Burroughs
B6500
Information Processing Systems
REFERENCE MANUAL

Burroughs

B 6500

INFORMATION PROCESSING SYSTEMS

REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

$5.00

Printed in U.S. America 9-69 1043676

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

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, Sales Technical Services, Burroughs Corporation, 6071 Second Avenue, Detroit, Michigan 48232.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

LIST OF ILLUSTRATIONS (cont)

LIST OF ILLUSTRATIONS (cont)

LIST OF ILLUSTRATIONS (cont)

LIST OF TABLES

INTRODUCTION

The Burroughs B 6500 is a medium to large, high speed Information Processing System. Some features that are incorporated in this system include:

a. Monolythic Circuitry.

b. Memory expandable to 524,288 words.

c. Memory Cycle Times of 1.2 microseconds or 600 nanoseconds.

d. Peripheral configuration expandable to 256 units.

e. Dual Input/Output Multiplexor permitting up to 20 simultaneous Input/Output (I/O) operations.

f. Data Communication Software for remote computing and file manipulation.

g. Disk File storage over 36 billion bytes (8-bit characters).

A unique hardware design, developed from years of successful experience with the B 5000 series, has resulted in the parallel design of the B 6500 hardware and software. Where traditionally hardware was designed prior to software development, parallel design assures that the hardware contains all necessary logic for efficient software packages, which in turn optimizes hardware capabilities. The B 6500 design affords a general "re-entrant" technique which permits multiple users to share a common object program. In addition, the systems further expand the use of hardware stack organization used in the B 5500. For example, the Segment Dictionary, a separate table for each program in the B 5500, has been placed in the base of the program stack in the B 6500. This part of the stack is used for multiple executions of the same program, thus implementing in the hardware many of the bookkeeping functions required to implement Master Control Program (MCP) re-entrancy.

To provide dynamic storage allocation, the B 6500 system employs and expands upon the Burroughs descriptor method of segmentation, first used on the B 5500, in lieu of some form of fixed-sized "paging" technique.

Designed to bring the user simplified programing, operational ease, and complete freedom of system expansion, the B 6500 offers a choice of three problem-oriented languages: COBOL for business applications and ALGOL and FORTRAN for solution of mathematical problems. Operator intervention is minimized by the MCP, which provides for complete system management.

The complete flexibility of programing and control of the processing pattern provides the B 6500 with smooth growth potential. Starting with a minimum configuration, the user may expand his system in small increments to accommodate a growing work-load. With each addition, the MCP automatically adjusts to attain increased system production and efficiency, expanding system multiprograming capabilities.

This reference manual describes the hardware characteristics of the B 6500 system. Because of the design concept of the B 6500, there exists a strong interdependence between the hardware and the Master Control Program (MCP). This material pertains only to the hardware considerations, whereas the MCP is discussed in a separate manual.

SECTION 1
SYSTEMS DESCRIPTION

GENERAL.

This manual explains how the B 6500 Information Processing System achieves flexibility and efficiency through a comprehensive system approach to problem solving without considering the areas of computer logic or circuit design. The program-independent modular system design efficiently uses available units to process programs and also permits system configuration changes without the need to reprogram or recompile. This approach also offers the user the advantages of simplified programing, ease of operation and a complete freedom of system expansion. The B 6500 is a compiler oriented system designed to accept the common languages; ALGOL, COBOL, and FORTRAN. The systems automatically handle memory assignments, program segmentation and subroutine linkages, eliminating many of the arduous programing tasks that are likely to produce errors. The programs are debugged and corrected at the source language level.

DESCRIPTION OF UNITS.

The B 6500 system configuration varies with application and workload requirements. The basic system includes one processor, one maintenance test routine processor, one system control and one desk console. The maximum system configuration includes 2 processors, 32 memory modules, 2 input/output multiplexors, 20 peripheral controls, 8 data communications processors, and 256 peripheral units. The central units are defined in table 1-1. The peripheral units available with this system along with their characteristics, are listed in Section 9. The Data Communication Sub-System is defined in Section 10.

Table 1-1

B 6500 Central Units Chart

Style Number	Description	Notes
B 6503	Basic System	2.5 megahertz clock
B 6504	Basic System	5.0 megahertz clock
B 6506	Basic System	5.0 megahertz clock
B 6503-1	Second Processor	2.5 megahertz clock
B 6504-1	Second Processor	5.0 megahertz clock
B 6506-1	Second Processor	5.0 megahertz clock
B 6713	Multiplexor, 4 data switching channels	1 allowed per B 6503 system
B 6713-1	Additional data switching channel	
B 6714	Multiplexor, 4 data switching channels	
B 6714-1	Additional data switching channel	
B 6716	Multiplexor, 4 data switching channel	
B 6716-1	Additional data switching channel	
B 6000	Optional Memory Control Cabinet	

Table 1-1 (cont)
B 6500 Central Units Chart

Style Number	Description	Notes
B 6001-2	98,304 Bytes (16,384 words)	1.2 microsecond memory for B 6503 and B 6504 systems.
B 6002-2	196,608 Bytes (32,768 words)	
B 6003-2	294,912 Bytes (49,152 words)	
B 6004-2	393,216 Bytes (65,536 words)	
B 6005-2	491,520 Bytes (81,920 words)	
B 6006-2	589,824 Bytes (98,304 words)	
B 6007-2	688,128 Bytes (114,688 words)	
B 6008-2	786,432 Bytes (131,072 words)	B 6008-2 is the maximum memory size permitted for the B 6503 system.
B 6010-2	983,040 Bytes (163,840 words)	
B 6012-2	1,179,648 Bytes (196,608 words)	
B 6016-2	1,572,864 Bytes (262,144 words)	
B 6020-2	1,966,080 Bytes (327,680 words)	
B 6024-2	2,359,296 Bytes (393,216 words)	
B 6032-2	3,145,728 Bytes (524,288 words)	

Table 1-1 (cont)
B 6500 Central Units Chart

Style Number	Description	Notes
B 6001-3	98,304 Bytes (16,384 words)	600 nanosecond memory for the B 6506 systems.
B 6002-3	196,608 Bytes (32,768 words)	
B 6003-3	294,912 Bytes (49,152 words)	
B 6004-3	393,216 Bytes (65,536 words)	
B 6005-3	491,520 Bytes (81,920 words)	
B 6006-3	589,824 Bytes (98,304 words)	
B 6007-3	688,128 Bytes (114,688 words)	
B 6008-3	786,432 Bytes (131,072 words)	
B 6010-3	983,040 Bytes (163,840 words)	
B 6012-3	1,179,648 Bytes (196,608 words)	
B 6016-3	1,572,864 Bytes (262,144 words)	
B 6020-3	1,966,080 Bytes (327,680 words)	
B 6024-3	2,359,296 Bytes (393,216 words)	
B 6032-3	3,145,728 Bytes (524,288 words)	

SYSTEM OPTIONS AND REQUIREMENTS.

The following list of requirements and options are available for the B 6500 systems:

a. A minimum of one special D.C. module is required in a B 6500 system. It can be installed in the following cabinets:

   1) Multiplexor.
   2) Processor.
   3) Peripheral Control.
   4) Data Communications.

b. A minimum of one ±12 volt inverter module is required in a B 6500 system. It can be installed in the following cabinets:

   1) Multiplexor.
   2) Processor.
   3) Peripheral Control.

NOTE

This module precludes the use of any other module in a cabinet.

c. A Flip Flop display supply module is required on the system and must be installed in the Multiplexor cabinet.

d. The Memory cabinets each must contain a special Memory supply for developing the regulated voltages required for the memory operation.

e. Each cabinet must contain an inverter for supplying power to its regulators. A 600 amp inverter is required in the Processor, Multiplexor and Data Communications cabinets. All other cabinets require a 400 amp inverter.

AUXILIARY CABINET.

Peripheral unit exchanges are located within auxiliary cabinets on the B 6500 system. This cabinet can accommodate varying combinations of exchanges depending on their physical size. Two of the various combinations that are possible are shown in figure 1-1.

Figure 1-1. Auxiliary Cabinets

The following exchanges are available for use on the B 6500 system.

a. Tape Exchange
   2X10
   2X8
   4X16

b. Disk File Exchange
   1X2
   2X5
   4X10
   4X20

SYSTEM POWER.

Main power is supplied to the system by 1 to 15 free standing A.C. power cabinets. Each power cabinet can furnish enough power for eight B 6500 cabinets. The power cabinets receive 3 phase A.C. from the wall breakers and convert it to 220 volt pulsating direct current. Each B 6500 cabinet contains an Inverter which supplies the regulated supply voltage required for use in its own component sections.

The AC module contains an AC control, the AC/DC converter and a OV/UV indication panel. Refer to figure 1-2 for a typical B 6500 power supply configuration.

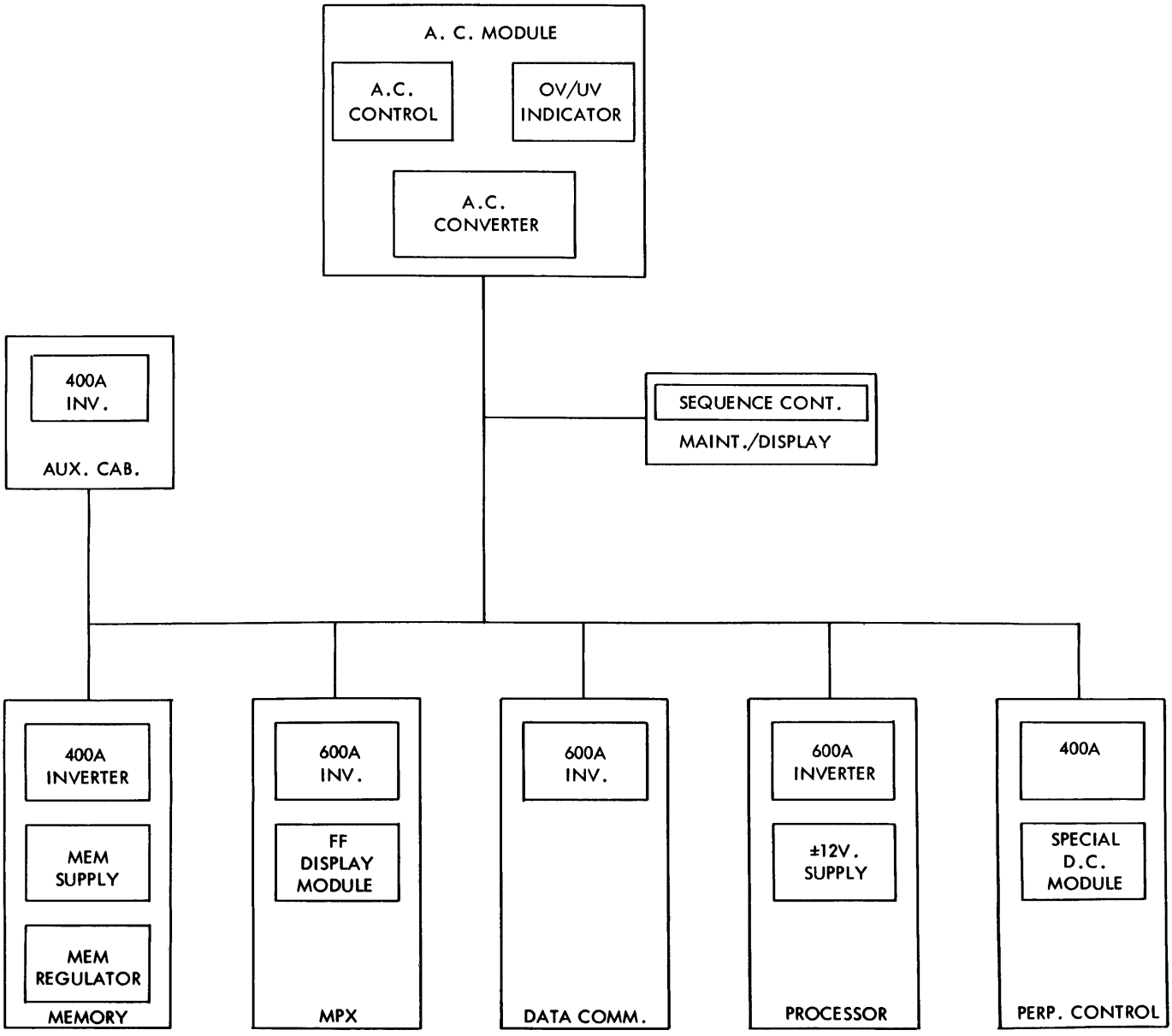


Figure 1-2. B 6500 Power Supply

PERIPHERAL CONTROL CABINET.

The PC cabinet can accommodate up to 10 peripheral controls. A maximum of 5 large controls can be used with 5 small controls, however, more than 5 small controls are possible if used in place of the large controls.

The following controls are available:

a. Large
   1. Magnetic tape
   2. Disk file
   3. Console Display

b. Small
   1. Card reader
   2. Card punch
   3. Line printer
   4. Paper tape reader
   5. Paper tape punch

The large control has a two byte buffer and the small control contains a one byte buffer, therefore either 8 or 16 bits may be transferred in parallel to the Multiplexor at a time. Local operations are performed by attaching a "Control switch" plug-on and two "Indicators" plug-ons to various cards in the control.

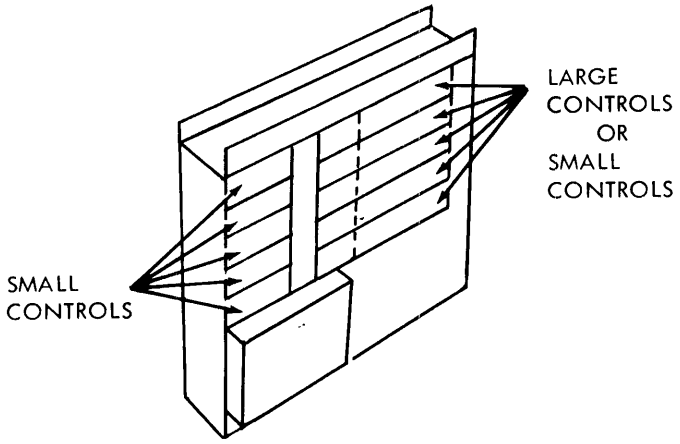


Figure 1-3. Peripheral Control Cabinet

SYSTEM ORGANIZATION.

Computer systems are generally organized around a central system that controls memory accesses, establishes I/O priority etc. In the B 6500 system this central control function has been distributed throughout the system by providing each peripheral unit with an associated control (figure 1-4). These peripheral controls, in conjunction with the multiplexor, provide independent but controlled access to main memory for each peripheral unit. The peripheral activity is supervised by the MCP which assigns outgoing data to the proper units or calls for required input data from others. Because the MCP is constantly aware of the available environment, the user program is efficiently executed whether units have been deleted for preventive maintenance or added because of increased work loads.

MASTER CONTROL PROGRAM.

The Master Control Program (MCP) provides overall system coordination and control of processing on the B 6500 system, minimizing operator intervention. The MCP obtains maximum use of the system components by controlling the sequence of processing, initiating all input/output operations and providing automatic handling procedures to meet virtually all processing conditions. Because many functions are performed under MCP control, changes in scheduling, system configuration and program size are readily accommodated.

CLOCKS.

The MCP for the B 6500 makes use of two hardware clocks: The real time clock and the interval timer. The real time clock has a 2.4 microsecond resolution and counts up to 24 hours. It is used by the MCP logging routines to provide extremely accurate timing information and also can be read by application programs. This clock is associated with the multiplexor and runs continuously, even when the processors are halted. The interval timer is a clock (one in each processor), which provides a predetermined timed interrupt for "time-slicing", loop hang-up etc. This interval varies from 512 microseconds to one second, in 512 microsecond intervals.

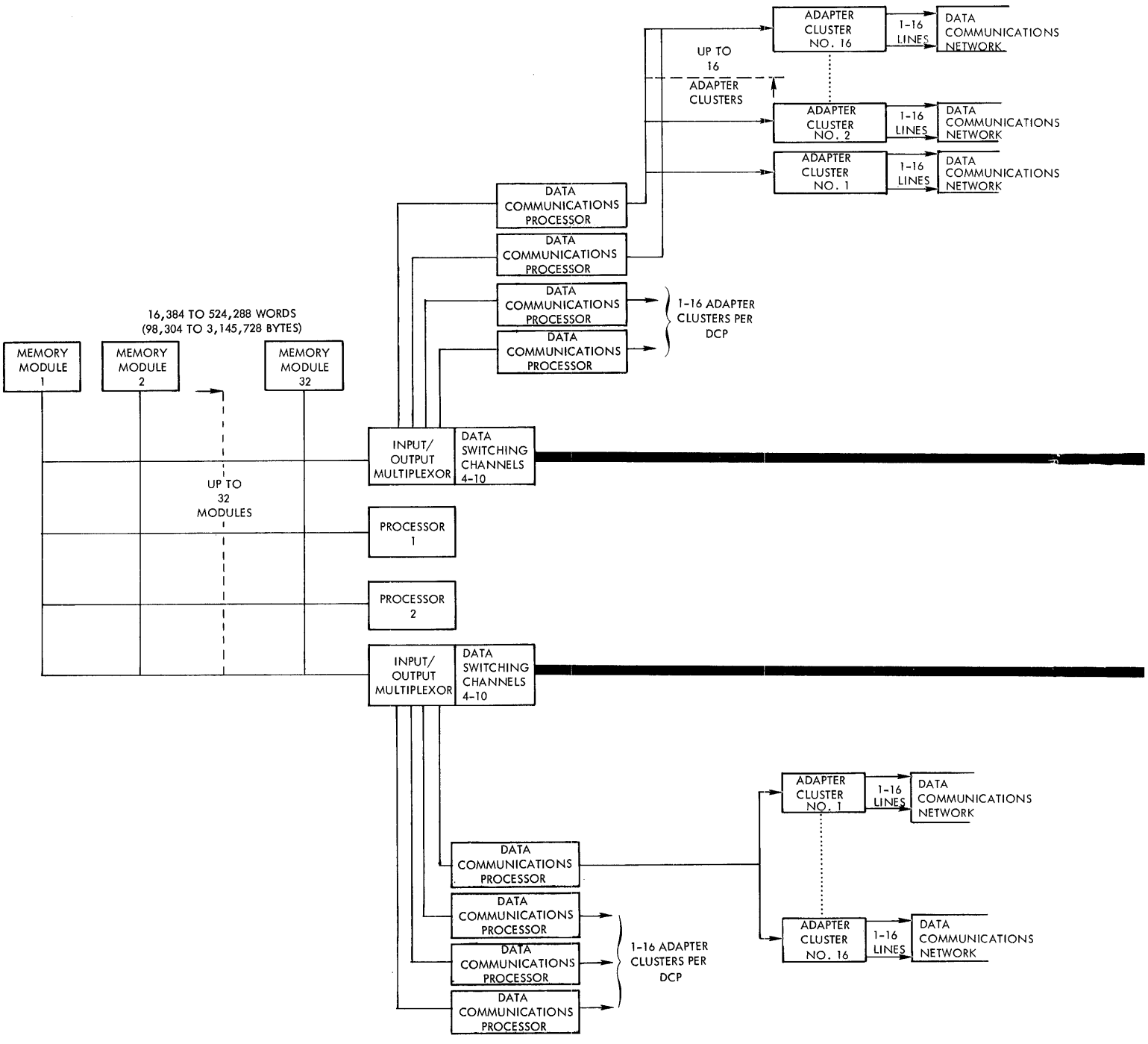


Figure 1-4. B 6500 Schematic Diagram (sheet 1 of 2)

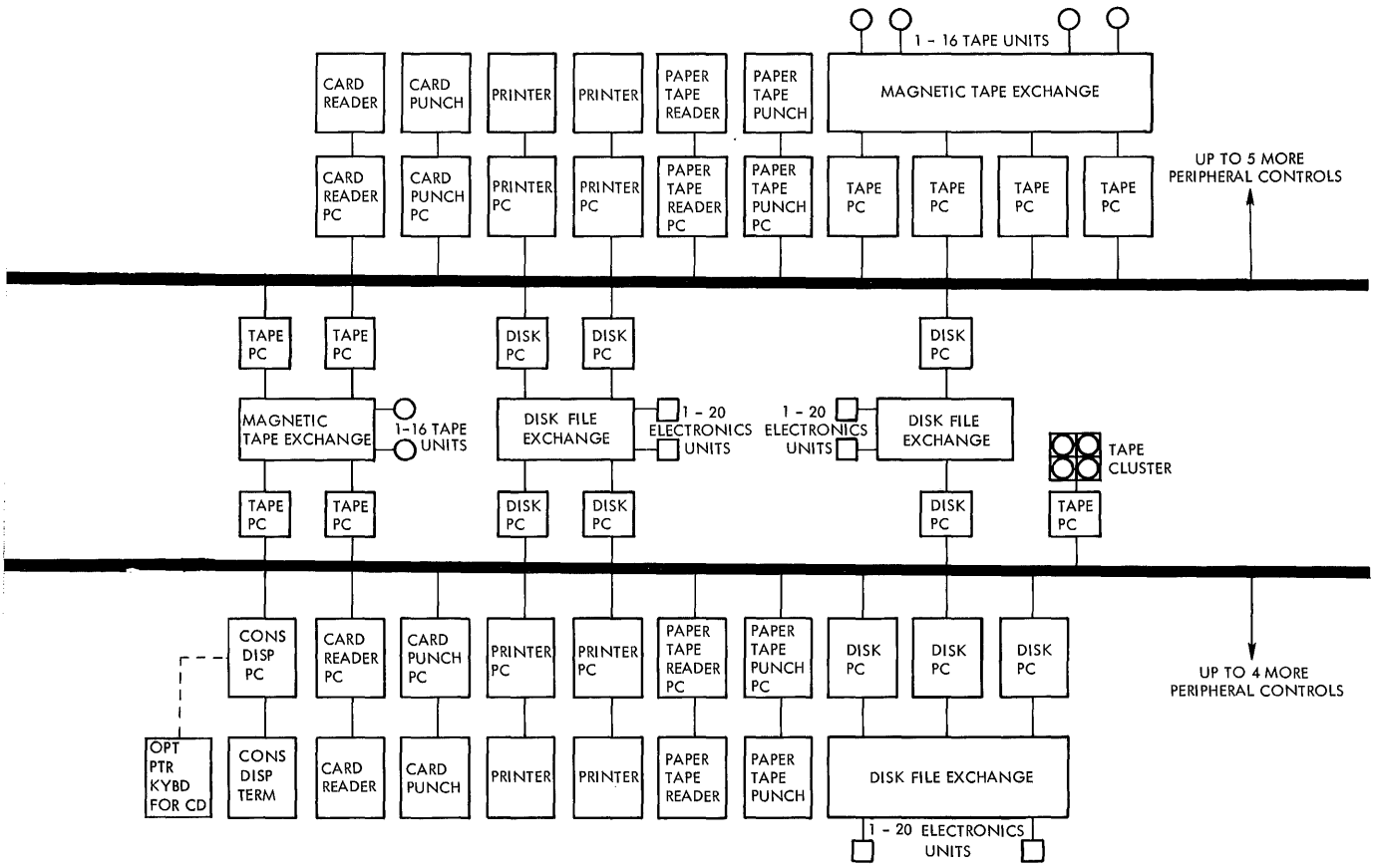


Figure 1-4. B 6500 Schematic Diagram (sheet 2 of 2)

PROCESSOR.

The B 6500 system accommodates either one or two processors, both capable of accessing any portion of total memory.

All B 6500 processors are parallel machines; the B 6503 has a clock frequency of 2.5 megahertz, the B 6504/6506 a clock frequency of 5 megahertz . Processors with different clock rates cannot be intermixed on the same system. The processor is basically word oriented, but has extensive multiword string manipulation capabilities for 4-bit, 6-bit, and 8-bit characters.

PROCESSOR STATES.

The processor operates in either of two states: control state for the MCP or normal state for user programs and certain MCP functions. In a dual-processor system either processor may handle external interrupts. Both processors may be in control state at the same time.

CONTROL STATE. Entry into a control state occurs when the processor enters or returns to a procedure marked as a control state procedure, or executes a Disable External Interrupts operator. In control state the handling of external interrupts is inhibited while the processor executes privileged instructions not available in normal state. Exit from control to normal state occurs whenever the MCP initiates a normal state procedure, exits back to a normal state procedure or executes an Enable External Interrupt operator. After an interrupt, return to the user's program may or may not be to the program that was operating when the interrupt occured.

NORMAL STATE. Normal state excludes use of privileged instructions required by the MCP and allows external interrupts. Exit from normal state occurs as a result of a Disable External Interrupt operator or by a call to a control state procedure; e.g., to initiate I/O. Many MCP functions are executed in normal state.

FEATURES.

Some of the processor features are:

a. Program code cannot be modified while in residence.

b. Hardware stack features provide efficient handling of temporary storage and subroutine requirements.

c. Control bits in each word provide efficient MCP or hardware action, depending upon the state of the control bits.

d. Memory protection, which prevents one program from affecting another, is provided by a combination of hardware and software features. Hardware features include detection of program attempts to index beyond an assigned data area. Another feature includes the use of a memory protect bit in each word to prevent a user program from altering program segments, data descriptors, segment descriptors, memory links, MCP tables, etc. The memory protect bits are set by the software. Attempts to alter information with this protect bit set will inhibit the write operation and generate an interrupt.

e. The B 6500 processor is designed to implement higher-level languages and to function under MCP control.

f. Major registers and control flip-flops in each of the processors contribute to system multiprocessing capabilities.

INTERRUPT SYSTEM.

The method of detecting and servicing system interrupts contributes to the ability of the B 6500 to process a mix of independent programs in an efficient manner. Under the constant, automatic management of the MCP, multiprocessing is the normal mode of operation. With one processor in the system, multiprograming (interlevel processing) is employed. A dual processor B 6500 System combines both multiprograming and parallel processing. The ability to multiprogram, parallel process, or both is defined as multiprocessing.

Extensive interrupt facilities initiate specific routines in the Master Control Program (MCP). Since the MCP maintains a central communications control, the interrupt transfers control to the MCP initiating operations that can proceed simultaneously with computation. Some MCP functions are: data transfer control, input/output control, error detection, etc.

There are two interrupt conditions: Internal (Processor Dependent) or External (Processor Independent). Each processor in the B 6500 system is provided with a private internal interrupt network to handle the processor dependent interrupt. Interrupts generated within the processor are fed into this network and retained until serviced by that processor. The processors also share the handling of external interrupts generated by input/output operations occuring on either Multiplexor. The command structure in conjunction with a stack provides for implementation of string notation and automatic linking of subroutines.

INTERRUPT HANDLING.
An interrupt causes the processor to initiate the following subroutine:

a. Mark the stack.

b. Insert an Indirect Reference Word into the stack, which addresses a reserved location of the stack where a link to the MCP interrupt routine has been stored.

c. Push all pertinent registers into the stack.

d. Insert into the stack an integer value defining the interrupt.

e. Insert a second parameter into the stack, giving other information about the interrupt.

f. Execute an Enter operator.

The MCP processes the interrupt when it recognizes the Enter Operator. The MCP reactivates the interrupted object program by returning through the normal subroutine mechanism.

OPERATOR DEPENDENT PROCESSOR INTERRUPTS.
The interrupts listed below are set only by the action of operators.

a. Presence bit.
b. Invalid index.
c. Exponent underflow.
d. Exponent overflow.
e. Interger overflow.
f. Divide by zero.
g. Invalid operand.
h. Bottom of stack.
i. Sequence error.
j. Segmented array.
k. Memory protect.
l. Programed operator.

Within a processor, only one operator dependent interrupt is set at any one time.

OPERATOR-INDEPENDENT PROCESSOR INTERRUPTS.
The operator-independent interrupts include:

a. Memory parity.
b. Stack overflow.
c. Invalid address.
d. Interval timer.
e. Instruction timeout.
f. Scan buss parity.
g. Stack underflow.
h. Invalid program word.
i. MPX parity.
j. Loop.

EXTERNAL INTERRUPTS.

External interrupts are fed into the processor interrupt system. If the interrupt network is disabled on one processor, the external interrupt signal is routed to the other, since both processors in a dual-processor system are able to respond and process external interrupts independently and simultaneously. The ability of either processor to handle interrupts is made possible because of a distributed interrupt network and the ability of both processors to be in control state at the same time. The activities of two processors in control state are coordinated (interlocked) by the software through the use of the Read With Lock mechanism. If both processors are handling interrupts, additional interrupts are retained for future processing.

A unique literal value is assigned to each external interrupt condition. This literal value is transmitted to the processor and placed into the stack as the processor acknowledges the external interrupt and enters the interrupt sequence.

The external interrupts include:

a. Processor to Processor.
b. I/O Finish.
c. Data Comm. Att'n Needed.
d. General Control Adapter.
e. External Interrupt (piggyback MPX).
f. Change of peripheral-unit status.

MAIN MEMORY.

Main memory is expandable from one to eight modules on a B 6503 system, and from one to 32 modules on B 6504 and B 6505 systems. Each memory module contains 16,384 words permitting a current maximum memory size of 524,288 words. Future provisions will allow for over one million words of storage.

MEMORY WORDS.

Each memory word contains 48 information bits, three control bits, and a parity bit. The three control bits are used to identify descriptors, provide memory protection, describe the type of data, and provide other control functions. The twenty-bit binary combinations can provide up to 1,048,576 memory addresses, though presently only 524,288 are used. Odd parity is used to check validity of information storage and transfers in the B 6500 system.

Each system has a memory test facility used for fault detection and isolation. When the unit test facility is used to check one of the modules, the others are available to the system.

MEMORY CYCLE TIMES.

The memory cycle time is 600 nanoseconds for the B 6506 systems and 1.2 microseconds for the B 6503 and B 6504 systems.

SECOND LEVEL MEMORY.

Burroughs unique head-per-track disk file subsystem provides the user with virtually unlimited expansion capability. The 20 to 60 millisecond average access time of the various disk file models permits extremely large programs and data segments to be stored on the disk and brought into main memory by the MCP when required.

INPUT/OUTPUT MULTIPLEXOR.

The Input/Output Multiplexor and associated peripheral control modules are used to control the transfer of data between memory and all peripheral equipment, independent of the processor. The multiplexor receives instructions from the processor and, with it's associated peripheral controls, executes these instructions. One or two multiplexors may be used with the B 6500 System. Each multiplexor is capable of processing up to ten simultaneous I/O operations with up to 20 peripheral units.

MULTIPLEXOR CONFIGURATION.

Each multiplexor provides four separate and independent units:

a. Data switching channels which provide the necessary linkage between the peripheral device (excluding data communications) and main memory.

b. Data communications processors which permit interfacing of remote devices to the B 6500.

c. A real time adapter which permits interfacing of real time devices such as wind tunnels and rocket stands.

d. The peripheral system configuration tables for software use.

DATA SWITCHING CHANNELS.

The number of data switching channels determines the number of simultaneous I/O operations that can be performed. The channels float, assigned by the multiplexor to peripherals upon initiation of an operation and released to the multiplexor for reassignment upon completion.

PERIPHERAL CONTROLS.

Two types of peripheral controls are available, large and small. The large controls are used with high-speed devices such as magnetic tape, disk files, and display consoles; the small controls are used with slower peripherals such as printers, card readers, and card punches. The large controls contain a two byte buffer and the small a one byte buffer. Each multiplexor can accommodate up to ten large and ten small controls. A small control may occupy a large control position.

SYSTEM EXPANSION.

The maximum configuration with two multiplexors (20 controllers per multiplexor) can be expanded further through the use of disk file and magnetic tape exchanges. Figure 1-5 illustrates a possible magnetic tape subsystem. Figure 1-6 illustrates a possible disk file subsystem.

PERIPHERAL CONTROL BUS.

A peripheral control (P.C.) bus extends from the multiplexor to the various peripheral controls (figure 1-7). Information in one- or two-byte groups can be sent along the bus to or from any peripheral control every 1.2 microseconds.

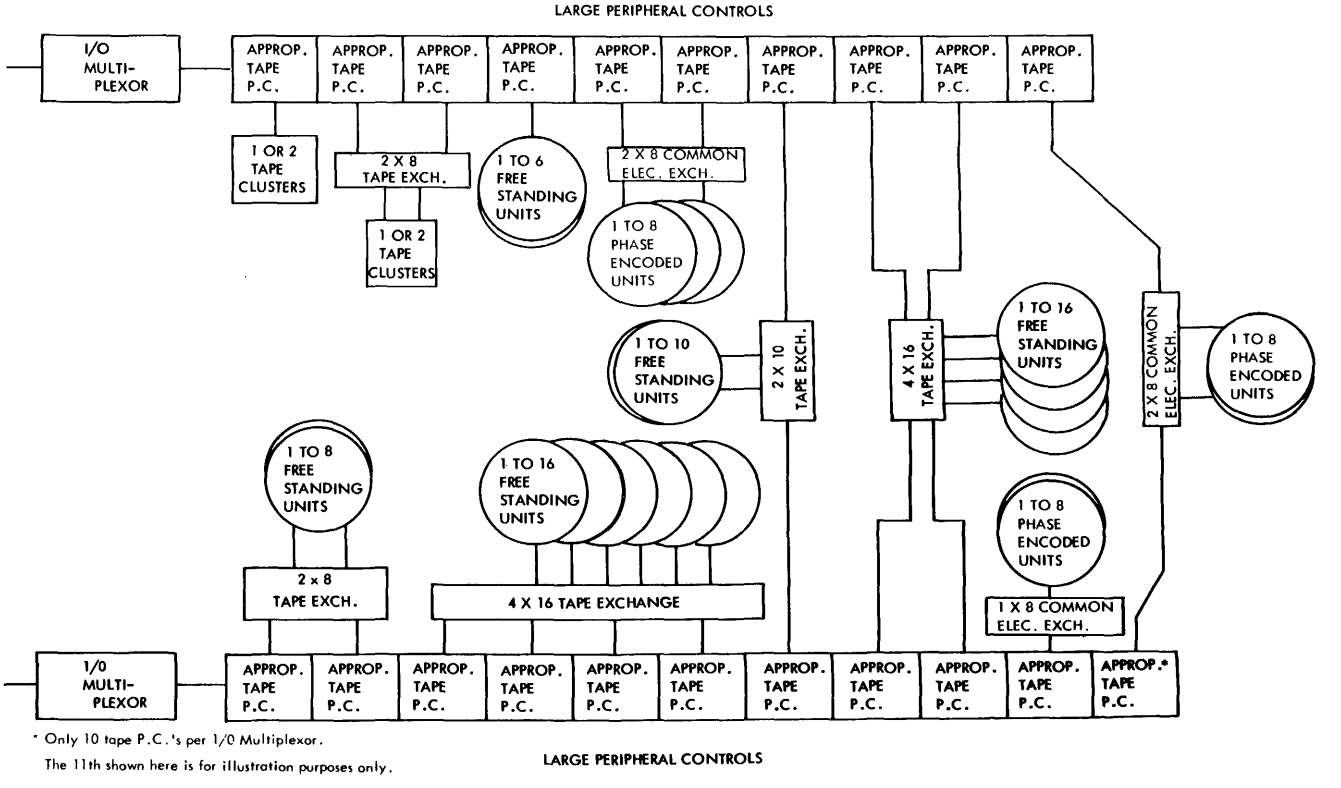


Figure 1-5. Possible Magnetic Tape Subsystem

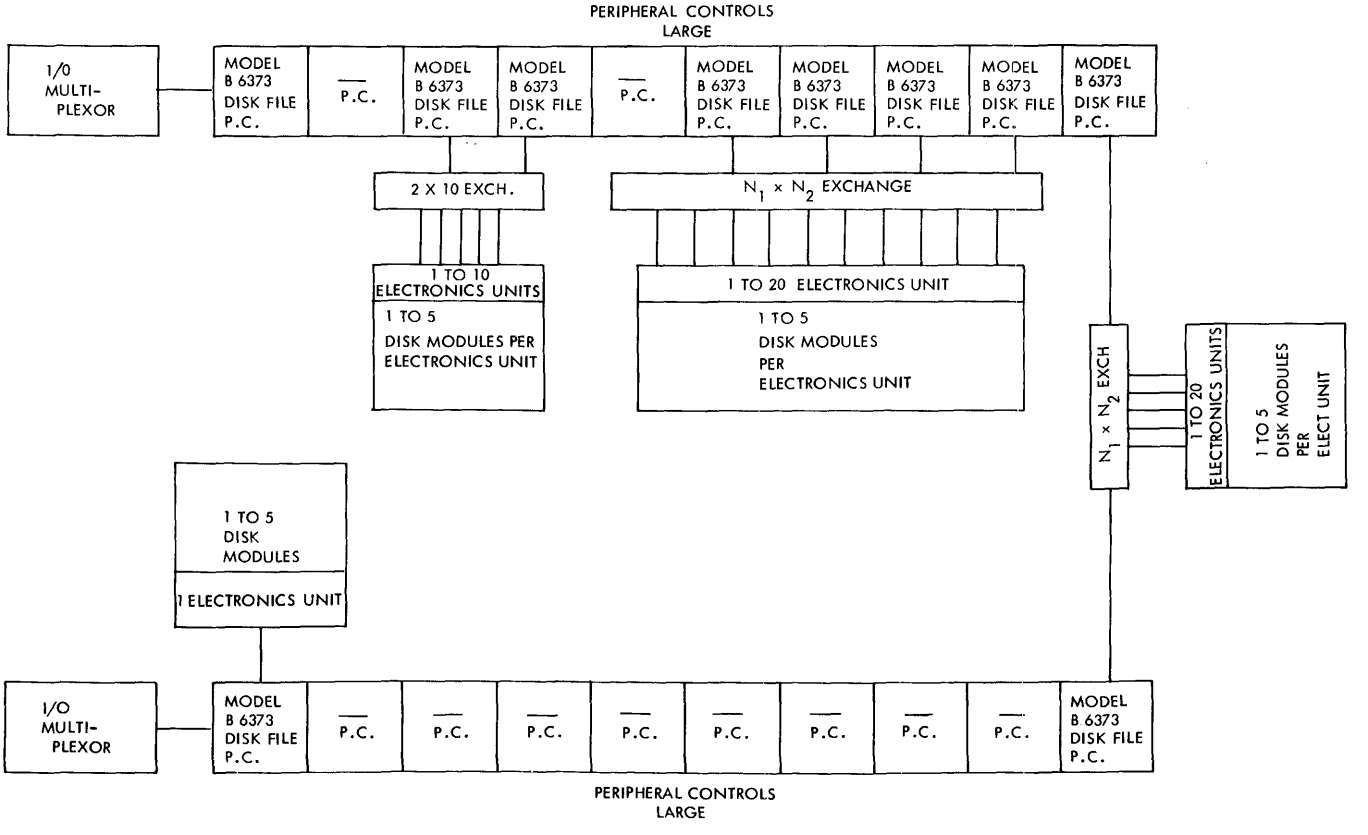


Figure 1-6. Possible Disk File Subsystem

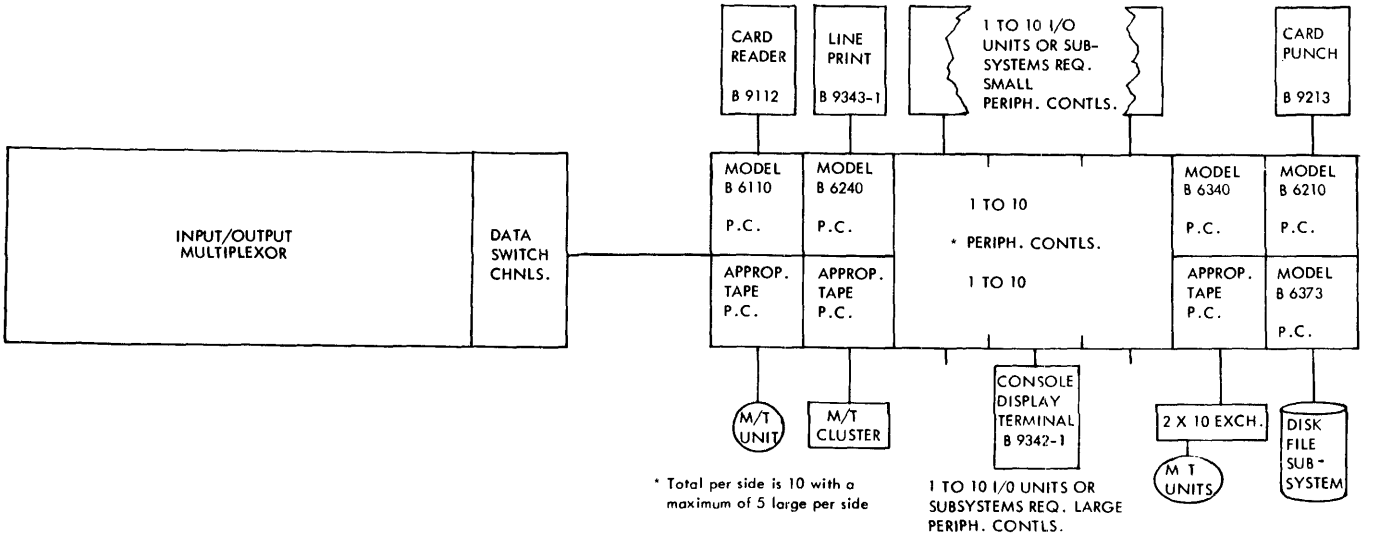


Figure 1-7. Input/Output Subsystem

PROCESSOR INITIATED I/O OPERATIONS.

Either processor can initiate an I/O operation on either multiplexor (in a two processor/two-multiplexor configuration) by executing an Initiate I/O command. This command transfers a Unit Number Word and an Area Descriptor to the multiplexor via the scan bus. The multiplexor then fetches the I/O Control Word located at the Area Base Address (in the Area Descriptor) and initiates the peripheral operation. An I/O Finished Interrupt is set after the peripheral operation is completed. The Result Descriptor is returned when either processor executes a Read Result Descriptor command.

PERIPHERAL CONTROLS.

Up to 20 peripheral controls can be used with each I/O multiplexor. The peripheral controls are housed in one or two B 6500 peripheral control cabinets. Each cabinet can accommodate 10 controls, five of which can be large controls and five small controls. The following peripheral controls are available:

a. Magnetic Tape.
b. Card Reader.
c. Card Punch.
d. Line Printer.
e. Paper Tape Reader.
f. Paper Tape Punch.
g. Disk File.
h. Console Monitor and Keyboard.

DATA COMMUNICATIONS PROCESSOR.

Because the B 6500 is designed for continuous multiprocessing, the systems readily accommodate applications and procedures requiring data communications. Realtime operations, remote computing, remote inquiry, and on-line programing become additions to the multiprocessing job mix of the B 6500. The data communications processor is the heart of the data communications network.

The Data Communications Processor (DCP) is a small special purpose computer which contains sufficient registers and logic to perform all basic functions associated with sending and receiving data. Up to four DCP's can be connected to an I/O Multiplexor, with each DCP capable of accommodating from one to 256 communications lines (figure 1-8). In a two Multiplexor system, this provides a B 6500 with the ability to service 2048 data communications lines.

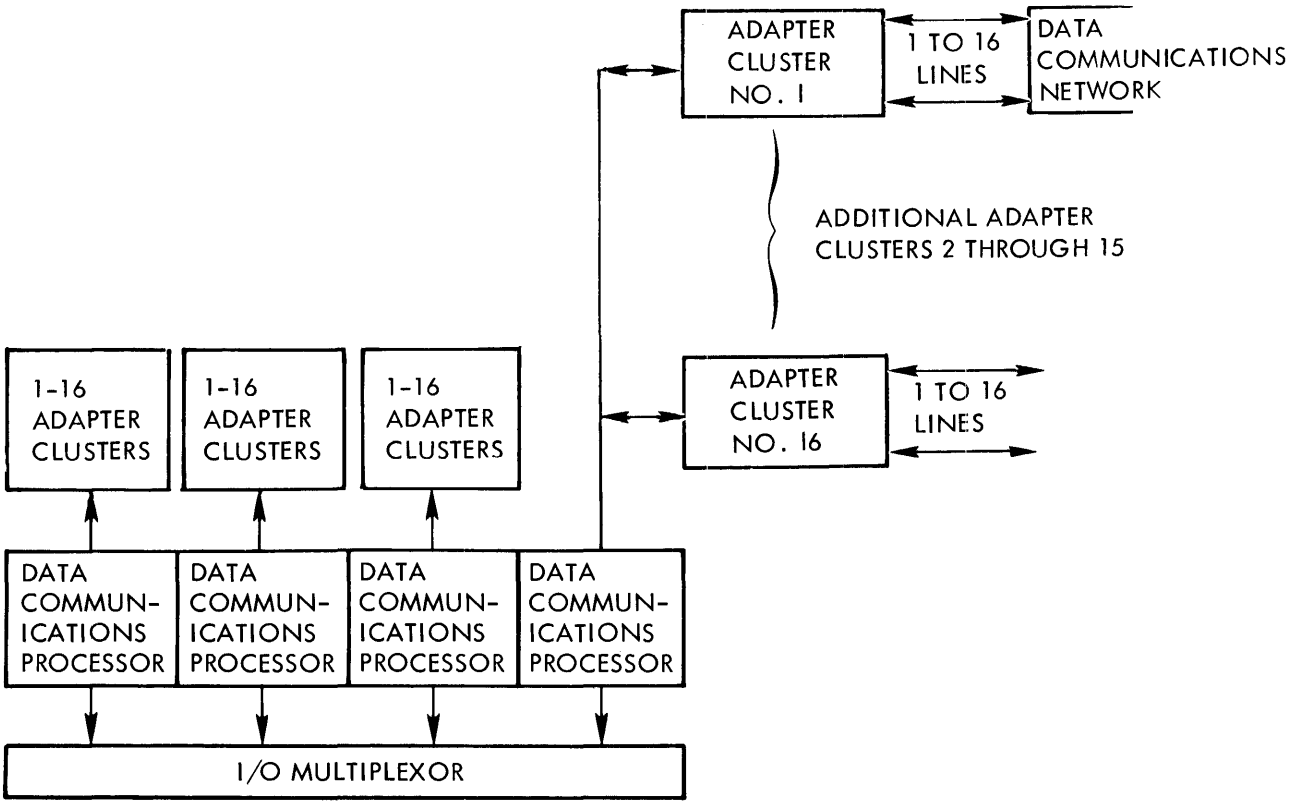


Figure 1-8. Organization of Data Communications Processor Remote Lines

DATA COMMUNICATIONS ADAPTERS.

Each communications channel requires an adapter which provides the logic to interface with a Data Set or to connect directly to a communications line. The following adapters are available:

a. B 6650-1 with the following characteristics:

1) Direct or modem connect.
2) Asynchronous.
3) Up to 600 BPS.
4) Two wire or 100 series modem.
5) Serial by bit transmission.
6) Half-Duplex mode.

b. B 6650-2 with the following characteristics:

1) Direct or modem connect.
2) Asynchronous.
3) Up to 1800 BPS.
4) Two wire or 202 series type Data Set.
5) Serial by bit transmission.
6) Half-Duplex mode.

c. B 6650-3 with the following characteristics:

1) Modem connect.
2) Synchronous.
3) Up to 2400 BPS.
4) 201 series type Data Set.
5) Serial by bit transmission.
6) Half-Duplex mode.

d. B 6650-4 same as B 6650-3 except: up to 4800 BPS.

e. B 6650-5 same as B 6650-3 except: up to 9600 BPS.

f. B 6650-6 Touch-Tone® Telephone Input.

g. B 6650-7 Audio Response.

h. B 6650-8 Automatic Dial Out.

® Registered Service Mark of A.T.T Co.

REAL TIME ADAPTER.

An optional real time adapter may be attached to an I/O multiplexor. Real time devices require custom engineering for interface with the real time adapter and the software.

SECTION 2
DATA REPRESENTATION

GENERAL.

Several data representations are used in the B 6500 Information Processing Systems for word and character oriented data. Each word contains 48 information bits, three tag bits and one parity bit (figure 2-1). The data field may be a 48 bit single-precision operand, or a sequence of characters in 8-bit, 6-bit or 4-bit format. The tag bits in positions 50 through 48 are control bits which identify descriptors, provide memory protection, etc. The tag bits are inaccessible to normal state (user) programs. The parity bit in position 51 assures correct information transfer between the processor and main memory or from the scratch pad to main memory.

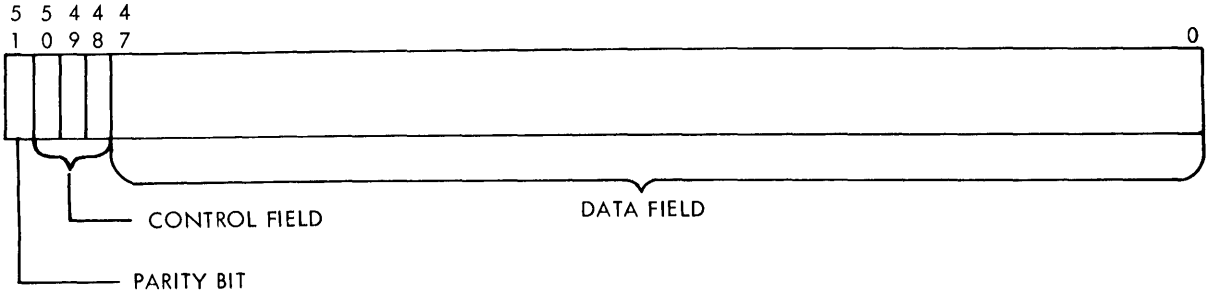


Figure 2-1. Basic Word Structure

INTERNAL CHARACTER CODES.

Extended Binary Coded Decimal Interchange Code (EBCDIC) is the primary internal character code of the B 6500. EBCDIC is an 8-bit alphanumeric code containing 4 zone and 4 numeric bits. The American Standard Code for Information Interchange (ASCII) is the primary data communication code. In addition, the Burroughs Common Language Code (BCL) provides interface compatibility with peripheral units. The pack operator allows greater packing density of numeric information by storing 4-bit digits in both the numeric and zone bit positions BCL and EBCDIC codes (figure 2-9).

NUMBER BASES.

Because the arithmetic operators are implemented in octal (base 8) and data display in registers and certain printed forms is Hexadecimal (base 16), an understanding of both octal and hexadecimal numbering systems is useful. A brief discussion of binary and decimal numbering systems is also included.

The decimal system is based on the ten digits, 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, and upon the powers of ten. Similarly, the binary system is based upon the two digits, 0 and 1, and the powers of two. Two raised to the third power (2^3) is 8, the base of the octal system. Likewise, 2 raised to the fourth power (2^4) is 16, the base of the Hexadecimal system. The decimal range for each number system is shown in figure 2-2.

DECIMAL	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
BINARY	0 1
OCTAL	0 1 2 3 4 5 6 7
DECIMAL	0 1 2 3 4 5 6 7 8 9
HEXADECIMAL	0 1 2 3 4 5 6 7 8 9 A B C D E F

Figure 2-2. Number Base Graphic Characters

The digits 0 through 9 and the alphabetic characters A through F are used to cover the 16 character requirement for the hexadecimal numbering system. The letter A is assigned a value of 10, B equals 11 etc., to F which equals 15.

HEXADECIMAL AND OCTAL NOTATION.

Since binary words are cumbersome to display, the more efficient methods of Hexadecimal and Octal notation are employed. The hexadecimal representation of a binary word is obtained by dividing the bits into groups of four with each group assigned a successive power of 16. A binary to octal conversion is obtained by dividing the bits into groups of three and assigning successive powers of 8 to each group (figure 2-3).

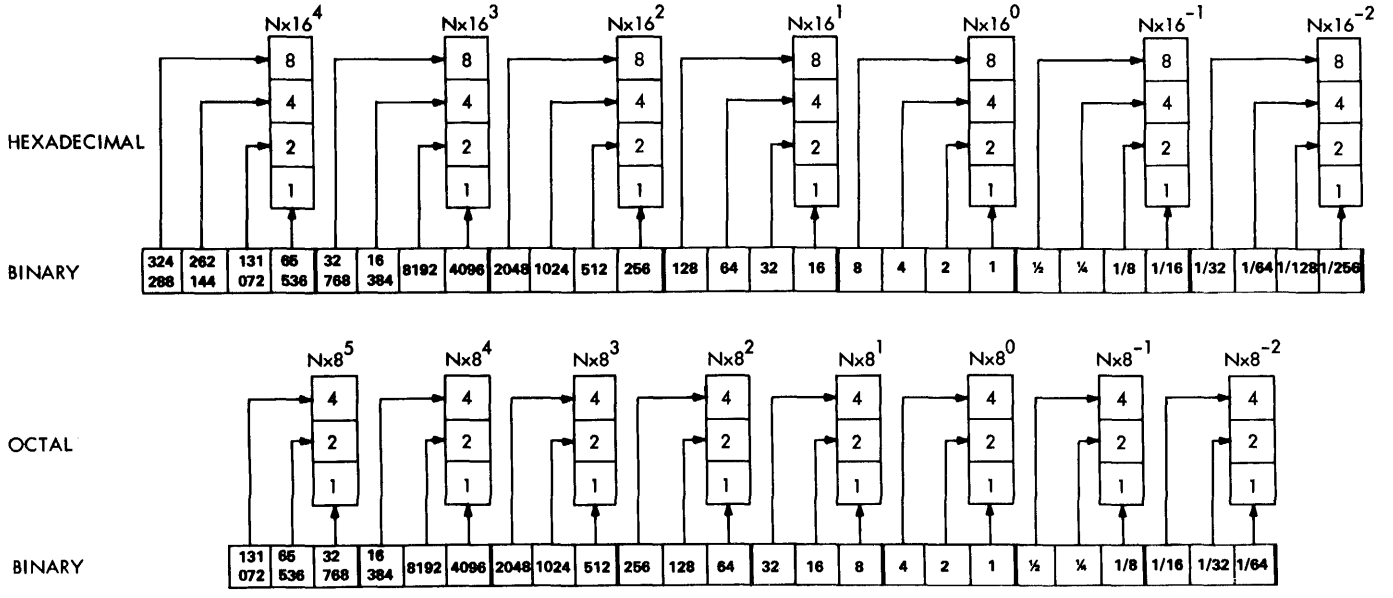


Figure 2-3. Binary to Hexadecimal and Octal Conversion

The relationship between octal, decimal and hexadecimal is shown in figure 2-4 using the decimal number 1013_{10} (equivalent to 1765_8 and $3F5_{16}$ where the subscript 8, 10, or 16 indicates the base).

$$
\begin{aligned}
1765_8 \equiv\ & 1\times8^3 + 7\times8^2 + 6\times8^1 + 5\times8^0 = \\
& 1\times512 + 7\times64 + 6\times8 + 5\times1 = \\
& 512 + 448 + 48 + 5 = 1013_{10}
\end{aligned}
$$

$$
\begin{aligned}
1013_{10} \equiv\ & 1\times10^3 + 0\times10^2 + 1\times10^1 + 3\times1^0 = \\
& 1\times1000 + 0\times100 + 1\times10 + 3\times1 = \\
& 1000 + 0 + 10 + 3 = 1013_{10}
\end{aligned}
$$

$$
\begin{aligned}
3F5_{16} \equiv\ & 0\times16^3 + 3\times16^2 + F\times16^1 + 5\times16^0 = \\
& 0\times4096 + 3\times256 + F\times16 + 5\times1 = \\
& 0 + 768 + 240 + 5 = 1013_{10}
\end{aligned}
$$

Figure 2-4. Relationship of Octal, Decimal and Hexadecimal Numbers

NUMBER CONVERSION.

CODED TO DECIMAL CONVERSION.

The conversion to base ten of the integral value of a number whose base is other than ten may be accomplished by the addition of computed place positions as shown in figure 2-4. Another method of conversion is by repeated multiplications and additions as shown in figure 2-5. The multiplier is the decimal value of the desired number base when using this system.

DECIMAL TO CODED.

The conversion of a Decimal number to any other base is accomplished by repeatedly dividing the number by the desired number base and retaining the successive remainders (figure 2-6).

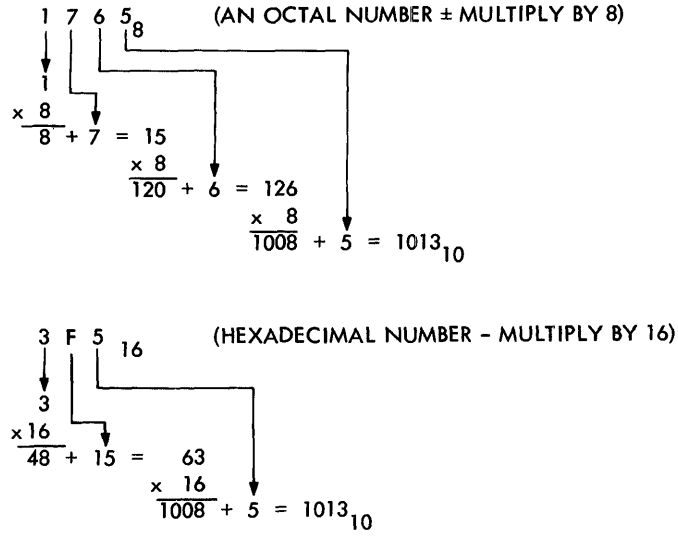


Figure 2-5. Hexadecimal and Octal To Decimal

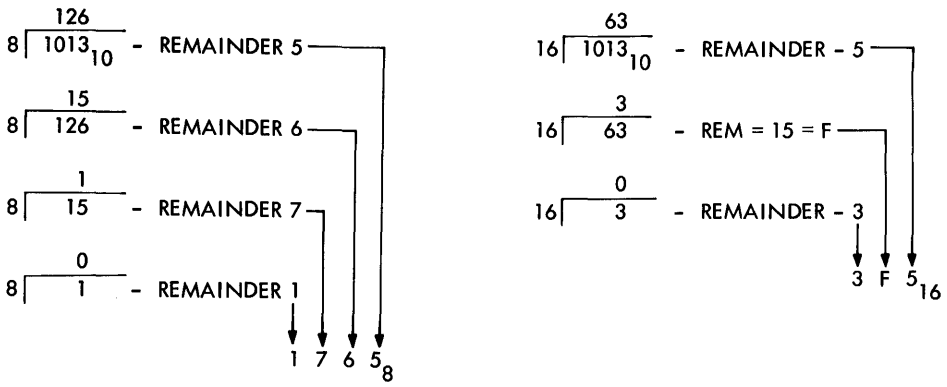


Figure 2-6. Decimal 1013_{10} To Hexadecimal and Octal

DECIMAL AND HEXADECIMAL TABLE CONVERSION.
(Use figure 2-7 for following computations.)

Hexadecimal to Decimal. Find the decimal value for each hexadecimal digit according to its position. Add these to obtain the decimal equivalent.

Decimal to Hexadecimal. Find the next lower decimal number and its Hexadecimal equivalent. Subtract and use difference to find the next decimal value and hexadecimal equivalent until the complete number is developed.

6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

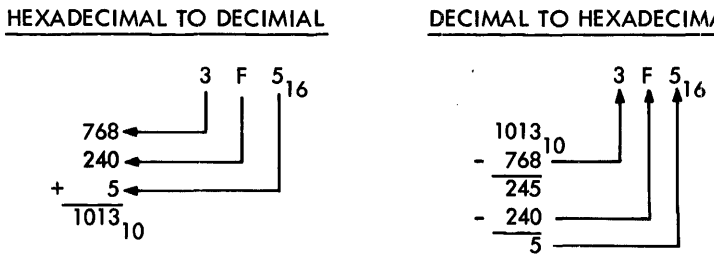


Figure 2-7. HEX and DEC Table Conversion

ORDER OF MAGNITUDE.

The order of number magnitude in relation to the 39 bit mantissa, decimal numbers and powers of base 16, 8, and 2 are shown in figure 2-8.

REGISTER BIT SET	NUMERIC EQUIVALENT	HEX.	OCTAL	BINARY
0	1 1.0	16^0	8^0	2^0
1	2 0.5			
2	4 0.25			
3	8 0.125		8^1	2^3
4	16 0.062 5	16^1		
5	32 0.031 25			
6	64 0.015 625		8^2	2^6
7	128 0.007 812 5			
8	256 0.003 906 25	16^2		
9	512 0.001 953 125		8^3	2^9
10	1 024 0.000 976 562 5			
11	2 048 0.000 488 281 25			
12	4 096 0.000 244 140 625	16^3	8^4	2^{12}
13	8 192			
14	16 384			
15	32 768		8^5	2^{15}
16	65 536	16^4		
17	131 072			
18	262 144		8^6	2^{18}
19	524 288			
20	1 048 576	16^5		
21	2 097 152		8^7	2^{21}
22	4 194 304			
23	8 388 608			
24	16 777 216	16^6	8^8	2^{24}
25	33 554 432			
26	67 108 864			
27	134 217 728		8^9	2^{27}
28	268 435 456	16^7		
29	536 870 912			
30	1 073 741 824		8^{10}	2^{30}
31	2 147 483 648			
32	4 294 967 296	16^8		
33	8 589 934 592		8^{11}	2^{33}
34	17 179 869 184			
35	34 359 738 368			
36	68 719 476 736	16^9	8^{12}	2^{36}
37	137 438 953 472			
38	274 877 906 944			
39	545 755 813 888		8^{13}	2^{39}

Figure 2-8. Order of Magnitude Table

DATA TYPES AND PHYSICAL LAYOUT.

CHARACTER TYPE.

Character representation may be 8-bit bytes, 6-bit characters, or 4-bit digits. The 8-bit EBCDIC (Extended Binary Coded Decimal Interchange Code) is the primary B 6500 code. When 8 or 6-bit numeric characters are used, the sign of the number is in the zone bits of the least significant character. For 4-bit digits, the sign is the most significant digit of the number. The number (-4259) is represented as 8, 6, & 4-bit characters in figure 2-9.

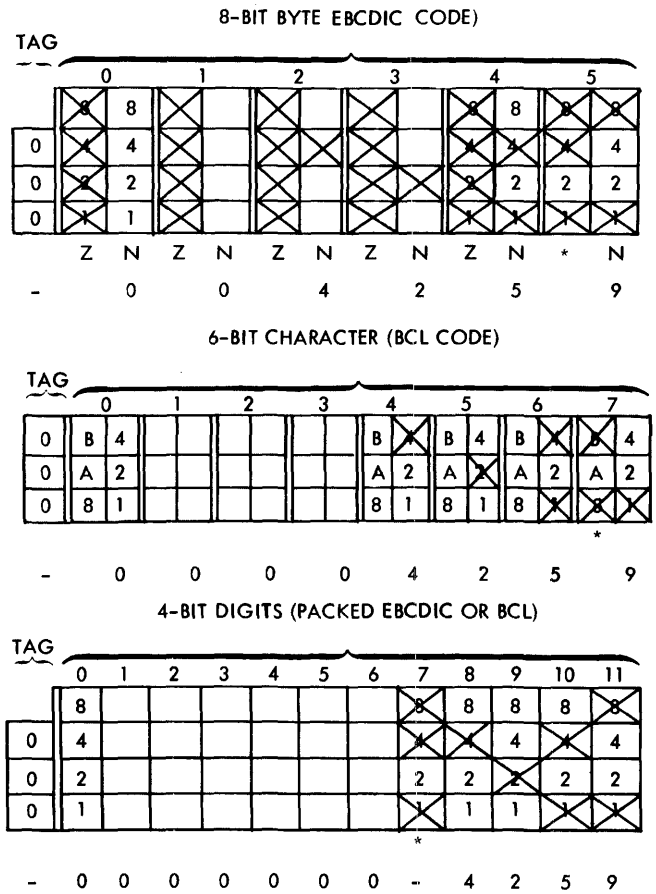


Figure 2-9. (-4259) in 8, 6, and 4-Bit Code

Table 2-2

Negative Sign Bit Configuration

Size	Sign Location	Negative	Positive
8-Bit	Zone, least significant char.	1101	Any bit configuration other than the negative combinations.
6-Bit	Zone, least significant char.	10	
4-Bit	Most significant digit	1101	

OPERANDS.

Operands may be used to represent either numeric or logical information in the B 6500 system. An operand may be single or double-precision. The tag bits of a memory word (bits 50, 49, 48) when zero, denotes a single-precision operand, and when two (bit 49 set), a double-precision operand. The structure of a single-precision operand is illustrated in figure 2-10 in a hexadecimal register format. Note that since the exponent is an octal scale factor, the single-precision operand is also shown in octal for reference. Figure 2-11 illustrates the double-precision operand in hexadecimal register format.

An integer is a single-precision operand with an exponent of zero. The maximum value of an integer is $+7777777777777_8$, 549755813887_{10} or $7FFFFFFFFF_{16}$.

As an example, the decimal number 12 (14_8, C_{16},) might be represented in any of the following forms:

a. In OCTAL format:

0000000000000014 (INTEGER)

1010000000000140 }
1020000000001400 } (Floating point, or REAL)
1131400000000000 }

b. In HEXADECIMAL format:

000000000000C (INTEGER)

208000000060 }
210000000300 } (Floating Point)
2A1800000000 }

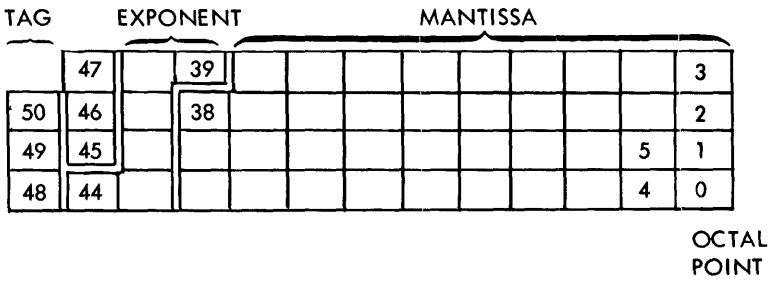


Figure 2-10-A. Single Precision Operand

[50:3] Tag field = 0 for Single Precision Operand.
[47:1] Unused.
[46:1] Sign of operand = 1 for negative.
[45:1] Sign of exponent = 1 for negative.
[44:6] Exponent.

The exponent is a binary number which with its sign, is an octal scale factor for the mantissa. The exponent is used for automatic scaling of operands when performing arithmetic, comparison and integer operations. The range of the exponent is from +63 to -63 for single-precision operands.

MANTISSA FIELD.
The mantissa is the significant part of the operand. The magnitude of the operand is obtained by multiplying the value contained in the mantissa by eight raised to the value of the exponent sign and exponent as follows:

$V = \pm M \times 8 \pm E$

V = Value of number

$\pm$M= Mantissa with sign

$\pm$E= Exponent with sign

The range of numbers that can be expressed in single-precision is $(8^{13} - 1) \times 8^{+63}$ to 1×8^{-51} and zero.

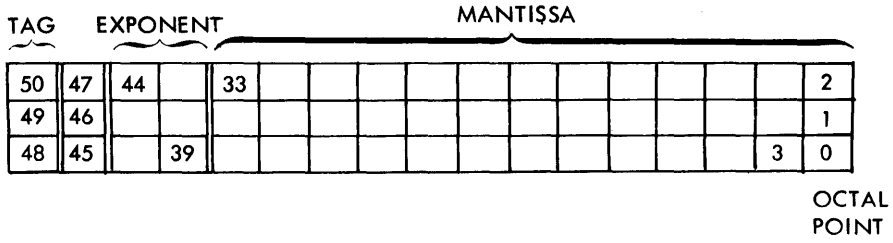


Figure 2-10-B. Single Precision Operand

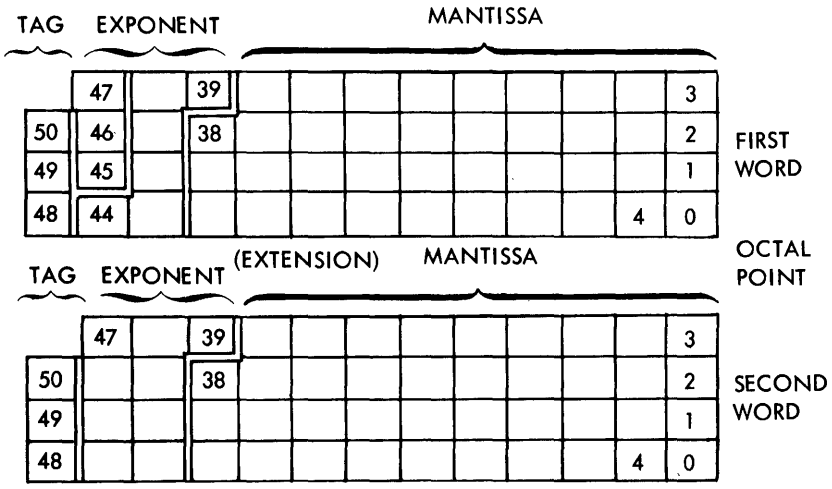


Figure 2-11. Double-Precision Operand

[50:3] Tag Field = 2 for double-precision operands.

The first word of the operand is identical to the single-precision operand except for bit position 49, which indicates that this is one of a pair of words.

The fractional part of the mantissa is contained in the mantissa extension field of the word.

The 15-bit exponent of a double-precision operand is formed by the concatenation of the exponent extension with the exponent. The exponent extension is more significant than the exponent.

LOGICAL OPERANDS.

Logical operands have one of two values: true (on) or false (off). Logical values are the result of Boolean operations or relational operations. Relational operators generate a logical value as the result of an algebraic comparison of two arithmetic expressions. Bit number zero (0) represents the logical value. Relational operators set bit number zero, and conditional operators use bit zero for the decision. Logical (Boolean) operators consider each bit, from 47 to bit 0, as an individual logical value, operating on the whole word. A logical value is expressed in the following form (figure 2-12).

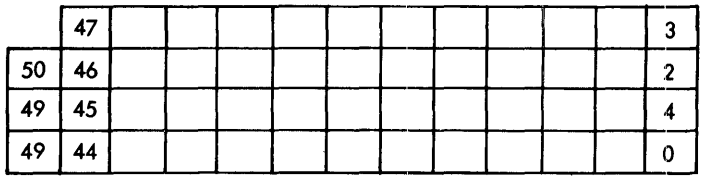


Figure 2-12. Logical Operand

[50:3] = 0 TAG = S.P. OPERAND
[0:1] = 1 TRUE, 0 FALSE

OPERATORS.

The operators used in the B 6500 systems are divided into three major categories; Primary, Variant and Edit. Details regarding the format and function of these operators are found in Sections 6, 7, 8, and 9.

SECTION 3

STACK AND POLISH NOTATION

THE STACK.

GENERAL.

The stack is an area of memory assigned to a job to provide storage for basic program and data references for the job. It also provides for temporary storage of data and job history. When a job is activated, four high-speed registers (A, X, B, and Y) are linked to the job's stack (figure 3-1). This linkage is established by the stack-pointer register (S), which contains the memory address of the last word placed in the stack. The four top-of-stack registers (A, X, B and Y) extend the stack to provide quick access for data manipulation.

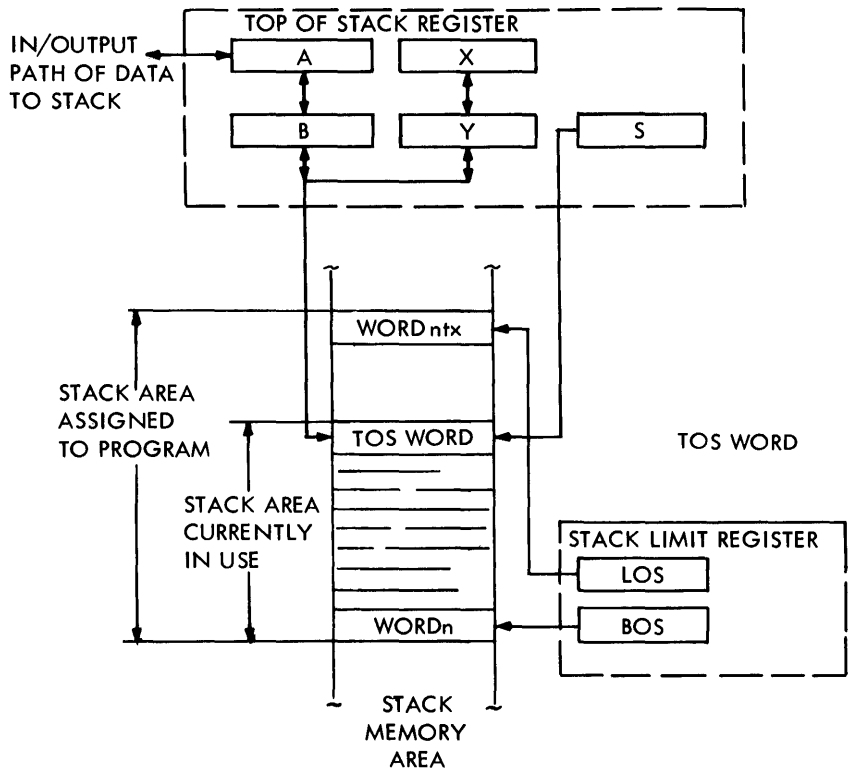


Figure 3-1. Top of Stack and Stack Bounds Registers

Data are brought into the stack through the top-of-stack registers in such a manner that the last operand placed into the stack is the first to be extracted. Total capacity of the top-of-stack register is two operands. Loading a third operand into the top-of-stack registers causes the first operand to be pushed from the top-of-stack registers into the stack. The stack-pointer register (S) is incremented by one as each word is placed into the stack and is decremented by one as each word is withdrawn from the stack and placed in the Top-of-Stack registers. As a result, the S register continually points to the last word placed into the job's stack.

BASE AND LIMIT OF STACK.

A job's stack is bound, for memory protection, by two registers, the Base-of-Stack register (BOSR) and the Limit-of-Stack register (LOSR). The contents of BOSR defines the base of the stack, and the LOSR defines the upper limit of the stack. The job is interrupted if the S register is set to the value contained in either LOSR or BOSR.

BI-DIRECTIONAL DATA FLOW IN THE STACK

The contents of the top-of-stack registers are maintained automatically by the processor to meet the requirements of the current operator. If the current operator requires data transfer into the stack, the top-of-stack registers receive the incoming data, and the surplus contents of the top-of-stack registers, if any, are pushed into the stack. Words are brought out of the stack into the top-of-stack registers for operators which require the presence of data in the top-of-stack registers, but do not explicitly move data into the stack.

DOUBLE-PRECISION STACK OPERATION.

The top-of-stack registers are operand-oriented rather than word-oriented. Calling a double-precision operand into the top-of-stack registers loads two memory words into the top-of-stack registers. The first word is loaded into the A register, where its tag bits are checked. If the value indicates double-precision the second

word is loaded into X. The A and X registers are concatenated, or linked together, to form the double-precision operand. The B and Y registers concatenate when a double-precision operand is moved to the B register. The double-precision operand reverts to single words as it is pushed from the B and Y registers into the stack. The concatenation is repeated when the double-precision operand is returned from the stack into the top-of-stack registers.

DATA ADDRESSING.

The B 6500 Processor provides three methods for addressing data or program code:

a. Data Descriptor (DD)/Segment Descriptor (SD).
b. Indirect Reference Word (IRW).
c. Stuffed Indirect Reference Word (SIRW).

The Data Descriptor (DD) and Segment Descriptor (SD) provide for addressing data or program segments located outside of the job's stack area. The Indirect Reference Word (IRW) and the Stuffed Indirect Reference Word (SIRW) address data located within the job's stack. The IRW and SIRW address components are both relative. The IRW addresses within the immediate environment of the job relative to a display register (described later in Non-local Addressing). The SIRW addresses beyond the immediate environment of the current procedure, and addresses relative to the base of the job's stack. Addressing across stacks is accomplished with an SIRW.

DATA DESCRIPTOR.

In general, the descriptor describes and locates data or program code associated with a given job. The Data Descriptor (DD) is used to fetch data to the stack or store data from the stack into an array located outside the job's stack area. The formats of the Data and Segment Descriptors are illustrated in Section 6. The ADDRESS field of both descriptors is 20 bits in length and contains the absolute address of an array in either system main memory or in the backup disk file as indicated by the Presence bit (P). The referenced data is in main memory when the presence bit is set.

PRESENCE BIT. A Presence Bit interrupt occurs when the job references data with a descriptor which has its P bit and copy bit equal to ZERO. The Master Control Program (MCP) recognizes the Presence Bit Interrupt and transfers data from the disk file to main memory. After the data transfer to main memory is completed, the MCP marks the descriptor present by setting the P-bit to ONE, and places the new main memory address into the ADDRESS field of the descriptor. The interrupted job is then reactivated.

INDEX BIT. A Data Descriptor describes either an entire array of data words, or a particular element within an array of data words. If the descriptor describes the entire array, the Index bit (I-bit) in the descriptor is ZERO, indicating that the descriptor has not yet been indexed. The LENGTH field of the descriptor defines the length of the data array.

INVALID INDEX.

A particular element of an array may be described by indexing an array descriptor. Memory protection is ensured during indexing operations by performing a comparison between the LENGTH field of the descriptor and the index value. An Invalid Index Interrupt results if the index value exceeds the length of the memory area defined by the descriptor, or if the index is less than zero.

VALID INDEX.

If the index value is valid, the LENGTH field of the descriptor is replaced by the index value, and the I-bit in the descriptor is set to ONE to indicate that indexing has taken place. The ADDRESS and INDEX fields are added together to generate the absolute machine address whenever a present, indexed Data Descriptor is used to fetch or store data.

The Double-Precision bit (D) is used to identify the referenced data as single or double-precision and has a direct affect on the indexing operation. The D-bit equal to ONE signifies double-precision and causes the index value to be doubled before indexing.

READ-ONLY BIT. The Read-Only bit (R) specifies that the memory area described by the Data Descriptor is read-only area. An interrupt results when an area is referenced through a descriptor with intentions of storing with the R bit set to ONE.

COPY BIT. The Copy bit (C) identifies a descriptor as a copy of a master descriptor and is related to the presence bit action. The copy-bit links multiple copies of an absent descriptor to the one master descriptor. The copy-bit mechanism is invoked when a copy is made in the stack, of an absent-Data Descriptor. If the Absent-Data Descriptor is the original (master) descriptor, the processor sets the copy bit to ONE and inserts the address of the master descriptor into the ADDRESS field. Thus, multiple copies of absent-data descriptors are all linked back to the master descriptor.

POLISH NOTATION.

GENERAL.

To understand the B 6500 stack, Polish notation must be understood. A problem that exists with most forms of mathematical notation is defining the boundaries of specific terms. This is eliminated with the use of parentheses, brackets, and braces. The expression 5Z + 7/2Z and (5Z + 7)/2Z express different functions of Z, but one could easily be used when the other was intended. However, with a complex equation, it becomes necessary to duplicate the use of the few types of delimiters that exist.

Polish notation is an arithmetical or logical notational system using only operands and operators arranged in sequence or string which eliminates the necessity for defining the boundries of any terms. Figure 3-2 presents a flow chart for conversion to Polish notation.

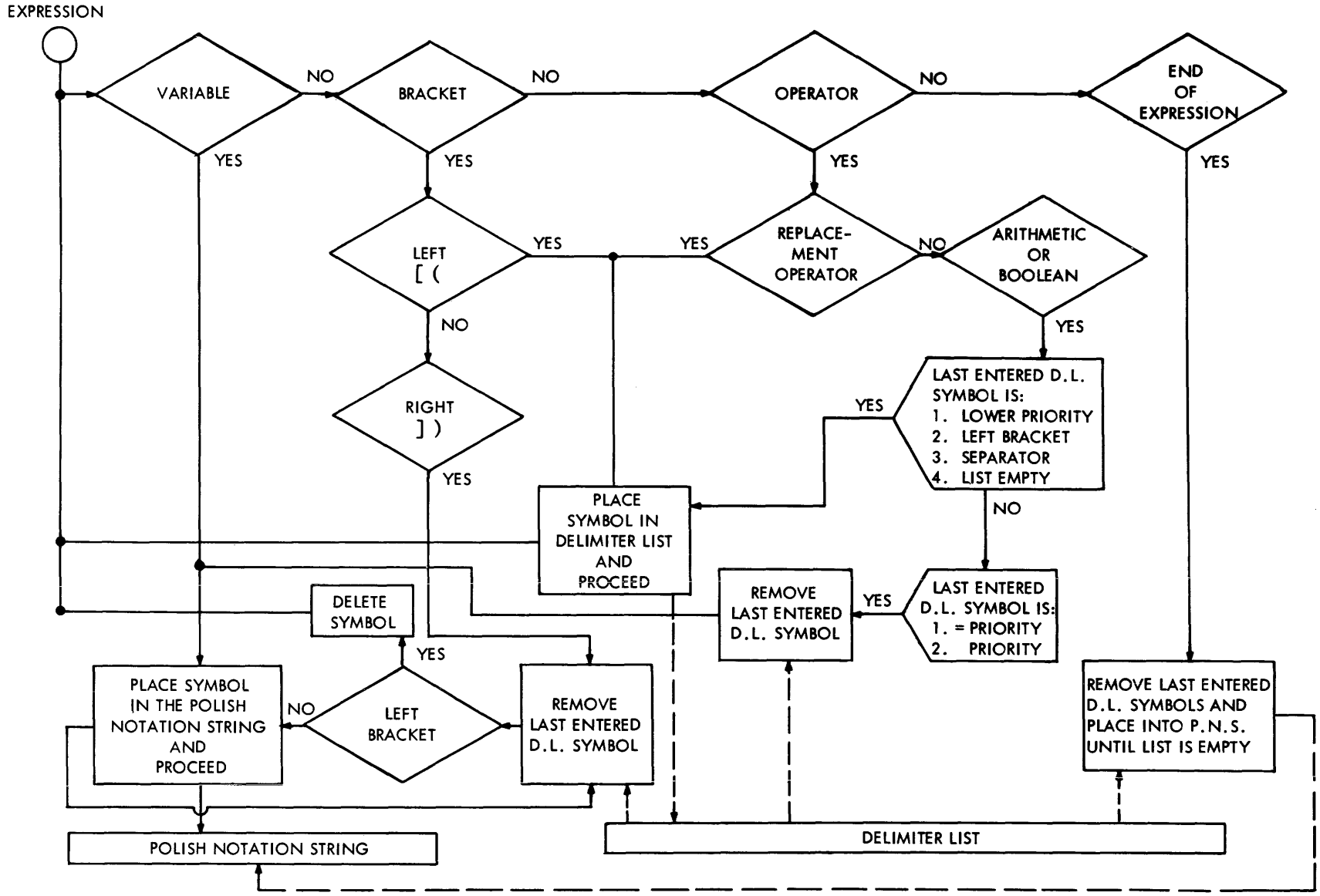


Figure 3-2. Polish Notation Flow Chart

RULES FOR GENERATION OF POLISH STRING.

The source of expression is:

Name	Action
Variable	Place variable in string being built and examine next symbol.
Operator-Separator	Place in delimiter list and examine next symbol.
Arithmetic or Boolean operator and last entered delimiter list symbol was: a. An operator of lower priority. b. A left bracket "[" or parenthesis "(". c. A separator. d. Nothing (delimiter list empty.	Place operator in the delimiter list and examine next source symbol.
Arithmetic or Boolean operator and last entered delimiter list symbol was: an operator of priority equal to or greater than the symbol in the source.	Remove the operator from the delimiter list and place in the string being built. Then compare the next symbol in the delimiter list against the source ex-expression symbol.
A right bracket "]" or parenthesis ")".	Pull from delimiter list until corresponding left bracket or parenthesis.

POLISH STRING.

The essential difference between Polish and conventional notation is that operators are written to the right of the operands instead of between them. For example, the conventional B + C is written B C + in Polish notation: A = 7(B+C) becomes BC + 7 x A =.

Any expression written in Polish notation is called a Polish string. In order to fully understand this concept, the rule for evaluating a Polish string should be known.

RULES FOR EVALUATING A POLISH STRING.

The rule is summarized in a few steps:

a. Scan the string from left to right.

b. Remember the operands and the order in which they occur.

c. When an operator is encountered:

   1) Record the last two operands encountered.

   2) Execute the required operation.

   3) Disregard the two operands.

   4) Consider the result of (2) as a single operand, the first of the next pair to be operated upon.

Following this rule, the Polish string B C + 7 x A := results in A assuming the value 7 (B + C) (table 3-1).

Table 3-1

Evaluation of Polish String BC + 7 x A :=

Step	Symbol Being Examined	Symbol Type	Operands Being Remembered and Their Order of Occurence (1 or 2) Before Operation	Operation Taking Place	Results of Operation
a.	B	Operand			
b.	C	Operand	1 B		
c.	+	Add Operator	2 C 1 B	B + C	(B + C)
d.	7	Operand	1(B + C)		
e.	x	Multiply Operator	2 7 1(B + C)	7 x (B + C)	7 x (B+C)
f.	A	Name	1 7(B + C)		
g.	:=	Replace Operator	2A 1 7(B + C)	A :=7(B + C)	A=7(B + C)

SIMPLE STACK OPERATION.

All program information must be in the system before it can be used. Input areas are set aside for information entering the system and output areas are set aside for information exiting the system, array and table areas are allocated to store certain types of data. Thus data is stored in several different areas: The input/output areas, data tables (arrays), and the stack. Since all work is done in the arithmetic registers, all information or data is transferred to the arithmetic registers and the stack.

At this point, an ALGOL assignment statement and the Polish notation equivalent will be related to the stack concept of operation. The example is Z:=Y + 2x(W+V), where := means "is replaced by." In terms of a computer program, this assignment statement indicates that the value resulting from the evaluation of the arithmetic expression is to be stored in the location representing the variable Z.

When Z:=Y + 2x(W+V) is translated to Polish notation, the result is Y2WV+x+Z:=. Each element of the example expression causes a certain type of syllable to be included in the machine language program when the source problem is compiled. The following is a detailed description of each element of the example, the type of syllable compiled, and the resulting operation (see figure 3-3 and table 3-2).

In the example statement, Y is added to a quantity; therefore, Y is brought to the stack as an operand. This is accomplished with a Value Call (VALC) syllable that references Y. The value 2 is then brought to the stack, with an eight bit literal syllable (LT8). Since W and V are to be added, the respective variables are brought to the stack with Value Call syllables. The ADD operator adds the two top operands and places the sum in the top of stack. This example assumes single-precision operands for simplicity not requiring use of the "X" and "Y" registers which are used in double-precision operations.

The multiply operator is the next symbol encountered in the Polish string and when executed, places the product "2x(V+W)" in the top of the stack. The next symbol, ADD, when executed leaves the final result "7+2x(W+V)" in the top of the stack.

Since Z is to be the recipient of a value, the address of Z must be placed into the stack just prior to the store command. This is accomplished with a name call syllable which places an Indirect Reference Word (IRW) in the stack. The IRW contains the address of Z in the form of an "address couple" that references the memory location reserved in the stack for the variable Z.

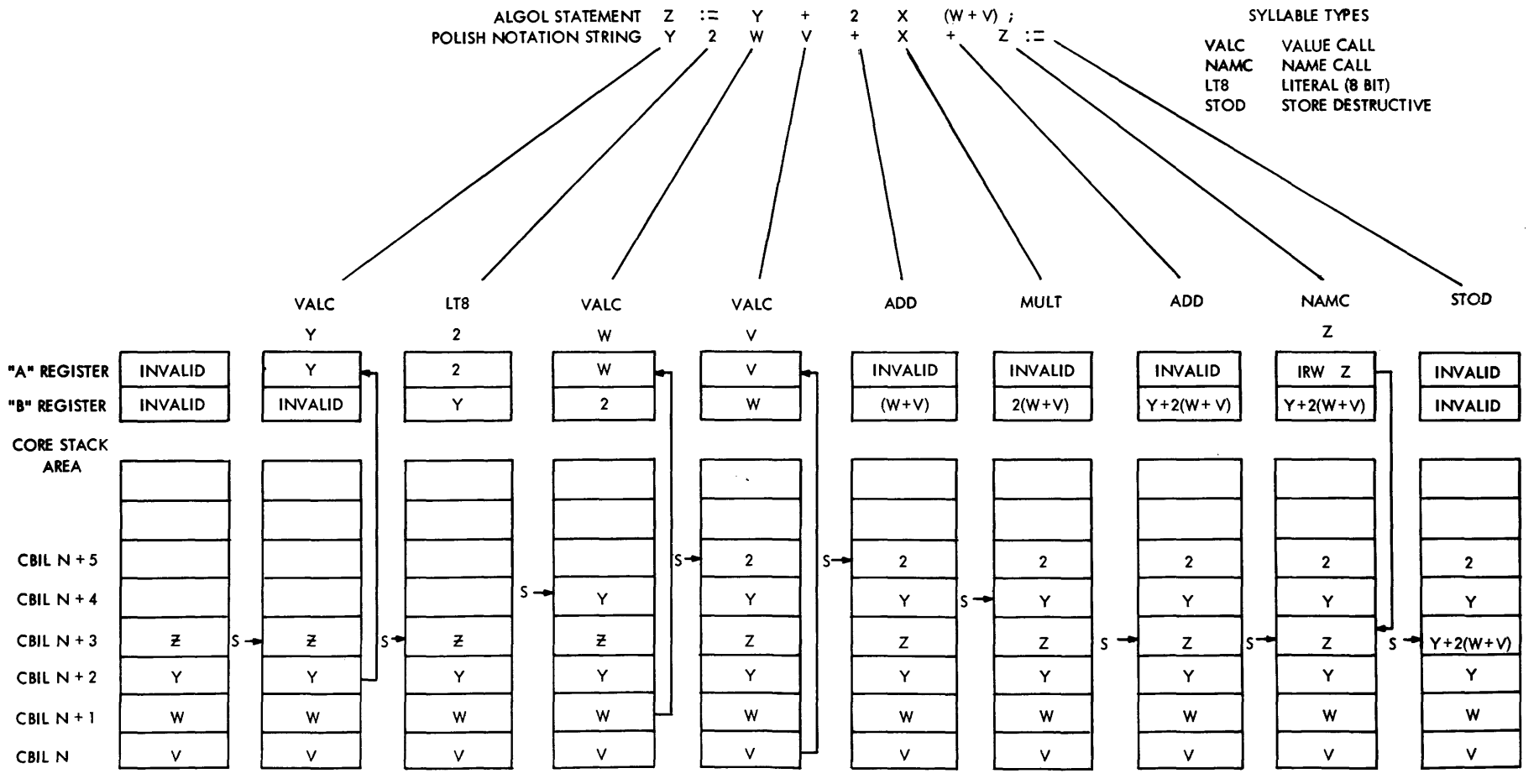


Figure 3-3. Stack Operation

Table 3-2
Description of Stack Operation

Execution Sequence	Polish Notation Element	Syllable Type Compiled	Function of Syllable During Running of the Program
0			Stack location of program variables illustrated.
1	Y	Value call for Y.	Place the value of Y in the top of the stack.
2	2	Literal 2.	Place a 2 in the top of the stack.
3	W	Value call for W.	Place the value of W in the top of the stack.
4	V	Value call for V.	Place the value of V in the top of the stack.
5	+	Operator Add.	Add the two top words in the stack and place the result in B register as the top of the stack.
6	x	Operator Multiply.	Multiply the two top of the stack operands. The product is left in the B register as the top of the stack. (The A register contains the multiplier and the B register the multiplicand.)

Table 3-2 (cont)
Description of Stack Operation

Execution Sequence	Polish Notation Element	Syllable Type Compiled	Function of Syllable During Running of the Program
7	+	Operator Add.	Add the two top words in the stack and leave the result in the B register as the top of the stack.
8	Z	Name call on Z.	Build an Indirect Reference Word that contains the address of Z and place it in the top of the stack.
9	:=	Operator Store Destructive.	Store an item into memory. The address to store into is indicated by an Indirect Reference Word or a Data Descriptor. The address can be above or below the item stored.

The Store syllable completes the execution of the statement Z:=Y + 2x(W+V). The store operation examines the two top of stack operands looking for an IRW or Data Descriptor. In this example the IRW addresses the location where the computed value of Z is to be stored. The stack is empty at the completion of this statement.

PROGRAM STRUCTURE IN MEMORY.

When a problem is expressed in a source language, portions of the source language fall into one of two categories. One describes the constants and variables that will be used in the program, and the other the computations that will be executed. When the source program is compiled, variables are assigned locations within the stack whereas the constants are embeded within the code stream that forms the computational part. A program residing in memory occupies separately allocated areas. Separately allocated means that each part of the program may reside anywhere in memory, and the actual address is determined by the MCP. In particular, the various areas are not assigned to contiguous memory areas. Registers within the processor indicate the bases of the various areas during the execution of a program.

MEMORY AREA ALLOCATION. The separately allocated areas of a program are:

a. Program Segments -- Sequences of instructions (syllables) that are performed by the processor in executing the program. Note that there is a distinction between program segments and data areas. The program segments contain no data, and are not modified by the processor as it executes the program.

b. Segment Dictionary -- This is a table containing one word for each program segment. This word tells whether the program segment is in main memory or on the disk, and gives the corresponding main memory or disk address of the program segment.

c. Stack Area -- This is the pushdown stack storage, which contains all the variables associated with the program, including control words which indicate the dynamic status of the job as it is being executed.

d. MCP Stacks and Segment Dictionary -- This area contains variables pertinent to the MCP. It also contains the MCP segment dictionary entries. Figure 3-4 shows in basic form the separately allocated areas of a program.

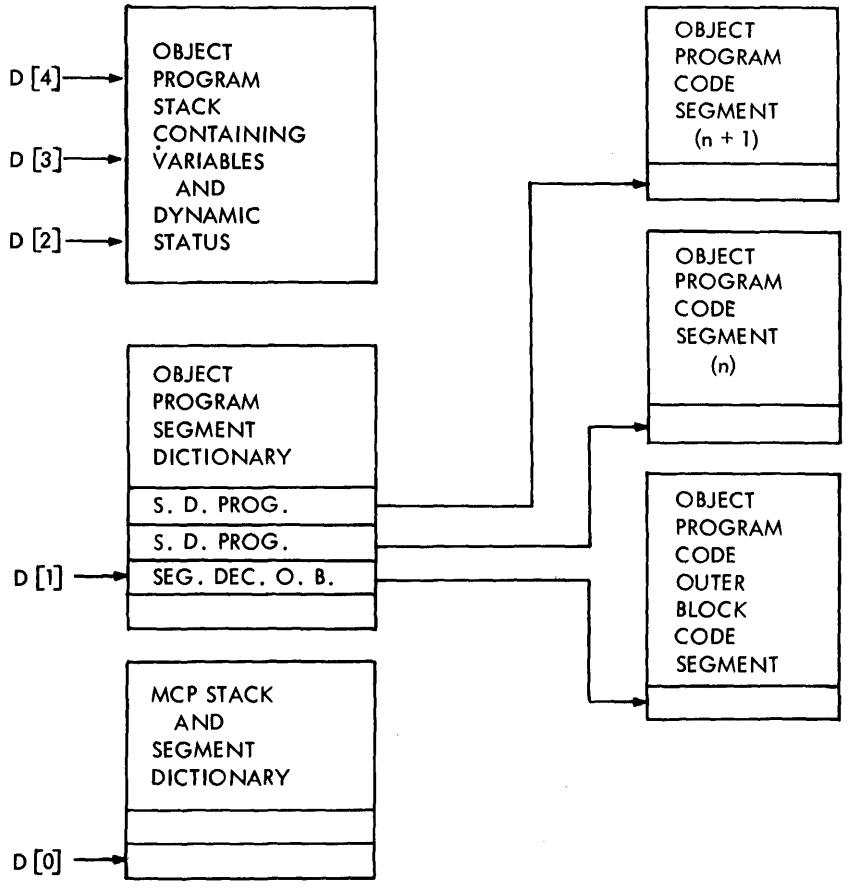


Figure 3-4. Object Program in memory

STACK-HISTORY AND ADDRESSING-ENVIRONMENT LISTS.
One very important aspect of the B 6500 is the retention of the dynamic history for the program being processed. Two lists of program history are maintained in the B 6500 stack, the stack-history list and the addressing-environment list. The stack-history list is dynamic, varying as the job proceeds along different program paths with changing sets of data. Both lists are generated and maintained by B 6500 hardware.

MARK STACK CONTROL WORD LINKAGE. The stack history is a list of Mark Stack Control Words (MSCW), linked together by their DF fields (figure 3-5). A MSCW is inserted into the stack as a procedure is entered, and is removed as that procedure is exited. Therefore, the stack history list grows and contracts with the procedural depth of the program. Mark Stack Control Words identify the portion of the stack related to each procedure. When the procedure is entered, its parameters and local variables are entered in the stack following the MSCW. When executing the procedure, its parameters and local variables are referenced by addressing relative to the MSCW.

STACK DELETION. Each MSCW is linked to the prior MSCW through the contents of its DF field to identify the point in the stack where the prior procedure began. When a procedure is exited, its portion of the stack is discarded. This action is achieved by setting the stack-pointer register (S) to address the memory cell preceding the most recent MSCW (figure 3-6). This top-most MSCW, addressed by another register (F), is deleted from the stack-history list by changing F to address the prior MSCW, placing it at the head of the stack history.

This is an efficient and convenient means of subroutine entry and exit.

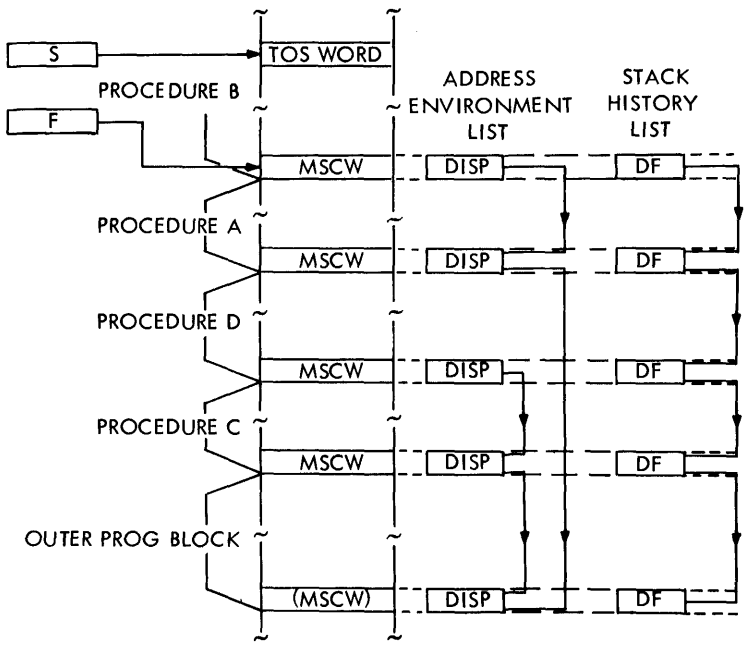


Figure 3-5. Stack History and Addressing Environment List

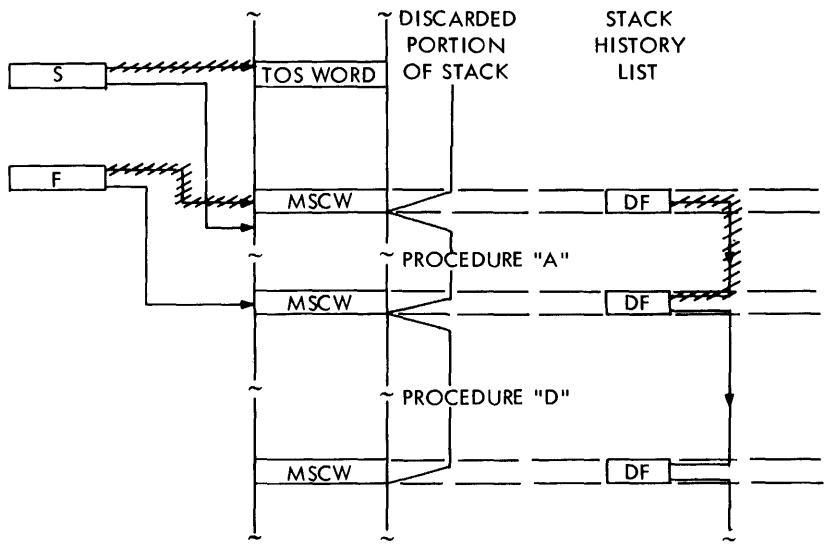


Figure 3-6. Stack Cut-Back Operation on Procedure Exit

RELATIVE-ADDRESSING. Analyzing the structure of an ALGOL program results in a better understanding of the relative-addressing procedures used in the B 6500 stack. The addressing environment of an ALGOL procedure is established when the program is structured by the programmer, and is referred to as the lexicographical ordering of the procedural blocks (figure 3-8). At compile time, the lex-

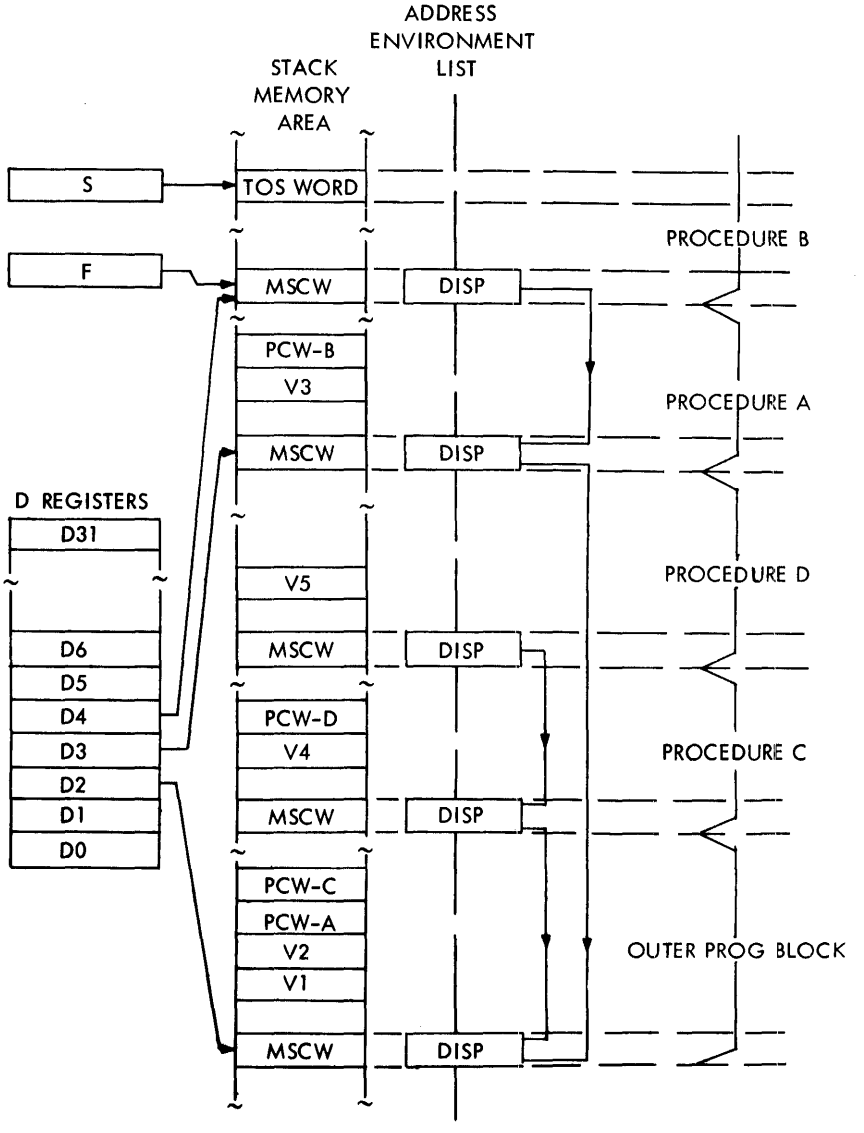


Figure 3-7. D Registers Indicating Current Addressing Environment

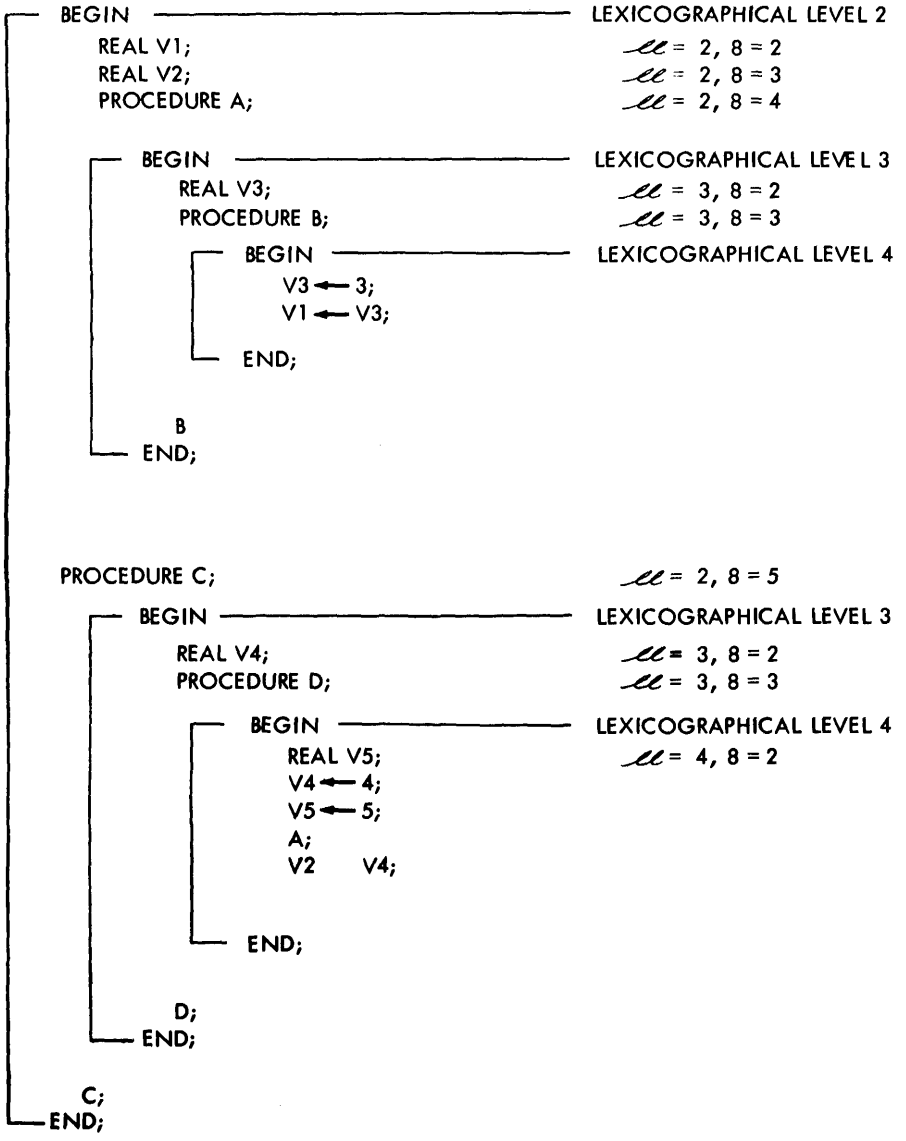


Figure 3-8. ALGOL Program With Lexicographical Structure Indicated

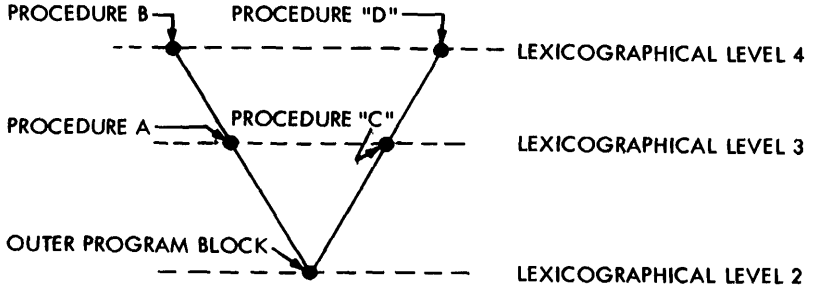


Figure 3-9. Addressing Environment Tree of ALGOL Program

icographical ordering is used to form address couples. An address couple consists of two items:

a. The addressing level (ℓℓ) of the variable,

b. An index value (S) used to locate the specific variable within its addressing level.

The lexicographical ordering of the program remains static as the program is executed, thereby allowing variables to be referenced via address couples as the program is executed.

Base of Addressing-Level Segment.

The B 6500 processor contains an array of D Registers (D0 through D31) which address the base of each addressing-level segment (figure 3-7). The local variables of all procedures are addressed relative to the D registers.

Absolute Address Conversion.

The address couple is converted into an absolute memory address when the variable is referenced. The addressing level portion of the address couple selects the D Register which contains the absolute memory address of the MSCW for the environment (addressing-level) in which the variable is located. The index value of the address couple is added to the contents of the D Register to generate the absolute memory address.

Multiple Variables With Common Address Couples.

The address couples assigned to the variables in a program are not unique. This is true because of the ALGOL scope-of-definition rules, which imply that two variables may have identical address couples if there is no procedure within which both of the variables can be addressed. This addressing system works because, whereas two variables may have the same address couples, there is never any doubt as to which variable is being referenced within any particular procedure.

Address Environment Defined.

There is a unique MSCW which each D Register must address during the execution of any particular procedure. The D Registers must be changed, upon procedure entry or exit, to address the correct MSCWs. The list of MSCWs which the D registers address is the addressing environment of the procedure.

Mark Stack Control Word Linkage.

The addressing environment of the program is maintained automatically by linking the MSCWs together in accordance with the lexicographical structure of the program. This linkage is the Stack Number (Stack No.) and Displacement (DISP) fields of the MSCW, and is inserted into the MSCW whenever the procedure is entered. The addressing environment list is formed by linking each Mark Stack Control Word to the MSCW immediately below the declaration for the procedure being entered. This forms a tree-structured list which indicates the addressing environment of each procedure (figure 3-7 and 3-9). This list is used to update the D Registers whenever a procedure entry or exit occurs.

STACK HISTORY SUMMARY. The entry and exit mechanism of the Processor hardware automatically maintains both the stack history and address-environment lists to reflect the current status of the program. Interrupt response is a procedure entry. Therefore, the system is able to respond to, and return from, interrupts conveniently. Upon recognition of an interrupt condition, the processor creates a MSCW, inserts an indirect reference word into the stack to address the interrupt-handling procedure, inserts a literal constant to identify the interrupt condition and a second parameter, and initiates an MCP interrupt-handling procedure. The D Registers are updated upon entry into the interrupt-handling procedure, to display all legitimate variables. Upon return, the D Registers are updated to display variables of the former procedure.

MULTIPLE STACKS AND RE-ENTRANT CODE.
The B 6500 stack mechanism provides a facility to handle several active stacks. These stacks are organized in a tree structure. The trunk of this tree structure is a stack which contains MCP global quantities.

LEVEL DEFINITION. A program is a set of executable instructions, and a job is single execution of a program for a particular set of data. As the MCP is requested to run a job, a level-1 branch of the basic stack is created. This level-1 branch contains the Descriptors the executable code and Read-only Data segments for the program. Emerging from this level-1 branch is a level-2 branch, containing the variables and data for this job. Starting from the job's stack and tracing downward through the tree-structure, one finds first the stack containing the variables and data for the job (at level 2), the program code to be executed (at level 1), and the MCP's stack at the trunk (level 0).

RE-ENTRANCE. A subsequent request to run another execution of an already-running program requires that only a level-2 branch be established. This level-2 stack branch sprouts from the level-1 stack of the already-running program. Thus two jobs which are different executions of the same program have a common node, at level-1, describing the executable code. It is in this way that program code is re-entrant and shared. It comes about simply from the proper tree-structured organization of the various stacks within the machine. All programs within the system are re-entrant, including all user programs as well as the compilers and the MCP itself.

JOB-SPLITTING. The B 6500 stack mechanism also provides the facility for a single job to split itself into two independent jobs. A most common use of this facility occurs when there is a point in a job where two relatively large independent processes must be performed. This splitting could be used to make full use of a multiprocessor configuration, or to reduce elapsed time by multiprogramming the independent processes.

A split of this type establishes a new limb of the tree-structured stack, with the two independent jobs sharing that part of the stack which was created before the split was requested. The process is recursively defined, and can happen repeatedly at any level.

STACK DESCRIPTOR. Stack branches are located by an array of descriptors, the stack vector array (figure 3-10). There is a data descriptor in this array for every stack branch. This data descriptor, the stack descriptor, describes the length of the memory area assigned to a stack branch, and its location in either main memory or disk.

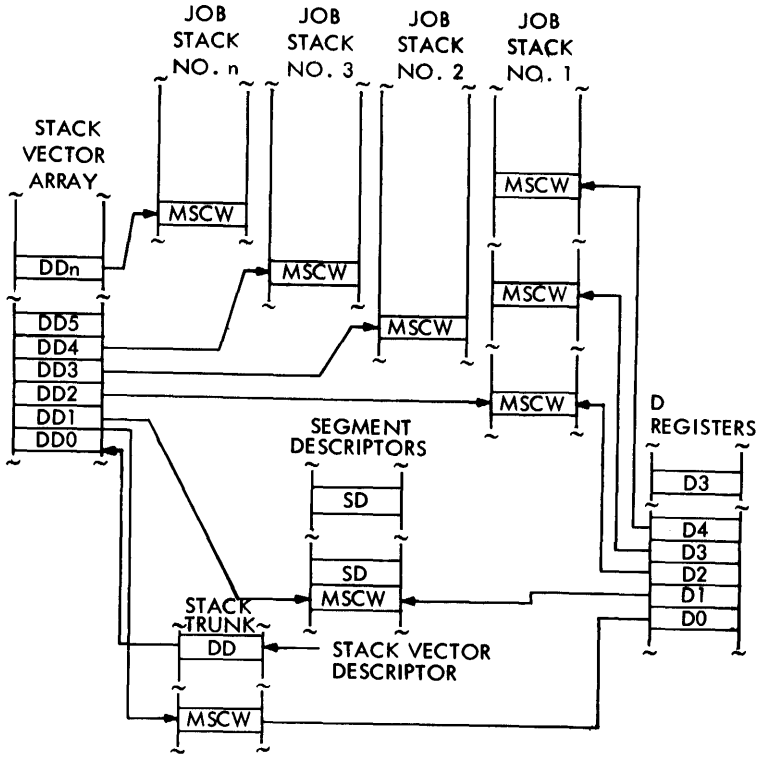


Figure 3-10. Multiple Linked Stacks

A stack number is assigned to each stack branch. The stack number is the index value of the stack descriptor in the stack vector array.

STACK VECTOR DESCRIPTOR. The stack vector array's size and location in memory is described by the stack vector descriptor. This descriptor is located in a reserved position of the stack's trunk (figure 3-10). All references to stack branches are made through the stack vector descriptor, indexed by the stack number.

PRESENCE BIT INTERRUPT. A Presence Bit Interrupt results when an addressed stack is not present in memory. This Presence Bit Interrupt facility permits stack overlays and recalls under dynamic conditions. Idle or inactive stacks may be moved from main memory to disk as the need arises, and when subsequently referenced generates a Presence Bit Interrupt to cause the MCP to recall the nonpresent stack from disk.

SECTION 4
MAJOR REGISTERS AND CONTROL PANELS

PROCESSOR REGISTERS.

GENERAL.

The Processor Registers and Flip Flops are displayed in the Display Cabinet of the system as shown in figure 4-1. Panel A displays the stack registers. Panel B is shared with the Multiplexor. Panels C, D, and E contain indicators and switches for the entire system.

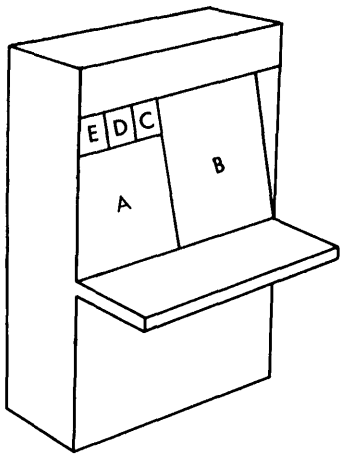


Figure 4-1. Processor Display Panels

PANEL A (Refer to figure 4-2)

A REGISTER. The A register is a 51 bit information register that holds one complete word. This register is the TOP OF STACK when AROF on indicates that it contains a valid word. It is used in many ways, arithmetic, boolean, character string, addressing, indexing, camparing, etc.

B REGISTER. The B register is a 51 bit information register considered as the second word in the Stack when the A Register is valid. It, too, has multiple usage such as arithmetic, boolean, character string, addressing, etc. The B register is valid when BROF is on.

C REGISTER. The C register is a 51 bit information register for general purpose use. It may contain an address, an IRW, an information word, a character or the "flash back" from a memory cycle.

X REGISTER. The X register is a 51 bit information register used basically as the second word of a double-precision operand.

Y REGISTER. The Y register is the counterpart of the X register for double-precision operands. It is the second-word of the B register operand.

P REGISTER. The P register is a 51 bit instruction register.

PANEL B (Refer to figure 4-3).
Panel B indicators are shared by the Processor and Multiplexor Flip Flops.

The PROC/MPX switches located on Panel C (Refer to figure 4-4) control the display mode of this panel.

Panel B is divided into related family and control groups. The Maintenance Diagnostic Logic (MDL) Processor is common to both display modes.

ROW A. This contains the Flip Flops for addressing the IC memories in the Memory Controller.

BRS0 => BRS7 - Base read select 0 thru 7
IRS0 => IRS7 - Index read select 0 thru 7
BWS0 => BWS7 - Base write select 0 thru 7
IWS0 => IWS7 - Index write select 0 thru 7
DRS0 => DRS5 - Display read select 0 thru 5
DWS0 => DWS5 - Display write select 0 thru 5

ROW B. This row contains the Flip Flops for the MDL Processor. There are three registers A01 => A10, B01 => B08, and C01 => C08 associated with this processor. Each has a dual purpose depending on the use of MDL on I/O testing or system testing. The other Flip Flops in Row B are for MDL control.

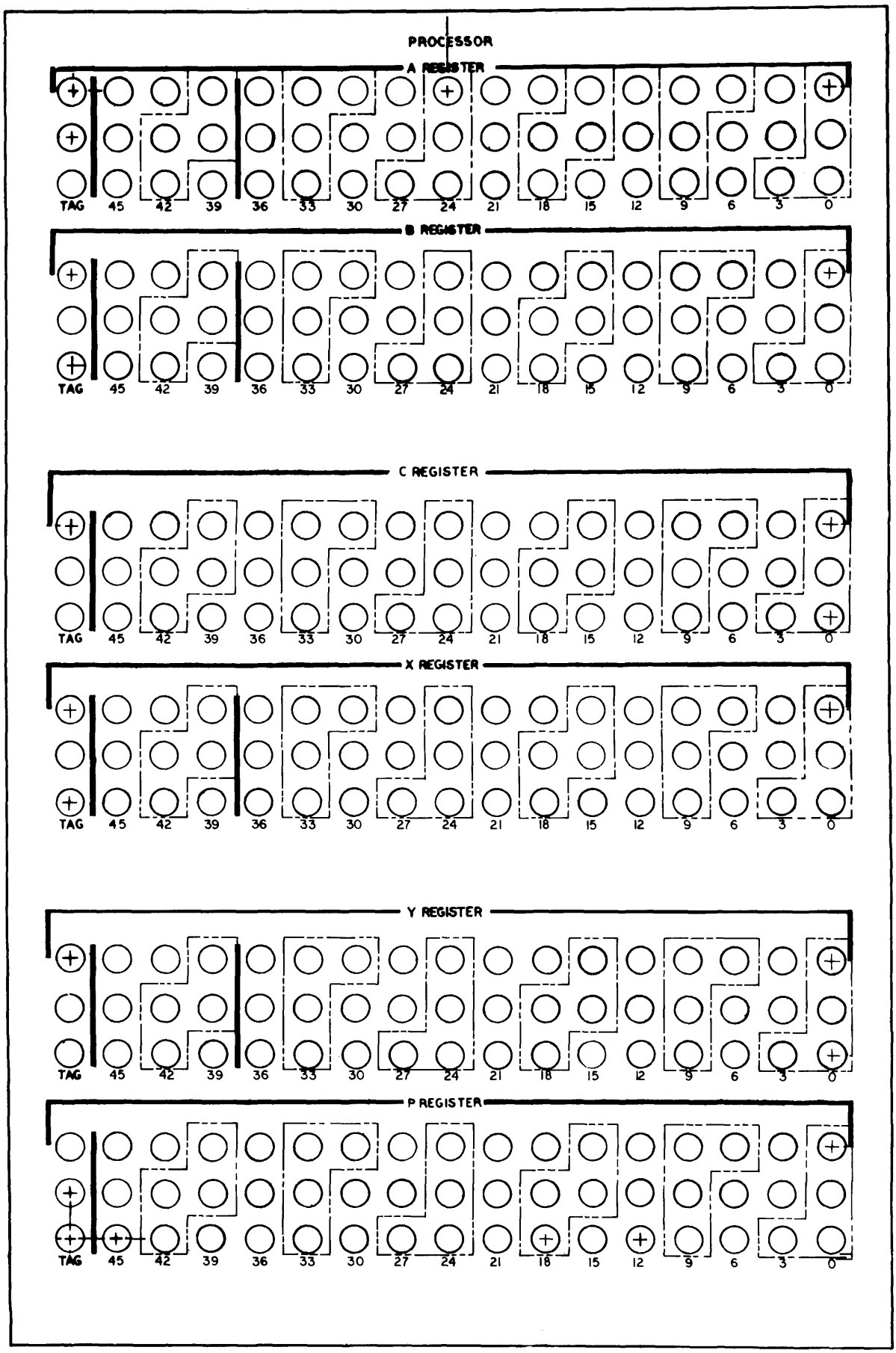


Figure 4-2. Processor Display Panel

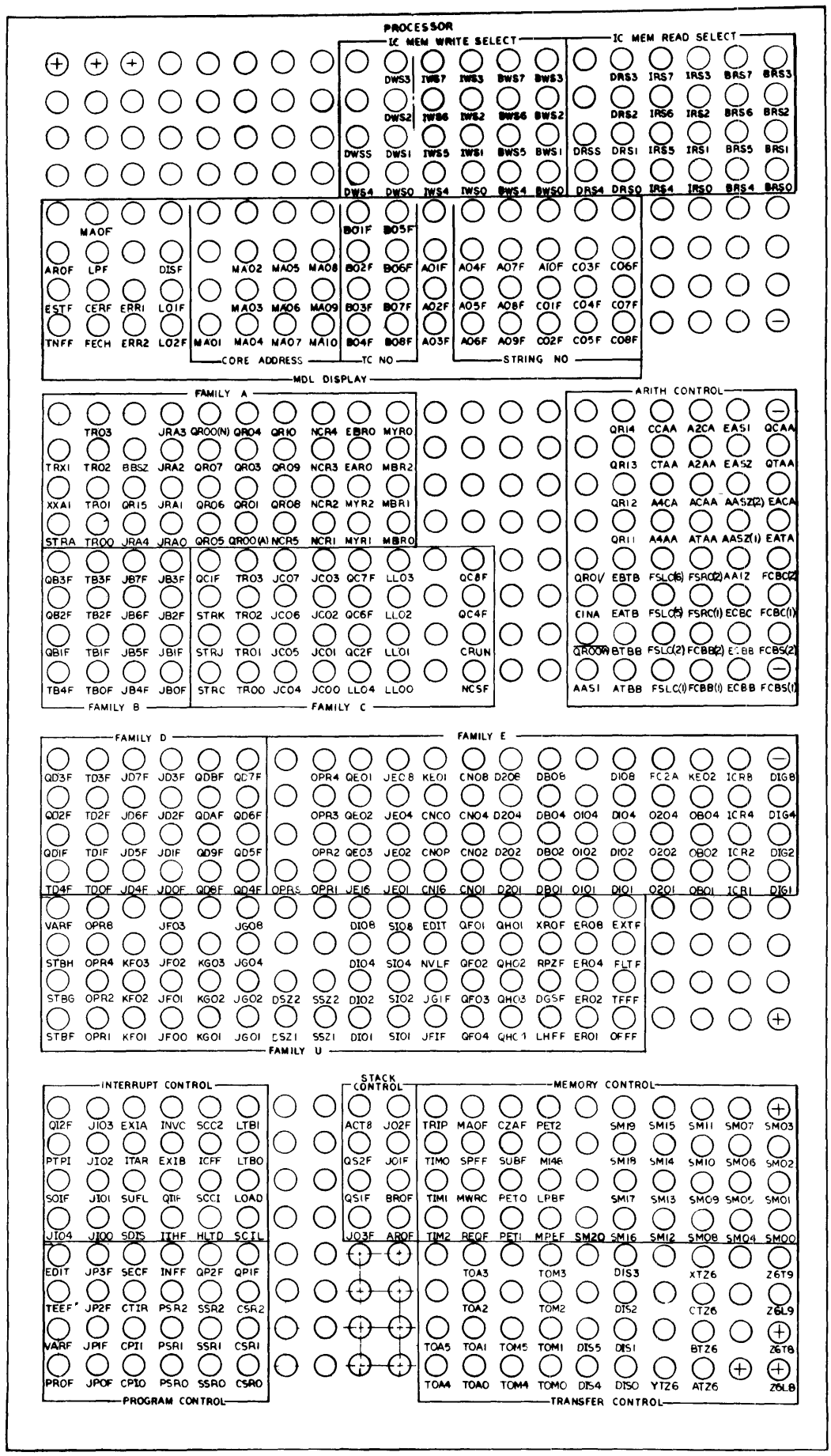


Figure 4-3. Processor/Multiplexor Display Panel

ROW C. This row contains Family A Flip Flops and one half of the Arithmetic Controller Flip Flops.

Family A.

TR00 => TR03	-	Contains the OP Code
JRA0 => JRA4	-	Sequence Count used in the OP Code flow
QR01	-	Pre-Carry INTO Adder
QR00(A)	-	Carry-in set control
QR02	-	High-speed clock phase control
QR03 => QR07	-	Logic control
QR08 => QR10	-	Temporary storage
QR11 => QR14	-	Q Counter
QR15	-	Interrupt flip flop
QR00/QR01	-	Carry-in reset control
QR00(N)/CINA	-	Multiple control
NCR0 => NCR5	-	N Counter
MBR0 => MBR2	-	B Register Mantissa Field Extension
MYR0 => MYR2	-	Y Register Mantissa Field Extension
EAR0	-	Extension of A Register Exponent Field
EBR0	-	Extension of B Register Exponent Field
STRA	-	Family A Strobe F.F. (turned on by the Program Controller thru the Z10 bus)
XXA1	-	Function Parallels STRA
TRX1	-	Function Parallels TR01

Arithmetic Control.

All other Flip Flops in this Controller are used for logic control. They are:

BBSZ	FCBB (2)
AAS1	FSRC (1)
ETBT	FSRC (2)
EATB	AA1Z
BTBB	ECBC
ATBB	ECCB
FSLC (1)	ECBB
FSLC (2)	FCBC (1)

FSLC (5)	FCB1 (2)
FSLC (6)	FCBS (1)
FCBB (1)	FCBS (2)

ROW D. This row contains the Family B and C Flip Flops.

Family B.

TB0F => TB4F - Contains the OP Code

JB0F => JB6F - Sequence Count used in the OP Code flow

QB1F => QB4F - Logic Control

Family C.

TR00 => TR03 - Contains the OP Code

JC00 => JC07 - Sequence Count used in the OP Code flow

LL00 => LL04 - Lexicographical level Flip Flops for the Program flow

STRC - Strobe Family C (Sub-routine)

STRJ - Strobe Family J (Value Call)

STRK - Strobe Family K (Name Call)

QC1F => QC8F - Logic Control

NCSF - Normal Control State Flip Flop OFF signifies Normal State

The Control State Flip Flop extends the Operator set to include some additional Operators and disables external interrupt detection by the Processor.

CRUN Family C Run Flip Flop

ROW E. This row contains the Family D and E Flip Flops.

Family D.

TDOF — Family D Strobe

TD1F => TD4F - Contains the OP Code

JDOF => JD7F - Sequence Count used in OP Code flow

QDIF => QD9F, QDAF, QDBF - Logic Control Flip Flops

Family E.

OPRS - Family E Strobe

OPR1 => OPR4 - Contain the OP Code

JE01 => JE16 - Sequence Count used in OP Code flow

DIG1 => DIG8 - Length Field

ICR1 => ICR8 - Input Convert

OB01 => OB04 - Octal Buffer Bit

0101 => 0104 - Octal 1 bit

0201 => 0204 - Octal 2 bit

DB01 => DB08 - Digit Buffer Bit

D101 => D108 - Digit 1 Bit

D201 => D208 - Digit 2 Bit

CN01 => CN16 - Counter

QE01 => QE03 - Logical Control

ROW F. This row contains the Family U (String OP) Flip Flops. Family U is the hardware logic for the STRING OP CONTROLLER.

OPR1 => OPR8 - Contains the OP Code for this controller

KF01 => KF03 - Extension of Sequence count for family F

JF00 => JF03 - Sequence Count used in Family F OP Code flow

KG01 => KG03 - Extension of Sequence count for family G or H

JG01 => JG08 - Sequence Count used in Family G or H OP Code flow

VARF - Variant Flip Flop to alter the OP Code

DSZ1 - Destination Size Less Significant Bit

DSZ2 - Destination Size More Significant Bit

SSZ1 - Source Size Less Significant Bit

SSZ2 - Source Size More Significant Bit

DI01 => DI08 - Destination Character Pointer

SI01 => SI08 - Source Character Pointer

EDIT - Edit Mode for String OPS

NVLF - Invalid OP Code

JGIF - JG Interrupt State

JFIF - JF Interrupt State

QF01 - Invalid OP Interrupt

QF02 - Presence Bit Interrupt

QF03 - Memory Protect Interrupt

QF04 - Segmented Array Interrupt

QH01 => QR01 - Logical Control

XROF - Register Occupied

RPZE - Logical Control

DGSF - Logical Control

LHFF - Logical Control

ER01 => ER08 - E Register Flip Flops (used for Memory Cycle requests during String OP Code flow)

EXTF - External Sign

FLTF - Float

TFFF - True False

OFFF - Overflow

ROW G. This row contains the Flip Flops for the Interrupt Controller, the Stack Controller and the Memory Controller.

Interrupt Controller.

JI00 => JI04 - Sequence Count for Controller Flow

S01F - Stack Overflow

PTPI - Processor to Processor Interrupt

QI1F, QI2F - Logical Control
EXIA - External Interrupt A (MPX-A)
EXIB - External Interrupt B (MPX-B)
ITAR - Interval Timer Armed
SUFL - Stack Underflow
SDIS - Syllable Dependent Interrupt
SCC1, SCC2 - Scan Counter Bit 1 and 2
ICFF - Interrupt Controller Run
HLTD - Halted
LOAD - Load
SCIL - Scan Interlock
LTB0, LTB1 - Load Timer Bit

Stack Controller.

J01F => J03F - Sequence Count for Controller flow
ACT8 - Address Couple to Z8 bus
QS1F, QS2F - Logical Control
AROF - The A Register Contains a Valid Word
BROF - The B Register Contains a Valid Word

Memory Controller.

SM00 => SM20 - Address Adder Output Flip Flops. These are for display only. (No Manual Set or reset Controls)

TRIP - Trip Control Invalid Address

TIM0 => TIM2 - Invalid Address Timer

MAOF - Memory Address Obtained

SPEF - Scan Bus Parity Error

MWRC - Memory Write Control

REQF - Memory Request

CZAF - Carry Zero Control

SUBF - Address Adder Subtract

PET0 => PET2 - Information Parity Test Control Register Bit

MI48 - Memory Protect Bit

LPBF - Line Parity Bit from Memory

MPEF - Memory Parity Error

ROW H. This row contains the Flip Flops for the Program Controller and Transfer Controller.

Program Controller.

JP0F => JP3F - Sequence Count for Controller Flow

PROF - The P register contains a Valid Word

VARF - Variant Mode FF (used to escape to 16 Bit Instruction)

TEEF - Table Enter Edit

EDIT - Edit Mode

CPI0, CPI1 - A 2 bit counter used to back up the PIR (Program Index Register)

CTIR - A 1 bit counter used to back up the TIR (Table Index Register)

SECF - Sec1 (Syllable Execute Complete Level) Saved

INFF - Inhibit Fetch FF (used to inhibit bringing a new program word to the P register)

PSR0 => PSR2 - Program Syllable Register 0=>5 Pointer (Points to next syllable to be executed from the P register)

QP1F, QP2F - Logic Control

SSR0 => SSR2 - Syllable Saved Register 0 (used to save the current position of PSR when in Table Mode)

CSR0 => CSR2 - Command Syllable Register 0=>5 (used to save the current position of PSR)

Transfer Controller.

TOA0 => TOA5 - Top of Aperature Flip Flops (used to select top bit of 48 bit field to be transferred thru the steering and mask network)

TOM0 => TOM5 - Top of Mask Flip Flops (used to select top bit of 48 bit field to be inhibited thru the steering and mask network)

DIS0 => DIS5 - Displacement Flip Flops (used in steering network to logically displace bits of a 48 bit field)

YTZ6
XTZ6
CTZ6
BTZ6
ATZ6 - Gating Flip Flops to the Z6 bus. (Allows the contents of the various registers to be gated to this bus)

Z6L8 - Z6 Bus Lower to Z8 Bus (Allows bits 13:14 to be transferred)

Z6T9 - Z6 Bus Top to Z8 Bus (Allows bits 39:20 to be transferred)

Z6L9 - Z6 Bus Lower to Z9 Bus (Allows bits 35:16 to be transferred)

Z6T9 - Z6 Bus Top to Z9 Bus (Allows bits 39:20 to be transferred)

GENERAL MAINTENANCE CONTROLS.

The maintenance control panel shown in figure 4-4 is panel C. It contains the indicators and necessary controls for maintenance of the B 6500 system. Units which cannot be controlled from this panel have their own local maintenance controls.

POWER CONTROLS.

Power supplied to the B 6500 system can be controlled by two sequence control circuits. (Sequence Control Circuit A and B). There are two sets of power control switches located on the upper-right corner of panel C. These are the power-on switch and the power-off switch. One set controls sequence control A, and the other controls sequence control B. Besides the power on and off switches there is a set of three toggle switches labeled connect-disconnect, "A", "B", or "C". These switches establish the mode in which the "Power on and off" switches are used. When switches A, B, and C are in the disconnect position they indicate that power section "A", "B", and "C" are controlled independently by their respective switches. When switch "C" is in the connect position, we can connect power section "A", and "B" to a third section "C" and have a common control.

Lamp indicators "1, 2, 4, and 8" indicate the failure of one of 15 AC modules. For example, if AC module #7 has failed, indicators labeled "1", "2" and "4" will turn on.

GENERAL CLEAR AND HALT-LOAD FUNCTION.

On the upper-right corner of the control panel there are two push-button switches labeled "General Clear A, and General Clear B". The domain of each of these switches depends on the positions of switches A, B, And C are in the disconnect mode, section A is cleared with the "GEN CLEAR A" switch, and section B is cleared with the "GEN CLEAR B" switch. If a third section "C" exists, it will have its own general clear. If switches A, B, and C are in the connect positions, sections A, B, and C are cleared whenever either one of general clear switch A or B is depressed.

There is no direct clear switch located at the operator's console; however, system's general clear from this unit is provided through the "load" switch. Whenever the load switch is depressed, the system is automatically cleared before the load command is executed.

The "HALT", "LOAD", and "CARD LOAD SELECT" switches are duplicated at the maintenance panel (panel C) for convenience of operation. These switches are located in the lower-left corner of panel C.

System's clear through Load Switch: When the "load" switch at either the console or the maintenance panel is depressed, a clear signal is generated. Both sections A and B are cleared. When the load switch is released, the load logic generates the load command which is transmitted to the data processors.

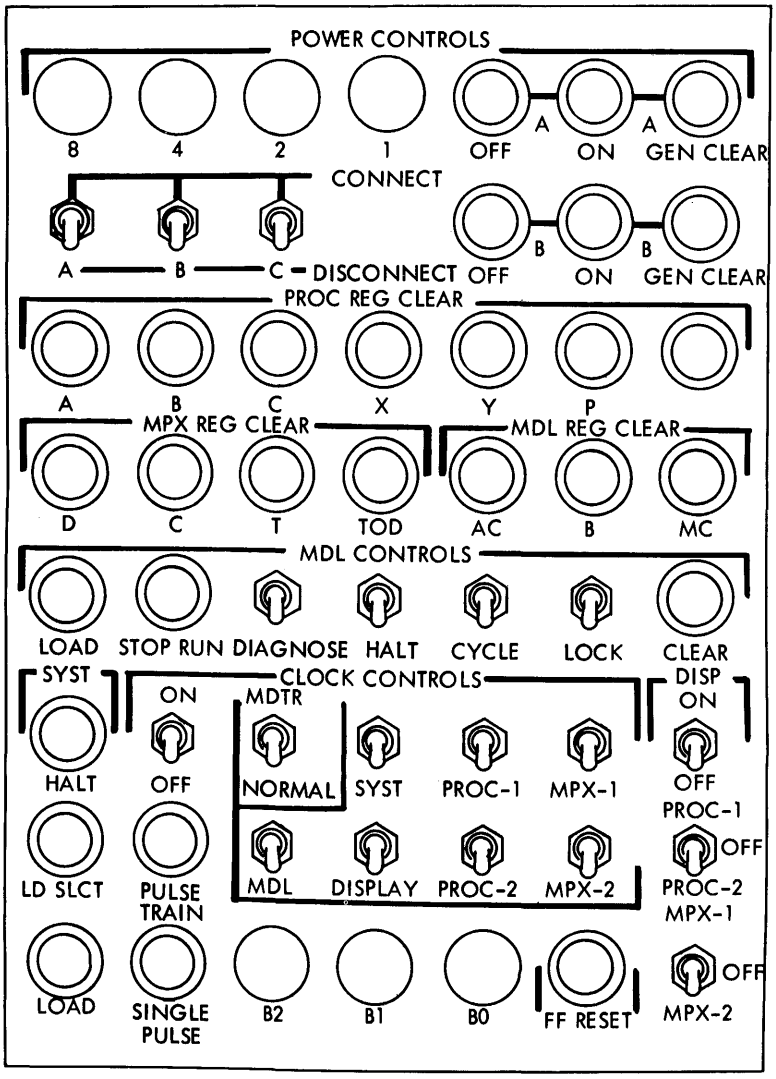


Figure 4-4. Panel C General Controls

PROCESSOR REGISTER CLEAR.
A set of six pushbutton switches is provided for individual clear of registers A, B, C, X, Y and P of the Data Processor selected by the display select switch.

MULTIPLEXOR REGISTER CLEAR.
The Multiplexor registers may be individually cleared with the switches listed below:

a. Switch D clears the Data Register.
b. Switch C clears the Command Register.
c. Switch T clears the Tag Register.
d. Switch TOD clears the Time of Day Register.

MDL REGISTER CLEAR.
The MDL registers may be individually cleared with the switches listed below:

a. Switch MC clears the Core Address.
b. Switch B clears the TC No.
c. Switch AC clears the String No.

MDL CONTROL SWITCHES.
This group of switches is used for loading and controlling the Maintenance Diagnostic Logic.

DISPLAY SELECT SWITCHES.
This group of switches is composed of three toggle switches located in the lower-right corner of the panel. The function of these switches is as follows:

a. On-Off Switch: This switch enables or disables the display logic.

b. Processor Select Switch: This is a three-position toggle switch which selects which of two processors is scanned by the MDL.

c. Multiplexor Select Switch: This is a three-position toggle switch which selects which of two multiplexors is scanned by the MDL.

CLOCK CONTROLS.
The clock control switches provide the means of inhibiting the system clock to the various components of the system.

Clock toggle switches when activated in the "up" position inhibit the following.

a. SYST - Entire system
b. PROC-1 - Processor #1
c. MPX-1 - Multiplexor #1
d. MDL - Maintenance Diagnostic Processor
e. Display - Display Logic
f. PROC-2 - Processor #2
g. MPX-2 - Multiplexor #2

SINGLE PULSE SWITCH.
This switch is used to produce a single clock when the clock has been inhibited.

PULSE TRAIN SWITCH.
This switch is used to produce a train of pulses. Each depression produces all the clock pulses that normally appear within a 500 nano second period.

INDICATORS B0, B1, B2.
These indicators indicate the logical time division of the Pulse Train.

MDTR/NORMAL SWITCH.
This switch is used to change the system from a normal mode of operation to that of Maintenance Diagnostic Logic.

FF RESET SWITCH.
This switch when depressed indicates that a flip-flop in the unit selected is to be reset.

HALT, LOAD, and LOAD SELECT SWITCHES.
The function of these switches is the same as their corresponding switches at the console. The Halt Switch is used to halt the system in an orderly manner. The Load Switch is used to perform a Load Operation as per the positions of the Load Select Switch. The Load Select Switch is used to select a Disk or Card Load operation.

The indicator is lit when Card Load is selected.

NOTE

For a detailed description of the Load operation refer to the description of the Operators panel (Section 4).

PROCESSOR MAINTENANCE CONTROLS (Panel E).
Each processor is provided with an independent maintenance control panel. These controls are additions over and above the console controls (Halt, load, power on/off ---, etc.) and the general systems controls (Panel C).

The I. C. Memory registers of the processor are not displayed by the system's display unit; however, certain switch controls located on the processor control panel allow control and display of these registers.

The control switches provided on the processor control panel and their related functions are described in this section. Reference is made to figure 4-6 which shows a front view of Panel E.

START SWITCH.
The start switch is a pushbutton type switch which functions to start, a halted processor, to execute the next operator syllable

pointed to by "PSR", "PIR", and "PBR". This switch is active only when the processor's clock is enabled and when depressed activates the "SECL" switch to cause the execution of the next operator syllable to be initiated in the normal manner.

CONDITIONAL HALT SWITCH.
This switch is a 2 position toggle switch which functions to enable the conditional halt operation to stop the data processor. The conditional halt operator functions as a "No-Op" when executed with the "CONDITIONAL HALT" switch in the down position and functions to stop the data processor when in the up position.

STOP SWITCHES.
The following set of stop switches enable the data processor to stop upon the occurrence of specified conditions. The exact action of these switches is modified by the position of the "STOP MODE" switches.

SECL SWITCH.
The SECL switch when in the "up" position causes the processor to stop after the execution of each operator syllable. It activates the "INFL" (inhibit fetch level).

INT-I SWITCH.
The stop on internal interrupt switch (INT-I) causes the data processor to stop upon the occurrence of an internal interrupt condition, when in the "up" position. The data processor stops displaying both the P1 and P2 interrupt parameters in the A and B registers just prior to entering the interrupt procedure.

EXT-I SWITCH.
The stop on external interrupt switch (ECT-I) causes the data processor to stop upon the occurrence of an external interrupt, when in the "up" position and if the interrupt system is so enabled. The data processor stops displaying the P1 and P2 interrupt parameters in the A and B registers, just prior to entering the interrupt procedure.

NORMAL/CONTROL STATE SWITCHES.
These are 2 position toggle switches which function to enable the stop switches to function when the data processor is in control state, normal state or both.

PARITY SWITCH.
This switch enables the processor to stop on a memory parity error.

UNIT CLEAR SWITCH.
The unit clear switch is a pushbutton type switch which functions to clear the flip-flops of the related data processor, when depressed.

LOCAL/REMOTE SWITCH.
This switch is a 2 position toggle switch which places the data processor to a local state, when placed in the "LOCAL" position. The processor unit functions normally when in the local state except for the following:

a. The scan bus is isolated from the system functionally, so that manual intervention within the processor will not interfere with the rest of the system.

b. The facilities of the "READ PROC REG" switches are enabled.

ADJ (0,0) SWITCH.
This is a pushbutton switch which activates the Push Down Stack Register operator to cause all TOS registers to be stored in memory, therby saving the contents of the A and B registers so that these registers may be used to subsequently manipulate the data processor's I.C. memory via the maintenance panel switches (READ-IC and WRITE-IC). The ADJ (0,0) switch is active only when the processor's clock is enabled.

READ IC SWITCH.
This is a pushbutton switch which initiates a Read Processor Reg-

ister operator to read the contents of a processor IC memory register into the A register (19:20). The address of the selected IC memory register must be placed into the B register prior to depressing this switch. The "READ IC" switch is active only when the processor's clock is enabled.

READ IC OPERATION.

a. Adjust 0,0.

b. Load the address in the B register.

c. Turn BROF on.

d. Depress the READ IC pushbutton; the contents of the addressed cell will appear in the A register.

WRITE IC SWITCH.

This switch is a pushbutton switch which activates a Set Processor Register operator to cause the contents of a processor IC memory register to be replaced with the contents of the A register. (19:20). The address of the selected IC memory register must be placed into the B register prior to depressing this switch. The "WRITE IC" switch is active only when the processor's clock is enabled.

WRITE IC.

a. Adjust 0,0.

b. Load the address in the B register.

c. Load the information to be written in the A register.

d. Turn on AROF and BROF.

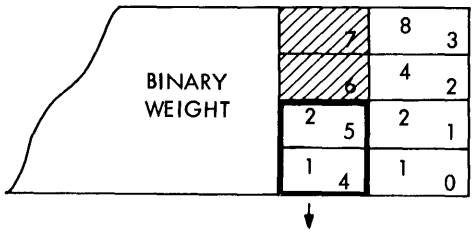
e. Depress the WRITE IC pushbutton; the contents of the A register will be written in the cell addressed.

READ PROC REG SWITCHES. These switches enable the read out and display of the related processor register (IC memory register). The register's contents are displayed only while the switch is depressed, releasing the switch allows the processor to revert to its prior state. The "READ PROC REG" switches activate a DC read out of the IC memory cells and as a result are enabled only when the processor is in "LOCAL". The "READ PROC REG" switches along with their functions are listed below:

a. Switch S is the read S register switch.
b. Switch F is the read F register switch.
c. Switch PBR is the read PBR register switch.
d. Switch PIR is the read PIR register switch.
e. Switch BOSR is the read BOS register switch.
f. Switch LOSR is the read LOS register switch.

NOTE

These IC memories are displayed in the SM register.



0 = DISPLAY REG 0 =⟩ 15
1 = DISPLAY REG 16 =⟩ 31
2 = INDEX REG 0 =⟩ 7
3 = BASE REG 0 =⟩ 7

Register Name	Usage	Decimal Address	Hexidecimal Address
D00		0=⟩	00=⟩
D31	Display	31	1F
PIR	Program Index	32	20
SIR	Source Index	33	21
DIR	Destination Index	34	22
TIR (BUF3)	Table Index	35	23
LOSR	Limit of Stack	36	24
BOSR	Base of Stack	37	25
F	MSCW Address	38	26
BUF	Used for Temporary Storage	39	27
PBR	Program Base	48	30
SBR	Source Base	49	31
DBR	Destination Base	50	32
TBR (BUF2)	Table Base	51	33
S	Top of Stack Address	52	34
SNR	Stack Number	53	35
PDR	Program Segment Descriptor Index	54	36
TEMP	Temporary Storage	55	37

Figure 4-5. Address Register

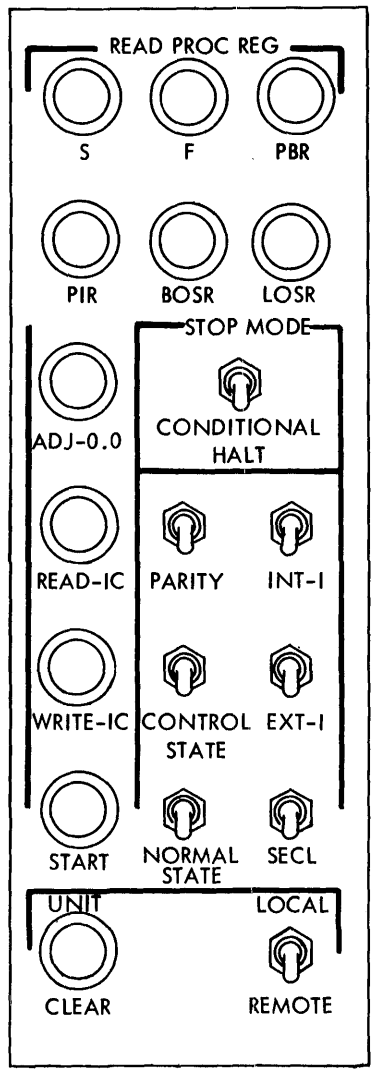


Figure 4-6. Panel E

MULTIPLEXOR REGISTERS AND FLIP FLOPS.

The MPX Registers and Flip Flops are displayed on Panel B as seen in figure 4-7. This panel is shared with the Processors for display mode.

ROW B.

This row contains the logical elements for Maintenance Diagnostic Logic. Each flip flop may be used in one of two ways, I/O testing or Data Processor testing.

FLIP FLOP	USE ON I/O TEST	USE ON DP TEST
FECH	OFF FOR I/O	ON FOR DP
AROF	NOT USED	A,C REGISTER OCCUPANCY
ESTF	TAPE VERTICAL PARITY	END OF STRING FLIP FLOP
TNFF	"TEST NOT" FLIP FLOP	"TEST NOT" FLIP FLOP
MAOF	MEMORY ACCESS OBTAINED	MEMORY ACCESS OBTAINED
LPF	BAD RECORD MEMORY	MEMORY INFO PARITY BIT
CERF	CONTROL PARITY ERROR	CONTROL PARITY ERROR
ERR1	SOLID ERROR	SOLID ERROR
ERR2	INTERMITTENT ERROR	INTERMITTENT ERROR
L01F, L02F	SEQUENCE COUNT	SEQUENCE COUNT
MA01 => MA10	MEMORY ADDRESS	MEMORY ADDRESS
B01F => B08F	TAPE READ CONTROL	DATA
A01F => A10F	CHARACTER BUFFER WORD BUFFER	COMMAND-DATA

ROW C.
This row contains the 51-bit data register used in I/O operations along with the following control flip flops:

PSYF - Processor Sync
PSRF - Processor Scan Request
SAOF - Scan Access Obtained — Scan Bus Control Flip Flops
MATF - Mark Access Time
STEF - Scan Transmission Error

ROW D.
This row contains the 60-bit Command Register used in I/O operations. Refer to figure 4-7 for a detailed description of this register.

ROW E.
This row contains the 10 sets of Associative Tag Register flip flops used for Scratch Pad Memory assignment. Also within each set of flip flops is the corresponding Read Scratch Pad Memory (RSPM) flip flop.

Row E also contains (5) MTRI flip flops, one for each pair of Tag registers.

ROW F.

This row contains the following MPX Control flip flops:

IC 1 => 8 Initiate Count Cycle for operational sequence flow.

KY 1 => 5 Key Register flip flops used as comparitor selection of Scratch Pad Memory slots.

LK 1 => 5 Link Register used on initiate cycle for Key Register selection.

A1 => A8
B1 => B8 Input Translator Digit Bits
C1 => C8
D1 => D8

ESCF Enable service cycle
EICF Enable initiate cycle
RRDF Read result descriptor
PCTF Service priority control
RCDF Read SPM to Command data register
MTOF Memory time zero
AP2F Address plus 2 store
LSAF Least significant address

ROW G.

This row contains the Time of Day Register and the Interrupt status bit flip flops.

TIME OF DAY 0 => 43 - This register contains 44 flip flops of which 36 are used for time-of-day. The other 6 are used when the entire register is being used during MTR logic card test.

IS 0 => 9 - Interrupt status bits.

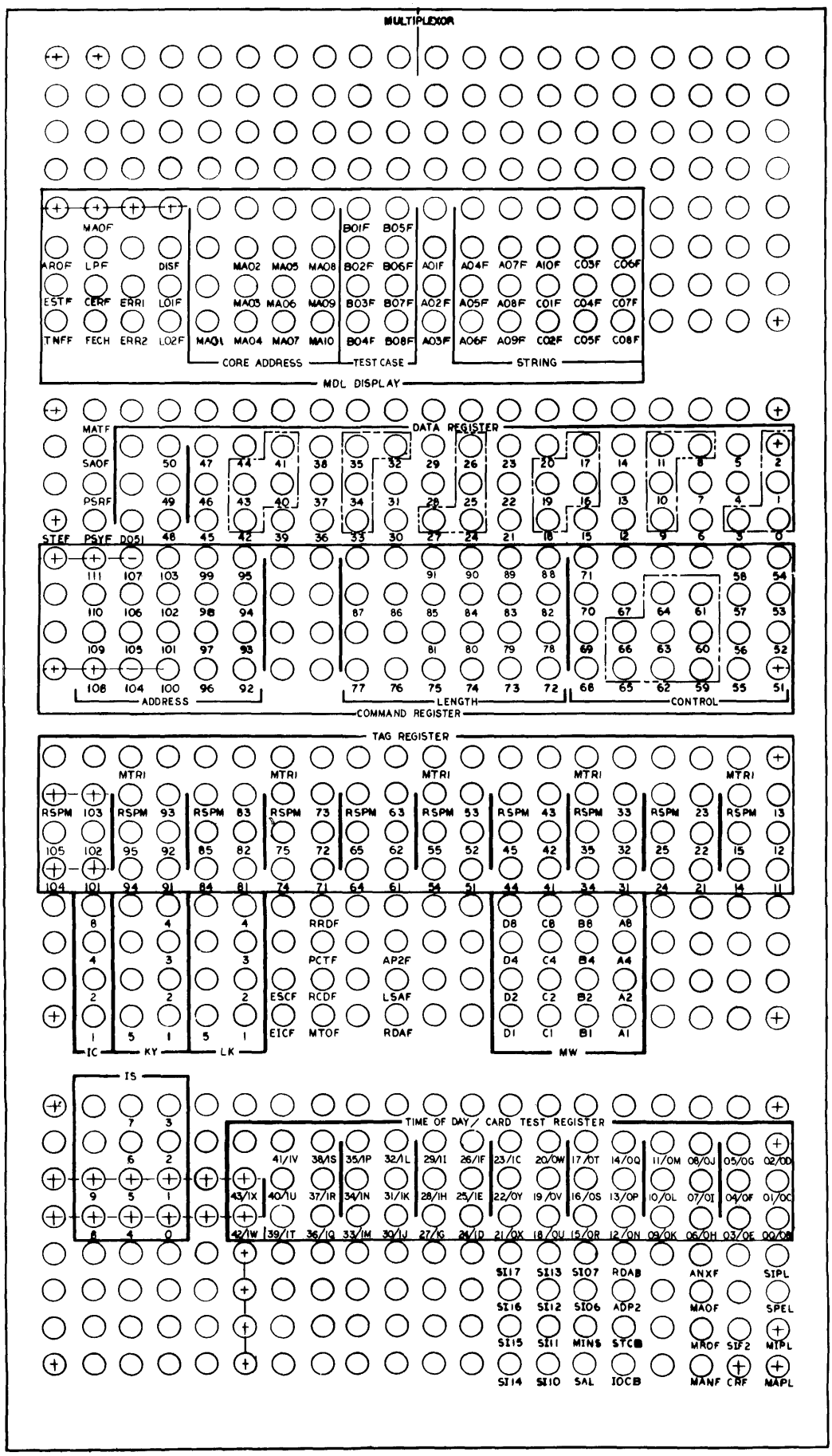


Figure 4-7. Panel B

ROW H.
This row contains the following control flip flops:

MAPL - Memory address parity error level.
MIPL - Memory information parity error.
SPEL - Scan parity error.
SIPL - Scan bus information parity.
CRF - Clear Flip Flop.
SIF2 - Scan In Flip Flop.
MANF - Memory access needed.
MROF - Memory read obtained.
MAOF - Memory access obtained.
ANXF - Allow Next Service Cycle Control.
IOCB - Input/Output Complete Bus.
STCB - Start Channel Bus.
ADP2 - Address Even Bus.
RDAB - Result Descriptor Available Bus.
LSAL - Least Significant Address.
MINS - Minus Bus Level
SI06 =⟩ SI07

MPX MAINTENANCE CONTROL PANEL.
Panel D as seen in figure 4-8 is used for local maintenance operations with the Multiplexor. Four types of operations can be accomplished using this panel:

a. Reading and writing the MPX scratch pad memory.
b. Reading and writing main memory.
c. Executing I/O descriptors.
d. Logic Card testing.

The requirements for these operations are twofold; the MPX must be in local using the Local/Remote switch, the MPX display mode must be active as well as system clock.

The following paragraphs will deal with the operational use of these maintenance switches to accomplish the above 4 modes.

WRITE SPM.

Single or Continuous writing into a SPM location addressed by the tag word is accomplished as follows:

a. Put MPX in Local mode.

b. Scan-in Tag word in the Tag Reg.

c. Scan-in the same Tag word into the Key Reg.

d. Scan-in the desired contents into the Command and Data Registers (112 bits).

e. Put Read/Write switch on the MPX maintenance control panel to the WRITE position.

f. Put Memory/SPM switch on the MPX maintenance control panel to the SPM position.

g. Activate Maintenance Mem/SPM Enable switch on the MPX maintenance control panel.

h. If single cycle operation is desired, press Start button for each SPM write cycle.

i. If continuous recycling is desired, activate the Recycle switch and press Start button to commence recycling.

j. To stop recycling, place Recycle switch to OFF position.

READ SPM.

Single or Continuous reading of a SPM location is accomplished the same as writing except for 2 steps.

Step 4 - Omit

Step 5 - Put the Read/Write switch to the READ position and proceed as in WRITE SPM mode.

WRITE MAIN MEMORY.

Single words can be written to main memory from the Data Register in the following manner:

a. Put MPX in Local mode.

b. Scan-in Memory Address into Command Reg.

c. Scan-in any desired bit pattern into the Data Reg. (Pattern will not clear out of the Data Reg. after each write operation.)

d. Put Read/Write switch on the MPX maintenance control panel to the WRITE position.

e. Put Memory/SPM switch on the MPX maintenance control panel to the MEM position.

f. Activate Maint. Mem/SPM Enable switch.

g. Press Start button for each memory write cycle.

NOTE

Activating Memory Request
Inhibit switch will disable
all logic that might set
MANF including local maintenance.

READ MAIN MEMORY.

Main memory cells may be read either singly or continuously from one address or consecutive addresses in the following manner:

a. Put MPX in Local mode.

b. Scan-in Memory Address into Command Reg.

c. If recycle, use "Write SPM" maintenance logic to write Command/Data Reg. into SPM (highest priority TAG Word with zeros.)

d. Put Read/Write switch on the MPX maintenance control panel to the READ position.

e. Put Memory/SPM switch on the MPX maintenance control panel to the MEM position.

f. Activate Maint. Mem/SPM Enable switch.

g. If single read cycle operation is desired, press Start button for each memory read cycle.

h. If continuous recycling is desired, activate the Recycle switch and press Start button to commence recycling.

i. To manually stop recycling place Recycle switch in OFF position.

j. If stop on error is desired during recycling, activate the Error Stop switch. If a memory parity error or time out occurs, recycling will stop with the Error flip-flop set. Pressing the Start button will clear the error and restart the cycling.

k. Note that activating Memory Request Inhibit switch will disable all logic that might set MANF including local maintenance.

l. Activating the Inhibit Memory Address Count switch, if so desired, will cause retention of the original memory address with each cycle. Otherwise, the memory address will be updated with each memory cycle.

EXECUTING I/O DESCRIPTORS.

SINGLE CYCLE. A single execution of an I/O descriptor found in the Command/Data register is defined below:

a. Put MPX in Local mode.

b. Scan-in Area and I/O Descriptors into Command/Data Registers. The specified Unit Designate will select the channel on which the descriptor is to be executed.

c. Utilize single "Write SPM" procedure for any SPM location using a code of 00001 in Key and Tag Registers.

NOTE

There must be at least one other Tag word available at the beginning of the test.

d. Place Maintenance Mem/SPM Enable switch in OFF position.

e. Place Maintenance Descriptor Enable switch in ENABLE position.

f. Press Start Button once to execute a single maintenance descriptor once for each depression of the Start button.

RECYCLE. Continuous executions of I/O descriptor found in the Command Data Register are accomplished as follows:

a. Steps 1 through 5 are the same as Maintenance Descriptor (single) procedure.

b. Activate Recycle switch.

c. Press Start button to commence recycling of the same maintenance descriptor. A new cycle will be intitiated upon completion of the previous I/O operations defined by the maintenance descriptor.

d. To manually stop recycling place Recycle switch to OFF position.

e. If stop on error is desired during recycling activate the Error Stop switch. Upon detection of a result descriptor error from the P.C. or an error in initiating the channel, recycling will stop with the Error flip-flop set. Pressing the Start button will clear the error and restart the cycling.

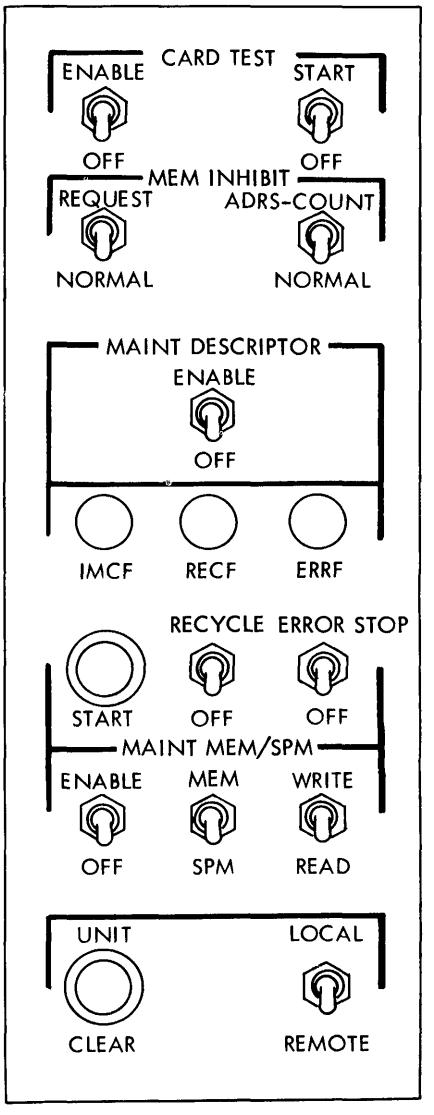


Figure 4-8. Panel D MPX Control Panel

LOGIC CARD TESTING.

Logic Card testing is accomplished by using a MDL test case tape, the Time of Day (TOD) register and a special single card slot located on the MPX backplane. The testing procedure is activated by putting the card test enable switch up, loading the TOD with the appropriate test code and activating the card test start switch. The output of the card under test will be displayed in the 44 flip flops that represent the TOD register.

OPERATORS CONTROL CONSOLE.

The Operators Control Console as seen in figure 4-9 contains an operators panel and a visual message control center for communicating with the Operating system. A total of 8 devices, such as Input Display or TC 500, may be used for this communication.

OPERATOR PANEL.

The operator panel includes the following switches and indicators.

POWER ON (Switch Indicator, White). This Switch/Indicator initiates the power on cycle for all Central System units. The indicator is lit and remains lit as long as power remains on.

NOTE

a. The peripheral units power must be turned on and off at each peripheral unit.

b. When power is turned on, Disk Load is selected.

POWER OFF (Switch, Brown). This switch initiates the power off cycle for all Central System units.

HALT (Switch/Indicator, Red). Halt the system stopping all I/O operations in an orderly manner. The indicator is lit when all processors have been halted.

RUNNING (Indicator, Yellow). This indicator is lit when the system is running. The run state is established by 2 second run multi's in each processor. Each processor multi is triggered by that processor executing an interrogate peripheral unit status operator. The run indicator is lit when the multi in any processor which is in remote is ON. If all processors are in local, the run indicator will also be lit.

LOAD SELECT (Switch/Indicator, Yellow). This switch selects between Disk Load and Card Load. Each time the switch is depressed, the selection is changed. The indicator is lit when Card Load is selected.

LOAD (Switch, Brown). The Load button is used to perform a load operation of the system. Two types of load can be performed as follows:

Card Load Operation.

The Card Load operation is used for initiating the system via the card reader. This type of initiation is used for reading a cold start deck or test routine decks. The following actions occur when the button is depressed and then released:

a. The load timer in the Processor interrupt controller is triggered to produce an 800 nanosecond (LSIG) signal which is sent to MPX-A.

b. Address registers LOSR, BOSR, F, STKNR, and Display 0 are set to Zero.

c. Register S is set to 8192.

d. PDR (Program Dictionary Index) is set to a value of 4.

e. PIR (Program Index Register) is set to a value of 1.

f. The Processor is forced into an idle state to await an expected I/O finished interrupt.

g. g. The MPX responds to the LOAD signal by jamming the appropriate unit number into the Command/Data register. The MPX sequence control logic is set to IC 02 and the card read cycle is started.

h. The information (a bootstrap program) on the EBCDIC punched card is read into the 1st twelve memory locations. This information must contain tag fields (6 chr/wd plus tag).

i. At the end of the successful card read, the MPX sends an I/O finished interrupt to the Processor. It responds by entering a hardward interrupt handling procedure. Memory cell DO 3 (absolute cell 003) contains a portion of the bootstrap program that is subsequently used to handle the interrupt and then causes the remaining card deck to be loaded.

Disk Load Operation.

The Disk Load operation is used for initiating the system by reading 8192 words from the 1st segments of disk memory. This type of an operation is used to bring the 1st portion of the operating system to core memory.

The same hardward functions take place as for card read except for two things:

a. A disk unit number is placed in the Command/Data register because the Load select switch selected a DISK LOAD.

b. The I/O finished interrupt reflects a disk operation instead of a card operation.

VISUAL MESSAGE CONTROL CENTER (Refer to Figure 4-10).

The Visual Message Control Center consists of one or more Input Display Modules each of which contains an input keyboard and a video output screen.

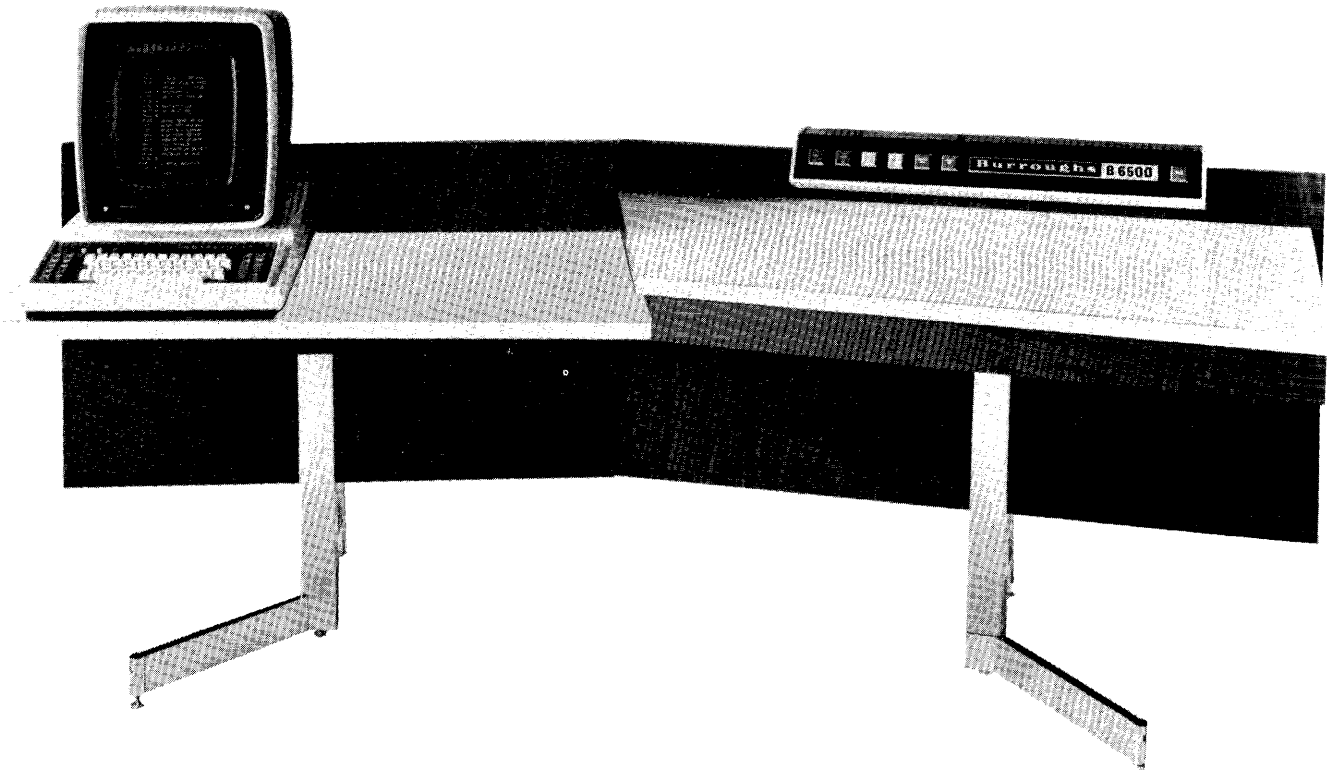


Figure 4-9. Operators Control Console

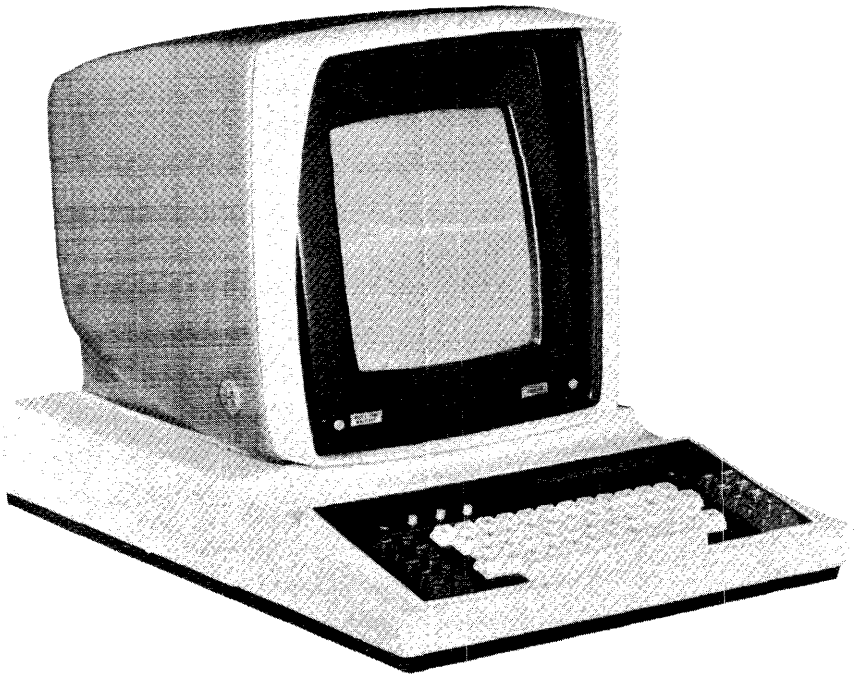


Figure 4-10. Visual Message Control Center

KEYBOARD CONTROL KEYS.

The following is a list of the keyboard control keys and their function. Refer to figure 4-11.

Key	Function
LOC	Places the system in the Local Mode, which lights the Local indicator.
REC	Places the system in the Receive Mode, which lights the Receive indicator.
XMIT	Places the system in the Transmit Mode, which lights the Transmit indicator.

Key	Function
PRINT	Causes the printer mode to be entered. This mode causes the Transmit light to blink. Returns to Local Mode on completion of the print.
⧖ ETX	End of text character. Places the end of text character at the cursor location. a. Shifted - Places the system into the form compose mode and blinks the Local light. b. Unshifted - Takes the system out of the form compose mode.
▷ US	Shift-In - Places a Shift-In (SI) character at the cursor location if the system is in the Form Compose Mode.
◁ RS	Shift-Out - Places a Shift-Out (SO) character at the cursor location if the system is in the Form Compose Mode.
↖ HOME	Causes the cursor to be moved to the home (upper left) position.
LINE ERASE	a. If the system does not have forms option, or if it is in the Form Compose Mode, Line Erase erases all data in the line except tab flags. Data is erased from the cursor position (including the cursor position) up to and including the last character in the line. b. If the system has Forms Options and is not in the Form Compose mode, Line Erase erases all data (except tab flags) that are not bracketed by Shift-In/Shift-Out.

Key	Function
LINE ERASE (cont)	c. Line Erase will not function unless Erase Lock is depressed simultaneously with Line Erase.
↖ CLEAR	a. Unshifted - Clear erases all data on the screen except tab flags and with Forms Option data bracketed by Shift-In/Shift-Out.
	b. Shifted - Clear erases all data on the screen and all tab flags.
	c. Clear will not function unless Erase Lock is depressed at the same time as Clear.
ERASE LOCK	Erase Lock is used as an interlock for Clear and Line Erase. Erase Lock must be depressed to permit operation of the Clear or Line Erase.
TAB	a. Unshifted - Tab causes the cursor to move forward to the next tab stop location. If no tab stop is found on a line, the cursor moves to the left edge of the next line.
	b. Shifted - Shifted Tab is Tab Set. Tab Set causes a tab stop flag to be entered at the cursor position in all lines.
TAB CLEAR	a. Unshifted - Tab Clear causes the removal of the tab stop flag located at the cursor position in all lines.
↓ (Line Feed)	Line Feed moves the cursor down one line. When the cursor is in the bottom line, L.F. causes it to reappear in the top line.

↑ (Reverse Line Feed)

Reverse Line Feed moves the cursor up one line. When the cursor is in the top line, RLF causes it to reappear in the bottom line.

← (Backspace)

Backspace cursor one character. When the cursor is at left edge of page, B.S. causes it to reappear at right edge of page in the same line.

→ (Forward Space)

Forward Space moves the cursor one space to the right. If the cursor is at right edge of page, F.S. causes it to reappear at the left edge down shifted one line. If the cursor is located in last position of bottom line, F.S. causes it to reappear in the Home position.

REPT

If the Repeat key (REPT) is depressed along with any other key except LOC, REC, XMIT, TAB CLEAR or CLEAR, that key will be repeated at a rate of about 15 Hertz. Depressed in conjunction with LOC, REC, XMIT, TAB CLEAR or CLEAR Repeat has no effect.

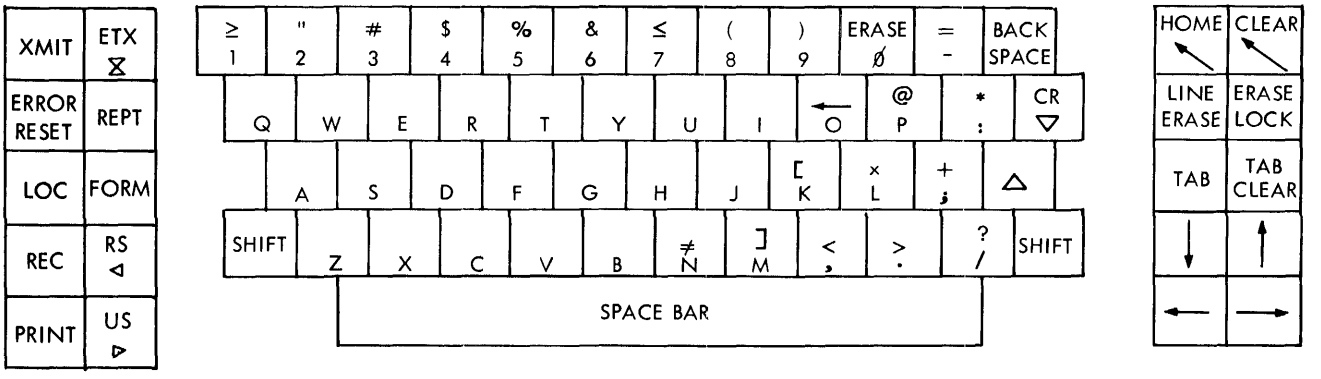


Figure 4-11. Keyboard Format

MEMORY TESTER.

The B 6500 includes a Memory Tester for diagnosing and testing any of the Memory modules attached to the system.

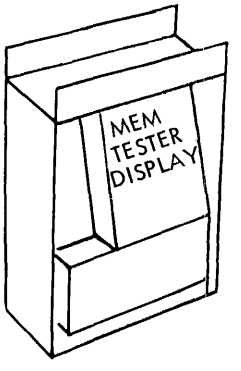


Figure 4-12. Memory Tester

The Memory tester is located in a small cabinet, with its display panel as shown in figure 4-12. The tester can be used in 2 modes, Non-Test or Test (figure 4-13).

NON-TEST.

Three types of operations:

a. Single cycle read or read/write.

b. Search memory(s) for specific data; search for equal or unequal.

c. Sample a given address for changes.

TEST.

The following operations performed using the test pattern switches:

a. None of the patterns selected checks for parity errors using the read only operation.

b. #1 Test-pattern selected enables a fixed test pattern.

c. #2 Test-pattern selected runs an all "one" test.

d. #3 Test-pattern selected runs an all "zero" test.

e. #4 Test-pattern selected runs a checkerboard pattern writing two zeros then two ones.

f. #5 Test-pattern selected runs the checkerboard complement pattern test.

g. #6 Test-pattern selected runs the bit complement pattern test.

h. #7 Test-pattern selected runs the complement bit complement pattern test.

i. #8 Test-pattern selected runs the full walking "one" pattern floating one test.

j. #9 Test-pattern selected runs the full walking "zero" pattern floating zero test.

k. #12 Test-pattern selected runs the memory clear pattern ?= Master reset test.

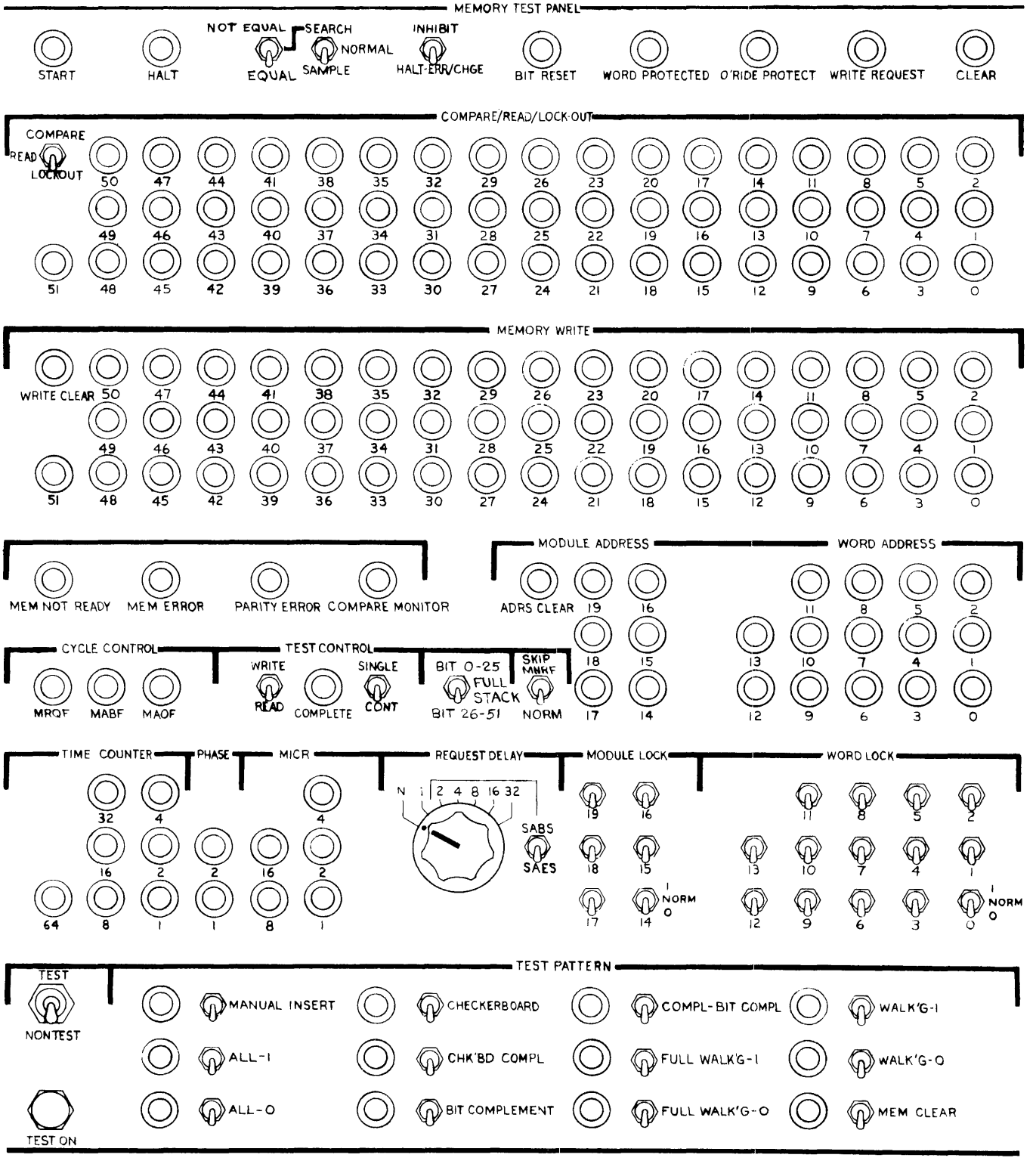


Figure 4-13. Memory Tester Panel

SECTION 5
SYSTEM CONCEPT

GENERAL.

The B 6500 system consists of one or more Processors, one or more I/O multiplexors, Main Memory, a Memory Tester, one or more Power modules, an Operators Console, a Maintenance Diagnostic Processor, a Display Panel, one to four Peripheral Control cabinets and the associated Peripheral equipment for Input/Output. This section generally defines the overall system hardware operation.

PROCESSOR.

The Processor produces the objective results of a program by performing the necessary arithmetic and logical functions of the program flow.

The Processor contains two major divisions: the Functional Resources and Operator Algorithms (figure 5-1). The Functional Resources are referred to as the "hardcore" of the Processor.

OPERATOR FAMILIES.

The Functional Resources are the Arithmetic Unit, Data Registers, Address Processor Unit and Seven Functional Controllers. The operator algorithms provide the logic required to control the functional flow of the program. The ten groups of these operators are called the Operator Family Controllers.

The Operator Family Controllers and Functional Controllers are linked by 13 busses (Z0 through Z12). These busses provide for data movement and signal routing within the processor (figure 5-2).

A bus is a group of wires used to transmit signals from one place to another. The busses within the transfer controller are etched on a single card connecting the same bit of all "hard registers" together, i.e., Bit 1 of Registers A, B, C, X and Y are all on the same physical card.

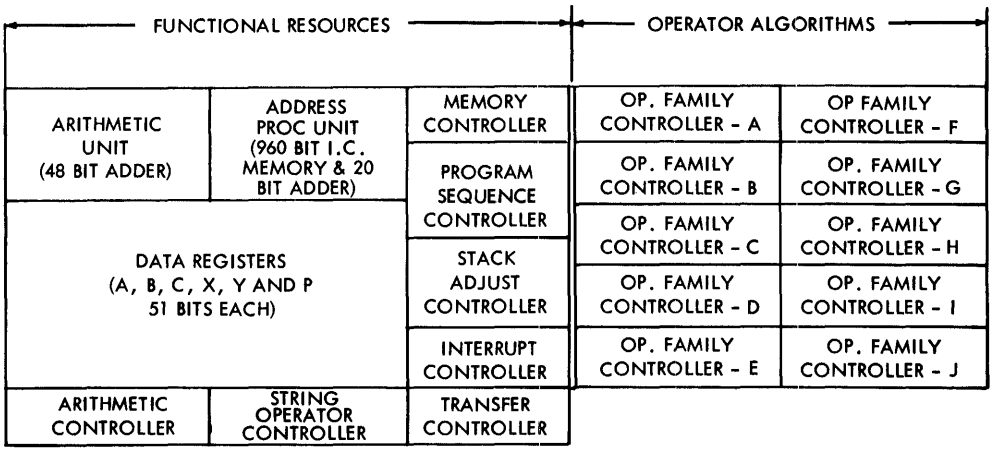


Figure 5-1. B 6500 Processor Organization

The operators are grouped into ten groups called the Operator Families (figure 5-1). The grouping of related operators into families minimizes the logic required in the processor. The Ten families of operators with a brief purpose for each are:

a. Family A OPS - Arithmetic Operators
b. Family B OPS - Logical Operators
c. Family C OPS - Sub-routine Operators
d. Family D OPS - B 6500 Word Oriented Operators
e. Family E OPS - Scaling Operators
f. Family F,G,H, OPS - String Operators
g. Family J OPS - Value Call
h. Family K OPS - Name Call

PROGRAM CONTROLLER (Refer to Figure 5-2). This controller controls the program flow in the following manner: first, it controls the transfer of a program word to the P register via the Memory Controller and Z3 bus in the Transfer Controller. This word contains six 8-bit instruction syllables. It also selects and decodes the syllable to be executed, and furnishes this OP code to all the Family Controllers thru the Z10 bus. The Program controller strobes the proper OP family allowing that OP family to proceed thru its logical

steps performing the function of that operator. At the completion of the operator a SECL (syllable execute complete level) is sensed by the Program Controller which then decodes the next syllable of the P register.

TRANSFER CONTROLLER (Refer to figure 5-2). The Transfer Controller has two major sections: a hard register section, referred to as stack registers, for data and program information, and an internal data transfer section. Six busses, Z1 thru Z6, are used for the normal data movement to and from the hard registers. Z1, Z2 and Z3 are input busses to these registers and Z4, Z5 and Z6 are output busses. The capacity of each bus is 51 bits.

Two special busses are used for arithmetic operations. Z7 is used for transferring data from the A, B or Y registers to the AA register of the high speed adder. Z0 is used for transferring data from the CC register of the high speed adder to the B, C or Y registers as shown in figure 5-5.

Stack Registers.

Each information register has 51 bit positions. Registers A, B, C, X and Y are for information handling during program flow. Register P contains one B 6500 program word. The P register contents are never written into Main Memory.

The Z3 and Z4 busses provide for bi-directional data flow between the hard registers and Main Memory or the Multiplexor.

The A and B registers are the Top of Stack registers, while X and Y are normally second-word information registers for double-precision operands. Register C is a general purpose register which provides temporary storage during syllable execution.

Internal Data Transfer Section (Refer to figure 5-3).

The internal transfer section permits the following data transfers between stack registers:

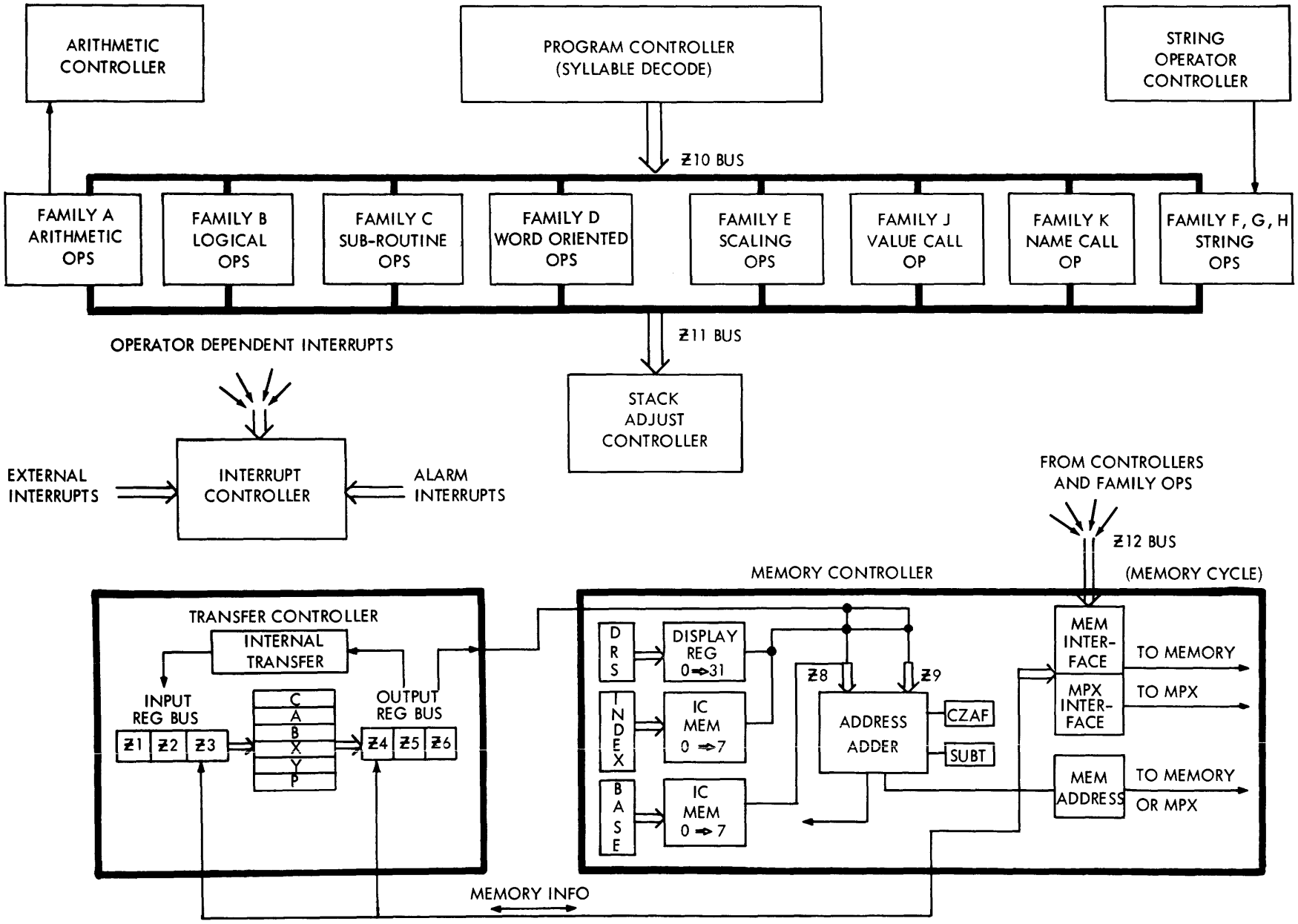


Figure 5-2. B 6500 Processor Block Diagram

a. A direct, full-word transfer path using the Z5 and Z2 busses.

b. A logical transfer path to create the results of the Family B (logical) operators, using the Z4 and Z3 busses. The logical transfer path also provides one additional full word transfer path between registers.

c. A steering Network and Mask network providing a field displacement between stack registers using the Z6 and Z1 busses.

d. An Insert Matrix providing character-handling operators with the ability to store into any of the 4, 6 or 8-bit fields using the Z5 and Z1 busses.

e. A transfer path to the address adder of MEMORY/MPX Controller via the Z6 to Z8 or Z9 busses. This path extracts one of four fields, (39:20), (35:16), (19:20) or (13:14), from a stack register during execution of operator syllables.

f. A data movement path to and from the high speed adder via the Z0 and Z7 busses.

Mask and Steering.

The mask and steering network moves bit fields from register to register, via the Z6 and Z1 busses. All bits are transferred to and from the busses in parallel. Two pointers set up a "window" defining the upper and lower limit of the bits being transferred to the accepting data register. A displacement register shifts the bits to the right, 0 to 47 bits from the position previously held in the sending data register.. The three controls used to steer and mask are:

a. TOA (TOP OF APERATURE) - the highest bit position of the accepting field (highest bit of the window).

b. TOM (TOP OF MASK) - the highest bit position to be inhibited on the transfer (lowest bit of the window).

c. DIS (DISPLACEMENT) - a right shift of the bits through the steering matrix.

The registers TOA, TOM, and DIS are set by the operator families or other controllers.

Mask and Steering Example.

Assume the C register contains a stuffed indirect reference word (SIRW) and it is necessary to extract the STKNR (stack number) field (bits 45:10) and place these bits into the INDEX field of the C register. The logic sets the window TOA := 29, TOM := 19, as shown in figure 5-4. The displacement register is set to 16: DIS := 16. The actual starting bit of the field is calculated as: TOA + DIS = 29 + 16 = 45.

All Bits in the C register are gated to the Z6 bus. The bits (except tag) are then shifted 16 places to the right with only the bits that align with the window appearing on the Z1 bus. The Z1 bus is then gated to the C register with the masked fields destroyed or retained depending on the operation performed.

ARITHMETIC CONTROLLER (Refer to figure 5-2). The Arithmetic Controller is a Functional Controller between the Stack Registers (A, B, C, X and Y) and the Mantissa Adder. This Controller is enabled by the Arithmetic Family Operators and other operator families that require the use of these facilities.

High Speed Adder.

Figure 5-5 depicts the logical flow of data to and from the high speed adder. The adder is made up of three 48-bit registers AA, BB, and CC and the associated add logic. The add logic receives its input from the AA and BB registers. The add logic output is fed into the CC register which feeds either the BB register or the hard registers via the Z0 bus.

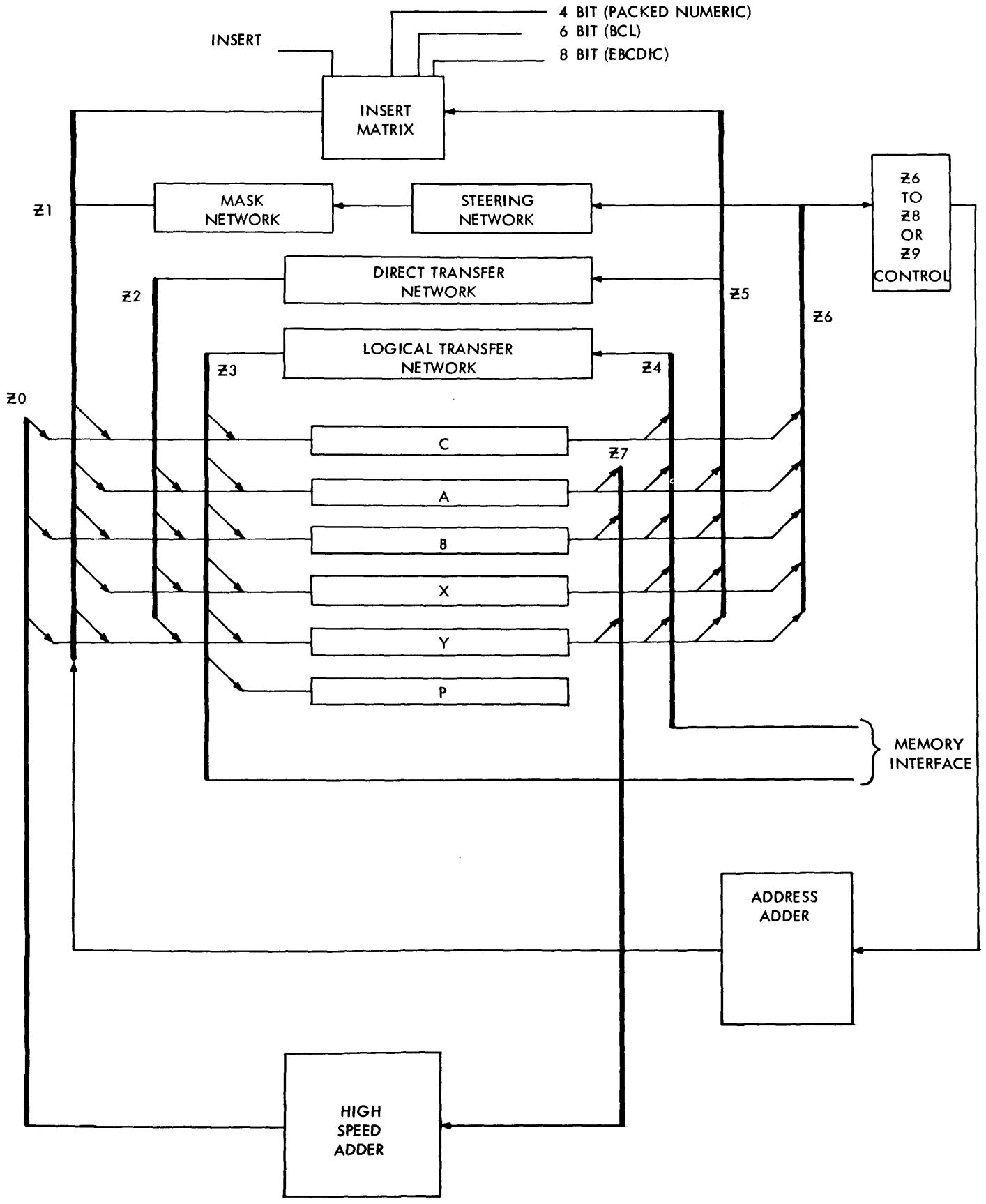


Figure 5-3. Internal Data Transfer Section

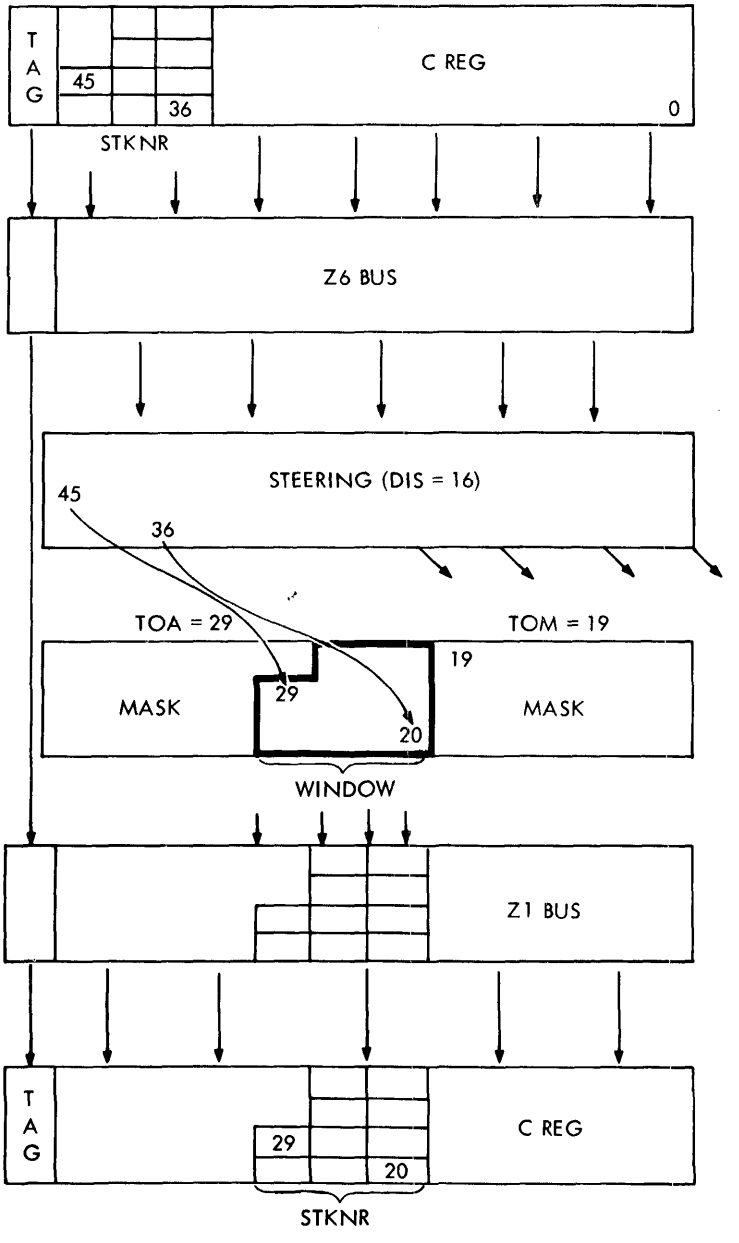


Figure 5-4. Mask and Steering

INTERRUPT CONTROLLER (Refer to figure 5-2). The Interrupt Controller provides a method intervening in the program flow when a predetermined condition arises.

This controller sets up the necessary control words in the stack for entry into the Interrupt-handling procedure. Two identifying words are placed in the stack by the operator or the Interrupt controller.

Internal interrupts are divided into two groups, operator dependent and operator independent interrupts.

The operator dependent interrupts are divided into two classes. Bit 24 of the interrupt ID identifies the interrupt as class 1, where the values of PIR, PSR, PBR and PDR are "consistent". Bit 23 identifies class 2 interrupts where the values were changed by the operator before the interrupt.

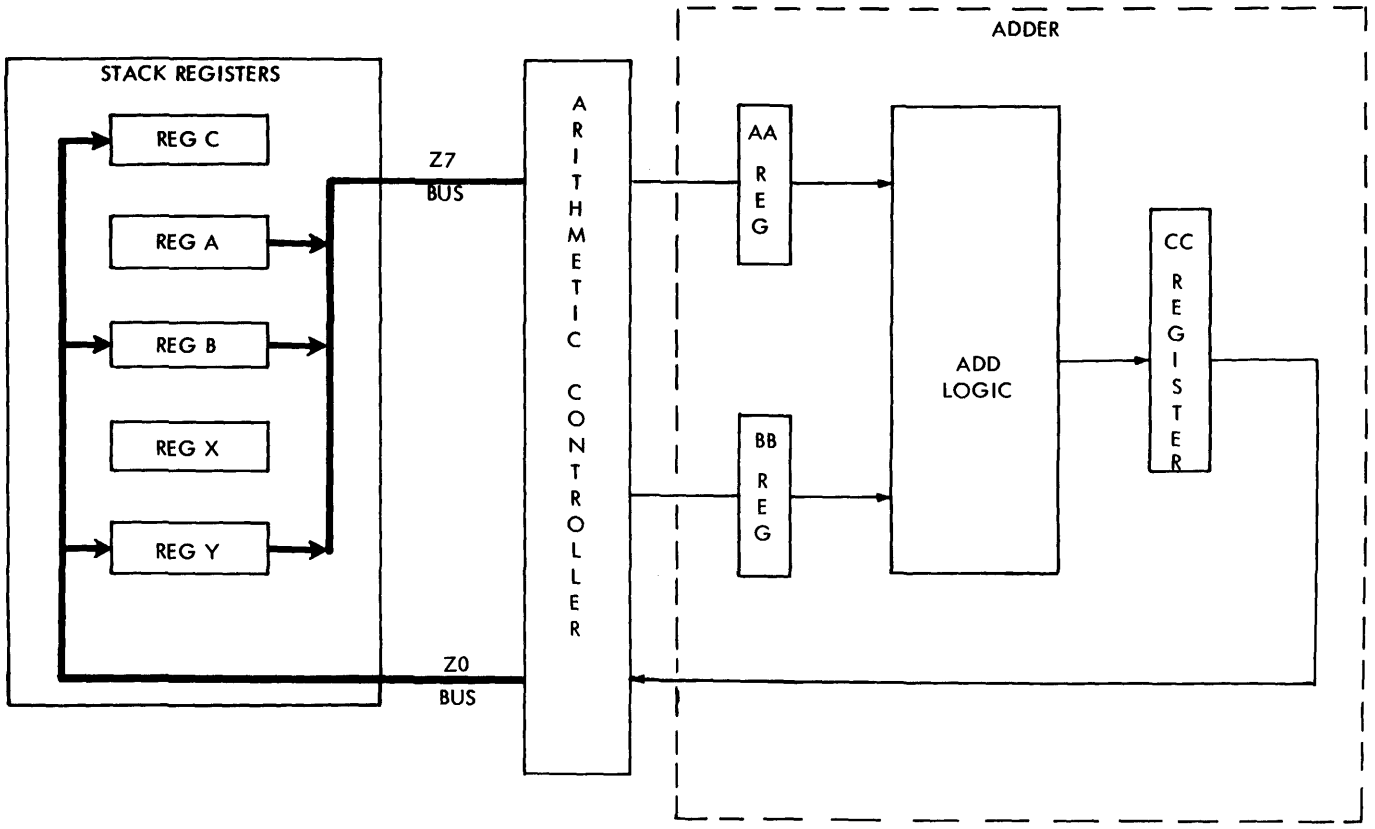


Figure 5-5. Arithmetic Control

Operator Dependent Interrupts.

These interrupt conditions are sensed by the operator and normally results in a premature termination of the operator under control of the operator's own logic. The operator inserts both P1 and P2 parameters into the TOS and activates the interrupt controller. PIR and

PSR are reset to the beginning of the current operator before the interrupt, thus the operator is restarted upon return to the interrupted procedure.

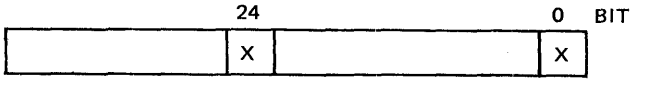
The operator-dependent interrupts are:

a. Memory Protect
b. Invalid Operand
c. Divide by Zero
d. Exponent Overflow
e. Exponent Underflow
f. Invalid Index
g. Integer Overflow
h. Bottom of Stack
i. Presence Bit
j. Sequence Error
k. Segmented Array
l. Programed Operator

Memory Protect.
This interrupt occurs when:

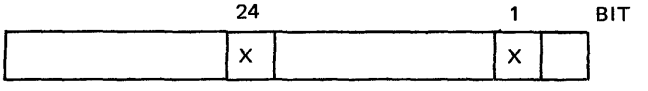
a. A STORE, OVERWRITE, or READ/LOCK is attempted using a Data Descriptor that has the read only bit on (bit 43). The operation is terminated prior to the memory access, leaving the descriptor in the A register.

b. A STORE is attempted into a word in memory that has a tag field representing PROGRAM CODE, RCW, MSCW, or SEGMENT DESCRIPTOR. The memory write is aborted when bit 48 is detected in the "flashback" word that is placed into the C register. The operation is terminated leaving the original addressing word in the A register.



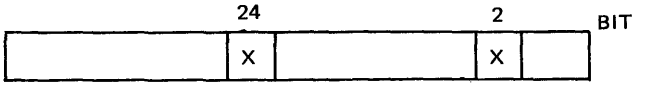
Memory Protect Interrupt ID

Invalid Operand.
This interrupt occurs when operators attempt to use the wrong types of control words or data. When control words and data are accessed, they are checked to meet the necessary requirements of the operator being executed. When the interrupt occurs, the operator is terminated prematurely.



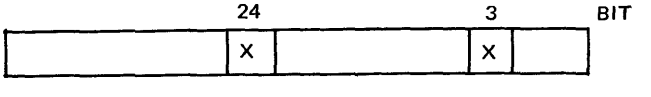
Invalid Operand Interrupt ID

Divide by Zero.
This interrupt results when a division operator is attempted with the divisor equal to zero. This interrupt terminates the operation prematurely, leaves the A register cleared, the interrupt ID in the B register and PSR and PIR backed up to point to the initiating operator.

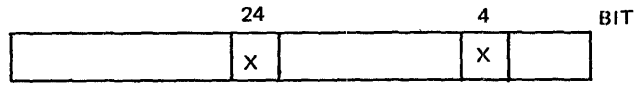


Divide by Zero Interrupt ID

Exponent Overflow and Underflow.
These interrupts occur when the capacity of the exponent field is exceeded for either single or double-precision arithmetic results. The interrupt ID is dependent on the exponent sign and both clear the A register.



Exponent Overflow Interrupt ID

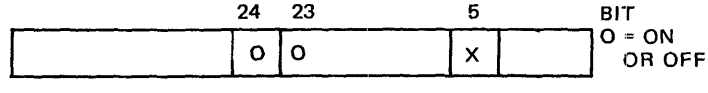


Exponent Underflow Interrupt ID

Invalid Index.
This interrupt is caused by an attempt to index by less than zero or not less than the upper bound (length) in the operations:

		Family
a.	Occurs Index	(A)
b.	Link List Lookup	(B)
c.	Index	(C)
d.	Move Stack	(C)
e.	Display Update	(C)
f.	Dynamic Branch	(C)
g.	Stuffed IRW (pseudo)	(C)
h.	Index and Load Name	(C)
i.	Index and Load Value	(C)

If an index outside the prescribed bound is attempted, the operator is terminated. Backing up PSR, PIR is only done on the first two operators.



Invalid Index Interrupt ID

NOTE

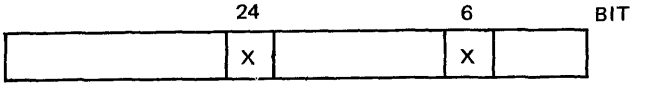
If bit 23 is on,
bit 24 is off.

Integer Overflow.

This interrupt occurs upon detection of attempted uses of operands greater than integer maximum value by operators that require integers. In general, the checking is performed before the operand is converted into an integer by reducing the exponent field. The following operators may invoke this interrupt.

a. Integer Divide (both SP and DP)
b. Integerize Truncated
c. Integerize Rounded
d. Occurs Index
e. Integerize Rounded, Double Precision

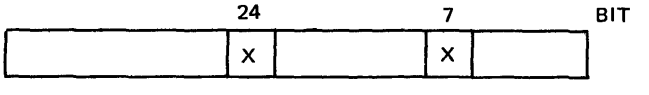
If the interrupt is invoked, the operator is terminated.



Integer Overflow Interrupt ID

Bottom of Stack.

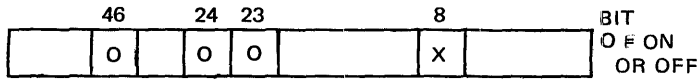
This interrupt is used to inform the Operating System that a RETURN or EXIT Operator has caused the program stack to be cut back to its base. If the condition arises, the operator will terminate with the last accessed RCW (Return Control Word) left in the A register.



Bottom of Stack Interrupt ID

Presence Bit.

This interrupt is used to inform the system that an attempt has been made to access a quantity not present in main memory. All operators that access memory with descriptors have the ability to set this interrupt. Special consideration is given to this type of an interrupt for data or procedure-dependent descriptors.



Presence Bit Interrupt ID

Special Consideration-Presence Bit Interrupts.
There are two classes of presence bit interrupt conditions.

a. Data Dependent
b. Procedure Dependent

Each class requires that the PIR and PSR value for the RCW be manipulated differently.

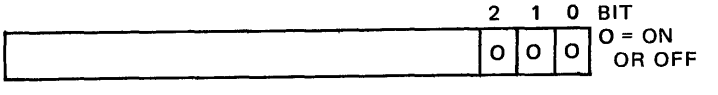
Data-Dependent Presence Bit. The Data-Dependent Presence Bit Interrupts are incurred while the processor is seeking data from within its current procedural environment. Recovery is achieved by re-executing the operator upon return from the "P-bit" interrupt-handling procedure.

The P-bit procedure makes the non-present reference present prior to returning to the interrupted program. The PIR and PSR setting for the current operator are saved in the RCW for data-dependent presence-bit interrupts.

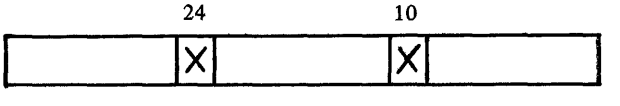
Procedure-Dependent Presence Bit. The Procedure-Dependent Presence Bit Interrupts are incurred when the processor attempts to enter a new procedural environment or to return to an old procedure. These interrupts occur during display up-date and when trying to "digest" a non-present segment descriptor. Recovery is achieved by the exit operator mechanism after the P-bit procedure has made the referenced area present. The processor has not yet fetched the first operator of the new procedure when this presence bit interrupt occurs; therefore, the PIR and PSR settings from the PCW or RCW, depending on whether an entry or exit was being performed, are saved when fabricating the RCW upon entry into the P-bit interrupt procedure.

Program Restart. In order to restart some operators after a presence bit interrupt, it is necessary for the P-bit procedure to return either an IRW or D.D. The "RT-bit" in the presence bit I.D. (P1) indicates to the P-bit procedure whether to perform an exit or return operator when returning to the interrupt program. The "RT-bit" is manipulated by the hardware prior to honoring the presence bit interrupt. Figure 5-6 (Presence Bit Interrupt Table) illustrates the (PSR, PIR), exit/return and "RT-bit" relationship to the various presence bit interrupt conditions.

Segmented Array.
This interrupt is used by the string operators as an upper limit boundary detection. Arrays in main memory may be segmented into groups of 256 words each, bounded on both ends by memory link words. Each word read from memory during string operator executions is checked for the presence of bit 48 (memory protect). If the bit is on, the segmented-array interrupt is set. String operator interrupts leave a special parameter in the A register. This indicates how many words in the stack, below the parameter, will be needed to restart the operation after the new segment of data has been brought to main memory.



A Register Parameter



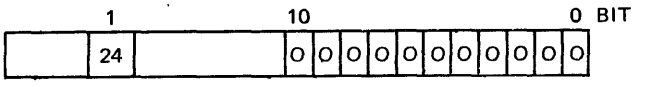
Segmented Array Interrupt ID

Programed Operator.
This interrupt is used as detection for invalid operator codes. Primary codes BC, E7, EF, F6, and F7 are detected and cause the interrupt. Each family controller detects these codes. Any invalid code not detectable will result in a loop timer interrupt. The programed operator interrupts are used as communicate operators to the system.

Presence Bit Interrupt Condition				RT Bit (3) (bit 46)	Returning Operator	PIR,PSR New RCW	Software Function
Data Dependent	Stack Vector Stack Vector D.D. during data reference	(5) DESC (1) IRW (stuffed)	Int. I.D.	0	Exit	S_n (4)	Locate not present D.D. via the IRW, make D.D. present, return I. W where noted.
		(2) IRW (stuffed)	Int.	1	Return	S_n (4)	
	Data Descriptor during data reference	(1) D.D. (copy)	Int. I.D.	0	Exit	S_n (4)	Search stack for copies of not present D.D., make Mom and copies present, return D.D. where noted.
		(2) D.D. (copy)	Int. I.D.	1	Return	S_n (4)	
Procedure Dependent	Stack Vector Stack Vector D.D. during display update	- D.D. (copy)	Int. I.D.	0	Exit	From RCW/PCW	
	Segment Descriptor	- S.D. (copy)	Int. I.D.	0	Exit	From RCW/PCW	Locate S.D. (Mom) via copy in P_2, AD Field of Copy points to Mom.

(1) Value Call or Enter
(2) All operators except Value Call, Enter, Move Stack, and Branch Return
(3) RT bit is packed in the Int. I.D. (P_1)
(4) Sn indicates the PIR and PSR point to current operator syllable
(5) Move Stack or Branch Return operators

Figure 5-6. Presence Bit Interrupt



Programed Operator Interrupt ID

Operator Independent Interrupts.

These interrupts are induced by conditions outside the operator or processor logic. They are divided into two groups, External Interrupts and Alarm Interrupts.

External Interrupts.

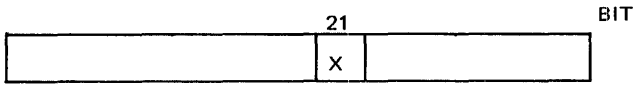
These interrupt conditions are anticipated and inform the system of some change in the external environment. They normally result in a momentary interruption of a program process which will be continued after handling or recording the interrupt condition. The external interrupts are recognized by the hardware operators. The program sequence controller senses the interrupt condition, inhibits activation of the next operator, and initiates an interrupt pseudo-operator in its place. PIR and PSR fields of the RCW address the next operator syllable so that the program will be restarted with the execution of the next syllable upon continuation. The external interrupts are:

a. Processor to Processor interrupt

b. Special Control interrupts
   1) Interval timer
   2) Stack overflow

c. Multiplexor interrupts
   1) I/O finish
   2) Data Communications
   3) General Control Adapter
   4) Change of Peripheral Status

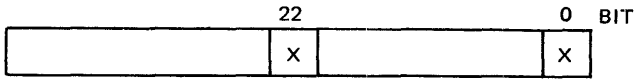
Processor to Processor.
This interrupt is used to interrupt another Processor on the system. When a Processor executes a HEYU operator, an external interrupt is sent to all other system processors. When the interrupt is recognized by a Processor, its interrupt controller clears the A register and sets the B register equal to the ID. The normal Interrupt Procedure entry is then executed.



Processor to Processor Interrupt ID

This interrupt is also used to initiate an Idle Processor on the system. It could also cause another Processor to suspend its operation on a program whose stack is about to be overlayed.

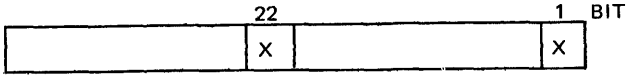
Interval Timer.
This interrupt is used for programmatic time slicing. The interval timer is activated by the SINT (Set Interval Timer) operator. The timer is set to the value of bits 10:11 of the B register and decrements every 512 microseconds until equal to zero. At this time, if the timer is still armed, the interrupt is set, leaving the ID in the B register and A register cleared. The maximum interval is 1 second. The timer is disarmed whenever the Processor handles an External interrupt.



Interval Timer Interrupt ID

Stack Overflow.
This interrupt is used to inform the operating system that the Stack Controller has sensed the use of the highest address allotted for this program's stack (LOSR, limit of stack register). The program is halted to allow the Operating system the option of allocating a

larger stack area or aborting the program. The interrupt controller leaves the A register cleared, the interrupt ID in the B register and PIR backed up if PROF is on.



Stack Overflow Interrupt ID

Multiplexor Interrupts.
The MPX interrupts may be handled by any system processor. A priority is established between multiplexors and processors to determine which Processor responds when an interrupt is present. This is necessary when multiple Processors and Multiplexors are present because they all share a common SCAN BUS.

Scan Bus Control.
Scan bus control is established by a closed loop circuit in which a control "bit" is passed from one Processor to another on every 3rd clock pulse.

A Processor may initiate a scan-bus operation when it has the control bit and the IIHF (inhibit external interrupt flip flop) is off.

Priority Handling Example.
Assume MPX-A and MPX-B have I/O finished interrupts occuring at the same time. Both Processors are operating with IIHF off and could therefore respond to an external interrupt. If both Processors were allowed to respond, a SCAN-IN of the interrupt literals would be attempted simultaneously on one common bus.

The 2nd priority established, is a left to right (LTRP), or right to left (RTLP) priority which allows a multiplexor to place its interrupt in the appropriate Processor. Figure 5-7 is a hypothetical system configuration that will be used for explanation.

Each of the Left-to-Right or Right-to-Left priorities are only true for one Processor at one time. LTRP is normally used to allow MPX-A to set its interrupt in Processor #1. RTLP is normally used to allow MPX-B to set its interrupt in Processor #2.

The priorities may be passed to another Processor when the IIHF is on. IIHF on in a Processor, causes the Priority to be passed and inhibits the interrupt controller from responding to any MPX interrupts. The priorities in a Processor are re-established when IIHF is reset.

Priority Handling With IIHF Set.

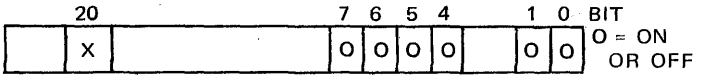
Assume Processor #1 had its IIHF set because it was in Control state. Setting this flip flop in Processor #1 causes the LTRP to be passed to #2. Now assume identical timed interrupts appear in both MPX-A and MPX-B. Both are recognized by the interrupt Controller in Processor #2. The interrupt controller in Processor #2 now assigned MPX-A the 1st priority and will subsequently SCAN-IN the interrupt literal from MPX-A while making MPX-B hold its interrupt line on. (The MPX interrupts are not reset until a SCAN-IN is performed.) The RTLP priority could also be passed to Processor #1 should it enter normal state while Processor #2 is in Control state, thus each system Processor is capable of handling external interrupts from either Multiplexor.

I/O Finished Data Communications.
Both interrupts are handled by the Interrupt Controller as follows:

a. A SCNI (SCAN-IN) operator is forced into the Processor at the next SECL to read the interrupt literal into the B Register.

b. An identification bit (20) is placed into the interrupt ID, the A register is cleared and PIR is backed up.

c. The normal operation of entry to the Interrupt Handling Procedure is then executed.



I/O Finished/Data Communications Interrupt ID

NOTE

Bits 1:2 identify which MPX the literal was read from.
MPX-A=01, MPX-B=10.

Bits 7:4 identify type of interrupt.
1001=I/O finished
0001=DCP #1
0010=DCP #2
0011=DCP #3
1111=Change of status

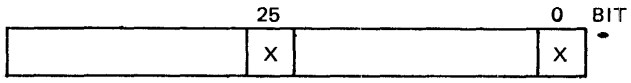
General Control Adapter.
This interrupt indicates a special control device such as an Analog device, a plotter, or some machine being controlled by the system wished to communicate to the Processor.

External MPX.
This interrupt will be used when a second Multiplexor is connected to one of the 4 word-interfaces of a Multiplexor, and it wishes to have one of its interrupts recognized.

Alarm Interrupts.
These interrupt conditions are not anticipated and inform the system of some detrimental change in environment. They normally result from either a programing error or hardware failure. The alarm interrupt conditions are recognized upon occurrence by the interrupt controller. The interrupt controller seizes control of the machine, clears the activated operator family, marks the TOS registers full and activates the pseudo interrupt operator. In either case the current operator is terminated prematurely. The alarm interrupts are:

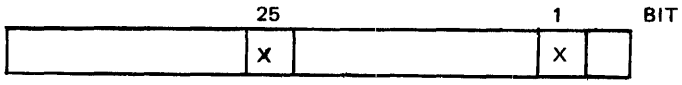
a. Loop
b. Memory Parity
c. MPX Parity
d. Invalid Address
e. Stack Underflow
f. Invalid Program Word

Loop.
This interrupt is invoked if the Processor hardware fails to provide a SECL (Syllable execute complete level) at least every 2 seconds. This could occur if an attempt is made to execute an invalid operator. Should the interrupt occur, the ID is left in the B register, the A register is cleared and PIR is backed up.



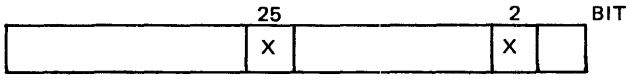
Loop Interrupt ID

Memory Parity.
This interrupt is invoked if the Memory Controller detects an even number of bits being transmitted between the Processor and Memory. Should the interrupt occur, the ID is left in the B register, the A register is cleared and PIR is backed up.



Memory Parity Interrupt ID

MPX Parity.
This interrupt is the same as Memory Parity except it is used for Processor/Multiplexor transfer.



MPX Parity Interrupt ID

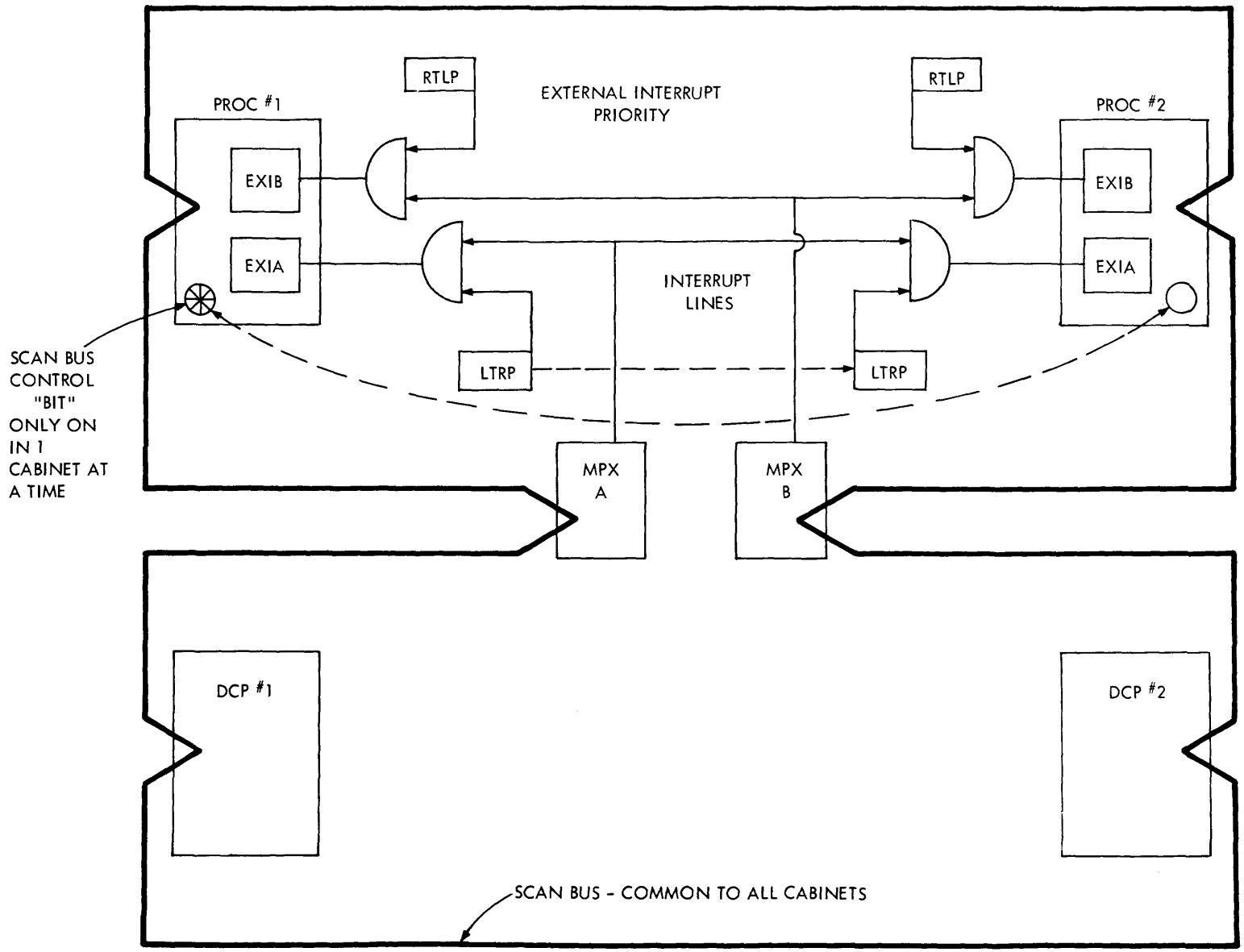
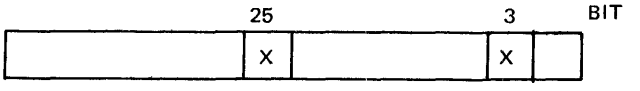


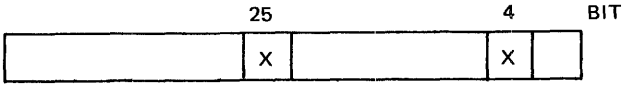
Figure 5-7. B 6500 Scan Bus Priority Control

Invalid Address.
This interrupt is set by the Memory Controller upon detecting an attempt to access a non-existent Memory module by a failure to obtain an acknowledgement to a memory request within 8 clock periods. The Memory Controller initiates the interrupt and the Interrupt Controller leaves the ID in the B register with the A register clear and PIR backed up.



Invalid Address Interrupt ID

Stack Underflow.
This interrupt is invoked if the Stack Controller detects an attempt to move the S register to an address less than BOSR (Bottom of Stack Register) during stack adjustment. Should the interrupt occur, the ID will be left in the B register, the A register is cleared and PIR backed up.



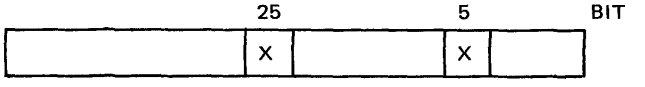
Stack Underflow Interrupt ID

Invalid Program Word.
This interrupt is invoked if one of the following conditions is encountered:

a. A word with a tag not equal to 3 is placed in the P register for execution. (Except in Table mode).

b. The Variant operator is decoded as the second part of a 2-syllable variant operator.

c. The Processor is in EDIT mode and a family strobe is emitted for another operator family. Should the interrupt occur, the ID is left in the B register, the A register is cleared and PIR is backed up.



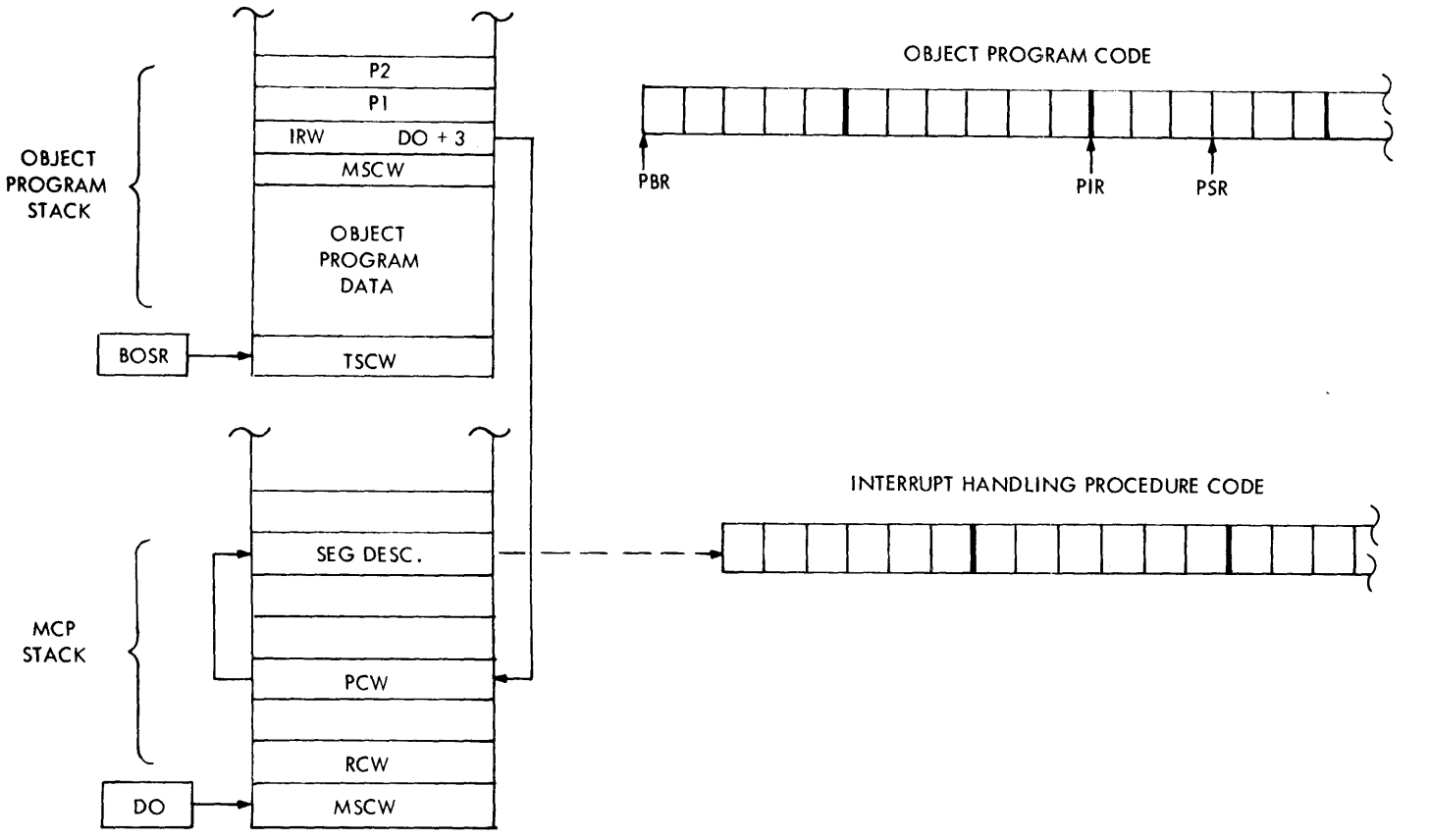
Invalid Program Word Interrupt ID

Interrupt Handling.
The occurrence of an interrupt condition causes the processor to enter an interrupt handling procedure after marking the stacks and inserting two interrupt parameters into the stack. The procedure entered is called from a reserved location (DO + 3), relative to the base (trunk) of the MCP stack. Figure 5-8 depicts the stack format just prior to and after entering the interrupt procedure.

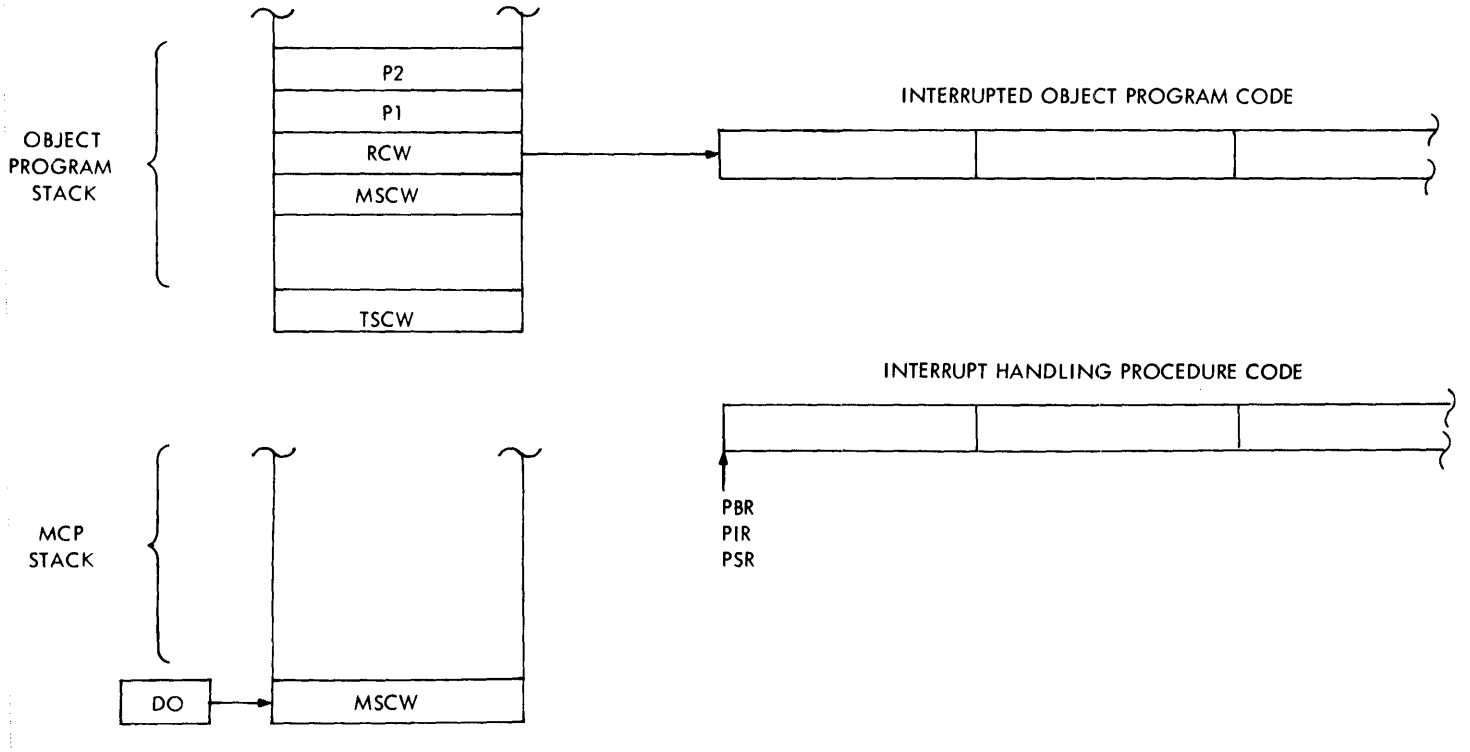
The two interrupt parameters P1 and P2 that are inserted into the stack as the interrupt condition is recognized are used to supply information describing the interrupt condition. The P1 parameter identifies the interrupt type and instructs the interrupt procedure how to return to the interrupted program. The P2 parameter supplies supplementary information about the interrupt condition (e.g., in the case of some presence bit interrupts P2 is a copy of the non-present descriptor).

The interrupt procedure is entered by inducing an enter operator with an IRW pointing to DO + 3 at F + 1. The hardware expects to find a PCW at DO +3; however, an IRW or IRW chain pointing to a PCW are legitimate conditions.

STRING OPERATOR CONTROLLER. The String Controller controls the character handling operators. It is integrated with the F, G, and H family hardware (figure 5-9). This controller is unique in many ways. One of the ways is by having the E register initiate memory cycle requests via the memory controller, during logical stepping



STACK FORMAT PRIOR TO CALLING THE INTERRUPT PROCEDURE.



STACK FORMAT AFTER ENTERING THE INTERRUPT PROCEDURE

Figure 5-8. Stack Format

of the operator flow. This allows simultaneous logic flow with memory cycles, to accelerate the logic flow. The E register decoding is shown in figure 5-10.

The String OP Controller contains one OP code register for all three families. There are two sequence registers; the JF registers are used for the Family F sequence flow together with a sequence extension register KF. The JG registers are used for the Family G and H sequence flow together with a sequence extension register KG.

CONTROL STATE/NORMAL STATE. Any B 6500 Processor has the ability to perform in either Normal or Control state. The difference between the two states is the inhibiting of external interrupts while performing in control state as well as enabling a few privileged operators. The Normal Control State flip flop (NCSF) and Inhibit Interrupt flip flop (IIHF) are both set when operating in control state.

The Processor switches to control state upon entering a procedure via a control state program control word or by the execution of disable external interrupt operator. Likewise it switches to normal state when entering a procedure via a normal state program control word or by the execution of the enable external interrupt operator.

The Operators that are enabled in Control State are:

a. Set Interval Timer
b. Scan Out

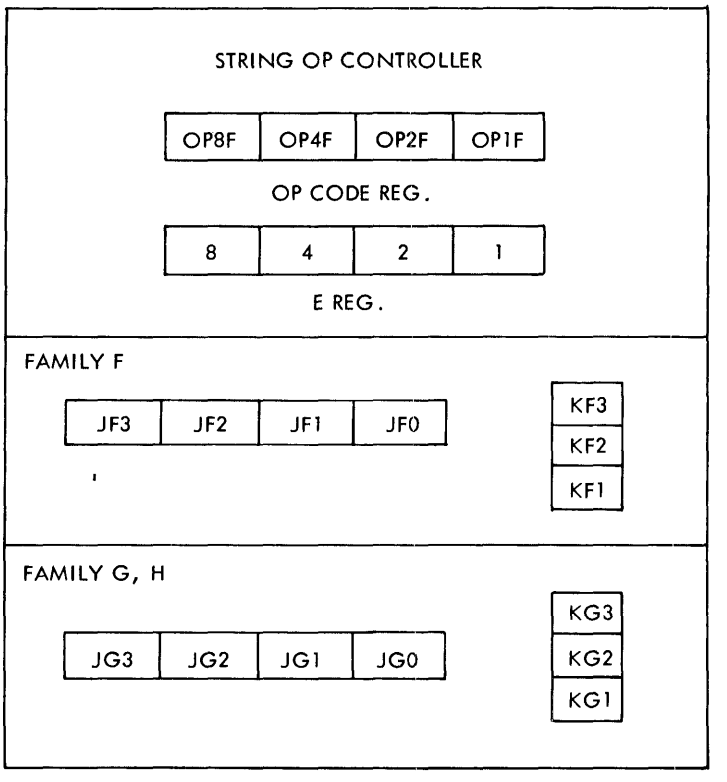


Figure 5-9. String OP Controller

E REG	FUNCTION	REG
1	READ	Y
2	"	B
3	"	C
4	"	X
5	"	A
9	WRITE-PROTECT	Y
10	"	B
11	"	C
12	"	X
13	"	A
14	OVER-WRITE	X
15	"	A

Figure 5-10. E Register Functions

INPUT/OUTPUT MULTIPLEXOR.

The Input/Output Multiplexor and associated peripheral control modules are used to control data transfers between memory and all peripheral equipment, independent of the processor. The multiplexor receives instructions from the processor and, together with its associated peripheral controls, executes them. Each multiplexor is capable of processing up to ten simultaneous I/O operations from up to 20 peripheral controls, handling a combined maximum of 256 peripheral devices (figure 5-11).

SCAN BUS.

The Scan Bus is the communications link between various components as seen in figure 5-11. It consists of 20 Address lines, 48 data Information lines, 1 Parity line and 11 Control lines. MPX or Data Communications operations are initiated via the Scan Bus.

COMMAND DATA REGISTER.

This 113 bit register is used with the Scratch Pad Memory for the control of Input Output data flow. The command portion of this register accepts an I/O Command from the Processor via the SCAN BUS and uses the data portion to accept or send information to the I/O devices via the peripheral control cabinets. Commands and partial data words are shuttled to and from the scratch pad memory between data character times. Full words are read or written to Main Memory without Processor intervention. An expanded Command Data word is shown in figure 5-12.

SCRATCH PAD MEMORY.

The Scratch Pad contains 120 bits of IC memory per word. The I/O MPX may contain from 4 thru 10 such words. These words provide temporary storage locations between command data word character collection times. In this way one Command Data register can service up to 10 simultaneous I/O operations. A fixed assignment (1 through 10) is given during the initiation of the I/O request and remains as such until the end of the I/O operation. The unit designate field as seen in figure 5-12 reflects this assignment.

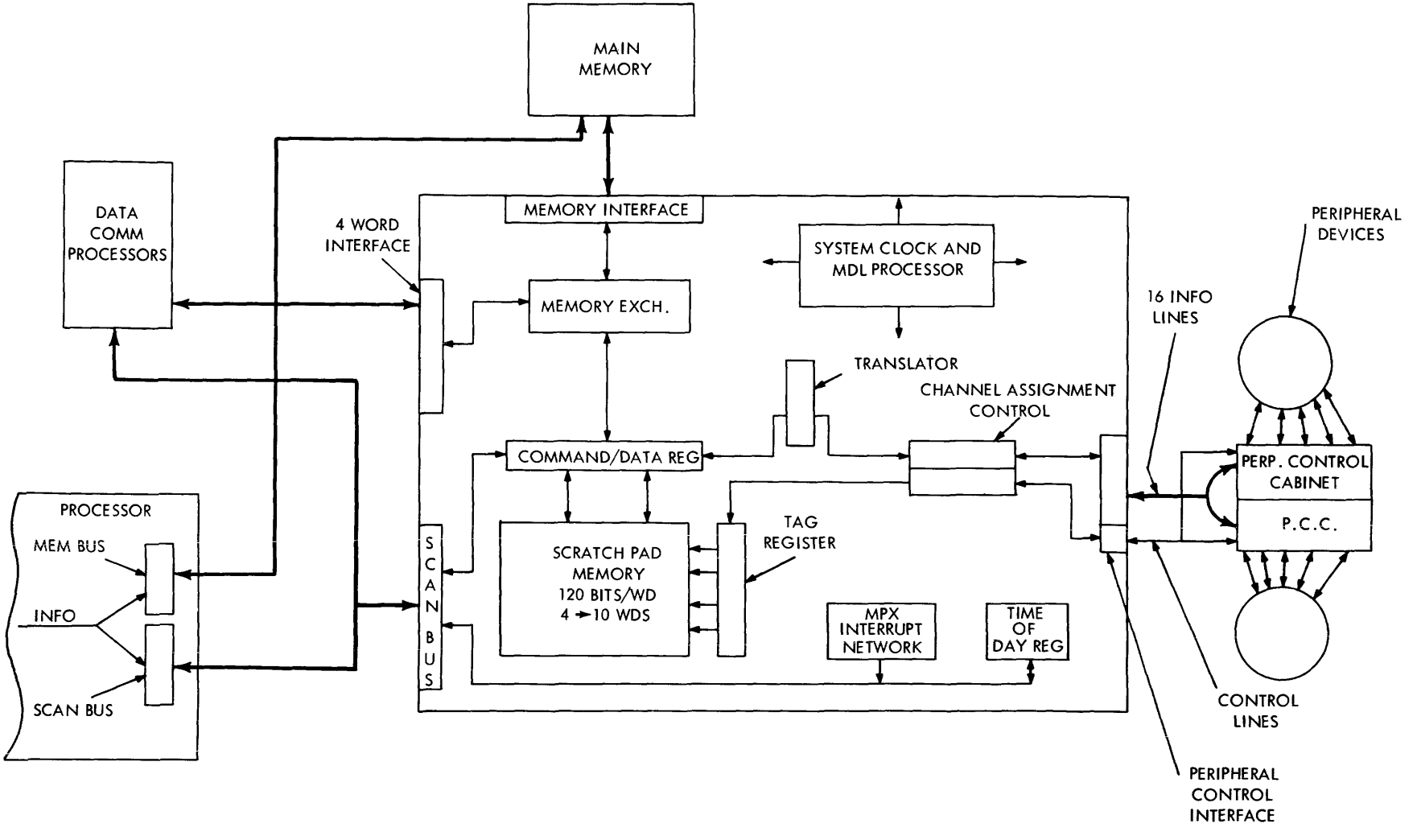


Figure 5-11. Multiplexor Block Diagram

TAG REGISTER.
The Tag Register (5 FF/SPM SLOT) associates a Scratch Pad Memory word with a specific I/O channel. This assignment is made when the initial I/O request is received from the Processor.

MEMORY EXCHANGE.
The Memory Exchange allows sharing of the Memory Interface lines between the MPX and Data Communications Processors. The Memory Exchange has 8 control lines, 20 address lines, 51 data lines and 1 parity line to the Memory interface.

INTERRUPT NETWORK.
The MPX Interrupt Network informs the Processors of an interrupt condition in the MPX. This indication remains true until one of the Processors reads the interrupt by a SCAN-IN command.

TIME OF DAY REGISTER.
The Time of Day Register is comprised of 36 flip flops used to accumulate increments (2.4 μsec) of time. The system Processors set or read these registers via the SCAN BUS.

CHANNEL ASSIGNMENT CONTROL.
The Channel Assignment Control assigns a priority to specific I/O devices. This is a fixed physical assignment as per system requirements.

CHARACTER TRANSLATOR.
Data flow between the MPC and Peripheral devices is translated in one of three ways:

a. Direct (no translation in the MPX)
b. 6 bit INTERNAL to BCL or vice versa
c. 8 bit EBCDIC to BCL or vice versa

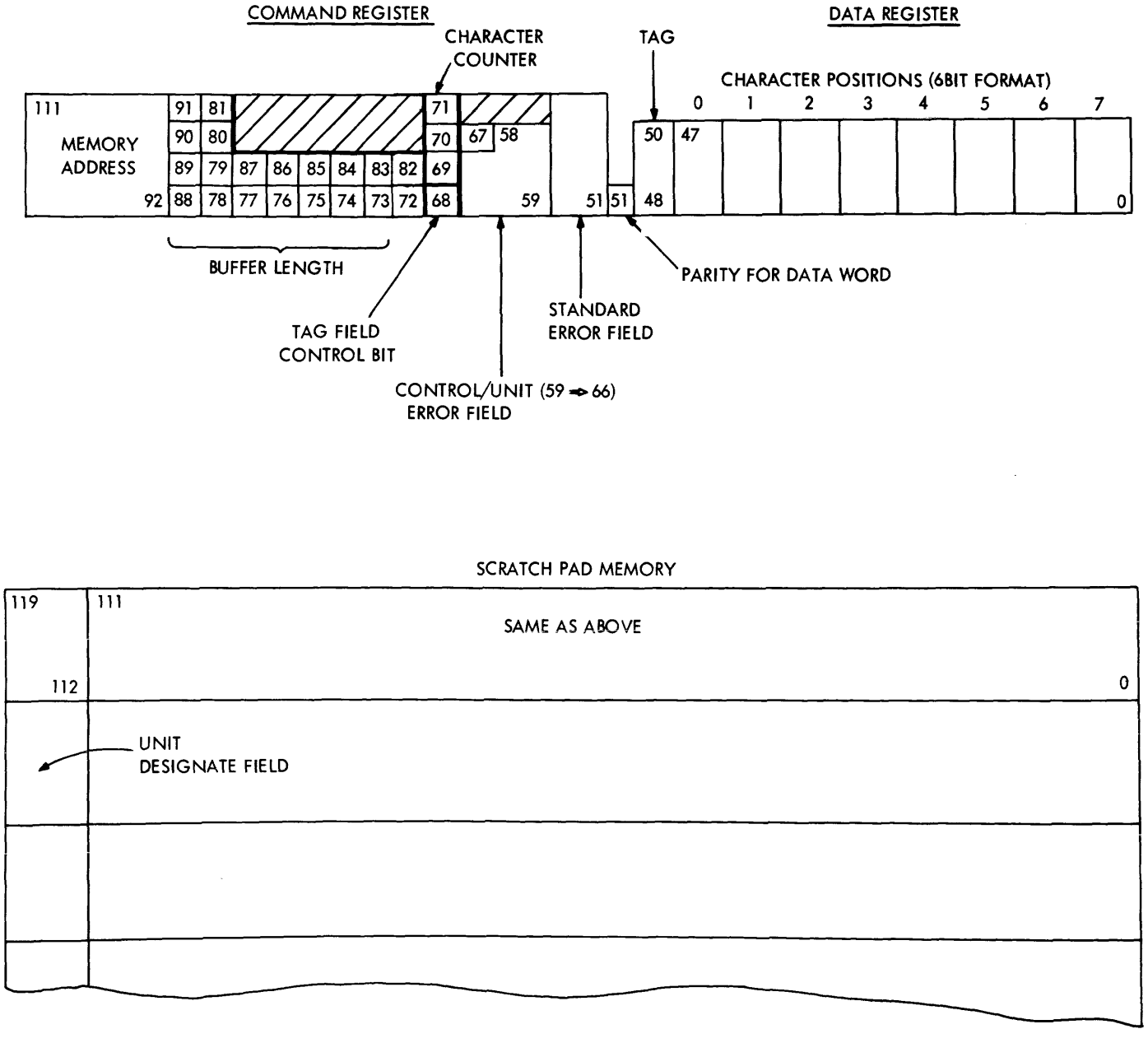


Figure 5-12. Command Data Register and Scratch Pad Memory

PERIPHERAL CONTROL INTERFACE.

The Peripheral Control Interface consists of 16 INFO lines and 12 Control lines which are bussed to all of the Peripheral controls. Four additional control lines are sent to each Peripheral Control for a total of 80. The additional control lines are:

a. BUSY/ - PCn
b. ARL - PCn (Access Request Level)
c. AGL - PCn (Access Granted Level)
d. CDL - PCn (Channel Designate Level)

The 16 info lines are used bi-directionally for 8-bit byte, or byte pair, transmission.

DATA COMMUNICATIONS INTERFACE.

The Data Comm Interface consists of 4, 20-wire cables sharing 2 word interfaces. Busses 2 and 4, 1 and 3 share the same memory request logic. Data Comm is routed through the MPX only to utilize the Memory Exchange of the Multiplexor.

SYSTEM CLOCK CONTROL AND MDL PROCESSOR.

The Multiplexor cabinet contains hardware that makes up the MDL Processor and System Clock.

SYSTEM CLOCK. The system clock is generated by a 10 megahertz crystal oscillator and shaped into 25 and 45 nanosecond width pulses. A Central Control divides and controls the basic clock for distribution to the entire system as follows:

a. Processor

Type	Basic Clock	Arithmetic Clock
B	5 megahertz	5 megahertz
C	2.5 megahertz	2.5 megahertz

b. I/O Multiplexors

5 megahertz 25 nanosec width

1.67 megahertz 25 nanosec width

c. Memory

5 megahertz 25 nanosec width

d. Peripheral Control

1.67 megahertz 45 nanosec width

e. Data Communications Processor

5 megahertz 25 nanosec width

MAINTENANCE DIAGNOSTIC PROCESSOR. The Maintenance Diagnostic Logic Processor (MDL) is a special purpose computer composed of an I/O Channel and a Data Processor. It is used for fault detection and isolation in the B 6500 Processor, B 6500 Multiplexors and the Peripheral Controls. The MDL Processor provides for three modes of operation: Display, Diagnose, and Detect.

Display Mode.

In this mode the MDL scan-out of eight flip flops per word progresses continuously in a loop under control of the display logic. It is used for indication and control of Processor and MPX flip flops.

Diagnose Mode.

In this mode the MDL Processor reads test cases from a tape unit, thru an I/O Channel, to memory. The MDL uses this information for logical testing of system components and halts at the end of a string of test cases when a failure is diagnosed.

Detect Mode.

This mode of operation is initiated in the same manner as diagnose mode; however, the test procedure is halted after the first failure of a test case.

INFORMATION FLOW FROM CARD READER TO MAIN MEMORY.

The information flow between a Card Reader and main memory is shown in figure 5-13. Three types cards may be read from the card reader.

ALPHA CARD READ.

Cards punched in the Alpha mode are decoded in the card reader from Card Code to 6-bit BCL EXTERNAL code. The character is transmitted to the information register in the Card Reader Control in the Peripheral Control Cabinet. The information (1 character) is held until the Multiplexor honors an access request and places the appropriate SPM word in its Command/Data register. I/O descriptor control bits 42 (translate) and 41 (6 or 8 bit) steer the character through the appropriate translator and place it in the next character position of the Data register. The data register can store 6 or 8 characters depending on the translator used. When the data register receives the last character of a word, a memory request cycle is initiated to write this full 52 bit word in memory. A tag field read is optional on this type of a card read, with any tag code (the first character of a word) allowable in this mode of operation.

BINARY CARD READ.

Cards punched in the binary mode contain twice as much information as those punched in Alpha mode. Each card column contains two characters. Positions 12, 11, 0, 1, 2 and 3 provide for one row of characters on the upper half while positions 4, 5, 6, 7, 8, and 9 provide for another row of characters on the lower half. Control bits 42 and 41 equal to zero bypasses the translator and causes direct transfer of information into the Data Register. The information contained in one card column is strobed twice (once for each half of the card) and presented to the multiplexor as two 6-bit characters. Tag read is optional in this mode but the only allowable code is Program tag (3).

EBCDIC CARD READ.

Cards punched in the EBCDIC mode are read in a similar fashion as binary mode, upper and lower half. However, the actions within the Peripheral Control are quite different. Three translations are required within the control before an 8 bit EBCDIC code is presented to the MPX data register. The first two occur as the upper and then lower halves of the card are strobed into the information register. The information register at this point represents the 12, 11, 0, 9 and 8 card punches directly and a binary configuration of punches 1 thru 7 as seen in figure 5-13. The information register is then decoded into EBCDIC code as it is presented to the information lines on its way to the Data register. When 6 bytes are collected in the data register, a memory request cycle is initiated to write the full 52 bit word. Tag read is optional in this mode with any tag code being permissable.

NOTE

Two other codes are available for use on the B 6500 system. They are ICT and BULL codes. Both are decoded by a special Alpha/Binary decoder (in the Card Reader) to BCL code.

MEMORY AND MPX CONTROLLER.

The Memory Controller responds to 21 commands decoded from nine INPUT lines. Figure 5-14 shows the 4 types of Memory Controller cycles that respond to these INPUT lines. During a core memory write, the contents of the cell being written are "flashed" back to the Processor. Certain Write operations are aborted by the memory if the memory protect bit (48) is on.

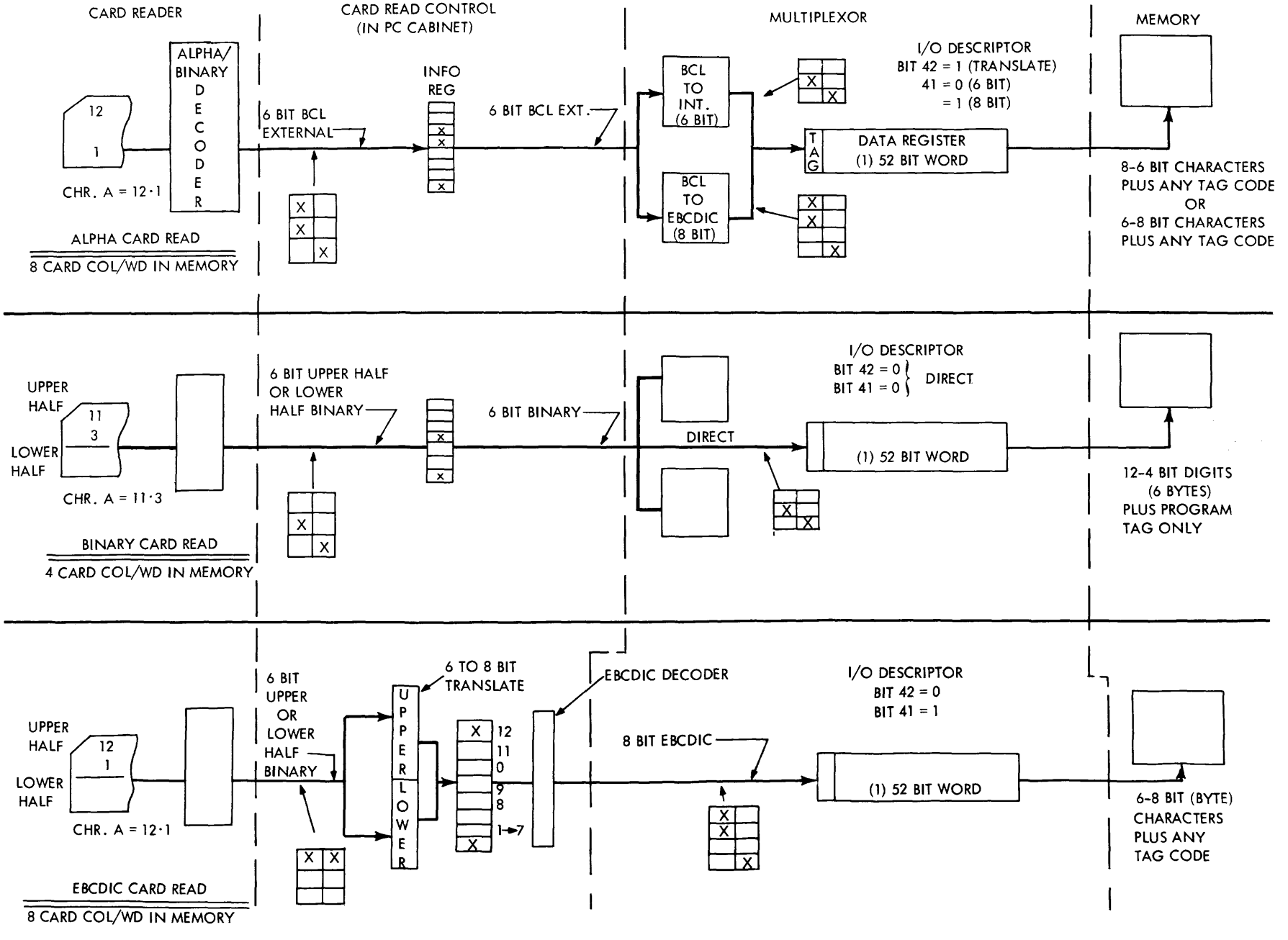
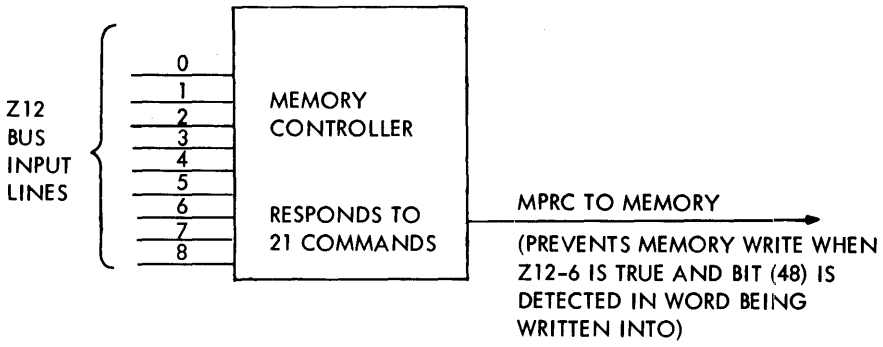


Figure 5-13. Data Information Flow



TYPE OF REQUESTING OPERATOR	MEMORY CONTROLLER FUNCTION	MEMORY CONTROLLER Z 12 LEVELS									PROCESSOR REGISTERS USED
		8	7	6	5	4	3	2	1	0	
READ	READ ONLY	1	0	0	1						A
		1	0	0		1					B
		1	0	0			1				C
		1	0	0				1			X
		1	0	0					1		Y
		1	0	0						1	P
OVERWRITE, STACK ADJ., READ WITH LOCK	OVERWRITE *	1			1						A
		1				1					B
		1					1				C
		1						1			X
		1							1		Y

NOTE

When the Overwrite function is used the Memory write is not aborted if the addressed area has the protect bit on.

The Read With Lock operator exchanges the contents of the A register with the contents of memory addressed by the B register.

Figure 5-14. Memory Controller Decoding

PROTECTED WRITE (PSEUDO)	PROTECTED** WRITE	1 1 1 1 1 1 1 1 1 1	1 1 1 1 1	A B C X Y

NOTE

When this function is used Memory write is aborted by detection of Protect bit. (no indication of abort is given).

STORE OPERATORS	PROTECTED WRITE/READ ***	1 1 1 1 1	1 1 1 1 1	A B C X Y

Figure 5-14. Memory Controller Decoding (cont)

The Memory/MPX Controller contains the following sections:

a. B 6500 Memory and MPX interface.
b. Address Adder.
c. Integrated Chip Memory.

The interface consists of two sections: a memory bus and a scan bus.

MEMORY BUS. The MEMORY BUS contains 20 address lines, 51 data (information) lines, 1 parity line and 8 control lines. It transmits information bi-directionally between MEMORY and Processor "hard registers" A, B, C, X, Y and P.

Control of the memory interface is thru the Z12 bus which is produced by FUNCTIONAL CONTROLLERS and FAMILY OPERATOR CONTROLLERS

when a memory cycle is desired.

SCAN BUS. The SCAN BUS contains 20 address lines, 48 data information lines, 1 parity line and 11 control lines. It provides an asynchronous communication path between the B 6500 Processors and B 6500 Multiplexors or B 6500 Data Communication Processors.

ADDRESS ADDER.
The Address Adder is a 20-bit parallel adder with inputs from the Z8 and Z9 busses, the Carry flip flop and the Subtract flip flop. The busses derive their addressing information from the 48 IC memories or from the "hard registers" via the Z6 bus in the transfer controller. The Carry flip flop and Subtract flip flop are used to modify the output address.

The output of the Address Adder is an input to the Memory Address register for memory selection or an input to one of the 20 bit IC memories.

INTEGRATED CHIP MEMORY.
The Memory Controller contains 48 IC memories, each containing 20 bits. Thirty-two of these display the current address of an object program. These D registers (D0 thru D31) provide for multiple levels of addressing. The D registers are controlled by Display READ/WRITE SELECT logic.

The other 16 IC memories are divided into two groups, base and index (0 thru 7). Each is a 20-bit memory used by Family Operator logic and Program sequence flow for base and index addressing:

a.	PBR	(0)	PROGRAM BASE
b.	SBR	(1)	SOURCE BASE
c.	DBR	(2)	DESTINATION BASE
d.	TBR (BUF2)	(3)	TABLE BASE
e.	S	(4)	TOP OF STACK ADDRESS
f.	SNR	(5)	STACK NUMBER
g.	PDR	(6)	PROGRAM DICTIONARY INDEX
h.	TEMP	(7)	TEMPORARY STORAGE

i.	PIR	(0)	PROGRAM INDEX
j.	SIR	(1)	SOURCE INDEX
k.	DIR	(2)	DESTINATION INDEX
l.	TIR (BUF3)	(3)	TABLE INDEX
m.	LOSR	(4)	LIMIT OF STACK
n.	BOSR	(5)	BASE OF STACK
o.	F	(6)	POINTS TO TOP MSCW
p.	BUF	(7)	TEMPORARY STORAGE

MAIN MEMORY.

ORGANIZATION.

Main memory in the B 6500 is organized so that any memory module can send information to, or receive information from both processors and both I/O multiplexors over any one of four information busses (see figure 5-15).

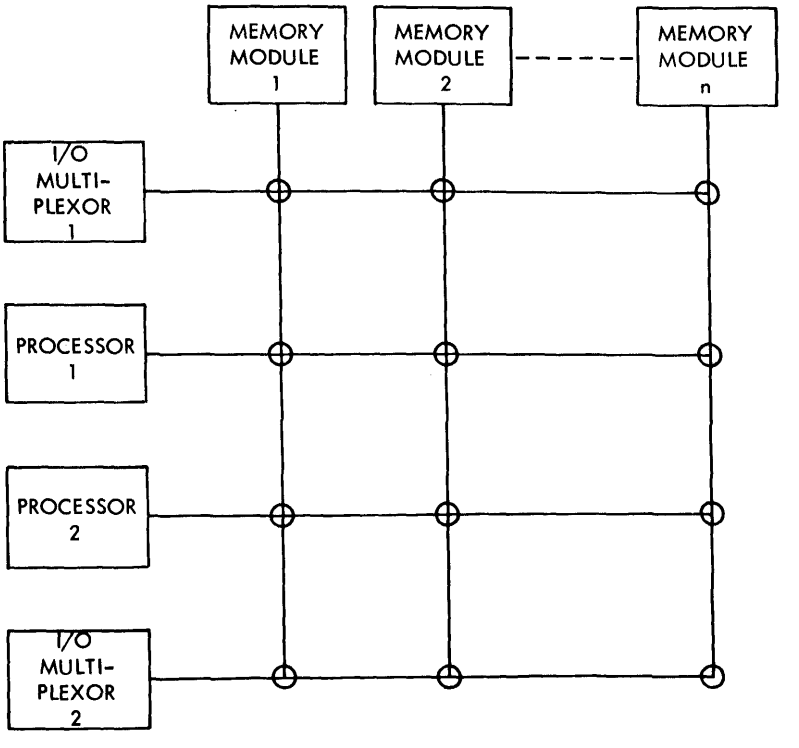


Figure 5-15. Memory Organization

The modules examine each word that is placed on the bus to determine whether that particular module is being addressed; if it is, linkage is set to receive the word. This eliminates the need for a central control to establish a linkage directing the word to the proper module. Two hundred nanoseconds after the memory cycle is initiated, the module grants access. In another 200 nanoseconds, the word is available to the bus, and 200 nanoseconds later the word is in the processor or I/O multiplexor register. Operation of each memory module is independent of the operation of any other memory module. Memory cycles can occur simultaneously within all four modules.

Information is transmitted along the bus in parallel, as illustrated in figure 5-16.

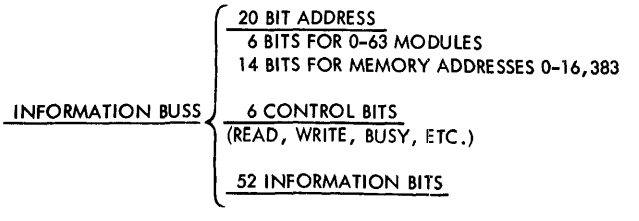


Figure 5-16. Information Transmission

MEMORY PROTECTION.

Memory protection prevents one program from affecting another with a combination of hardware and software features. One of the hardware features is automatic detection of an attempt by a program to index beyond its assigned data area. Another is a memory protect bit in each word to prevent user programs from writing into memory words which have the protect bit set. (The protect bit is set by the software.) Any attempt to alter protected data is inhibited and an interrupt is generated. Thus a user program cannot change program segments, data descriptors, or any program words or MCP tables during execution.

CABINET CONFIGURATION.

The B 6500 Main Memory consists of 1 to 32 memory modules containing 16,384 words each. Up to three modules and associated hardware can be housed in one Memory Cabinet (49,152 words). Each cabinet has a memory controller which responds to six requestors for memory accesses. The requestors are:

a. Processor #1 or #2
b. Multiplexors A or B
c. Memory Testor
d. MDL Processor

INTERFACE.

The requesting unit's memory interface contains five hubs (except for the MDL Processor). Each hub has 80 bus lines for bi-directional communication with memory. Each memory cabinet has six hubs, one hub for each possible requestor. A typical maximum size system is shown in figure 5-17. Notice how the hubs within the requestors are all tied to the same address and information flow lines. Take the example of a Processor requesting access to Memory module zero in cabinet zero. The Processor places the address and information on the busses. It is seen by all of the memory controls, but only accepted by module zero because of the address decoding in Memory Cabinet zero. This means that each Memory Control must have the ability to accept addresses from six different requestors and connect them to one of three memory modules. This is accomplished by a crosspoint control located within the memory control (figure 5-18). There are three sets of crosspoint controls for each requestor within each memory control. Three requestors may gain simultaneous access to the same memory cabinet if they are addressing separate memory modules.

PRIORITY.

A priority system, which is activated prior to the crosspoint controls, prevents conflicts when more than one requestor is addressing the same memory module.

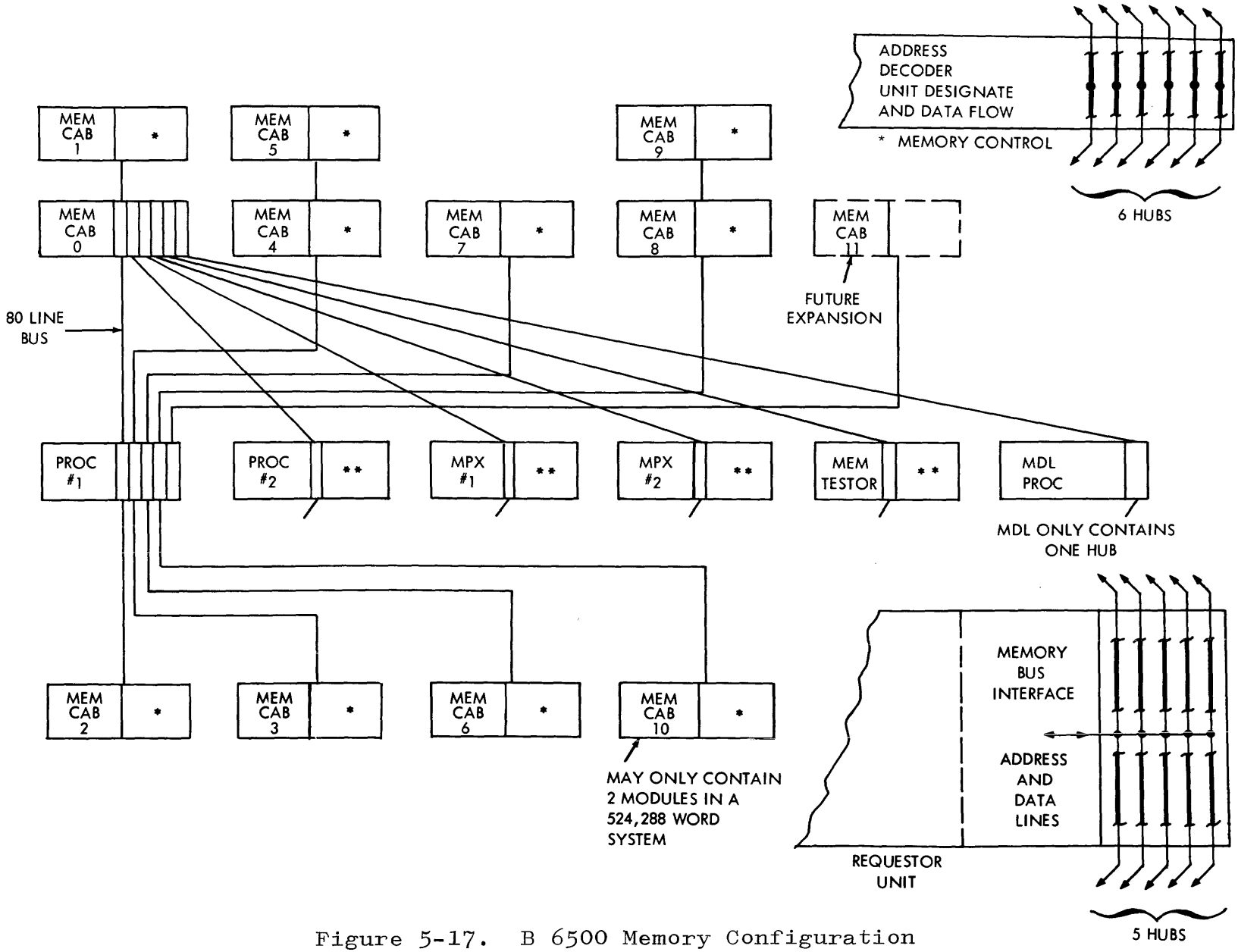


Figure 5-17. B 6500 Memory Configuration

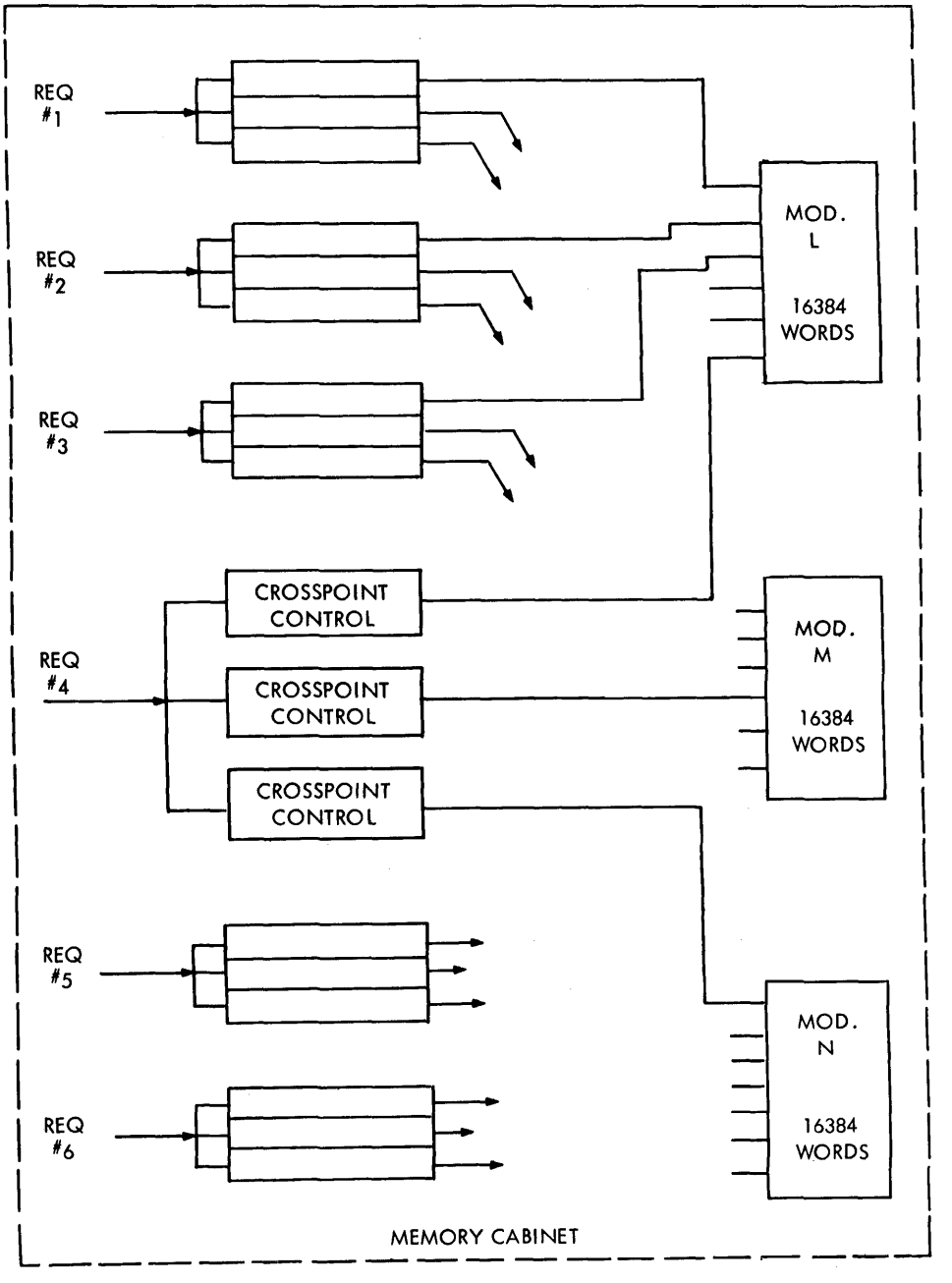


Figure 5-18. Memory Module Selection

Request hub #1 has the highest priority and any of the six requesting units can be attached to this point by the Field Engineer.

MEMORY REGISTERS.

Each Memory module contains 2 core stacks, a MIR (a 52-bit memory information register), and the appropriate timing and control logic necessary for reading and writing (figure 5-19). The memory cycle is divided into two parts, a destructive read where the information is read into the MIR's, and a write into the cores from the MWR's. The MWR's are loaded from one of the six requesters. When a memory protect bit (48) is on during the read portion of the cycle, and the operation is not overwrite, the information is rewritten from the MIR's.

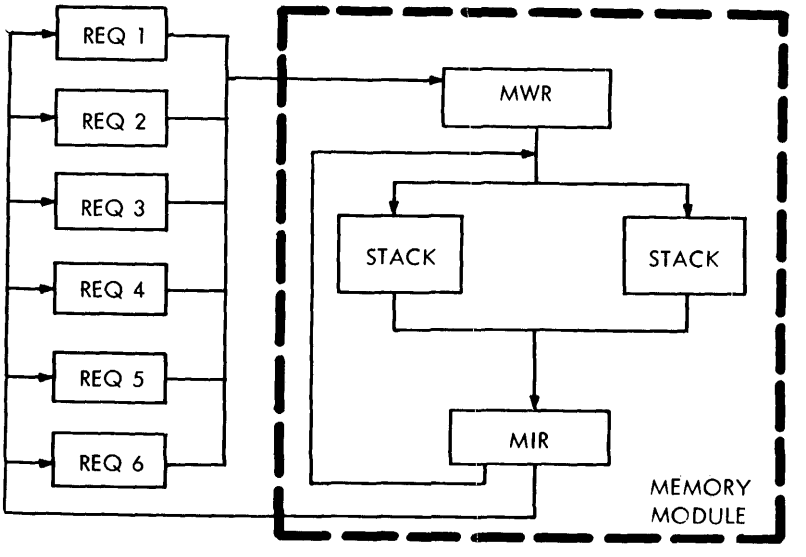


Figure 5-19. Memory Registers

MEMORY ADDRESSING.

Memory modules are addressed by 20 bits (figure 5-20). Bits 0 thru 13 are used for word selection and bits 14 thru 19 are used for module selection.

MEMORY INTERLACING.

Each memory module has the ability to interlace every other word to the next consecutive module. Interlacing is controlled by a pluggable jumper located on each module and provides the advantage of faster memory accesses when consecutive words are addressed.

Interlacing saves time because the next consecutive access may be requested in an adjacent module while the first module is finishing its cycle. Bit 14 of the module select address is exchanged with bit zero when interlacing is used. Examples of module and word selection when using the interlace option are shown in figure 5-20. This feature can be quickly enabled or disabled by a field engineer.

HEXADECIMAL ADDRESS	INTERLACE ADDRESS	MODULE	WORD
00000	00000	0	0
00001	04000	1	0
04000	00001	0	1
04001	04001	1	1
08000	08000	2	0
08001	0C000	3	0
0C000	08001	2	1
0C000	0C000	3	1
10000	10000	4	0
10001	14000	5	0

MODULE SELECT		WORD SELECT		
19	15	11	7	3
18	14	10	6	2
17	13	9	5	1
16	12	8	4	0

Figure 5-20. Interlace Addressing

MEMORY TESTING.

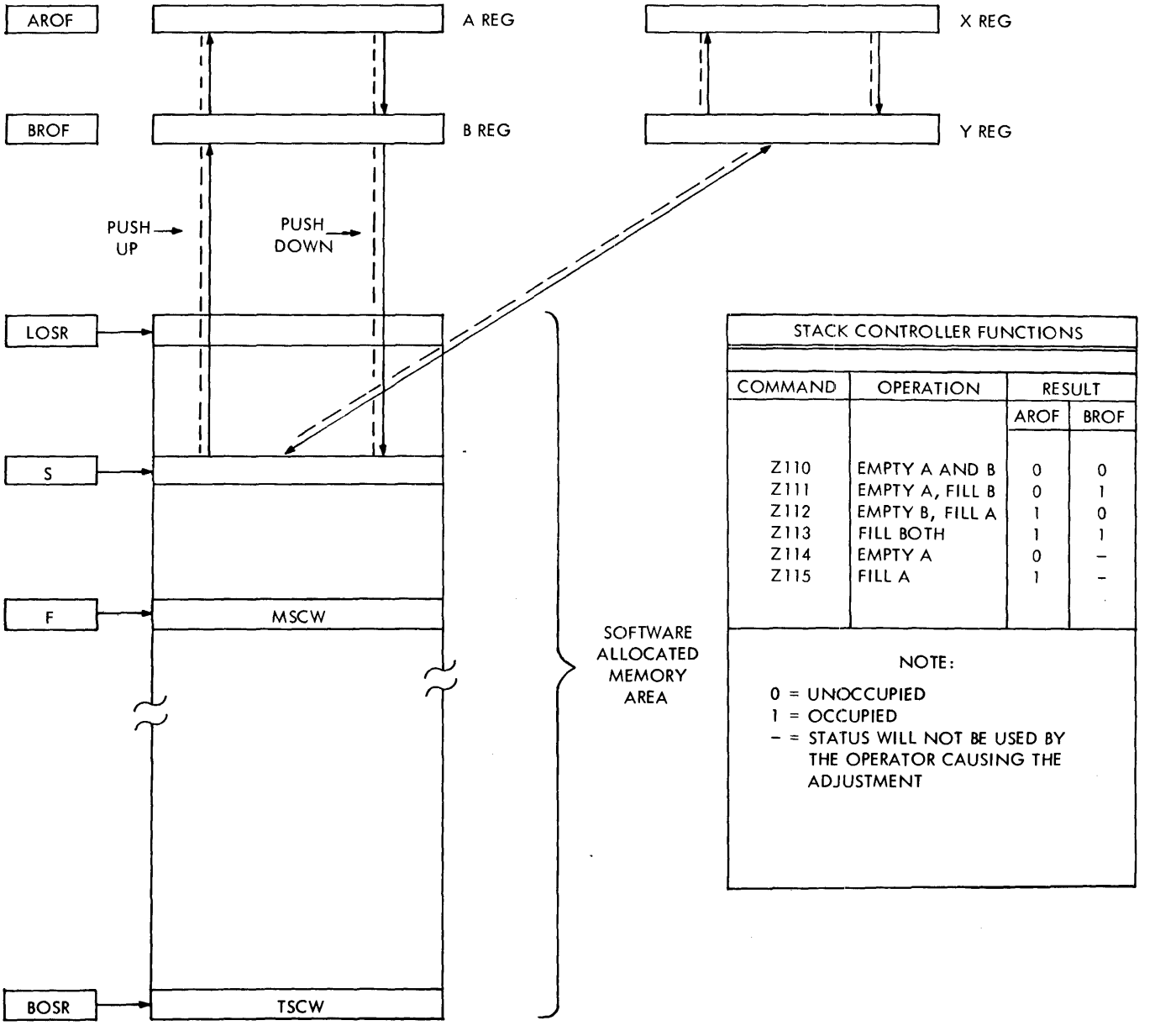
Each system includes a test facility which can exercise any of the memory modules. When the test facility is being used with one of the memory modules, the other modules can be used by the system, if the module being tested is not interlaced. If it is, the option must be disabled before testing can take place.

STACK CONTROLLER.

The B 6500 provides automatic stack adjustment as required by the operators. These requirements are supplied to the Stack Controller on the Z11 bus from the Operator Families and other Functional Controllers.

The Stack Controller manipulates data between Main Memory and the A and B registers on both pop-up and push-down cycles. The X and Y registers are included in the adjustment cycles when double-precision operands are involved.

A typical program stack is shown in figure 5-21. The Stack Controller determines whether a pop-up or push-down cycle will be initiated. All other Controllers remain idle until an ADJC (Adjust complete) is sent to the Controller that initiated the adjustment.



STACK CONTROLLER FUNCTIONS

COMMAND	OPERATION	RESULT	
		AROF	BROF
Z110	EMPTY A AND B	0	0
Z111	EMPTY A, FILL B	0	1
Z112	EMPTY B, FILL A	1	0
Z113	FILL BOTH	1	1
Z114	EMPTY A	0	–
Z115	FILL A	1	–

NOTE:

0 = UNOCCUPIED
1 = OCCUPIED
– = STATUS WILL NOT BE USED BY THE OPERATOR CAUSING THE ADJUSTMENT

Figure 5-21. Hardware Stack Adjustment

SECTION 6
PROGRAM OPERATORS

GENERAL.

The machine language operators are composed of syllables in a program string. The operators are divided into three major classes, Primary, Variant and Edit: The operators are either Primary Mode, Variant Mode, or Edit Mode.

SYLLABLE ADDRESSING AND SYLLABLE IDENTIFICATION.

SYLLABLE FORMAT AND ADDRESSING.

A machine language program is a string of syllables which are normally executed sequentially. Each word in memory contains six 8-bit syllables. The first syllable of a program word is labeled syllable 0 and is formed by bits 47 thru 40 (figure 6-1).

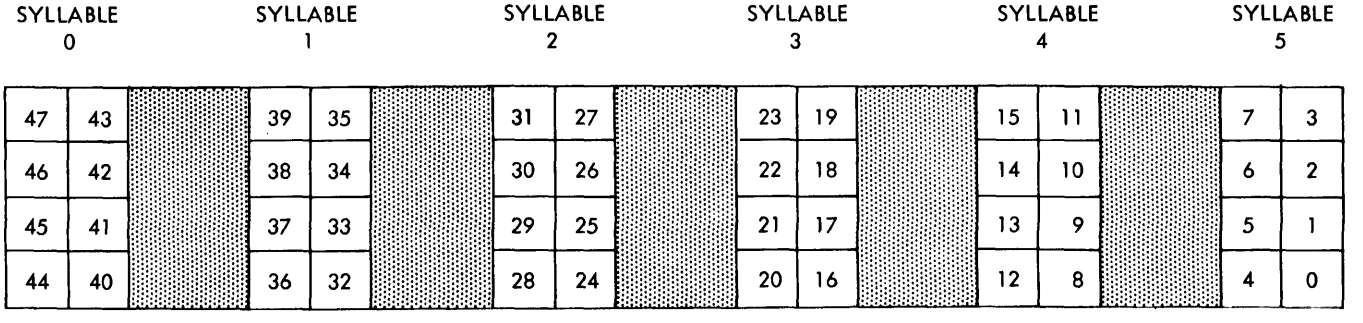
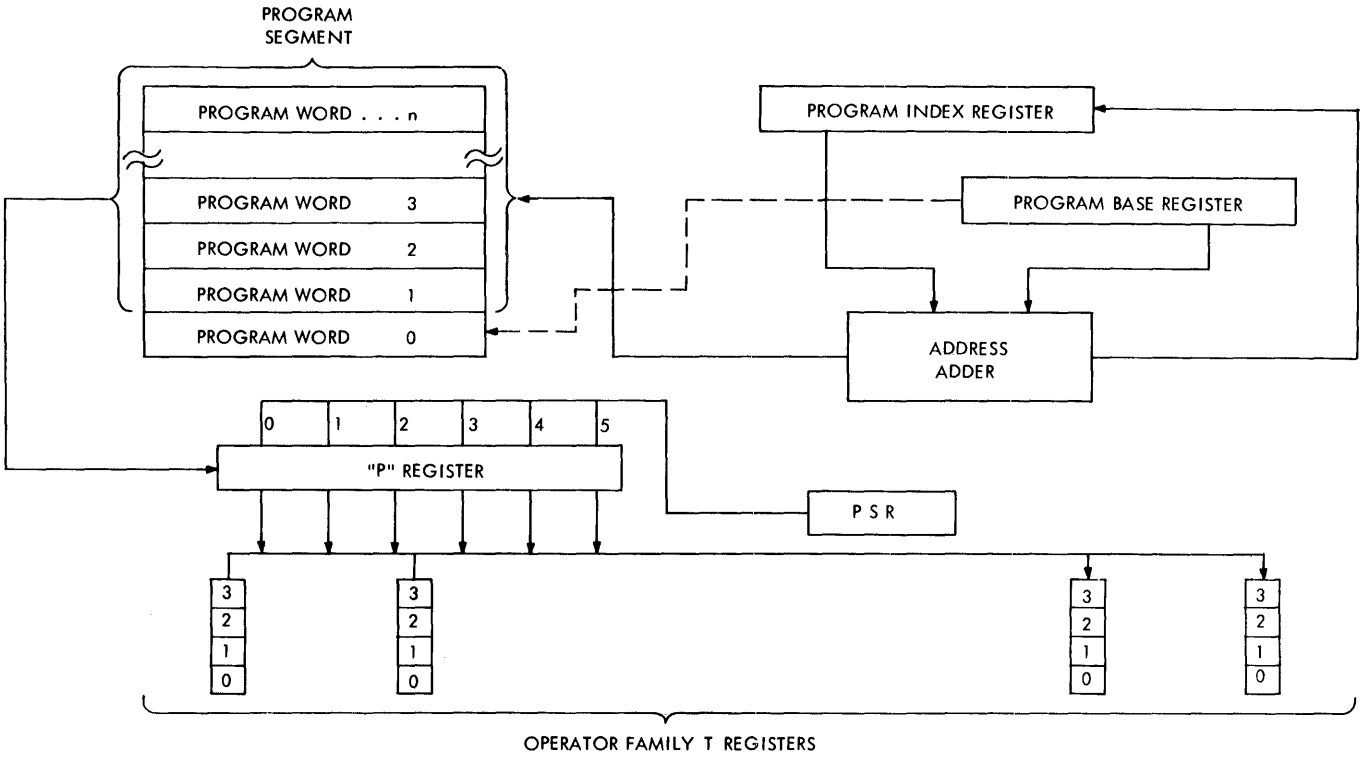


Figure 6-1. Program Word

P AND T REGISTERS.

The P Register contains the currently active program word. The T Registers are the control (instruction) registers. There is one four-bit T register in each operator family. These registers contain the operation to be executed in a particular operator family. The four high-order bits of the operator syllable are decoded to select the operator family to receive the strobe pulse, (execute pulse). The PSR (Program Syllable Register) points to the next syllable to be used and also determines when a new program word is required in the P register.

When a new program word is required it is brought from the memory location indicated by the sum of PBR (Program Base Register) and PIR (Program Index Register). This program word is placed in the P register and PSR is set to the first syllable of the next operator. PIR is incremented by 1 to address the next required program word (figure 6-2).



OPERATOR FAMILY "T" REGISTERS

Figure 6-2. Program Word, Syllable Addressing

OPERATION TYPES.

Operations are grouped into 3 classes: Name call, Value Call, and operators. The two high-order bits (bits 7 and 6) determine whether a syllable begins a Value Call, Name Call or operator (figure 6-3).

(BITS 7 & 6) Identification	Syllable Type	# of Syllables	Function
00	Value Call	2	Brings an operand into the stack.
01	Name Call	2	Brings an IRW into the stack.
1X	Other Operators	1 => 12	Performs the specified operation.

Figure 6-3. Syllable Decode Table

NAME CALL. Name Call builds an Indirect Reference Word in the stack. Stack adjustment takes place so that the "A" register is empty. The six low-order bits of the first syllable of this operator are concatenated with the eight bits of the following syllable to form a 14-bit address couple. The address couple is placed, right-justified, into the "A" register, with the remainder of the "A" register set to zero. The TAG field of the "A" register is set to 001 and the register is marked full.

VALUE CALL. Value Call loads into the top of the stack the operand referenced by the address couple formed in the same manner as in the Name Call operator. If the referenced Memory Location is an Indirect Reference Word or a Data Descriptor, additional memory accesses are made until the operand is located. The operand is then placed in the top of stack registers. The operand may be either single or double-precision, causing either one or two words to be loaded into the stack.

OPERATORS. Operators vary from 1 to 12 syllables in length. The first syllable of each operator determines the number of additional syllables forming the operator. Upon completion of each operator, the program counter addresses the first syllable beyond all of the syllables comprising the operator.

Operators work on data as either full words (48) data bits plus tag bits) or as strings of data characters. Word operators work with operands (single or double-precision) in the top of the stack.

String operators are used for transferring, comparing, scanning, and translating strings of digits, characters, or bytes. In addition, a set of micro-operators provides a means of formating data for input/output.

The string operators use source and destination pointers which are located in the stack. These pointers set the following hardware registers:

a. Source Base Register - (SBR).
b. Source Word Index Register - (SIR).
c. Source Byte Index Register - (SIB).
d. Source Size Register - (SSZ).
e. Destination Base Register - (DBR).
f. Destination Word Index Register - (DIR).
g. Destination Byte Index Register - (DIB).
h. Destination Size Register - (DSZ).

In some of the string operators the source pointer may not be used. In this case, an operand may be in the stack; its characters are circulated as it is being used.

String operators have an optional Update function, producing updated source and destination pointers and count. At completion of an operation the source and destination pointers are updated as follows:

a. If the source is an operand it is left in the stack.

b. If the pointer is a descriptor, the Word Index fields and Byte Index fields are updated from SIR/DIR and SIB/DIB. The String Size fields are updated from SSZ/DSZ.

c. If the pointer is a Data Descriptor or a non-indexed String Descriptor, it is converted to an Indexed String Descriptor and updated.

If both the source and destination descriptors have size fields equal to zero, the size registers indicate 8-bit character size. When both a source and destination are required and the size field of one is equal to zero and the other is not, then the size field of the non-zero descriptor is used.

If neither size field is equal to zero and the size fields are not equal and the operator is not Translate, the invalid operand interrupt is set and the operator is terminated. The size field is considered equal to zero when the source is an operand.

WORD DATA DESCRIPTOR.

Word Data descriptors refer to data areas, including input/output buffer areas. The Word data descriptor defines an area of memory starting at the base address contained in the descriptor. The size of the memory area in operands is contained in the length field of the descriptor. Word Data descriptors may directly reference any memory word address from zero through 1,0485,575 (current maximum is 524,288 words). The structure of the Word Data descriptor is illustrated in figure 6-4 and contains the following:

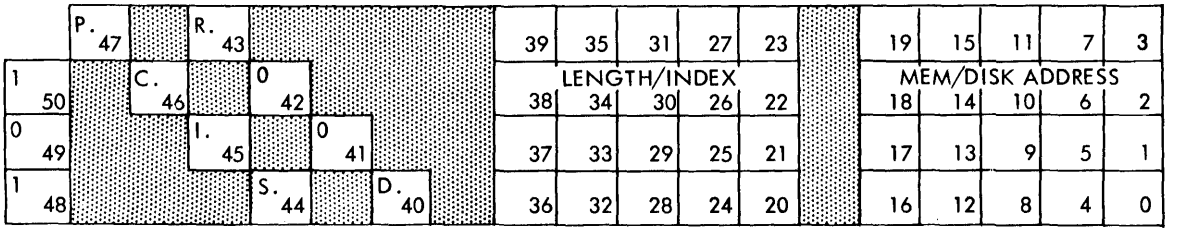


Figure 6-4. Word Data Descriptor

a. Bit 50:3, a tag of 101.

b. Bit 47:1, the presence bit, indicates the presence or absence of data in main memory. A zero causes a presence bit interrupt whenever the descriptor is used by a processor to obtain non-present data. A one indicates that the data described is in main memory.

c. Bit 46:1, the copy bit. A zero indicates that this is the original descriptor for the particular data area. A one indicates that this descriptor is a copy of the original descriptor.

d. Bit 45:1, the indexed bit. A zero indicates that an indexing operation is required before the descriptor may be used to obtain data. A one indicates that indexing has already taken place and the index value is stored in bit positions 39:20 (Length/Index).

e. Bit 44:1, the segmented bit. A zero indicates that the data is not segmented. A one indicates that the data is divided into segments.

f. Bit 43:1, the read-only bit. A zero indicates that the data may be referenced for reading or writing. A one indicates that the area cannot be used for data storage.

g. Bit 42:2, a zero indicates a word data descriptor.

h. Bit 40:1, the double-precision bit. A zero indicates single-precision operands, a one indicates double-precision operands.

i. Bit 39:20, contains either the length of the memory area (If bit 45 = 0) or an index value (if bit 45 = 1). If bit 45 equals zero, the descriptor has not been indexed. This field is used for size checking during the indexing operation. If bit 45 equals one, the descriptor has been indexed. For a double-precision operation, the index is doubled after index size checking, and the result is stored in the index field.

j. Bit 19:20, contains either a main memory or disk address. If the presence bit (bit 47) equals one, this field contains the memory address of data. If the presence bit equals zero and the copy bit (bit 46) equals zero, this field contains the disk address of the data. If the presence

bit equals zero and the copy bit equals one, this field contains the memory address of the original descriptor.

STRING DESCRIPTOR.

String Descriptors refer to strings of 4-bit digits, 6-bit characters or 8-bit bytes. The String Descriptor defines an area of memory starting at the base address contained in the descriptor. The size of the memory area in characters is contained in the length field of the descriptor. The structure of the String Descriptor is illustrated in figure 6-5 and contains the following information:

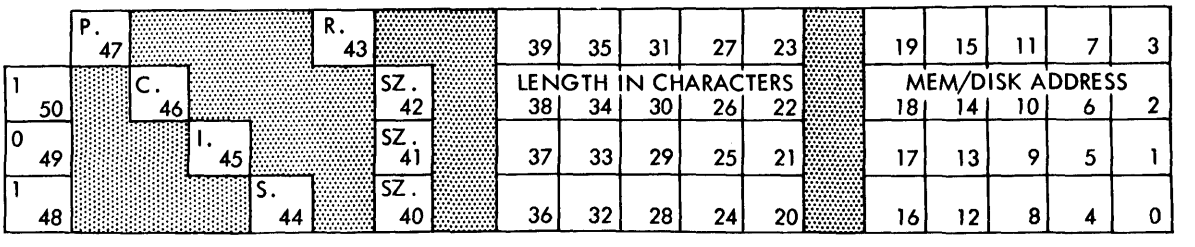


Figure 6-5. String Descriptor (Non-indexed)

a. Bit 50:3, a tag of 101.

b. Bit 47:1 the presence bit. A zero causes a presence bit interrupt if the descriptor is used to access data. A one indicates the data is present in main memory.

c. Bit 46:1, the copy bit. A zero indicates that this is the original descriptor for the particular data area. A one indicates that this descriptor is a copy of the original descriptor.

d. Bit 45:1, the indexed bit. A zero indicates indexing is required. A one indicates that indexing has taken place and the word and character index are length/index field (see figure 6-6).

e. Bit 44:1, the segmented bit. A zero indicates that the data area is not segmented. A one indicates that the data is segmented.

f. Bit 43:1, the read only bit. A zero indicates that the data may be referenced for reading or writing. A one indicates that the data can be read only.

g. Bit 42:3, character size field. 100 indicates 8-bit bytes, 011 indicates 6-bit characters, and 010 indicates 4-bit digits.

h. Bit 39:20, contains either the length of the memory area (bit 45=0) or an index value (bit 45=1). When bit 45 equals zero, this field contains the length of the area in digits, characters or bytes. This field is used for size checking during indexing operations. When bit 45 is equal to one, bits 39:4 contain a byte index and bits 35:16 contain a word index as illustrated in figure 6-6.

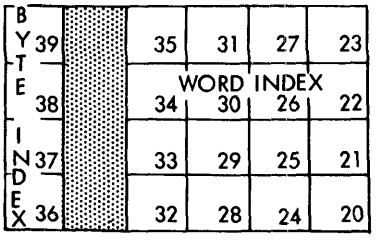


Figure 6-6. Byte/Word Index Field

i. Bit 19:20, contains either a main memory or a disk address. If the presence bit (bit 47) is one, the field contains a memory address of the data. If both the presence bit and the copy bit (bit 46) are equal to zero, the field contains the disk address of the non-present data. If the presence bit is zero and the copy bit is one, the field contains the memory address of the original descriptor.

SEGMENT DESCRIPTOR.

The segment descriptor (figure 6-7) describes a program segment and contains the following information:

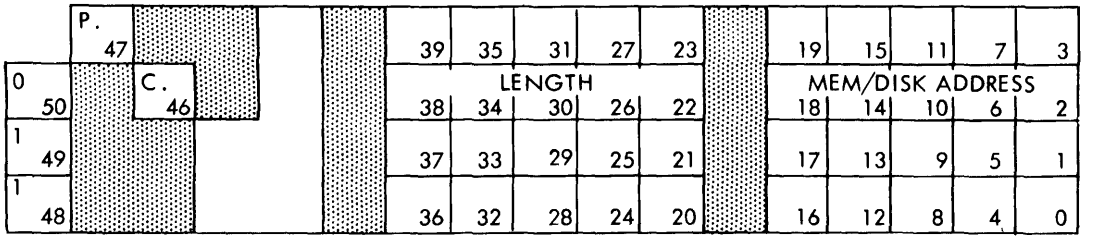


Figure 6-7. Segment Descriptor

a. Bit 50:3, a tag of 011.

b. Bit 47:1, the presence bit. A zero indicates that the segment is absent from main memory.

c. Bit 46:1, the copy bit. A zero bit indicates that this is the original segment descriptor. A one indicates that this is a copy of the original segment descriptor.

d. Bit 45:1, unused.

e. Bit 44:5, unused.

NOTE

unused bits may be
either zero or one.

f. Bit 39:20, specifies the length of the program segment in words.

g. Bit 19:20 contains either the main memory address or the disk file address. If the present bit (bit 47 equals one, the field contains the main memory address of the program segment. If both the presence bit and the copy bit (bit 46)

equal zero, the field contains the disk address of the non-present program segment. If the presence bit equals zero and the copy bit equals one, the field contains the absolute memory address of the original program segment descriptor.

MARK STACK CONTROL WORD.

The Mark Stack Control Word (MSCW), with the Return Control Word, provides a linking mechanism for the history of previous control-register settings through the stack.

The MSCW is placed in the stack by the Mark Stack operator. The MSCW is organized as illustrated in figure 6-8 and provides the following data:

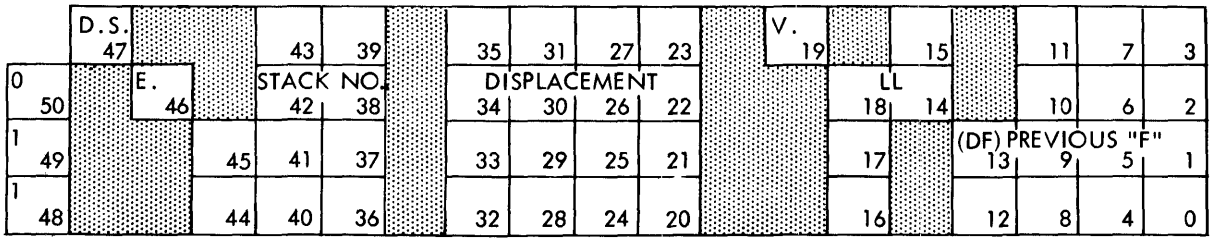


Figure 6-8. Mark Stack Control Word

a. Bit 50:3, a tag of 011.

b. Bit 47:1, the different-stack bit. A zero indicates that the stack-number field refers to the current stack. A one indicates that the stack-number field refers to a different stack.

c. Bit 46:1, the environment bit. A zero indicates an inactive MSCW, generated directly by the Mark Stack operator. The procedure entry has not been performed. A one denotes an active MSCW generated upon entry into a procedure, at which time the environment fields (stack number, displacement, and value fields) are stored into the Mark Stack Control Word.

d. Bit 45:10, the stack-number field, contains the number of the stack from which the PCW was obtained at procedure-entry.

e. Bit 35:16, the displacement field, which, when added to the stack base address, locates the Mark Stack Control Word of the prior lexicographic level.

f. Bit 19:1, the value bit. A zero indicates that the MSCW was generated during any operation that will be restarted from the beginning. A one indicates that the operator must continue after the Exit or Return which refers to this MSCW (e.g., an accidental entry by a Value Call).

g. Bit 18:5, the LL field denotes the lexicographical level at which the program was running when the procedure was entered.

h. Bit 13:14, denotes the stack history. This field locates in the stack, the preceding MSCW (i.e., the previous "F" register setting).

PROGRAM CONTROL WORD.

The Program Control Word (PCW), and the Mark Stack Control Word are used during entry into a procedure. The organization of the PCW is illustrated in figure 6-9 and contains the following:

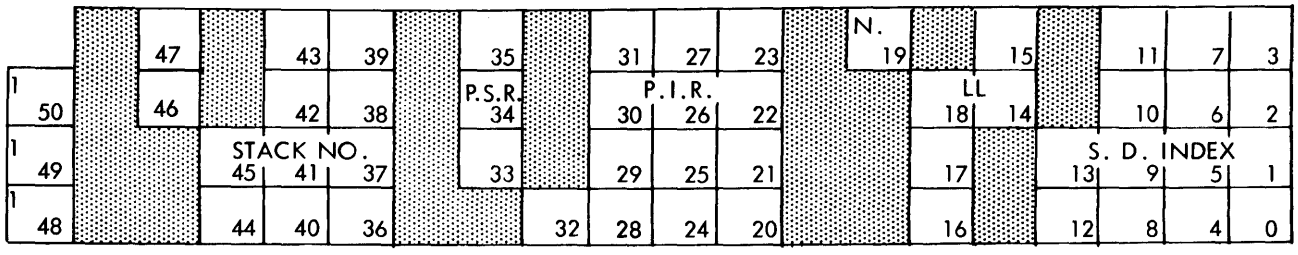


Figure 6-9. Program Control Word

a. Bit 50:3, a tag of 111.

b. Bit 47:1, unused.

c. Bit 46:1, unused.

d. Bit 45:10, stack number containing the PCW.

e. Bit 35:3, defines the program syllable within the word located by PIR.

f. Bit 32:13, an index to the Program Base Register.

g. Bit 19:1, normal state (zero) or control state (one).

h. Bit 18:5, the level of the procedure being entered.

i. Bit 13:14, the segment descriptor index. Bits 12 through zero specify the value to be added to the address located by either D register zero or one. When bit 13 equals zero, D register zero is selected; when bit 13 equals one, D register one is selected.

RETURN CONTROL WORD.

The Return Control Word and the Mark Stack Control Word are used for subroutine handling. The Return Control Word stores the environment to which the subroutine will return. The organization of the Return Control Word is illustrated in figure 6-10 and contains the following:

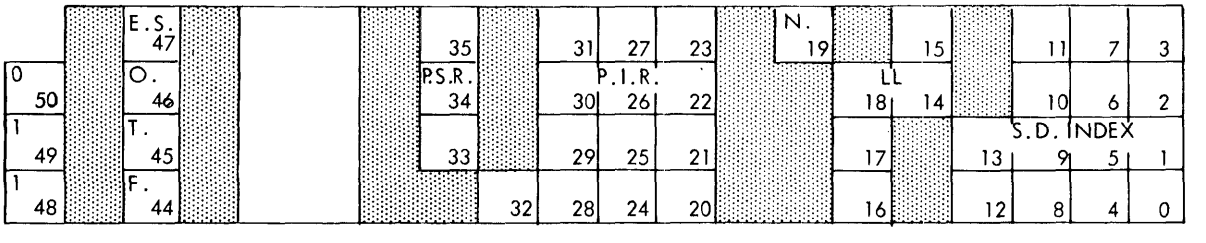


Figure 6-10. Return Control Word

a. Bit 50:3, a tag of 011.

b. Bit 47:1, External Sign flip flop.

c. Bit 46:1, Overflow flip flop.

d. Bit 45:1, True/False flip flop.

e. Bit 44:1, Float flip flop.

NOTE

43:1 will probably be TFOF, True/False flip flop occupied flip flop.

f. Bit 43:8 unused.

g. Bit 35:3, the program syllable of the operator to be executed after return from the subroutine.

h. Bit 32:13, the PIR setting of the operator to be executed next in the calling routine.

i. Bit 19:1, a normal state (zero) or control state (one) procedure.

j. Bit 18:5, the level of the calling procedure when the RCW is generated (at procedure entry).

k. Bit 13:14, the segment descriptor index. Bits 12 through zero specify the value to be added to the address located by either D register 0 or 1. When bit 13 = zero, D register zero is selected; when bit 13 = one, D register one is selected.

INDIRECT REFERENCE WORD.

Referencing a variable within the current addressing environment of a procedure is accomplished through the address couple in the Indirect Reference Word (IRW), and the Segment Descriptor Index of the Program Control Word (PCW). Both references are relative to the D Register specified by the address couple. The bit format of the IRW is shown in figure 6-12.

STUFFED INDIRECT REFERENCE WORD.

Reference to variables outside the current environment is accomplished by a (stuffed) SIRW. This addressing is relative to the base of the stack in which the variable is located.

The SIRW contains the stack number, the location (DISP) of the MSCW, and the displacement of the variable relative to the MSCW. The absolute memory location of the variable is formed by adding the contents of DISP and displacement to the base address of the referenced stack from the stack descriptor. The contents of the SIRW (with the exception of displacement) is dynamic and is accumulated as the program is executed. The stack number and DISP fields are entered into the SIRW by a special operator (STFF). The bit format of SIRW is shown in figure 6-11.

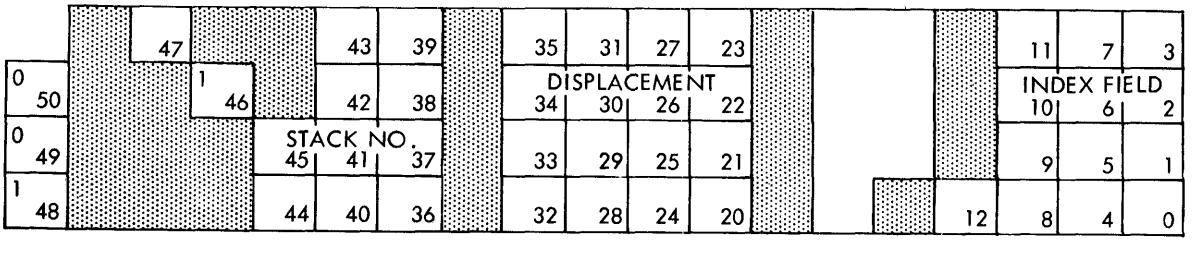


Figure 6-11. Stuffed Indirect Reference

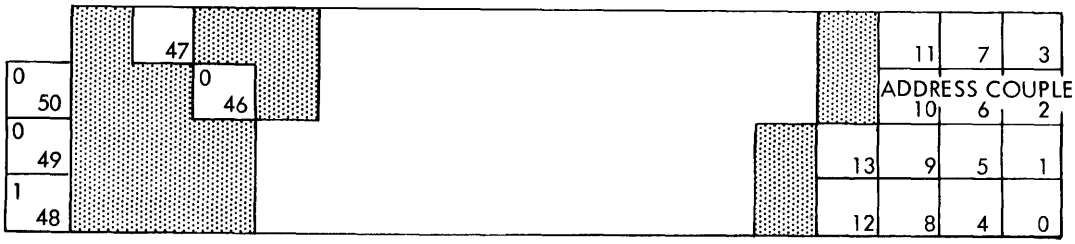


Figure 6-12. Normal Indirect Reference Word

a. Bit 50:3, a tag of 001.

b. Bit 47:1, unused.

c. Bit 46:1, the environment bit. A one indicates a Stuffed IRW. A zero indicates a Normal IRW.

d. Bits 45:10, stack number. When bit 46 equals one, specifies the number of the stack containing the address.

e. Bit 45:26, unused, when bit 46 equals zero.

f. Bit 35:16, displacement field. When bit 46 equals one, this value added to the stack base address locates a Mark Stack Control Word.

g. Bit 19:6, unused.

h. Bit 13:14, index field. When bit 46 equals one, the index value is added to the contents of the D register specified by the Mark Stack Control Word. Bit 13 is always zero.

i. Bit 13:14, when bit 46 equals zero, is divided into two functional fields (figure 6-13). Each field is variable in length. The first sub-field, designated LL, selects one of the D registers. The second sub-field is an index value which is added to the contents of the selected D register to form an absolute address. The lengths of the sub-fields are defined by the current program level as shown in Table 6-1.

Table 6-1

Sub-Field Lengths

Program Level	Length of LL Field (Bits)	Length of Index Field (Bits)
0-1	1	13
2-3	2	12
4-7	3	11
8-15	4	10
16-31	5	9

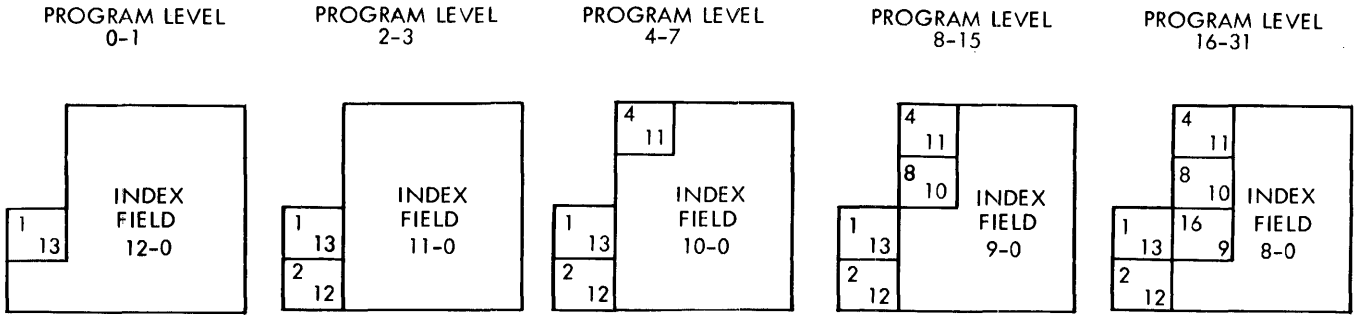


Figure 6-13. Program Level Bit Assignment

NOTE

The bit order of the
LL field is inverted.

STEP INDEX WORD.

The Step Index Word figure 6-14 is used by the Step and Branch operator, to increase efficiency in iteration loops. It contains the following information:

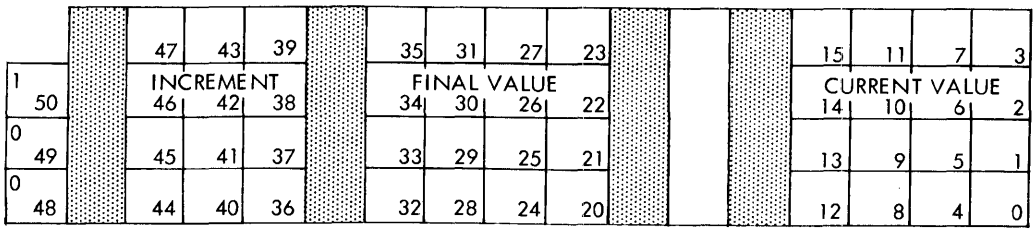


Figure 6-14. Step Index Word

a. Bit 50:3, a tag of 100.

b. Bit 47:12, the value of the increment to be added to the current value field.

c. Bit 35:16, the final value, used to terminate the iteration loop.

d. Bit 19:4, unused.

NOTE

These bits must be zero.

e. Bit 15:16, the current value or count.

SECTION 7
PRIMARY MODE OPERATORS

GENERAL.

This section defines the primary operator's functions. In each case the operator's name, mnemonic, and hexadecimal code is shown.

The universal operators are also included in this section.

ARITHMETIC OPERATORS.

The arithmetic operators usually require two operands in the top of stack registers. These operands are combined by the arithmetic process specified with the result placed in the top of the stack. The operands may be either single-precision, double-precision, or intermixed. The specified arithmetic process adapts automatically to the data environment, with single-precision process invoked if both operands are of the single-precision type and a double-precision process invoked if either operand is of the double precision type.

Each double-precision operand occupies two words. The second word of the operand is an extension of the first word of the operand, i.e., the mantissa of the first word of the operand may be an integer but the mantissa of the second word is always a fraction. When the top of stack registers are full, the first word of the first operand occupies the A register, the second word of the first operand occupies the X register. The first word of the second operand occupies the B register, the second word of the second operand occupies the Y register. Therefore, double-precision arithmetic processes operate on four words in the stack instead of two as in single-precision operations. Double-precision arithmetic leaves a two-word result in the top of the stack.

Add, Subtract, and Multiply operations with two integer operands yield an integer result if no overflow occurs. If one or both operands is non-integer, or if the result generates an overflow, the result is non-integer.

Upon entry into any operator the hardware stack-adjust function fills or empties the top of stack register as required by the operator. If either register contains an incorrect word, the operator is terminated with an invalid operand interrupt.

ADD (ADD) 80.
The operands in the A register and the B register are added algebraically with the sum left in the B register. At the end of the operation the A register is marked empty, and the B register is marked full.

If only one of the operands is double-precision the single-precision operand's extension register is set to zero. The B register is marked as a double-precision operand at completion of the operation.

If the mantissa signs and the exponents are equal, the mantissas are added and the sum placed in the B register. If the sum exceeds 13(26) octal digits, the mantissa of the sum is shifted right one octade, rounded, and the exponent is algebraically increased by one.

If the exponents are equal but the mantissa signs are unequal, the difference of the mantissas with the appropriate sign is placed in the B register.

If the exponents are unequal, the operands are first aligned. If the alignment causes the smaller operand to be shifted right 14(27) octal places, the larger operand is the result.

If the alignment causes the smaller operand to be shifted right, but less than 14(27) octal places, the digits of the smaller operand shifted out of the register are saved and used to obtain the rounded result.

If the signs of the operands are equal, the mantissas are added and the sum placed in the B register. If the sum does not exceed 13(26) octal digits, the last digit shifted out of the register

is used to round the result. If the sum is 14(27) octades the mantissa in B (Y) is rounded to 13(26) digits.

If the signs of the operands are unequal, an internal subtraction takes place with the rounded result placed in the B register.

If the result has an exponent greater than +63 (+32,767), the exponent overflow interrupt is set. If the result has an exponent less than -63 (-32,767) the exponent underflow interrupt is set.

SUBTRACT (SUBT) 81.
The operand in the A register is algebraically subtracted from the operand in the B register with the difference left in the B register. The operation is the same as for the Add operator except for initial sign comparisons.

MULTIPLY (MULT) 82.
The operand in the A register is algebraically multiplied by the operand in the B register. The rounded product is left in the B register.

If the mantissa of either operand is zero, the B register is set to zero.

If both mantissas are non-zero, the product of the mantissas is computed. If the product contains more than 13(26) digits, it is normalized and rounded to 13(26) digits. A mantissa of all sevens is not rounded.

If the result has an exponent greater than +63 (+32,767), an exponent overflow interrupt is set. If the result has an exponent less than -63 (-32,767), an exponent underflow interrupt is set.

EXTENDED MULTIPLY (MULX) 8F.
The operands in the A and B registers are algebraically multiplied and a double-precision product is placed in the B and Y registers. The A register is marked empty and the B register marked full.

The actions outlined for Multiply operations also apply to this operator.

If either or both operands are double-precision, then a normal double-precision operation occurs.

DIVIDE (DIVD) 83.
The operand in the B register is algebraically divided by the operand in the A register, with the quotient left in the B register. After the operation the A register is marked empty, and the B register is marked full.

If the mantissa of the B register is zero, the B register is set to zero. If the A register mantissa is equal to zero, the divide by zero interrupt is set. In either case the operation is terminated.

If the mantissas of both operands are non-zero, they are normalized and the operand in the B register is divided by the operand in the A register. The quotient is developed to 14(27) digits, rounded to 13(26) digits, and left in the B register.

If the result has an exponent greater than +63 (32,767) the exponent overflow interrupt is set. If the result has an exponent less than -63 (32,767) the exponent underflow interrupt is set.

INTEGER DIVIDE (IDIV) 84.
The operand in the B register is algebraically divided by the operand in the A register and the integer part of the quotient is left in the B register. After the operation the A register is marked empty and the B register is marked full.

If the mantissa of the B register is zero, the B register is set to zero. If the mantissa of the A register is zero, the divide by zero interrupt is set. The operation is terminated in either case.

If the mantissas of both operands are non-zero, they are normalized. If the exponent of the B register is algebraically less than the

exponent of the A register after both operands have been normalized, the B register is set to zero. If the exponent of the B register is algebraically equal to or greater than the exponent of the A register the divide operation proceeds until an integer quotient or a quotient of 13(26) significant digits is calculated.

If an integer quotient is developed, the quotient is left in the B register with zero exponent for S.P. and the exponent set to 13 for D.P. If a non-integer quotient is developed, the integer overflow interrupt is set.

REMAINDER DIVIDE (RDIV) 85.
The operand in the B register is algebraically divided by the operand in the A register to develop an integer quotient. The remainder of this Division is left in the B register. If this remainder is an integral value it is in the form of an integer (exponent = 0 for S.P., 13 for D.P.). After the operation the A register is marked empty, the B register is marked full.

If the mantissa of the B register is zero, the B register is set to zero. If the mantissa of the A register is zero the divide by zero interrupt is set. In either case the operation is terminated.

If both mantissas are non-zero, both operands are normalized. If the exponent of the B register is algebraically less than the exponent of the A register after both operands have been normalized, the operand in the B register is the result. If the exponent of the B register is algebraically equal to or greater than the exponent in the A register, the divide operation proceeds until an integer quotient is developed and the remainder is then placed in the B register.

If a non-integer quotient is developed, the integer overflow interrupt is set and the operation is terminated.

INTEGERIZE, TRUNCATED (NTIA) 86.
The operand in the B register is converted to integer form without rounding and left in the B register.

If the operand in the B register can not be integerized, i.e., the exponent is greater than the number of leading zeros in the operand, the integer overflow interrupt is set and the operation is terminated.

INTEGERIZE, ROUNDED (NTGR) 87.
The operand in the B register is converted to integer form. Rounding takes place if the absolute value of the fraction is greater than 4. The rounded result is left in the B register.

If the operand in the B register can not be integerized, i.e., the exponent is greater than the number of leading zeros in the operand, the integer overflow interrupt is set and the operation is terminated.

The operand is rounded if necessary by adding one to the mantissa. If a non-integer results from this operation, the integer overflow interrupt is set.

TYPE-TRANSFER OPERATORS.
SET TO SINGLE-PRECISION, TRUNCATED (SNGT) CC.
The operand in the B register is set to a single-precision operand, or in the case of a data descriptor, the double-precision bit is set to zero.

If the word in the B register is a non-indexed, double-precision data descriptor, the double-precision bit is cleared to zero and the length field multiplied by 2.

If the double-precision operand in the B register has an exponent greater than +63 the exponent overflow interrupt is set. If the exponent is less than -63 exponent underflow is set, and the operation is terminated.

If the operand in the B register is a double-precision operand with an exponent less than +63 or greater than -63 the operand is normalized, and the tag field in the B register is set to single precision.

If the word in the B register is not an operand or a Data Descriptor, then the invalid operand interrupt is set and the operation terminated.

If the operand is single-precision, it is normalized and the operation is terminated.

SET TO SINGLE-PRECISION, ROUNDED (SNGL) CD.
The operand in the B register is changed to a rounded, single-precision operand.

If the double-precision operand in the B register has an exponent greater than +63 the exponent overflow is set. If the exponent is less than -63 the exponent underflow is set. In either case the operation is terminated.

If the operand in the B register is a double-precision operand with an exponent less than +63 or greater than -63 the operand is normalized, the tag field in the B register is set to single-precision, and the operand in the B register is rounded from the Y register. The Y register is set to zero.

If a carry is developed during the rounding operation the operand is adjusted and the new exponent is checked as above.

If the operand is a single-precision operand, the operand is normalized and no rounding occurs. The action is as stated for the Set to Single-Precision, Truncated.

SET TO DOUBLE-PRECISION (XTND) CE.
The word in the B register is set to a double-precision operand with the Y register set to zero. If a single-precision data descriptor is present in the B register the double precision bit is set to one.

If the word in the B register is a data descriptor with both the index bit and double-precision bit zero, the double-precision bit is set to one and the length field is divided by two.

If the operand in the B register is a double-precision operand the operation is complete. If it is a single-precision operand the tag field in the B register is set to double-precision and the Y register is set to all zeros.

If the word in the B register is not an operand or a Data Descriptor, then the invalid operand interrupt is set and the operation terminated.

LOGICAL OPERATORS.

LOGICAL AND (LAND) 90.

Each bit of the B operand, except for the tag bits, is set to one where a one appears in the corresponding bit positions in both the A operand and the B operand. The other information bits of the B operand are set to zero. The tag of the B operand is not disturbed, unless the tag of the A operand specifies double-precision, in which case, the B operand tag is set to double-precision.

LOGICAL OR (LOR) 91.

All bit positions of the B operand except the tag bits, are set to one if the corresponding bit position in either the A operand or the B operand is one, otherwise the bit is set to zero. The tag bits are set to the value of the second item in the stack except when the A operand is double-precision, in which case, the B register tag is set to double-precision.

LOGICAL NEGATE (LNOT) 92.

Every bit in the A operand is complemented except the tag field, which remains unchanged.

LOGICAL EQUIVALENCE (LEQV) 93.

Each bit of the B operand is set to 1 except the tag bits, when the corresponding bits of the A operand and the B operand are equal. Each bit of the B operand is set to 0 except the tag bits, when the corresponding bits of the A and B operands are not equal. The tag field is normally set to the value of the second item in the stack except when the A operand is double-precision, in which case the B register tag is set to double-precision.

RELATIONAL OPERATORS.

The relational operators perform algebraic comparison on the operands in the A register and the B register. The single precision result is left in the B register. The result is an operand in integer form with the value one if the relationship has been met or an operand with all information bits set to zero if the relationship was not met. All relational operations compare the B operand to the A operand.

LOGICAL EQUAL (SAME) 94.

All bits, including tag bits, of the A operand and B operand are compared. If all bits are equal, a single precision operand with bit zero set to one and all other information bits set to zero is stored in the B register. Otherwise, a single-precision operand with all information bits set to zero is stored in the B register.

GREATER THAN (GRTR) 8A.

If the B operand is algebraically greater than the A operand, the B register is set to an integer form one. Otherwise, all bits in the B register are zero.

GREATER THAN OR EQUAL (GREQ) 89.

If the B operand is algebraically greater than or equal to the A operand, the B register is set to an integer form one. Otherwise, all bits in the B register are zero.

EQUAL (EQUL) 8C.

If the operands in the B and A registers are algebraically equal, the B register is set to an integer form one. Otherwise, all bits in the B register are zero.

LESS THAN OR EQUAL (LSEQ) 8B.

If the B operand is algebraically less than or equal to the operand in the A register, the B register is set to an integer form one. Otherwise, the B register bits are all zero.

LESS THAN (LESS) 88.
If the operand in the B register is algebraically less than the operand in the A register, set the B register to an integer form one. Otherwise, the bits in the B register are all zero.

NOT EQUAL (NEGL) 8D.
If the operand in the B register is not algebraically equal to the operand in the A register, set the B register to an integer form one. Otherwise, the bits in the B register are all cleared to zero.

BRANCH OPERATORS.
Branch instructions break the normal sequence of serial instruction fetches. Branching may be either relative to the base address of the current program segment or to a location in another program segment. Branch operators may be conditional or unconditional.

BRANCH FALSE (BRFL) A0.
If the low order bit of the A register is zero, the Program Index Register and Program Syllable Register are set from the next two syllables in the program string. Otherwise, PIR and PSR are advanced three syllable positions.

The two syllables following the actual operator syllable form the new PIR and PSR settings as follows: The three high order bits are placed into Program Syllable Register and the next 13 low order bits are placed in the Program Index Register. The Program Register (P) is marked empty to cause an access to the new program word.

BRANCH TRUE (BRTR) A1.
If the low order bit of the A register is one, the Program Index Register and Program Syllable Register are set from the next two syllables in the program string. Otherwise, PIR and PSR are advanced three syllable positions. The Branch True Operator uses the two syllables as described for the Branch False operator.

BRANCH UNCONDITIONAL (BRUN) A2.
Program Index Register and the Program Syllable Register are set

from the next two syllables of the program string. The Branch Unconditional operator uses the two syllables as described for the Branch False operator.

DYNAMIC BRANCH FALSE (DBFL) A8.
If the low order bit of the B register is zero and the word in the A register is a Program Control Word, or an indirect reference to one, branch to the specified syllable of that program segment.

If the low order bit of the B register is zero and the word in the A register is an operand, PIR and PSR are set from this operand.

If the word in the A register is an operand, it is used in the following manner: The operand is made into an integer. If it is negative or is greater than 16,384, the invalid index interrupt is set and the operation is terminated. If bit zero of the operand is zero, PSR is set to zero, otherwise PSR is set to three. The next higher order 20 bits are placed in the Program Index Register. The Program Register is then marked empty to cause access to the new program word.

DYNAMIC BRANCH TRUE (DBTR) A9.
If the low order bit of the B register is one and the word in the A register is a Program Control Word, or an indirect reference to one, branch to the specified syllable of the program segment.

If the low order bit of the B register is one and the word in the A register is an operand, PIR and PSR are set from this operand.

The operand in the A register is used in this operator in the manner described for the Dynamic Branch False operator.

DYNAMIC BRANCH UNCONDITIONAL (DBUN) AA.
If the word in the A register is a Program Control Word or an indirect reference to one, branch to the specified syllable of the program segment.

If the word in the A register is an operand, PIR and PSR are set from this operand.

The operand in the A register is used in this operator in the same manner described for the Dynamic Branch False operator.

STEP AND BRANCH (STBR) A4.
The increment field of the step-index word addressed by the contents of the A register, is added to its current-value field. If the current-value field is then greater than the final-value field, Program Index Register and Program Syllable Register are set from the next two syllables from the program string. Otherwise, Program Index Register and the Program Syllable Register are advanced three syllables. The step-index word is replaced in memory.

If no SIW is in memory, and if an operand is found, it is left in the stack. The A register is set to zero, PIR/PSR are advanced and the next operator is executed. If no operand is encountered, the invalid operand interrupt is set.

UNIVERSAL OPERATORS.
NO OPERATION (NOOP) FE.
No operation takes place when this syllable is encountered. PIR and PSR are advanced to the next operator. This operator is also valid in Variant Mode and Edit Mode.

CONDITIONAL HALT (HALT) DF.
This operator halts the processor if the conditional halt switch is in the ON position. If the conditional halt switch is OFF, the operator is treated as a NOOP. This operator is also valid in Variant Mode and Edit Mode.

INVALID OPERATOR (NVLD) FF.
This operator sets the invalid operand interrupt. It is also valid in Variant Mode and Edit Mode.

STORE OPERATORS.
The store operators use the words in the A register and B register. The operand in the B register is stored in memory at the location addressed by an Indirect Reference Word or a Data Descriptor. If the A register contains an operand a hardware interchange takes

place so that the operand is in the B register.

STORE DESTRUCTIVE (STOD) B8.
If the word in the A register is an operand the A and B operands are interchanged. The Data Descriptor or IRW in the A register is the address in memory where the operand in the B register (B, Y registers) is stored. After the operand is stored, the A register and the B register are marked empty and the operation is complete.

If the word addressed by the Indirect Reference Word is a Program Control Word, accidental entry occurs.

If the word addressed by the Data Descriptor has the memory protect bit on (bit 48), the memory protect interrupt is set and the operation is terminated.

If the presence bit in the Data Descriptor is zero the presence bit interrupt is set. After the data has been made present the operation is restarted.

STORE NON-DESTRUCTIVE (STON) B9.
This operator functions the same way as the Store Destructive operator except that at the completion of this operator the operand is left in the B register.

OVERWRITE DESTRUCTIVE (OVRD) BA.
This operator functions the same way as the Store Destructive, except that it overrides memory protection checks.

OVERWRITE NON-DESTRUCTIVE (OVRN) BB.
This operator functions the same way as the Store Non-Destructive, except that it overrides memory protection checks.

STACK OPERATORS.
EXCHANGE (EXCH) B6.
The operands in the A register and the B register are exchanged. The A and B registers may contain either operands or control words. The control words are treated as operands by this operator.

DELETE TOP OF STACK (DLET) B5.
This operator marks the A register empty.

DUPLICATE TOP OF STACK (DUPL) B7.
The operand found in the B register is copied into the A register. The A register is marked full.

PUSH DOWN STACK REGISTERS (PUSH) B4.
This operator stores the valid word/words from the A register and/or B register into the memory portion of the stack. The A and B registers are marked empty.

LITERAL CALL OPERATORS.
LIT CALL ZERO (ZERO) B0.
This operator sets the A register to zero and marks the register full. The result is a single-precision operand.

LIT CALL ONE (ONE) B1.
This operator sets the A register low order bit (bit 0) to one, leaving all other bits set to zero. The A register is marked full. The result is a single-precision operand.

LIT CALL 8 BITS (LT8) B2.
The syllable following the operator is the literal value to be placed in the A register bits 7:8. The rest of the A register is set to zero. The A register is marked as full and the Program Syllable Register is set to the syllable following the literal.

LIT CALL 16 BITS (LT16) B3.
The next two syllables following the operator are a 16-bit literal value that is placed in the A register bits 15:16. The rest of the register is set to zero. The A register is marked full and PSR is advanced past the 16-bit literal.

LIT CALL 48 BITS (LT48) BE.
The next program word is placed in the A register, and the A register tag is set to zero. The A register is marked full, and PIR and PSR are advanced to the program syllable following the 48-bit literal value. This operator requires that the 48 bit literal in

the program string be word synchronized if the operator syllable is in any syllable position other than syllable 5, the syllables intervening are not executed and are filled with invalid OP-Codes.

MAKE PROGRAM CONTROL WORD (MPCW) BF.
This operator performs a Lit Call 48 Bits as described above; however, the tag is set to a PCW (111) and the Stack Number Register is placed in bits 45:10. The A register is marked full.

INDEX AND LOAD OPERATORS.
INDEX (INDX) A6.
The Index operator places the integerized value of the B register into the 20-bit length/index field of the Descriptor in the A register. The Descriptor is marked indexed (bit 45 is set to one). The A register is marked full and the B register is marked empty.

If the word in the A register is an operand, the A operand is exchanged with the B operand. If the word in the A register is neither a Descriptor nor an Indirect Reference Word Pointing to a Descriptor, the invalid operand interrupt is set and the operation is terminated.

If the indexing value is negative or greater than or equal to the length field of the Descriptor the invalid index interrupt is set and the operation is terminated.

If the descriptor is segmented, the index is partitioned into two portions by dividing it by the proper divisor determined by the type of data referenced by the descriptor, (d. p. word-128, s. p. word-256, 4-bit digit-3072, 6-bit character-2048, or 8-bit byte-1536). The quotient is used as an index to the given descriptor to fetch the array-row descriptor. The remainder is used to index the row descriptor.

If the Double-Precision bit (bit 45) in the Descriptor is one, the index value in the B register is doubled. The balance of the operation is as described in the first paragraph of this operator.

INDEX AND LOAD NAME (NXLN) A5.
This operator performs an Index operation, then after the word in the A register is indexed, the Data Descriptor pointed to by this word is brought to the A register. The Copy bit (bit 46) of the Data Descriptor is set to one and the A register is marked full. If the presence bit (bit 47) is off, the address of the original descriptor is placed in the address field of the stack copy. If the word accessed by the index word in the A register is not a Data Descriptor the invalid operand interrupt is set and the operation is terminated.

If the Data Descriptor accessed by the indexed word in the A register has the Index bit (bit 45) set to one the invalid operand interrupt is set and the operation is terminated.

INDEX AND LOAD VALUE (NXLV) AD.
This operator performs an Index operation, then after the word in the A register is indexed the operand pointed to by this descriptor is brought to the A register. The A register is marked full.

If the word accessed is other than an operand the invalid operand interrupt is set and the operator is terminated.

LOAD (LOAD) BD.
The Load operator places the word addressed by the IRW or INDEXED DATA DESCRIPTOR in the A register.

If at the start of this operator the A register contains other than a Data Descriptor or an Indirect Reference Word pointing at a Data Descriptor, the invalid operand interrupt is set and the operation is terminated.

If the word pointed at by the Data Descriptor is another Data Descriptor, that Data Descriptor is marked as a copy (Copy bit [bit 46] is set to one) and if the presence bit (bit 47) is off, the address of the original is placed in bits 19:20 of the copy in the stack.

SCALE OPERATORS.

Higher-level languages such as COBOL require integer arithmetic. The Scale Operators provide the means of aligning decimal points prior to performing the arithmetic operations. In addition, the Scale Right operators provide for binary to decimal conversions.

SCALE LEFT (SCLF) C0.

This operator uses the second syllable as the scale factor. The operand to be scaled is placed in the B register and integerized. The resulting integer is then multiplied by 10 raised to the power specified by the scale factor.

If scaling of a single-precision operand results in overflow the single-precision operand is converted to a double-precision integer. A double-precision integer is defined as a double-precision operand with an exponent equal to 13.

If scaling of the operand results in an exponent greater than 13, (double-precision operand), the overflow FF is set to one.

DYNAMIC SCALE LEFT (DSLF) C1.

This operator performs the same operation as the Scale Left operator; however, scale factor is taken from the A register rather than the program syllable following the operation syllable. The operand in the A register is integerized before the scale.

SCALE RIGHT SAVE (SCRS) C4.

This operator uses its second syllable as the scale factor. The operand to be scaled is placed in the B register and is then integerized. The resultant integer is then effectively divided by 10 raised to the power specified by the scale factor.

The quotient resulting from the division is left in the A register. The operand in the B register is the remainder which is converted to decimal (4 bit digits) and is left justified. A and B registers are both marked full.

If the scale factor is greater than 12, the invalid operand interrupt is set and the operation is terminated.

DYNAMIC SCALE RIGHT SAVE (DSRS) C5.
This operator performs the same operation as the Scale Right Save operator; however, the scale factor is obtained from the A register rather than the program syllable following the operation syllable. The operand in the A register is integerized before being used.

SCALE RIGHT TRUNCATE (SCRT) C2.
This operator performs a Scale Right function using its second syllable as the scale factor. The B register is marked as empty at the conclusion of this operator.

DYNAMIC SCALE RIGHT TRUNCATE (DSRT) C3.
This operator performs the same operation as the Scale Right Truncate except that the scale factor is found in the A register and is first integerized by the operator.

SCALE RIGHT FINAL (SCRF) C6.
This operator performs a Scale Right operation except that the quotient in the A register is deleted by marking the A register empty. The sign of the quotient is placed in the external sign flip flop.

If the quotient was non-zero at the conclusion of the operation the overflow flip flop is set.

DYNAMIC SCALE RIGHT FINAL (DSRF) C7.
This operator performs a Scale Right Final operation with the scale factor found in the A register which is integerized by the operator before use.

SCALE RIGHT ROUNDED (SCRR) C8.
This operator performs a Scale Right operation and the quotient is rounded by adding one to it if the most significant digit of the remainder is equal to or greater than five. The remainder is deleted from the stack by marking the B register empty.

DYNAMIC SCALE RIGHT ROUND (DSRR) C9.
This operator performs a Scale Right Rounded operation with the scale factor found in the A register.

BIT OPERATORS.
The Bit operators are concerned with a specified bit in the A register and/or B register.

BIT SET (BSET) 96.
This operator sets a bit in the A register. The bit that is set is specified by the program syllable following the operation syllable.

If the program syllable defining the bit to be set has a value greater than 47, the invalid-operand interrupt is set and the operation is terminated.

DYNAMIC BIT SET (DBST) 97.
This operator performs a Bit Set Operation upon the bit specified by the operand in the top of stack register. This word is integerized before using it as a bit number.

If the word in the top of stack register is not an operand an invalid operand interrupt is set and the operation is terminated.

If after being integerized the operand is less than zero or greater than 47, an invalid operand interrupt is set and the operation is terminated.

BIT RESET (BRST) 9E.
This operator resets a bit in the A register. The bit that is reset is specified by the syllable following the operation syllable.

If the program syllable defining the bit to be reset has a value greater than 47, an invalid-operand interrupt is set and the operation is terminated.

DYNAMIC BIT RESET (DBRS) 9F.
This operator performs a Bit Reset operation upon the bit specified by the operand in the top of stack register.

If the word in the top of the stack register is not an operand an invalid operand interrupt is set and the operation is terminated.

If after being integerized the operand is less than zero or greater than 47, an invalid operand interrupt is set and the operation is terminated.

CHANGE SIGN BIT (CHSN) 8E.
The sign bit (bit 46) of the top-of-stack operand is complemented, i.e., if it is a one it is set to zero, if zero it is set to one.

TRANSFER OPERATORS.
The Transfer Operators transfer any field of bits from one word in the stack to any field of another word in the stack.

FIELD TRANSFER (FLTR) 98.
This operator uses its following three syllables to establish the pointers used in the field transfer. This is done in the following manner: The second syllable of the operator is K. The third syllable of the operator is G. The fourth syllable of the operator sets the L register.

The field in the A register, starting at the bit position addressed by G is transferred into the B register starting at the bit position addressed by K. The length of the field in the A and B registers is defined by L. When the specified number of bits have been transferred, the A register is set to empty the B register is marked full and the operation is complete.

If the second or third syllables of the operator are found to be greater than 47 or the fourth syllable is greater than 48, the invalid operand interrupt is set and the operation is terminated.

DYNAMIC FIELD TRANSFER (DFTR) 99.
This operator performs a Field Transfer operation with the exception that the B register operand is L. The B register is then reloaded from the stack and this operand is G. The B register is again loaded from the stack and this operand is K.

If any of the three operands is a non-integer, it is first integerized. Each is checked for a value less than zero or greater than 47 or 48 as specified in Field Transfer above. If either of these conditions exist in any one of the three operands, an invalid operand interrupt is set and the operation is terminated.

FIELD ISOLATE (ISOL) 9A.
This operator isolates a field of the word in the A register placing it right justified in the B register. The balance of the B register is cleared to zeros. The A register is marked empty and the B register is marked full.

The operator uses its second and third syllables as the BIT pointers. The second syllable of the operator addresses the starting bit of the field in the A register. The third syllable of the operator specifies the length of the field to be isolated.

If the value of the second syllable is greater than 47 or the value of the third syllable is greater than 48 an invalid operand interrupt is set and the operation is terminated.

DYNAMIC FIELD ISOLATE (DISO) 9B.
This operator performs a Field Isolate operation except that the first item in the stack specifies the length of the field to be isolated. The second operand in the stack addresses the bit in the word of the third item in the stack that is to be isolated.

If after being integerized the value of the first item in the stack is less than zero or greater than 47 an invalid operand interrupt is set and the operation is terminated.

If after being integerized the value of the second item in the stack is less than zero or greater than 48 an invalid interrupt is set and the operation is terminated.

FIELD INSERT (INSR) 9C.
This operator inserts a field from the A register into the B register word. The field in the A register is right justified with the length of the field specified by the third syllable of the operator. The second syllable of the operand addresses the starting bit of the field in the B register. At completion the A register is marked empty and the B register is marked full.

If the value of the second syllable of the operator is greater than 47 an invalid operand interrupt is set and the operation is terminated.

If the value of the third syllable of the operator is greater than 48 an invalid operand interrupt is set and the operation is terminated.

DYNAMIC FIELD INSERT (DINS) 9D.
This operator performs a Field Insert operation except the first item in the stack is used as the insert field data. The second item in the stack is used to specify the length of the field. The third item in the stack is used to address the starting bit in the receiving field in the B register. When operation is complete the A register is marked empty and the B register is marked full.

If after being integerized the value of the second item in the stack is less than zero or greater than 48 an invalid operand interrupt is set and the operation is terminated.

If after being integerized the value of the third item in the stack is less than zero or greater than 47 an invalid operand interrupt is set and the operation is terminated.

STRING TRANSFER OPERATORS.

String Transfer operators give the system the ability to transfer characters or words from one location in memory to another location in memory. The source and destination pointers are set from String Descriptors in the stack.

TRANSFER WORDS, DESTRUCTIVE (TWSD) D3.

This operator requires three items in the top of the stack, an operand, a String Descriptor or operand, and a String Descriptor. The first operand is integerized and used as the count or repeat field. The second item is either the source data or a descriptor which points at the source string and the third item is used to address the destination string. The number of words specified by the repeat field are transferred from the source to the destination. At completion of the operation the A and the B registers are marked empty.

This operation calls the Execute Single Micro, Transfer Words, and End Edit operators before continuing with the program string.

If the memory protect bit is found on during the execution of the Transfer Words operator, the segmented array interrupt is set and the operation is terminated.

TRANSFER WORDS, UPDATE (TWSU) DB.

This operator performs the Transfer Words operator except that at the completion of the transfer of data, the source and destination pointers are updated to point to the location in memory where the transfer ended. The A and B registers are both marked full.

TRANSFER WORDS, OVERWRITE DESTRUCTIVE (TWOD) D4.

This operator performs a Transfer Words, Destructive operation, overriding the memory protection checks.

TRANSFER WORDS, OVERWRITE UPDATE (TWOU) DC.

This operator performs a Transfer Words, Update operation, overriding the memory protection checks.

TRANSFER WHILE GREATER, DESTRUCTIVE (TGTD) E2.
This operator transfers characters from a location in memory pointed to by the source pointer, to a location in memory pointed to by the destination pointer, until the number of characters specified has been transferred or the compare fails.

The first item in the stack is used as the delimiter. The second item in the stack is the maximum number of characters to be transferred. The third item in the stack is the source data or a source pointer and the fourth item in the stack is the destination pointer.

The delimiter character is retained while a call Execute Single Micro operator initiates this operation. The character count is placed in the repeat field register as the EXSD is completed. The source and destination strings are checked for memory protection. The source character is then compared with the delimiter. The result of the compare is set in the True/False flip flop. If the condition is met the TFFF is set to one, if it is not met it is set to zero.

If the number of characters transferred was equal to the repeat field the True/False flip flop will remain set to one. The A and B registers are marked empty and the operation is complete.

If the comparison fails, the number of characters not transferred is placed in the A register and the True/False flip flop is set to zero.

If the first operand in the stack is not a S.P. operand an invalid operator interrupt is set and the operation is terminated.

If either the source or destination word has a memory protect bit on (bit 48=1) the segmented array interrupt is set and the operation is terminated.

If the second item in the stack is a descriptor it is used as the source pointer and the length field or repeat field is set to 1,048,575. All comparisons are binary (EBCDIC Collating Sequence).

TRANSFER WHILE GREATER UPDATE (TGTU) EA.

This operator performs a Transfer While Greater operation and updates the source pointer and destination pointer to point at the next characters in the source and destination strings. The Repeat count is updated to give the number of characters not transferred. If the operation is terminated because the relationship is not met, the source pointer points at the character that failed the comparison.

TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE (TGED) E1.
This operator performs a Transfer While operation using the relation greater than or equal to.

TRANSFER WHILE GREATER OR EQUAL, UPDATE (TGEW) E9.
This operator performs a Transfer While Greater or Equal operation. The source pointer, destination pointers, and count are updated at the conclusion of the operator.

TRANSFER WHILE EQUAL, DESTRUCTIVE (TEGD) E4.
This operator performs a Transfer While operation with the relation used in comparison being equal.

TRANSFER WHILE EQUAL, UPDATE (TEGU) EC.
This operator performs a Transfer While Equal operation. The source pointer, the destination pointer and count are updated at the conclusion of the operator.

TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE (TLED) E3.
This operator performs a Transfer While operation, using the Less than or Equal comparison.

TRANSFER WHILE LESS OR EQUAL, UPDATE (TLEU) EB.
This operator performs a Transfer While Less or Equal operation. The source pointer, destination pointer and count are updated at the conclusion of the operator.

TRANSFER WHILE LESS, DESTRUCTIVE (TLSD) E0.
This operator performs a Transfer While operation using the Less than comparison.

TRANSFER WHILE LESS, UPDATE (TLSU) E8.
This operator performs a Transfer While Less operation. The source pointer, destination pointer and count are updated at the conclusion of the operator.

TRANSFER WHILE NOT EQUAL, DESTRUCTIVE (TNED) E5.
This operator performs a Transfer While operation, with the not equal comparison.

TRANSFER WHILE NOT EQUAL, UPDATE (TNEU) ED.
This operator performs a Transfer While Not Equal operation. The source pointer, the destination pointer and count are updated at the conclusion of the operator.

TRANSFER UNCONDITIONAL, DESTRUCTIVE (TUND) E6.
This operator performs a Transfer While Greater or Equal, Destructive operation forcing a zero delimiter. This causes all characters to be equal or greater than the delimiter thus transfer will continue for the length of the repeat field.

TRANSFER UNCONDITIONAL, UPDATE (TUNU) EE.
This operator performs a Transfer Unconditional operation. The source pointer, the destination pointer and count are updated at the conclusion of the operator.

STRING ISOLATE (SISO) D5.
This operator places in the top of the stack, right justified, the number of characters specified by the repeat field. The first item in the stack is the number of characters in the repeat field. The second item in the stack is either an operand or a descriptor used as the source pointer.

This operator calls and executes the Execute Single Micro, Single Pointer operation before proceeding as above.

If the number of bits to be transferred is greater than 48 the item is double-precision.

If the number of bits is greater than 96 an invalid operand interrupt is set and the operation is terminated.

If the source data has the memory protect bit (bit 48) set to one the segmented array interrupt is set and the operation is terminated.

COMPARE OPERATORS.

The Compare Operators perform the specified compare of two strings of data. The True/False flip flop is conditioned by the results of the compare.

COMPARE CHARACTERS GREATER, DESTRUCTIVE (CGTD) F2.

This operator compares the characters of the two character strings. If the characters in the B string are greater than the characters in the A string the True/False flip flop is set to one. If not the True/False flip flop is set to zero.

The first item in the stack is an operand which contains the length of the fields being compared. The second item in the stack is an operand or a descriptor pointing at the character string to be compared against. The third item in the stack is a descriptor pointing at the character string to be compared.

The operator compares characters until it encounters a pair which are unequal. If the B string character is greater than the A string character, the TRUE/FALSE F.F.is left set otherwise it is reset. Memory access then continues until the repeat count is exhausted.

If the Repeat count is less than or equal to zero, the True/False F.F. is reset.

If either of the data strings has the memory protect bit on (bit 48=1) the segmented array interrupt is set and the operation is terminated.

All comparisons are by the binary character position in the collating sequence.

COMPARE CHARACTERS GREATER, UPDATE (CGTU) FA.

This operator performs a Compare Characters Greater operation. The source pointer and destination pointer are updated at the conclusion of the operator.

COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE (CGED) F1.
This operator performs the Compare Characters operation with the comparison being greater than or equal. If the repeat count $\leq$ 0, the True/False flip flop is set to zero.

COMPARE CHARACTERS GREATER OR EQUAL, UPDATE (CGEU) F9.
This operator performs a Compare Character Greater or Equal operation. The source pointer and destination pointer are updated at the conclusion of the operator.

COMPARE CHARACTERS EQUAL, DESTRUCTIVE (CEGD) F4.
This operator performs the Compare Characters operation using the Equal comparison. If the repeat count $\leq$ 0, then TFFF is set to one.

COMPARE CHARACTERS EQUAL, UPDATE (CEGU) FC.
This operator performs a Compare Characters Equal operation. The source pointer and destination pointer are updated at the conclusion of the operator.

COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE (CLED) F3.
This operator performs the Compare Characters operation with the Less than or Equal comparison. If the repeat count $\leq$ 0, then TFFF is set to zero.

COMPARE CHARACTERS LESS OR EQUAL, UPDATE (CLEU) FB.
This operator performs a Compare Characters Less or Equal operation. The source pointer and destination pointers are updated at the conclusion of the operator.

COMPARE CHARACTERS LESS, DESTRUCTIVE (CLSD) F0.
This operator performs the Compare Characters operation with the Less than comparison. If the repeat count $\leq$ 0, the TFFF is set to zero.

COMPARE CHARACTERS LESS, UPDATE (CLSU) F8.
This operator performs a Compare Characters Less operation. The source pointer and the destination pointer are updated at the conclusion of the operator.

COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE (CNED) F5.
This operator performs the Compare Characters operation using the Not equal relation. If the repeat count $\leq$ 0, then TFFF is set to zero.

COMPARE CHARACTERS NOT EQUAL, UPDATE (CNEU) FD.
This operator performs a Compare Characters Not Equal operation. The source pointer and the destination pointer are updated at the conclusion of the operator.

EDIT OPERATORS.
TABLE ENTER EDIT, DESTRUCTIVE (TEED) DO.
This operator is used to control edit micro-instructions. These edit micro-instructions are contained in memory as a table and not as part of the normal program string. Upon entering this operator program execution is transferred to a table of micro-instructions. The last micro-instruction in this table must be the End Edit operator (see section 9). The table contains Edit Mode operators.

The first item in the stack is a descriptor pointing to the table of Edit Micro-Instructions. The second item in the stack is a S.P. operand or a descriptor pointing at the source string. The third item in the stack is a descriptor pointing at the destination.

If the first item in the stack is not a descriptor the invalid operand interrupt is set and the operation is terminated.

If the second item in the stack is a S.P. operand it is the source string.

If the third item in the stack is not a descriptor the invalid operand interrupt is set and the operation is terminated.

If the destination pointer descriptor has the Read Only bit set to one (bit 43) the memory protect interrupt is set and the operation is terminated.

If the length is less than 13 the operand in the top of the stack is a single-precision operand. If the operand is 13 or greater the result is a double-precision operand.

If the length is not less than 25 an invalid operand interrupt is set and the operation terminated.

If the second item in the stack is an operand it is the source string, and is composed of 8-bit bytes.

If the source data has the memory protect bit (bit 48) set to one the segmented array interrupt is set and the operation is terminated.

PACK, UPDATE (PACU) D9.
This operator performs a Pack operation, updating the source pointer at the completion of the operator.

INPUT CONVERT OPERATORS.
INPUT CONVERT, DESTRUCTIVE (ICVD) CA.
This operator converts either 6-bit BCL code, 8-bit EBCDIC or 4-bit digit code to an operand for internal arithmetic operations.

The first item in the stack is an operand that is integerized to form the repeat field. The second item in the stack is a descriptor used as a source pointer.

The Input Convert operator calls on the Pack operator. After this operation is complete the 4-bit digit operand is converted to an operand of the equivalent binary value.

The sign of the converted operand is then set from the True/False flip flop. If the converted operand is a single-precision operand the True/False flip flop is then set to one. If the converted operand is a double-precision operand the True/False flip flop is set to zero.

At the completion of the operator the B register is marked full. The tag field is set to indicate either a single or a double-precision operand.

If the item in the top of stack after being integerized is greater than 23 the invalid operand interrupt is set and the operation is terminated.

INPUT CONVERT, UPDATE (ICVU) CB.
This operator performs an Input Convert operation. The source pointer is updated at the completion of the operator.

READ TRUE FALSE FLIP FLOP (RTFF) DE.
This operator places the status of the True/False flip flop into the low order bit position of the A register. The rest of the A register is set to all zeros. The A register is marked full at completion of this operator.

SET EXTERNAL SIGN (SXSN) D6.
This operator places the mantissa sign of the top word of the stack in the External Sign flip flop. This operand is not deleted from the stack at the end of the operation.

READ AND CLEAR OVERFLOW FLIP FLOP (ROFF) D7.
Places the status of the Overflow flip flop in the least significant bit of the A register, sets the rest of the A register to zero, marks the register full and sets the Overflow flip flop to zero.

SUBROUTINE OPERATORS.
VALUE CALL (VALC) 00 => 3F.
This operator loads into the A register the operand addressed by the address couple formed by the concatenation of the six low order bits of the first syllable and the eight bits of the following syllable. The A register is marked full. Figures 7-1 and 7-2 are simplified flow charts of the Value Call operator.

This operator makes multiple memory accesses if the word accessed is either an indexed descriptor, Program Control Word, or an Indirect Reference Word.

If the word accessed is an indexed Data Descriptor the word addressed by the Data Descriptor is brought to the top of the stack. If the double-precision bit (bit 40) in the Data Descriptor is equal to one, the other half of the double-precision operand is brought to the X register.

If the word accessed by the Data Descriptor is another indexed Data Descriptor the word addressed by that Data Descriptor is brought to the top of the stack, and the above paragraph is repeated.

If a Data Descriptor does not address an operand SIW or another indexed Data Descriptor an invalid operand interrupt is set and the operation is terminated.

If the word accessed by the Value Call is an Indirect Reference Word the word addressed by the IRW is accessed and evaluated. If the word is an operand it is placed in the top of the stack.

If the word accessed by the Indirect Reference Word is another IRW the operation continues as described above.

If the word accessed by the Indirect Reference Word is an indexed Data Descriptor the operator proceeds as described above for Data Descriptors.

If the word accessed by the Indirect Reference Word is a Program Control Word an accidental entry into the subroutine addressed by the PCW is initiated. A Mark Stack Control Word and a Return Control Word are placed in the stack and an entry into the program is made. Upon completion of the program a Return operator will re-enter the flow Value Call at the label IRW, figure 7-2. .

NAME CALL (NAMC) 40 => 7F.
This operator builds an Indirect Reference Word in the A register. The address couple is formed by concatenating the 6 low order bits of the first syllable and the 8 bits of the following syllable. The A register is marked full and the operation is complete.

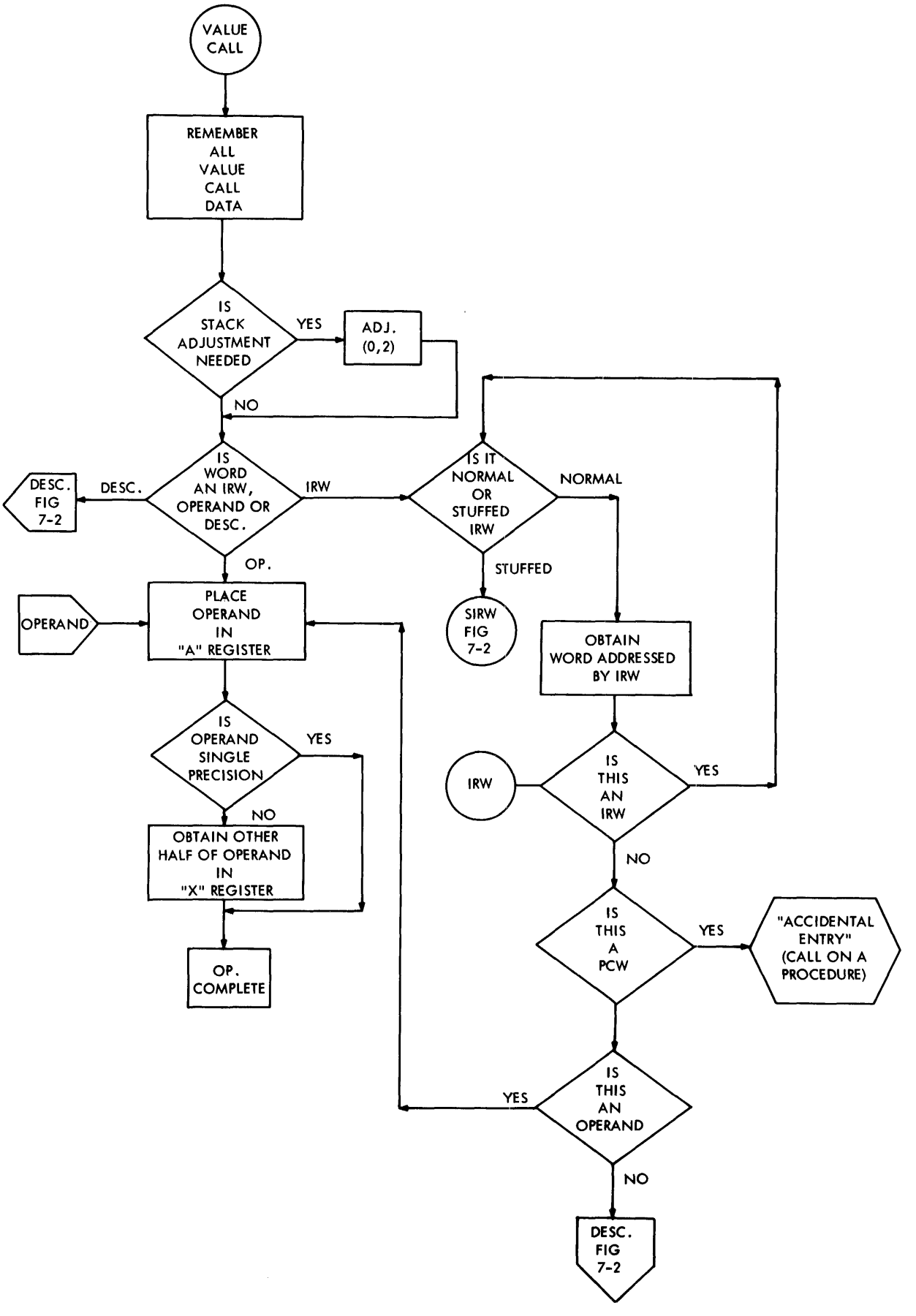


Figure 7-1. Flow of Value Call Operator

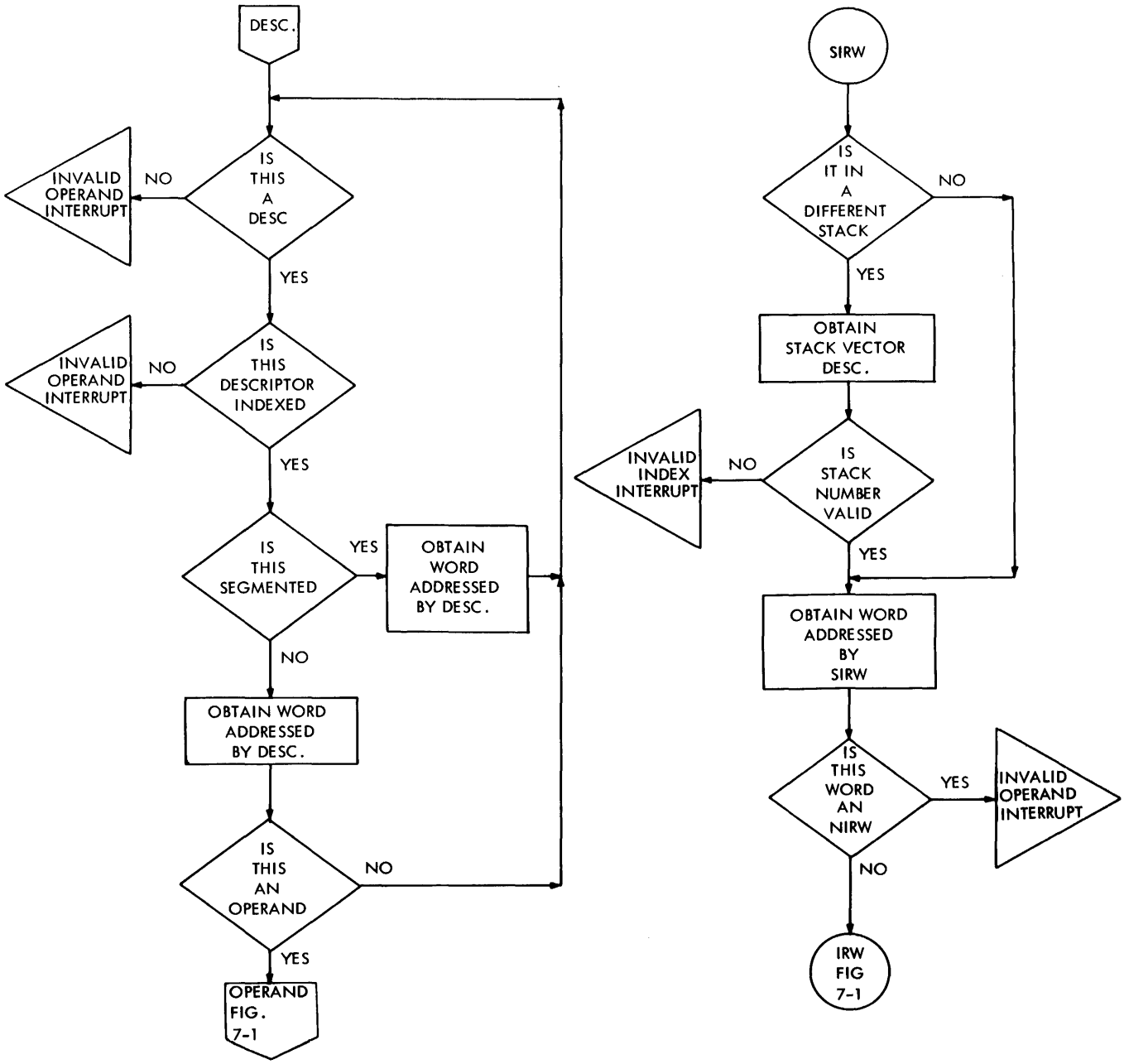


Figure 7-2. Flow of Value Call Operator (cont)

EXIT OPERATOR (EXIT) A3.
This operator returns to a calling procedure from a called procedure resetting all control registers from the Return Control Word and the Mark Stack Control Word. The Exit operator does not return a value to the calling routine. Figure 7-3 shows a simplified flow chart of the Exit operator.

RETURN OPERATOR (RETN) A7.
This operator performs an Exit operator with the exception that an operand or name in the B register is returned to the calling procedure. If a name is returned, and the V bit (bit 19) in the MSCW is on, the name is evaluated to yield an operand as described in VALC. Figure 7-4 shows a simplified flow chart of the Return operator.

ENTER OPERATOR (ENTR) AB.
This operator is used to cause an entry into a procedure from a calling procedure. Entry is to the program segment and syllable addressed by the Program Control Word. Figure 7-5 shows a simplified flowchart of the Enter operator.

The Enter operator accesses the Indirect Reference Word at F + 1 which points to the Program Control Word. The operator then builds a Return Control Word into the stack at F + 1.

EVALUATE (EVAL) AC.
This operator loads the A register with an indexed Data Descriptor or an Indirect Reference Word that addresses A "target", which may be an SIW, an Un-Indexed Data Desc, a String Desc, or an operand. The "target" may be referenced through a chain of descriptors, accidental entries, or Indirect Reference Words. In any case memory accesses will continue to be made until the target is located. The A register is left containing the Data Descriptor or the Indirect Reference Word which addresses the target. Figure 7-6 is a simplified flow chart of the Evaluate operator.

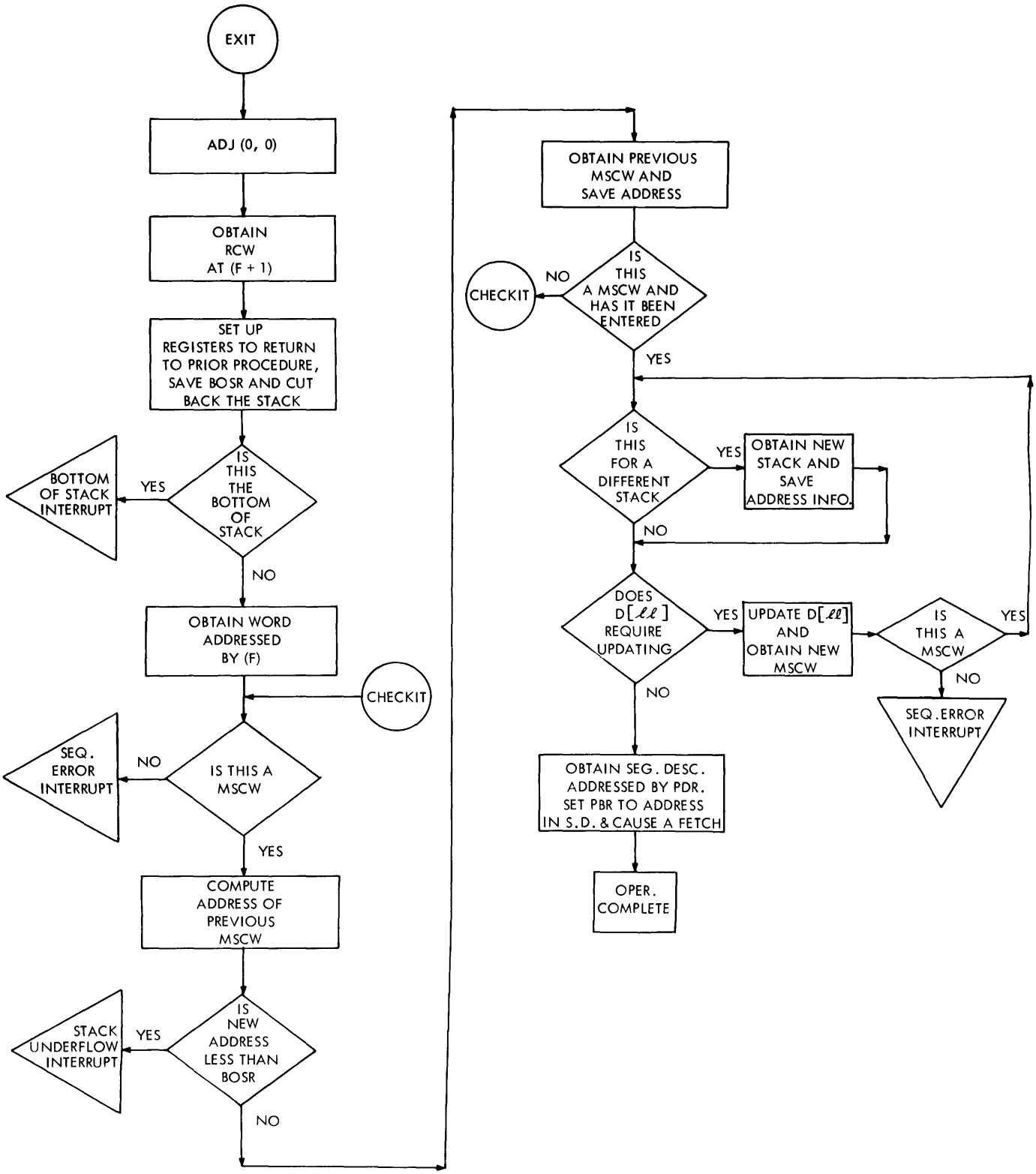


Figure 7-3. Flow of Exit Operator

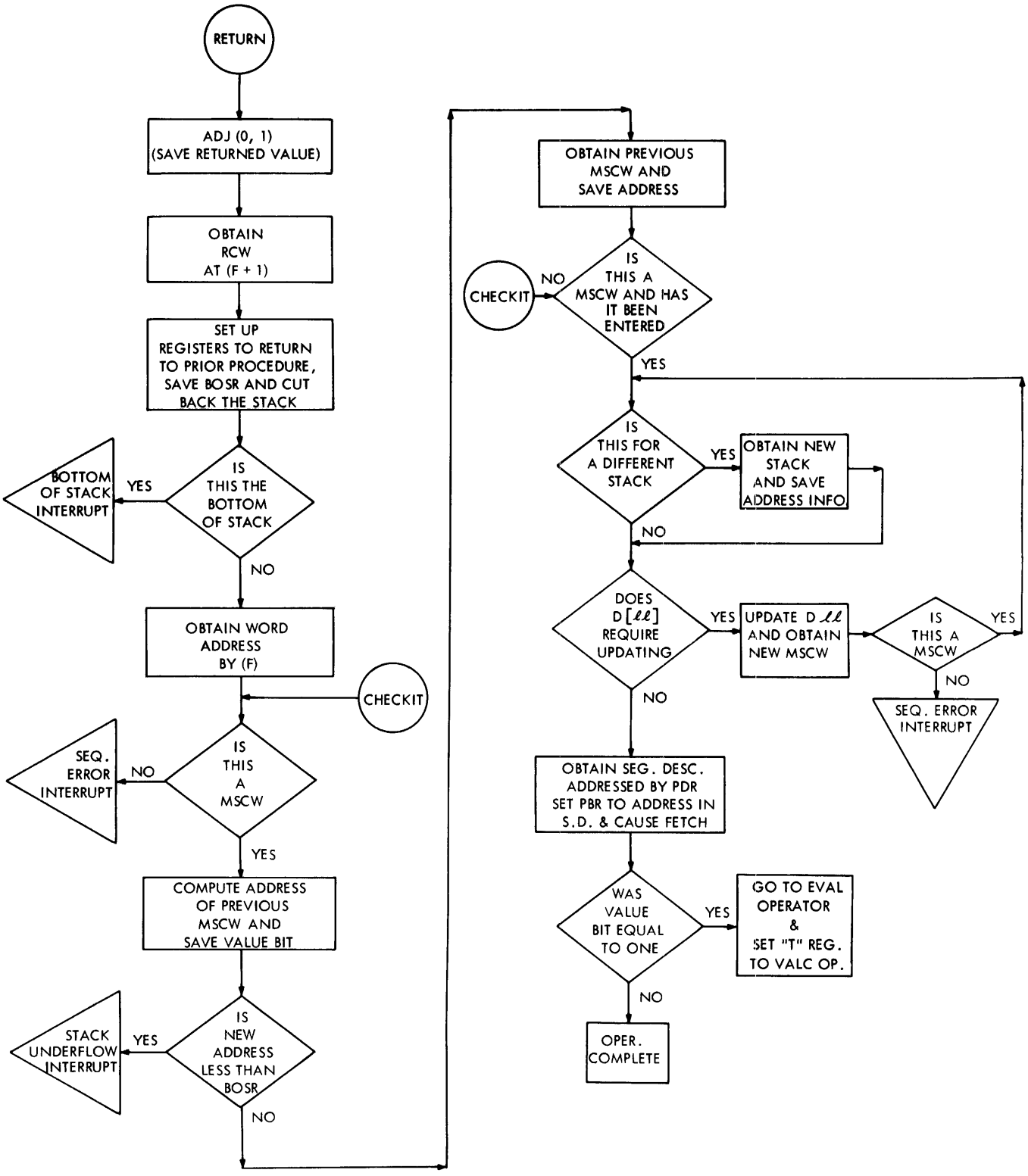


Figure 7-4. Flow of Return Operator

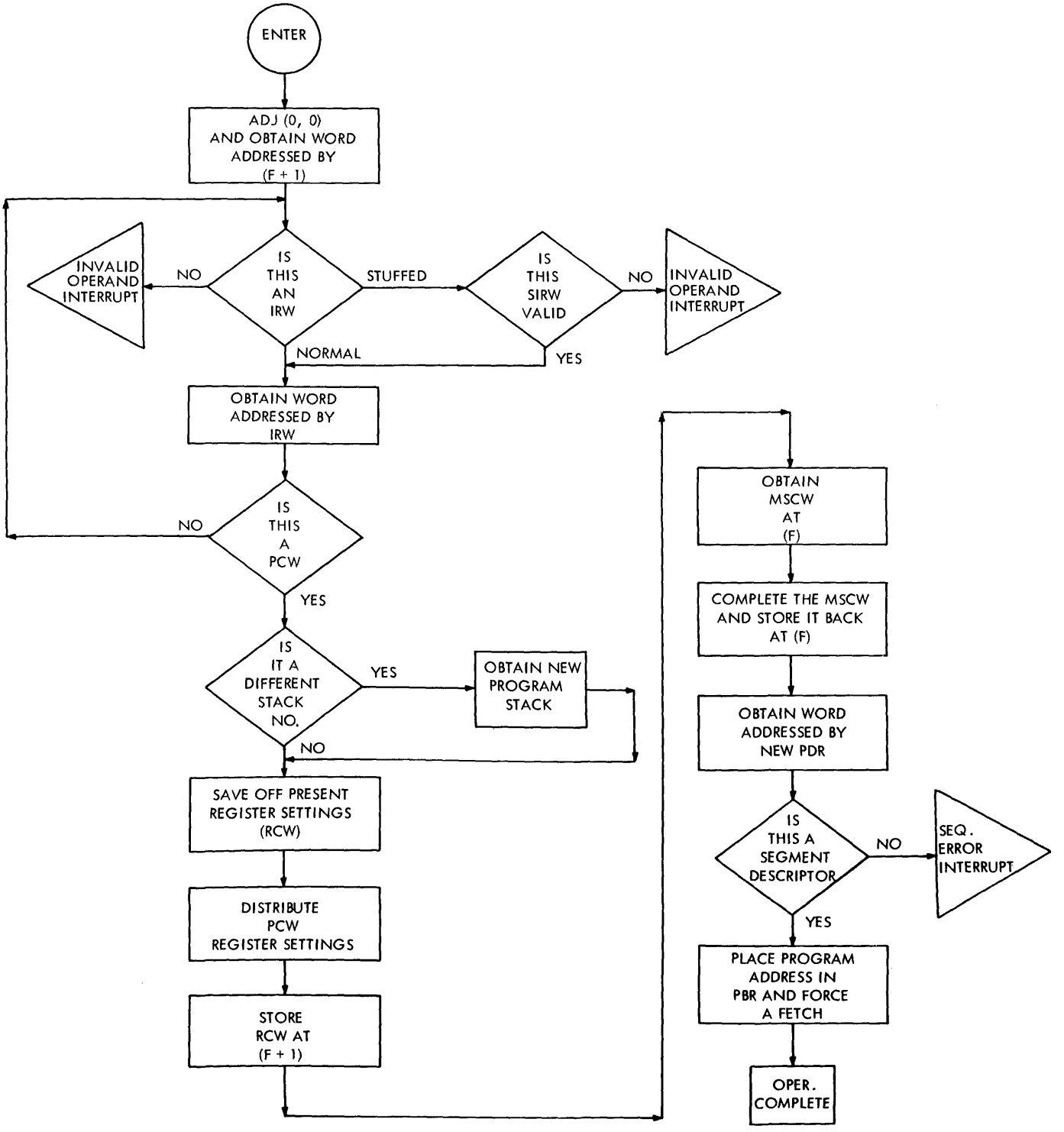


Figure 7-5. Flow of Enter Operator

An indexed Data Descriptor is left in the A register when the target is referenced by an indexed Data Descriptor; a stuffed Indirect Reference Word is left in the A register when the target is referenced by Indirect Reference Words.

If the A register does not contain a Data Descriptor or an Indirect Reference Word at the start of this operator an invalid operand interrupt is set and the operation is terminated.

MARK STACK OPERATOR (MKST) AE.
This operator places a Mark Stack Control Word in the B register containing a pointer to the previous Mark Stack Control Word in the stack. It adjusts the stack to push the MSCW into Memory.

This operator is used to mark the stack when entry into a procedure is anticipated.

STUFF ENVIRONMENT (STFF) AF.
This operator changes a normal Indirect Reference Word to a stuffed Indirect Reference Word so that a quantity may be referenced from a different addressing environment. The displacement field locates the MSCW below the quantity and the index field locates the quantity relative to the MSCW. Figure 7-7 shows a simplified flow chart of the Stuff Environment operator.

If the word in the A register at the start of the operator is not an Indirect Reference Word an invalid operand interrupt is set and the operation is terminated.

If when creating this stuffed IRW other than a MSCW is accessed a sequence error interrupt is set and the operation is terminated.

INSERT MARK STACK OPERATOR (IMKS) CF.
This operator builds a Mark Stack Control Word and places it below the two top-of-stack quantities.

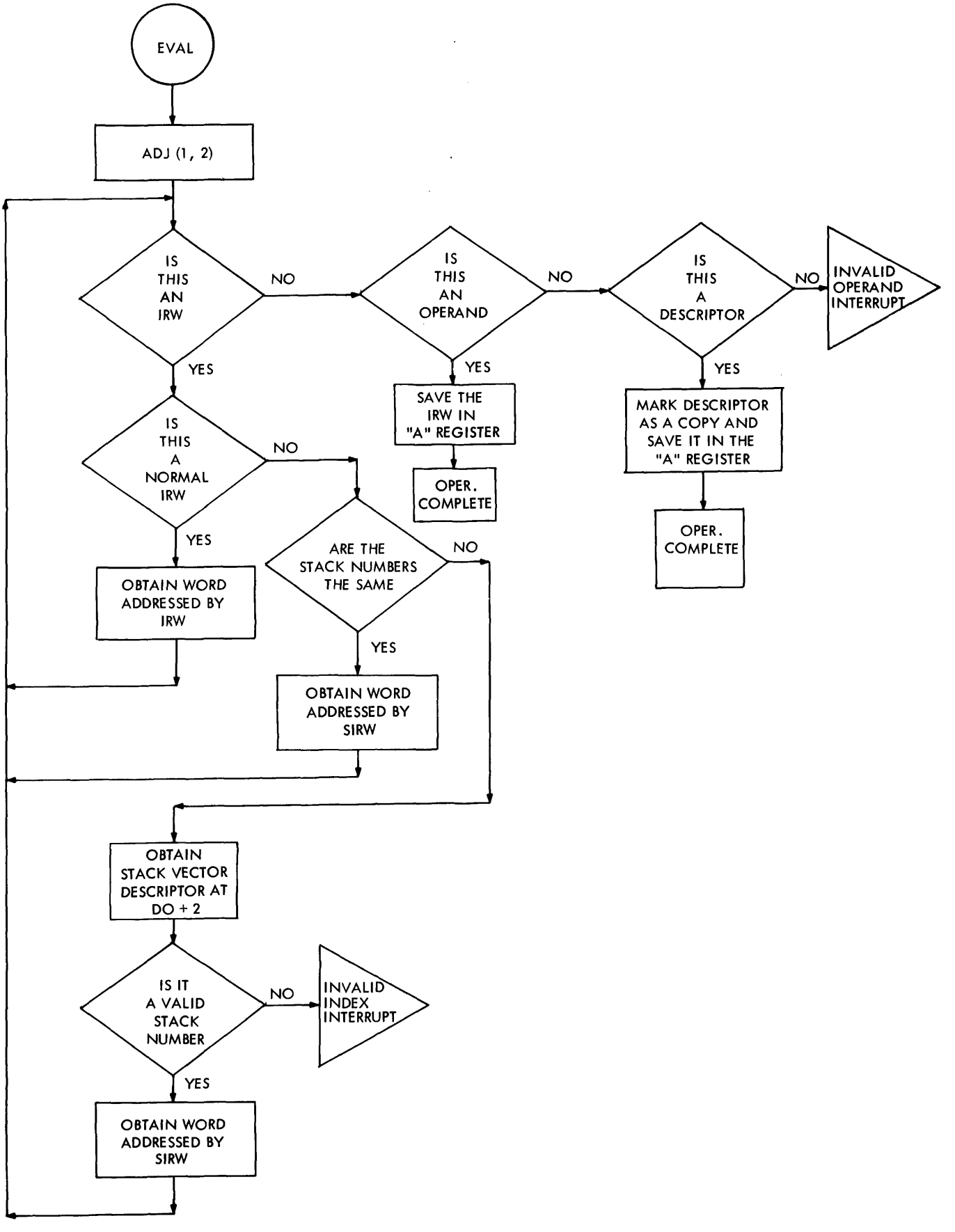


Figure 7-6. Flow of Evaluate Operator

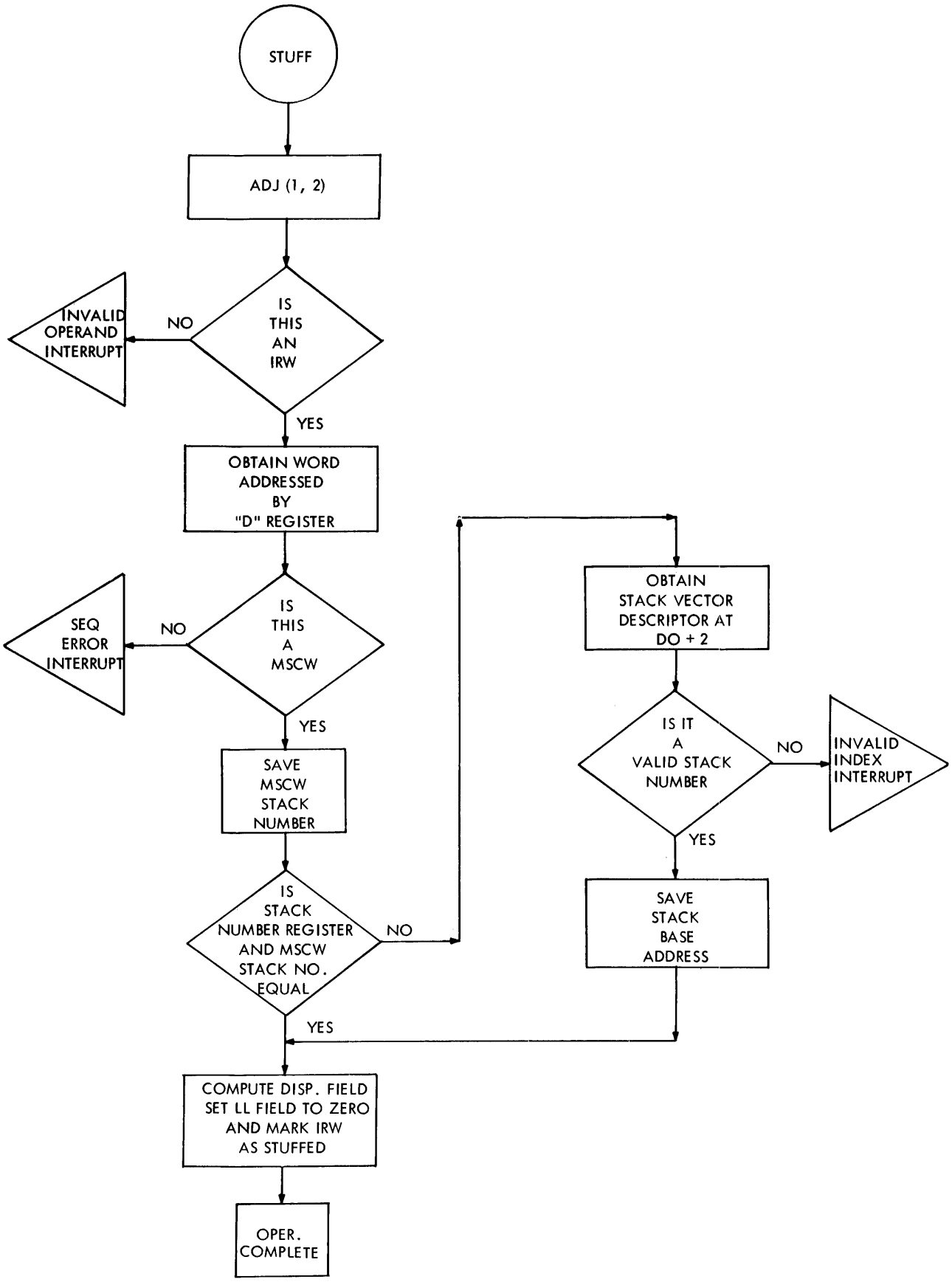


Figure 7-7. Flow of Stuff Environment Operator

SECTION 8
VARIANT MODE OPERATION AND OPERATORS

GENERAL.

The Variant Mode of operation extends the number of operation codes. These operators are not used as often and require two syllables; the first is the "Escape TO 16 Bit Instruction" (VARI) operator. When the VARI operator is encountered the following syllable is the actual operation and the syllable pointer is positioned beyond the two syllables. The VARI operator is valid only for the syllables covered in this section.

Variant codes E0 thru EF are detected and cause a programed operator interrupt. All other unassigned variant codes cause no action and result in a loop timer interrupt.

Variant Mode operations are both word and string-oriented operators.

OPERATORS.

SET TWO SINGLES TO DOUBLE (JOIN) 9542.

The operands in the A and B registers are combined to form a double-precision operand that is left in the B and Y registers.

The operand in the A register is placed in the Y register. The A register is marked empty and the B register tag field is set to double-precision.

SET DOUBLE TO TWO SINGLES (SPLT) 9543.

The SP(DP) operand in the B register is changed to two single-precision operands which are placed in the A and the B registers, both registers are marked full.

If the operand in the B register is a single operand, the A register is set to zero and the A and B registers are marked full. Both the A and the B register tag fields are set to single-precision.

If the operand in the B register is a double-precision operand the Y register operand is placed in the A register and the tag fields of both the A and B registers are set to single-precision.

IDLE UNTIL INTERRUPT (IDLE) 9544.
This operator suspends processor program execution until restarted by an external interrupt. IIHF is reset to allow external interrupts.

SET INTERVAL TIMER (SINT) 9545. (Control State Oper.)
This operator places the 11 low-order bits of the B register into the Interval Timer register, and arms the timer. The Interval Timer decrements each 512 microseconds, interrupting the processor when it reaches zero and is still armed. The Interval Timer is disarmed when the processor is interrupted by an external interrupt.

The operand used to set the Interval Timer is integerized before the 11 low-order bits are used. If the operand can not be integerized an integer overflow interrupt is set and the operation is terminated.

ENABLE EXTERNAL INTERRUPTS (EEXI) 9546.
This operator causes the processor to enter normal state allowing it to respond to external interrupts. This is accomplished by setting the Normal-Control State (NCSF) and the Interrupt Inhibit (IIHF) flip flops to zero.

DISABLE EXTERNAL INTERRUPTS (DEXI) 9547.
This operator causes the processor to ignore external interrupts. This is accomplished by setting the Interrupt Inhibit flip flop to one and entering control state.

SCAN OPERATORS.
The Scan operators communicate between the processor and the I/O Data Com., or General Control Subsystems via a two section scan bus. One section consists of 32 address and control lines and the other section, 48 data lines. The Scan-In functions read information

from the subsystem to the top of stack register in the processor. The Scan-Out functions write information from the top of stack registers in the processor to the subsystem.

Parity is checked during transmission of both address and information, and a SCAN-BUS parity error interrupt is generated if the check fails.

READ TIME OF DAY CLOCK.
This operation transfers the time-of-day register from the multiplexor to the B register. It is important to note that if the system has multiple multiplexors only one time-of-day clock is active. MPX A responds when a multiplexor is not designated.

As this operation is initiated, the A register contains the code word shown in figure 8-1.

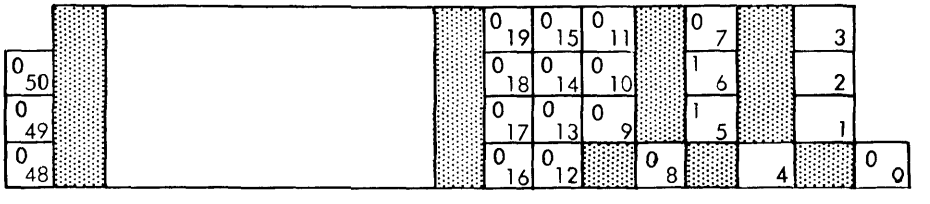


Figure 8-1. Read Time-Of-Day Code Word

The time-of-day word resulting from this operation is shown in figure 8-2. The B register is marked full and the A register is marked empty at the completion of this operation.

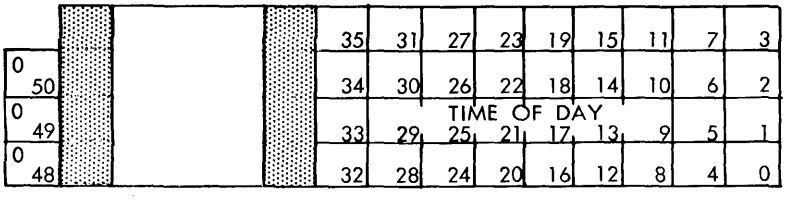


Figure 8-2. Time of Day Word

READ GENERAL CONTROL ADAPTER.

This operation places the contents of one of the three general control registers into the B register. Figure 8-3 shows the format of the function code word that is present in the A register as the operation is initiated.

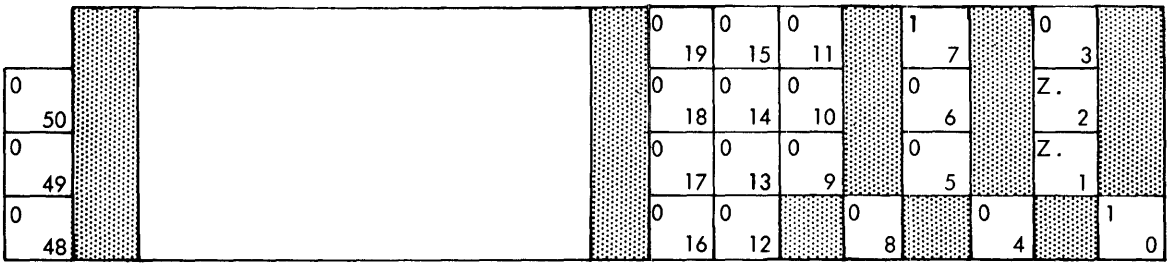


Figure 8-3. Read General Control Adapter Code Word

There are four General Control designations:

a. Z = 0001, GCA A
b. Z = 0010, GCA B
c. Z = 0100, GCA C
d. Z - 1000, GCA D

The N field can address or read one of four, 48-bit general control adapter registers. The registers and their addresses are:

a. N = 00, Input Register.
b. N = 01, Interrupt Mask.
c. N = 10, Interrupt Register.
d. N = 11, Output Register.

The A register is marked empty, the B register contains the word read from the general control adapter and is marked full as this operation is completed.

READ RESULT DESCRIPTOR.

This operation places a result descriptor into the B register from the multiplexor specified. The A register contains the code word shown in figure 8-4.

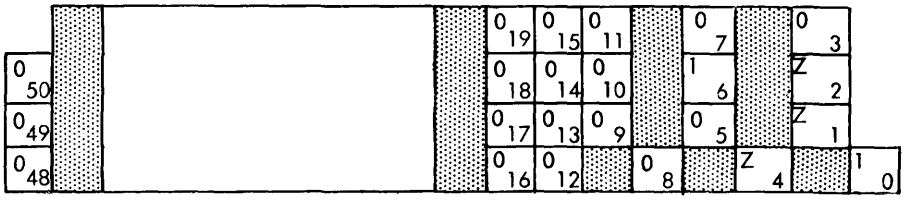


Figure 8-4. Read Result Descriptor Code Word

Multiplexor designations are:

a. Z = 0001, MPX A
b. Z = 0010, MPX B

At the completion of this operation the B register contains the result descriptor shown in figure 8-5. The B register is marked full and the A register is marked empty. The result is not defined if the multiplexor has no result descriptor.

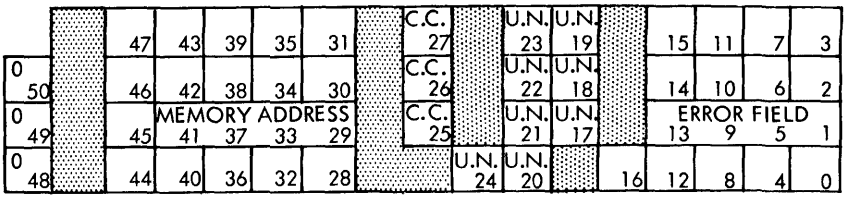


Figure 8-5. Result Descriptor

The result descriptor error field is divided into a standard error field and unit error field. The unit error field bit assignments are defined individually for each peripheral control:

a. Bit 0, Exception.
b. Bit 1, Attention.
c. Bit 2, Busy.
d. Bit 3, Not ready.
e. Bit 4, Descriptor Error.
f. Bit 5, Memory Address
g. Bit 6, Memory Parity Error.
h. Bit 16, Memory Protect.

The "U.N." field in figure 8-5 is the unit number field. The "C.C." field is the character count field.

READ INTERRUPT MASK.
This operation places the interrupt mask word into the B register from the multiplexor specified. The A register contains the code word shown in figure 8-6.

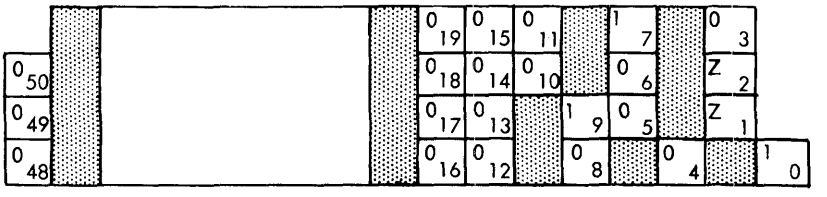


Figure 8-6. Read Interrupt Mask Code Word

a. Z = 0001, MPX A
b. Z = 0010, MPX B

The B register contains the interrupt mask word as shown in figure 8-7 at the completion of this operator. The B register is marked full, the A register is marked empty.

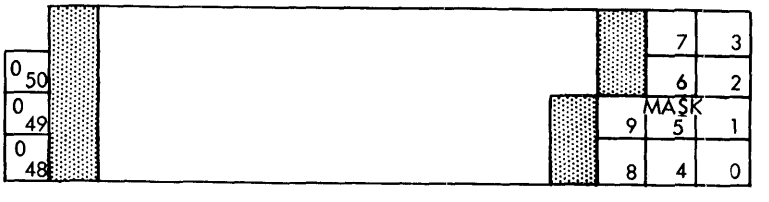


Figure 8-7. Interrupt Mask Word

The mask bit assignments are:

a. Bit 0, Status Change.
b. Bit 1, D.C.P. - 1.
c. Bit 2, D.C.P. - 2.
d. Bit 3, D.C.P. - 3.

e. Bit 4, D.C.P. - 4.

f. Bit 9, I/O finished.

The bit is set in the interrupt mask if recognition of the interrupt is inhibited.

READ INTERRUPT REGISTER.

This operation places an interrupt register word into the B register from the multiplexor specified. The A register contains the code word shown in figure 8-8.

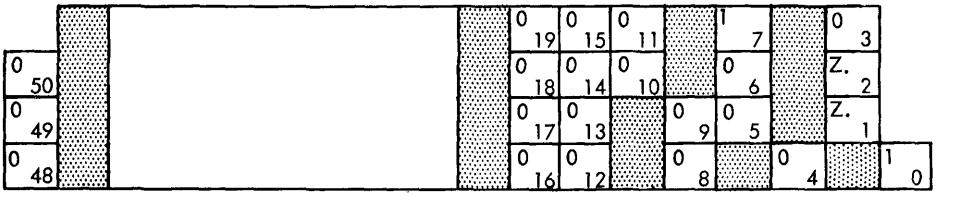


Figure 8-8. Read Interrupt Register Code Word

a. Z = 0001 MPX A

b. Z = 0010 MPX B

The B register contains the interrupt register word as shown in figure 8-9 at the completion of this operation and is marked full, the A register is marked empty.

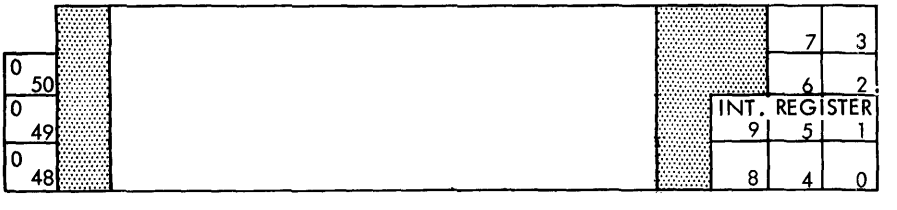


Figure 8-9. Interrupt Register Word

The interrupt register bit assignments are:

a. Bit 0, Status Change.
b. Bit 1, D.C.P. - 1.
c. Bit 2, D.C.P. - 2.
d. Bit 3, D.C.P. - 3.
e. Bit 4, D.C.P. - 4.
f. Bit 9, I/O Finish.

The bit is set in the Interrupt Status Register if the interrupt is pending.

READ INTERRUPT LITERAL.
This function places the interrupt literal word into the B register from the multiplexor specified. The A register contains the code word shown in figure 8-10.

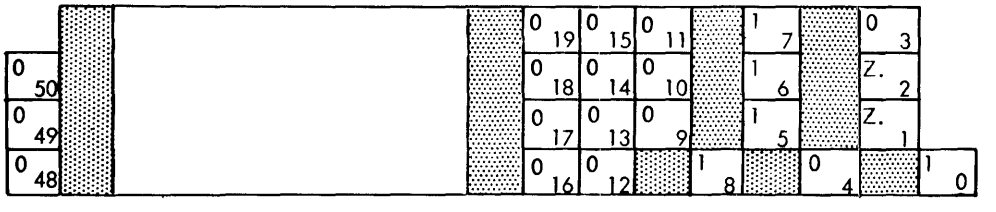


Figure 8-10. Read Interrupt Literal Code Word

Multiplexor designations are:

a. Z = 0001 MPX A
b. Z = 0010 MPX B

At the completion of this operation the B register contains the interrupt literal word as shown in figure 8-11 and is marked full, the A register is marked empty.

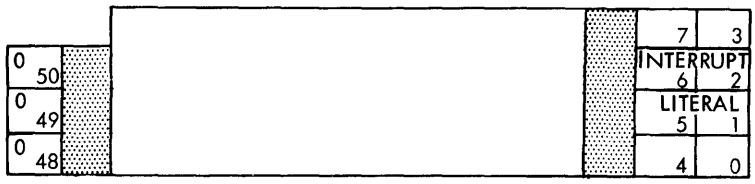


Figure 8-11. Interrupt Literal Word

The interrupt literal bit assignments are:

a. Bits 3(4), 0001 = MPX A.
0010 = MPX B.

b. Bits 7(4), 0001 = D.C.P. - 1.
0010 = D.C.P. - 2.
0011 = D.C.P. - 3.
0100 = D.C.P. - 4.
1001 = Multiplexor I/O finished.
1111 = Status Change.

INTERROGATE PERIPHERAL STATUS.

This operation places one of eight possible status vector words into the B register from one of the multiplexors. A B 6500 may have up to 256 peripheral units designated in the system. This configuration requires eight status vector words, each indicating the ready status of 32 units. Vector word 0 interrogates the status of units 0 through 31, vector 1 the status of units 32-63, etc. The A register contains the code word shown in figure 8-12.

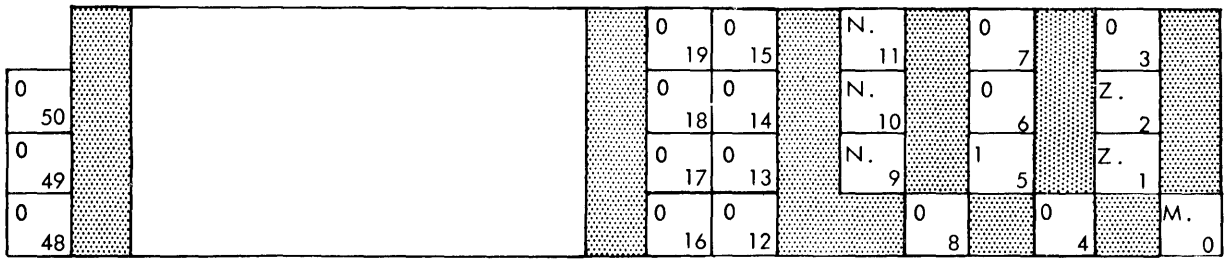


Figure 8-12. Interrogate Peripheral Status Code Word

Multiplexor designations are:

a. Bit 0 ; M = 0, All multiplexors are to respond.
M = 1, Multiplexor designated by Z to respond.

b. Bit 4(4); Z = 0001 MPX A
Z = 0010 MPX B

c. Bits 11(3), N = Status vector number, 0 thru 7.

At completion of this operation, the B register contains the status vector word addressed by the value of N with the status vector word in a format shown in figure 8-13. The B register is marked full and the A register is marked empty.

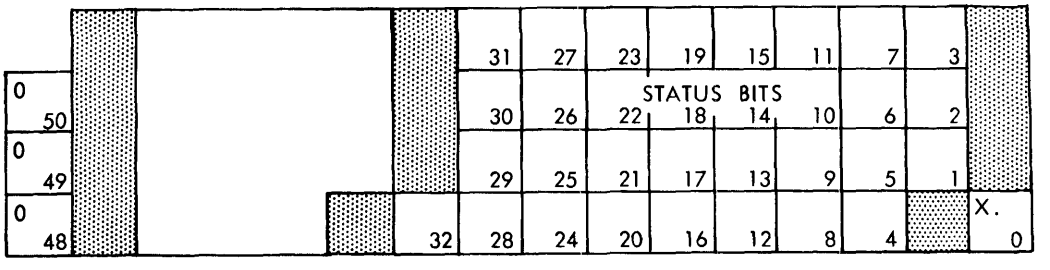


Figure 8-13. Status Vector Word

A status-change bit is assigned to each line printer or display unit and indicates completion of paper-motion or input request.

The X bit in the status vector word is on if the word is valid.

INTERROGATE PERIPHERAL UNIT TYPE.

This operation places the peripheral unit type word into the B register from one of the multiplexors. The A register contains the code word shown in figure 8-14.

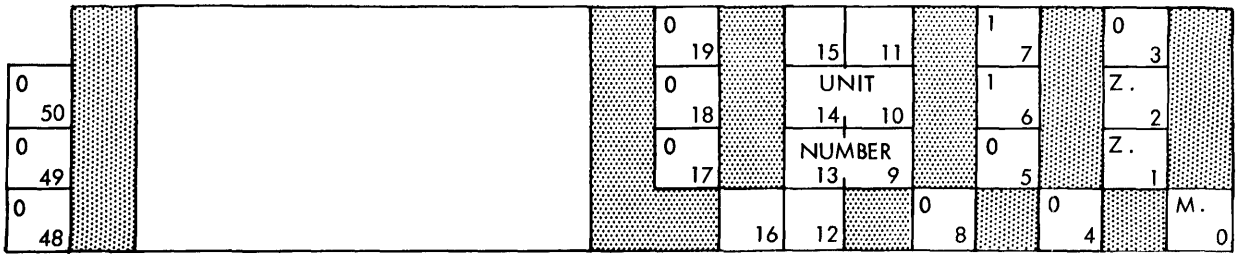


Figure 8-14. Interrogate Peripheral Unit Type Code Word

a. M = 0, All multiplexors to respond.

b. M = 1, Multiplexor designated by Z respond.

When M = 1, the Z field MPX designations are:

a. Z = 0001, MPX A.
b. Z = 0010, MPX B.

Upon completion of this operator the B register contains the peripheral unit type word as shown in figure 8-15 and is marked full; the A register is marked empty.

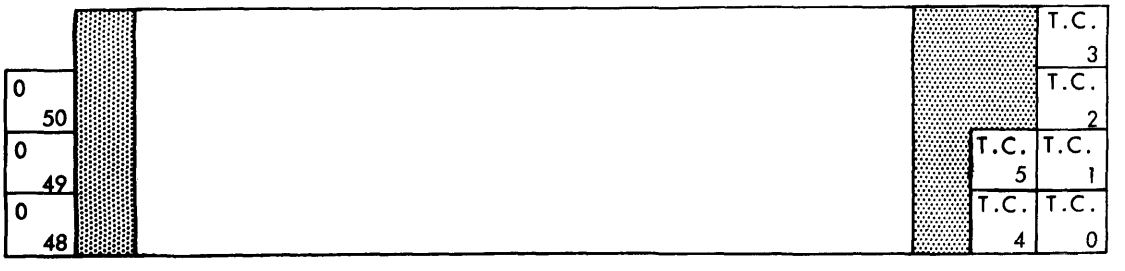


Figure 8-15. Unit Type Code Word

The following codes identify the units:

	Code	Unit
a.	00	No unit.
b.	01	Disk File.
c.	02	Display.
d.	04	Paper-Tape Reader.

	Code	Unit	
e.	05	Paper-Tape Punch.	
f.	06	Buffered, Line-Printer I, BCL Drum.	
g.	07	Unbuffered, Line-Printer, BCL Drum.	
h.	09	Card Reader.	
i.	0B(11)	Card Punch.	
j.	0D(13)	Magnetic Tape(7 track).	With status vector information.
k.	0E(14)	Magnetic Tape (9 track N.R.Z.).	
l.	0F(15)	Magnetic Tape (9 track P.E.).	
m.	1D(29)	Magnetic Tape (7 track).	No status vector information.
n.	1E(30)	Magnetic Tape (9 track N.R.Z.).	
o.	1F(31)	Magnetic Tape (9 track P.E.).	
p.	26(38)	Buffered Line-Printer, EBCDIC-subset drum.	
q.	27(39)	Unbuffered Line-Printer, EBCDIC-subset drum.	

INTERROGATE I/O PATH.

This operation determines the availability or absence of an access to a specified unit. The result word is placed in the B register. The A register contains the code word shown in figure 8-16.

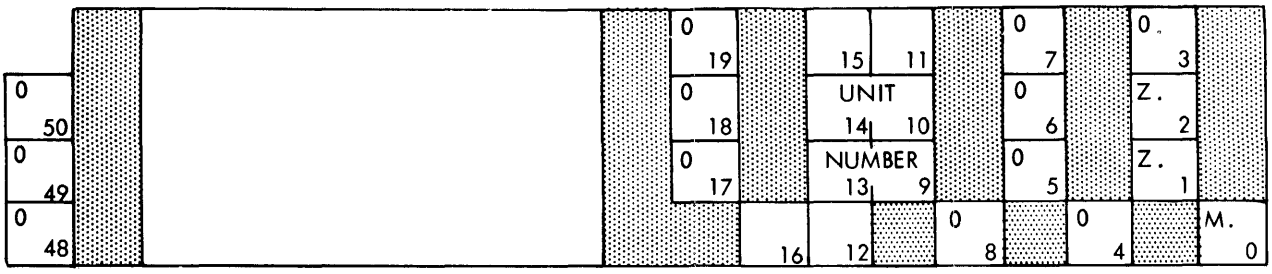


Figure 8-16. Interrogate I/O Path Code Word

Primary Multiplexor designations are:

a. M = 0, All multiplexors respond.

b. M = 1, Multiplexor designated by Z to respond.

Multiplexors designations with M=1 are:

a. Z = 0001, Multiplexor A.
b. Z = 0010, Multiplexor B.

At the completion of this operation the B register contains the result word shown in figure 8-17 and is marked full; the A register is marked empty.

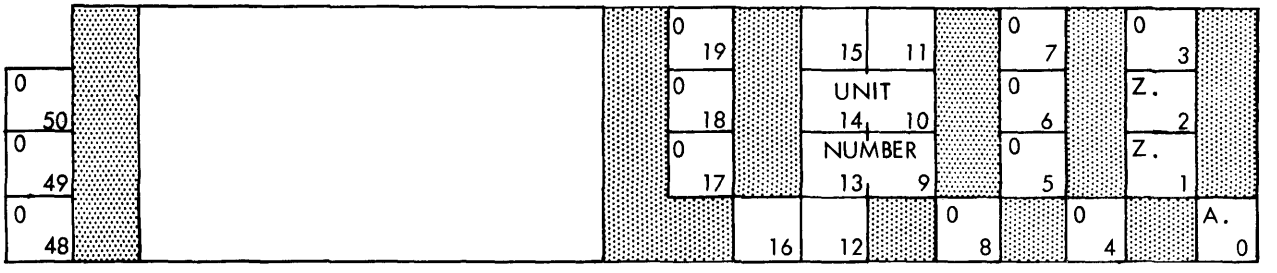


Figure 8-17. I/O Path Result Word

The A bit indicates path availability:

a. A = 0, No path available.
b. A = 1, Path is available.

The Z field identifies the multiplexor when a path is available:

a. Z = 0001, Path is via multiplexor A.
b. Z = 0010, Path is via multiplexor B.

A data channel consists of a data switching channel and a peripheral control.

SCAN OUT (SCNO) 954B.
Scan Out places bits 0-19 of the top-of-stack word on the scan-bus address lines, and the second stack word on the scan-bus information lines. An Invalid Address interrupt results if the address-word is invalid. The A and B registers are empty upon successful completion of a Scan-Out.

SET TIME OF DAY CLOCK.

This operation transfers the time of day information from the B register to the time of day register in the multiplexor (figure 8-19). The code word shown in figure 8-18 is in the A register. MPX A responds when a multiplexor is not designated. An invalid-operand interrupt results if the processor is not in control state.

At the completion of this operation the A and B registers are marked empty.

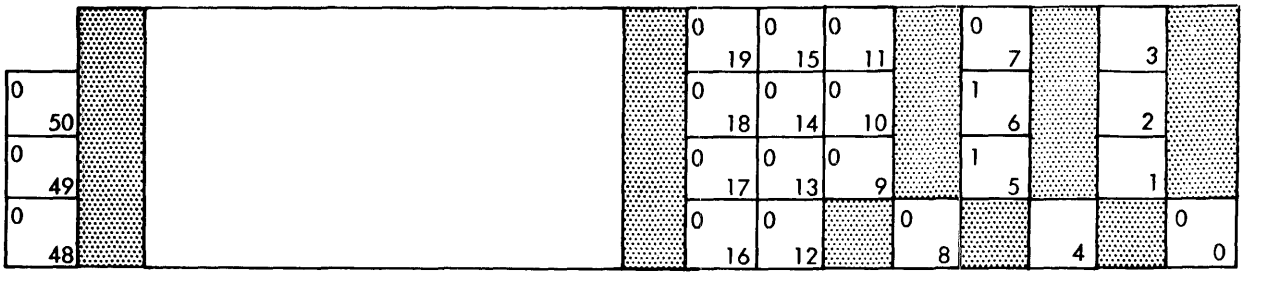


Figure 8-18. Set Time of Day Clock Code Word

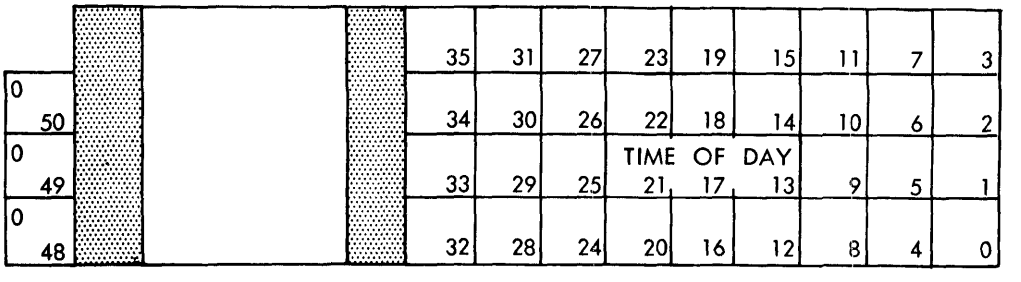


Figure 8-19. Time of Day Word

SET GENERAL CONTROL ADAPTER.

This operation sets one of three addressable general control adapter registers from the word in the B register. The three general control adapter registers that can be set are the output register, interrupt mask register and the interrupt register.

The A register contains the code word shown in figure 8-20 and the B register contains the output, the interrupt mask or the interrupt word.

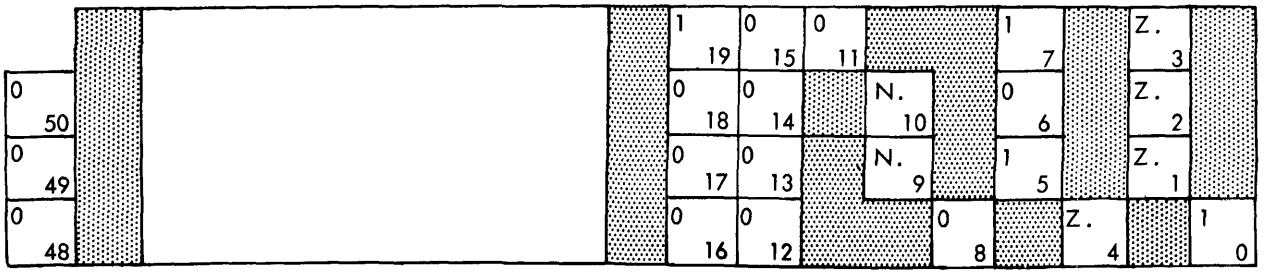


Figure 8-20. Set General Control Adapter Code Word

Multiplexor designations are:

a. Z = 0001, MPX A.
b. Z = 0010, MPX B.

Output, interrupt, or interrupt mask register designations are:

a. N = 00, Output.
b. N = 01, Interrupt mask register.
c. N = 10, Interrupt register.

At the completion of this operator both the A and B registers are marked empty.

INITIATE I/O. (Control State Only).
This operation initiates an I/O unit specified by the code word in the A register. The code word format is shown in figure 8-21.

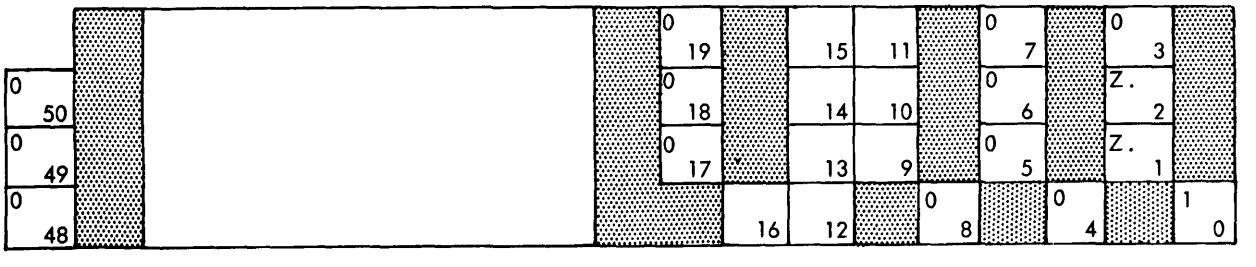


Figure 8-21. Initiate I/O Code Word

The B register holds the area descriptor and has the format shown in figure 8-22. The area descriptor points to the base address of the I/O area where the I/O control word is located (figure 8-23).

At completion of this operator the A and B registers are marked empty.

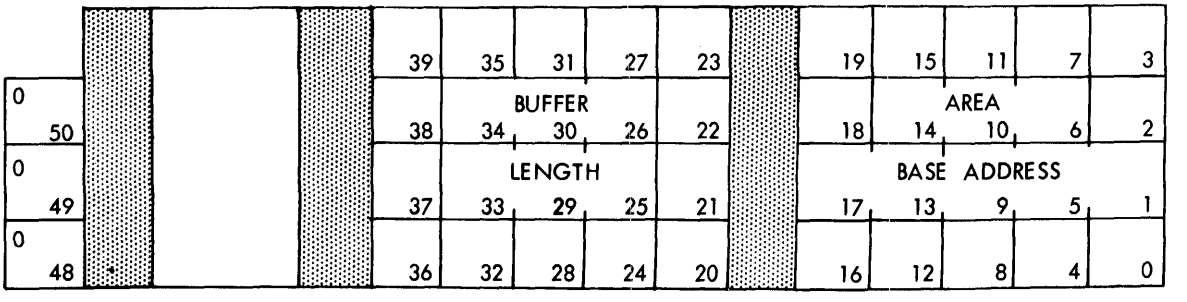


Figure 8-22. Area Descriptor

The I/O control word pointed to by the area descriptor is transferred to the multiplexor. It is divided into a standard control field and a unit control field. The unit control field bit assignments are defined individually for each control:

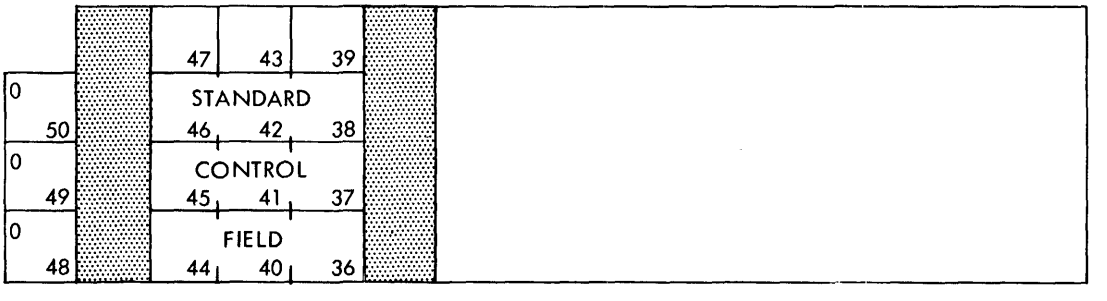


Figure 8-23. I/O Control Word

	Bit	Assignment	Bit=0	Bit=1
a.	47,	Reserved	-----	-----
b.	46,	Reserved	-----	-----
c.	45,	Attention	No	Yes
d.	44,	Read/write	write	read
e.	43,	Memory Inhibit	No	Yes
f.	42,	Translate	No	Yes
g.	41,	Frame length	6-bit	8-bit
h.	40,	Memory protect	No	Yes
i.	39,	Backward transfer	No	Yes
j.	38,	Test	No	Yes
k.	37-36,	Tag field transfer	37=1	36=0
l.	37-36,	Store program tag	37=0	36=1
m.	37-36,	Store single-precision tag	37=0	36=0
n.	37-36,	Store double-precision tag	37=1	36=1

READ PROCESSOR IDENTIFICATION (WHOI) 954E.
This operator places in the A register a single-precision operand containing the value of the processor ID register. The register is marked full.

INTERRUPT OTHER PROCESSOR (HEYU) 954F.
This operator sets the processor interrupt register of the other processor.

OCCURS INDEX (OCRX) 9585.
This operator places in the B register a new index value calculated from the Index Control Word (ICW) in the A register (figure 8-24) and the operand in the B register (figure 8-25).

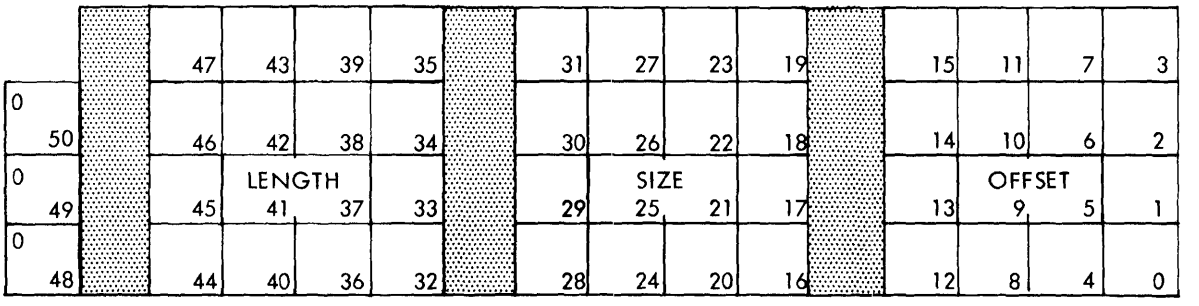


Figure 8-24. Index Control Word

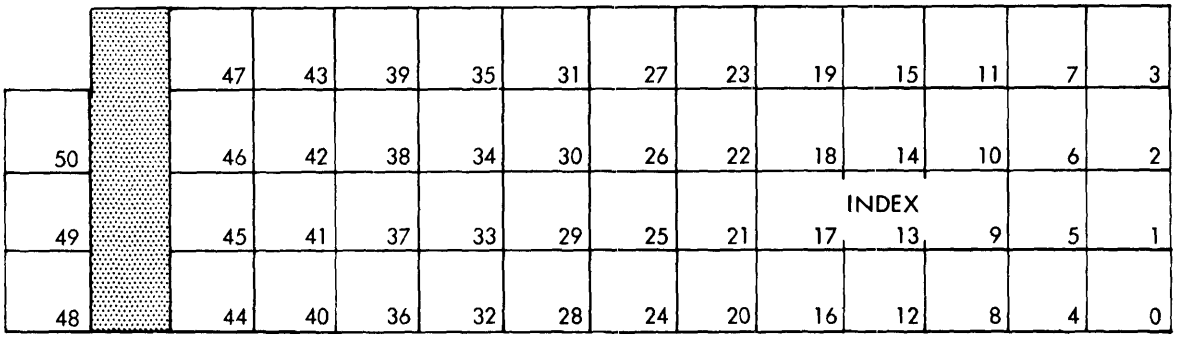


Figure 8-25. Index Word

The index word in the B register is integerized: if the index is greater than the maximum integer value (549,755,813,887) the integer overflow interrupt is set and the operation terminated.

The length field of the ICW is multiplied by the index value minus 1, and that value is added to the offset field of the ICW. This result is the new index. The A register is marked empty and the B register is marked full.

If either the ICW or the operand has a value of zero, the invalid-index interrupt is set and the operation is terminated.

If the index value is less than zero or greater than the size field of the ICW, the invalid-index interrupt is set and the operation is terminated.

INTEGERIZED, ROUNDED, DOUBLE-PRECISION (NTGD) 9587.
This operator creates a double-precision, rounded integer in the B register from the operand in the B register. The B register is marked full. If the word in the B register at the start of this operator is not an operand, the invalid-operand interrupt is set and the operation is terminated.

If the operand in the B register is larger than 8↑26-1 in absolute value, the integer-overflow interrupt is set and the operation is terminated.

The B register is marked as a double-precision operand (tag bits set to 010) and the exponent is set to 13.

LEADING ONE TEST (LOG2) 958B.
This operator locates the most significant "one" bit of the word in the B register and places the location of that bit into the B register (bit number + 1).

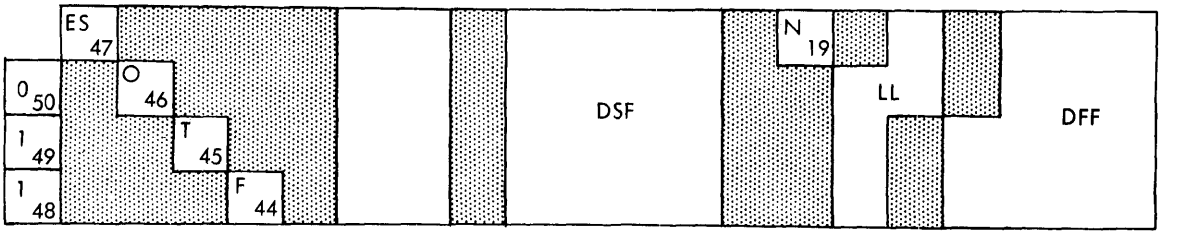
If a one bit is not sensed the B register is set to zero.

The B register is marked full.

MOVE TO STACK (MVST) 95AF.
This operator causes the processor's environment (or addressing space) to be moved from the current stack to the program stack specified by the operand in the B register.

The operator builds a Top of Stack Control Word (figure 8-26) and places it at the base of the current stack as addressed by the Base of Stack Register.

The operand in the B register is integerized and checked for invalid index against the stack vector. The value in the B register is added to the address field of the stack vector Descriptor (at D[0]+2), to address the descriptor for the new stack.



ES - EXTERNAL SIGN FLIP-FLOP
O - OVERFLOW FLIP-FLOP
T - TOGGLE, TRUE-FALSE FLIP-FLOP
F - FLOAT FLIP-FLOP
DSF - DELTA S-REGISTER FIELD; VALUE OF rS RELATIVE TO BOSR
N - NORMAL-CONTROL STATE FLIP-FLOP
LL - ADDRESSING LEVEL
DFF - DELTA F-REGISTER FIELD; VALUE OF rF RELATIVE TO rS

Figure 8-26. Top of Stack Control Word (TSCW)

The Data Descriptor for the requested stack is accessed. If its presence bit is on, the address field is placed into the Base of Stack Register. The Top of Stack Control Word is brought up and the stack is marked "active" by storing the processor ID at the base of the stack. The TSCW is distributed and the D registers are updated.

If during the integerization the operand in the B register is too large, the integer-overflow interrupt is set and the operation is terminated.

If the index value is less than zero or greater than the length field of the Data Descriptor for the stack vector array, an invalid index interrupt is set and the operation is terminated.

SET TAG FIELD (STAG) 95B4.
This operator sets the tag field (bits 50:3) in the B register to the value of bits 2:3 of the operand in the A register. At the completion of the operation the A register is marked empty and the B register is left full.

READ TAG FIELD (RTAG) 95B5.
This operator replaces the word in the A register with a single-precision operand equal to the tag field of that word. The tag bits are placed in bits 2:3. The A register is marked full.

ROTATE STACK UP (RSUP) 95B6).
This operator permutes the top three operands of the stack so that the first operand has become second, the second has become the third, and the third has become the first (see figure 8-27).

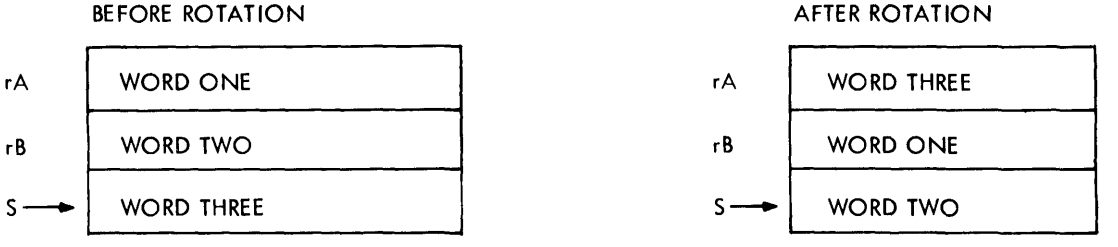


Figure 8-27. Stack Rotation Up

ROTATE STACK DOWN (RSDN) 95B7.
This operator permutes the top three operands of the stack so that the first has become third, the second has become the first, and the third has become the second (see figure 8-28).

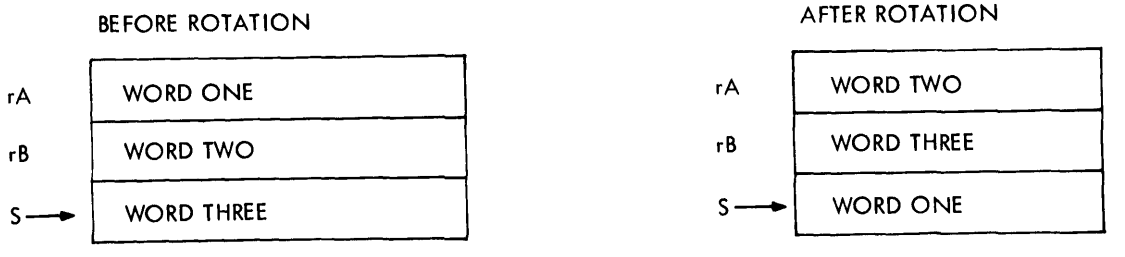


Figure 8-28. Stack Rotation Down

READ PROCESSOR REGISTER (RPRR) 95B8.
This operator reads the contents of one of the eight Base registers, eight Index registers or one of the 32 D registers into the A register.

The six low order bits of the A register selects the processor register to be read.

The decoding of these six bits is as follows:

a. Bits 5 & 4 = 10 = Index Register

b. Bits 2:3 = 0, = PIR

= 1, = SIR
= 2, = DIR
= 3, = TIR, BUF3
= 4, = TOSR
= 5, = BOSR
= 6, = F
= 7, = BUF

c. Bits 5 & 4 = 11 = Base Register

d. Bits 2:3= 0, = PBR

= 1, = IBR
= 2, = DBR
= 3, = TBR, BUF2
= 4, = S
= 5, = SNR
= 6, = PDR
= 7, = TEMP

If Bit 5 is zero, bits 4:5 select the D register equal to the binary value of the bits. (i.e., Bits 4:5 = 00101 selects D register 5.)

The A register at the completion of this operation contains the contents of the register that was selected and is marked full.

SET PROCESSOR REGISTER (SPRR) 95B9.
This operator places the contents of the address field of the A register into one of the eight Base registers, eight Index registers or 32 D registers selected by the six low-order bits of the word in the B register.

The decoding of the six bits is the same as in the Read Processor Register operator.

The A and B registers are marked empty.

READ WITH LOCK (RDLK) 95BA.

This operator performs the same operation as the Overwrite operator (see section 7) with the exception that the word which was in memory before the overwriting is left in the B register.

COUNT BINARY ONES (CBON) 95BB.
This operator counts the number of one bits in the S.P.(D.P.) operand in the A register. At the completion of the operation the total count is left in the A register with the register marked full.

LOAD TRANSPARENT (LODT) 95BC.
This operator performs a Load operator (see section 7) if the word in the A register is a Data Descriptor or an Indirect Reference Word. If it is not either of these, bits 19:20 of the A register are used as the address to bring an operand to the A register. Copy bit action does not occur.

LINKED LIST LOOKUP (LLLU) 95BD.
This operator searches a linked list of words.

The operator starts with an operand in the top of the stack as the index pointer. The second word in the stack is a non-indexed Data Descriptor to the array containing the linked list. The third word in the stack is an operand that is the argument.

The base address of the linked list, the length of the list and the argument value are saved throughout the entire operator process.

The word addressed by the base address plus the index value is read and checked for a value of zero in the address (Link) portion of the word (zero denotes the end of the linked list). If the link is non-zero, bits 47:28 are compared to the argument value. If the argument of the linked-list word is less than the argument value, the actions of this paragraph are repeated using the link as the new index.

When the value of the argument field of the linked-list word is equal to or greater than the argument value the operation is complete. The index pointing to the word whose link points to the argument which satisfies the test is left in the A register and is marked full.

If the value of the link portion of the linked list word is equal to zero, the A register is set to minus one (-1), and marked full as the operation is completed.

If the index value in the linked list word is greater than the length value from the descriptor, an invalid index interrupt is set and the operation is terminated.

When the first word in the stack at the start of this operator is not an operand an invalid-operand interrupt is set and the operation is terminated.

If the Data Descriptor has been indexed, the invalid-operand interrupt is set and the operation is terminated.

MASKED SEARCH FOR EQUAL (SRCH) 95BE.

At the start of this operator the word in the A register must be a Data Descriptor. The operand in the B register is a 51 bit mask. The Data Descriptor in the A register and the mask in the B register are saved and the 51 bit argument word is placed into the B

register. If the descriptor is indexable (bit 45 equal to zero), the index bit (bit 45) is set and one is subtracted from the length field. If bit 45 is equal to one the data descriptor is already indexed, therefore, that index is the starting value.

The word addressed by the descriptor is placed in the A register and ANDed with the mask word. The result of this AND function is tested to determine if it is identical to the argument word.

If the comparison is not equal the index field of the descriptor is decreased by one and the operation is repeated. If the index field is equal to zero, the A register is set to a minus one value and marked full. The B register is marked empty.

If an equal comparison is made, the A register contains the index pointing at the last word compared and is marked full. The B register is marked empty.

UNPACK ABSOLUTE, DESTRUCTIVE (UABD) 95D1.
This operator unpacks a string of 4-bit digits into 6-bit characters or 8-bit bytes. At the start of the operator the word in the A register defines the length of the operand in the B register which is the string of digits to be unpacked.

The third word in the stack is a string descriptor addressing the destination of the string.

As the specified number of digits are transferred to the destination, zone fill is as follows:

a. If the destination size is 6 bits (BCL), the characters are transferred to the destination with the two zone bits set to zero.

b. If the destination size is 8 bits (EBCDIC) the bytes are transferred to the destination string with the four zone bits set to 1111.

c. If the destination size was 0, it is set to 8-bits and handled as in (b) above.

UNPACK ABSOLUTE, UPDATE (UABU) 95D9.
This operator performs an Unpack Absolute operation, and at the completion of the operation, the destination pointer is updated and left in the stack.

UNPACK SIGNED, DESTRUCTIVE (USND) 95D0.
This operator performs an Unpack operation, with an added function, if the External Sign flip flop is set then a zone of 10 is set in the last character for 6-bit or a zone of 1101 is set in the last byte for 8-bit.

If the destination size is 4-bit, the first digit position of the destination string is set to 1101 if the External Sign flip flop is set. If the External Sign flip flop is zero the first digit is set to 1100.

UNPACK SIGNED, UPDATE (USNU) 95D8.
This operator performs an Unpack Signed operation, and at the completion of the operator, updates the destination pointer.

TRANSFER WHILE TRUE, DESTRUCTIVE (TWTD) 95D3.
This operator transfers characters from the source string to the destination string for the number of characters specified by the length operand while the stated relationship is met. If the relationship is not met the transfer is terminated at that point. The relationship is determined by using the source character to index a table. If the bit indexed is a one the relationship is true.

The operator uses the top four words in the stack to set up registers.

The stack words are used as follows: The top word addresses the table; the second word is the length of the string to be transferred; the third word in the stack is an operand or a descriptor, addressing the source string or a single-precision operand which is the source string; the fourth word in the stack is a descriptor pointing at the destination string.

The table is indexed as follows to obtain the decision bit: The source character is expanded to eight bits, if necessary, by appending two or four leading zero bits. The three high-order bits of these eight select a word from the table, indexing the table pointer. The remaining five bits of the expanded source character select a bit from this word by their value.

TRANSFER WHILE TRUE, UPDATE (TWTU) 95DB.
This operator performs a Transfer While True operation and updates the source pointer, the destination pointer and repeat count.

If all the characters specified by the length field are transferred, the True/False flip flop is set to one (true); otherwise, it is set to zero (false).

TRANSFER WHILE FALSE, DESTRUCTIVE (TWFD) 95D2.
This operator performs a Transfer While operation testing for a zero bit in the table.

TRANSFER WHILE FALSE, UPDATE (TWFU) 95DA.

This operator performs a Transfer While False operation, updating the source pointer, the destination pointer, and the repeat count.

If all the characters specified by the length field are transferred, the True/False flip flop is set to one (true); otherwise, it is set to zero (false).

TRANSLATE (TRNS) 95D7.
This operator translates the number of characters specified as they are transferred from the source string to the destination string.

The translation uses a table containing the translated characters. The word in the top of the stack is a descriptor that addresses the translation table. The second operand in the stack specifies the length of the string. The third word in the stack is a descriptor addressing the source string (or an operand which is the source string) and the fourth word in the stack is a descriptor

addressing the destination string. The source and destination are updated at the end of the operation.

The translation occurs as follows: The specified string character is used as an index into the table to locate a character. The located character is transferred to the destination string.

The least significant 32 bits of each table word provide 4 eight bit characters. The table sizes are as follows:

a. 4-bit digits provide a 4 word table length.

b. 6-bit characters provide a 16 word table length.

c. 8-bit bytes provide a 64 word table length.

SCAN WHILE GREATER, DESTRUCTIVE (SGTD) 95F2.
This operator scans a string while the characters in the source string are greater than a delimiter character; or until the number of characters specified have been scanned.

At the completion of this operator if all the characters have been scanned the True/False flip flop is set to one. If the scan was stopped by the delimiter test before the end of the string the True/False flip flop is set to zero.

At the start of this operator the delimiter character is right justified in the top word of the stack. The length of the string to be scanned is the second word of the stack. The source pointer is the third word in the stack.

If the second word in the stack is a descriptor, it is the source pointer and the length of the character string is set to 1,048,575.

SCAN WHILE GREATER, UPDATE (SGTU) 95FA.

This operator performs a Scan While Greater operation, and updates the count and the source pointer. The updated source pointer lo-

cates the character that stopped the scan. The number of characters not scanned is placed in the A register, and the register marked full.

SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE (SGED) 95F1.
This operator performs a Scan While operation while the characters in the source string are equal to or greater than the delimiter character.

SCAN WHILE GREATER OR EQUAL, UPDATE (SEGU) 95F9.
This operator performs a Scan While Greater or Equal operation and updates the count and the source pointer.

SCAN WHILE EQUAL, DESTRUCTIVE (SEQD) 95F4.
This operator performs a Scan While operation while the characters in the source string are equal to the delimiter character.

SCAN WHILE EQUAL, UPDATE (SEQU) 95FC.
This operator performs a Scan While Equal operation and updates the count and the source pointer.

SCAN WHILE LESS OR EQUAL, DESTRUCTIVE (SLED) 95F3.
This operator performs a Scan While operation while the characters in the source string are equal to or less than the delimiter character.

SCAN WHILE LESS OR EQUAL, UPDATE (SLEU) 95FB.
This operator performs a Scan While Less or Equal operation and updates the count and source pointer.

SCAN WHILE LESS, DESTRUCTIVE (SLSD) 95F0.
This operator performs a Scan While operation while the characters in the source string are less than the delimiter character.

SCAN WHILE LESS, UPDATE (SLSU) 95F8.
This operator performs a Scan While Less operation, and updates the count and the source pointer.

SCAN WHILE NOT EQUAL, DESTRUCTIVE (SNED) 95F5.
This operator performs a Scan While operation while the characters in the source string are not equal to the delimiter character.

SCAN WHILE NOT EQUAL, UPDATE (SNEU) 95FD.
This operator performs a Scan While Not Equal operation, and updates the count and the source pointer.

SCAN WHILE TRUE, DESTRUCTIVE (SWTD) 95D5.
This operator uses each source character as an index into a table to locate a bit in the same fashion as the transfer while True operators. If the bit located is a one, the relationship is true and the scan continues.

The first word in the stack is a descriptor addressing the table. The second and third words in the stack are as they are for all Scan While operators.

SCAN WHILE TRUE, UPDATE (SWTU) 95DD.
This operator performs a Scan While True operation and updates the count and the source pointer. The number of characters not scanned is placed in the A register.

SCAN WHILE FALSE, DESTRUCTIVE (SWFD) 95D4.
This operator performs a Scan While False operation, except the relation is true if the bit found by indexing into the table is zero.

SCAN WHILE FALSE, UPDATE (SWFU) 95DC.
This operator performs a Scan While False operation, and updates the count and the source pointer.

SECTION 9
EDIT MODE OPERATION AND OPERATORS

GENERAL.

The purpose of the Edit Mode operators is to perform editing functions on strings of data. The editing functions are those which are normally involved in preparing information for output. They include such operators as Move, Insert, and Skip, in the form of micro-operators in either the program string or in a separate table. In the program string, they are single micro-operators and are entered by use of the Execute Single Micro or Single Pointer operators (see section 7). If the micro-operators are in a table, the table becomes the program string that is to be executed. This table is entered by means of the Table Enter Edit operators (see section 7), and is exited through the End Edit micro-operator as defined later in this section.

When using any of the Edit micro-operators the proper pointers must be in the stack. Each of the micro-operators assume that if a source pointer is used, a source pointer String Descriptor or the source string itself as an operand will be present in the stack. If a destination pointer is used a String Descriptor must be present in the stack.

If the source or destination data has the memory protect bit (bit 48) equal to one, the segmented-array interrupt is set and the current micro-operator is terminated.

EDIT MODE OPERATORS.

The Edit Mode operators are described in the following paragraphs of this section.

MOVE CHARACTERS (MCHR) D7.

This micro-operator transfers characters from the source string to the destination string.

If the operator was entered by the Table Enter Edit operator (see

section 7), the number of characters to be transferred is specified by the syllable following the operator syllable.

If the operator is entered by the Execute Single Micro operator (see section 7), the number of characters to be transferred is specified by the operand in the top of the stack.

MOVE NUMERIC UNCONDITIONAL (MVNU) D6.
This micro-operator transfers the four low-order bits of the characters of the source string to the destination string. If the destination string character size is 6 bits (BCL) the zone bits are set to 00. If the destination string character size is 8 bits (EBCDIC) the zone bits are set to 1111.

If the operator was entered by use of the Table Enter Edit operator (see section 7) the number of characters to be transferred is specified by the syllable following the operator syllable.

If the operator is entered by executing the Execute Single Micro operator (see section 7), the number of characters to be transferred is specified by the operand in the top of the stack.

MOVE WITH INSERT (MINS) D0.
This micro-operator performs a Move Numeric Unconditional or an insert operation under the control of the Float flip flop.

In Table Edit mode the second syllable is the repeat value and the third syllable is the character to be inserted under control of the Float flip flop.

In Execute Single Micro mode the repeat field value is the top word of the stack and the insert character is in the syllable following the micro-operator syllable.

If the Float flip flop is zero and the numeric portion of the source characters is zero, the insert character is moved to the destination string.

If the Float flip flop is zero, or if the Float flip flop is on,

the Float flip flop is set and the source character, with numeric zone, is moved to the destination.

The number of characters transferred from the source string to the destination string is defined by the repeat value.

MOVE WITH FLOAT (MFLT) D1.
In Table Edit mode the second syllable is the repeat value (the number of characters to transfer). The third, fourth, and fifth syllables are the three insert characters. In single-micro mode, the three insert characters are in the second, third, and fourth syllables.

If the Float flip flop is zero and the numeric portion of the character in the source string is zero, the first-insert character is transferred to the destination string.

If the Float flip flop is zero and the numeric portion of the character in the source string is not zero the Float flip flop is set. If the External Sign flip flop is a one, the second-insert character is transferred to the destination string. If the External Sign flip flop is zero the third-insert character is transferred to the destination string. Then the numeric version of the source character is transferred.

If the Float flip flop is one the numeric equivalent of the source character is transferred to the destination.

This operation continues for the number of characters defined by the repeat field value.

This operator can be entered by the Execute Single Micro operator, with the repeat field value in the top word of the stack.

SKIP FORWARD SOURCE CHARACTERS (SFSC) D2.
This micro-operator increments the source pointer registers.

If this micro-operator or any of the following Skip micro-operators

is entered by the execution of the Execute Single Micro operator the number of characters to be skipped is specified by the operand in the top of the stack. If entry is by the execution of the Table Enter Edit operators, the number of characters to be skipped is specified by the syllable following the micro-operator syllable.

SKIP REVERSE SOURCE CHARACTERS (SRSC) D3.
This micro-operator decrements the source pointer registers.

Also see Skip Forward Source Characters micro-operator, second paragraph.

SKIP FORWARD DESTINATION CHARACTERS (SFDC) DA.
This micro-operator increments the destination pointer registers.

SKIP REVERSE DESTINATION CHARACTERS (SRDC) DB.
This micro-operator decrements the destination pointer registers.

RESET FLOAT (RSTF) D4.
This micro-operator sets the Float flip flop to zero.

END FLOAT (ENDF) D5.
This micro-operator transfers the character in the second syllable of this operator to the destination string if the Float flip flop is zero and the External Sign flip flop is one.

If the Float flip flop is zero and the External Sign flip flop is zero then the character in the third syllable of this operator is transferred.

If the Float flip flop is equal to one, then it is reset and no characters are transferred.

INSERT UNCONDITIONAL (INSU) DC.
This micro-operator places an insert character into the destination string the number of times specified by the repeat value. When entered by a Table Enter Edit operator, the REPEAT is in the syllable following the micro-operator syllable, and the insert character is in the next syllable.

If this micro-operator is entered by an Execute Single Micro operator, the character to be inserted is in the second syllable and the repeat value is specified by the operand that is in the top of the stack.

INSERT CONDITIONAL (INSC) DD.
This micro-operator inserts a string consisting of one of two characters into the destination. The length of the string is given by the repeat value from the table or the stack.

If the Float flip flop is zero the first insert character is inserted into the destination string.

If the Float flip flop is one the second insert character is inserted into the destination string.

The insert characters follow the repeat value syllable in Table Enter Edit operation or the micro-operator syllable in Execute Single Micro operations.

INSERT DISPLAY SIGN (INSG) D9.
This micro-operator places in the destination string the character defined by the syllable following the micro-operator syllable if the External Sign flip flop is equal to one.

If the External Sign flip flop is equal to zero this operator places in the destination string the character defined by the third syllable of this operator.

INSERT OVERPUNCH (INOP) D8.
This micro-operator places a sign overpunch in the destination string character of either 10 for BCL or 1101 for EBCDIC if the External Sign flip flop is equal to one.

If the External Sign flip flop is equal to zero the operator leaves the destination string character unaltered.

END EDIT (ENDE) DE.
This operator terminates a string of Edit micro-operators in Table Enter Edit operation mode.

The micro program string in the table must end with the End Edit operator.

SECTION 10
INPUT/OUTPUT MULTIPLEXOR AND PERIPHERAL CONTROLS

GENERAL.

The internal processing speed of the B 6500 is complemented by equally powerful input/output (I/O) hardware to achieve a well-balanced computing system. Transfer of all data between memory and all peripheral devices is controlled by the I/O multiplexor, independent of the processor. One or two of these multiplexors may be attached to a B 6500, each one capable of processing up to ten I/O operations simultaneously, from any of 28 peripheral devices.

OPERATION.

A peripheral control bus extends from the multiplexor to the various peripheral devices. Attached along this bus are from one to 20 peripheral controls (figure 10-1). Information in one or two-byte groups can be sent along the bus to or from any peripheral control every 1.2 microseconds.

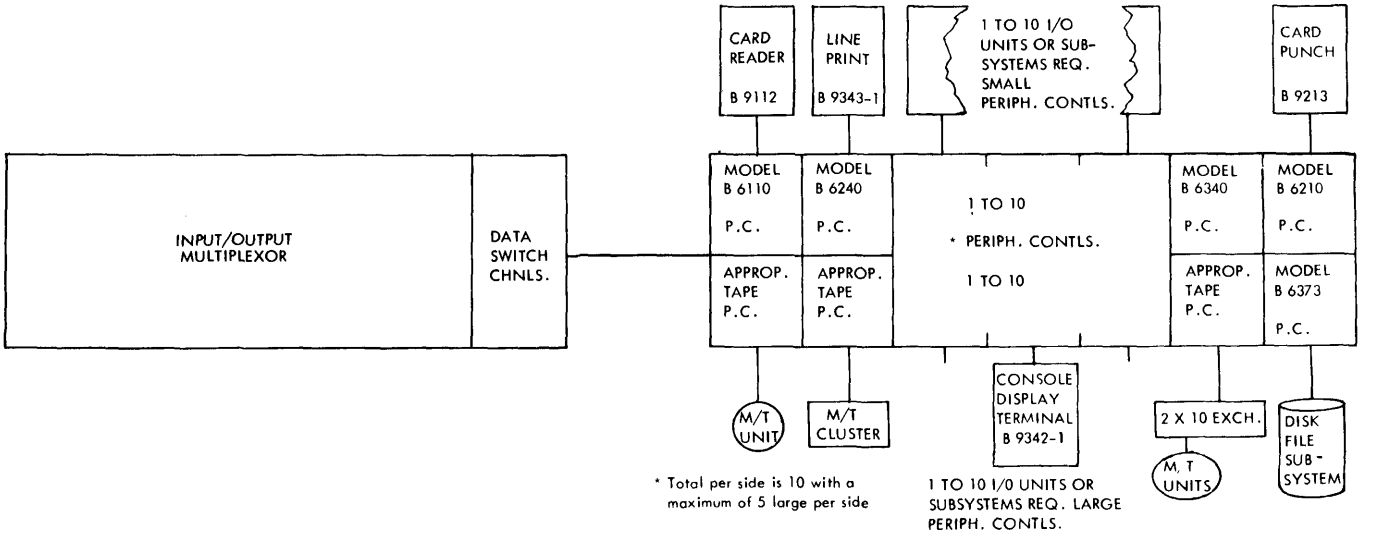


Figure 10-1. Input/Output Subsystem

Either processor can initiate an operation on either multiplexor, in a two processor/two multiplexor configuration, by executing a Scan In/Out instruction. This instruction transfers an address Word and a Data Word to the multiplexor. If the address Word specifies an Initiate I/O operation, then the data word is an Area Descriptor. The multiplexor fetches the I/O Control Word located at the Area Base Address (from the Area Descriptor) and initiates the peripheral operation. Upon completion, the I/O Finish Interrupt is set. The Result Descriptor is returned when processor executes a Read Result Descriptor command.

DESCRIPTOR FORMATS.

The formats of the Address Word, Area Descriptor, and I/O Control Word, respectively, are illustrated in figure 10-2.

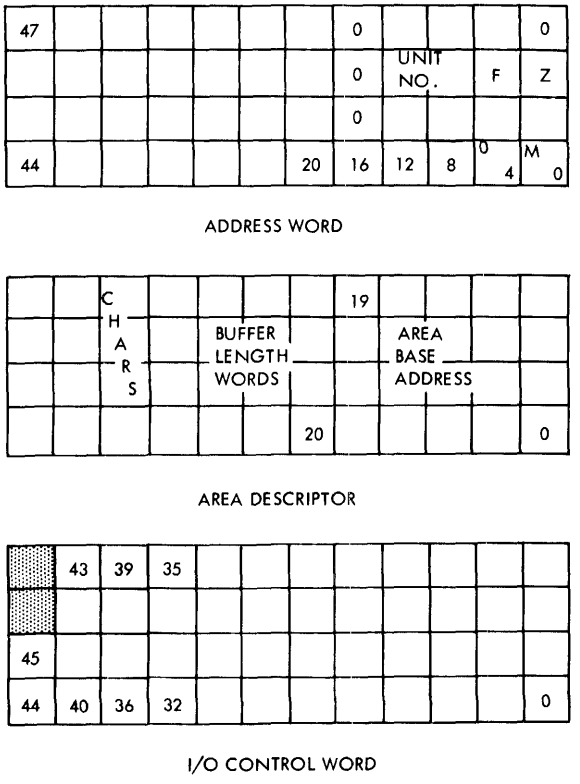


Figure 10-2. I/O Descriptor Formats

ADDRESS WORD.

When M of the address word equals 0, all active multiplexors respond to the descriptor. When M equals 1, the multiplexor specified by the Z field responds to the command. (The 2-bit Z field designates a specific multiplexor.) When Z equals 01 and M is 1, multiplexor A is selected. When Z equals 10 and M is 1, multiplexor B is selected. All other bit combinations in the Z field are not used. F-field codes are listed in table 10-1.

AREA DESCRIPTOR.

The area base address specifies the base address of the memory area. Buffer length indicates the size of the area. The first word of the area is the I/O Control Word.

I/O CONTROL WORD.

The I/O Control Word contains a standard control field and a unit control field. Bits 35 - 0, the unit control field, are unique for each peripheral control. Bits 45 - 36, the standard control field, are defined as follows:

Bit	Assignment	Bit = 0	Bit = 1
45	Attention	No	Yes
44	Read/Write	Write	Read
43	Memory Inhibit	No	Yes
42	Translate In Unit	No	Yes
41	Frame Length	6-bit	8-bit
40	Memory Protect	No	Yes
39	Backward	No	Yes
38	Test	No	Yes
37	1001 (Tag bit		
36	0101 field)		
	Store double-precision.		
	Store single-precision.		
	Store program tags.		
	Tag field transfer.		

Table 10-1

F Field Codes

Scan Oper.	F Bits 8765	Mnem.	Multiplexor Operation
OUT	0000	IOIL	Designated MPX to Initiate an I/O Operation. Bits 16 through 9 contain Unit Designate.
	0011	STOD	Set the Time Of Day Register.
	0100	SSIM	Set the Interrupt Mask Register.
	0000	IIOP	Interrogate I/O path for upcoming Initiate I/O operation.
IN	0001	IPST	Interrogate Peripheral Status of the designated Status Vector.
	0010		Read Result Descriptor.
	0011	RTOD	Read Time of Day Register.
	0100	SRIR SRIM	Read Interrupt Register or Interrupt Mask Register.
	0110	IPUT	Interrogate Peripheral Unit Type.
	1111	SRIL	Read Interrupt Literal.

RESULT DESCRIPTOR.

The format of the Result Descriptor is shown in figure 10-3.

Bits 47:20 indicate the final memory address at which the I/O operation terminated. Bits 16:17 the error field, is subdivided into a standard error field and a unit error field. The unit error field bit assignments, bits 15:9 are unique for each peripheral control. The standard error field bit assignments, bits 6:7 and 16 are as follows:

Bit	Assignment
16	Memory Protection Error
6	Memory Parity Error
5	Memory Address Error
4	Descriptor Error
3	Not Ready
2	Busy
1	Attention
0	Exception

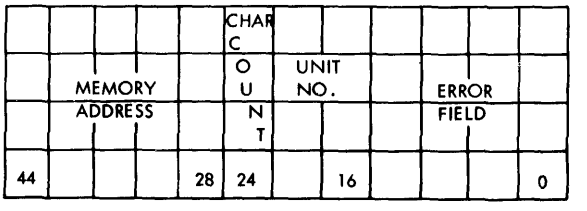


Figure 10-3. Result Descriptor Format

PERIPHERAL UNITS AND ASSOCIATED PERIPHERAL CONTROLS.

Up to 256 I/O devices may be attached to a 2 multiplexor system. These devices communicate with the multiplexor through a maximum of 20 peripheral controls. One peripheral control cabinet houses 10 controls, 5 large and 5 small. Table 10-2 lists the peripheral controls available excluding magnetic tape and disk file which are listed separately.

CONSOLE.

The Console Control Center (figure 10-4) includes the Supervisory Display and Keyboard, which allows the operator to communicate with the system. The B 6340 Single Line Control connects the Console Control Center and the multiplexor. Up to eight units can be serviced by one Single line control. Figures 10-5 and 10-6 depict the I/O Control word and the result descriptor for the Single Line Control.

Style	Peripheral Units	PC Style	PC Type	Peripheral Controls
B 9111	800 CPM Card Reader	B 6110	Small	Card Reader Control
B 9112	1400 CPM Card Reader	B 6110	Small	Card Reader Control
B 9120	500-1000 CPS Paper Tape Reader	B 6120	Small	Paper Tape Reader Control
B 9213	300 CPM Punch	B 6210	Small	Card Punch Control
B 9220	100 CPS Paper Tape Punch	B 6220	Small	Paper Tape Punch Control
B 9242-1	860 LPM Printer (120 Prt. Pos.)	B 6240	Small	Line Printer Control
B 9243-1	1100 LPM Printer(120 Prt.Pos.,44 Ch.)	B 6240	Small	Line Printer Control
B 9342-1 B 9342-2	Console Display Terminal Optional Printer/Keyboard	B 6340	Large	Console Display & Optional Printer

Table 10-2. Peripherals and Controls

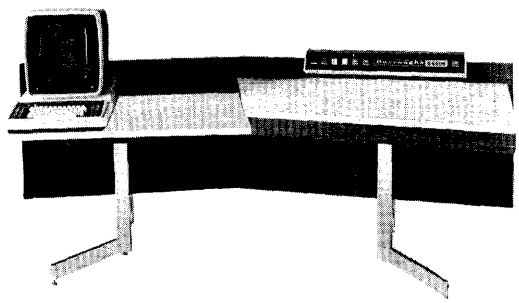


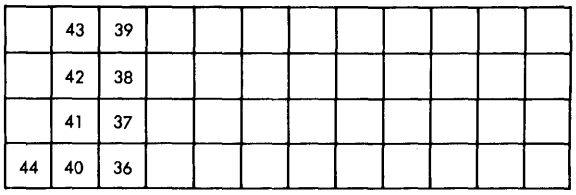
Figure 10-4. Console Control Center

47					27			15	11	7	
								14	10	6	
					25		17	13	9		
				28	24		16	12			0

6:7	Standard Error Field
7	Memory Access Error
7 & 9	Information Parity Error
10	Control Message
11	No ETX
12	Unit ID - B9342-1

15	Time Out
16	Memory Protect Error (Read Only)
24:8	Unit Designate
27:3	Char. Count
47:20	Memory Address

Figure 10-5. Single Line Control Result Descriptor



45 = ATTENTION

44 = 1 READ
= 0 WRITE

43 = 0

42 = 0

41 = 1 8 bit

40 = 0

39 = 0

38 = 0

37 = 0

36 = 0

} tag bit field

Figure 10-6. Single Line Control I/O Control Word

CARD READER.

The B 6110 Card Reader Control can be used with either the B 9111 (800 cpm) or B 9112 (1400 cpm) card readers (figure 10-7). The input hopper and the output stacker have a capacity of 2400 cards each. The card readers accept alpha, binary or EBCDIC card codes. Alpha card code is converted to BCL by the card reader, which is then converted into internal BCL or EBCDIC by translators in the multiplexor. EBCDIC card code is converted to internal EBCDIC by the card reader control (B 6110). When reading binary punched cards no translation is made.

The card readers can read 51, 60, or 80 column punched cards. Optional features include the ability to read 40 column Treasury checks and round holes in Postal Money Orders. Cards of varying thickness are acceptable; however, card thickness and length must be consistent during any one run. Figures 10-8 and 10-9 depict the I/O control word and the result descriptor for card reader operations.

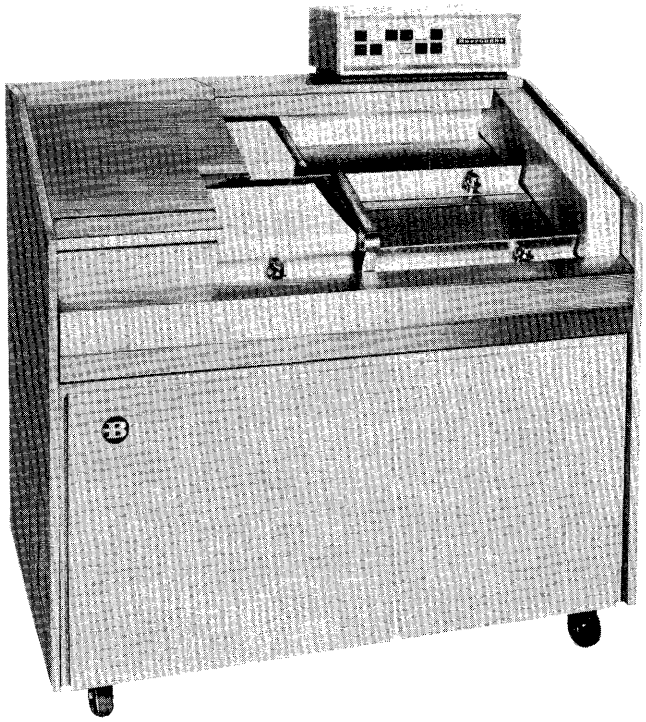


Figure 10-7. Card Reader

	42										
	41	37									
44	40	36									

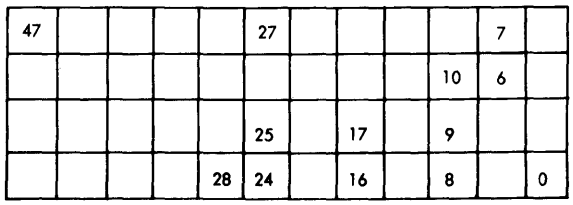
Figure 10-8. Card Read I/O Control Word

44 = 1
40 = 1 Memory protect
39 = 0
38 = 0

Alpha		EBCDIC
42 = 1		42 = 0
41 = 0	6 bit	41 = 1
41 = 1	8 bit	

Binary	
42 = 0	37 tag bit
41 = 0	36 field
37 = 0	

Figure 10-8. Card Read I/O Control Word (cont)



6:7	Standard Error Field
7	Memory Access Error
8	Read Check
7 & 9	Validity Error
10	Control card (alpha only)
16	Memory Protect Error
24:8	Unit Designate
27:3	Character Count
47:20	Memory Address

Figure 10-9. Card Read Result Descriptor

CARD PUNCH.

The B 6210 Card Punch Control is used with the B 9213 Card Punch (figure 10-10), which can punch either binary, alpha, or EBCDIC code at a rate of 300 cards per minute. Pre-punched cards may be used, but previously punched columns cannot be repunched. The card punch has a 1000 card capacity input hopper and three output stackers (primary, auxiliary and error) which have a capacity of 1200 cards each. Stacker selection is accomplished programmatically. Figures 10-11 and 10-12 depict the I/O control word and the result descriptor for the card punch operation.

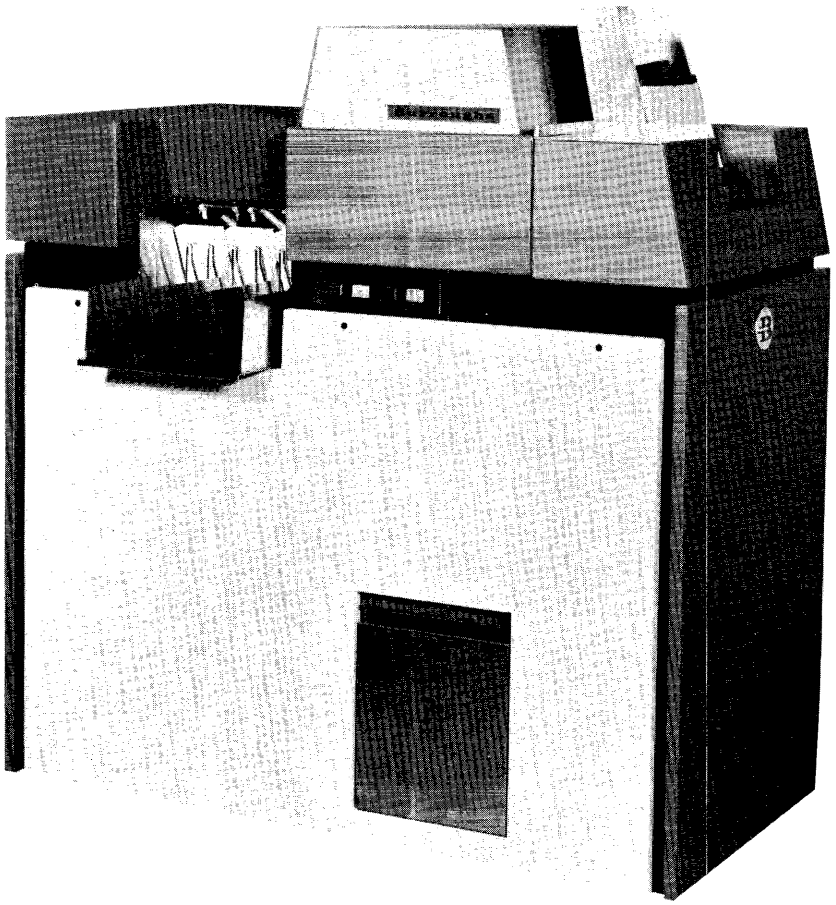
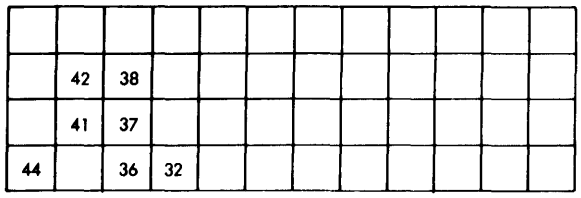


Figure 10-10. Card Punch



44 = 0
38 = 0
32 = 1 Auxiliary stacker

37 tag bit
36 = 0 field

Alpha
42 = 1
41 = 0 6 bit
41 = 1 8 bit

Binary
42 = 0
41 = 0
37 = 0

EBCDIC
42 = 0
41 = 1

Figure 10-11. Card Punch I/O Control Word

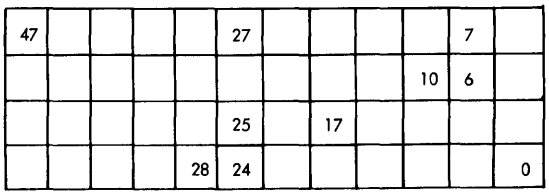


Figure 10-12. Card Punch Result Descriptor

6:7	Standard Error Field
7	Punch Check
7 & 10	Memory Access Error
24:8	Unit Designate
21:3	Character Count
41:20	Memory Address

Figure 10-12. Card Punch Descriptor (cont)

LINE PRINTERS.

Two line printers (figure 10-13) are available for use on the B 6500 system. The B 9242 prints 860 lines per minute (LPM) and the B 9243, 1100 LPM. Both printers are available with either 120 or 132 print positions. Both have vertical skipping and end-of-page formatting controlled by a punched paper tape. The B 6240 Line Printer Control connects the printer to the multiplexor. Translators in the multiplexor convert internal BCL or EBCDIC into BCL for transmission to the printer control. Figures 10-14 and 10-15 show the Printer I/O control word and result descriptor.

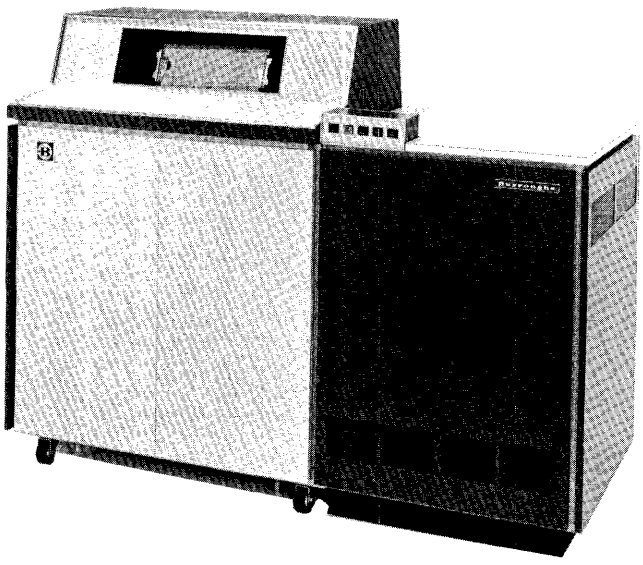
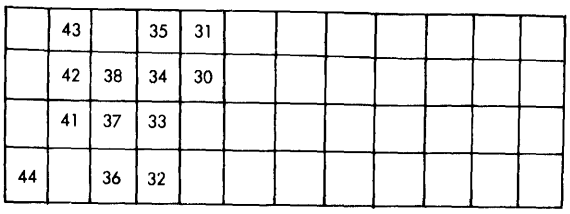


Figure 10-13. Line Printer



44 = 0

43 = 0 Print

43 = 1 Space - Inhibit Data Transfer

42 = 1 Translate to BCL

41 = 0 6 bit; 41 = 1 8 bit

38 = 0

37 } tag bit

36 = 0 } field

35:5 Skip to Channel 1 => 11

31 = 1 Double Space } only if 35:5

30 = 1 Single Space } equals zero

Figure 10-14. Line Printer I/O Control Word

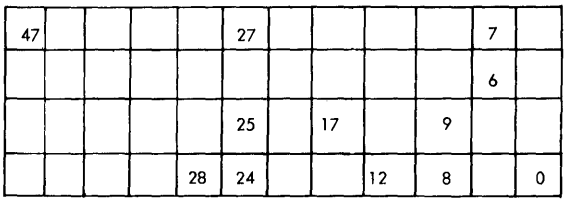


Figure 10-15. Line Printer Result Descriptor

6:7	Standard Error Field
8:2	Bit Transfer Error
9:3	Print Check
24:8	Unit Designate
27:3	Character Count
47:20	Memory Address

Figure 10-15. Line Printer Result Descriptor (cont)

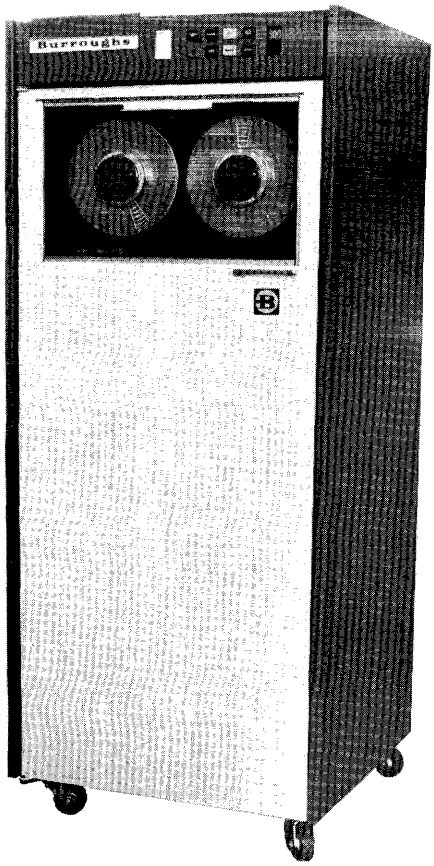
MAGNETIC TAPE SUBSYSTEM.

A magnetic tape subsystem can include from one to four tape controls servicing from one to sixteen magnetic tape units. Within a single tape system all tape units must be used at the same speed and all controls must be of the same type.

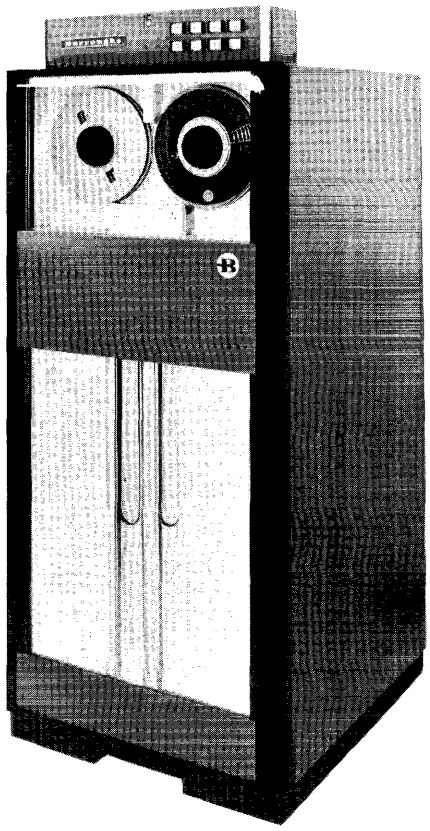
A magnetic tape exchange is required when more than one control or more than six magnetic tape units are used.

The number of magnetic tape units on a system is limited only by the number of exchanges and peripheral controls employed. The user may choose either 7-channel tape or 9-channel tape which may be intermixed, provided this is not attempted on the same subsystem. The user may also select any of four packing densities up to 1600 bits per inch and transfer rates from 9,000 to 240,000 bytes per second.

A choice of physical construction may be made between two free standing devices which house one tape unit per cabinet (figure 10-16), or the cluster unit (figure 10-17) which houses up to four tape units per cabinet. The magnetic tape units are capable of reading and spacing in either a forward or reverse direction. Table 10-3 lists the available magnetic tape subsystems. Figure 10-18 shows possible configurations of these subsystems.



72KB MTU



144/192/240KB MTU

Figure 10-16. Free Standing Magnetic Tape Unit

Figure 10-17. Cluster Tape Unit

Table 10-3

Available Magnetic Tape Subsystems

Style	Description of Magnetic Tape Subsystems	Appropriate Peripheral Control		Exchanges		
		Style	Quantity	Style	Type	Function
B 9381-2,3,4 36KB Cluster	9 channel 800/200* BPI, 45 IPS, 2 to 8 Tape Stations	B 6381-1	1	None	None	1 Tape Operation
		B 6381-1	1 or 2	B 6481	2x8	2 Tape Operation
B 9382-2,3,4 72KB Cluster	9 channel 1600 BPI, 45 IPS, 2 to 8 Tape Stations	B 6381-2	1	None	None	1 Tape Operation
		B 6381-2	1 or 2	B 6481	2x8	2 Tape Operation
B 9380-2,3,4 36KC Cluster	7 channel 200/556/800 BPI, 45 IPS, 2 to 8 Tape Stations	B 6381-3	1	None	None	1 Tape Operation
		B 6381-3	1 or 2	B 6480	2x8	2 Tape Operation
B 9391 72 KC Free Stand'g Tape Unit	Either 1 to 6, 1 to 10, or 1 to 16 Tape Units, 7 Channel, 200, 556, and 800 BPI, 90 IPS	B 6391-3	1	None	None	1 Tape Operation
		B 6391-3	1 or 2	B 6490	2x10	2 Tape Operation
		B 6391-3	1 to 4	B 6492	4x16	4 Tape Operation
B 9392 72KB Free Stand'g Tape Unit	Either 1 to 6, 1 to 10 or 1 to 16 Tape Units, 9 channel, 800/200* BPI, 90 IPS	B 6393-1	1	None	None	1 Tape Operation
		B 6393-1	1 or 2	B 6490	2x10	2 Tape Operation
		B 6393-1	1 to 4	B 6492	4x16	4 Tape Operation
B 9394-1 24, 66, or 96KC Free Stand'g Tape Unit	Either 1 to 6, or 1 to 10 Tape Units, 7 channel, 200, 556 or 800 BPI, 120 IPS	B 6391-4	1	None	None	1 Tape Operation
		B 6391-4	1 or 2	B 6490	2x10	2 Tape Operation
B 9394-2 96 KB Free Stand'g Tape Unit	Either 1 to 6, 1 to 10 or 1 to 16 Tape Units, 9 channel, 800/200* BPI, 120 IPS	B 6393-3	1	None	None	1 Tape Operation
		B 6393-3	1 or 2	B 6490	2x10	2 Tape Operation
		B 6393-3	1 to 4	B 6492	4x16	4 Tape Operation
B 9393-1 144KB Free Stand'g Tape Unit	1 to 8 Tape Units 9 channel Phase Encoded 1600 BPI, 90 IPS	B 6393-2	1	B 6493-1	1x8	1 Tape Operation
		B 6393-2	1 or 2	B 6493-2	2x8	2 Tape Operation
B 9393-2 192KB Free Stand'g Tape Unit	1 to 8 Tape Units 9 channel Phase Encoded 1600 BPI, 120 IPS	B 6393-2	1	B 6493-1	1x8	1 Tape Operation
		B 6393-2	1 or 2	B 6493-2	2x8	2 Tape Operation
B 9393-3 240KB Free Stand'g Tape Unit	1 to 8 Tape Units 9 channel Phase Encoded 1600 BPI, 150 IPS	B 6393-2	1	B 6493-1	1x8	1 Tape Operation
		B 6393-2	1 or 2	B 6493-2	2x8	2 Tape Operation

* A Model B 6681 (for Clusters) or B 6691 or B 6692 (for Free Standing Units) Optional Adapter must be attached to each peripheral control on a 9 channel 800 BPI tape system to provide 200 BPI capability.

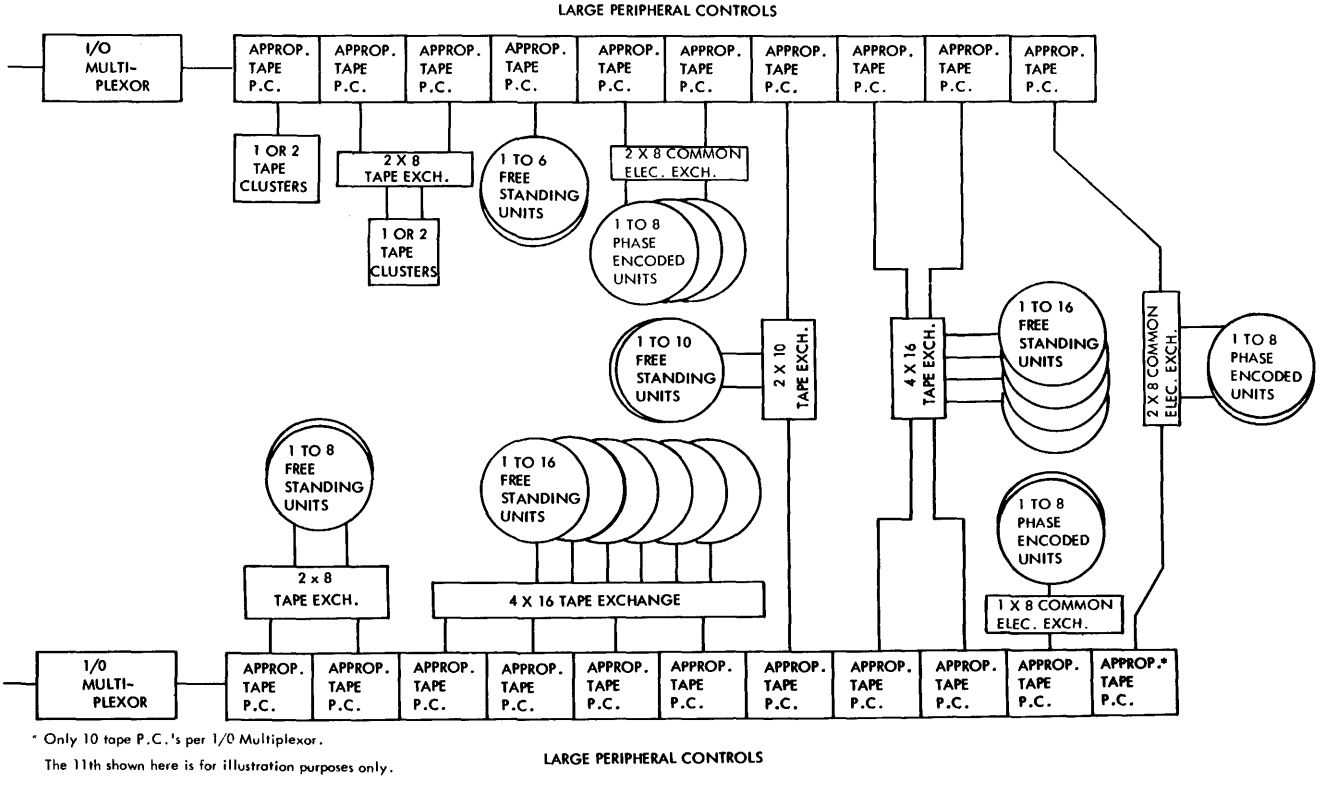


Figure 10-18. Magnetic Tape Configuration

Figure 10-19 shows the B 6500 magnetic tape I/O control word. This word is used to depict the various types of magnetic tape operations possible that are listed in table 10-4. When an operation is finished the result descriptor returned is shown in figure 10-20.

OPERATION	STANDARD CONTROL FIELD								
	44	43	42	41	40	39	38	37	36
READ BCL	1	0	1	O	O	O	0	O	O
READ BINARY	1	0	0	0	O	O	0	O	O
READ EBCDIC	1	0	0	1	O	O	0	O	O
SPACE	1	1				O	0		
WRITE BCL	0	0	1	O		0	0	O	
WRITE BINARY	0	0	0	0		0	0	O	
WRITE EBCDIC	0	0	0	1		0	0	O	
ERASE	0	1	O	O		0	0	O	
WRITE TM	0	0				0	0		
REWIND	0	1				1	0		
TEST							1		

BIT 35 = 0 AND 34 = 1

Table 10-4. Magnetic Tape Operations

	43	39	35	31	27	23					
	42	38	34	30	26						
	41	37	33	29							
44	40	36	32	28			16				

44 = 1 Tape Read
= 0 Tape Write
43 = 1 Memory Inhibit
42 = 1 Translate
41 = 0 6 bit; = 1 8 bit
40 = 1 Memory Protect
39 = 0 Forward; = 1 Backward
38 = 0
37:2 Tag bit field
35:2 Equal to zero

Figure 10-19. I/O Control Word Magnetic Tape

33:4 Format

100Ø	800 BPI
101Ø	555 BPI (7 track only)
110Ø	200 BPL
1111	1600 BPI (9 track only)
000Ø	Unit selected density

30 = 0 even parity
= 1 odd parity

9 Track Read only

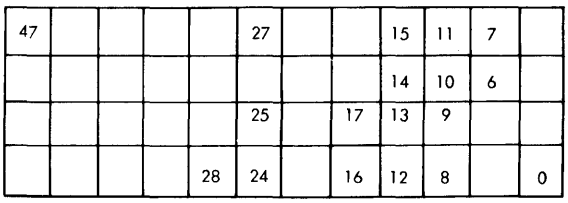
29 = 1 CRC Correction

28:2 If 29 = 1 then track to be corrected.

Space Only

23:8 decimal value of number of records to be spaced, 100 max.

Figure 10-19. I/O Control Word Magnetic Tape (cont)



6:7 Standard Error Field

7 Memory Access Error

8 End of tape or beginning of tape

9 Read - end of file; write - lock out

10 Incomplete Record

11 Oversized Record

Figure 10-20. Magnetic Tape Result Descriptor

11:2	Density (test only) 00 - 800 BPI 01 - 200 BPI 10 - 555 BPI 11 - BPI
7 & 10 & 11	Mag tape parity error
12	CRC correction possible, bits 15:3 defines track
13	Non-present option
15	6 ft. blank tape
16	Memory Protect Error (read only)
24:8	Unit Designate
27:3	Character Counter
47:20	Memory Address

Figure 10-20. Magnetic Tape Result Descriptor (cont)

DISK FILE SUBSYSTEM.

The Disk File Subsystem is an extremely high-speed, modular, random information storage system. A basic system consists of one electronics unit and from one to five storage units, see figure 10-21. If more than one basic subsystem is used then an exchange may be installed to connect the two subsystems to a disk file control. Figure 10-22 shows various disk file configurations allowed on a B 6500 system. The exchanges involved are located within the auxiliary cabinets that are attached to the peripheral control cabinets. Each of the disk file controls are the large size controls, therefore, they must be located only in positions zero through four in the PCC.

The various types of disk file subsystems and their capacities and speeds are indicated in table 10-5. Figures 10-23 and 10-24 indicate the disk file I/O control word and the disk file result descriptor.

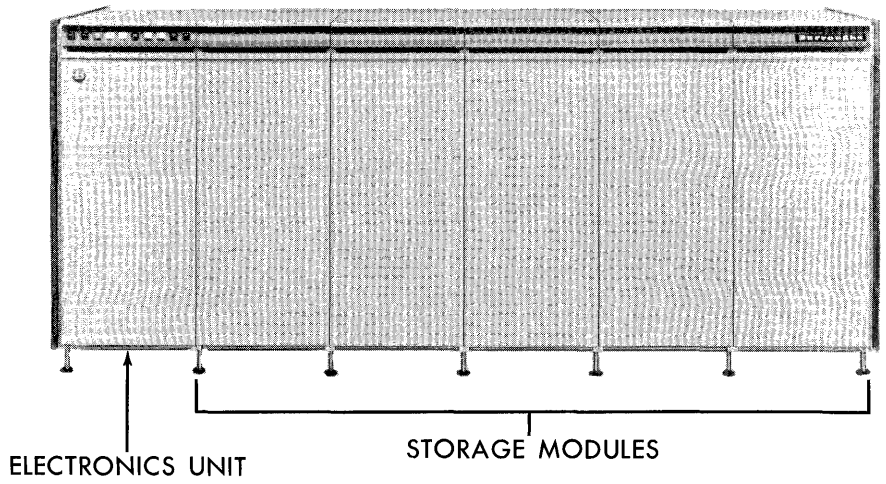


Figure 10-21. Basic Disk File Subsystem

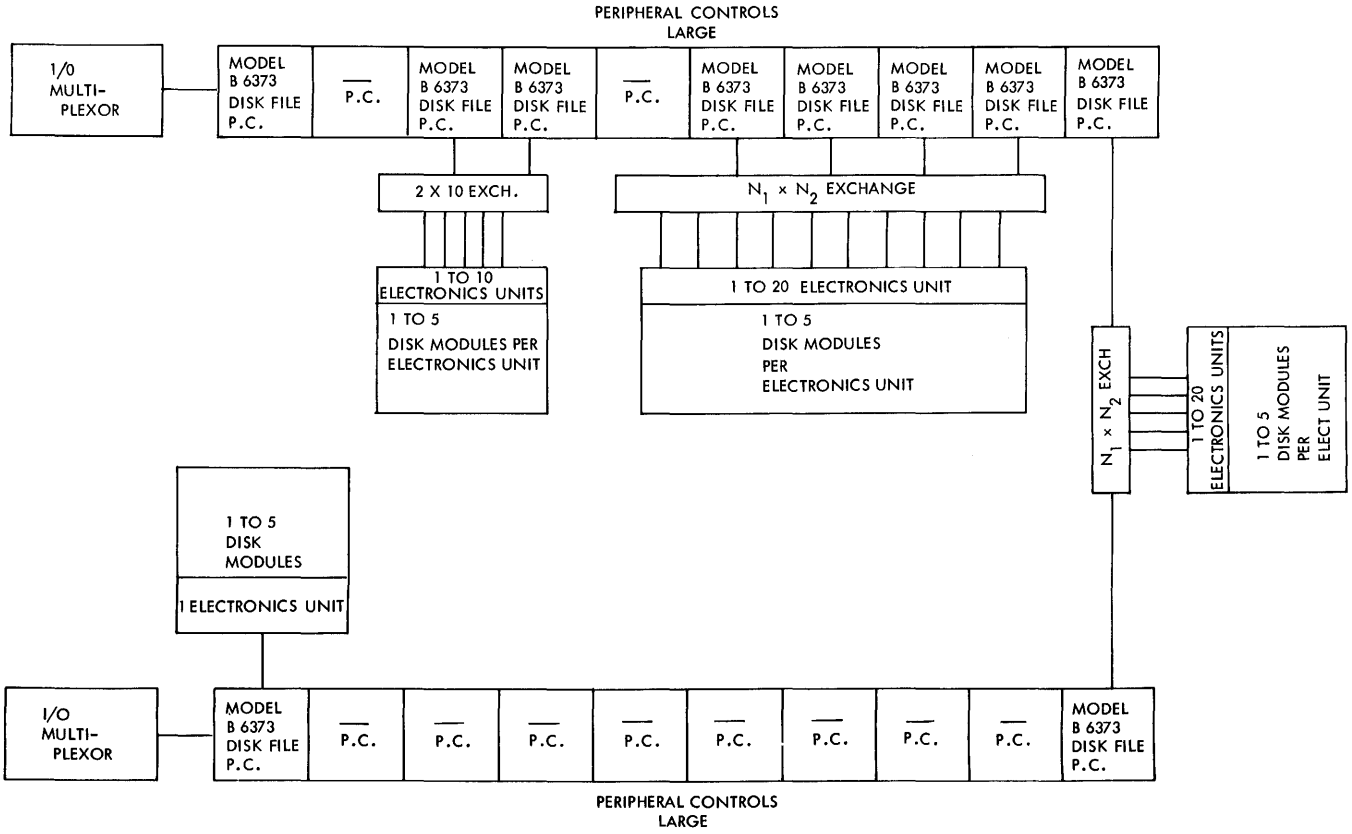


Figure 10-22. Disk File Configurations

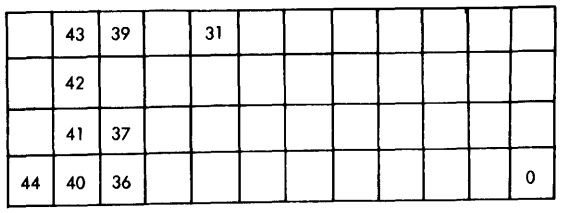
Table 10-5

Disk File Subsystem Types

Disk		Electronic Unit	Exchange			Peripheral Control	
Style	Description	Style	Style	Type	Function	Style	Quantity
B 9372-11	10.87 mill. bytes or 14.5 mill. char. - 20 ms	B 9371-7	B 6471	N_1xN_2	4x20 with Appropriate Adapters & Extension*	B 6373 B 7373	1 to 4
B 9375-10	Data Memory Bank 133 mill. char. or 100 mill. bytes – 23 ms	** B 9371-8					
B 9376-10	Addt'l 26.6 mill. char. or 20 mill. byte Increments – 23 ms	** B 9371-8					
B 9375-12	Data Memory Bank 133 mill. char. or 100 mill. bytes - 40 ms	** B 9371-9					
B 9376-12	Addt'l 30.5 mill. char. or 22.8 mill byte Increments – 40 ms	** B 9371-9					
B 9375-13	Data Memory Bank 133 mill. char. or 100 mill. bytes-60 ms	** B 9371-10					
B 9376-13	Addt'l 26.6 mill. char. or 20 mill. byte Increments – 60 ms	** B 9371-10					

* A B 6471-5 or B 7471-5 Control Adapter (N_1) is required for each control in the subsystem and a B 6471-6 or B 7471-6 EU Adapter (N_2) is required for each electronic unit in the subsystem. The B 6471-7 or B 7471-7 Exchange Extension is required to go above 10 EU adapters on the subsystem.

** Data Memory Banks and Increments include an electronic unit for every 5 disk modules; however additional optional EU's may be ordered for more paths to the disk modules, using stated EU style numbers.



44 = 1 } Disk File READ
43 = 0 }

44 = 1 } READ CHECK
43 = 1 }

44 = 0 } WRITE
43 = 0 }

42 = 0
41 = 1
40 = 1 Memory Protect
39 = 1 Maintenance Segment
37 } Tag Bit
36 } Field
31:24 Disk File ADDRESS (decimal)

Figure 10-23. Disk File I/O Control Word

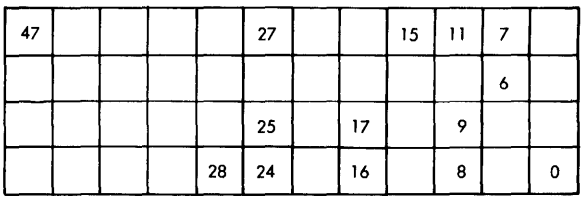


Figure 10-24. Disk File Result Descriptor

6:7	Standard Error Field
7	Memory Access Error
8	Unit busy
9	Write lock out
7 & 9	Disk Read Error
11	Went not ready
15	Time out
16	Memory protect (READ only)
24:8	Unit Designate
27:3	Character counter
47:20	Memory ADDRESS

Figure 10-24. Disk File Result Descriptor (cont)

PAPER TAPE.

The B 9120 Paper Tape Reader, figure 10-25, is capable of reading punched paper tape at a rate of 1000 characters per second and metalized mylar tape or fanfold tape at a rate of 500 characters per second. Baudot and BCL to EBCDIC code translation is automatic. All other codes are read directly into memory and may be translated programmatically. The reader can accommodate 5-, 6-, 7-, or 8 channel tape as selected by the operator. Tape widths of 11/16, 7/8, or 1 inch are interchangeable.

The paper tape punch, see figure 10-26, is capable of punching a standard paper tape format in either BCL or Baudot code. The punch accommodates 5-, 6-, 7-, or 8 channel tape at a maximum rate of 100 characters per second, punching ten characters to the inch. Standard tape widths of 11/16, 7/8, and 1 inch may be used in either the oiled paper tape, dry paper tape, metalized mylar tape, or laminated mylar tape.

Each paper tape I/O control, reader or punch, can accommodate only one paper tape unit each. The controls are the small size controls which can be set into a PCC cabinet as either a right hand or a left hand control.

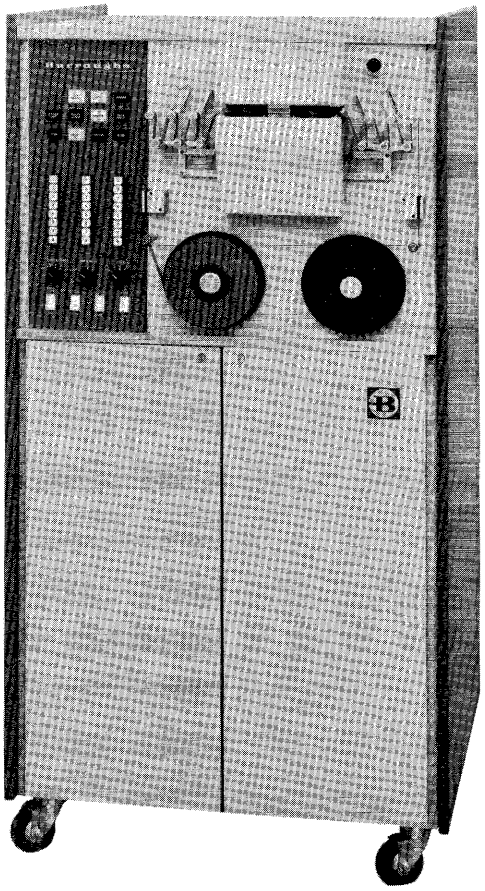


Figure 10-25. B 9120 Paper Tape Reader

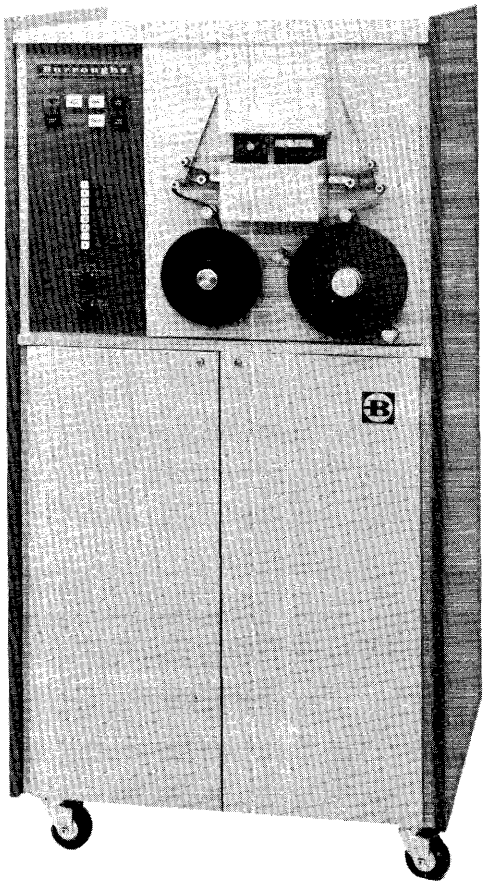
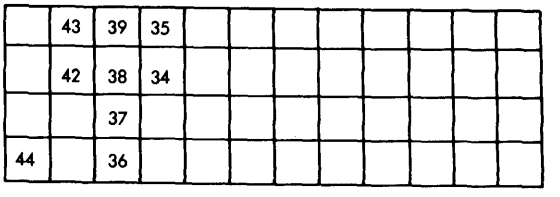


Figure 10-26. B 9220 Paper Tape Punch

Figure 10-27 indicates the paper tape control word and the various paper tape operations possible on the B 6500. Figure 10-28 indicates the paper tape result descriptor.



44 = 1 Tape read

= 0 Tape punch

43 = 1 Inhibit data transfer

42 = 1 Translate

39 = 0 Forward; = 1 Backward

38 = 1 Test

37:2 Tag field bits

35 & 36 Formats:

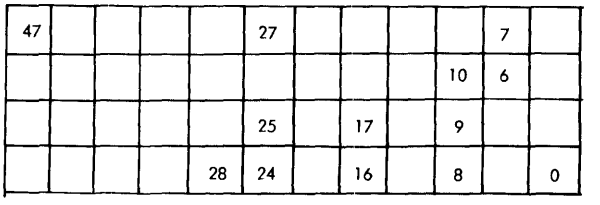
10 - 8 bit no parity

00 - 7 bit info plus 1 parity

01 - 6 bit info plus 1 parity

	44	43	42	41	40	39	38	37	36	35	34
READ BCL	1	0	1	O	O	0	0	O	O	0	1
READ BINARY	1	0	0	O	O	0	0	O	O	0	0
WRITE BCL	0	0	1	O	O	0	0	O	0	0	1
WRITE BINARY	0	0	0	O	O	0	0	O	0	O	O
PUNCH LEADER	0	1		O	O	0	0	O	0		
FWD SPACE	1	1		O		0	0	0			
BKWD SPACE	1	1		O		1	0	0			
REWIND	0	1				1	0				

Figure 10-27. Paper Tape I/O Control Word and Operations



6:7	Standard Error Field
7	Memory Access Error
8	Read - EOT or BOT
	Punch - Low Tape
7 & 9	Read - tape parity error
10	Incomplete record
16	Memory protect error

Figure 10-28. Paper Tape Result Descriptor

SECTION 11
B 6500 DATA COMMUNICATIONS SYSTEM

GENERAL.

The B 6500 Data Communications System is comprised of one or more of each of the following units:

a. Data Communications Processor (D.C.P.).
Each B 6500 Peripheral Control Multiplexor accommodates up to 4 D.C.P.'s through the word interfaces. The word interfaces provide access to the B 6500 main memory.

b. Adapter Cluster.
One Adapter Cluster services up to 16 Line Adapters which may have dissimilar characteristics. A maximum of 16 Adapter Clusters may be connected to one Data Communications Processor. It is also possible to connect an Adapter Cluster between two Data Communications Processors. This allows the Adapter Cluster to be serviced from either D.C.P.

c. Line Adapter.
Each communication line requires at least one Line Adapter. With some types of terminals two Line Adapters may be required. Up to 16 Line Adapters are accommodated by one Adapter Cluster.

The B 6500 Data Communications System can service a maximum of 2048 communications lines. A typical system configuration is shown in figure 11-1.

DATA COMMUNICATIONS PROCESSOR (D.C.P.).

The Data Communications Processor (D.C.P.) is a special purpose processor. It handles the transmitting and receiving of messages over the many data communications lines. A part of that task is answering calls, terminating calls, observing the formal line disciplines, polling operations and the formatting of messages.

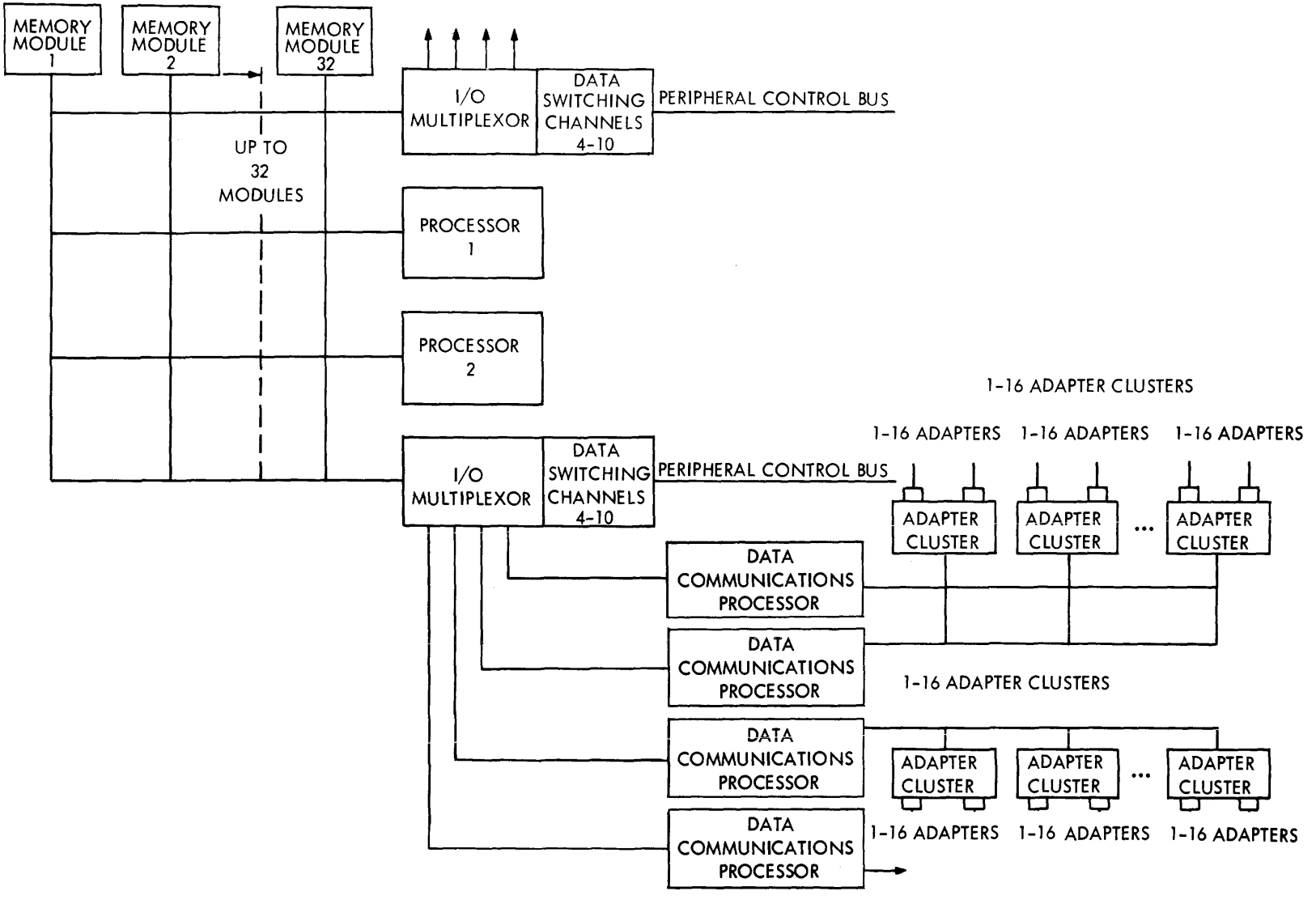


Figure 11-1. B 6500 System Configuration Including Data Communications

The Data Communications Processor is a stored program computer obtaining its program instructions either from B 6500 main memory or from an optional local memory. Through the use of the local memory the throughput of the D.C.P. is significantly increased due to the reduction in instruction fetch time.

If the optional local memory is not present, the Data Communications Processor shares the B 6500 system main memory with the other units of the B 6500. The memory allocation for the D.C.P. is controlled by the B 6500 Master Control Program. Data exchanges occur when the B 6500 processor initiates a D.C.P. operation and when the D.C.P. finishes an operation, i.e. I/O complete signal from the D.C.P.

The internal form of the Data Communications Processor is shown in figure 11-2. The Data Communications Processor is an elementary micro-programed processor. Two-address and three-address instructions, operating on 8-bit bytes, are used by the Data Communications Processor. The byte organization fits into a basic half-word (three byte) structure permitting efficient half-word transfers within the Data Communications Processor. The functions of the D.C.P. are accomplished with a small array of intercommunicating registers, a simple arithmetic-logical unit and an eight word scratchpad memory.

For complete information on all Data Communications Processor registers and memories, refer to the Data Communications Processor Reference Manual.

ADAPTER CLUSTER.

The Adapter Cluster is the interface between the Data Communications Processor and the data-communication Line Adapters. Each Adapter Cluster services up to 16 Line Adapters. Data transmission rates of from 45.5 B.P.S. to 4800 B.P.S. are handled by the Adapter Cluster simultaneously.

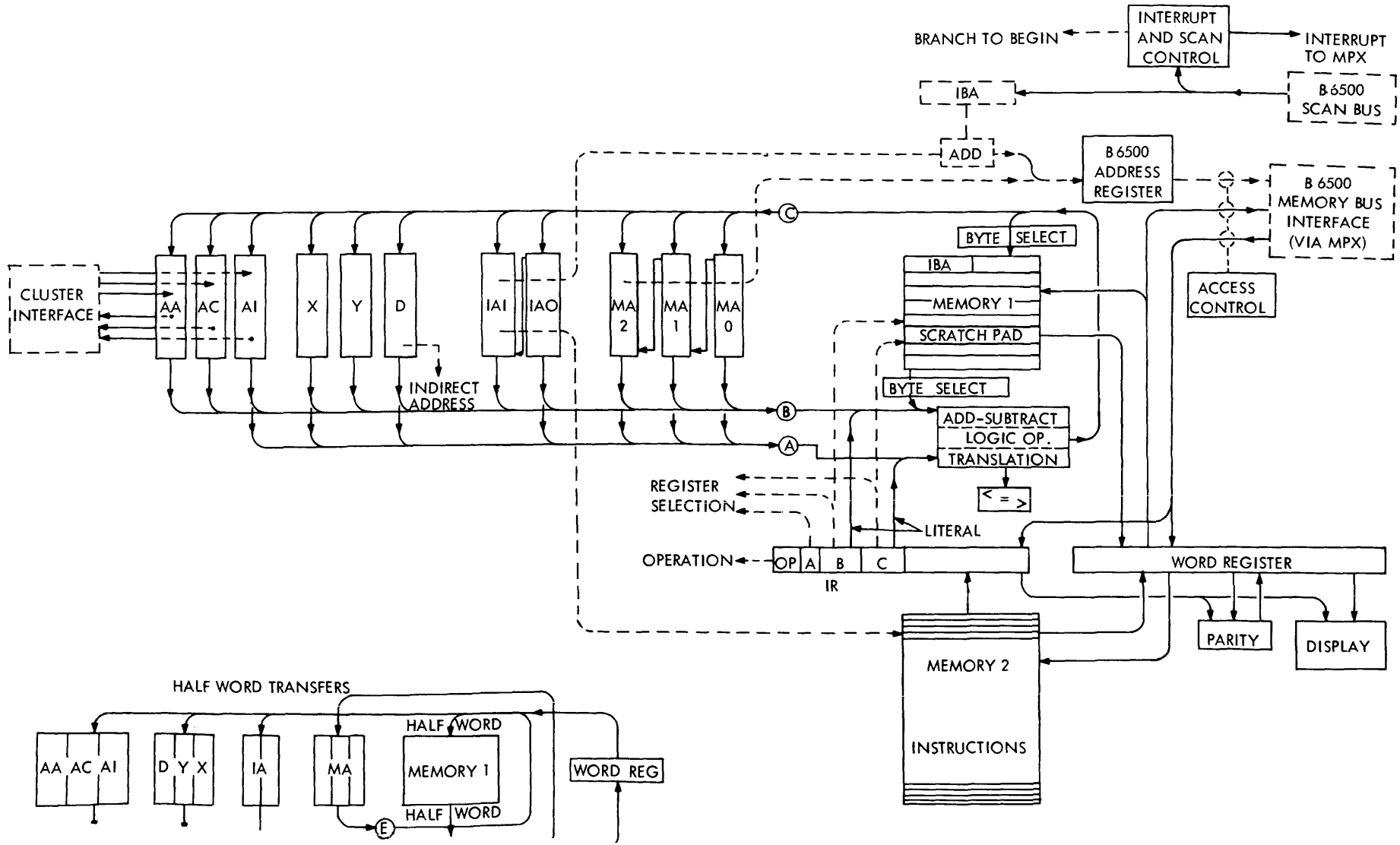


Figure 11-2. DCP Block Diagram

Figure 11-3 shows a block diagram of the Adapter Cluster. The Adapter Cluster basic functions are:

a. Line termination: Scanning, clocking and temporary storage.

b. Character assembly and disassembly.

c. Synchronization attainment and maintenance.

d. Timer operation to maintain line discipline.

e. Some character recognition logic. (Mainly synchronization characters for the various line disciplines).

f. Provide the ability to exchange information with one or two Data Communications Processors.

The Adapter Cluster functions in a manner that makes it appear transparent to most characters and message formats. However as stated in item (e) above it does recognize the SYN characters in order to attain and retain synchronization when operating in the synchronous mode.

LINE ADAPTER.

The Line Adapter types that are provided allow the Data Communications Processor to interface with data sets, Voice Response Systems and the direct connection to remote devices. Each Line Adapter terminates one line. The Line Adapter handles the exchange of bits or characters between the Adapter Cluster and the data communication line. The buffer of each type of Line Adapter contains either one bit or one character, depending on the type. Table 11-1 shows a table of terminal compatibility.

For more detailed information on all phases of the Data Communications Processor refer to the Data Communications Processor Reference Manual.

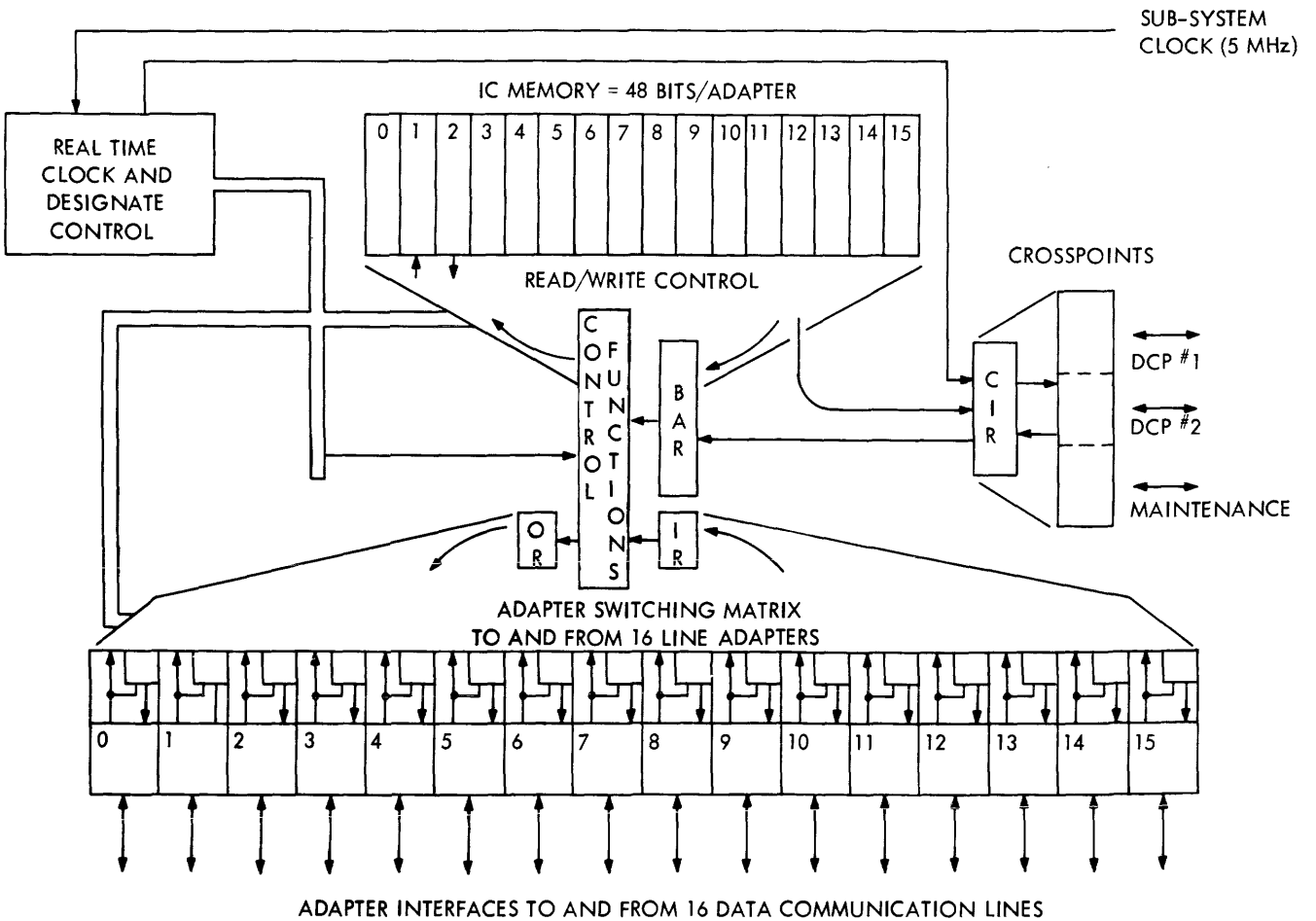


Figure 11-3. Adapter Cluster

Table 11-1

Data Communications Terminal Compatibility

	Leased	Switched	Dir. Conn.	Asynch.	Synch.	Modem Type	Speed Range
TWX Service		X		X		811B	Up to 150 BPS
W. E. Model 33	X	X	X	X		103	Up to 110 BPS
W. E. Model 35 (also 8A1 Sel Calling)	X	X	X	X		103	Up to 110 BPS
W. E. Model 37	X	X	X	X		103	Up to 165 BPS
B 9351 Series Display	X	X	X	X	X	103, 202 or 201	Up to 2400 BPS
B 9352 Series Display	X	X	X	X	X	103, 202 or 201	Up to 2400 BPS
Model 28/83B3 (or equiv. Western Union Service)	X		X	X			Up to 110 BPS
TC 500 Terminal	X	X	X	X		202	Up to 1200 BPS

Table 11-1 (cont)

Data Communications Terminal Compatibility

	Leased	Switched	Dir. Conn.	Asynch.	Synch.	Modem Type	Speed Range
B 300/B 340/B 500	X	X	X		X	201	Up to 2400 BPS
B 2500/B 3500	X	X	X		X	201	Up to 2400 BPS
B 5500	X	X	X		X	201	Up to 2400 BPS
Honeywell 120	X				X	201	Up to 2400 BPS
IBM 1030	X		X	X		202	Up to 14.8 CPS
Automatic Calling Unit		X				801	

APPENDIX A
OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
ADD	ADD	80
BIT RESET	BRST	9E
BIT SET	BSET	96
BRANCH FALSE	BRFL	A0
BRANCH TRUE	BRTR	A1
BRANCH UNCONDITIONAL	BRUN	A2
CHANGE SIGN BIT	CHSN	8E
COMPARE CHARACTERS EQUAL DESTRUCTIVE	CEQD	F4
COMPARE CHARACTERS EQUAL, UPDATE	CEQU	FC
COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED	F1
COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU	F9
COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD	F2
COMPARE CHARACTERS GREATER, UPDATE	CGTU	FA
COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED	F3
COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU	FB
COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD	F0

APPENDIX A (cont)

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
COMPARE CHARACTERS LESS, UPDATE	CLSU	F8
COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED	F5
COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU	FD
CONDITIONAL HALT (all modes)	HALT	DF
COUNT BINARY ONES	CBON	95 BB
DELETE TOP OF STACK	DLET	B5
DISABLE EXTERNAL INTERRUPT	DEXI	95 47
DIVIDE	DIV	83
DUPLICATE TOP OF STACK	DUPL	B7
DYNAMIC BIT RESET	DBRS	9F
DYNAMIC BIT SET	DBST	97
DYNAMIC BRANCH FALSE	DBFL	A8
DYNAMIC BRANCH TRUE	DBTR	A9
DYNAMIC BRANCH UNCONDITIONAL	DBUN	AA
DYNAMIC FIELD INSERT	DINS	9D
DYNAMIC FIELD ISOLATE	DISO	9B
DYNAMIC FIELD TRANSFER	DFTR	99
DYNAMIC SCALE LEFT	DSLF	C1

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
DYNAMIC SCALE RIGHT FINAL	DSRF	C7
DYNAMIC SCALE RIGHT ROUND	DSRR	C9
DYNAMIC SCALE RIGHT SAVE	DSRS	C5
DYNAMIC SCALE RIGHT TRUNCATE	DSRT	C3
ENABLE EXTERNAL INTERRUPTS	EEXI	95 46
END EDIT (edit mode)	ENDE	DE
END FLOAT (edit mode)	ENDF	D5
ENTER	ENTR	AA
EQUAL	EQUL	8C
ESCAPE TO 16-BIT INSTRUCTION	VARI	95
EVALUATE DESCRIPTOR	EVAL	AC
EXCHANGE	EXCH	B6
EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU	DD
EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD	D2
EXECUTE SINGLE MICRO, UPDATE	EXSU	DA
EXIT	EXIT	A3
EXTENDED MULTIPLY	MULX	8F
FIELD INSERT	INSR	9C
FIELD ISOLATE	ISOL	9A

APPENDIX A (cont)

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
FIELD TRANSFER	FLTR	98
GREATER THAN	GRTR	8A
GREATER THAN OR EQUAL	GREQ	89
IDLE UNTIL INTERRUPT	IDLE	95 44
INDEX	INDX	A6
INDEX AND LOAD NAME	NXLN	A5
INDEX AND LOAD VALUE	NXLV	AD
INPUT CONVERT, DESTRUCTIVE	ICVD	CA
INPUT CONVERT UPDATE	ICVU	CB
INSERT CONDITIONAL (edit mode)	INSC	DD
INSERT DISPLAY SIGN (edit mode)	INSG	D9
INSERT MARK STACK	IMKS	CF
INSERT OVERPUNCH (edit mode)	INOP	D8
INSERT UNCONDITIONAL (edit mode)	INSU	DC
INTEGER DIVIDE	IDIV	94
INTEGERIZE, ROUNDED	NTGR	97
INTEGERIZE, TRUNCATED	NTIA	96
INTEGERIZE, ROUNDED DOUBLE-PRECISION	NTGD	95 97

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
INTERRUPT OTHER PROCESSORS	HEYU	95 4F
INVALID OPERATOR (all modes)	NVLD	FF
JOIN TWO SINGLES TO DOUBLE	JOIN	95 42
LEADING ONE TEST	LOG2	95 8B
LINKED LIST LOOKUP	LLLU	95 BD
LESS THAN	LESS	88
LESS THAN OR EQUAL	LSEQ	8B
LIT CALL ONE	ONE	B1
LIT CALL ZERO	ZERO	B0
LIT CALL 8 BITS	LT8	B2
LIT CALL 16 BITS	LT16	B3
LIT CALL 48 BITS	LT48	BE
LOAD	LOAD	BD
LOAD TRANSPARENT	LODT	95 BC
LOGICAL AND	LAND	90
LOGICAL EQUAL	SAME	94
LOGICAL EQUIVALENCE	LEQV	93
LOGICAL NEGATE	LNOT	92
LOGICAL OR	LOR	91

APPENDIX A (cont)

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
MAKE PROGRAM CONTROL WORD	MPCW	BF
MARK STACK	MKST	AE
MASKED SEARCH FOR EQUAL	SRCH	95 BE
MOVE CHARACTERS (edit mode)	MCHR	D7
MOVE NUMERIC UNCONDITIONAL (edit mode)	MVNU	D6
MOVE TO STACK	MVST	95 AF
MOVE WITH FLOAT (edit mode)	MFLT	D1
MOVE WITH INSERT (edit mode)	MINS	D0
MULTIPLY	MULT	82
NAME CALL	NAMC	40 => 7F
NO OPERATION (all modes)	NOOP	FE
NOT EQUAL	NEQL	8D
OCCURS INDEX	OCRX	95 85
OVERWRITE DESTRUCTIVE	OVRD	BA
OVERWRITE NON-DESTRUCTIVE	OVRN	BB
PACK DESTRUCTIVE	PACD	D1
PACK UPDATE	PACU	D9
PUSH DOWN STACK REGISTERS	PUSH	B4

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
READ AND CLEAR OVERFLOW FLIP-FLOP	ROFF	D7
READ PROCESSOR IDENTIFICATION	WHOI	95 4E
READ PROCESSOR REGISTER	RPRR	95 B8
READ TAG FIELD	RTAG	95 B5
READ TRUE/FALSE FLIP-FLOP	RTFF	DE
READ WITH LOCK	RDLK	95 BA
REMAINDER DIVIDE	RDIV	85
RESET FLOAT (edit mode)	RSTF	D4
RETURN	RETN	A7
ROTATE STACK DOWN	RSDN	95 B7
ROTATE STACK UP	RSUP	95 B6
SCALE LEFT	SCLF	C0
SCALE RIGHT FINAL	SCRF	C6
SCALE RIGHT ROUND	SCRR	C8
SCALE RIGHT SAVE	SCRS	C4
SCALE RIGHT TRUNCATE	SCRT	C2
SCAN IN	SCNI	95 4A
SCAN OUT	SCNO	95 4B
SCAN WHILE EQUAL, DESTRUCTIVE	SEQD	95 4F

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
SCAN WHILE EQUAL, UPDATE	SEQU	95 FC
SCAN WHILE FALSE, DESTRUCTIVE	SWFD	95 D4
SCAN WHILE FALSE, UPDATE	SWFU	95 DC
SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED	95 F1
SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU	95 F9
SCAN WHILE GREATER, DESTRUCTIVE	SGTD	95 F2
SCAN WHILE GREATER, UPDATE	SGTU	95 FA
SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED	95 F3
SCAN WHILE LESS OR EQUAL, UPDATE	SLEU	95 FB
SCAN WHILE LESS, DESTRUCTIVE	SLSD	95 F0
SCAN WHILE LESS, UPDATE	SLSU	95 F8
SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED	95 F5
SCAN WHILE NOT EQUAL, UPDATE	SNEU	95 FD
SCAN WHILE TRUE, DESTRUCTIVE	SWTD	95 D5
SCAN WHILE TRUE, UPDATE	SWTU	95 DD
SET EXTERNAL SIGN	SXSN	D6
SET INTERVAL TIMER	SINT	95 45

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
SET PROCESSOR REGISTER	SPRR	95 B9
SET TAG FIELD	STAG	95 B4
SET TO DOUBLE-PRECISION	XTND	CE
SET TO SINGLE-PRECISION, ROUNDED	SNGL	CD
SET TO SINGLE-PRECISION, TRUNCATED	SNGT	CC
SKIP FORWARD DESTINATION CHARACTERS (edit mode)	SFDC	DA
SKIP FORWARD SOURCE CHARACTERS (edit mode)	SFSC	D2
SKIP REVERSE DESTINATION CHARACTERS (edit mode)	SRDC	DB
SKIP REVERSE SOURCE CHARACTERS (edit mode)	SRSC	D3
SPLIT DOUBLE TO TWO SINGLES	SPLT	95 43
STEP AND BRANCH	STBR	A4
STORE DESTRUCTIVE	STOD	B8
STORE NON-DESTRUCTIVE	STON	B9
STRING ISOLATE	SISO	D5
STUFF ENVIRONMENT	STFF	AF

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
SUBTRACT	SUBT	81
TABLE ENTER EDIT, DESTRUCTIVE	TEED	D0
TABLE ENTER EDIT, UPDATE	TEEU	D8
TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND	E6
TRANSFER UNCONDITIONAL, UPDATE	TUNU	EE
TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD	E4
TRANSFER WHILE EQUAL, UPDATE	TEQU	EC
TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED	E1
TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU	E9
TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD	E2
TRANSFER WHILE GREATER, UPDATE	TGTU	EA
TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED	E3
TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD	95 D2
TRANSFER WHILE FALSE, UPDATE	TWFU	95 DA
TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD	95 D3
TRANSFER WHILE TRUE, UPDATE	TWTU	95 DB
TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU	EB

OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
TRANSFER WHILE LESS, DESTRUCTIVE	TLSD	E0
TRANSFER WHILE LESS, UPDATE	TLSU	E8
TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED	E5
TRANSFER WHILE NOT EQUAL, UPDATE	TNEU	ED
TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD	D4
TRANSFER WORDS OVERWRITE UPDATE	TWOU	DC
TRANSFER WORDS, DESTRUCTIVE	TWSD	D3
TRANSFER WORDS, UPDATE	TWSU	DB
TRANSLATE	TRNS	95 D7
UNPACK ABSOLUTE, DESTRUCTIVE	UABD	95 D1
UNPACK ABSOLUTE, UPDATE	UABU	95 D9
UNPACK SIGNED, DESTRUCTIVE	USND	95 D0
UNPACKED SIGNED, UPDATE	USNU	95 D8
VALUE CALL	VALC	00 =〉 3F

APPENDIX B
OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
DF	CONDITIONAL HALT (UNIVERSAL OPERATOR)	HALT
FE	NO OPERATION (UNIVERSAL OPERATOR)	NOOP
FF	INVALID OPERATOR (UNIVERSAL OPERATOR)	NVLD
00 => 3F	VALUE CALL	VALC
40 => 7F	NAME CALL	NAMC
80	ADD	ADD
81	SUBTRACT	SUBT
82	MULTIPLY	MULT
83	DIVIDE	DIVD
84	INTEGER DIVIDE	IDIV
85	REMAINDER DIVIDE	RDIV
86	INTEGERIZE, TRUNCATED	NTIA
87	INTEGERIZE, ROUNDED	NTGR
88	LESS THAN	LESS
89	GREATER THAN OR EQUAL	GREQ
8A	GREATER THAN	GRTR
8B	LESS THAN OR EQUAL	LSEQ
8C	EQUAL	EQUL

OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
8D	NOT EQUAL	NEQL
8E	CHANGE SIGN BIT	CHSN
8F	EXTENDED MULTIPLY	MULX
90	LOGICAL AND	LAND
91	LOGICAL OR	LOR
92	LOGICAL NEGATE	LNOT
93	LOGICAL EQUIVALENCE	LEQV
94	LOGICAL EQUAL	SAME
95	ESCAPE TO 16-BIT INSTRUCTION	VARI
96	BIT SET	BSET
97	DYNAMIC BIT SET	DBST
98	FIELD TRANSFER	FLTR
99	DYNAMIC FIELD TRANSFER	DFTR
9A	FIELD ISOLATE	ISOL
9B	DYNAMIC FIELD ISOLATE	DISO
9C	FIELD INSERT	INSR
9D	DYNAMIC FIELD INSERT	DINS
9E	BIT RESET	BRST

OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
9F	DYNAMIC BIT RESET	DBRS
A0	BRANCH FALSE	BRFL
A1	BRANCH TRUE	BRTR
A2	BRANCH UNCONDITIONAL	BRUN
A3	EXIT	EXIT
A4	STEP AND BRANCH	STBR
A5	INDEX AND LOAD NAME	NXLN
A6	INDEX	INDX
A7	RETURN	RETN
A8	DYNAMIC BRANCH FALSE	DBFL
A9	DYNAMIC BRANCH TRUE	DBTR
AA	DYNAMIC BRANCH UNCONDITIONAL	DBUN
AB	ENTER	ENTR
AC	EVALUATE DESCRIPTOR	EVAL
AD	INDEX AND LOAD VALUE	NXLV
AE	MARK STACK	MKST
AF	STUFF ENVIRONMENT	STFF
B0	LIT CALL ZERO	ZERO

OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
B1	LIT CALL ONE	ONE
B2	LIT CALL 8 BITS	LT8
B3	LIT CALL 16 BITS	LT16
B4	PUSH DOWN STACK REGISTERS	PUSH
B5	DELETE TOP OF STACK	DLET
B6	EXCHANGE	EXCH
B7	DUPLICATE TOP OF STACK	DUPL
B8	STORE DESTRUCTIVE	STOD
B9	STORE NON-DESTRUCTIVE	STON
BA	OVERWRITE DESTRUCTIVE	OVRD
BB	OVERWRITE NON-DESTRUCTIVE	OVRN
BD	LOAD	LOAD
BE	LIT CALL 48 BITS	LT48
BF	MAKE PROGRAM CONTROL WORD	MPCW
C0	SCALE LEFT	SCLF
C1	DYNAMIC SCALE LEFT	DSLF
C2	SCALE RIGHT TRUNCATE	SCRT
C3	DYNAMIC SCALE RIGHT TRUNCATE	DSRT

OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
C4	SCALE RIGHT SAVE	SCRS
C5	DYNAMIC SCALE RIGHT SAVE	DSRS
C6	SCALE RIGHT FINAL	SCRF
C7	DYNAMIC SCALE RIGHT FINAL	DSRF
C8	SCALE RIGHT ROUND	SCRR
C9	DYNAMIC SCALE RIGHT ROUND	DSRR
CA	INPUT CONVERT, DESTRUCTIVE	ICVD
CB	INPUT CONVERT, UPDATE	ICVU
CC	SET TO SINGLE-PRECISION, TRUNCATED	SNGT
CD	SET TO SINGLE-PRECISION, ROUNDED	SNGL
CE	SET TO DOUBLE-PRECISION	XTND
CF	INSERT MARK STACK	IMKS
DO	TABLE ENTER EDIT, DESTRUCTIVE	TEED
D1	PACK DESTRUCTIVE	PACD
D2	EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD
D3	TRANSFER WORDS, DESTRUCTIVE	TWSD
D4	TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD
D5	STRING ISOLATE	SISO

OPERATORS, NUMERICAL LIST PRIMARY MODE

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
D6	SET EXTERNAL SIGN	SXSN
D7	READ AND CLEAR OVERFLOW FLIP-FLOP	ROFF
D8	TABLE ENTER EDIT, UPDATE	TEEU
D9	PACK UPDATE	PACU
DA	EXECUTE SINGLE MICRO, UPDATE	EXSU
DB	TRANSFER WORDS, UPDATE	TWSU
DC	TRANSFER WORDS OVERWRITE UPDATE	TWOU
DD	EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU
DE	READ TRUE/FALSE FLIP-FLOP	RTFF
E0	TRANSFER WHILE LESS, DESTRUCTIVE	TLSD
E1	TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED
E2	TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD
E3	TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED
E4	TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD
E5	TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED
E6	TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND
E8	TRANSFER WHILE LESS, UPDATE	TLSU

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
E9	TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU
EA	TRANSFER WHILE GREATER, UPDATE	TGTU
EB	TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU
EC	TRANSFER WHILE EQUAL, UPDATE	TEQU
ED	TRANSFER WHILE NOT EQUAL, UPDATE	TNEU
EE	TRANSFER UNCONDITIONAL, UPDATE	TUNU
F0	COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD
F1	COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED
F2	COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD
F3	COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED
F4	COMPARE CHARACTERS EQUAL, DESTRUCTIVE	CEQD
F5	COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED
F8	COMPARE CHARACTERS LESS, UPDATE	CLSU
F9	COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU
FA	COMPARE CHARACTERS GREATER, UPDATE	CGTU

PRIMARY MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
FB	COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU
FC	COMPARE CHARACTERS EQUAL, UPDATE	CEQU
FD	COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU

VARIANT MODE.

95 42	SET TWO SINGLES TO DOUBLE	JOIN
95 43	SET DOUBLE TO TWO SINGLES	SPLT
95 44	IDLE UNTIL INTERRUPT	IDLE
95 45	SET INTERVAL TIMER	SINT
95 46	ENABLE EXTERNAL INTERRUPTS	EEXI
95 47	DISABLE EXTERNAL INTERRUPTS	DEXI
95 4A	SCAN IN	SCNI
95 4B	SCAN OUT	SCNO
95 4E	READ PROCESSOR IDENTIFICATION	WHOI
95 4F	INTERRUPT OTHER PROCESSORS	HEYU
95 85	OCCURS INDEX	OCRX
95 87	INTEGERIZE, ROUNDED, DOUBLE-PRECISION	NTGD
95 8B	LEADING ONE TEST	LOG2

VARIANT MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
95 AF	MOVE TO STACK	MVST
95 B4	SET TAG FIELD	STAG
95 B5	READ TAG FIELD	RTAG
95 B6	ROTATE STACK UP	RSUP
95 B7	ROTATE STACK DOWN	RSDN
95 B8	READ PROCESSOR REGISTER	RPRR
95 B9	SET PROCESSOR REGISTER	SPRR
95 BA	READ WITH LOCK	RDLK
95 BB	COUNT BINARY ONES	CBON
95 BC	LOAD TRANSPARENT	LODT
95 BD	LINKED LIST LOOKUP	LLLU
95 BE	MASKED SEARCH FOR EQUAL	SRCH
95 DO	UNPACK SIGNED, DESTRUCTIVE	USND
95 D1	UNPACK ABSOLUTE, DESTRUCTIVE	UABD
95 D2	TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD
95 D3	TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD
95 D4	SCAN WHILE FALSE, DESTRUCTIVE	SWFD
95 D5	SCAN WHILE TRUE, DESTRUCTIVE	SWTD

OPERATORS, NUMERICAL LIST PRIMARY MODE

VARIANT MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
95 D7	TRANSLATE	TRNS
95 D8	UNPACK SIGNED, UPDATE	USNU
95 D9	UNPACK ABSOLUTE, UPDATE	UABU
95 DA	TRANSFER WHILE FALSE, UPDATE	TWFU
95 DB	TRANSFER WHILE TRUE, UPDATE	TWTU
95 DC	SCAN WHILE FALSE, UPDATE	SWFU
95 DD	SCAN WHILE TRUE, UPDATE	SWTU
95 F0	SCAN WHILE LESS, DESTRUCTIVE	SLSD
95 F1	SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED
95 F2	SCAN WHILE GREATER, DESTRUCTIVE	SGTD
95 F3	SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED
95 F4	SCAN WHILE EQUAL, DESTRUCTIVE	SEQD
95 F5	SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED
95 F8	SCAN WHILE LESS, UPDATE	SLSU
95 F9	SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU
95 FA	SCAN WHILE GREATER, UPDATE	SGTU
95 FB	SCAN WHILE LESS OR EQUAL, UPDATE	SLEU

OPERATORS, NUMERICAL LIST PRIMARY MODE

VARIANT MODE.

HEXADECIMAL CODE	NAME	MNEMONIC
95 FC	SCAN WHILE EQUAL, UPDATE	SEQU
95 FD	SCAN WHILE NOT EQUAL, UPDATE	SNEU

EDIT MODE.

D0	MOVE WITH INSERT	MINS
D1	MOVE WITH FLOAT	MFLT
D2	SKIP FORWARD SOURCE CHARACTERS	SFSC
D3	SKIP REVERSE SOURCE CHARACTERS	SRSC
D4	RESET FLOAT	RSTF
D5	END FLOAT	ENDF
D6	MOVE NUMERIC UNCONDITIONAL	MVNU
D7	MOVE CHARACTERS	MCHR
D8	INSERT OVERPUNCH	INOP
D9	INSERT DISPLAY SIGN	INSG

VARIANT MODE.

DA	SKIP FORWARD DESTINATION CHARACTERS	SFDC
DB	SKIP REVERSE DESTINATION CHARACTERS	SRDC
DC	INSERT UNCONDITIONAL	INSU

OPERATORS, NUMERICAL LIST PRIMARY MODE

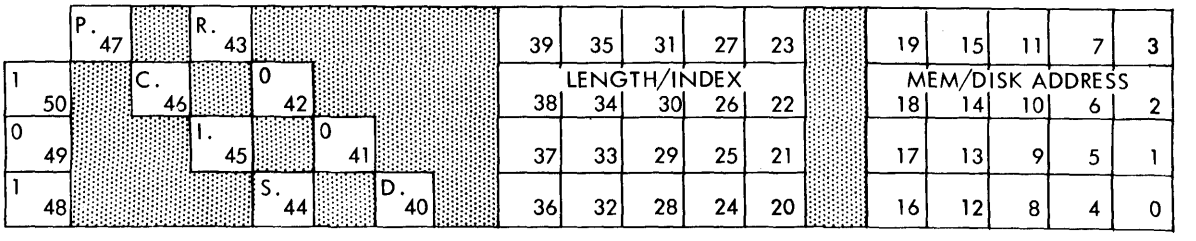
VARIANT MODE.

HEXADECIMAL

CODE	NAME	MNEMONIC
DD	INSERT CONDITIONAL	INSC
DE	END EDIT	ENDE

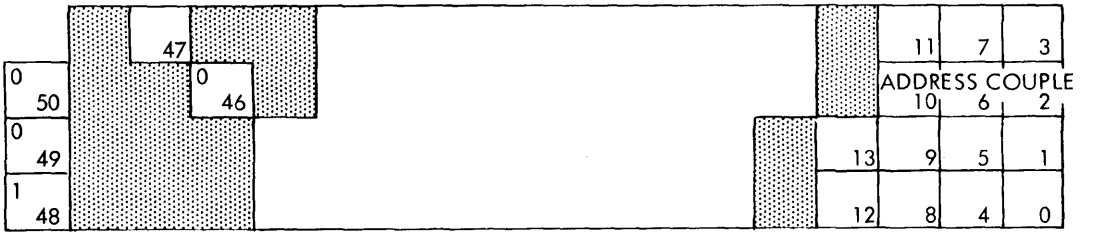
APPENDIX C

CONTROL WORD FORMATS



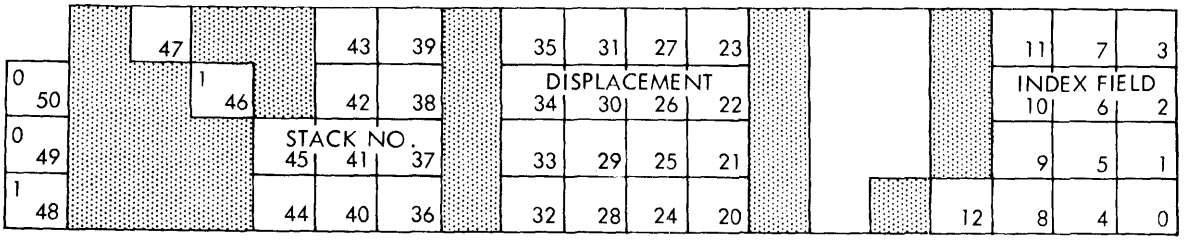
DATA DESCRIPTOR

P = PRESENCE BIT	C = COPY BIT	I = INDEX BIT	S = SEGMENTED BIT
1 = PRESENT IN MAIN MEMORY	1 = A COPY	1 = INDEXED	1 = AREA SEGMENTED
0 = NOT PRESENT IN MAIN MEMORY	0 = ORIGINAL	0 = NON INDEXED	0 = NOT SEGMENTED
READ ONLY BIT 1 = READ ONLY 0 = READ/WRITE	42 & 41 MUST = 00 FOR DATA DESC.	D = DOUBLE-PRECISION BIT 1 = DOUBLE-PRECISION DATA 0 = SINGLE-PRECISION DATA	

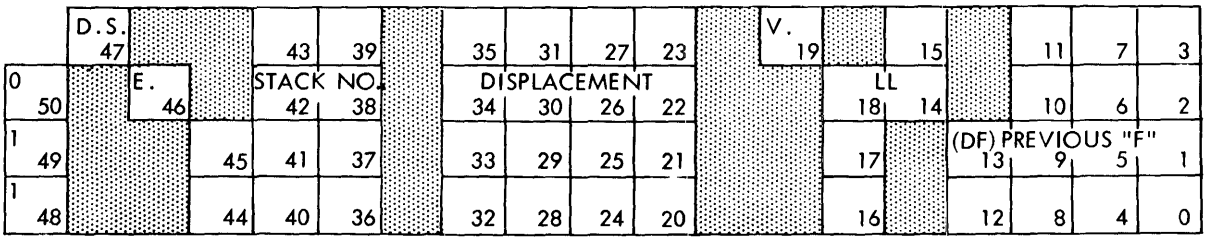


NORMAL INDIRECT REFERENCE WORD

CONTROL WORD FORMATS

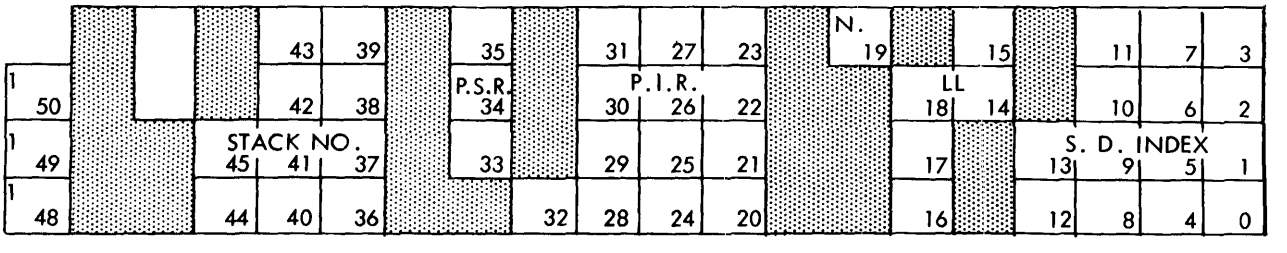


STUFFED INDIRECT REFERENCE WORD



MARK STACK CONTROL WORD

D.S. = DIFFERENT STACK BIT	E. = ENVIRONMENT	V. = VALUE BIT
1= A NON-CURRENT STACK	1 = ACTIVE MSCW	1 = RETURN A VALUE
0 = THIS CURRENT STACK	0 = INACTIVE MSCW	0 = RESTART FROM BEGIN



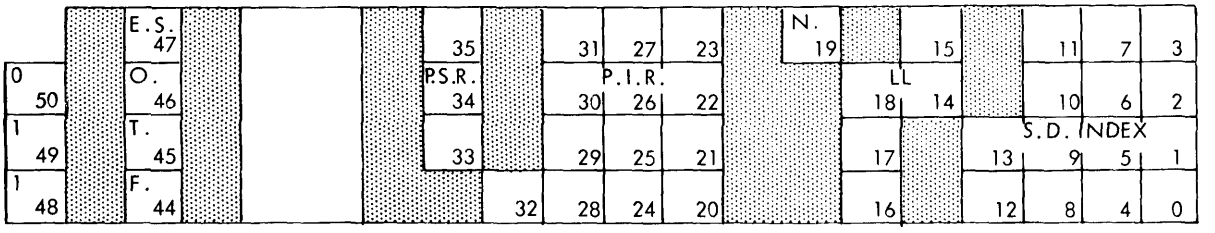
PROGRAM CONTROL WORD

CONTROL WORD FORMAT

N = NORMAL/CONTROL STATE F/F SD = Segment Descriptor

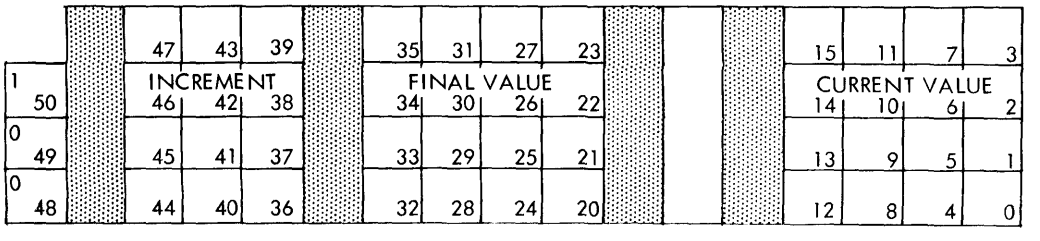
1 = CONTROL STATE

0 = NORMAL STATE



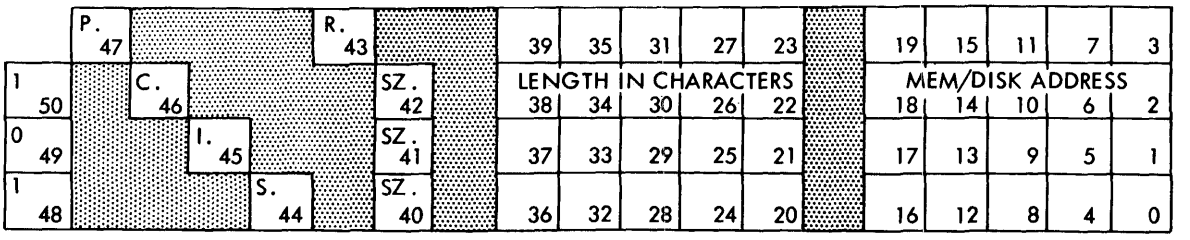
RETURN CONTROL WORD

E.S. = EXTERNAL SIGN BIT 1 = NEGATIVE 0 = POSITIVE	0 = OVERFLOW F/F 1 = OVERFLOW 0 = NO OVERFLOW	Tr = TRACE MODE T = TRUE/FALSE F/F 1 = TRUE 0 = FALSE TFOF = TRUE/FALSE F/F OCCUPIED F/F 1 = TFFF VALID 0 = TFFF NOT DETERMINED
F = FLOAT F/F 1 = FLOAT 0 = NO FLOAT	N = NORMAL/CONTROL F/F 1 = CONTROL STATE 0 = NORMAL STATE	



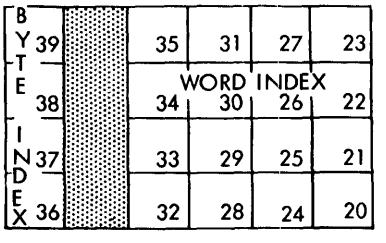
STEP INDEX WORD

CONTROL WORD FORMATS



STRING DESCRIPTOR (NON-INDEXED)

P = PRESENCE BIT	C = COPY BIT	I = INDEX BIT	S = SEGMENTED BIT
1 = PRESENT IN MAIN MEMORY	1 = A COPY		1 = STRING SEGMENTED
0 = NOT PRESENT IN MAIN MEMORY	0 = ORIGINAL	0 = NON-INDEXED	0 = NOT SEGMENTED
R = READ ONLY BIT 1 = READ ONLY 0 = READ/WRITE	SIZE = 4 => 8-BIT BYTE SIZE = 3 => 6-BIT CHARACTER SIZE = 2 => 4-BIT DIGIT		



STRING DESCRIPTOR (INDEXED)

APPENDIX C (cont)

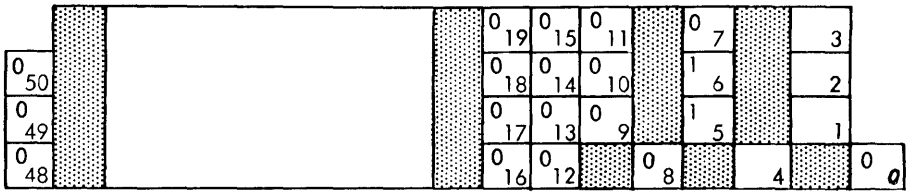
CONTROL WORD FORMATS

P = PRESENCE BIT	C = COPY BIT	I = INDEX BIT	S = SEGMENTED BIT
1 = PRESENT IN MAIN MEMORY	1 = A COPY	1 = INDEXED	1 = STRING SEGMENTED
0 = NOT PRESENT IN MAIN MEMORY	0 = ORIGINAL		0 = NOT SEGMENTED
R = READ ONLY BIT 1 = READ ONLY 0 = READ/WRITE	SIZE = 4 => 8-BIT BYTE SIZE = 3 => 6-BIT CHARACTER SIZE = 2 => 4-BIT DIGIT		

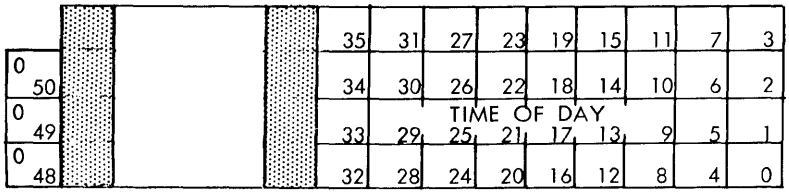
APPENDIX D

SCAN FUNCTION CODE WORDS

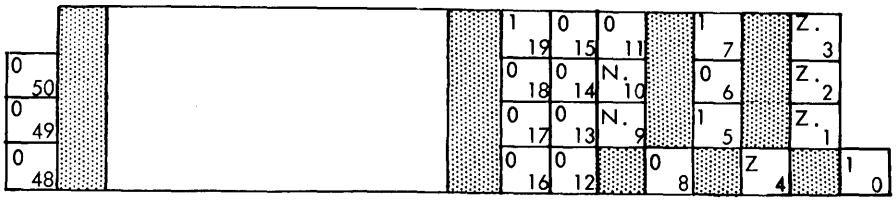
(SCAN IN)



Function Code Read Time of Day Clock (0011)



Time of Day (Binary) Word Returned



Function Code Read General Control Adapter (0101)

Z = 0001, GCA A is to respond

Z = 0010, GCA B is to respond

Z = 0100, GCA C

Z = 1000, GCA D

N = 00, Read GCA Input Register

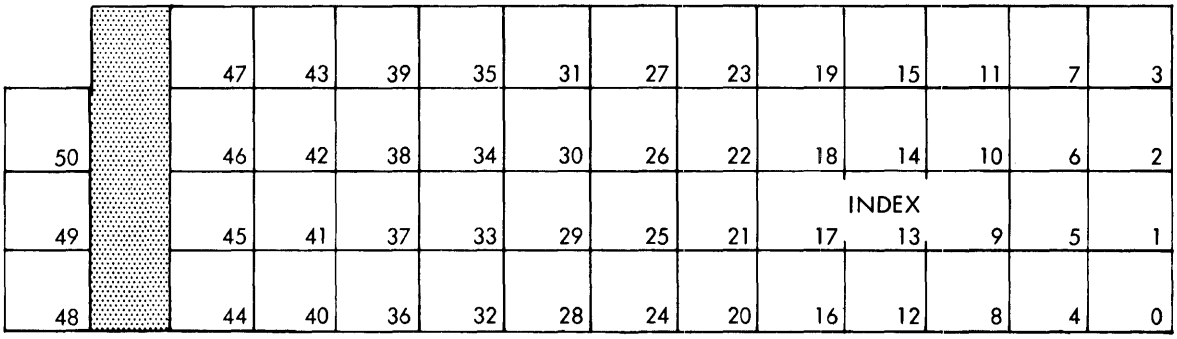
N = 01, Read GCA Interrupt Mask Register

N = 10, Read GCA Interrupt Register

N = 10, Read GCA Output Register

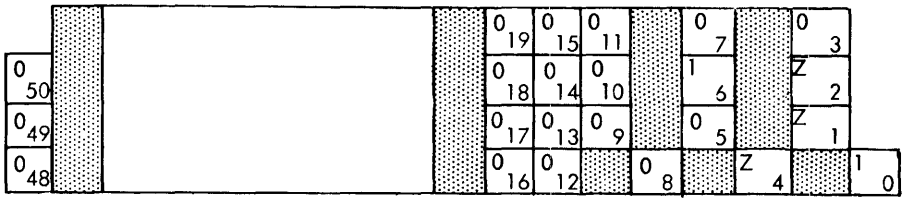
SCAN FUNCTION CODE WORDS

(SCAN IN) (cont)



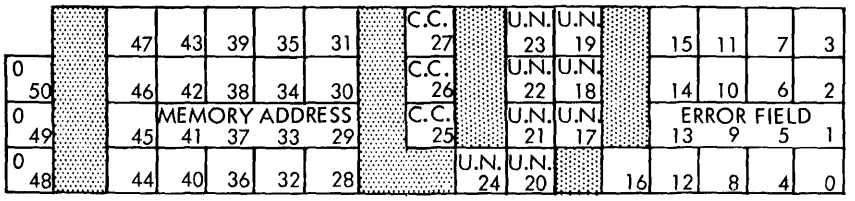
a. G.C.A. Register Word Returned

b. G.C.A. Register Word Sent To Multiplexor



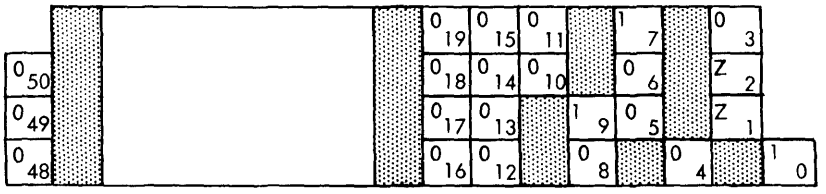
Function Code Read Result Descriptor (0010)

(SCAN IN) (cont)



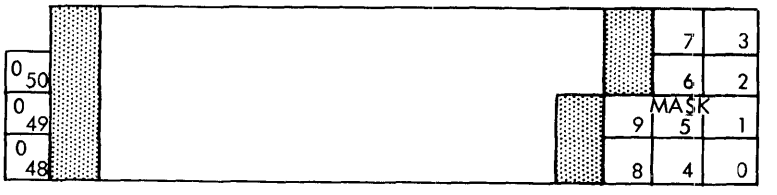
Result Descriptor Word Returned

Bit 0 = Exception
Bit 1 = Software Attention
Bit 2 = Busy
Bit 3 = Not Ready
Bit 4 = Descriptor Error
Bit 5 = Memory Address Error
Bit 6 = Memory Parity Error
Bit 16 = Memory Protection Error
Bits 15:9 are Unit Error Field (see MPX section)



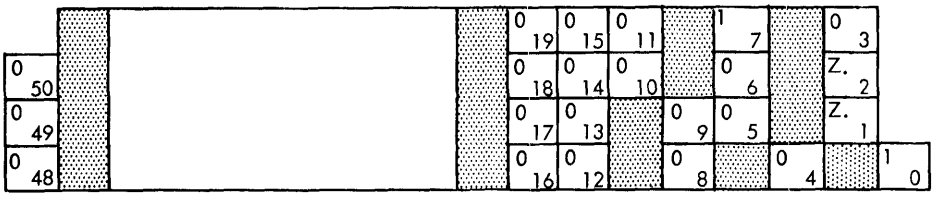
Function Code Read Interrupt Mask (10100)

(SCAN IN) (cont)

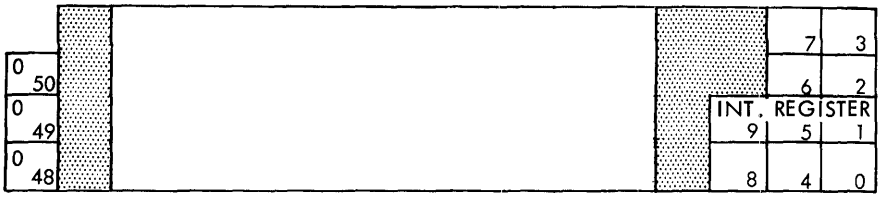


Interrupt Mask Word Returned

Bit 9 = Multiplexor I/O Finish
Bit 1 = Data Comm. Processor 1
Bit 2 = Data Comm. Processor 2
Bit 3 = Data Comm. Processor 3
Bit 4 = Data Comm. Processor 4
Bit 0 = Status Change



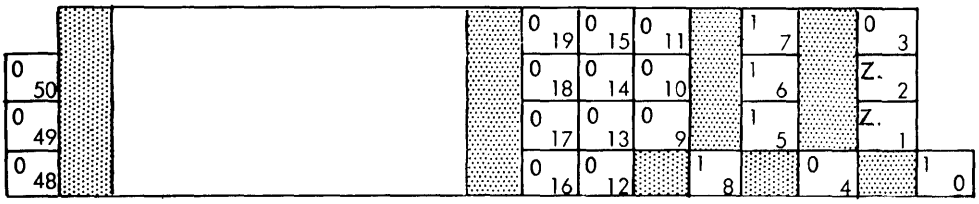
Function Code Read Interrupt Register (0100)



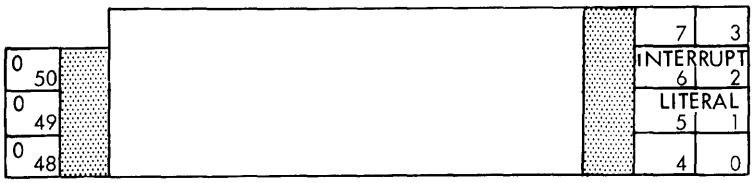
Interrupt Register Word Returned

(SCAN IN) (cont)

Bit 9 = Multiplexor I/O Finish
Bit 1 = Data Comm. Processor 1
Bit 2 = Data Comm. Processor 2
Bit 3 = Data Comm. Processor 3
Bit 4 = Data Comm. Processor 4
Bit 0 = Status Change Interrupt



Function Code Read Interrupt Literal (1111)

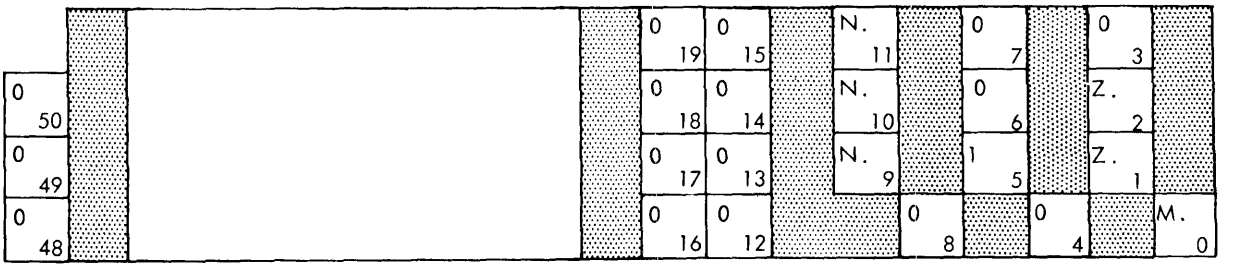


Interrupt Literal Word Returned

Bits 1:2 = 01 = Multiplexor A
10 = Multiplexor B

Bits 7:4 = 0001 = D.C.P. 1
0010 = D.C.P. 2
0011 = D.C.P. 3
0100 = D.C.P. 4
1001 = I/O Finished
1111 = Status Change

(SCAN IN) (cont)



Function Code Interrogate Peripheral Status (0001)

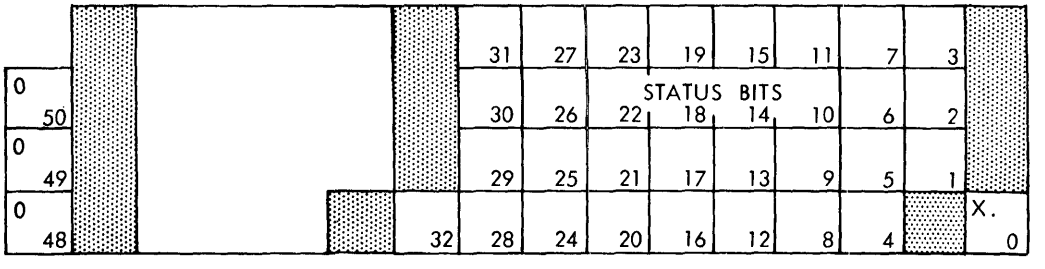
M = 0 = All Multiplexors to respond

M = 1 = Multiplexor designated by Z to respond

Z = 01 = Designates Multiplexor A

Z = 10 = Designates Multiplexor B

N = 0 => 7 Status Vector Number (in Binary)

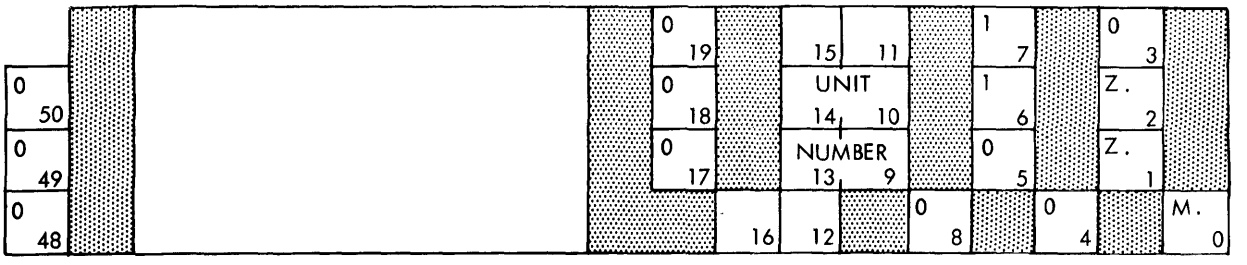


Unit Status Word Returned

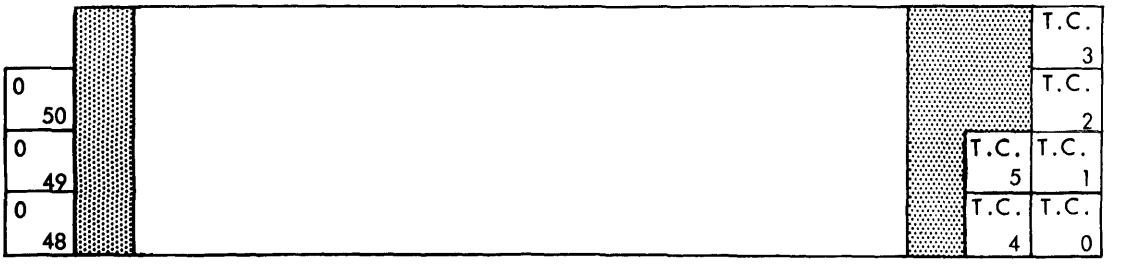
X = 0 = Status word not present

X = 1 = Status word present

(SCAN IN) (cont)



Function Code Interrogate Peripheral Type (0110)

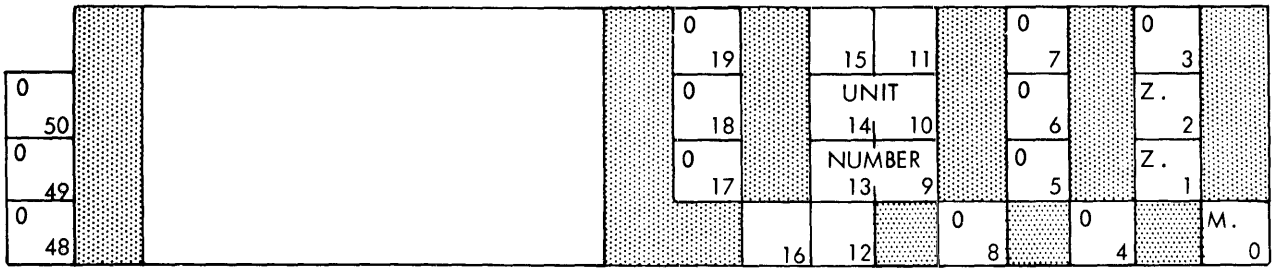


Unit Type Word Returned

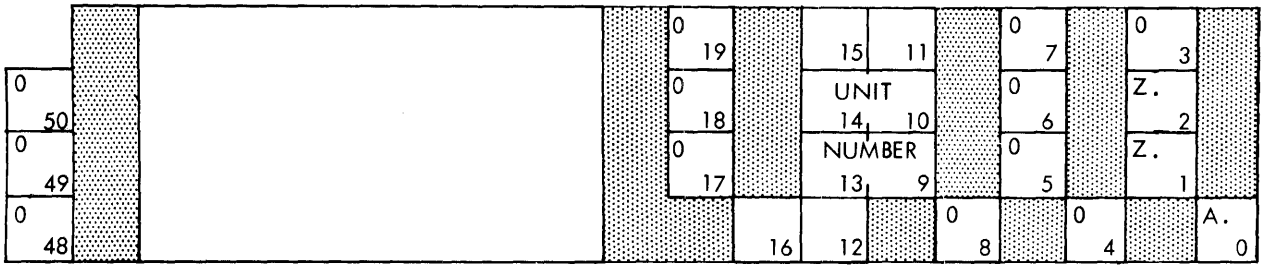
T.C. = 00 = No Unit
01 = Disk File
02 = Display
04 = Paper Tape Reader
05 = Paper Tape Punch
06 = Line Printer, Buffered, BCL drum
07 = Line Printer, Unbuffered, BCL drum
08 = Card Reader
0A = Card Punch
0B = Magnetic Tape (7 track) } Exchange
0C = Magnetic Tape (9 track NRZ) } Exchange
0D = Magnetic Tape (9 track P.E.) } Exchange
1D = Magnetic Tape (7 track) } Serial or Cluster
1E = Magnetic Tape (9 track NRZ) } Serial or Cluster
1F = Magnetic Tape (9 track P.E.) } Serial or Cluster

(SCAN IN) (cont)

T.C. (cont) = 26 = Line Printer, Buffered, EBCDIC drum
27 = Line Printer, Unbuffered, EBCDIC drum



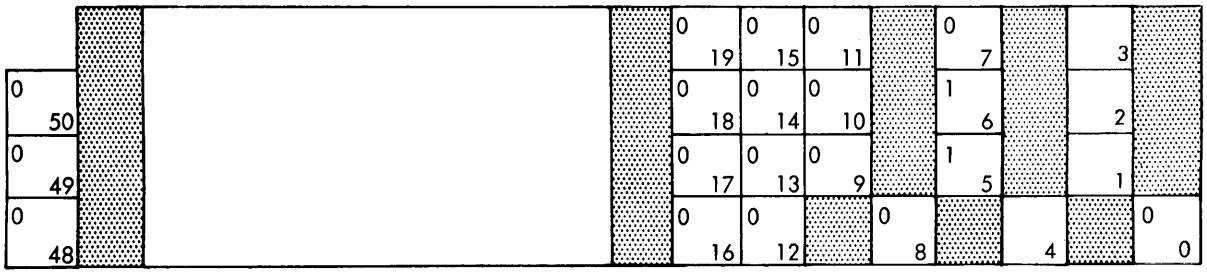
Function Code Interrogate Input/Output Path (0000)



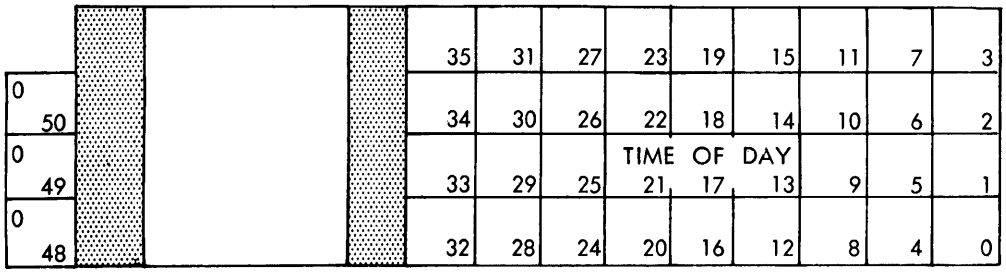
Input/Output Path Word Returned

A = 0 = No Path Available
A = 1 = Path is Available
Z = 01 = Path via Multiplexor A
Z = 10 = Path via Multiplexor B
Z = 11 = Path via Either Multiplexor

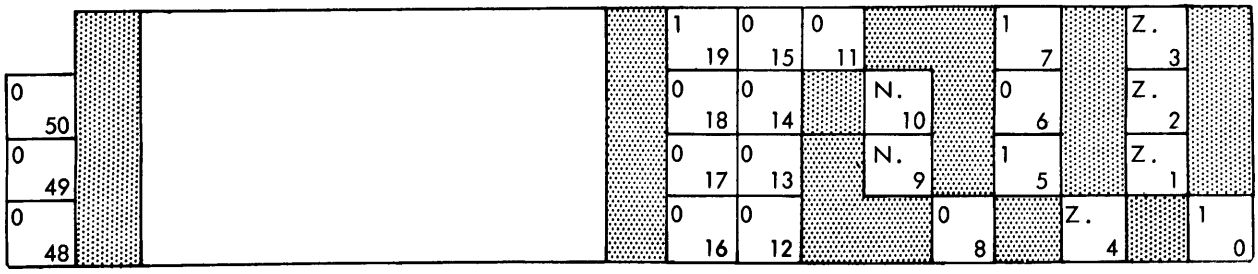
(SCAN OUT)



Function Code Set Time of Day Clock (0011)



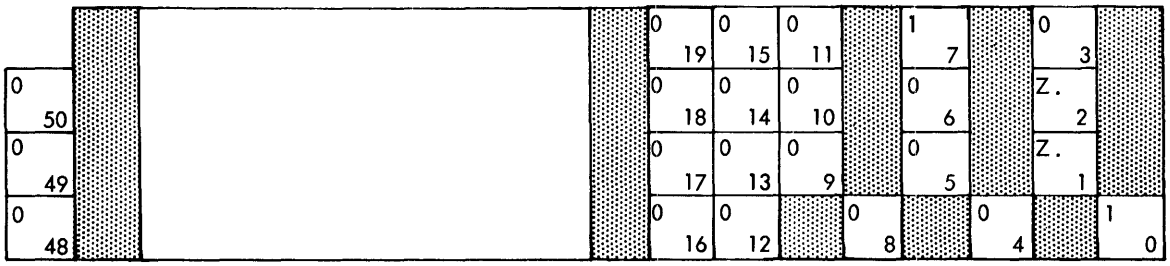
Time of Day Word (Binary) To Multiplexor



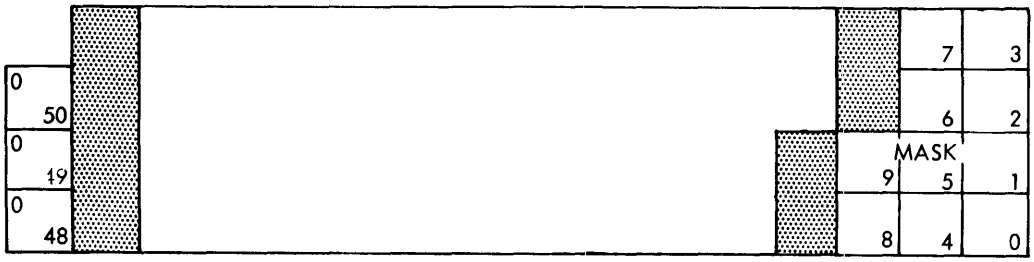
Function Code Set General Control Adapter (0101)

(SCAN OUT) (cont)

Z = 0001 = GCA A is to Respond
Z = 0010 = GCA B is to Respond
Z = 0100 = GCA C
Z = 1000 = GCA D
N = 00 = Set GCA Output Register
N = 01 = Set GCA Interrupt Mask Register
N = 10 = Set GCA Interrupt Register



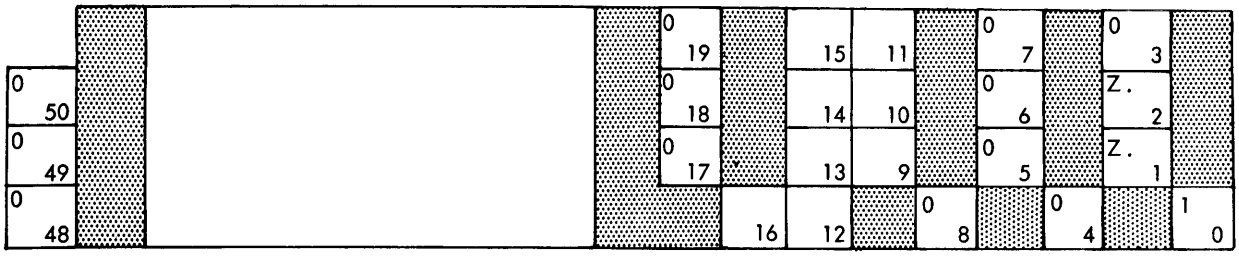
Function Code Set Interrupt Mask (0100)



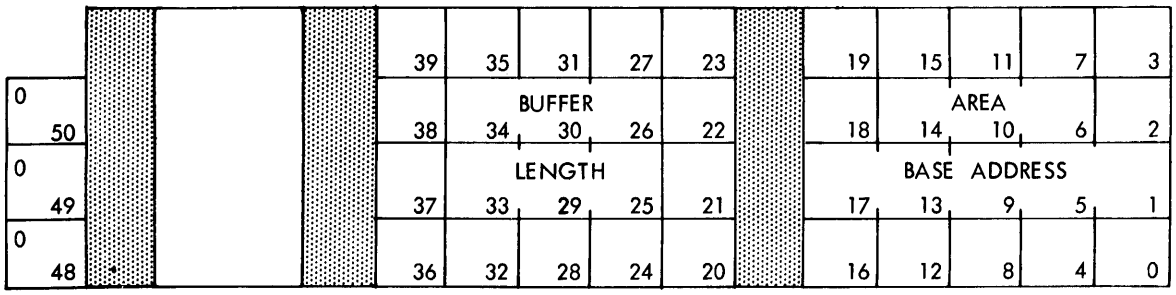
Interrupt Mask Word Sent To Multiplexor

Bit 9 = Multiplexor
Bit 1 = Data Comm. Processor 1
Bit 2 = Data Comm. Processor 2
Bit 3 = Data Comm. Processor 3
Bit 4 = Data Comm. Processor 4
Bit 0 = Status Change Interrupt

(SCAN OUT) (cont)



Function Code Initiate I/O (0000)



Area Descriptor Word Sent To Multiplexor

APPENDIX E
DATA REPRESENTATION

CODES.

GRAPHIC	BCL EXTERNAL	BCL INTERNAL	EBCDIC INTERNAL	HEXADECIMAL GRAPHIC
Blank	01 0000	11 0000	0100 0000	40
.	11 1011	01 1010	0100 1011	4B
[	11 1100	01 1011	0100 1010	4A
(	11 1101	01 1101	0100 1101	4D
<	11 1110	01 1110	0100 1100	4C
←	11 1111	01 1111	0100 1111	4F
&	11 0000	01 1100	0101 0000	50
$	10 1010	10 1010	0101 1011	5B
*	10 1100	10 1011	0101 1100	5C
)	10 1101	10 1101	0101 1101	5D
;	10 1110	10 1110	0101 1110	5E
≤	10 1111	10 1111	0101 1111	5F
-	10 0000	10 1100	0110 0000	60
/	01 0001	11 0001	0110 0001	61
,	01 1011	11 1010	0110 1011	6B
%	01 1100	11 1011	0110 1100	6C
=	01 1110	11 1101	0111 1110	7E
]	01 1110	11 1110	0101 1010	5A
"	01 1111	11 1111	0111 1111	7F
#	00 1011	00 1010	0111 1011	7B
@	00 1100	00 1011	0111 1100	7C
:	00 1101	00 1101	0111 1010	7A
>	00 1110	00 1110	0110 1110	6E
≥	00 1111	00 1111	0111 1101	7D

DATA REPRESENTATION

GRAPHIC	BCL EXTERNAL	BCL INTERNAL	EBCDIC INTERNAL	HEXADECIMAL GRAPHIC
+	11 1010	01 0000	1100 0000	C0
A	11 0001	01 0001	1100 0001	C1
B	11 0010	01 0010	1100 0010	C2
C	11 0011	01 0011	1100 0011	C3
D	11 0100	01 0100	1100 0100	C4
E	11 0101	01 0101	1100 0101	C5
F	11 0110	01 0110	1100 0110	C6
G	11 0111	01 0111	1100 0111	C7
H	11 1000	01 1000	1100 1000	C8
I	11 1001	01 1001	1100 1001	C9
x (Mult.)	10 1010	10 0000	1101 0000	D0
J	10 0001	10 0001	1101 0001	D1
K	10 0010	10 0010	1101 0010	D2
L	10 0011	10 0011	1101 0011	D3
M	10 0100	10 0100	1101 0100	D4
N	10 0101	10 0101	1101 0101	D5
O	10 0110	10 0110	1101 0110	D6
P	10 0111	10 0111	1101 0111	D7
Q	10 1000	10 1000	1101 1000	D8
R	10 1001	10 1001	1101 1001	D9
≠	01 1010	11 1100	0110 1101	6D
S	01 0010	11 0010	1110 0010	E2
T	01 0011	11 0011	1110 0011	E3
U	01 0100	11 0100	1110 0100	E4
V	01 0101	11 0101	1110 0101	E5
W	01 0110	11 0110	1110 0110	E6
X	01 0111	11 0111	1110 0111	E7
Y	01 1000	11 1000	1110 1000	E8
Z	01 1001	11 1001	1110 1001	E9

DATA REPRESENTATION

GRAPHIC	BCL EXTERNAL	BCL INTERNAL	EBCDIC INTERNAL	HEXADECIMAL GRAPHIC
0	00 1010	00 0000	1111 0000	F0
1	00 0001	00 0001	1111 0001	F1
2	00 0010	00 0010	1111 0010	F2
3	00 0011	00 0011	1111 0011	F3
4	00 0100	00 0100	1111 0100	F4
5	00 0101	00 0101	1111 0101	F5
6	00 0110	00 0110	1111 0110	F6
7	00 0111	00 0111	1111 0111	F7
8	00 1000	00 1000	1111 1000	F8
9	00 1001	00 1001	1111 1001	F9
?	00 0000	00 1100	0110 1111	ALL OTHER CODES (see notes)

NOTES

a. EBCDIC 0100 1110 also translates to BCL 11 1010.

b. EBCDIC 1111 is translated to BCL 00 0000 with an additional flag bit on the most significant bit line (8th bit). This function is used by the unbuffered printer to stop scanning.

c. EBCDIC 1110 0000 is translated to BCL 00 0000 with an additional flag bit on the next to most significant bit line (7th bit). As the print drums have 64 graphics and space this signal can be used to print the 64th graphic. The 64th graphic is a "CR" for BCL drums and a "¢" for EBCDIC drums.

d. The remaining 189 EBCDIC codes are translated to BCL 00 0000 (? code).

e. The EBCDIC graphics and BCL graphics are the same except as follows:

	BCL	EBCDIC
1)	≥	' (single quote)
2)	x (multiply)	!
3)	≤	¬ (not)
4)	≠	_ (underscore)
5)	←	|

	ZONE	+ 9	- 9	0 9	9	+ 0 9	+ - 9	- 0 9	+ - 0 9	+ 0	+ -	- 0	+ - 0	+	-	0				
NUM	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX	NUM	
81	0	NUL	DLE			SP	&	-						+{	-}	/	0	0	81	
1	1	SOH	DC1					/		a	j	~		A	J		1	1	1	
2	2	STX	DC2		SYN					b	k	s		B	K	S	2	2	2	
3	3	ETX	DC3							c	l	t		C	L	T	3	3	3	
4	4									d	m	u		D	M	U	4	4	4	
5	5	HT		LF						e	n	v		E	N	V	5	5	5	
6	6		BS	ETB						f	o	w		F	Ø	W	6	6	6	
7	7	DEL		ESC	EOT					g	p	x		G	P	X	7	7	7	
8	8		CAN			←				h	q	y		H	Q	Y	8	8	8	
81	9		EM			≤	≠	≥	.	i	r	z		I	R	Z	9	9	9	
82	A					[	]	!	:									A	82	
83	B					o	$	,	.#									B	83	
84	C	FF	FS		DC4	<	*	%	@									C	84	
85	D	CR	GS	ENQ	NAK	(	)	_	'									D	85	
86	E	SO	RS	ACK		+	;	>	=									E	86	
87	F	SI	US	BEL	SUB	|	¬	?	"									F	87	
NUM	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX	NUM	
	ZONE	9 +	9 -	9 0	9		-	0		0	- +	-	- +	9 0	9	9 0	9 0 -			

B 6500 EBCDIC/HEX CARD CODE

Use of the B 6500 EBCDIC/HEX Card Code Chart.

a. Locate the desired EBCDIC graphic code within the table.

b. The two-part Hexadecimal Code is read as follows:

1) The first part is found in the vertical column above or below the desired EBCDIC code.

2) The second part is found in the horizontal row either to the right or left of the desired EBCDIC code.

a) Examples:

SYN = 32
F = C6

c. The two-part Card Code is found in the same manner as HEX (b) except the zone and numeric bits are read from the very outer portion of the table.

1) Examples:

SYN = 9 2
F = + 6

2) The card code exceptions to the above procedure are enclosed in heavy lines on the chart and are defined below:

a) 00 = + 0981 (NUL)
b) 10 = + -981 (DLE)
c) 20 = - 0981
d) 30 = + -0981
e) 40 = BLANK
f) 50 = + (&)
g) 60 = - (-)

h) 70 = + - 0
i) C0 = + 0 ({) ($\overset{+}{0}$)
j) D0 = - 0 (}) ($\overset{-}{0}$)
k) E0 = 0 82 (\)
l) F0 = 0 (0)
m) 61 = 0 1 (/)
n) E1 = -09 1
o) 6A = + - (¦)

APPENDIX G

HEXADECIMAL-DECIMAL CONVERSION TABLE

The table in this appendix provides for direct conversion of decimal and hexadecimal numbers in the ranges:

Hexadecimal	Decimal
000 to FFF	0 to 4095

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

Hexadecimal	Decimal
1000	4096
2000	8192
3000	12288
4000	16384
5000	20484
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	53248
E000	57344
F000	61440

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
010	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
020	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
030	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
040	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
050	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
060	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
070	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
080	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
090	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
0A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
0B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
0C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
0D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
0E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
0F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
100	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271
110	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287
120	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303
130	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
140	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335
150	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351
160	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367
170	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383
180	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399
190	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415
1A0	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431
1B0	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447
1C0	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463
1D0	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479
1E0	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495
1F0	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
200	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527
210	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543
220	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559
230	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575
240	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591
250	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607
260	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623
270	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639
280	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655
290	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671
2A0	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687
2B0	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703
2C0	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719
2D0	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735
2E0	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751
2F0	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767
300	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783
310	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799
320	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815
330	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831
340	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847
350	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863
360	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879
370	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895
380	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911
390	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927
3A0	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943
3B0	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959
3C0	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975
3D0	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991
3E0	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

APPENDIX G (cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

INDEX

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: _____ FORM: _____

_____ DATE: _____

_____

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

cut along dotted line

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

STAPLE

FOLD DOWN SECOND FOLD DOWN

Postage Will Be Paid by Addressee

No Postage Stamp Necessary If Mailed in the United States

BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
6071 Second Avenue
Detroit, Michigan 48232

attn: Sales Technical Services
Systems Documentation

FOLD UP FIRST FOLD UP


Wherever There's
Business There's
Burroughs