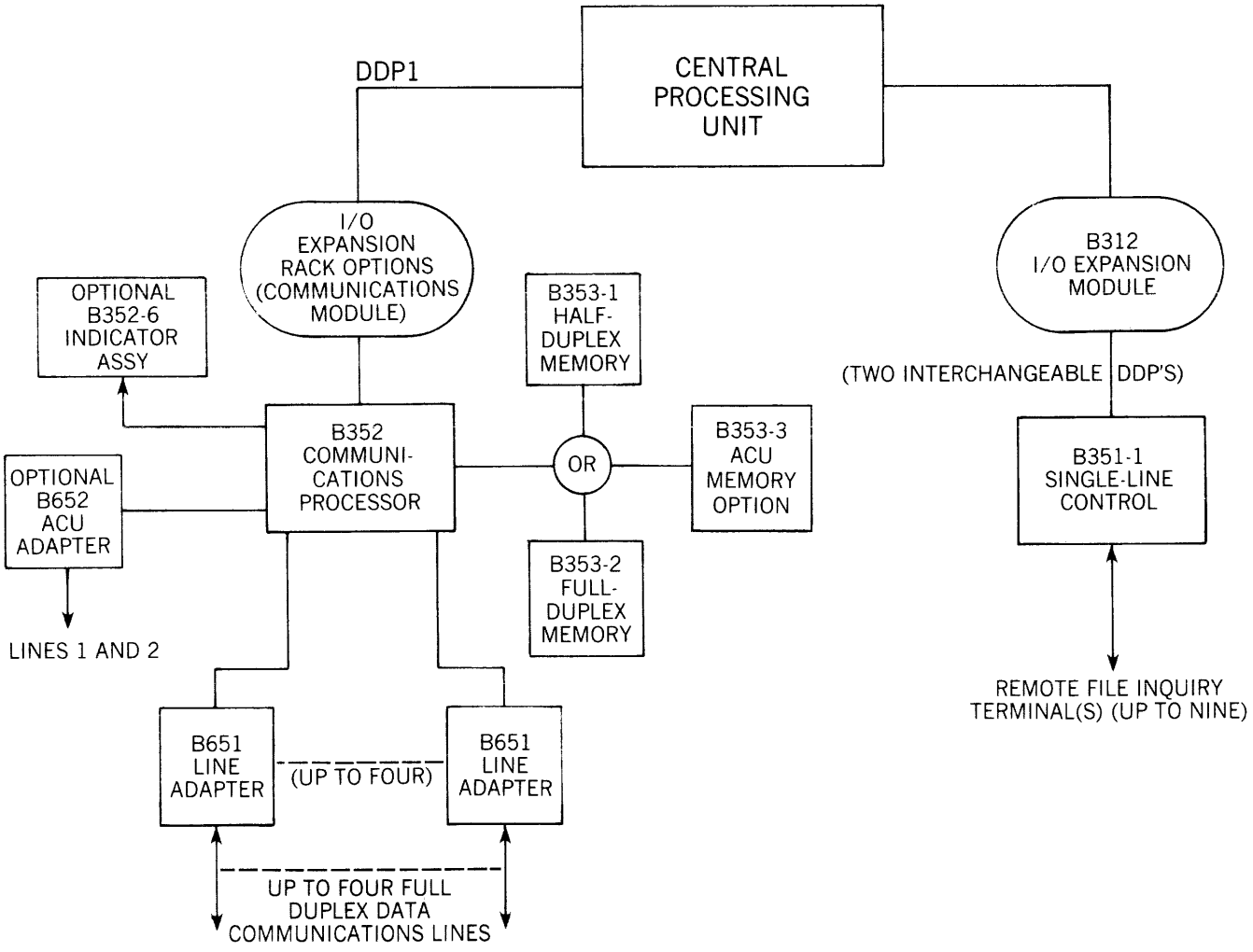



**Figure 1-4. Data Communications Subsystem Configurator**

b. The B 353-1 four-line half-duplex option, or the B 353-2 four-line full-duplex option. (Installation of one selected option is required for operation; a B 353-3 memory option is available for use with the ACU option.

c. One data set interface cable for each line adapter-to-data set interface, or a direct-connect kit for each interface to be directly connected to a terminal (TDI interface).

d. A B 652 ACU (Automatic Calling Unit) option, which provides the capability of automatic dialing on two of the four communication lines.

e. The optional B 352-6 Data Communications Console Indicator Assembly, which provides transmit and receive monitoring indicators for each communications line serviced.

The B 352, together with the mentioned options and cabling, comprise the communications module. The B 352, unlike the other I/O controls, is programmable and executes program instructions to control the flow of data between the system processor and data communications channels. This capability, which normally is implemented by execution of a firmware program in the system processor, frees the system processor to accomplish more useful data processing functions. The system processor only monitors the operational status of the B 352 in this application.

The subsystem can handle synchronous or asynchronous data transmissions and can accommodate two basic types of interfaces: data set/modem (RS-232-C or CCITT) and two-wire direct (unbalanced). Combinations of interfaces (up to four) may be used; a B 651 Line Adapter is required for each interface.

All standard data sets/modems may be interfaced by a B 651 Line Adapter.

A complete functional description of the data communications subsystem is provided in Section 4.

## SINGLE-LINE DATA COMMUNICATIONS SUBSYSTEM

The optional single-line data communications capability on B 720 Systems is provided for a special and restricted application of data communications: file inquiry operation. The file inquiry capability, which is implemented through the use of a B 351-1 Single-Line Control (SLC), permits an operator at a remote inquiry station to make disk data file inquires under user programmatic control.

The remote terminals, interfaced by the B 351-1 SLC in this application are TD 700, TD 800, and TD 801 Terminal Display Units. Only one B 351-1 SLC, which occupies two interchangeable I/O ports in the B 312 I/O expansion module, may be used in the system. Up to nine terminals may be interfaced by the SLC (mixture of types if desired).

The terminals are interfaced with the system processor over a private (direct-connect), asynchronous, 9600-baud communications lines by the B 351-1 SLC. A complete functional description of file inquiry operation is provided in Section 4.

# SECTION 2

# PROGRAMMING SYSTEM DESCRIPTION

## GENERAL CONCEPTS

The B 720 programming system incorporates the latest data processing concepts, such as soft (interpretive) structure and dynamic interpreter configurator. The B 720 Central Processing Unit (CPU) is referred to as the "host machine." Interpreter and system programs are loaded into the host machine to form a "virtual machine." Therefore, there can be many variations of virtual machines, using one host machine, if a variety of interpreters are available. An interpreter may be oriented toward general purpose applications and thus can be used with a variety of programs. However, some interpreters may be highly oriented to a particular program application.

## HIGH-LEVEL LANGUAGES

The B 720 can be described as a system that executes programs under control of "microinstructions." It is impractical for applications programmers to write microprograms; therefore, various higher-level languages have been developed for customer use on the system. It is not the intent of this reference manual to describe these higher-level languages, but to describe the relationship that exists between the microprogram and the higher-level languages.

S (Secondary)-language instructions are intermediate instructions which are equivalent to the machine language of a conventional system. For each S-instruction there exists a string of microinstructions which interpretively executes the function specified by that S-instruction. It must be remembered that the S-instruction does not directly cause the hardware to perform a function.

S-instructions may completely specify a compiler level language. Burroughs Corporation has defined the S-instructions and has written the interpreters and compiler programs for the COBOL, RPG, and NDL languages. These compiler programs generate the S-language instructions to perform the various operations specified by the higher-level language.

## INTERPRETERS

In addition to the S-language program (compiler user program), another program referred to as an interpreter is utilized. An interpreter system has been developed which satisfies each of the higher level S-languages, as well as utility programs. It is the function of the interpreter to fetch the S-language instructions from main memory and to interpret or execute the instructions. The S-instructions are decoded, and a series of microinstructions are executed to cause the hardware to perform the function specified by the S-language instruction. When the execution of a series of microinstructions (representing an S-instruction) is completed, the interpreter fetches the next S-language instruction and the operation continues in this manner. The series of microinstructions used for each S-language instruction may be stored in main memory or called from disk.

The S-language programs and the interpreter system are located and loaded from disk under control of the system loader. On completion of the load, the system loader then passes control to the S-language program.

## DYNAMIC INTERPRETER CONFIGURATOR

The dynamic interpreter configurator is based on the noncommitment of resources until the actual need for these resources arises. The proper utilization of resources enables the user to get the maximum system throughput per dollar investment.

System configuration occurs at program load (from disk) time. The system loader is provided with information about the required subsystem device controllers, subsystem buffers, number of SPM words, and the actual interpreter structure needed by the S-level program being loaded. The system loader then constructs the required interpreter and allocates the remaining memory for the required buffers, SPM, and user program.

User (S-level space is within the area specified by the program base and limit registers and is divided into two portions: overlayable and nonoverlayable (or resident) sections. Compiler generated code is completely independent of absolute memory locations and consequently is not restricted to particular areas of memory.

Proper control of input/output resources and a flexible system architecture are other features which provide for proper utilization of resources on the system.

## BREAKOUT

At any time during execution, an S-level program may be interrupted to enable the loading and execution of a different program. The interruption is called "breakout", and the program along with the virtual machine registers are saved on the disk. After the desired program is executed, execution of the interrupted program may be resumed at the point where breakout occurred. Only one level of breakout may occur; that is, only one program can be saved.

## PROGRAMMING SYSTEM STRUCTURE

The elements of the complete programming system are categorized into the following groups:

a. System Software, which consists of the operating system programs essential to the primary startup and operation of the system.

b. Program product development aids (such as NDL), which facilitate the generation of application programs by a user.

c. Application programs, which are designed for specific user functions (such as character recognition and sorting, production control, and item processing).

## SYSTEM SOFTWARE

System software, or operating system programs, consists of initialization programs, interpreters, and system software functional utility programs. The B 700 System Software Operation Guide, form 1082492, contains descriptions of the application and operation of these system software elements.

*INITIALIZATION PROGRAMS*

Initialization programs, consisting of cold start and warm start programs, provide the means to bring the system up to normal operating status. Cold start programs consist of the following:

a. The Disk Primer/Cold Start (DPCS) program, which prepares a new system disk cartridge, copies a backup cartridge, and loads the system loader from disk to main memory.

b. Bootstrap loaders which provide the minimum control required to load the Disk Primer/Cold Start program from four types of input media to main memory.

When the system loader program is loaded by DPCS, it is used to define the I/O environment and load the interpreter from disk to main memory. At the conclusion of this process, the system attains the normal operating status. DPCS consists of the following four routines used in the cold-start process:

a. INIT, which initializes a disk cartridge to the interpreter format by providing a variable number of protected spaces on disk for system software, plus four protected directory segments.

b. LOAD, which loads the interpreter from input media to the protected area on disk and thus creates a system disk.

c. COPY, which copies a system disk to a backup disk. Only those tracks occupied by the system software load are copied.

d. BOOT, which reads the system loader program from system disk to main memory.

SYSTEM LOADER PROGRAM

The System Loader program is used to define the system configuration. The definition process occurs in the following sequence:

a. The peripheral device and system configuration is defined.

b. Translation algorithms and associated tables are defined for those devices that require them.

c. Total memory size is defined.

The System Loader is read from the system disk under control of the BOOT routine in DPCS, or from the warm start program during a warm start or restart. For a warm start or restart, the System Loader is already resident on the system disk. In all cases, the program is stored at the same reserved starting location on disk and is read into main memory, starting at location 0000, and is given control.

WARM START PROGRAM (WSTRT)

The Warm Start program, identified as WSTRT, provides the minimum controls necessary to transfer the system loader program from disk to main memory. The WSTRT program is contained on punched paper tape and is loaded to memory by means of the integral memory loader.

*INTERPRETERS*

Interpreters are microprograms that comprise the actual machine instructions to which the equipment responds. B 720 system interpreters are designed to emulate the operation of other computer systems at the user level. Interpreters for COBOL, RPG, and NDL have been developed for use on B 720 systems.

Intrinsic segments of the interpreter firmware are provided to initialize and copy a user disk cartridge (INITD), print the contents of machine memory and registers (SYSDUMP), and trace and list each step of a COBOL or RPG program (TRACE).

*INTERPRETER FIRMWARE.*

The interpreter firmware is comprised of the following segments and routines which form the minimum interpreter necessary to load program memory and bring the system up to operating status:

a. Common interpreter segments (ACOMMON), which comprise the interpreter segments common to all software elements of the systems. ACOMMON is always resident in memory.

b. BCOMMON, which contains the elements necessary to perform the operation as specified by the user COBOL or RPG program. BCOMMON is transferred to main memory at the time that a program which requires it is loaded from disk to memory for execution.

c. CCOMMON, which contains the elements necessary to support the B 700 onboard COBOL Compiler and onboard NDL Compiler. CCOMMON is transferred to main memory only when required.

d. Interpreter overlays (OVERLAY), which are retained on disk until required during program execution. A part of main memory is reserved, such that only the single overlay actually in use is occupying it. The dynamic maintenance of the overlays in main memory is an automatic function of the system firmware.

e. Intrinsic segments, which are brought into memory only if actually required by the program to be executed, or if desired by the system operator to be executed. Memory not filled with relative segments is made available to the S-level program. Relative segments include those portions of the interpreter required for data communication controllers, System Dump, and Trace.

f. Fatal Error Routine (FATAL), which is used by the operating system to alert the operator to the existence of a condition that impairs proper functioning of the system. Error conditions handled by FATAL include certain hardware errors, address limit errors, and I/O errors.

*UTILITY PROGRAMS*

The B 720 System is provided with a library of functional utility programs that perform various utility operations, such as loading files from card devices to disk, or dumping program/data disk files to magnetic tape. These programs are loaded and executed like any other program. Application of a particular utility program on a system depends on the peripheral device configuration of the system. The available library of utility programs is as follows:

| Designation | Name |
|---|---|
| CDGEN | Check Digit Table Generator |
| CDLST | Card List |
| CRDLD | Load Object Program From Cards to Disk |
| DDLST | Disk Directory List |
| DFLST | Disk File List |
| DMPPG | Disk File Dump and Purge |
| DSKTP | Dump Disk Files to Tape |
| INTTP | Create Cold Start Tape From 80-Column Cards |
| ODP80 | Punch Object Program, Disk to 80-Column, R/P/P |
| ODP96 | Punch Object Program, Disk to 96-Column, R/P/P |
| O8096 | 80-Column to 96-Column Object Program Conversion |
| PTCPY | Paper Tape Copy With Verify And Fancy Punch |
| SQASH | Disk Space Reallocation |
| TPCPY | Copy Tape To Tape And Verify |
| TPDSK | Dump Tape File To Disk |
| TPLST | Tape List |

The B 700 System Operating Guide, form 1082492, provides descriptions and operating instructions for each utility program.

*GENERAL SORT PROGRAM*

Also provided with the system software is the General Sort Program (SORT), which performs all or part of the operations necessary to provide and control a sorted tag file associated with a particular master file and/or sort the master file itself. SORT has the following five selectable modes of operation:

a. Tag sort mode, which creates a tag file from the master file and sorts the file according to specified keys.

b. Record sort mode, which sorts the master file according to specified keys and purges any deleted master records.

c. Full sort mode, which performs combined functions of tag and record sorts.

d. Basic sort mode, which takes an existing unsorted tag file and sorts it according to specified keys.

e. Update sort mode, which sorts and merges an existing overflow tag file with an existing sorted tag file to create a new updated tag file. A new (empty) overflow file is opened.

Refer to the B 700 System Software Operation Guide, form 1082492, for a complete description and operating instructions.

*COBOL COMPILER*

The B 720 may implement the B 700 series onboard COBOL Compiler, which consists of an input program (B7CBL) that collects COBOL statements from various media and uses six separate processing phases to produce the compiled program. If any syntax errors are detected during input operations, certain phases of the processing sequence are bypassed. The detected errors are displayed to allow the user to perform error correction without requiring code generation.

The B 700-series COBOL Compiler allows the user the capability of compiling from 80-column cards, 96-column cards, disk files, tape files, or system console input. Source patches may be included from 80- or 96-column cards, or from the system console. The user may optionally save the resultant source file on disk, cassette, tape, 80-column cards, or 96-column cards.

Output listings are available on the wide-line printer or the console. The resulting object program is placed on disk. Refer to the B 700 COBOL Reference Manual, form 1064391, for a complete description of B 700 COBOL.

*RPG COMPILER*

The B 700 RPG (Report Program Generator) Compiler is available onboard to the users of B 720 Systems. The compiler consists of a front-end program (B7RPG), which collects RPG source from various media, and six separate programs which are phases of the RPG Compiler. As each portion of the compiler completes its program, it automatically calls the next phase. However, if there are any syntax errors, several phases will be eliminated, allowing the user to see the source errors without the expense of code generation. The time of a compile varies and depends on the number and type of RPG source submitted.

The B 700 RPG Compiler provides the user with the capability of compiling from 80-column cards, 96-column cards, disk files, tape files or console input. If either tape or disk is used as input, a merge feature is available from 80- or 96-column cards, or the console. When a merge function is used, the user optionally may save the resultant source file on disk or tape. A disk source file is always created for each compile (reserved name ($$$)).

Refer to the B 700 System RPG Reference Manual, form 1073897, for a complete description of B 700 RPG.

*NDL COMPILER*

The B 720 Network Definition Language Compiler (B7NDL) consists of an input phase that collects NDL statements from various media, and uses several processing phases to produce a compiled program and firmware tables. If any syntax errors are detected during input, certain phases of the processing sequence are bypassed and the errors are displayed. Thus, the user is permitted to perform error correction before code generation and table building are attempted.

B7NDL allows the user the capability of compiling from 80-column cards, 96-column cards, disk files, tape files, cassette files, or the console. If either tape, cassette, or disk is used as input, source patches may be included from 80-column or 90-column cards or the console. At his option, the user may save the resultant source file on disk or tape, or he may create a new card file.

Output listings are available on the line printer or on the system console. The resulting object program and tables are placed on disk.

Refer to the B 720 NDL Reference Manual, form 1080868, for a complete description.

## VIRTUAL MACHINE DESCRIPTION

### CONCEPTS AND STRUCTURE

The use of the term "virtual machine" in describing the B 720 Computer System is based on the fact that data and instruction representation, as defined for the system, is independent of the actual machine hardware used to effect the implementation of the system. The typical computer "assembler" produces a machine code representation of well-defined instructions which, when properly applied, directly influence the hardware to produce the desired results. This is not the case with a virtual machine. The virtual machine object code does not directly influence the hardware: in fact, it may bear no resemblance to the true machine code of the host machine. Instead, there exists an interface (interpreter), which translates the virtual machine code into host machine code for actual execution of the instructions.

Normally, the virtual machine programmer need not be aware of this fact. It suffices to presume that the virtual machine acts like any other machine in that its instruction mnemonics exist in one-to-one correspondence with directly executable machine code parameters. There are important points, however, which must always be remembered:

a. The program must share memory space on the system with the interpreter.

b. The interpreter itself may rely on segmentation in order to free memory for the program.

c. The interpreter is itself a program which is subject to rules and constraints.

The internal representation of data consists of character strings of eight-bit ASCII, numeric strings having eight-bit ASCII representation, and fixed-sized fields of signed BCD (four-bit) numeric digits. Hexadecimal data may be moved and compared as character strings.

### MEMORY ORGANIZATION

The main memory of the processor is functionally divided into two major areas: interpreter memory (microprogram memory, MPM) and user memory. Figure 2-1 shows the general organization and allocation of main memory.

*INTERPRETER MEMORY*

Interpreter memory, or Microprogram Memory (MPM), consists of resident machine code areas, overlayable areas, and a variable areas as shown in figure 2-1. Additional information about the MPM is presented in Section 3, System Processor Description.

*USER MEMORY*

A characteristic of earlier implementations, which remains central to the language, is the partition of user memory (program memory) into two distinct sub-memories having well-defined boundaries.

The first partition is called the scratchpad memory (SPM). SPM is limited in size (256 X 64-bit words maximum) and is intended primarily as an I/O workspace. There is no connotation of speed associated with its name. It is, in fact, host memory like any other. No executable instructions occur in SPM. SPM for a given program is only as large as required (in some cases no SPM is required), and the unused portion is available as CPM (central processor memory).

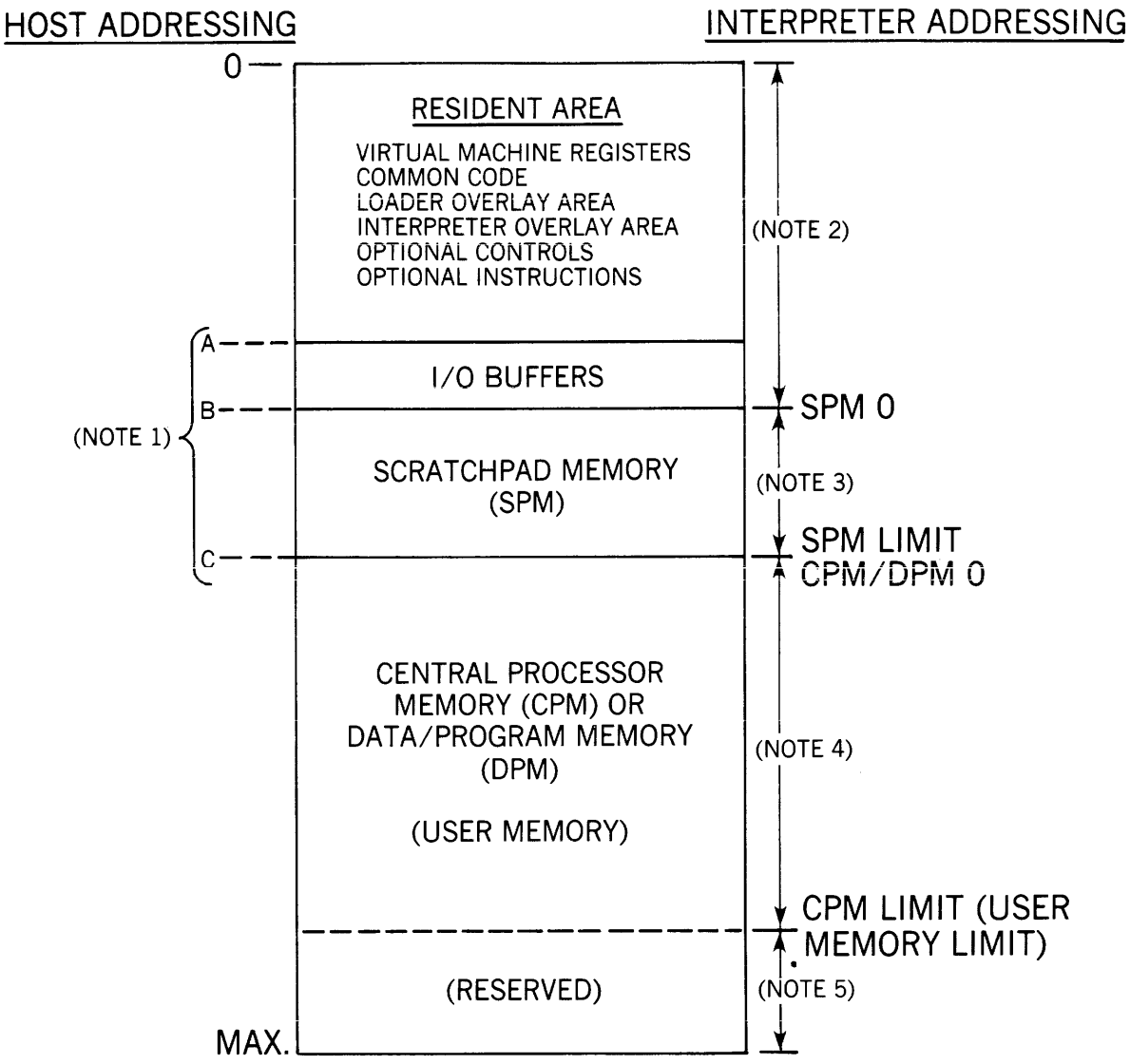
The second partition is called central processor memory, or data-program memory (CPM or DPM). The size of CPM available to the user is limited by the total host memory, minus:

a. Interpreter requirements.

b. I/O buffer requirements.

c. SPM requirements.

It is within CPM that the virtual machine program




Notes:
1. Each of these boundaries is variable, depending on implementation and program requirements.
2. Not addressable by interpreter program.
3. Only as large as required.
4. Maximum CPM required for this program.
5. This space **may** be available to the interpreter program through indexing.

**Figure 2-1. Typical Memory Allocation**

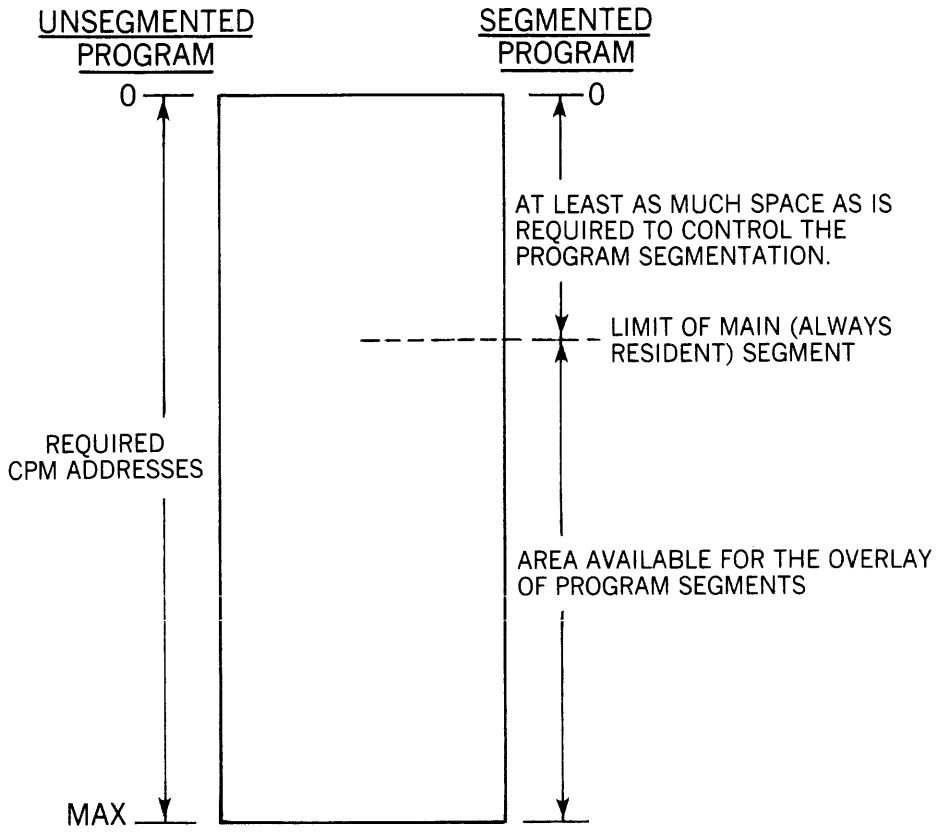



**Figure 2-2. Typical Program/Data Memory (CPM/DPM) Allocation**

instructions reside, as well as any non-I/O data areas. CPM may be segmented so that portions of the program may overlay one another and share memory on a sequential basis. Some minimal amount of always-resident object code is required in order to control segmentation.

A typical memory map is shown in Figure 2-1 (Different implementations may vary in configuration.)

The portion of memory assigned to the CPM itself may be further expanded as shown in Figure 2-2.

SPM contains only data areas and I/O descriptors. CPM may contain data areas as well as program instructions, mixed in any order, subject to the following restrictions:

a. Word address 0 is reserved and always contains a branch to the first executable instruction.

b. Data and instructions never share the same word. Each time a switch is made from data declarations to instruction coding and vice versa, an update of the location counter is made to the next word address.

## WORD ORGANIZATION

As stated previously, the virtual machine memory (both SPM and CPM) is subdivided into fixed-size units (words). These words form the basis for most data manipulations within the machine. A word consists of 64 bits which, depending on the circumstances, may be thought of as:

a. Eight 8-bit characters.
b. Eight 8-bit hexadecimal bytes.
c. Sixteen 4-bit decimal digits.
d. Sixteen 4-bit hexadecimal digits.
e. One 4-bit sign digit, plus fifteen 4-bit decimal digits.
f. Two 32-bit half-words, each containing four 8-bit characters or eight 4-bit digits, and so forth.

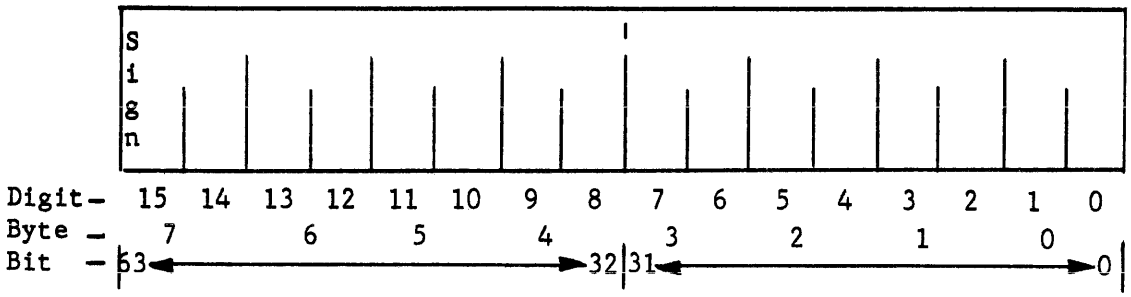
In most cases, the smallest unit of data address is the 32-bit half-word; however, certain instructions may access characters or digits or bits.

The following position numbering conventions are

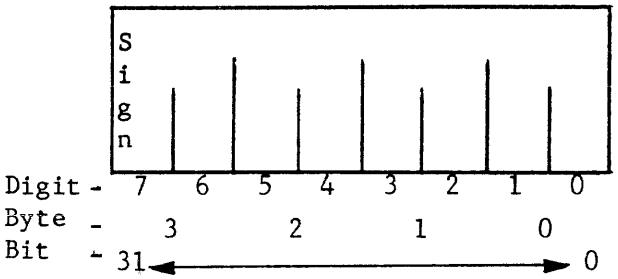
very important:

a. All numbering begins with zero.
b. All numbering within a word is right-to-left.

Thus, a full-word is numbered as follows:




A half-word is numbered as follows:



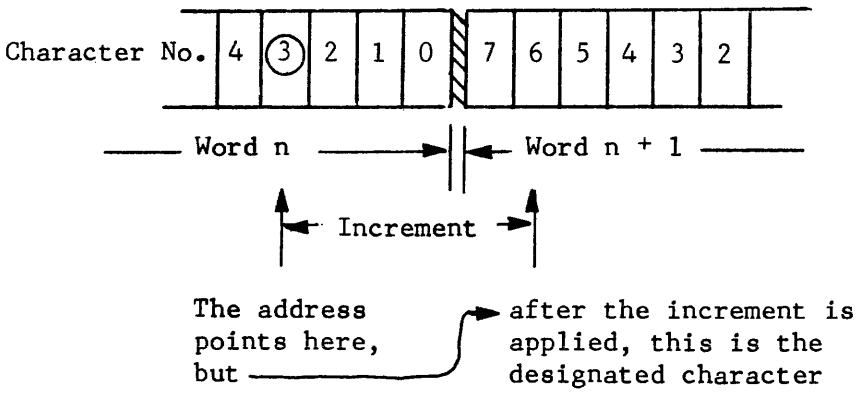

These position numbers are used when referring to specific components of a word or half-word.

Incrementing and indexing of data addresses follows these rules:

a. The first element of a group of elements has increment zero.
b. Incrementing is from *left-to-right.*

Example:

A reference address of character 3 of a word, plus an increment of five characters is as follows:

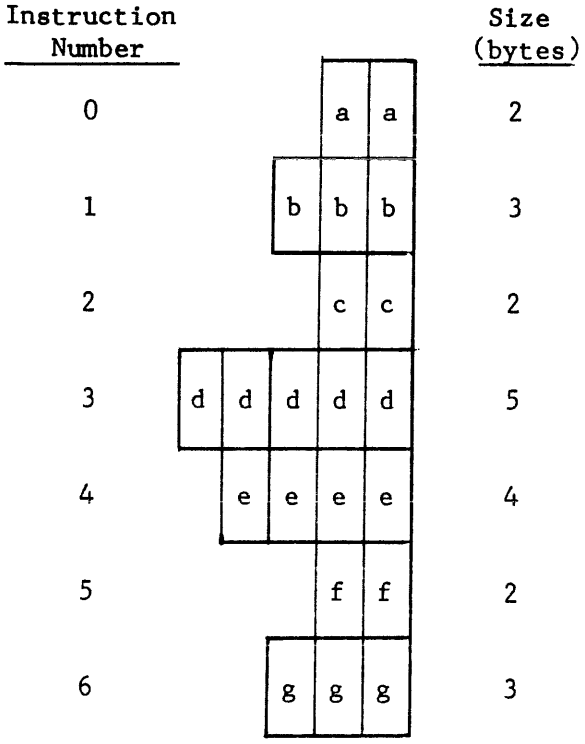


## INSTRUCTION FORMAT

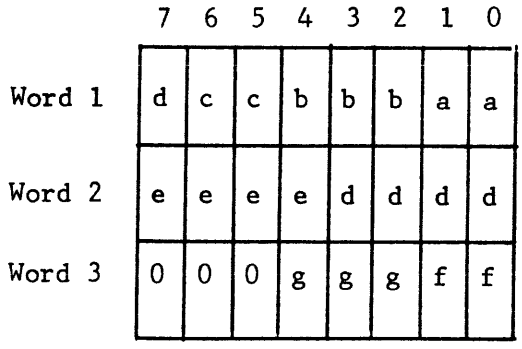
The virtual machine architecture features a variable-length instruction set consisting of 2-, 3-, 4-, and 5-byte instructions. The object code is packed into the CPM words beginning in character 0 of a word and proceeding from *right-to-left.* Instruction object code may cross full-word or half-word boundaries. This right-to-left packing, if not clearly understood, could cause much confusion during interpretation of a program memory dump printout or during decoding of program object card images.

Example:

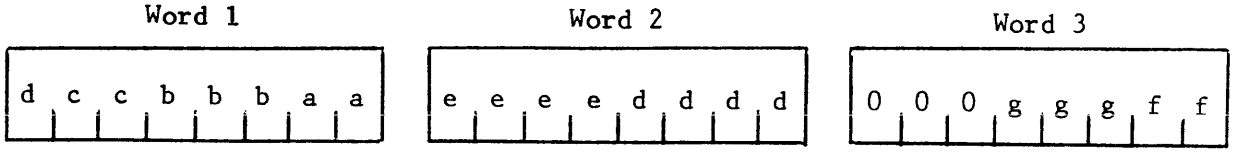
Consider the following program:




It would be packed into CPM as shown:

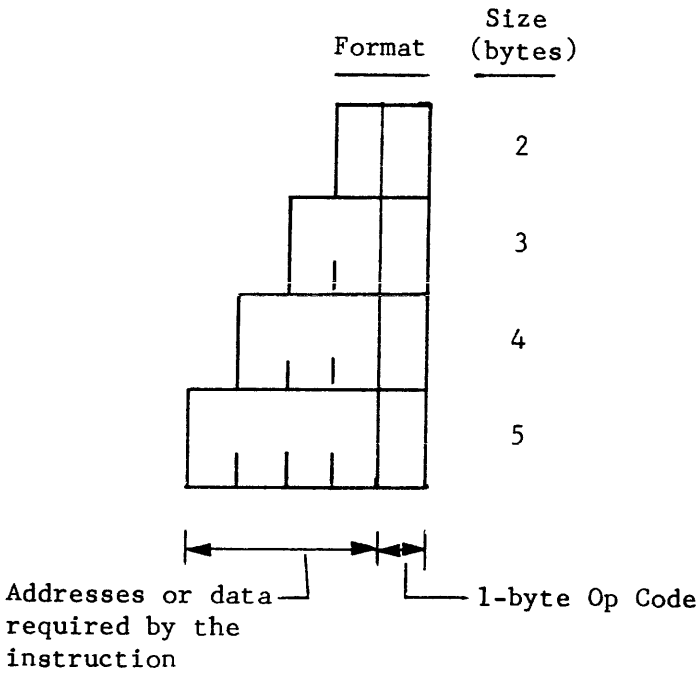


However, on a memory dump printout, the words are displayed horizontally as follows:




It can be seen that a sort of progressive effect is produced.

The instructions are made up as follows:




The size of an instruction depends upon the amount of parametric data required for its execution. Every instruction requires a one-byte operation code (OP CODE), which is the right-most byte, and is a unique bit pattern which identifies the instruction to the interpreter. Because there are eight bits in a byte, it can be seen that there is a maximum of $2^8 = 256$ unique executable virtual machine instructions.

Unlike many languages, an operation code is not unique for a given mnemonic (the four-letter name of an instruction).

In many instances, a single mnemonic will generate one of several possible operation codes, depending on the characteristics of the data involved. Appendix B provides a complete list of instruction codes and mnemonics.

The parametric data associated with a given operation code will vary according to the needs of the instruction. The most general case of parametric data is an address. The address may be of a memory location which contains data, or it may be the address of an instruction. There are several similar but distinct address formats which are defined for the virtual machine:

a. CPM data address: full-word.
b. CPM data address: half-word.
c. CPM data address: character.
d. CPM data address: digit.
e. SPM data address: full-word.
f. SPM data address: half-word.
g. SPM data address: character.
h. SPM data address: digit.
i. CPM instruction address: character.

All addressing is in binary, and is based ultimately on the word address of the entity. CPM addresses and SPM addresses are differentiated by the most significant digit of the address, which, for SPM addressing, is all binary ONE'S (hexadecimal F). For CPM addressing, the digit is any other binary pattern.

A maximum of four digits (16 bits) is available for addressing. Of these 16 bits, the least-significant two bits are normally used, on data addresses, to indicate the optional application of indexing (by adding index register contents as a positive offset) to the data address.

Address formats are denoted in documentation by a bit-for-bit representation using letters to indicate the meaning of a particular bit:

| Bit | Meaning |
|---|---|
| W | Bit is part of the word address. |
| H | Bit is used to indicate a half-word. |
| X | Bit refers to indexing. |
| C | Bit is a character position number bit. |
| D | Bit is a digit position number bit. |
| I | Bit is always true. |
| O | Bit is always false. |
| - | Bit value does not enter into the address (i.e., it is a "don't care" bit). |

For clarity, the four digits which constitute an address are separated by slashes. The typical address formats of the virtual machine are as follows:

Note

XX is to be interpreted as follows (for any instruction):

If XX = 00, no indexing is applied. (In rare cases, 00 means apply Index Register 4).

If XX = 01, apply Index Register 1.

If XX = 10, apply Index Register 2.

If XX = 11, apply Index Register 3.

a. CPM Data Address (Full-Word):
/WWWW/WWWW/WWWW/WOXX/

b. CPM Data Address (Half-Word):
/WWWW/WWWW/WWWW/WHXX/

Here bit H specifies which half-word within the given full-word address, where:
H = 0 = left half-word.
H = 1 = right half-word.

c. CPM Data Address (Character):
/WWWW/WWWW/WWWC/CCXX/ (indexed)
/WWWW/WWWW/WWWW/WCCC/ (unindexed)

Where:CCC is the character position number within the given full-word address.

d. CPM Data Address (Digit):
/WWWW/WWWW/WWDD/DDXX/ (indexed)
/WWWW/WWWW/WWWW/DDDD/ (unindexed)

Where:DDDD is the digit position number within the given full-word address. (The unindexed format is not currently used.)

e. SPM Data Address (Full-Word):
/1111/-WWWW/WWWW/WOXX/

NOTE

The maximum SPM word address is 255. The most significant digit of all ONE'S is a logical indication of SPM. Certain instructions which utilize SPM will use a short form of SPM full-word addressing consisting of word bits only:

/WWWW/WWWW/.

f. SPM Data Address (Half-Word):
/1111/-WWW/WWWW/WHXX/
Where: bit H meaning is identical with that of the CPM half-word address.

g. SPM Data Address (Character):
/1111/WWWW/WWWC/CCXX/ (indexed)
/1111/-WWW/WWWW/WCCC/ (unindexed)

Where: bits CCC give the character position number within the given full-word address.

h. SPM Data Address (Digit):
/1111/WWWW/WWDD/DDXX/ (indexed)
/1111/WWWW/WWWW/DDDD/ (unindexed)

Where: DDDD is the digit position number within the given full-word address. (The unindexed format is not currently used in the full form; the short form WWWW/WWWW/DDDD/is utilized.)

i. CPM Instruction Address (Character):
/WWWW/WWWW/WWWW/WCCC/
Where: CCC is the character position number of the operation code within the given full-word address.

Some instructions require addresses as parameters which are variations of the above formats. Note also that other implementations of the virtual architecture have required that CPM be further subdivided into "blocks" or "tracks" as well as words. Using such nomenclature, equivalent addressing is as follows:
/BBBB/TTTX/WWWW/WHXX/= /WWWW/
/WWWW/WWWW/WHXX/
Where: B bits refer to block number and T bits refer to track number.

Because SPM and CPM are distinguished by a high order address digit of 1111, it is obvious that the maximum CPM word address is:

/1110/1111/1111/1 = hexadecimal 1DFF = 7679

Few host machines, however, provide this much space for CPM. Because some address formats will pre-empt word address bits to other purposes, then not all instructions are able to address all of CPM or SPM.

## VIRTUAL MACHINE REGISTERS

The term "register" usually implies a data storage device having the capability of influencing the data stored within it. Thus, registers are used for most of the special-purpose data manipulations within a computer. Of course, they may also be used to a limited extent for general storage; however, such usage is considered insignificant for the purposes of this document.

The virtual machine architecture includes a number of registers. Because these registers are few in number, and their functions are distinct and important, they are referred to by names rather than by addresses. The names assigned to some of the registers may not be used as names by the programmer.

The most important of the virtual machine registers is the accumulator (ACUM). It serves as a storage device having the capability of:

a. Decimal arithmetic, including multiply and divide.
b. Logical shifting.
c. Field isolation.
d. Communication with the virtual machine control program.
e. Logical (Boolean) arithmetic.
f. Buffer register for low-volume input from the operator console.

In format, the accumulator is identical to a full-word of memory. It may interact, however, with half-words of memory by using only its least-significant seven digits, plus its sign digit (digit position 15). Furthermore, specialized instructions can access the accumulator in a digit mode or a character mode.

The next most important set of registers within the virtual machine is the set of four index registers (IX1, IX2, IX3, and IX4). Each of these registers is a 16-bit binary register. The primary function of IX1, IX2, and IX3 is to contain positive values to be used optionally to increment (offset) a given address (as, for instance, indexing tables or arrays). These registers may also have special functions for certain instructions. IX4 is rarely available for address incrementing; generally, it is used as a special purpose counter by a number of instructions. At times, the index registers may themselves contain memory addresses. Depending on usage, the address format may be:

a. 00WW/WWWW/WWWW/WWWH/(CPM label literal).
b. 0011/11-W/WWWW/WWWH/(SPM label literal).
c. 0000/WWWW/WWWW/DDDD/(SPM digit).
d. WWWW/WWWW/WWWW/WCCC/(CPM character).
e. 1111/-WWW/WWWW/WCCC/(SPM character).

Where a given format is to apply, it will be so specified for the instruction. When used as an increment, the content of an index register is generally in half-words.

Any arithmetic which involves an index register will be in binary mode. Subtraction is 2's-complement. Index register contents are defined to be positive, although arithmetic may generate a complement (negative) value. The detection of complemented index register contents must be programmatic.

Other registers within the machine have functions which are allied with specific instructions or groups of instructions. A complete list of virtual machine registers is provided in Appendix C.

## VIRTUAL MACHINE CONDITION FLAGS.

The virtual machine contains one 64-bit word which is dedicated to the use of named single-bit condition flags. Each digit of the word is called a group, and contains four single-bit flags with related meanings. The groups are named by a single letter; the flags within a group by a single number or letter. A flag is always referred to by both its group and flag names.

Several of the flags have predefined meanings. Most of these are set and reset by the interpreter. The remainder of the flags are general-purpose flags. Their meanings are defined by the programmer.

Any flag may be set, reset, changed (complemented), and tested by the S-level program. The possible values of a single flag, and the synonyms of these values are:

a. 0 = reset = false = off = F.
b. 1 = set = true = on = T.

A complete list of the flag groups and their flags is contained in Appendix D. The predefined flags are described below.

### *A GROUP (ACCUMULATOR FLAGS)*

The accumulator flag group is a picture of the accumulator sign digit (digit 15) which allows bit-for-bit access to the data stored in the accumulator sign. Any change to the accumulator sign digit is reflected in the A group and vice versa. These flags are:

M = per thousand (attribute of the data).
C = per hundred (attribute of the data).
S = special (special purpose indicator, errors, etc.).
\- = minus (negative value flag).

*C GROUP (COMPARISON FLAGS)*

The comparison flags are set/reset by the various compare and search instructions to indicate the results of the operations. These flags are:

H = high flag (greater than)
E = equal flag (equal to)
L = low flag (less than)
U = reserved (not presently used)

*T GROUP (TEST FLAGS)*

The test flags are set/reset by various incrementing instructions to indicate overflow conditions. These flags are:

P = console out of paper (optional).
I = index register overflow.
L = forms limit exceeded (console printer).
O = arithmetic overflow.

*K GROUP (OPERATOR CONTROL KEYS)*

The operator control key flags are set/reset by various console keyboard entry instructions to indicate the operator control key (OCK) with which the operator terminated the input. These flags are:

1 = OCK 1 (OCK I)
2 = OCK 2 (OCK II)
3 = OCK 3 (OCK III)
4 = OCK 4 (OCK IIII)

*G GROUP (I/O ERROR RECOVERY)*

The I/O error recovery flags are set/reset by the SKID instruction which is a general-purpose operator-recoverable error routine. The SKID accepts only four responses (other than "Abort Program") which it then reflects in the G group flags. The four responses are the numeric keys 1, 2, 3, or 4, depressed simultaneously with the shift key (shifted numerics).

These flags are:
1 = shifted 1 key.
2 = shifted 2 key.
3 = shifted 3 key.
4 = shifted 4 key.

Note that the actual error recovery procedure must be programmatic. SKID merely serves as a vehicle for operator communication.

The remainder of the flag groups are general-purpose flags. The groups are:

S, B, D, E, F, P, R, V, W, X, and Y.

Each group has four flags, accessed by number 1, 2, 3, or 4.

# SECTION 3

# SYSTEM PROCESSOR DESCRIPTION

## GENERAL

This section contains a description of the B 720 Central Processing Unit (CPU), which is the central operating and controlling element in the system. The physical features of the B 720 CPU are shown in Figure 3-1; Figure 3-2 is a general functional block diagram.

Functionally, the CPU consists of three major sections:

a. The memory section, which consists of a semiconductor main (shared) memory and an integrated-circuit read-only nanoprogram memory (NPM). The main memory provides both data/program memory (DPM) and microprogram memory (MPM) storage. The MPM and NPM comprise the firmware storage area of the CPU.

b. The processor section, which contains the logic and control circuitry used in performing arithmetic, processing, I/O interface, memory control, and system control functions. This section also supplies the basic internal and external timing to the CPU.

c. The I/O section, which consists of the circuitry used in controlling and interfacing I/O subsystem operations. This section also includes the memory loader and load interface elements.

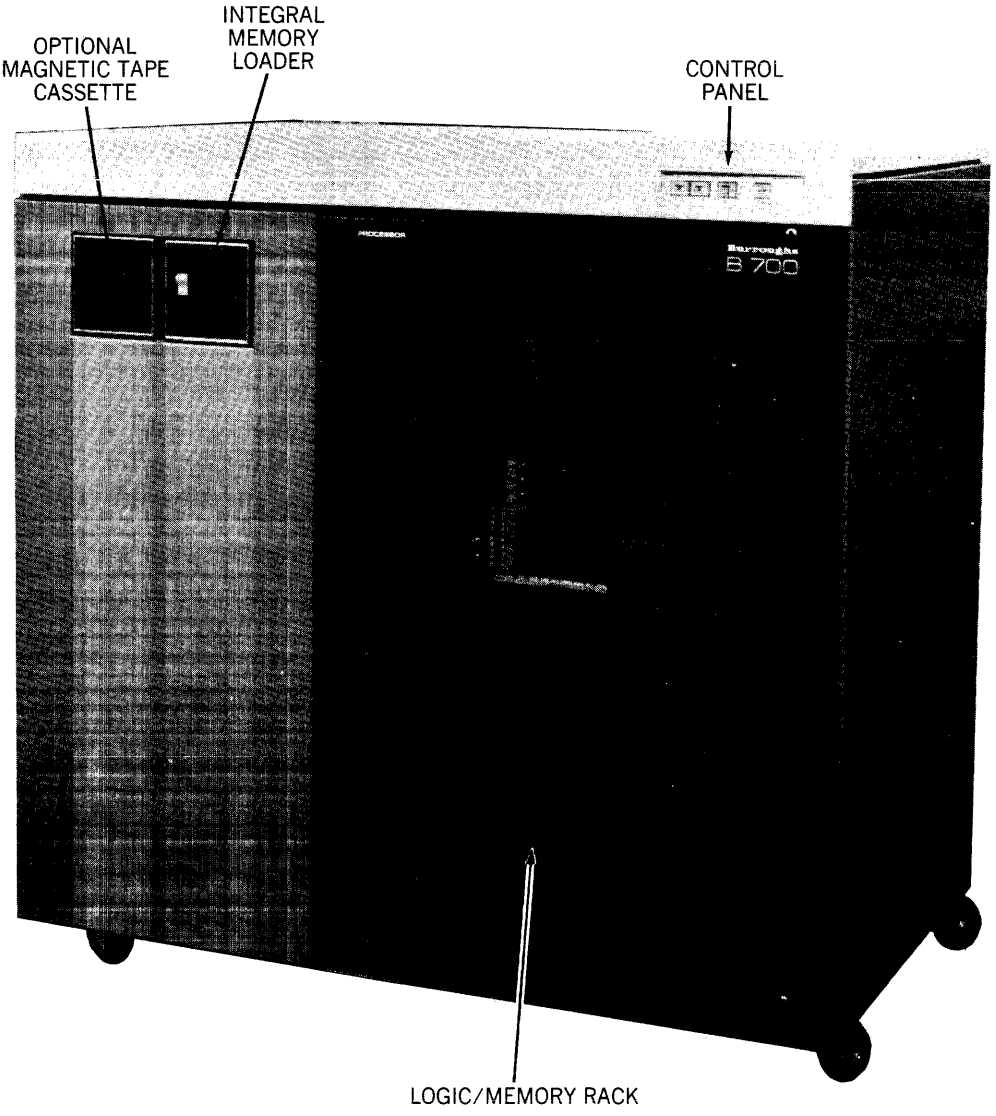



**Figure 3-1. B 720 Central Processing Unit**

A power supply group provides and controls regulated operating voltages for CPU circuitry and external devices (as required). An F.E. (Field Engineering) control section provides test and monitoring functions for equipment maintenance purposes.

The CPU is unlike conventional processors in that the basic logic operations are organized into controlled building blocks external to the processor section. The logic in these building blocks is not like the normal hard-wired control logic used in conventional processors. Instead, the control logics are replaced by control signals generated by the firmware store section. This approach adds a great deal of flexibility to the processor and utilizes a minimum amount of hardware.

The CPU is a word-addressable unit with a semiconductor main memory that is expandable, in increments of 8K bytes, up to 98K bytes. The basic system uses a minimum of 16K bytes of main memory and 512 words of nanoprogram memory. The main memory is a Random Access Memory (RAM), while the nanomemory is a Read Only Memory (ROM).

The internal clock pulse rate of the CPU is 1 MHz.

## MEMORY SECTION

The memory section of the CPU is divided into three functional memories: the data program memory (DPM), the microprogram memory (MPM), and the nanoprogram memory (NPM). (See figure 3-2.) The DPM and MPM share the semiconductor random access main memory (RAM), while the NPM resides in a separate integrated-circuit read-only memory (ROM). Functionally, the MPM and NPM comprise the firmware storage area of the CPU. The MPM stores microinstructions, and the NPM stores nanoinstructions. The DPM stores data and user programs. Section 2 describes the memory allocation.

The main (shared) memory is a modular memory with a capacity of 16 to 96 K-bytes. The minimum 32K capacity may be expanded, in 8K increments, to 96K by installing additional B 31-2 Memory Modules. A total of 12 B 31-2 modules may be used.

## FIRMWARE STORE SECTION

The firmware store section, which uses RAM integrated memory for microprogram storage and ROM

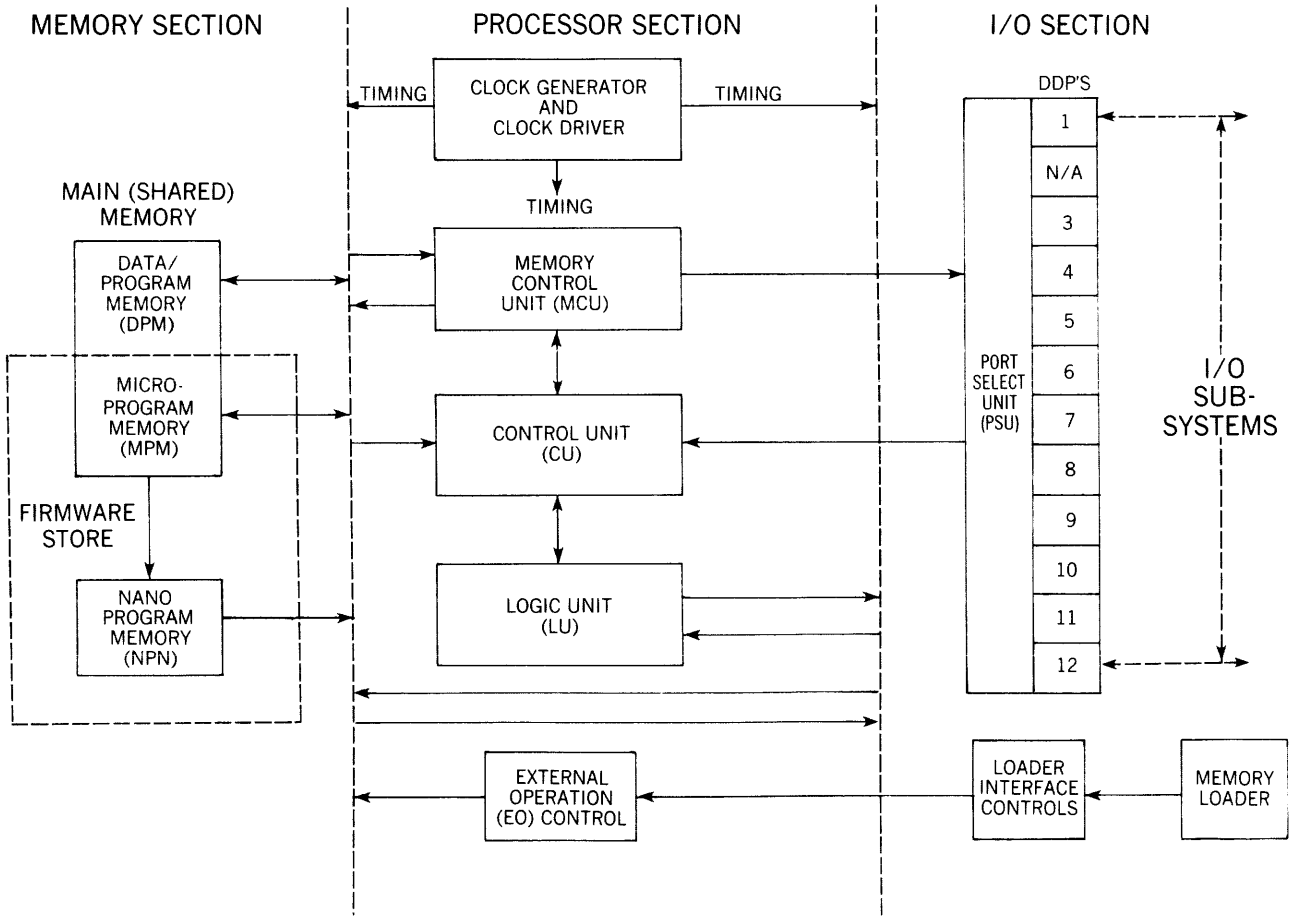



**Figure 3-2. CPU General Functional Block Diagram**

integrated-circuit memory for nanoprogram storage, contains the systematic controls required by the processor section. Because the main RAM memory is also used for data/program storage, addressing and data control is accomplished by shared memory controls. The microprogram memory outputs are called microprogram codes and are actual 16-bit instructions. These instructions are classified as either type I for nanomemory addressing, or type II for loading specific registers within the processor section.

The nanoprogram memory outputs are called nanocodes and are actual 55-bit instructions. Figure 3-3 shows the configuration of microprogram and nanoprogram codes. Descriptions of the functionals registers and elements are provided in the processor section description.

Nanoinstructions have three major classifications:
a. No conditional logic.
b. Conditional logic met.
c. Conditional logic not met.

In MPM storage, correct parity is generated (bit 17) when writing, and checked when reading. Bit 56 in NPM storage is used to check parity when reading nanocodes. Odd parity is used for MPM and NPM validity.

## DATA/PROGRAM (USER) MEMORY SECTION

The Data/Program Memory (DPM) section is that portion of main memory which contains user type data and S-level programs and is controlled by shared memory logic in conjunction with external operations.

*SHARED MEMORY CONTROLS*

Microprogram memory and S-level memory share the same physical memory in the CPU. The shared memory (SM) controls are used for addressing both segments of memory and for transferring data to and from memory.

The size of shared memory ranges from a minimum of 8,192 words (16,384 bytes) to a maximum of 49,152 words (98,304 bytes) when all memory options are used. Extensions to memory are in physical increments of 4,096 words (8,192 bytes). The specification of a memory address in excess of the address limit results in an error condition.

Because the memory is shared by S-level and MPM, provisions are made for addressing the two segments separately. The memory controls contain selection gating for accepting the memory address from either the base register (BR), the MAR (for S-level memory), or from the incrementer (for MPM memory), as determined by the appropriate memory controls. Up to a maximum of 16,384 words (32,768 bytes) of the shared memory may be reserved for MPM, depending on the requirements of the microprogram. The division between S-level and MPM is soft, allowing the upper limit of MPM to be varied at the discretion of the system software.

An external (EXT) timer, which is a binary counter, sets the EXT flip-flop every 125 milliseconds. EXT is reset by testing and can be used for various applications by the programmer.

*S-LEVEL MEMORY OPERATION*

Addressing of S-level memory is accomplished by use of an external operation command which specifies a memory read or write operation. In this case the external operation controls enable the appropriate control signals, causing the next memory address to be taken from output select lines 1 through 16. Data being written to S-level memory is transferred through the S-memory (SM) controls from the MIR bus of the processor. Data read from memory goes only to the B register of the processor through the SM controls and the external bus. All data transfers to and from S-level memory are in 16-bit parallel format. Parity is generated for all words written into memory and is checked on all words read from memory.

*MEMORY STANDBY MODE*

A standby mode is provided so that, during equipment maintenance operations, power can be maintained to the first 8 K-bytes of memory to facilitate servicing.

## PROCESSOR SECTION

The processor section is the major section of the CPU and performs operations defined by the program stored in the firmware store section, under the control of microcodes and nanocodes. The processor section is subdivided into five functional circuit areas:
a. Logic unit (LU).
b. Memory control unit (MCU).
c. Control unit (CU).
d. Clock generator (CG).
e. External operation controls (EO).

## LOGIC UNIT

The Logic Unit (LU) performs the shifting, arithmetic, and logic functions and provides a set of scratch-pad registers for temporary storage. The LU also provides the registers for the data interfaces to and from the I/O controls and S-level memory (DPM). The major registers and elements comprising the Logic Unit are the A registers, B register, memory information register, adder, and barrel switch.

*A REGISTERS (A1, A2 AND A3)*

The A registers are 16-bit registers used for the temporary storage of data being transferred from the Barrel Switch (BSW) to the adder. Any or all A registers may be selected for input from the BSW. The registers serve as a primary input to the adder and are individually selectable.

## MICROPROGRAM CODES

MPM BITS

| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | 17 |
|---|---|
| 0 0 — AMPCR — | PARITY BIT |
| 0 1 — MPCR & INCR — | |
| 1 0 — MPCR & INCR — | |
| 1 1 0 0 — SAR — — LIT — | |
| 1 1 0 1 — SAR — 0 0 0 0 0 0 0 0 | |
| 1 1 1 0 0 0 0 0 — LIT — | |
| 1 1 1 1 — NANOADDRESS — | |

0 = UNUSED

## NANOPROGRAM CODES

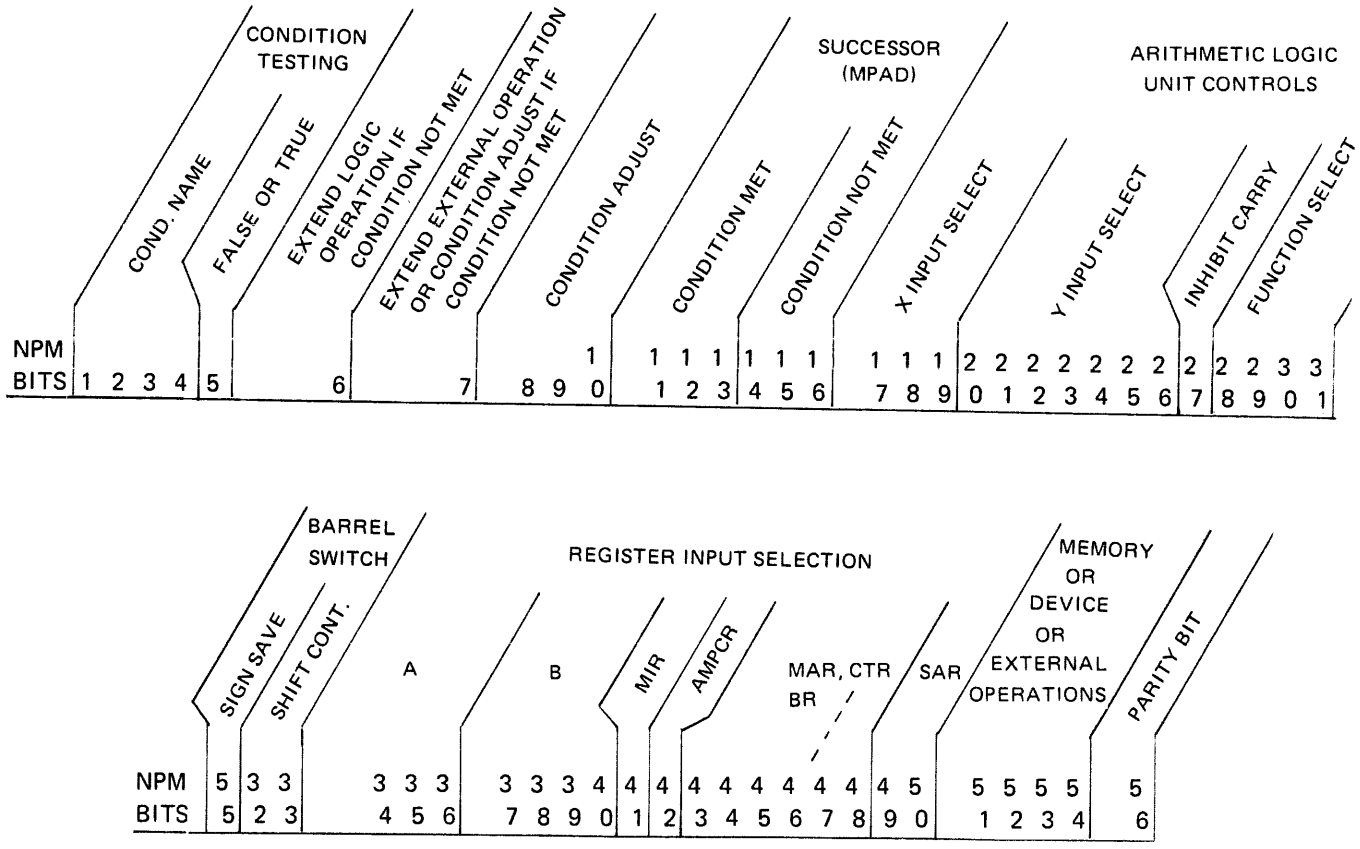



**Figure 3-3. Microprogram and Nanoprogram Codes**

*B REGISTER*

The B register is a 16-bit register which provides the primary interface from both S-level (data/program) memory and the I/O controls. It serves as the secondary input to the adder, and can be used for temporary storage of certain information resulting from arithmetic operations. The only destination for this data is the adder. The nanocode allows the manipulation of the least significant bit, most significant, and/or the 14 central bits of the B-register contents upon transfer to the adder.

*MEMORY INFORMATION REGISTER (MIR)*

The MIR is a 16-bit register used primarily to buffer information being written in memory or sent to a device. MIR is to main memory, an I/O control or to the B register.

*ADDER*

The adder performs the arithmetic and Boolean operations on data from the B-register, literal/counter register(s), alternate microprogram count register, or base register/memory address register. Output from the adder goes unconditionally to the barrel switch but may also be sent to the B-register if so specified.

*BARREL SWITCH (BSW)*

The Barrel Switch (BSW) is a matrix of gates used to shift a parallel input data word a number of places left or right, end-off, or end-around. The shift amount is specified by the contents of the Shift Amount Register (SAR). Data input to the barrel switch is from the adder. Destinations are the A registers, B register, memory information register, alternate microprogram count register, memory address register, either base register, or the shift amount register.

## MEMORY CONTROL UNIT

The Memory Control Unit (MCU) is used primarily for memory and I/O device addressing. The major registers and elements comprising the MCU are described below.

*MICROPROGRAM COUNT REGISTER (MPCR)*

The MPCR is a 14-bit register which contains the instruction address for the microprogram. MPCR contains the current instruction address except when an "EXECUTE" instruction is performed. MPCR can be loaded by a type II instruction or with the output of the incrementer. MPCR output goes to both the incrementer and the AMPCR.

*ALTERNATE MICROPROGRAM COUNT REGISTER (AMPCR)*

The AMPCR is a 14-bit register which contains the jump or return address for program jumps and subroutine returns within microprograms. The address is one less than the position to be jumped to and two less than the position to be returned to. AMPCR can be loaded from MPCR, from the 14 LSB of the barrel switch, or by a type II microinstruction fetch from microprogram memory. AMPCR output goes to the incrementer.

*INCREMENTER*

The incrementer adds 0, 1, or 2 to the selected input from either MPCR or AMPCR. The output of the incrementer is the input to the MPCR and also provides an address to the microprogram memory. The MPAD address controls select both the input source and the amount to be incremented.

*MICROPROGRAM ADDRESS CONTROLS (MPAD)*

The MPAD controls are used to control the loading of MPCR and AMPCR, the selection of MPCR and AMPCR input to the incrementer, and the selection of the value (0, 1, or 2) to be used in incrementing. The selection of controls and a true or false successor (testing for alternative actions) is done during phase 1 of the microinstruction and involves the results of condition testing and successor determination associated with bits 1 through 16 of the nanoinstruction. If a type II microinstruction is executed, the controls for a step successor are forced.

*MEMORY ADDRESS REGISTER (MAR)*

The MAR is an eight-bit register which holds the eight LSB of a memory address. MAR is concatenated to either base register 1 or base register 2 to form the absolute address. MAR may be loaded from either the least-significant byte of the barrel switch or from the literal register. The output is to S-level memory. The MAR along with the currently selected concatenated register may also be sent to the adder.

*BASE REGISTER 1 (BR1)*

Base register 1 is an eight-bit register which holds a device address or the base address of a 256-word block of memory data. When used for a memory address, BR1 is concatenated with MAR to form an absolute memory address that is transferred by the output select gates to S-level memory. The concatenated registers may also be sent to the adder.

When used to hold a device address, (BR1 is interfaced by the output select gates to the port select unit, where it is used to address the IOC to be used. The input to BR1 is from the most significant byte of BSW. Once placed in the selected mode by a memory/device command, BR1 remains selected until a command is issued selecting BR2.

*BASE REGISTER 2 (BR2)*

Base register 2 functions exactly like BR1, thus providing a second register for holding memory/device addresses. Once placed in the selected mode by a mamory/device command, BR2 remains selected until a command is issued to select BR1.

*OUTPUT SELECTION GATES (OS)*

The output selection gates select the specific source (BR1/MAR or BR2/MAR) of S-level memory or device addresses. A Read/Write "1" command selects BR1/MAR; whereas, Read/Write "2" selects BR2/MAR. If no source is specified, the register selection remains unchanged. OS output is to the port select unit and/or to the adder.

*COUNTER (CTR)*

The CTR is an eight-bit counter used for loop control and other counting functions. CTR is loaded through the MAR/CTR selection gates from either the literal register or the least-significant byte of the barrel switch. CTR can be used as an input to the most-significant byte of the adder. A counter overflow results in setting the Counter Overflow flag (COV) in the control unit condition register; COV is reset either by testing the bit or by loading the counter with a new value.

*LITERAL REGISTER (LIT)*

The LIT is an eight-bit register used as temporary storage for literals in the microprogram. LIT is loaded from microprogram memory using a type II microinstruction. LIT may be used as an input to the least-significant byte of the adder, and/or through the MAR/CTR input selection gates to either the counter register or the MAR.

*MAR/CTR INPUT SELECTION GATES*

The selection gating consists of eight bits and is used to select an input to the MAR or CTR from either the LIT register or the least-significant eight bits of BSW. These functions are mutually exclusive.

*CONTROL REGISTER*

The control register is a 40-bit register used to store all control signals from the nanomemory that are not used in phase 1. Certain control signals are decoded before being strobed into the control register while others are decoded on output. Gate delays introduced into the control signal sequence determine whether the coding is done before or after the register. The register includes both MPAD controls and phase 3 controls and thus contains their respective clock gating networks. The control register is physically contained in the MCU, CU, and EO registers and provides nanocontrols to all sections of the processor.

## CONTROL UNIT

The Control Unit (CU) performs two main control functions: shift amount control and storage of condition indicators, data and status interrupts, and status indicators. The control register, defined within the MCU, distributes nanocontrols throughout the processor. The elements comprising the control unit are described below.

*SHIFT AMOUNT REGISTER*

The shift amount register and its associated logic is used to control the loading of shift amounts and the sequencing of shift operations. (Refer to barrel switch description.)

*CONDITION REGISTER AND SELECT*

The condition register has six of the 16 selectable condition bits, only one of which may be selected and tested by a nanoinstruction. If an attempt is made to reset and set a condition bit at the same time, the set condition is dominant except on counter overflow (COV), in which case the reset condition is dominant.

*SIGN-SAVE FLIP-FLOP*

The sign-save flip-flop is used to store either adder overflow or barrel switch end-off bits.

*CLOCK GENERATOR AND CLOCK DRIVER*

The Clock Generator (CG) contains a 10-MHz oscillator and a divider which produce the 1-MHz system timing clock (S-clock); that is, a 50-nanosecond pulse every microsecond. The S-clock is distributed throughout the processor, except to the logic unit. The logic unit is clocked by the phase-3 clock (3-clock), derived from the S-clock. The main function of the phase-3 clock is to inhibit destination register selection during an extended phase-3 condition; these registers are clocked upon phase-3 completion. A 10-MHz clock pulse is also distributed to peripheral device subsystems that use a miniprocessor. A third set of controls on the CG board is used for generating the clear signals to the processor elements. The Clock Driver (CD) contains drivers for clock pulse buffering and distribution.

*EXTERNAL OPERATION CONTROLS (EO).*

Memory and I/O operations are mutually exclusive due to the fact that both share common input busses. The selection of either memory or a device is controlled through the decoding of nanobits or from an I/O subsystem that has a direct memory access channel (DMAC). The nanobits, which are sent to a control register in the EO controls, specify whether a memory operation or an I/O operation is to take place. The nanobits also define the function that is to be performed, and select the base register to be used in the addressing operation. If a memory read or write operation is indicated, the command is fully decoded, and the appropriate interface signals are generated to the memory. An I/O subsystem with a direct-memory access channel can "steal" a processor main-memory cycle by inhibiting the MIR register from the MIR bus and using the MIR bus for memory access. (Refer to description of DMAC under I/O control section heading.)

Device read and write operations are decoded before being sent to the port selector, where they are buffered and gated so as to cause the device

operations to be terminated at the proper times. However, the Address/Status Request (ASR) signal is decoded directly from the nanobits and sent to the port selector as a separate signal without gating. This is to allow immediate enabling of the status word lines to the external bus since the status word is returned to the processor in the same clock time.

*ERROR DETECTION*

The processor automatically detects error conditions and initiates the following:

a. The system clock is inhibited and the HALT indicator (on EO card edge) is turned on.

b. The microprogram address (to be accessed) is forced to 0. (INCR and MPCR are *not* cleared.) Location 0 must contain a CPCR call instruction to direct the firmware to the appropriate error-processing routine and to store the contents of MPCR in AMPCR for reporting to the operator.

c. The four-bit error descriptor code is displayed by the EO2 card error indicators, which are card-edge indicators visible through the front panel of the processor. The error descriptor code is also placed on EXT bus bits 13 through 16, and the exception bit is placed on EXT bus bit 1. These EXT bits are read by the error-processing routine.

NOTE

When the EXT1 (HALT) bit is turned on, the console ERROR indicator is also lit on the system console.

d. The HALT indicator is turned off.

e. When the error processing routine is executed, the error indicators and EXT bits are cleared.

f. When the error processing routine is completed, the initial error condition is reset to permit processing any subsequent error condition.

If a second error condition occurs prior to the resetting of the first condition, the system locks in the halt condition (all system clocks are inhibited) and the descriptor code of the second error condition is displayed on the EO2 indicators.

If the error stop switch is set, or the load mode is set, the system locks in the halt condition on the first error.

Table 3-1 shows the configuration of error codes presented on the EXT bus and displayed by the EO2 card-edge indicators.

Note: EXT bits 13 thru 16 are indicated by EO bits 8, 4, 2, 1, from top to bottom on EO card.

In general, the error indications have the following meanings:

a. Memory loader parity read error detected in the media data during read-in of data from memory loader to microprogram memory.

b. MPM parity error detected in the MPM portion of shared memory.

c. Read-after write MPM parity error detected during load on microprogram memory read-after-write check during loading from console paper tape reader.

d. DPM parity error detected in the DPM portion of shared memory.

e. NPM parity error detected in nanoprogram memory word.

f. Memory address limit error exceed memory limit register setting.

**Table 3-1. Error Descriptor Codes**

| | EO Indicators (EXT Bits) | | | | |
|---|---|---|---|---|---|
| Halt (1) | 8 (13) | 4 (14) | 2 (15) | 1 (16) | Error |
| 0 | 0 | 0 | 0 | 0 | None |
| 1 | 0 | 0 | 0 | 0 | Transient |
| 1 | 0 | 0 | 0 | 1 | Loader Error |
| 1 | 0 | 0 | 1 | 0 | Load MPM Parity |
| 1 | 0 | 0 | 1 | 1 | Load Address |
| 1 | 0 | 1 | 0 | 0 | Nano Parity |
| 1 | 0 | 1 | 0 | 1 | Nano Address |
| 1 | 0 | 1 | 1 | 0 | MPM Parity |
| 1 | 0 | 1 | 1 | 1 | MPM Address |
| 1 | 1 | 0 | 0 | 0 | Not Used |
| 1 | 1 | 0 | 0 | 1 | DPM Address, Write |
| 1 | 1 | 0 | 1 | 0 | DPM Parity |
| 1 | 1 | 0 | 1 | 1 | DPM Address, Read |
| 1 | 1 | 1 | 0 | 0 | Not Used |
| 1 | 1 | 1 | 0 | 1 | Steal Address, Write |
| 1 | 1 | 1 | 1 | 0 | Steal Parity |
| 1 | 1 | 1 | 1 | 1 | Steal Address, Read |

## I/O CONTROL SECTION

The I/O control section of the B 720 CPU is the interface for control communications and data transfer between the processor section and the various I/O and data communications subsystems. The I/O section has four major functional units or areas: the port select unit, the device-dependent ports, the I/O controls, and the loader interface area. Direct-memory access circuitry is also implemented for direct accessing between certain I/O subsystems and the memory section.

The Port Select Unit (PSU) interfaces all peripheral device I/O subsystems and data communications subsystems and the memory load subsystem. The PSU maintains input and output control for the various interfaces and establishes service and interrupt priorities.

The Device-Dependent Ports (DDP's) are the I/O backplane or module areas that accommodate the various I/O controls (IDC's) used in the B 720. These ports are physically and functionally categorized into two groups: special (or dedicated) DDP's, and interchangeable (standard) DDP's. A total of 11 DDP's are available if the I/O expansion module and communications subsystem are installed. An IOC is designed

to interface a particular device or interface by converting levels, interpreting signals, converting formats, synchronizing or timing operations, and buffering data. (Refer to Section I for a complete description of the DDP and IOC configuration.)

The allocation of DDP's to IOC's is predicated on a priority service basis, established during system installation for the particular I/O configuration used.

## DEVICE ADDRESSING

The PSU receives device addresses from the base registers (BR1 and BR2) in the processor. The firmware selects a device for access by sending to the PSU the device address contained in the base register and pointed to by the read/write command. This binary address, sent through the output select (OS) gates, defines the device IOC to the port selectors, as shown in figure 3-4.

The most-significant bit of the address is used by each IOC to specify control or data word format.

NOTE

Control format on a device read operation is status information.

Input information to the addressed IOC comes from the processor section MIR. Output information from the addressed IOC (EXT lines 1 through 16) is made available only to the B register in the processor section.

## INTERRUPT HANDLING

Interrupts are always sent from the IOC's to the port selector, even though the control may not be addressed. Interrupts from unaddressed IOC's are an Input Request (IRQ) to the processor-section control unit from the highest priority device and are generated by either status interrupt or data interrupt signals. Handling of the interrupt condition is the responsibility of the firmware store program. The program must issue an address status request command, placing the reason for interrupt from the IOC on the EXT lines, along with the address of the IOC position from the port selector. Interrupts from addressed IOC's to the processor-section control unit are an unsolicited request (URQ) for a status interrupt and a solicited request (SRQ) for a data interrupt. Handling of these interrupts is also the responsibility of the firmware store program.

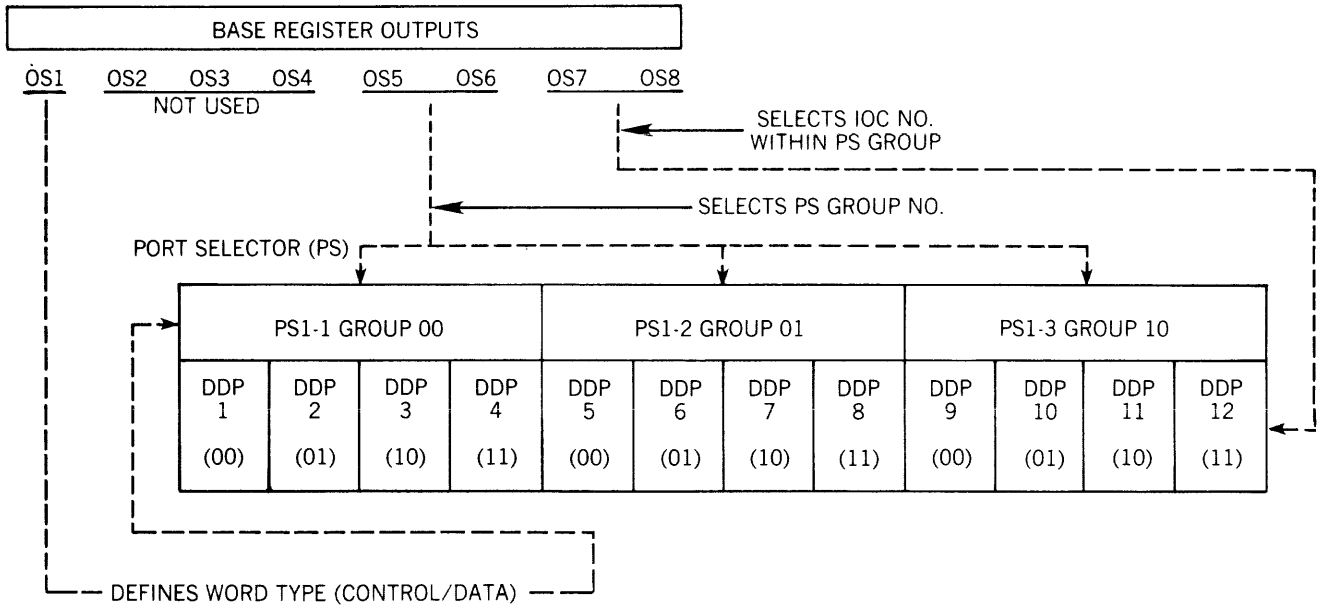



**Figure 3-4. I/O Device Addressing**

## MEMORY LOADING

Direct memory loading is facilitated by an integral photoelectric paper-tape reader mounted in the left-front section of the CPU cabinet (figure 3-1). Loader interface (LI) controls permit data to be transferred directly from the paper tape coder to microprogram memory. Data is written into the MPM one word at a time, including a generated parity bit. These words are written sequentially up through main memory when the processor LOAD push button is enabled.

## DIRECT MEMORY ACCESS

Direct memory access (DMAC) channels are provided for the disk and programmable data communication (PMLC) subsystems to enable transfer of data and control words between main memory and the subsystems. Direct memory access is performed without any processor intervention required and thus frees the processor to perform other tasks.

A process referred to as "stealing" is used in direct memory transfers; that is, the disk IOC or PMLC "steals" one cycle (clock period) of processor time for each data/control word transfer. This process is controlled for both DMAC channels by circuits in the disk IOC. The steal control circuits receive steal request signals from the disk IOC and the PMLC and grant memory access on a service priority basis. The disk I/O subsystem has top priority. The steal control circuits also receive MPM data outputs directly from the shared memory control circuits of the processor and apply the data to the common IOC data bus. This bus handles both memory data retrieved through the direct-memory access and control words and/or data sent to an IOC under processor logic control.

# SECTION 4

# DATA COMMUNICATIONS SUBSYSTEMS

## GENERAL

Two types of data communication configurations are available for use with the B 720 System: programmable communications processing and single-line (file inquiry) data communications. This section provides a general description of these communications systems and their operation in the B 720.

## PROGRAMMABLE COMMUNICATIONS PROCESSING

Data communications processing is implemented in the B 720 by use of the B 352 Communications Processor and required and optional associated items which comprise the communications module. Section I provides a general description of this configuration which operates as an independent subsystem.

The B 352 executes program instructions to control the flow of data between the system processor and up to four data communications channels. Up to four half-duplex or four full-duplex channels may be serviced simultaneously. These four channels operate independently of each other at all line rates except in some instances at the highest operating rates.

## SUBSYSTEM INTERFACE

Figure 4-1 shows the interfacing of the subsystem elements in a configuration for operation on four synchronous and/or asynchronous channels. Asynchronous interface connections are also possible for two-wire direct station connections. Either direct-connect method is permissible for any of the four channels, provided that the interconnecting cable is less than 1,000 feet.

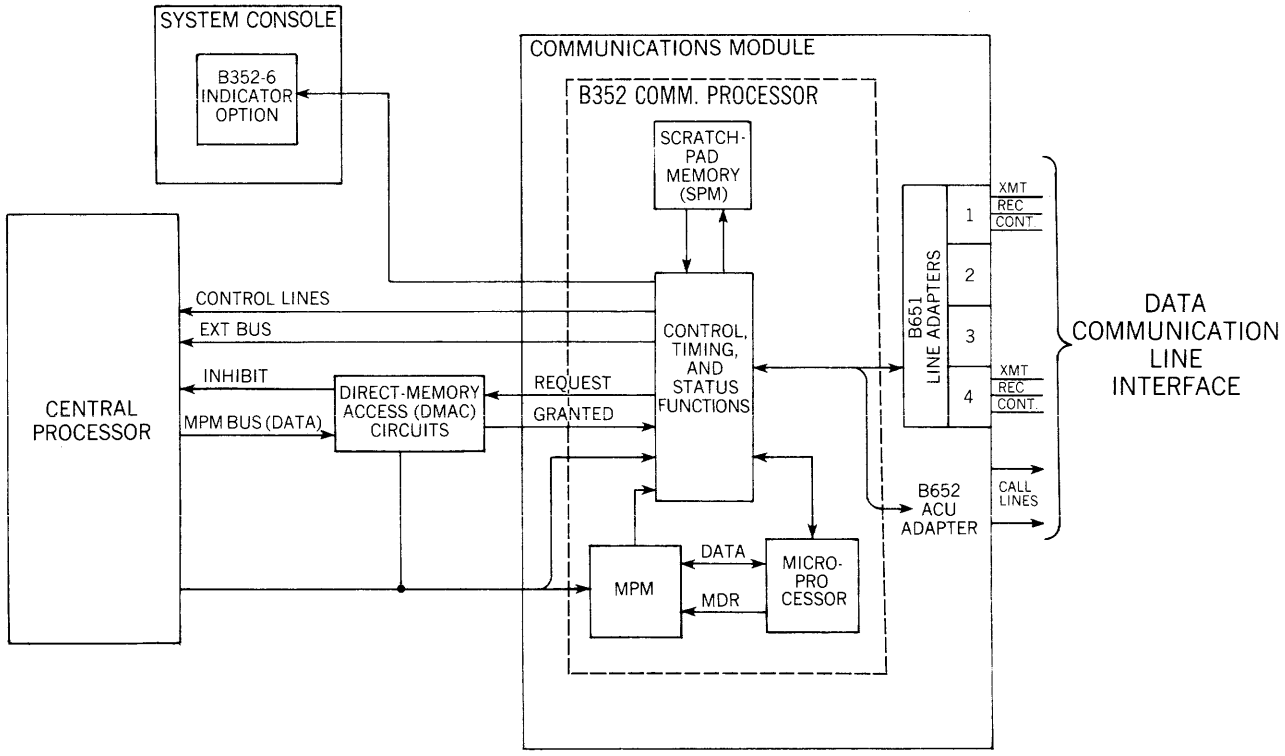



**Figure 4-1. Multiline Data Communications Subsystem Interface**

All message data that flows between the B 352 on any channel and a station is bit-serial in strings that form characters (bytes). Thus, the transmission of a character to a station requires that it be shifted serially onto the channel data line. For receive operations, the serial bits must be assembled by the B 352 into eight-bit characters or 16-bit words, which in turn can be stored by the B 352 directly into a buffer area or returned to the processor B register for subsequent manipulation. The capability for directly accessing the MPM for reading or writing data is often referred to as "DMAC" (direct memory access). Thus the B 352 serves to functionally interface serial data lines with the 16-bit parallel data requirements of the system.

Figure 4-2 illustrates the types of terminal device interfaces. Each configuration is tailored to the terminal station by the firmware/software that drives the B 352. Except when synchronous data sets are used, bit rates for transfers are determined by the NDL program written for the specific installed equipment. When synchronous interfaces are used, the transmit/receive clocks are provided by the data set and no data transfer timing signals are supplied other than a "replay" of the transmit clock received from the data set. Because the four-line data set capability is almost entirely under firmware/software control, nearly any line procedure requirement can be met.

In summary, the communications module can provide the following standard or optional capabilities:

a. Synchronous interface in standard configuration at 75 to 9600 baud rates.

b. Asynchronous interface at 75 to 9600 baud rates.

c. Direct-connect two-wire interfaces.

d. Data Set/Modem interface in accordance with RS-232-C standards, synchronous or asynchronous.

e. Data Set/Modem interface in accordance with CCITT standard V.24.

f. Half-duplex or full-duplex modes.

g. Data Set/Modem connections over switched or leased lines.

h. Automatic dial/answer operation on switched lines.

i. Variable code handling capabilities.

## FUNCTIONAL OPERATION

Figure 4-3 is a general functional block diagram of The B 352, which can be controlled to operate instructions that are fetched from main memory of the system processor. This fetch process can be allowed to occur as an automatic function or can be controlled by the processor as a single-instruction function. As a result of executing these instructions, the B 352 is able to scan each of its four universal line adapters (line interfaces) and maintain control over their activities. Data can be either received or transmitted

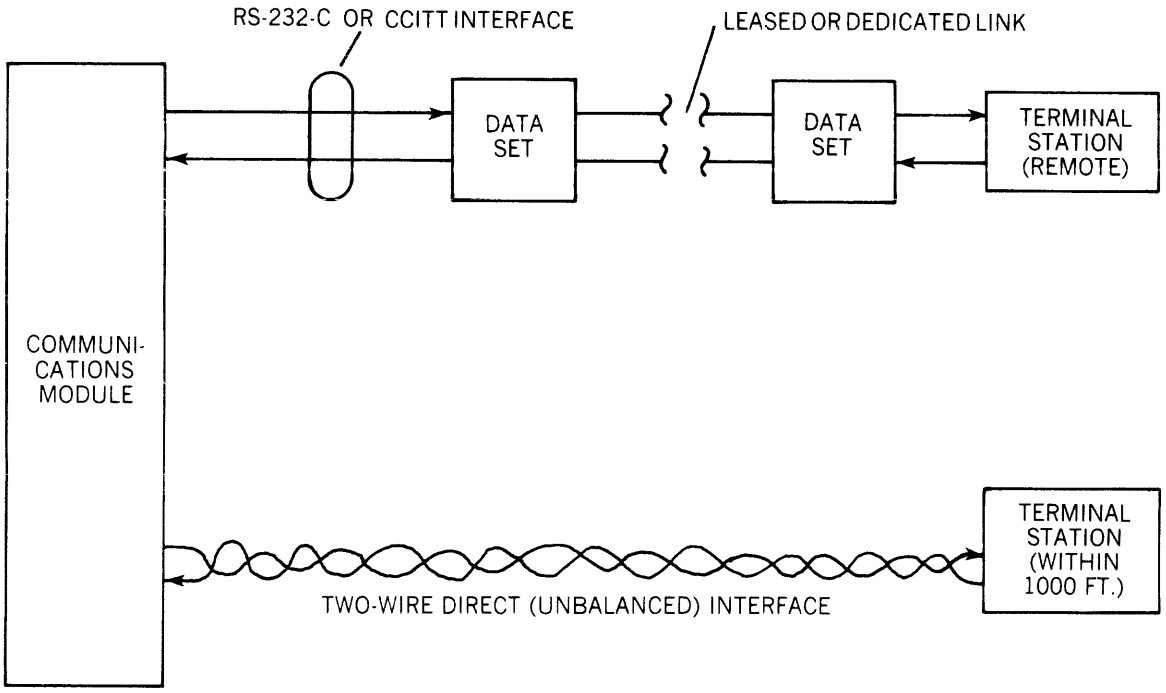



**Figure 4-2. Terminal Station Interface Configurations**

through the adapters and stored or fetched from the processor main memory without requiring processor intervention.

Because the B 352 is a firmware/software driven control, its system functions is firmware/software dependent and variable. This is in contrast with other I/O controls that perform a given system function as directed by a control word. The specific logic functions, capabilities, or actions that the B 352 accomplishes can be understood without a knowledge of the related firmware/software system; however, an in-depth understanding cannot be attained without some software knowledge.

## FIRMWARE/SOFTWARE CONSIDERATIONS

The relationship between the firmware/software elements may be readily understood by viewing two imaginary systems: each capable of independent operation by use of independent interpreter firmware and user software contained within them. One system (processor A) is assumed to have a peripheral complement that suits a general processing operating environment. The other system (processor B) is assumed to be structured with I/O controls suited for a multiple data set I/O line servicing environment.

The user software in both processors is an S-level language requiring interpretation. However, due to the unique requirements imposed by data set control variables, the S-level program in processor B is

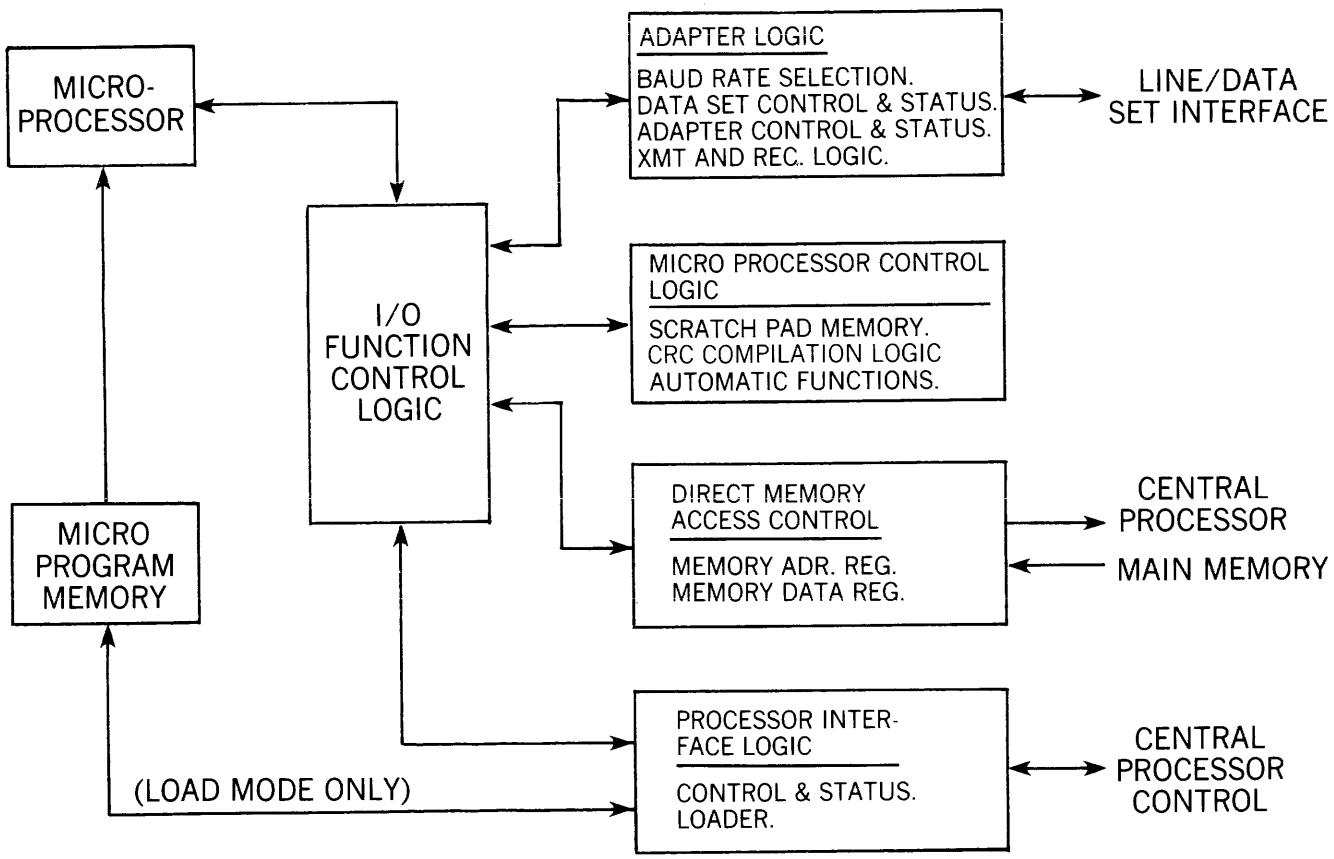



**Figure 4-3. Communications Processor Functional Block Diagram**

modified to handle data network peculiarities in a convenient manner. The resulting S-level language is named the Network Definition Language (NDL). While still an S-level language, NDL is distinguished from the normal S-level language by the special, enhanced S-level NDL language. A follow-on result of an essentially new language is that a new and/or modified interpreter is required in order for the new NDL program constructs to have meaning. Therefore, the original imaginary systems are arranged as follows:

| Processor A | Processor B |
|---|---|
| S-Level Interpreter | NDL Interpreter. |
| S-Level User Program. | NDL User Program. |

The function of the processor B system is of little value unless it can (1) receive data to be transmitted from a user and (2) allow a user to make use of the text of data set messages. Therefore, a marriage between the two processor systems is necessary to permit a dynamic environment to exist for enhancement of user software capabilities. This can be accomplished by identifying and allowing the memory storage portions of both systems to be common rather than independent. By this means, the text of messages can be stored in a buffer area which is identified to both processors as the same area. Data entering through processor B and associated terminals can now be accessed by processor A for subsequent processing. The buffer area used in this instance is usually referred to as the RECEIVE buffer. Conversely, another buffer area can be used to assemble the text of output messages for transmission thru a data set to a remote station. This buffer is normally referred to as the TRANSMIT buffer.

The commonality of memory is only part of the answer for providing the enhanced software capability previously mentioned. The two user programs operating in processors A and B must be intimately related and additionally, one must have controlling power over the other. Processor A is the processor that is given controlling power because its firmware contains special control routines. Processor A, in the course of executing an S-level user program, will probably assemble a data file that is to be sent to a remote station for additional processing, printout, or storage.

The user program can thus contain an instruction such as "WRITE file-X TO terminal station destination" or "READ file-X FROM terminal station." Because these are not normal S-level type instructions, a decoder routine in processor A is called upon to implement the instruction or identify an illegal operation code. This decoder routine has an assigned area in memory into which key data for the "WRITE or READ" function is stored. This assigned memory area is usually referred to as the Job Status Table.

The NDL program in processor B can scan the Job Status Table to locate jobs to perform and record the status of such jobs as not started, in process, or completed in addition to other pertinent information. The NDL program in processor B will perform all functions required to accomplish the jobs assigned for it to do without requiring the aid of processor A. Therefore, processor A can be performing other computational rather than communications functions.

The NDL user program is written specifically to handle a particular configured system and network of terminal stations. Changes to the terminal station network usually requires program modifications. These changes and corresponding NDL program modification are completely under control of the user. The types of control information that is written into the NDL program determines such parameters as: (1) validity of terminal requests, (2) station busy status, (3) rate of terminal service required, (4) line frequency selection, and (5) line selection. Thus, the user program of processor A only has to execute a single instruction for sending a file or message to terminal XX; the NDL program and processor B does all the necessary execution and control generation to implement the required transmission. In application, processor A is the central processor. The memory of the processor contains the S-level interpreter, S-level user program, and the NDL S-level program. The memory integral to the B 352 contains the NDL interpreter. These firmware/software relationships are summarized as follows:

a. The S-level interpreter is a microinstruction-level program resident in system processor main memory and used by the processor to decode S-level program instructions for implementation.

b. The S-level user program is resident in system processor memory and is interpreted by the S-level interpreter.

c. The I/O buffer area is used to store received data or output data to be transmitted over data lines.

d. The NDL tables are resident in system processor memory and generated when the NDL program is compiled. These tables are used as a driver cross-reference by the NDL object program.

e. The NDL interpreter is resident in the memory of the B 352 and is similar in nature to the S-level interpreter, except for NDL object program execution.

## INTERFACE COMMUNICATIONS

The initial communication by the processor to the B 352 is the transmission of a control word. The processor informs the B 352 that a control word is being sent to it by also sending control signals. These control signals are standard processor control signals for I/O controls. The presence at the B 352 of signals INST (instruction) and WRITE signify that a control word is present and should be accepted, decoded, and executed. The control word is sent to the B 352 over

the MIR bus and is used to place or remove the B 352 from its load mode of operation. The load mode is used to load the memory of the B 352 with the NDL interpreter necessary for NDL program execution. Once loaded, execution of an NDL program is accomplished by the running of this NDL interpreter within the B 352. However, master control of the running of the NDL interpreter is retained by the firmware of the central processor. This firmware exerts its control by sending data words to the B 352. These data words are actually instruction words which are decoded by the B 352.

The presence of processor control signal WRITE and the absence of the instruction signal INST identifies to the B 352 that a data word is present and should be accepted. Whereas this data word would be retransmitted to a terminal device/channel in other I/O controls, the B 352 decodes the content of the data and acts upon them as instructions. Data words received by the B 352 in this manner are actually another type of control word. Thus the B 352 can be viewed as receiving two types of control words: one being called a control word and having limited functional usage and one being called a data word but containing control instructions.

The central processor can read status words from the B 352 in the same manner that status words are read from other I/O controls. Certain control signals cause the B 352 to return a status word. The status word is a coded 16-bit word which informs the processor and its firmware of the current status of the B 352. This transfer takes place via the EXT bus and the data is placed in the system processor B register.

Another means of communication between the central processor and the B 352 is initiated by use of the special direct memory access circuitry. This means of communication permits the B 352 to write data into or read data from the MPM of the system processor. This capability is shared with the disk I/O control, which has a higher priority in obtaining direct memory access. When memory access is required, the B 352 produces a request signal which, if no other control is requesting, is returned as a request granted signal. A granted request inhibits the system processor for one clock interval and thus allows the requesting control to "steal" a memory cycle for either writing data to or reading data from memory. When the B 352 accesses system processor main memory to read data, the supplied address may be of the buffer area, the NDL table area, or the NDL object program area. When the NDL object program area is addressed, the usual reason for access is to fetch the next instruction to be executed by the B 352. Access to memory for writing will usually be to the buffer area for storing of input data received by the B 352 from its data line interface.

The B 352 can interface with up to four line adapters that may be terminated by data sets or be directly connected with some digital device having a compatible direct connect interface circuit. Certain interface circuit lines are monitored for line status changes. In this manner the B 352 can record the occurrence of key events in a status character formed in the line adapter portion associated with the interface. Periodically, firmware of the B 352 will cause this status character to be read, decoded, and appropriate response actions result. When more than one line adapter is used, each adapter is sampled (scanned) in the same manner. If, for example, an adapter is scanned and a status character bit is set specifying input data ready, the presence of a request to input data for the central processor is signified. The response to this request is the generation of an appropriate sequence of commands for implementing the line procedure requirements of the interface and device used. Once the input data character is assembled, another scan of the line adapter returns a status word with a different bit set to signify input data character ready.

Upon assembling two such characters, the B 352 stores the two-character word directly in the system processor main memory. Thus the B 352 is constantly busy scanning line adapters and processor requests for transmitting or receiving data to service any requirements. While actively scanning, any conflicts of data flow are resolved with regard to simultaneous occurrences, direction, error flags, and line procedures as may be required. The B 352 thus is a traffic manager or data handler for the system processor. Because of its semi-antonomous operating environment, the B 352 frees the system processor to execute the calculative/manipulative portions of user programs.

## OPERATIONAL INDICATORS

The B 352 can be optionally equipped with circuitry for providing signals to light-emitting diode (LED) indicators. These indicators, normally installed on the system console panel, are for monitoring receive and transmit data activity. The receive indicator for either channel is lit when the B 352 is receiving data (SPACE condition) over the line. The transmit indicator is lit when a SPACE condition is being transmitted and is unlit for a MARK condition.

## FILE INQUIRY SUBSYSTEM (SINGLE-LINE DATA COMMUNICATIONS)

File inquiry operation is a special and restricted application of data communications provided by use of the optional single-line data communications capability of the system.

In this application, the term "inquiry" refers to the technique by which the contents of a computer's storage is interrogated from a remote terminal device. Inquiry programs cannot, therefore, cause data to be stored by a computing system. Hence, "inquiry" can be thought of as a special and restricted application of data communications.

The Burroughs B 720 Inquiry System permits an operator at a remote inquiry station (TD 700 or TD 800 Terminal Display) attached to the central system to make disk data file inquiries under user programmatic control. The inquiry capability is based on the use of COBOL programs written by the user according to his specific needs. An inquiry program is run from a remote terminal and may temporarily interrupt a normal system job. The COBOL compiler provides the protection for existing data files and for files on other peripheral devices, thus maintaining the integrity of normal system jobs.

In inquiry programs, the only files that may be selected other than the inquiry file are disk files. These disk files must be opened INPUT or TAG-FILE and closed WITH LOCK. The console may also be accessed in inquiry programs.

## SYSTEM EQUIPMENT AND SOFTWARE REQUIREMENTS

The following system equipment and software elements are required to implement B 700 File Inquiry operation in a B 720 System:

a. A Central Processing Unit (CPU) with a minimum main-memory capacity of 32K bytes.

b. A B 9343 Console and B 346/B 346-1 console I/O control.

c. An A 9480-12 or A 9481-12 Disk Cartridge Drive with a 4.6M-byte capacity (minimum) and a B 489-2 Disk I/O Control.

d. A B 351-1 Single-Line Control (SLC).

e. One or more TD 700, TD 800, and/or TD 801 Terminal Display Units (maximum of nine individual types or combination of types).

f. A medium systems COBOL compiler.

g. B 700 S-level Interpreter.

h. A user-supplied Inquiry Program.

Note that only one SLC may be installed in a CPU, and it requires two ports (DDP's) in the B 312 I/O expansion module. A *maximum* of nine terminals may be connected to the SLC in any combination of types (TD 700, TD 700 with extended memory, TD 800, or TD 801). The terminals communicate with the B 720 over a private (direct-connect), asynchronous, 9600-Baud line interfaced by the SLC.

## FIRMWARE REQUIREMENTS

The system firmware required for inquiry operation has a number of additions and enhancements to implement the inquiry capability. The inquiry firmware consists of two distinct segments: (1) the Inquiry Interrupt Controller and (2) the Inquiry Controller/Handler. Only one segment can reside in memory at any given time because of memory size limitations. The Inquiry Interrupt Controller segment is resident initially when the Inquiry firmware has been warmstarted and any later time that the Inquiry system is idling. The Inquiry Controller/Handler segment is present only after an Inquiry program load request has been recognized.

### *INQUIRY INTERRUPT CONTROLLER*

The main functions of the Inquiry Interrupt Controller segment are: (1) to identify an initial inquiry interrupt, and (2) to periodically poll the terminals for inquiry activity. The Inquiry Interrupt Controller interfaces directly with the interpreter through the System Loader to load the individual phases and the Inquiry Controller/Handler. The I/O Manager interface enables the Interrupt Controller to receive and transmit the control characters of the polling sequence.

To reduce the amount of resident code, the Inquiry Interrupt Controller is subdivided into three phases; only one is resident in memory at a given time.

### *INQUIRY CONTROLLER/HANDLER*

The Inquiry Controller/Handler segment of the inquiry firmware is much larger and more powerful than the Inquiry Interrupt Controller. It is comprised of two parts: the Handler, and the Controller.

The Handler is that portion of code which provides the system interface when the user program is not being executed. The Handler performs the following functions:

a. Accepts the program name from the terminal. (The presence of a name is ascertained by the fact that the Interrupt Controller requested that the Handler be loaded into memory.)

b. Outputs a message (on the console) containing the terminal address of the active terminal and the requested program.

c. Checks the name length. If the length is less than five characters, it informs the terminal and console operators with the message "ILLEG PROG NAME".

d. Passes the program name to the System Loader for action.

The System Loader locates and loads the specified program. If this process is completed without error, control is passed on to the program. If there is an error condition, control is returned to the Handler and one of the appropriate error messages is displayed on both the terminal and the console.

Following the display of an error message and/or at end-of-job of the inquiry program, the Handler polls the terminals, starting with the terminal immediately following the last active terminal in the poll table. (This table is assembled at warmstart time in the order that the terminal addresses are entered and is the same list used by the Interrupt Controller in its polling.) If the polling indicates terminal activity, the Handler accepts the program name and proceeds as described above.

Polling is terminated when the Handler has completed one pass through the poll list, including the last active terminal, without any activity. At this point the "GO TO CONTENTION" message is issued, and the "**END INQUIRY**" message is printed on the console. The Handler now initiates a reload of the Inquiry Interrupt Controller, and the Inquiry system goes into the idle state.

The Controller portion of the Inquiry Controller/Handler operates the same as any other I/O controller in the system. Once an Inquiry program has begun execution, it attempts to perform a terminal operation (either a read or a write operation). During the actual message transfer, it is the Controller which takes the characters from the line and places them in the buffer for the read operation (or performs the reverse if the operation is a write).

The controller firmware is responsible for indicating errors when a problem between the terminal and program buffer sizes exists. Also, the controller ignores the parameters which have been entered with the program name when no initial parameters are accepted by the program. The COBOL program, however, is responsible for recovery when it expects parameters and the operator has entered none. If the operator at the terminal fails to enter data, or fails to allow the program to display data within the time limit set at Ready-Idle, the Controller timeout entrance terminates the operation and reports the condition to the inquiry program. The Controller does a limited number of retries in the case of garbled transmissions.

## INQUIRY SYSTEM PROGRAMMING

The inquiry capability implemented in B 720 Systems is designed to execute inquiry programs written in COBOL. The following features, capabilities, and restrictions must be considered when developing inquiry programs:

a. An inquiry program can be loaded *only* by requests from a terminal, not from the console.

b. A non-inquiry program *cannot* be requested from a terminal.

c. The inquiry program can control *only* the individual terminal which initiated the program, and operates only in conjunction with the disk (read-only mode) and the console. All other peripheral activity is inhibited.

d. If the program expects parameters with the program name, and the terminal operator does not enter them, the program is responsible for safe recovery from this condition.

e. The program is responsible for controlling the positioning of the terminal cursor, clearing the screen, and generating the control characters for terminal options (such as Forms Mode and Programmatic Mode Control).

f. Additional flexibility in error recover can be provided by employing "USE" routines.

g. The availability to the program of the terminal address characters, through the "MOVE" construct, permits flexibility in providing for the different memory sizes of the terminals.

h. Because there are several possible buffer sizes, the buffer size must be passed to the System Loader when it loads an Inquiry program.

The following list summarizes the inquiry constructs of a B 700 COBOL.

a. Attributes.

b. The MODE IS clause of the OBJECT-COMPUTER paragraph of the COBOL Environment Division.

c. The inquiry option of the SELECT statement of the COBOL Environment Division.

d. The inquiry options of the following seven verbs used in the Procedure Division of a COBOL program.

1. CLOSE.
2. IF.
3. MOVE.
4. OPEN.
5. READ.
6. USE.
7. WRITE.

NOTE

The use of the ZIP statement is not permitted in inquiry programs.

# SECTION 5

# READER SORTER SUBSYSTEM

## GENERAL

The capability of sorting and processing encoded documents is implemented in a B 720 System by use of a B 700 Reader Sorter Subsystem in an equipment system consisting of other required units and subsystems. Figure 5-1 shows the minimum B 721 System configuration required to implement reader sorter operation. The minimum configuration consists of:

a. A Central Processing Unit with a minimum of 32-bytes of user memory.

b. A B 9343 System Console and the B 346 Console Control.

c. A Reader Sorter Subsystem including the A 9135 or B 9136 Reader Sorter and the B 131 Reader-Sorter Control.

d. A Magnetic Disk Subsystem including an A 9480-12 or A 9481-12 Disk Cartridge Drive and a B 489-2 Disk Control.

e. A Line Printer Subsystem including an A 9247-2/12 or A 9249-1/2 Line Printer and the B 243 or B 244 Line Printer Control.

## READER SORTER SUBSYSTEM OPERATION

Figure 5-2 shows the general interface configuration of the Reader Sorter Subsystem. In application, the reader-sorter operates under control of instruction (control) words sent over the control line from the processor. These control words specify operations concerning the feeding and reading of documents by the reader-sorter (for example: start flow mode and stop feeding). Signals are transferred from the Processor to the Reader-Sorter Control over the MIR (Memory Information Register) Lines, along with data write information (pocket select or pocket light data) for the reader-sorter. Subsystem timing is provided by a 1-MHz clock from the processor.

Signals from the reader sorter to the processor consist of data read information, which is a binary-coded representation of the encoded numbers and symbols read from the documents (one four-bit digit at a time). Status information is also sent to represent the status of documents in transport (for example:

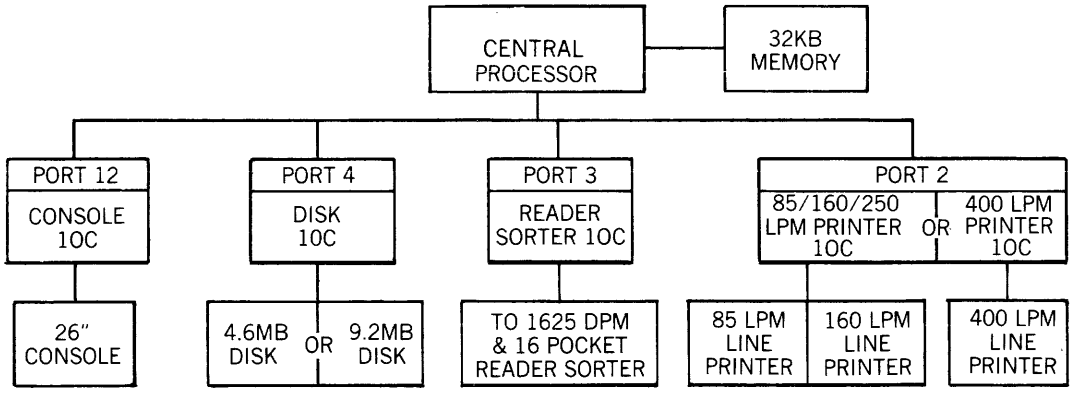



**Figure 5-1. Basic System Configuration for Reader Sorter Operation**

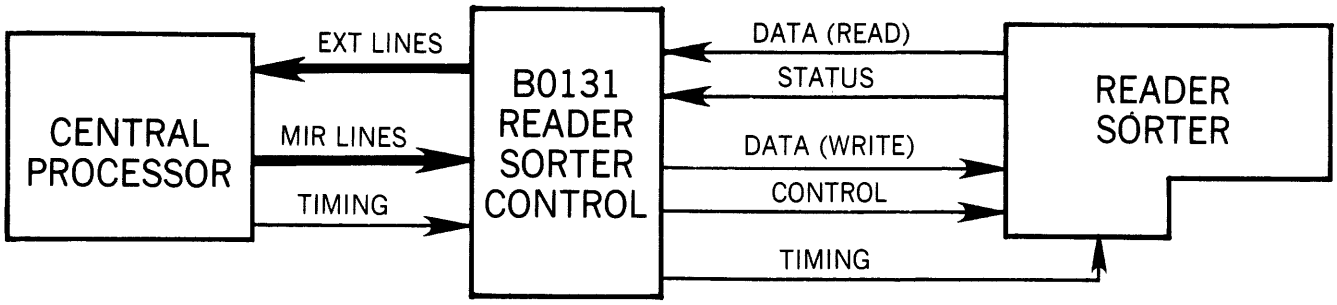



**Figure 5-2. Reader Sorter Subsystem Interface**

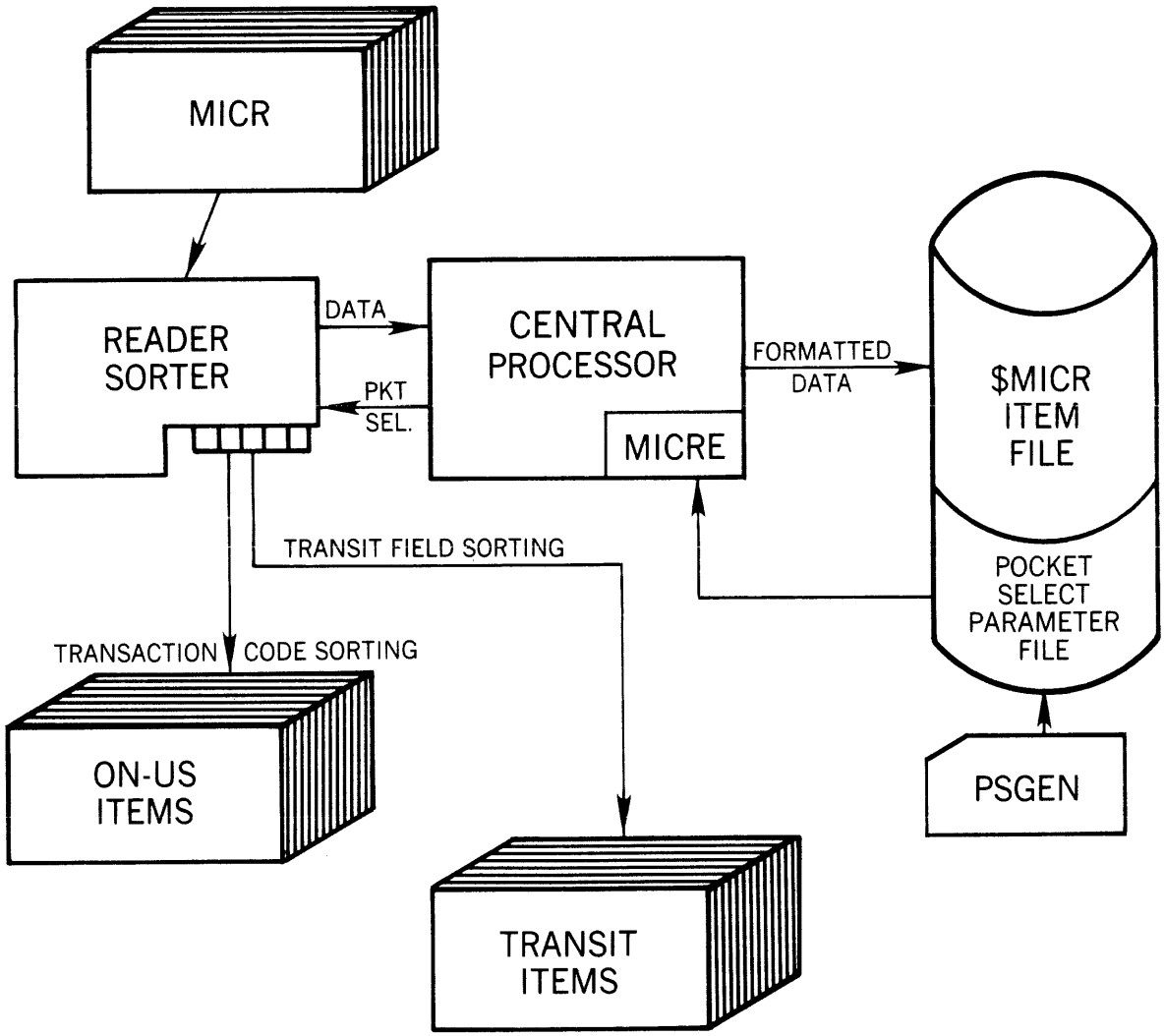



**Figure 5-3. MICR Entry System**

document has cleared the read area; document missort; jam; and sorter not ready). Device address information is also sent over the status lines. Both Data Read and Status information are sent to the processor over the EXT lines.

The subsystem is controlled by MICRE, which is an M-Level Program that builds document transaction files (called $MICR) on disk from items read by the reader-sorter. Pocket selection of items read is under control of the MICRE program using pocket select parameter files built by the use of the S-Level PSGEN utility program. Pocket selection is limited to the transit and transaction code fields. MICRE and PSGEN constitute a MICR Entry system which can be used to build files for a user written proof and transit program. The general operation of the MICR entry system is shown in Figure 5-3.

## READER SORTER CHARACTERISTICS

The A 9135 Reader Sorter used with B 720 Systems is capable of reading MICR-encoded documents at speeds up to 900 per minute (depending on document length). Two models are available: the A 9135-2 has eight sorting pockets, and the A 9135-3 has twelve sorting pockets (See figure 5-4.) Although offline sorting is available on these models, sorting is under control of the central processor when operating with the B 700 System.

A complete description of the Reader Sorter Subsystem is provided in the B 700 Reader Sorter Subsystem Reference Manual, form 1082500.

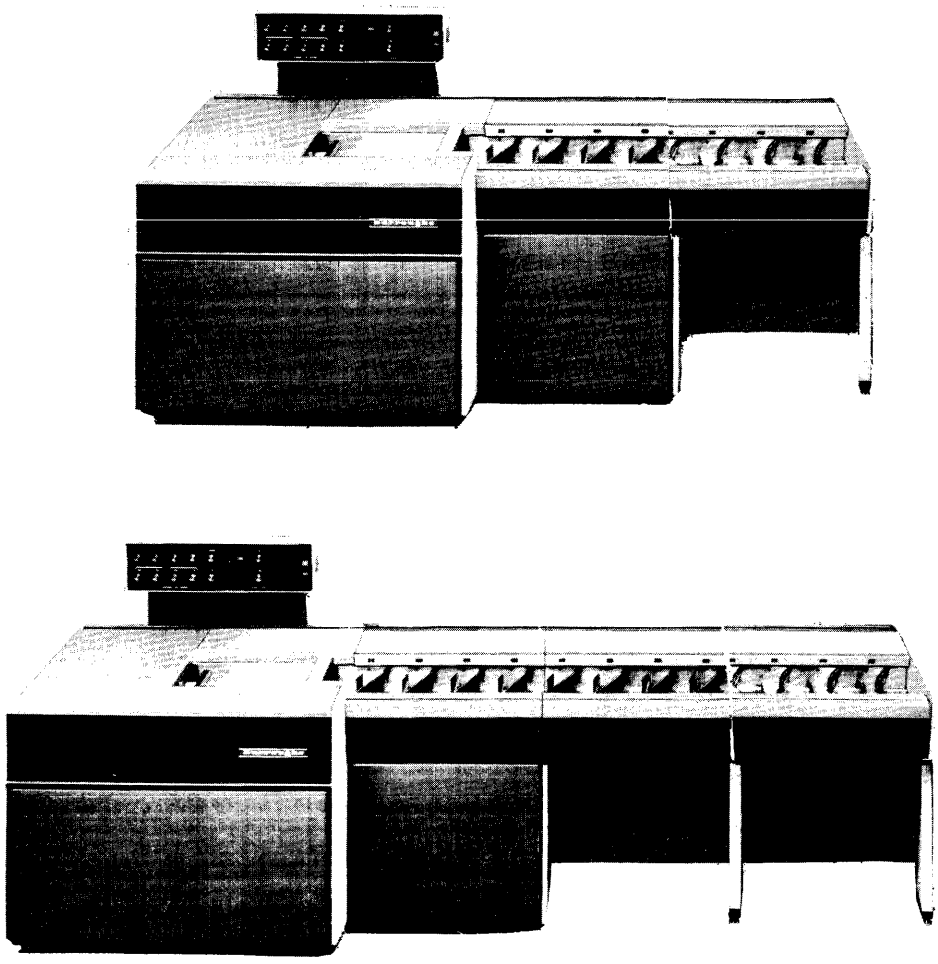


**Figure 5-4. A 9135 Reader Sorter**

# APPENDIX A

# ABBREVIATIONS, ACRONYMS, AND TERMS

This appendix defines the unique abbreviations, acronyms, and terms used in this manual.

| Abbr, Acronym, or Term | Meaning |
|---|---|
| Breakout | The interruption of a currently executing program to enable the loading and execution of another program. |
| B7CBL | B 700-Series onboard COBOL Compiler input program. |
| B7NDL | B 700-Series Network Definition Language Compiler input program. |
| B7RPG | B 700-Series Report Program Generator (RPG) Compiler input program |
| Communications Processor | A "front-end" preprocessing module used with the system processor to perform automatic data communications handling functions. |
| CPM | Central Processor Memory (also Data-Program Memory, DPM). Part of memory available for user programs and interpreter requirements. |
| CPU | Central Processing Unit (Central Processor). |
| Data Communications (System) | The combination of all the links, interface equipment, and system software required to effect transmission of information between communications stations. |
| Dynamic Interpreter Configurator | The non-commitment of resources until the actual need for these resources arises |
| DDP | Device-Dependent Port (circuit provisions in processor backplane to accommodate I/O control or interfacing circuitry). |
| DMA or DMAC | Direct Memory Access. Direct access to main memory be certain I/O controls. Unburdens processor to perform more useful functions. |
| File Inquiry (Operation) | The technique by which the contents of computer storage is interrogated from a remote terminal device. |
| Firmware | Combination of stored logic (microprogram) and uncomitted hardware logic. |
| Host Machine | The physical or basic system that accommodates interpreters and system programs to form a variation of a virtual machine. |
| Interpreter | An S-level program that fetches S-language instructions from main memory and interprets or executes them. |
| IOC | Input/Output (I/O) Control (interfacing device between processor and peripheral device or subsystem). |
| Line Adapter | Circuitry required to interface a particular type of line in a multiline data communications subsystem. |
| M-Language (M-Level) | Low-level programming language that uses microcode or microinstructions. |
| MICR | Magnetic Ink Character Recognition (reader-sorter capability). |
| MPM | Microprogram Memory. Main memory storage area for microinstructions. |
| NDL | Network Definition Language. A high-level language used in data communications applications. |

# APPENDIX A (Cont)

| Abbr, Acronym, or Term | Meaning |
|---|---|
| NPM | Nanoprogram Memory. Main memory storage area for nanoinstructions. |
| OCR | Optical Character Recognition (reader-sorter capability). |
| PPT/EPC | Punched-Paper Tape/Edge-Punched Card media. |
| Port | A backplane or functional area accommodating input/output interface controls (also DDP). |
| SLC | Single-Line Control (B 351-1 Data Communications Control). |
| S-Language (S-Level) | An intermediate programming language that uses instructions equivalent to the machine language of a conventional system. Each S-instruction represents a string of microinstructions to which the circuitry actually responds. |
| SM | Shared Memory. Memory shared by microprogram memory and S-level memory. |
| SPM | Scratchpad Memory. Part of user memory used primarily for an I/O workspace. |
| System Software | The programming elements comprising the operating system software essential to the startup and operation of the system. |
| Virtual Machine | The computer system, formed by interpreters and programs loaded into a host machine, to effect the implementation of the system. Independent of actual machine hardware. |

# APPENDIX B

# INSTRUCTION LIST

This appendix lists, in operation code (OP CODE) sequence, the instructions available to the system.

| Op Code | Instruction | Op Code | Instruction | Op Code | Instruction | Op Code | Instruction |
|---|---|---|---|---|---|---|---|
| 00 | ADD ACUM IX | 20 | | 44 | ADD LIT ACUM | 64 | |
| 01 | | 21 | | 45 | SUB LIT ACUM | 65 | COMN MEM/2 ACUM |
| 02 | MVE ACUM IX | 22 | | 46 | MVE LIT ACUM | 66 | |
| 03 | | 23 | | 47 | COMN LIT ACUM | 67 | SIND |
| 04 | | 24 | | 48 | ADD MEM ACUM | 68 | PNS- ACUM |
| 05 | | 25 | | 49 | ADD MEM/ 2 ACUM | 69 | PNS+ ACUM |
| 06 | MVD | 26 | | 4A | ADD ACUM MEM | 6A | PN ACUM |
| 07 | | 27 | | 4B | ADD ACUM MEM/2 | 6B | PC+- |
| 08 | REM/DATE | 28 | AL | 4C | SUB MEM ACUM | 6C | TK |
| 09 | LSR | 29 | AR | 4D | SUB MEM/ 2 ACUM | 6D | POS |
| 0A | INK | 2A | ALTO | 4E | SUB ACUM MEM | 6E | |
| 0B | | 2B | ARTO | 4F | SUB ACUM MEM/2 | 6F | |
| 0C | RST | 2C | ALR | 50 | CONV DB | 70 | SKC LGA |
| 0D | SET | 2D | OT | 51 | CONV BD | 71 | SKC LEA |
| 0E | CHG | 2E | BOIN/BOAL | 52 | LATA | 72 | SKC LLA |
| 0F | | 2F | CPMB | 53 | COMN MEM ACUM | 73 | SKC LUA |
| 10 | | 30 | SRR | 54 | MUL MEM ACUM | 74 | SKC AS |
| 11 | | 31 | PKA | 55 | DIV MEM ACUM | 75 | SKC ES |
| 12 | | 32 | PKB | 56 | MUL LIT ACUM | 76 | SKC ANS |
| 13 | | 33 | PKC | 57 | DIV LIT ACUM | 77 | SKC ENS |
| 14 | | 34 | CLKB, CT, RR, ALRM, STOP, OFF | 58 | | 78 | DUMP |
| 15 | | 35 | PC | 59 | | 79 | HASH |
| 16 | | 36 | | 5A | | 7A | BOZP |
| 17 | | 37 | RPOS/SPOS | 5B | | 7B | |
| 18 | NK ACUM | 38 | (TEST) | 5C | | 7C | TRAC |
| 19 | NKR ACUM | 39 | | 5D | | 7D | SETB |
| 1A | SK | 3A | SIZE | | | | |
| 1B | SKC AZ | 3B | KBT | | | | |
| 1C | | 3C | | | | | |
| 1D | LAT | 3D | | | | | |
| 1E | | 3E | | | | | |
| 1F | SKID | 3F | | | | | |
| 40 | MVE MEM ACUM | 60 | BRU | | | | |
| 41 | MVE MEM/ 2 ACUM | 61 | SRJ | | | | |
| 42 | MVE ACUM MEM | 62 | BRPS | | | | |
| 43 | MVE ACUM MEM/2 | 63 | BRC AZ | | | | |

# APPENDIX B (Cont)

| Op Code | Instruction | Op Code | Instruction | Op Code | Instruction | Op Code | Instruction |
|---|---|---|---|---|---|---|---|
| 5E | | 7E | RSTB | | | | |
| 5F | | 7F | LOAD | C4 | MVE FULL WORDS | E4 | TKM |
| 80 | BRC LGA | A0 | | C5 | MVE FULL WORDS | E5 | |
| 81 | BRC LEA | A1 | | C6 | SRCH RG | E6 | SRCH HI |
| 82 | BRC LLA | A2 | MAMK | C7 | SRCH EQ | E7 | SRCH LO |
| 83 | BRC LUA | A3 | | C8 | SEA EQ | E8 | ADD LIT MEM |
| 84 | BRC AS | A4 | BRU IA | C9 | SEA LW | E9 | ADD LIT MEM/2 |
| 85 | BRC ES | A5 | EDIT | CA | SEA LS | EA | SUB LIT MEM |
| 86 | BRC ANS | A6 | | CB | | EB | SUB LIT MEM/2 |
| 87 | BRC ENS | A7 | | CC | MVE MEM MEM | EC | MVE LIT MEM |
| 88 | ADD MEM/ LIT IX | A8 | | CD | MVE MEM MEM/2 | ED | MVE LIT MEM/2 |
| 89 | SUB MEM/ LIT IX | A9 | | CE | MVE MEM/ 2 MEM | EE | MUL LIT MEM IX |
| 8A | MVE MEM/ LIT IX | AA | | CF | MVE MEM/ 2 MEM/2 | EF | |
| 8B | COM MEM/ LIT IX | AB | LOGL | D0 | COMN MEM MEM | F0 | ADD MEM MEM |
| 8C | MVE IX MEM | AC | | D1 | COM MEM MEM | F1 | ADD MEM MEM/2 |
| 8D | | AD | | D2 | COMN MEM/2 MEM/2 | F2 | ADD MEM/2 MEM |
| 8E | | AE | CDC | D3 | | F3 | ADD MEM/2 MEM/2 |
| 8F | | AF | CDV | D4 | COMN LIT MEM | F4 | SUB MEM MEM |
| 90 | NK MEM | B0 | RETM | D5 | COMN LIT MEM/2 | F5 | SUB MEM MEM/2 |
| 91 | NKR MEM | B1 | LODM | D6 | ADDB LIT MEM | F6 | SUB MEM/2 MEM |
| 92 | | B2 | IATM LJ | D7 | SUBB LIT MEM | F7 | SUB MEM/2 MEM/2 |
| 93 | | B3 | IATM RJ | D8 | PNS- MEM | F8 | BRBS |
| 94 | EAM | B4 | DATA COMM | D9 | PNS+ MEM | F9 | BRBN |
| 95 | | B5 | | DA | PN MEM | FA | |
| 96 | | B6 | | DB | | FB | MVCH |
| 97 | PA | B7 | | DC | | FC | SRJS |
| 98 | | B8 | | DD | | FD | READ/WRITE/ MVBS/MVSB |
| 99 | | B9 | | | | | |
| 9A | | BA | | DE | | FE | DAC |
| 9B | (TEST) | BB | | DF | | FF | TBRU |
| 9C | | BC | | | | | |
| 9D | | BD | | | | | |
| 9E | | BE | | | | | |
| 9F | SCAN | BF | | | | | |
| C0 | CPRS | E0 | MV4 | | | | |
| C1 | EXPD | E1 | CMCH | | | | |
| C2 | (TKM SPECIAL) | E2 | CLC | | | | |
| C3 | SPRD | E3 | CLCS | | | | |

# APPENDIX C

# VIRTUAL MACHINE REGISTERS

This appendix lists, in mnemonic name sequence, the functional registers used in the virtual machine. The bit size, full name, function, and access means is listed for each register.

| Name | Size (bits) | Full Name | Function | Load By | Read By |
|---|---|---|---|---|---|
| ACUM | 64 | Accumulator | Arithmetic, shifting, field isolation, communication with system control, logical arithmetic, console input. | Various | Various |
| DBR | 16 | Descriptor Base Register | I/O descriptor table base address storage. | LODM | RETM |
| IX1 | 16 | Index Register One | Address positive increment. Some special purposes. | Various | Various |
| IX2 | 16 | Index Register Two | Same as IX1. | Various | Various |
| IX3 | 16 | Index Register Three | Same as IX1. | Various | Various |
| IX4 | 16 | Index Register Four | Special purpose counters or addresses. | Various | Various |
| BCPR | 16 | BCP (System Control Program) Communicate Register | Communication from system control program to S-level program. | - | RETM |
| LLCR | 8 | Left Platen Forms Count Register | Forms control for line count of left platen (L-series console). | LODM | RETM |
| LLLR | 8 | Left Platen Forms Limit Register | Forms control for line count limit of left platen (L-series console). | LODM | RETM |
| LPKR | 16 | Program Key Table Register | Holds base address of active program key (PK) table. | LODM | RETM |
| LPNR | 16 | Numeric Print Mask Address Register | Holds base address of active numeric print mask table. | LODM | RETM |
| LRCR | 8 | Right Platen Forms Count Register | Forms control for line count of right platen (L-series console). | LODM | RETM |
| LRLR | 8 | Right Platen Forms Limit Register | Forms control for line count limit of right platen (L-series console.) | LODM | RETM |

# APPENDIX C (Cont)

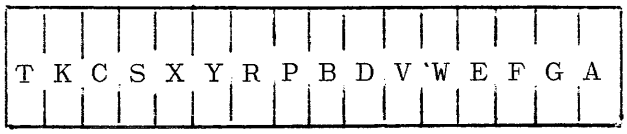
| Name | Size (bits) | Full Name | Function | Load By | Read By |
|---|---|---|---|---|---|
| POS | 8 | Desired Print Position Register | L-series console printer positioning. | POS; various print commands | - |
| REM | 64 | Remainder Register | Holds: a. Remainder after DIV b. Scaled-off digits after MUL. | MUL, DIV | REM |
| RR | 1 | Ribbon Register | Controls ribbon color of L-series console. | RR; various print commands | - |
| SR (LSR) | 5 | Shift (or Scale) Register | Controls scaling and/or rounding for MUL and DIV. | LSR | - |
| SRJC | 16 | SRJ Stack Pointer Register | Points to latest entry in 8-deep circular subroutine "return to" stack. | SRJ; SRJS; TBRU | SRR |
| SRJS | 16 | SRJS Stack Pointer Register | Points to latest entry in 4-deep circular subroutine "return from" stack. | SRJS | SRR |
| TDNO | 16 | Inquiry Terminal Identifier | Holds two-character identifier of active inquiry terminal | - | RETM |

# APPENDIX D

# FLAG LIST

This appendix shows the layout and functions of the flags available in the flag register of the virtual machine.

FLAG GROUP LAYOUT:



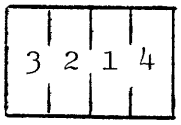

GROUP INDEX NUMBER 0 1 2 3 4 5 6 7 8 9 A B C D E F

| FLAG LAYOUT | | FLAG WORD |
|---|---|---|
| TEST FLAGS<br>GROUP 0 | T GROUP<br>P I L O | P-Console paper limit exceeded<br>I-Index overflow<br>L-Forms limit<br>O-Overflow |
| CONTROL KEY FLAGS<br>GROUP 1 | K GROUP<br>3 2 1 4 | 1 - OCK I<br>2 - OCK II<br>3 - OCK III<br>4 - OCK IIII |
| COMPARISON FLAGS<br>GROUP 2 | C GROUP<br>H E L U | H - High<br>E - Equal<br>L - Low<br>U - Reserved |

# APPENDIX D (Cont)

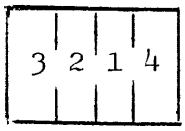
General Purpose Groups

S, X, Y, R, P, B, D, V, W, E, F
(meaning assigned by program usage)




SKID - Operator recoverable error flags
GROUP 14

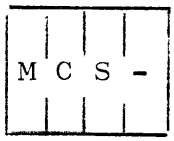
G GROUP




1 - Shifted 1
2 - Shifted 2
3 - Shifted 3
4 - Shifted 4

Accumulator sign flags
GROUP 15

A GROUP




M - Per thousand
C - Per hundred
S - Special purpose
- - Negative

cut along dotted line

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 720 SYSTEMS
Reference Manual

FORM: 1082484
DATE: 4-75

CHECK TYPE OF SUGGESTION:

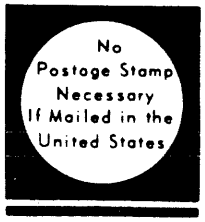
☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____ DATE _____
TITLE _____
COMPANY _____
ADDRESS _____
_____

STAPLE

FOLD DOWN SECOND FOLD DOWN


**BUSINESS REPLY MAIL**
First Class Permit No. 46, Wayne, Pennsylvania

**Burroughs Corporation**
200 West Lancaster Avenue
Wayne, Pennsylvania 19087

Attn: Systems Documentation
TIO EAST

FOLD UP FIRST FOLD UP


1082484 4-75 Printed in U.S. America