Burroughs B6700

Information Processing Systems

REFERENCE MANUAL

Wolf

Burroughs
B 6700
INFORMATION PROCESSING SYSTEMS
REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

PRICED ITEM

Printed in U. S. America

5-72

1058633

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 addressed directly to Burroughs Corporation, P.O. Box 4040, El Monte, California 91734, Attn: Publications Department, TIO–West.

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)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

LIST OF ILLUSTRATIONS (cont)

LIST OF ILLUSTRATIONS (cont)

LIST OF TABLES

INTRODUCTION

The Burroughs B 6700 is a medium to large, high-speed Information Processing System. The following are some of the features incorporated in this system:

1. Monolythic Circuitry.
2. Memory expandable to 1,048,576 words.
3. Memory Cycle Times of 1.2 microseconds, 1.5 microseconds, and 500 nanoseconds.
4. Peripheral configuration expandable to 256 units.
5. Triple – Input/Output Processor system permitting up to 36 simultaneous Input/Output (I/O) operations.
6. Data Communication Software for remote computing and file manipulation.
7. 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 6700 hardware and software. Whereas hardware traditionally 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 6700 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 6700. 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 6700 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 6700 offers a choice of problem-oriented languages, some of these languages are: 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 6700 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 6700 system. Because of the design concept of the B 6700, 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 6700 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 6700 is a compiler oriented system designed to accept the problem-oriented 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 in the source language.

DESCRIPTION OF UNITS

The B 6700 system configuration varies with application and workload requirements. The minimum system includes one processor, one disk file storage unit, one magnetic tape unit, one input/output processor, one memory module, one console display unit, one card reader and one printer. The maximum system configuration includes 3 processors, 64 memory modules (16,384 words each), 3 input/output processors, 60 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 10. The Data Communications Sub-System is defined in section 11.

Table 1-1.

B 6700 Central Units Chart

Style No.	Description	Processor Speed (MHz)	No. Of Input/ Output Processors	Add'l. Input/ Output Processors	Notes
B 6711	One processor	2.5/2.5	1	0	Used only with 65k memory modules.
B 6721	One processor (can have a B 6721-1 second processor)	2.5/2.5	1	1	Used only with 65k memory modules.
B 6712	One processor	2.5/5.0	1	1 or 2	Used with any of the available memory modules.
B 6722	Two processors	2.5/5.0	1	1 or 2	Used with any of the available memory modules.
B 6714	One processor	5.0/5.0	1	1 or 2	Used with any of the available memory modules.
B 6724	Two processors	5.0/5.0	1	1 or 2	Used with any of the available memory modules.
B 6734	Three processors	5.0/5.0	2	1	Used with any of the available memory modules.

Table 1-1. (Cont'd.)
B 6700 Central Units Chart

Style No.	Description	Processor Speed (MHz)	No. Of Input/ Output Processors	Add'l. Input/ Output Processors	Notes
B 6780	Input/Output Processor				
B 6780-1	Data switching channel, up to 12 per input/ output processor	–	–	–	Optional.
B 6790	Maintenance Diagnostic Logic Processor for B 6722, B 6724, & B 6734 Systems (second input/output processor is required).	–	–	–	Optional.
B 6791	Power Supply	–	–	–	Optional.
B 6000	Memory control cabinet	–	–	–	Optional.
B 6004-1	98,304 bytes (16,384 words (1.2 μs memory module	–	–	–	–
B 6005-1	393,216 bytes (65,536 words) 1.5 μs memory module	–	–	–	–
B 6006-1	98,304 bytes (16,384 words) 500 ns memory module	–	–	–	–

SYSTEM OPTIONS AND REQUIREMENTS

The following lists the requirements and some of the available options for the B 6700 systems:

1. A minimum of one special DC module is required in a B 6700 system. It can be installed in the following cabinets:
   a. Input/Output Processor.
   b. Processor.
   c. Peripheral Control.
   d. Data Communications.
2. A minimum of one ± 12 volt inverter module is required in a B 6700 system. It can be installed in the following cabinets:
   a. Input/Output Processor.
   b. Processor.
   c. Peripheral Control.

NOTE

The use of this module in a cabinet precludes the use of any other module in that same cabinet.

3. A Flip Flop display supply module is required on the system and must be installed in the Input/Output Processor cabinet.
4. The Memory cabinets each must contain a special Memory supply for developing the regulated voltages required for the memory operation.
5. Each cabinet must contain an inverter for supplying power to its regulators. A 600 ampere inverter is required in the Processor, Input/Output Processor and Data Communications cabinets. All other cabinets require a 400 ampere inverter.

Auxiliary Cabinet

Peripheral unit exchanges are located within auxiliary cabinets of the B 6700 system. These cabinets can accommodate varying combinations of exchanges. Two of the combinations that are possible are shown in figure 1-1.

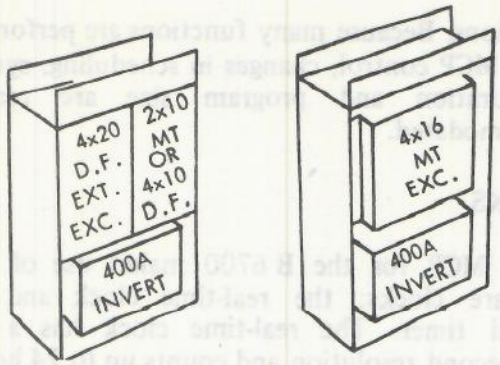


Figure 1-1. Auxiliary Cabinets

The following exchanges are available for use on the B 6700 system:

1. Tape Exchange
   - 2 x 10
   - 2 x 8
   - 4 x 16
2. Disk File Exchange
   - 1 x 2
   - 2 x 5
   - 4 x 10
   - 4 x 20

Disk File Optimizer

The disk file optimizer functions to optimize the transfer of information between a processor of the B 6700 system and its associated disk file subsystem in order to improve the transfer rate. A detailed description of the disk file optimizer is given in section 12 of this manual.

System Power

Main power is supplied to the system by 1 to 15 free standing AC power cabinets. Each power cabinet can furnish enough power for eight B 6700 cabinets. The power cabinets receive 3 phase AC from the wall breakers and convert it to 220 volt pulsating direct current. Each B 6700 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 (overvoltage/undervoltage) indication panel. Refer to figure 1-2 for a typical B 6700 power supply configuration.

Peripheral Control Cabinet

The PC cabinet can accommodate up to 10 peripheral controls. A maximum of five large controls can be used with up to five small controls. Some of the small controls may be used in place of the large controls.

The following controls are available:

1. Large
   a. Magnetic tape
   b. Disk file
   c. Console Display
2. Small
   a. Card reader
   b. Card punch
   c. Line printer
   d. Paper tape reader
   e. Paper tape punch

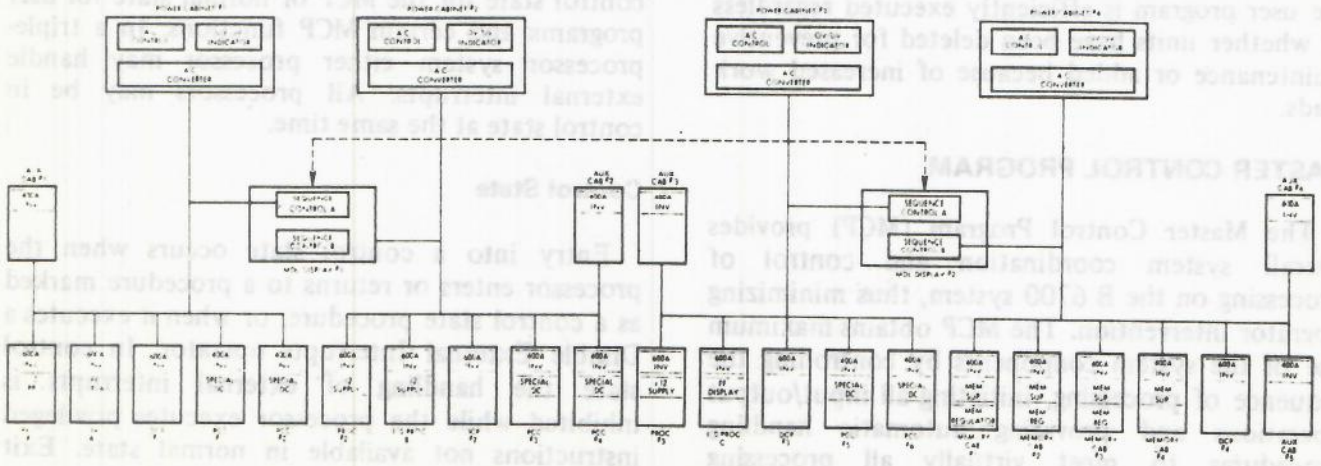


Figure 1-2. B 6700 Power Supply

Some of the controls have a two-byte buffer and others contain a one-byte buffer; therefore, either 8 or 16 bits may be transferred in parallel to the Input/Output Processor at a time. Local operations are performed by attaching a "Control switch" plug-on and "Indicators" plug-ons to various cards in the control.

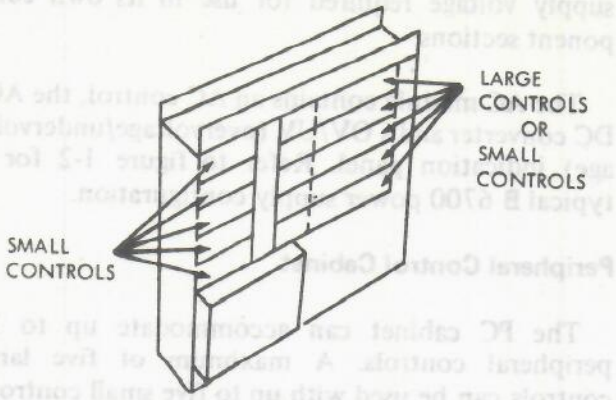


Figure 1-3. Peripheral Control Cabinet

SYSTEM ORGANIZATION

Computer systems are generally organized around a central system that controls memory accesses, establishes I/O priority, etc. In the B 6700 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 input/output processor, 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 regardless of 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 6700 system, thus minimizing operator intervention. The MCP obtains maximum use of the system conponents 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 6700 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 input/output processor 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.

PROCESSOR

The B 6700 system accommodates either one, two or three processors, either capable of accessing any portion of total memory.

All B 6700 processors are multiprocessing machines with available clock frequencies of 2.5 megahertz and 5 megahertz. (Refer to table 1-1.) Processors with different clock rates cannot be intermixed on the same system. The processor is basically word oriented, but has extensive multi-word string manipulation capabilities for four-bit, six-bit, and eight-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 triple-processor system either processor may handle external interrupts. All 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 when it 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 state 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 has occurred, return to the user's program may or may not be to the program that was operating when the interrupt occurred.

Normal State

Normal state excludes use of privileged instructions required by the MCP but 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:

1. Program code cannot be modified while in residence.
2. Hardware stack features provide efficient handling of temporary storage and subroutine requirements.
3. Control bits in each word provide efficient MCP or hardware action, depending upon the state of the control bits.
4. 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.
5. The B 6700 processor is designed to implement higher-level languages and to function under MCP control.
6. 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 6700 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 (interleaved processing) is employed. A dual- or triple-processor B 6700 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 communications control, the interrupt transfers control to the MCP thereby initiating operations that can proceed simultaneously with computation. Some MCP functions are as follows: 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 6700 system is provided with a private, internal interrupt network to handle processor-dependent interrupts. Interrupts generated within the processor are fed into this network and serviced by that processor. The processors also share the handling of external interrupts generated by input/output operations occurring on any input/output processor. The command structure, in conjunction with a stack, provides the implementation of string notation and automatic linking of subroutines.

Interrupt Handling

An interrupt causes the processor to perform the following sequence of operations:

1. Mark the stack.
2. Insert into the stack an Indirect Reference Word, which addresses a reserved location of the stack where a link to the MCP interrupt routine has been stored.
3. Push all pertinent registers into the stack.
4. Insert into the stack an integer value defining the interrupt.
5. Insert a second parameter into the stack, giving other information about the interrupt.
6. Execute an Enter Operator.

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

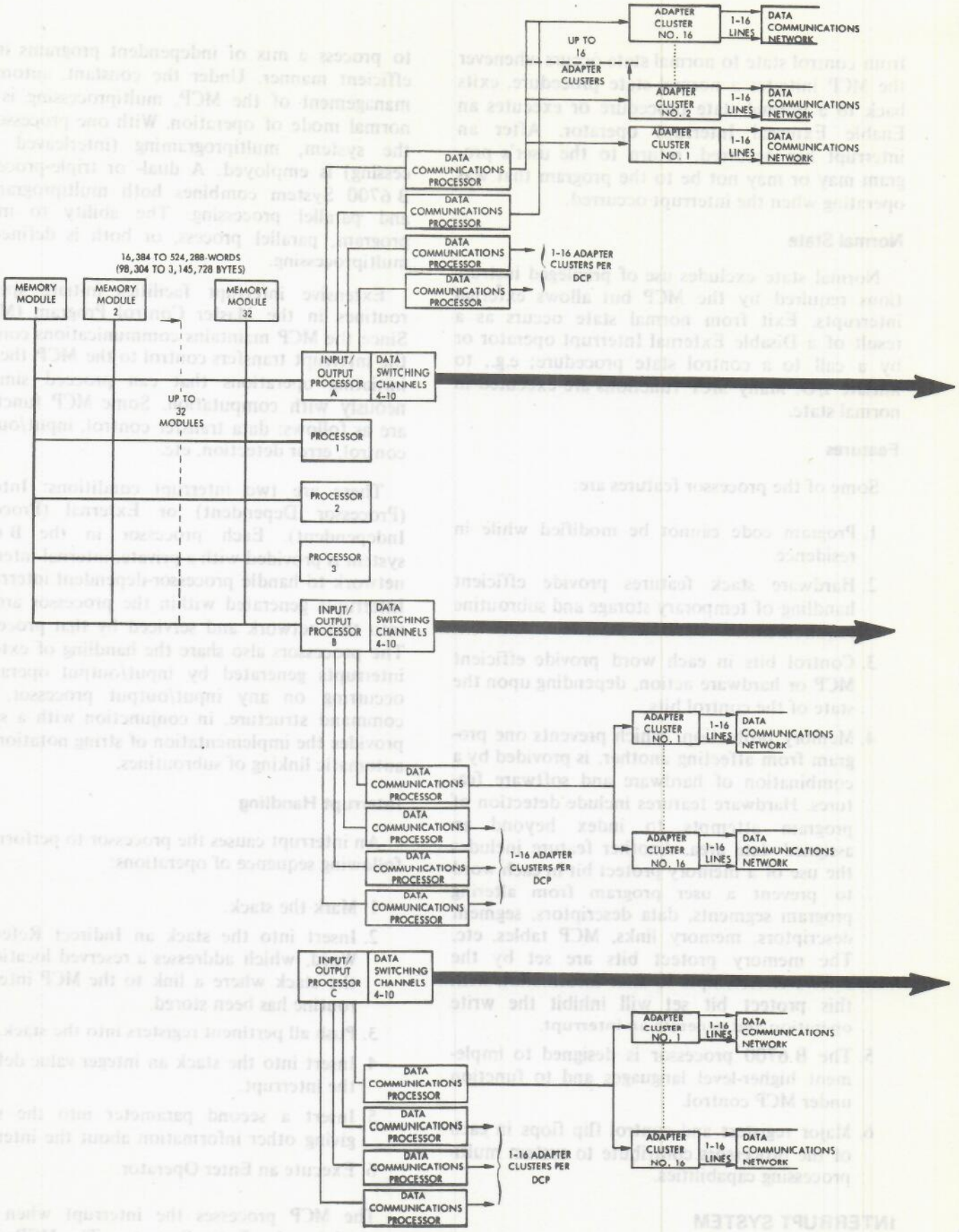


Figure 1-4. B 6700 Representative Configuration (sheet 1 of 2)

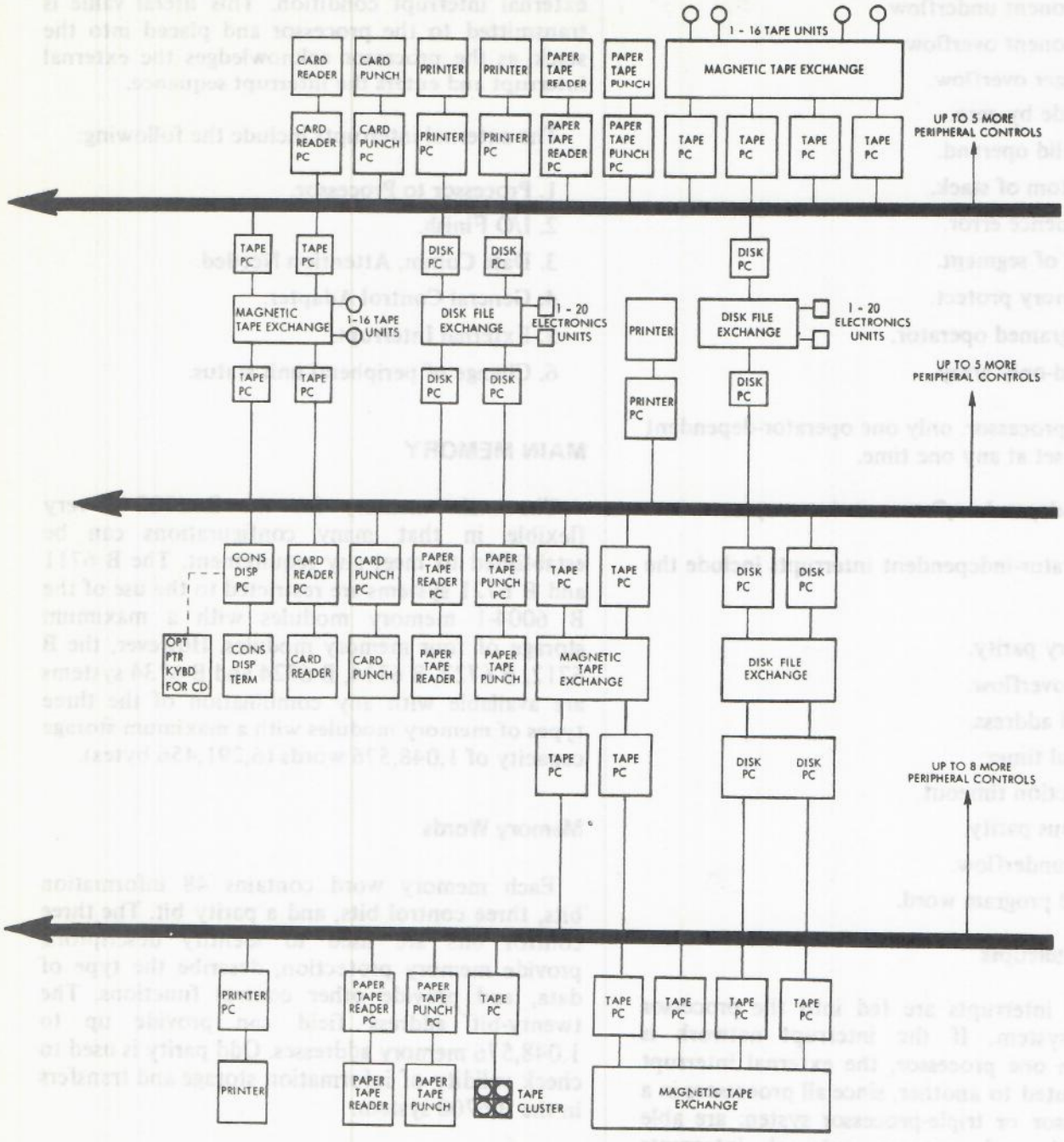


Figure 1-4. B 6700 Representative Configuration (sheet 2 of 2)

Operator-Dependent Processor Interrupts

The interrupts listed below are set only by the action of operators.

1. Presence bit.
2. Invalid index.
3. Exponent underflow.
4. Exponent overflow.
5. Integer overflow.
6. Divide by zero.
7. Invalid operand.
8. Bottom of stack.
9. Sequence error.
10. End of segment.
11. Memory protect.
12. Programed operator.
13. Read-only array.

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

Operator-Independent Processor Interrupts

The operator-independent interrupts include the following:

1. Memory parity.
2. Stack overflow.
3. Invalid address.
4. Interval timer.
5. Instruction timeout.
6. Scan bus parity.
7. Stack underflow.
8. Invalid program word.

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 another, since all processors in a dual-processor or triple-processor system are able to respond and process external interrupts independently and simultaneously. The ability of any processor to handle interrupts is made possible because of a distributed interrupt network and the ability of all processors to be in control state at the same time. The activities of all processors in control state are coordinated (interlocked) by the software through the use of the Read With Lock mechanism. If all 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 the following:

1. Processor to Processor.
2. I/O Finish.
3. Data Comm. Attention Needed.
4. General Control Adapter.
5. External Interrupt.
6. Change of peripheral-unit status.

MAIN MEMORY

The main memory for the B 6700 is very flexible in that many configurations can be established to meet any requirement. The B 6711 and B 6721 systems are restricted to the use of the B 6004-1 memory modules with a maximum storage of four memory modules. However, the B 6712, B 6722, B 6714, B 6724 and B 6734 systems are available with any combination of the three types of memory modules with a maximum storage capacity of 1,048,576 words (6,291,456 bytes).

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 address field can provide up to 1,048,576 memory addresses. Odd parity is used to check validity of information storage and transfers in the B 6700 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

Three types of main memory modules are available for the B 6700 with the following cycle times:

1. 1.5 microsecond cycle time with 65,536 words of storage per module.
2. 1.2 microsecond cycle time with 16,384 words of storage per module.
3. 500 nanosecond cycle time with 16,384 words of storage per module.

Refer to table 1-1 for the B 6700 central units chart for additional information concerning main memory.

SECOND LEVEL MEMORY

Burroughs head-per-track disk file subsystems provide the user with virtually unlimited expansion capability. The 23- to 40-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 PROCESSOR

The Input/Output Processor and associated peripheral control modules are used to control the transfer of data between memory and all peripheral equipment, independent of the processor. The input/output processor receives instructions from the processor and, with its associated peripheral controls, executes these instructions. One, two or three input/output processors may be used with the B 6700 System. Each input/output processor is capable of processing up to ten simultaneous I/O operations with up to 20 peripheral units.

Input/Output Processer Configuration

Each Input/Output Processor provides four separate and independent units:

1. Data switching channels which provide the necessary linkage between the peripheral device (excluding data communications) and main memory.
2. Data communications processors which permit interfacing of remote devices to the B 6700.
3. Real-time adapters which permit interfacing of real time devices such as wind tunnels and rocket stands.
4. 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," and are assigned by the input/output processor to peripheral controls upon initiation of an operation and released to the input/output processor 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 input/output processor 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 three input/output processors (20 controls per input/output processor) can be expanded further through use of exchanges. Figure 1-5 illustrates how the exchanges interact between the magnetic tape controls and the magnetic tape units. Figure 1-6 depicts how the exchanges interact between the disk file controls and the disk file units.

Peripheral Control Bus

A peripheral control (P.C.) bus extends from the input/output processor 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.

Processor-Initiated I/O Operations

Any processor can initiate an I/O operation on any input/output processor (in a three processor/three input/output processor configuration) by

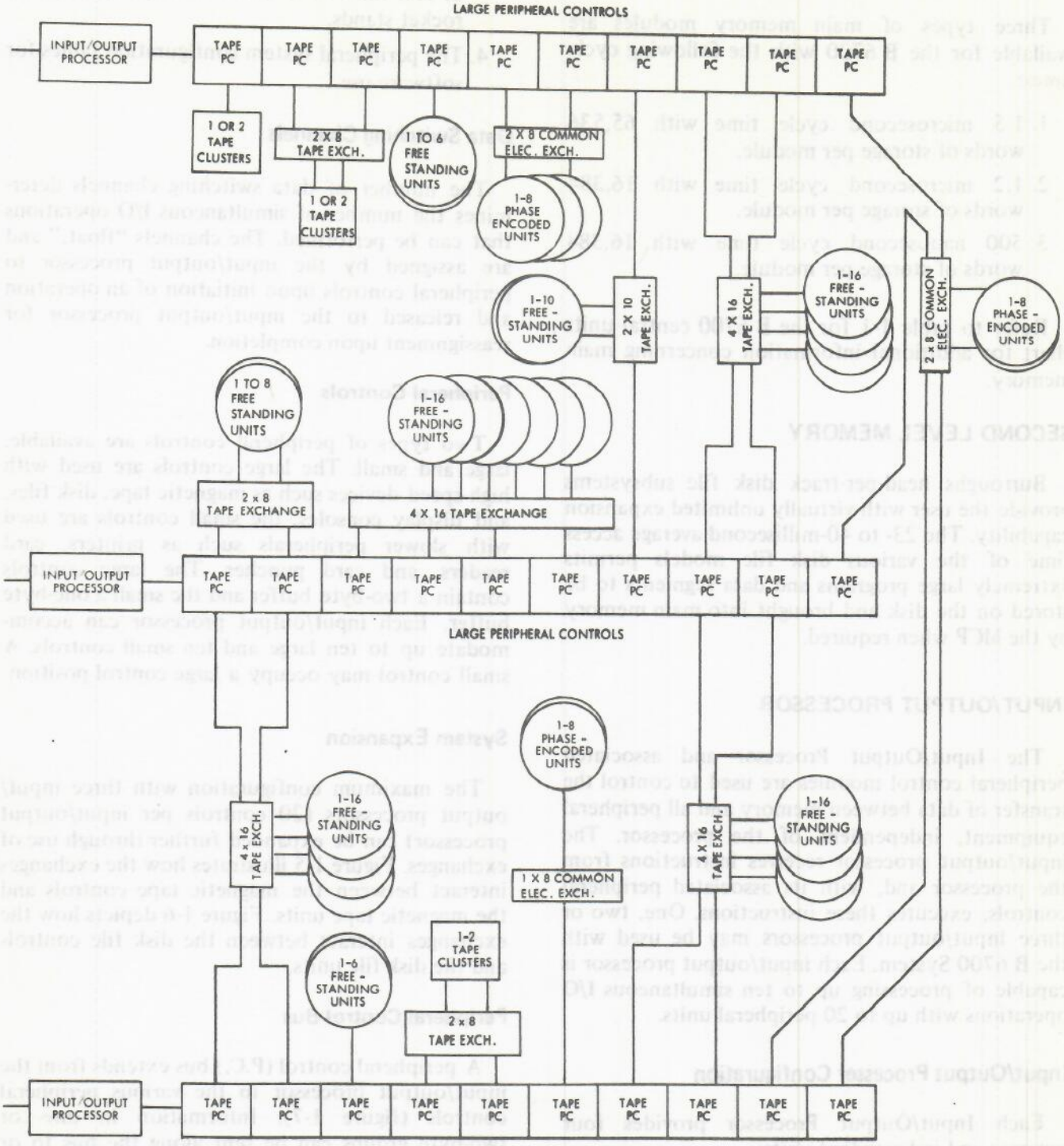


Figure 1-5. Magnetic Tape Subsystem Relationships

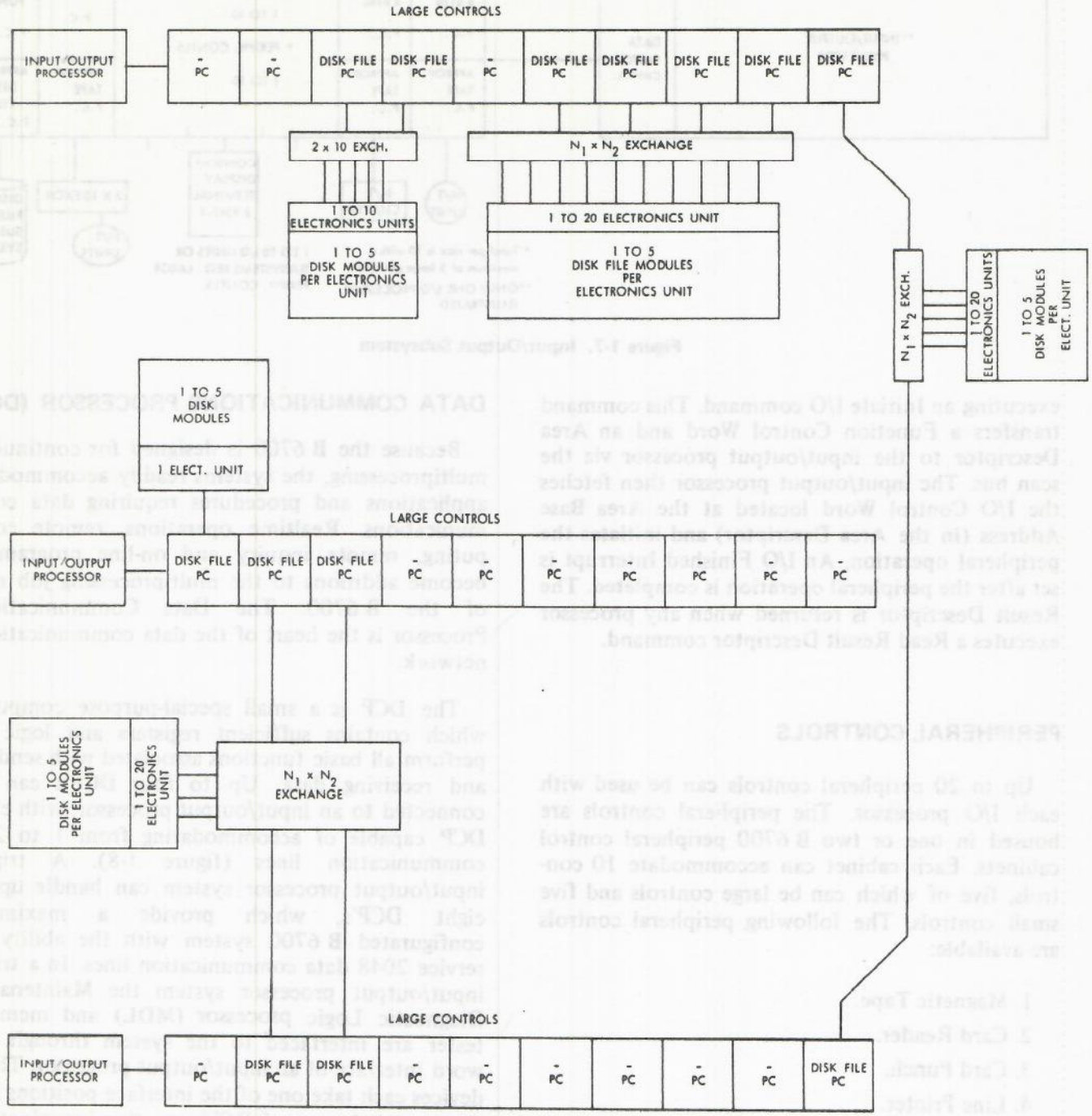


Figure 1-6. Disk File Subsystem Relationships

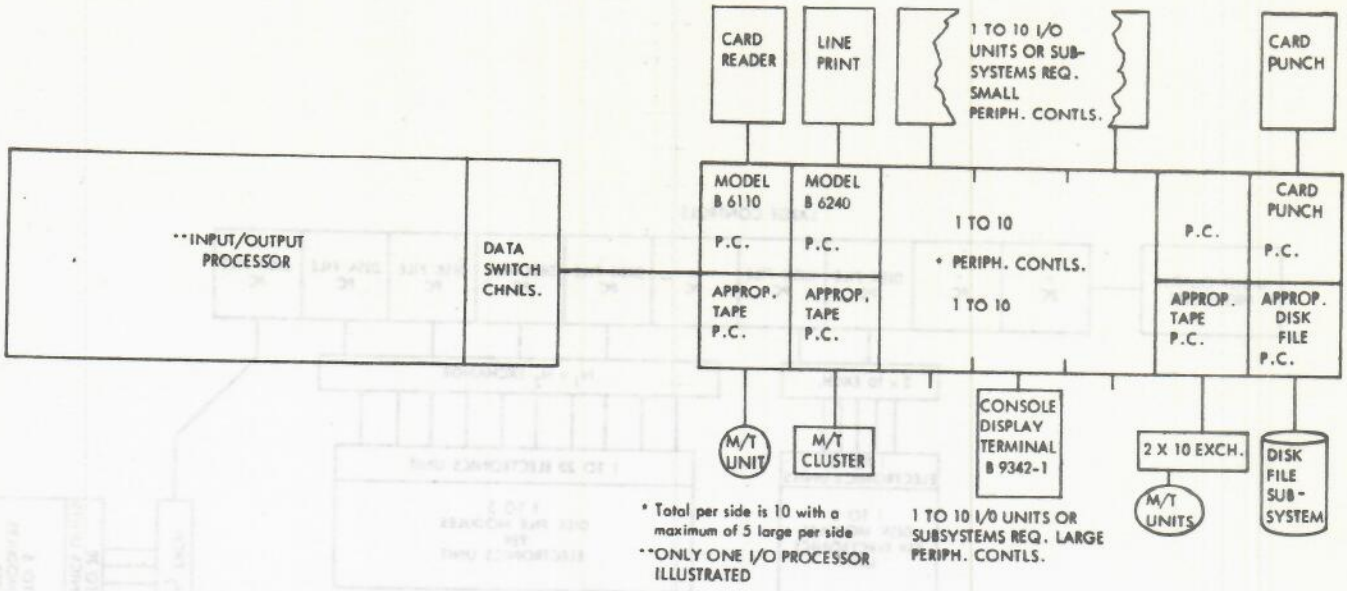


Figure 1-7. Input/Output Subsystem

executing an Initiate I/O command. This command transfers a Function Control Word and an Area Descriptor to the input/output processor via the scan bus. The input/output processor 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 any processor executes a Read Result Descriptor command.

PERIPHERAL CONTROLS

Up to 20 peripheral controls can be used with each I/O processor. The peripheral controls are housed in one or two B 6700 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:

1. Magnetic Tape.
2. Card Reader.
3. Card Punch.
4. Line Printer.
5. Paper Tape Reader.
6. Paper Tape Punch.
7. Disk File.
8. Console Monitor and Keyboard.
9. Disk-Pack Drive.

DATA COMMUNICATIONS PROCESSOR (DCP)

Because the B 6700 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 6700. The Data Communications Processor is the heart of the data communications network.

The 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 input/output processor, with each DCP capable of accommodating from 1 to 256 communication lines (figure 1-8). A triple-input/output processor system can handle up to eight DCP's, which provide a maximum configurated B 6700 system with the ability to service 2048 data communication lines. In a triple input/output processor system the Maintenance Diagnostic Logic processor (MDL) and memory tester are interfaced to the system through the word interface of an input/output processor. These devices each take one of the interface positions and thus preclude use of DCP's on this input/output processor.

DATA COMMUNICATIONS ADAPTERS

Each communications channel requires an adapter which provides the logic to interface with a

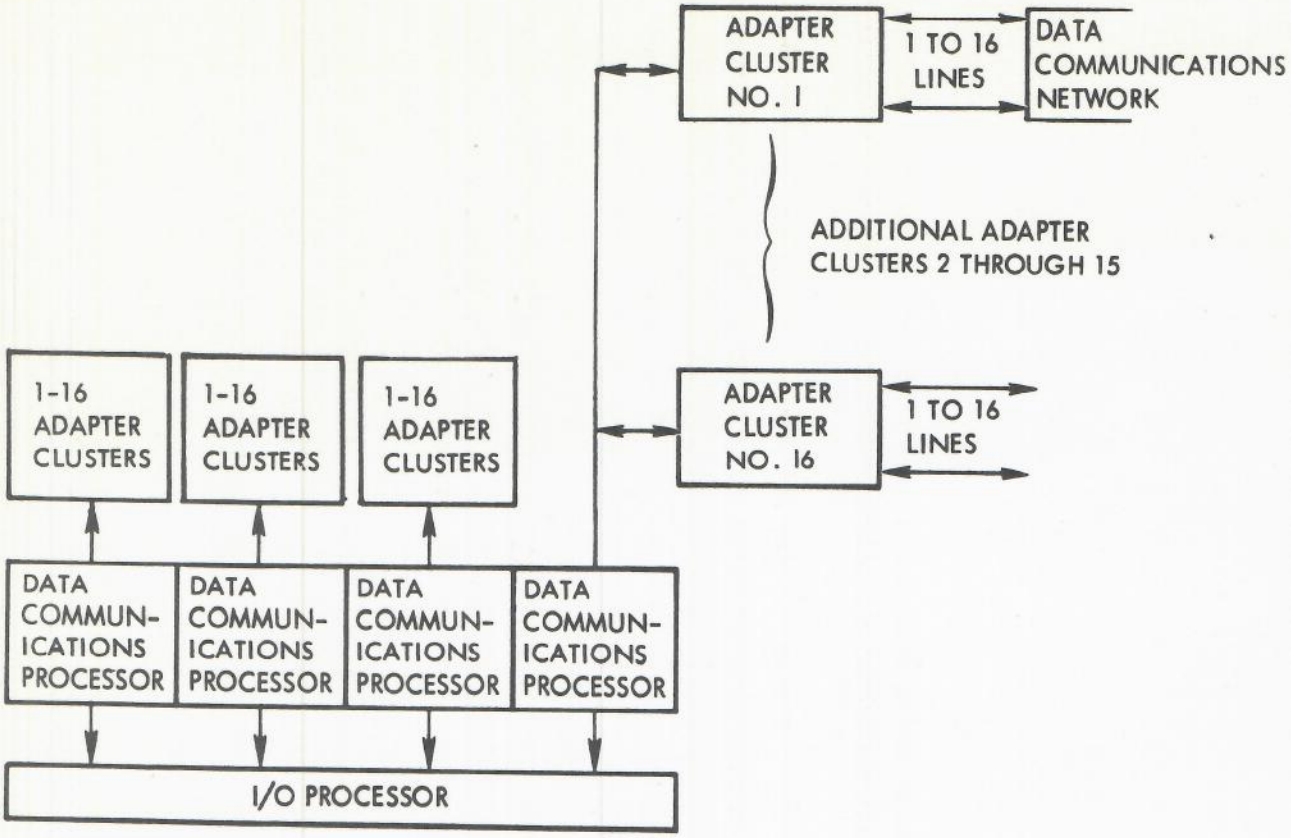


Figure 1-8. Organization of Data Communications Processor Remote Lines

Data Set or to connect directly to a communications line. The following adapters are available:

1. B 6650-1 with the following characteristics:
   a. Direct or modem connect.
   b. Asynchronous.
   c. Up to 600 BPS.
   d. Two wire or 100 series modem.
   e. Serial-by-bit transmission.
   f. Half-Duplex mode.

2. B 6650-2 with the following characteristics:
   a. Direct or modem connect.
   b. Asynchronous.
   c. Up to 1800 BPS.
   d. Two wire or 202 series type Data Set.
   e. Serial-by-bit transmission.
   f. Half-Duplex mode.

3. B 6650-3 with the following characteristics:
   a. Modem connect.
   b. Synchronous.
   c. Up to 2400 BPS.
   d. 201 series type Data Set.
   e. Serial-by-bit transmission.
   f. Half-Duplex mode.

4. B 6650-4: same as B 6650-3 except that it can handle up to 4800 BPS.

5. B 6650-5: same as B 6650-3 except that it can handle up to 9600 BPS.

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

7. B 6650-7: Audio Response.

8. B 6650-8: Automatic Dial Out.

Real-Time Adapter

Real-time adapters may be attached to an input/output processor. Real-time devices require custom engineering for interface with the real-time adapters and the software.

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

SECTION 2
DATA REPRESENTATION

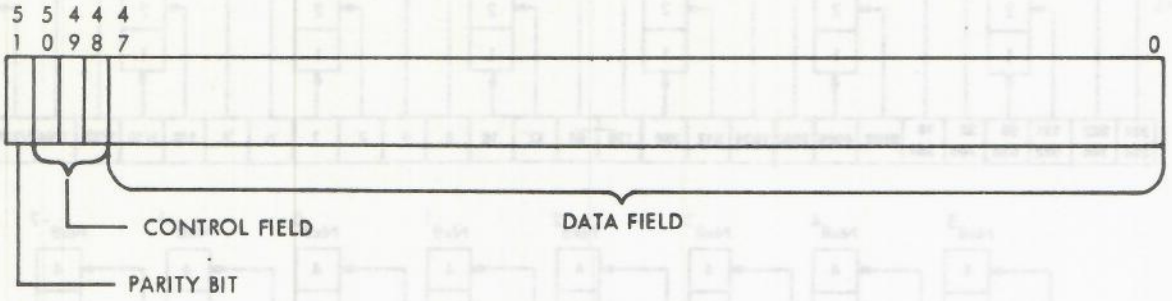


Figure 2-1. Basic Word Structure

GENERAL

Several data representations are used in the B 6700 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 eight-bit, six-bit or four-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 checks for correct information transfer between the processor and main memory or from the scratch pad memory to main memory.

INTERNAL CHARACTER CODES

Extended Binary Coded Decimal Interchange Code (EBCDIC) is the primary internal character code of the B 6700. EBCDIC is an eight-bit alphanumeric code containing four zone and four 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. Numeric EBCDIC and BCL codes may be packed into four-bit digits by internal commands which delete the zones and compress the numeric portion of the characters.

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 first 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 first 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 comprise 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).

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).

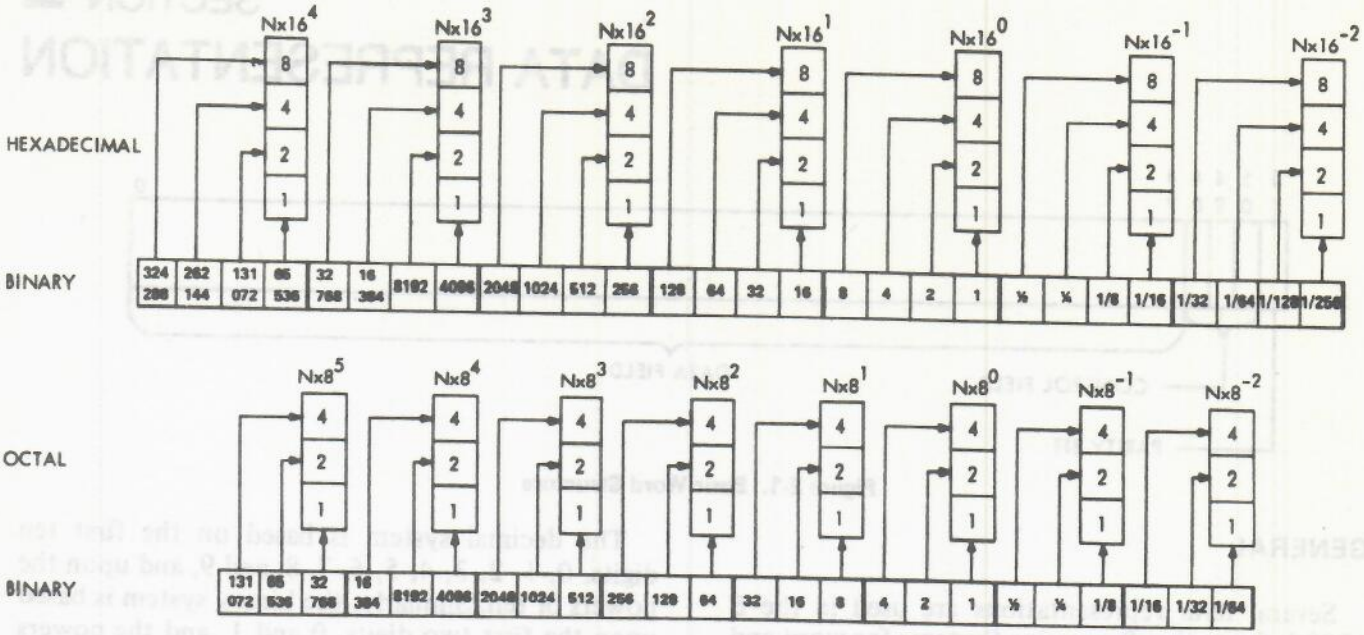


Figure 2-3. Binary-to-Hexadecimal and Octal Conversion

$1765_8 \equiv 1 \times 8^3 + 7 \times 8^2 + 6 \times 8^1 + 5 \times 8^0 =$
$1 \times 512 + 7 \times 64 + 6 \times 8 + 5 \times 1 =$
$512 + 448 + 48 + 5 = 1013_{10}$

$1013_{10} \equiv 1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 3 \times 1^0 =$
$1 \times 1000 + 0 \times 100 + 1 \times 10 + 3 \times 1 =$
$1000 + 0 + 10 + 3 = 1013_{10}$

$3F5_{16} \equiv 0 \times 16^3 + 3 \times 16^2 + F \times 16^1 + 5 \times 16^0 =$
$0 \times 4096 + 3 \times 256 + F \times 16 + 5 \times 1 =$
$0 + 768 + 240 + 5 = 1013_{10}$

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

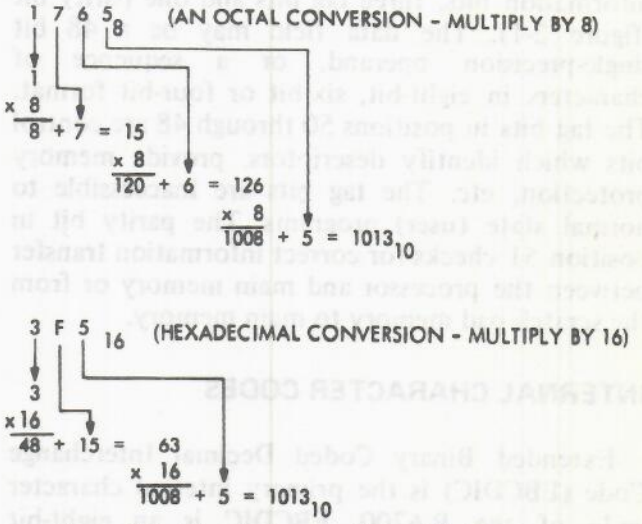


Figure 2-5. Hexadecimal and Octal To Decimal

NUMBER CONVERSION

Coded To Decimal Conversion

The conversion to base 10 of the integral value of a number whose base is other than 10 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 this system is used (figure 2-5).

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).

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 values to obtain the decimal equivalent.

DECIMAL-TO-HEXADECIMAL:

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

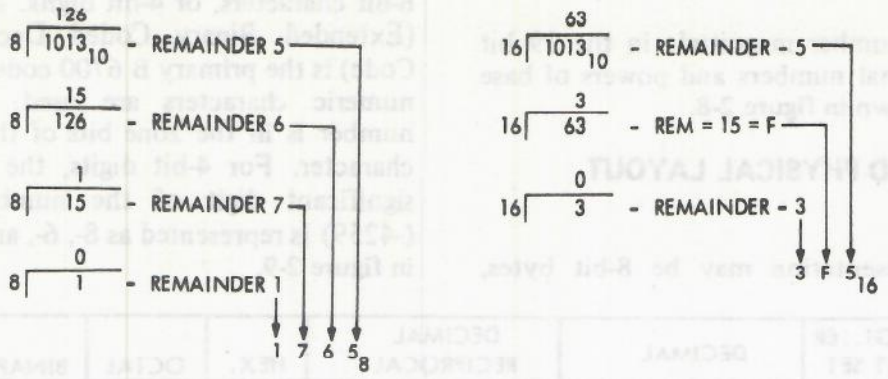


Figure 2-6. Decimal 1013_{10} To Hexadecimal and Octal

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

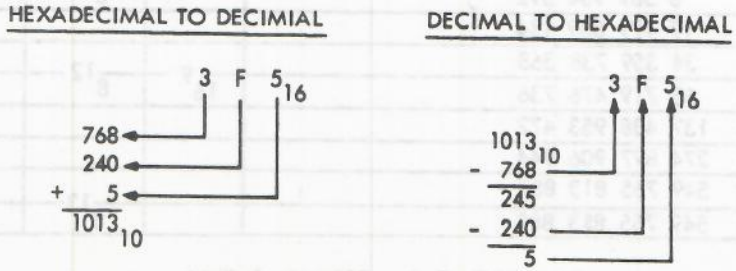


Figure 2-7. HEX and DEC Table Conversion

ORDER OF MAGNITUDE

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

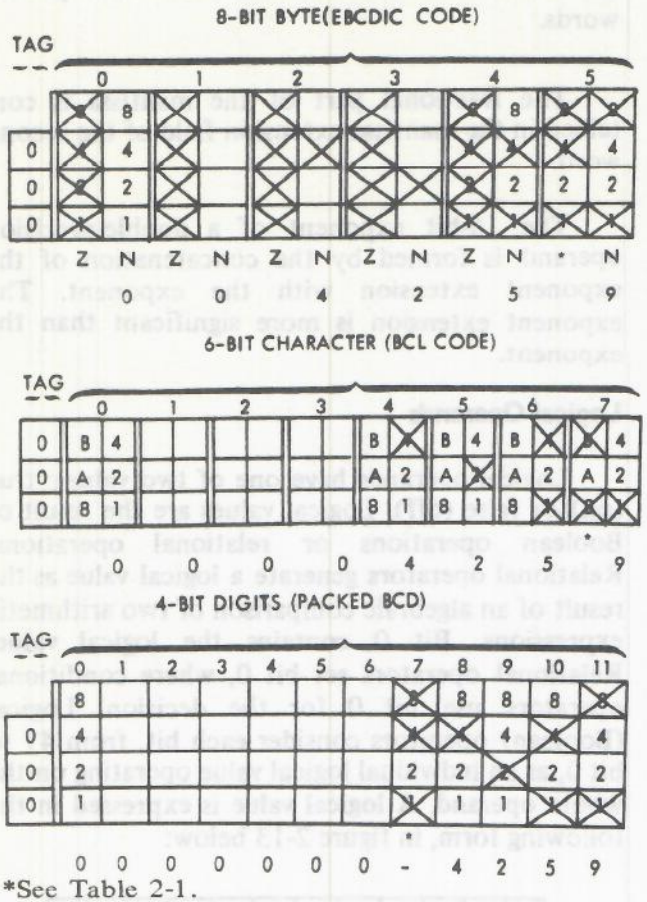
DATA TYPES AND PHYSICAL LAYOUT

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 6700 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-, and 4-bit characters in figure 2-9.

REGISTER BIT SET	DECIMAL	DECIMAL RECIPROCAL	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	549 755 813 887				
	549 755 813 888			8^{13}	2^{39}

Figure 2-8. Order of Magnitude Table



*See Table 2-1.

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

Operands

Operands may be used to represent either numeric or logical information in the B 6700 System. An operand may be single or double precision. When the tag bits of a memory word (bits 50, 49, 48) are 0 (000), they denote a single-precision operand. When the tag bits are 2 (010), i.e., bit 49 set, they denote 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 power, the single-precision operand is also shown for reference purposes, in octal (figure 2-11). Figure 2-12 illustrates the double-precision operand in hexadecimal register format.

An integer is a single-precision operand with an exponent of 0. The maximum value of an integer is $+7777777777777_8$, 549755813887_{10} or $7FFFFFFFF_{16}$.

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

1. In octal format:
   0000000000000014 (integer)
   1010000000000140
   1020000000001400 (floating point, or real)
   1131400000000000
2. In hexadecimal format:
   00000000000C (integer)
   208000000060
   210000000300 (floating point)
   259800000000

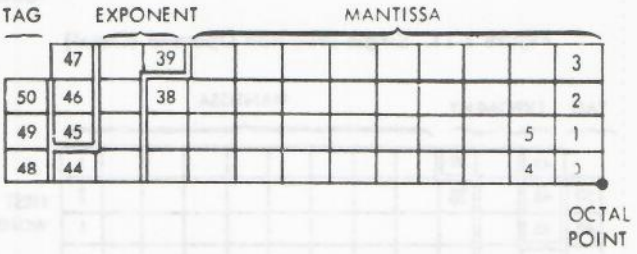


Figure 2-10. Single-Precision Operand (Hexadecimal)

[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 arithmetic, comparison and integer operations are being performed. The range of the exponent is from +63 to -63 for single-precision operands.

Table 2-1. Negative Sign Configurations

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	

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; double-precision is $(8^{13} - 1) \times 8^{+32,767}$ to $1 \times 8^{-32,755}$ and zero.

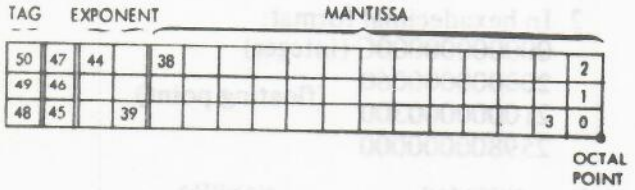


Figure 2-11. Single-Precision Operand (Octal)

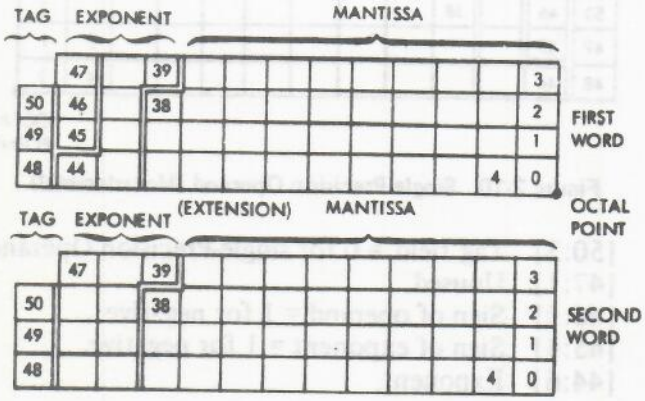


Figure 2-12. Double-Precision Operand

[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 second 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 0 contains the logical value. Relational operators set bit 0, where conditional operators use bit 0 for the decision. Logical (Boolean) operators consider each bit, from 47 to bit 0, as an individual logical value operating on the whole operand. A logical value is expressed in the following form, in figure 2-13 below:

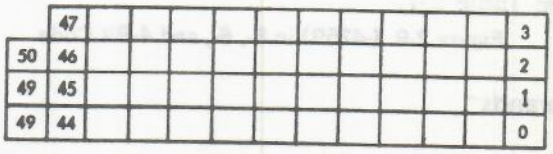


Figure 2-13. Logical Operand

[50:3] = 0 tag = Single-precision operand
[0:1] = 1 true, 0 false

OPERATORS

The operators used in the B 6700 systems are divided into three major categories: primary, variant and edit. Details regarding the formats and functions 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; the stack provides storage for the 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.

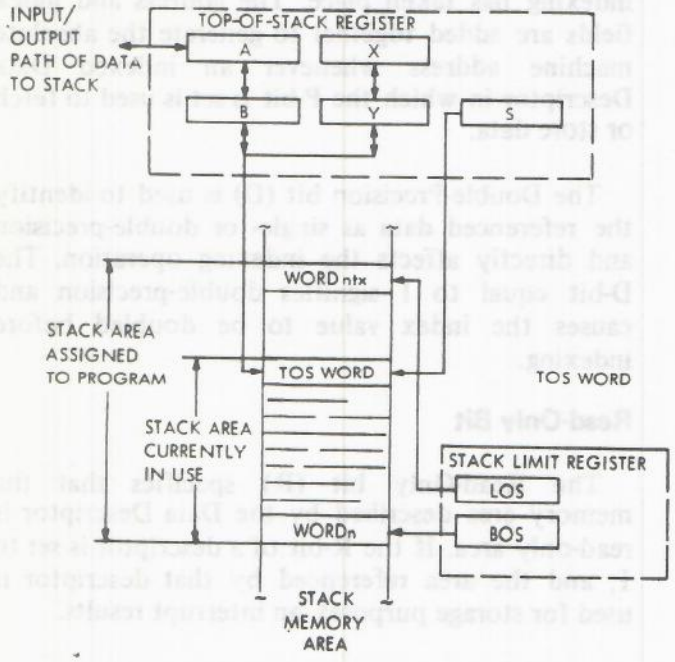


Figure 3-1. Top-of-Stack and Stack Bounds Registers

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 extrated. 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 1 before a word is placed into the stack and is decremented by 1 after a 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 bounded, for memory protection, by two registers: the Base-of-Stack register (BOSR) and the Limit-of-Stack register (LOSR). The contents of BOSR define the base of the stack, and the contents of LOSR define 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, if any, of the top-of-stack registers, are pushed into the stack. Words are brought out of the stack into the top-of-stack registers. These words are used by operators which require the presence of data in the top-of-stack registers. These operators, however, do no 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 causes two memory words to be loaded 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 the X register. 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 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 6700 processor provides three methods for addressing data or program code:

1. Data Descriptor (DD)/Segment Descriptor (SD).
2. Indirect Reference Word (IRW).
3. Stuffed Indirect Reference Word (SIRW).

The Data Descriptor (DD) and Segment Descriptor (SD) provide for the addressing of 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, the addressing being 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 to 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 in each of these descriptors is 20 bits in length; this field contains the absolute address of an array in either system main memory or in the backup disk file, as indicated by setting of 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 by means of a descriptor in which the P-bit is equal to 0; i.e., the data is located in a disk file, rather than in main memory. The Master Control Program (MCP) recognizes the Presence Bit Interrupt and transfers data from disk file storage to main memory. After the data transfer to main memory is completed, the MCP marks the descriptor present by setting the P-bit to 1, 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 0, 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 is 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 0.

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 1 to indicate that indexing has taken place. The address and index fields are added together to generate the absolute machine address whenever an indexed Data Descriptor in which the P-bit is set 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 directly affects the indexing operation. The D-bit equal to 1 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. If the R-bit of a descriptor is set to 1, and the area referenced by that descriptor is used for storage purposes, an interrupt results.

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 (i.e., the presence bit is off) to the one master descriptor. The copy bit mechanism is invoked when a copy is made in the stack. If it is a copy of the original, absent descriptor, the processor sets the copy bit to 1 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.

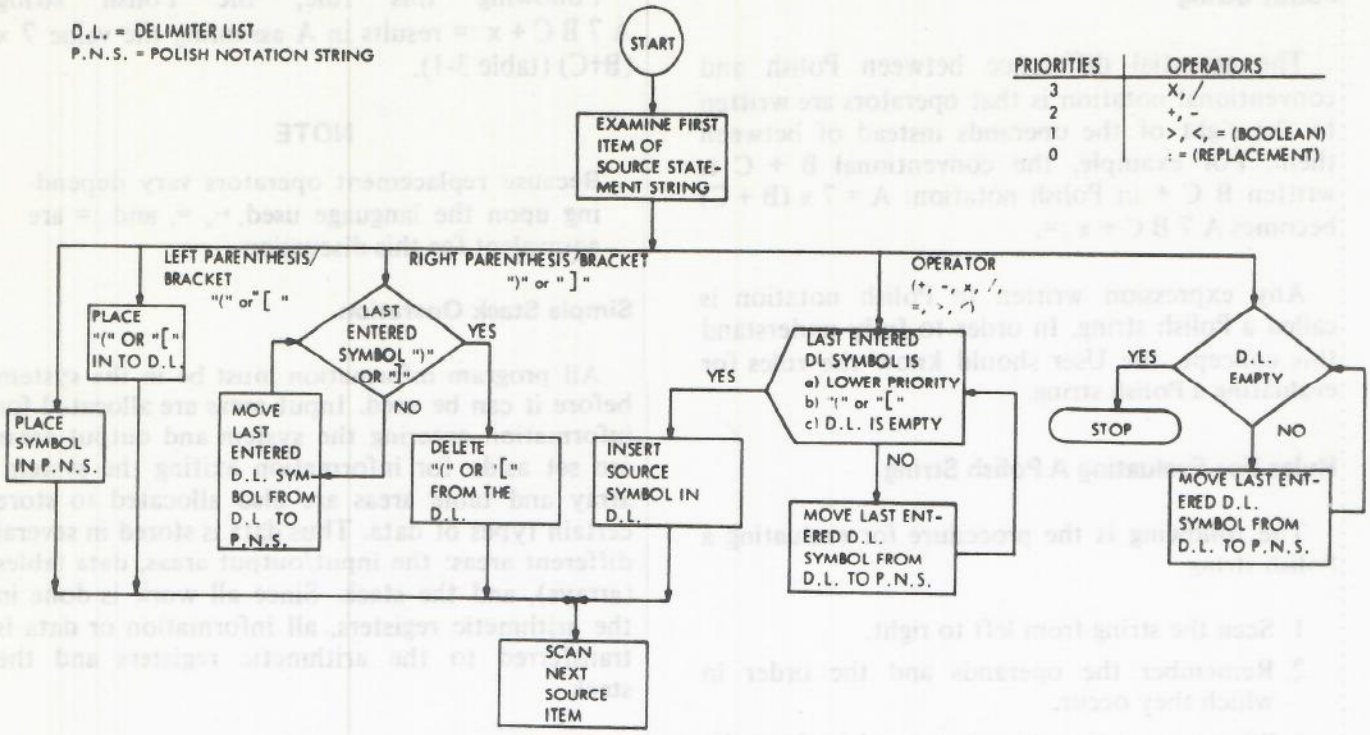


Figure 3-2. Polish Notation Flow Chart

POLISH NOTATION

General

Polish notation is an arithmetical or logical notational system using only operands and operators arranged in sequence or strings, thus eliminating the necessity for defining the boundaries of any terms. Figure 3-2 presents a flow chart for conversion to Polish notation.

Simplified Rules For Generation Of Polish String

The source of expression is as follows:

Name	Action
Variable or constant	Place variable or constant in string being built and examine next symbol.
Operator-separator "(" or "["	Place in delimiter list and examine next symbol.
Arithmetic or Boolean operator and last-entered delimiter list symbol were as follows:	Place operator in the delimiter list and examine next source symbol.
1. An operator of lower priority.	Remove the operator from the delimiter list and place it in the string being built. Then compare the next symbol in the delimiter list against the source expression symbol.
2. A left bracket "[" or parenthesis "(".	Pull from delimiter list until corresponding left bracket or parenthesis.
3. A separator.	Move last-entered delimiter list symbols to Polish notation string until empty.
4. Nothing (delimiter list empty).	
A right bracket "]" or parenthesis ")".	
End of expression.	

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 x (B + C) becomes A 7 B C + x :=.

Any expression written in Polish notation is called a Polish string. In order to fully understand this concept, the User should know the rules for evaluating a Polish string.

Rules For Evaluating A Polish String

The following is the procedure for evaluating a Polish string:

1. Scan the string from left to right.
2. Remember the operands and the order in which they occur.
3. When an operator is encountered perform the following:
   a. Record the last two operands encountered.
   b. Execute the required operation.
   c. Disregard the two operands.
   d. Consider the result of (b) above as a single operand, the first of the next pair to be operated upon.

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

NOTE

Because replacement operators vary depending upon the language used, ←, =, and := are equivalent for this discussion.

Simple Stack Operation

All program information must be in the system before it can be used. Input areas are allocated for information entering the system and output areas are set aside for information exiting the system; array and table areas are also 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.

Table 3-1 Evaluation of Polish String A 7 B C + x :=

Step	Symbol Being Examined	Symbol Type	Operands Being Remembered and Their Order of Occurrence (1 or 2) Before Operation	Operation Taking Place	Results of Operation
1.	B	Operand			
2.	C	Operand	1 B		
3.	+	Add Operator	2 C 1 B	B + C	(B + C)
4.	7	Operand	1(B + C)		
5.	x	Multiply Operator	2 7 1 x (B + C)	7 x (B + C)	7 x (B + C)
6.	A	Name	1 7 x (B + C)		
7.	:=	Replace Operator	2 A 1 7 x (B + C)	A :=7x(B + C)	A=7x(B + C)

When Z:=Y + 2x(W+V) is translated to Polish notation, the result is ZY2WV+ x +:=. 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, 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 by 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.

Since Y is to be added to a quantity, Y is brought into the top of 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, for simplicity, single-precision operands 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; when executed, it places the product "2x(W+V)" 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.

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

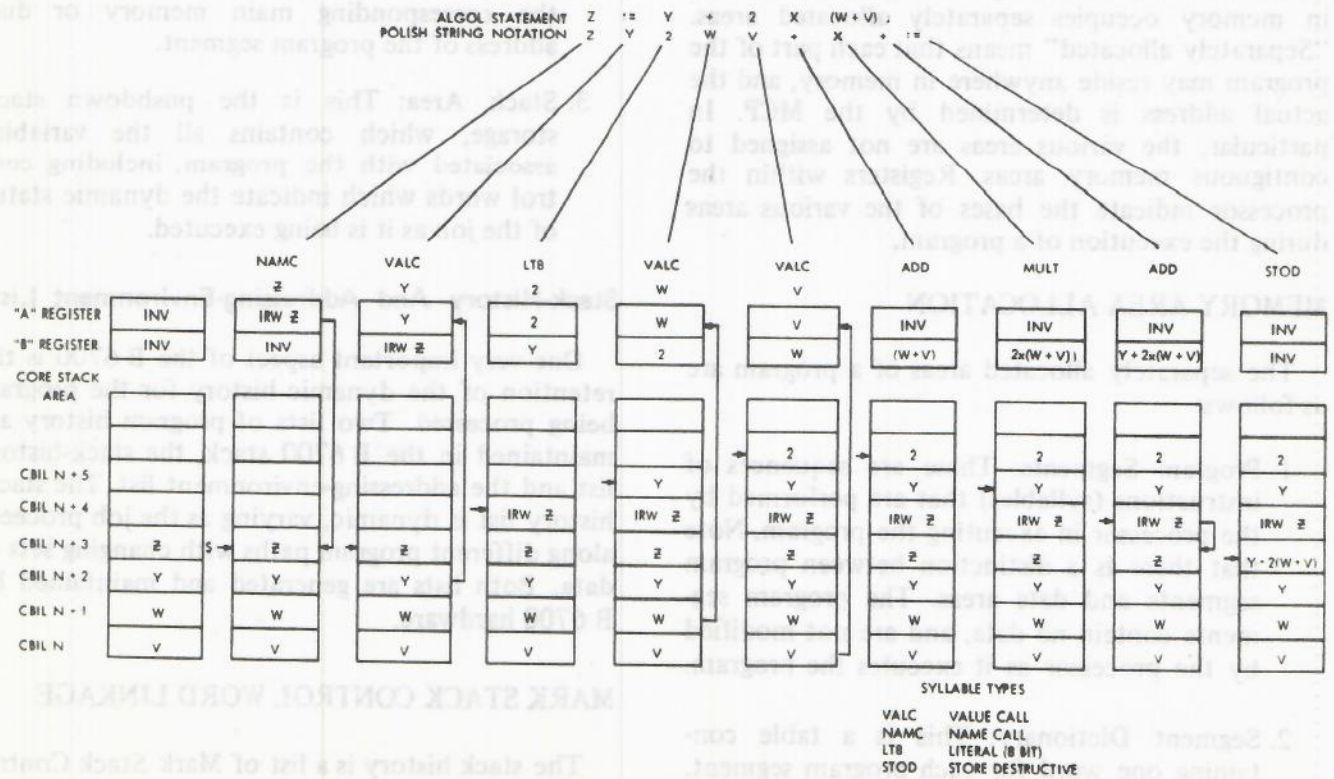


Figure 3-3. Stack Operation

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	Z	Name call for Z.	Build an indirect reference word that contains the address of Z and place it in the top of the stack.
2	Y	Value call for Y.	Place the value of Y in the top of the stack.
3	2	Literal 2.	Place a 2 in the top of the stack.
4	W	Value call for W.	Place the value of W in the top of the stack.
5	V	Value call for V.	Place the value of V in the top of the stack.
6	+	Operator add.	Add the two top words in the stack and place the result in B register as the top of the stack.
7	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.
8	+	Operator add.	Add the two top words in the stack and leave the result in the B register as the top of the stack.
9	:=	Operator store destructive.	Store an item into memory. The address in which to store is indicated by an indirect reference word or a data descriptor. The address can be above or below the item stored.

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 as follows:

1. Program Segments: These are 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.

2. 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.

3. 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.

Stack-History And Addressing-Environment Lists

One very important aspect of the B 6700 is the retention of the dynamic history for the program being processed. Two lists of program history are maintained in the B 6700 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 6700 hardware.

MARK STACK CONTROL WORD LINKAGE

The stack history is a list of Mark Stack Control Words (MSCW), linked together by their Displace-

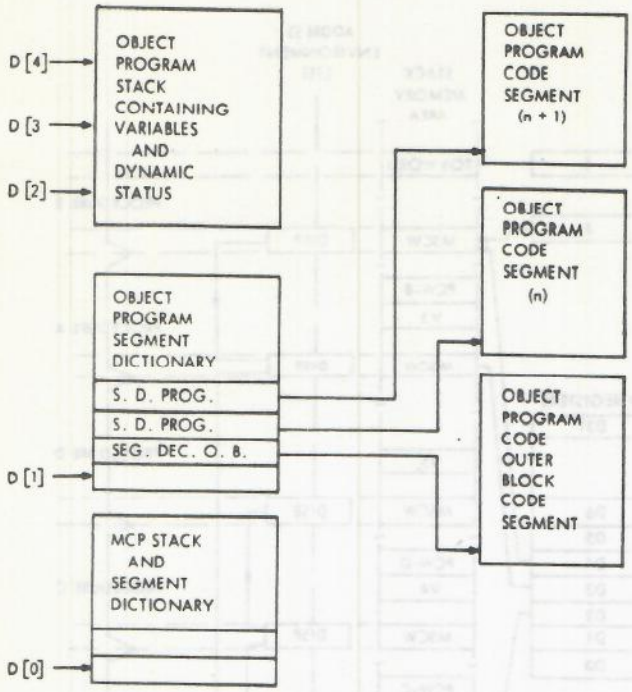


Figure 3-4. Object Program in Memory

ment Fields (DF) (figure 3-5). An 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 the procedure is executed its parameters and local variables are referenced by addressing relative to the MSCW.

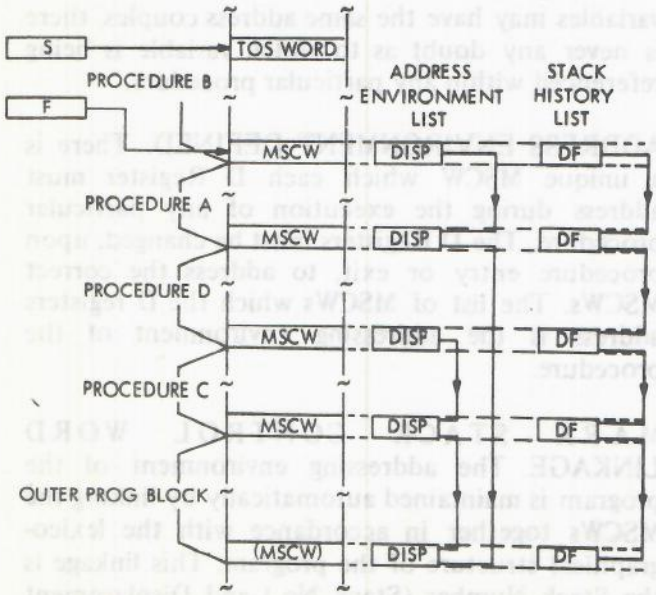


Figure 3-5. Stack History and Addressing Environment List

STACK DELETION

Each MSCW is linked to the prior MSCW through the contents of its DF field in order 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 topmost MSCW, addressed by another register (F), is deleted from the stack-history list by changing F to address the prior MSCW, placing this MSCW at the head of the stack history.

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

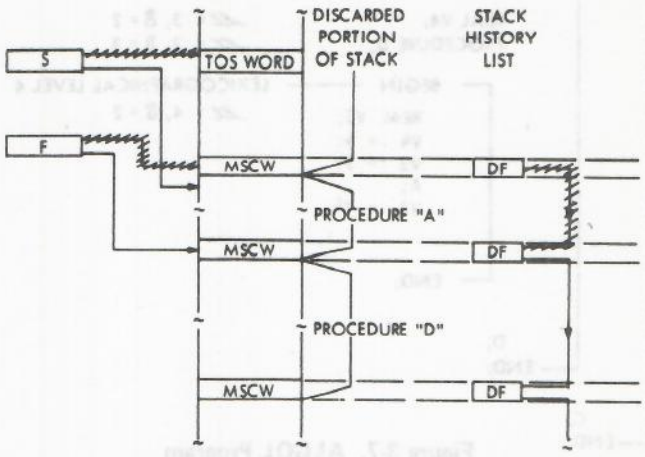


Figure 3-6. Stack Cut-Back Operation on Procedure Exit

RELATIVE-ADDRESSING

Analyzing the structure of an ALGOL program results in a better understanding of the relative-addressing procedures used in the B 6700 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 3-7). At compile time, the lexicographical ordering is used to form address couples. An address couple consists of two items:

1. The addressing level (ℓℓ) of the variable,
2. An index value (§) 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.

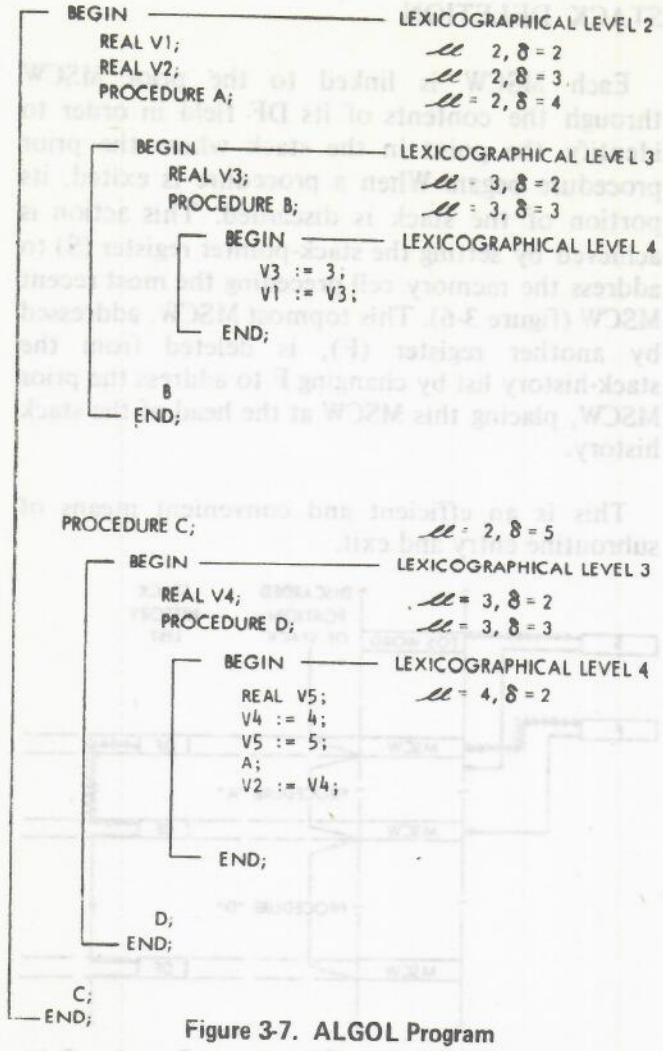


Figure 3-7. ALGOL Program With Lexicographical Structure Indicated

BASE OF ADDRESSING–LEVEL SEGMENT. The B 6700 processor contains an array of D Registers (DO through D31). These registers address the base of each addressing-level segment (figure 3-8). 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

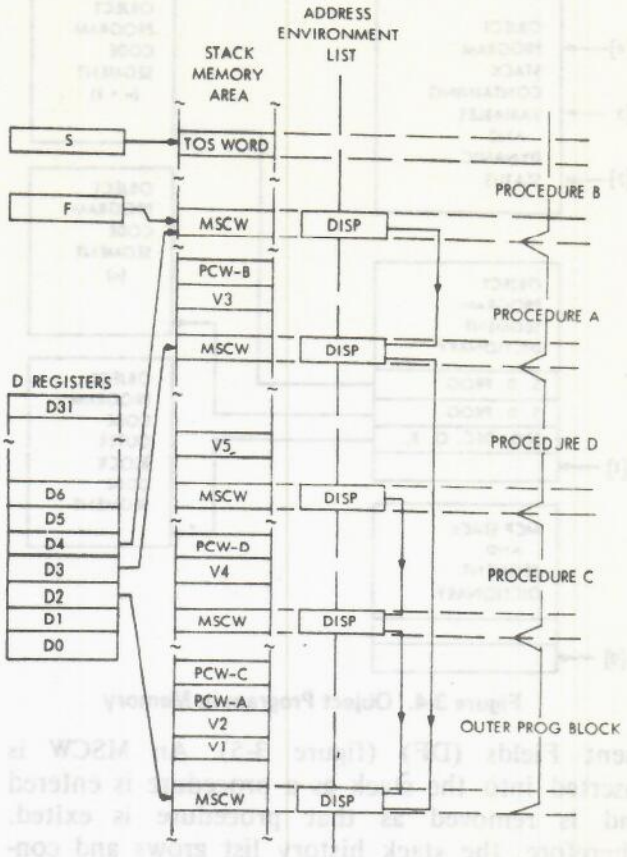


Figure 3-8. D Registers Indicating Current Addressing Environment

unique. This is true because of the ALGOL scope-of-definition rules, which imply that if there is no procedure which can address both of any two quantities, then these two quantities may unambiguously have the same address couple. 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 MSCW 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 (figures 3-8 and 3-9). This list is used to update the D Registers whenever a procedure entry or exit occurs.

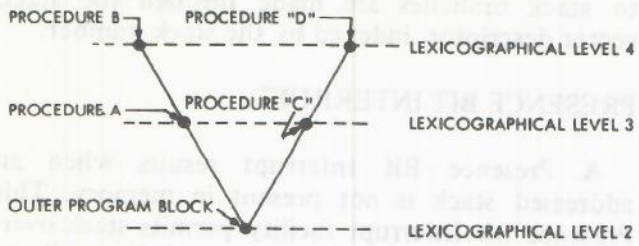


Figure 3-9.
Addressing Environment Tree of ALGOL Program

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 conveniently respond to, and return from, interrupts. 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 from this procedure, the D Registers are updated to display variables of the former procedure.

Multiple Stacks And Re-Entrant Code

The B 6700 stack mechanism provides a facility for handling several active stacks, which are organized in a tree structure. The trunk of this tree structure is a stack containing MCP global quantities.

LEVEL DEFINITION

A program is a set of executable instructions, and a job is a 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 pointing to 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 segment descriptor 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 emerges 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. This results 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.

JOB–SPLITTING

The B 6700 stack mechanism also provides the facility for a single job to split itself into two independent jobs. A 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 can be used to make full use of a multiprocessor configuration, or to reduce elapsed time by multiprograming 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.

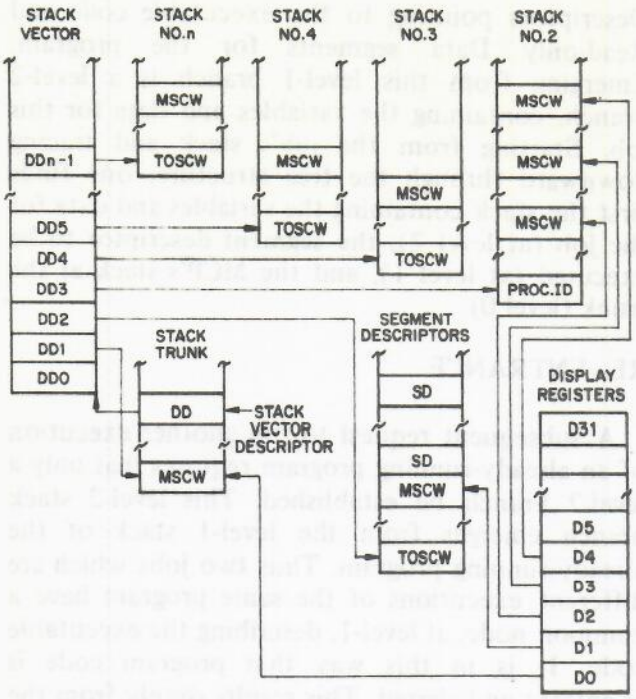


Figure 3-10. Multiple Linked Stacks

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 array size of the stack vector and its location in memory is described by the stack vector descriptor, located in a reserved position of the trunk of the stack (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 a stack is subsequently referenced, a Presence Bit Interrupt is generated to cause the MCP to recall the non-present 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 input/output processor. Panels C, D, and E contain indicators and switches for the entire system.

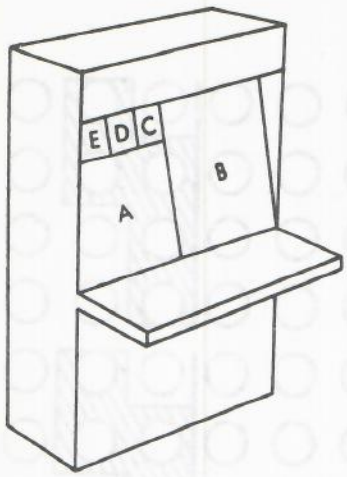


Figure 4-1. Processor Display Panels

Panel A (figure 4-2).

NOTE

Although Panels A and B are shown separately in this manual, they are actually overlayed on one template. The lines and mnemonics are printed in two colors; one to identify processor registers and flip flops, and the other to identify those used in the input/output processor.

P REGISTER

The P register is a 51-bit instruction register.

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.

A REGISTER

The A register is a 51-bit information register that holds one complete word. This register is the top-of-stack when the A Register Occupied flip flop (AROF) indicates that it contains a valid word. It is used in many ways: arithmetic, Boolean, character string, addressing, indexing, comparing, 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 B Register Occupied flip flop (BROF) is on.

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.

Panel B (figure 4-3).

Panel B indicators are shared by the processor and input/output processor flip flops.

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

Panel B is divided into related family and control groups. The Maintenance Diagnostic Logic (MDL) Processor is common to both display modes, i.e., processor or input/output processor flip flops.

ROW A

This row contains the flip flops for addressing the integrated circuit (IC) memories in the Memory Controller.

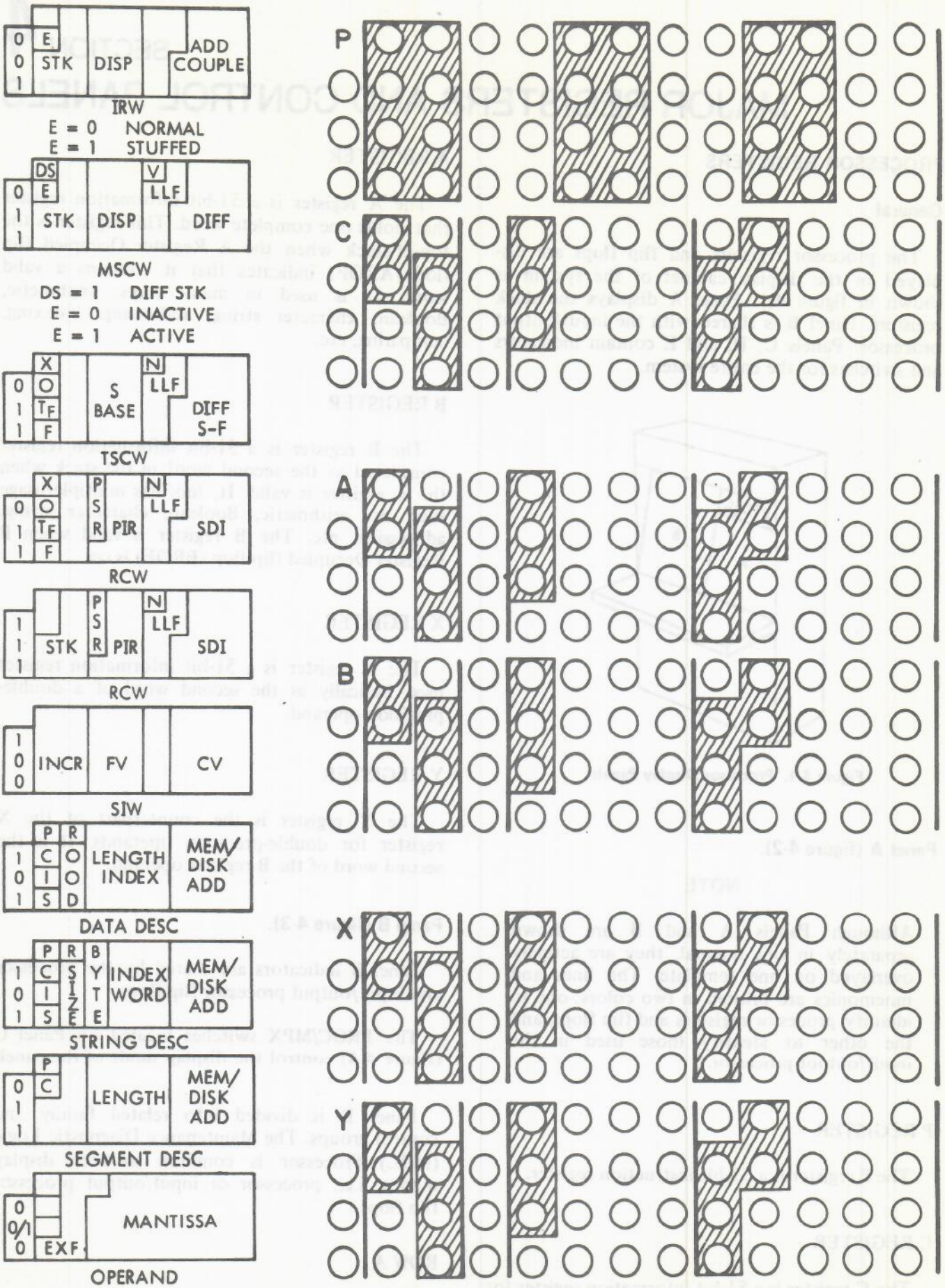


Figure 4-2. Processor Register Panel A

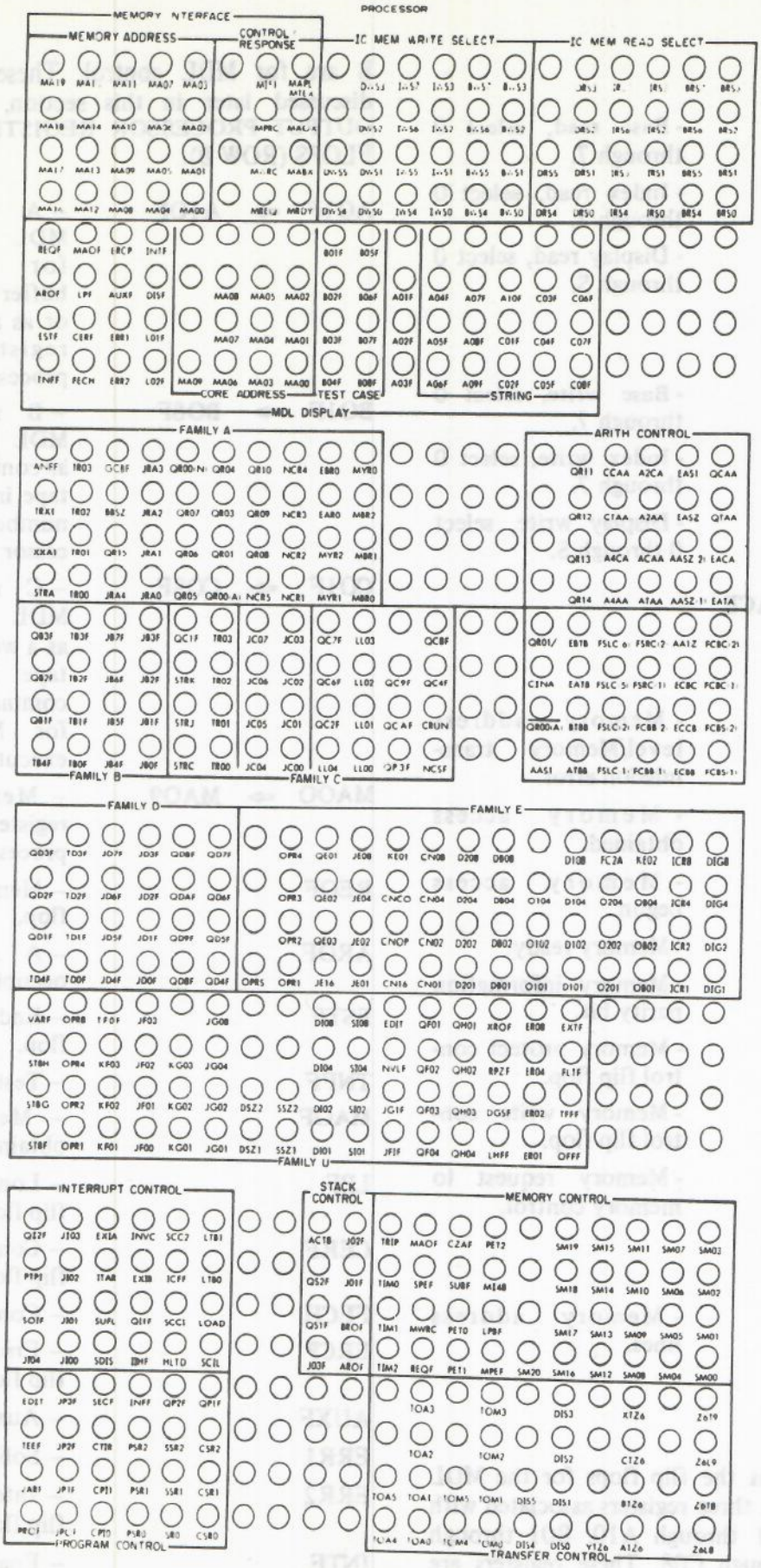


Figure 4-3. Processor Display Panel B

IC Mem Read Select

BRSO ⇒ BRS7 - Base read, select 0 through 7.

IRSO ⇒ IRS7 - Index read, select 0 through 7.

DRSO ⇒ DRSS - Display read, select 0 through S.

IC Mem Write Select

BWSO ⇒ BWS7 - Base write, select 0 through 7.

IWSO ⇒ IWS7 - Index write, select 0 through 7.

DWSO ⇒ DWSS - Display write, select 0 through S.

MEMORY INTERFACE

Control/Response

MAPL/MTEX - Memory address level/Memory transmission error.

MAOX - Memory access obtained.

MABX - Memory access begun.

MRDY - Memory ready.

MI51 - Memory information parity bit.

MPRC - Memory protect control flip flop.

MWRC - Memory write control flip flop.

MREQ - Memory request to memory control.

Memory Address

MAOO ⇒ MA19 - Memory address lines.

ROW B

This row contains the flip flops for the MDL processor. There are three registers associated with this processor: A01 through A10, B01 through B08, and C01 through C08. These registers are used for system testing. The other flip flops in row B are for MDL control. These flip flops are discussed later in this section, under INPUT/OUTPUT PROCESSOR REGISTERS AND FLIP FLOPS (ROW B).

A01F ⇒ A10F – A register in the MDL processor; used for character/word buffer for tape input or as a command/data register for MDL processor execution.

B01F ⇒ B08F – B register in the MDL processor; used as control flip flops for tape input, or test case number for MDL processor execution.

C01F ⇒ C08F – C register in the MDL processor; used as a word buffer to the tape input or as a command/data register for MDL processor execution.

MAOO ⇒ MA09 – Memory address register for the MDL processor.

REQF – Memory request flip flop.

AROF – A and C register occupied flip flop.

ESTF – End of string flip flop.

TNFF – Test not flip flop.

MAOF – Memory access obtained flip flop.

LPF – Longitudinal parity flip flop.

CERF – Control parity error flip flop.

FECH – Control flip flop.

ERCP – Error complement flip flop.

AUXF – Auxiliary flip flop.

ERR1 – Solid error flip flop.

ERR2 – Intermittent error flip flop.

INTF – Enable display cycle flip flop.

DISF	– Discrepancy flip flop.
LO1F	– Sequence counter flip flop.
LO2F	– Sequence counter flip flop.

ROW C

This row contains Family A flip flops and one-half of the Arithmetic Controller flip flops.

Family A

TROO ⇒ TRO3	– Contains the OP code
JRAO ⇒ JRA4	– Sequence count used in the OP code flow
QRO1	– Pre-Carry into adder
QROO (A)	– Carry-in control for multiply
QRO2	– High-speed clock phase control
QRO3 ⇒ QRO7	– Logic control
QRO8 ⇒ QR1O	– Temporary storage
QR11 ⇒ QR14	– Q Counter
QR15	– Interrupt flip flop
QROO/QRO1/	– Carry-in reset control
QROO(N)	– Multiply carry in flip flop
NCR1 ⇒ NCR5	– N counter
MBRO ⇒ MBR2	– B-register mantissa field extension
MYRO ⇒ MYR2	– Y-register mantissa field extension
EARO	– Extension of A-register exponent field
EBRO	– Extension of B-register exponent field
STRA	– Family A strobe flip flop (turned on by the Program controller through the Z10 bus)
XXA1	– Function parallels STRA
TRX1	– Function parallels TRO1

Arithmetic Control

All other flip flops in the Arithmetic Controller are used for logic control. They are as follows:

BBSZ	FCBB (2)		
AAS1	FSRC (1)		
EBTT	FSRC (2)		
EATB	AA1Z		
BTBB	ECBC		
ATBB	ECCB		
FSLC (1)	ECBB		
FSLC (2)	FCBC (1)		
FSLC (5)	FCBC (2)	FCBC (1)	QROO1/
FSLC (6)	FCBS (1)	$\overline{\text{QROO/A}}$	
FCBB (1)	FCBS (2)	CINA	

ROW D

This row contains the Family B and C flip flops.

Family B

TBOF ⇒ TB4F	– Contains the OP code
JBOF ⇒ JB7F	– Sequence count used in the OP code flow
QB1F ⇒ QB3F	– Logic control

Family C

TROO ⇒ TRO3	– Contains the OP code
JCOO ⇒ JCO7	– Sequence count used in the OP code flow
LLOO ⇒ LLO4	– Lexicographical level flip flops for the Program flow
QP3F	– Extension of QP1F and QP2F
STRC	– Strobe family C (subroutine)
STRJ	– Strobe family J (Value Call)
STRK	– Strobe family K (Name Call)
QC1F ⇒ QCBF	– Logic control
ANFF	– Logic control
NCSF	– Normal/control State flip flop: When

this flip flop is reset, "off" signifies normal state; "on" signifies control state

The Control State flip flop provides an extension to the operator set to include additional operators; it also 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
QD1F ⇒ QD9F, QDAF, QDBF	– Logic control flip flops

Family E

OPRS	– Family E strobe
OPR1 ⇒ OPR4	– Contain the OP code
JEO1 ⇒ JE16	– Sequence count used in OP code flow
DIG1 ⇒ DIG8	– Length field
ICR1 ⇒ ICR8	– Input convert
OBO1 ⇒ OBO4	– Octal buffer bit
O101 ⇒ OBO4	– Octal 1 bit
O2O1 ⇒ O2O4	– Octal 2 bit
DBO1 ⇒ DBO8	– Digit buffer bit
D1O1 ⇒ D1O8	– Digit 1 bit
D2O1 ⇒ D2O8	– Digit 2 bit
CNO1 ⇒ CN16	– Counter
QEO1 ⇒ QEO3	– Logical control
FC2A	– Transfer CC to AA
KEO1, KEO2	– Logical control
CNCO	– Counter control
CNOP	– Counter

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
KFO1 ⇒ KFO3	– Extension of sequence count for Family F
JFOO ⇒ JFO3	– Sequence count used in family F OP code flow
KGO1 ⇒ KGO3	– Extension of sequence count for Family G or H
JGO1 ⇒ JGO8	– 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
DXZ2	– Destination size more significant bit
SSZ1	– Source size less significant bit
SSZ2	– Source size more significant bit
DIO1 ⇒ DIO8	– Destination character pointer
SIO1 ⇒ SIO8	– Source character pointer
EDIT	– Edit mode for string OPS
NVLF	– Invalid OP Code
JGIF	– JG interrupt state
JFIF	– JF interrupt state
QFO1	– Invalid OP interrupt
QFO2	– Presence bit interrupt
QFO3	– Memory protect interrupt
QFO4	– Segmented array interrupt
QHO1 ⇒ QHO4	– Logical control
XROF	– Register occupied
RPZF	– Logical control

DGSF	– Logical control
LHFF	– Logical control
ERO1 ⇒ ERO8	– E-Register flip flops (Used for memory cycle requests during string OP code flow.)
EXTF	– External sign
FLTF	– Float
TFFF	– True false
OFFF	– Overflow
STBF	– Strobe for family F
STBG	– Strobe for family G
STBH	– Strobe for family H

ROW G

This row contains the flip flops used for Interrupt Control, Stack Control and Memory Control.

Interrupt Control

JIOO ⇒ JIO4	– Sequence count for controller flow
SOIF	– 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
LTBO, LTB1	– Load timer Bit
INVC	– Invalid code (Tag in P ≠ 3)
IIHF	– Set programmatically to inhibit external interrupt handling by the processor.

Stack Control

JO1F ⇒ JO3F	– 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 Control

SMOO ⇒ SM2O	– Address adder output flip flops. These are for display only. (No manual set or reset controls)
TRIP	– Trip control invalid address
TIMO ⇒ 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
PETO ⇒ PET2	– Information parity test control register
MI48	– Memory protect bit
LPBF	– Line parity bit from memory
MPEF	– Memory parity error

ROW H

This row contains the flip flops used for Program Control and Transfer Control.

Program Control

JPOF ⇒ JP3F – Sequence count for controller flow

PROF – The P register contains a valid word

VARF – Variant mode flip flop (Used to enter the variant mode; see Section 8.)

TEEF – Table enter edit

EDIT – Edit mode

CPIO, CPI1 – A two-bit counter used to back up the PIR (program index register)

CTIR – A one-bit counter used to back up the TIR (table index register)

SECF – SECL (syllable execute complete level) saved

INFF – Inhibit fetch flip flop (used to inhibit bringing a new program word to the P register)

PSRO ⇒ PSR2 – Program syllable register O ⇒ 5 pointer (points to next syllable to be executed from the P register)

QPIF, QP2F – Logic control

SSRO SSR2 – Syllable saved register O (Used to save the current position of PSR when in table mode.)

CSRO ⇒ CSR2 – Command Syllable register O ⇒ 5. (Used to save the current position of PSR.)

Transfer Controller

TOAO ⇒ TOA5 – Top-of-Aperture Register (Used to select top bit of 48-bit field to be transferred through the steering and mask network.)

TOMO ⇒ TOM5 – Top-of-mask register (used to select top bit of 48-bit field to be inhibited through the steering and mask network)

DISO ⇒ DIS5 – Displacement register (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.)

Z6T8 – 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 6700 system. Units which cannot be controlled from this panel have their own local maintenance controls.

Power Controls

The power supplied to the B 6700 system is controlled by sequence control circuits located in the MDL display cabinet. There are two sequence control circuits (sequence control circuits A and B) in one MDL display cabinet; a maximum of two MDL display cabinets can be used per system. There are two sets of power control switches located on the upper-right corner of panel C on the MDL display cabinet (see figure 4-4A). One set of these switches controls sequence control circuit A, and the other controls sequence control circuit B.

In addition, there is also a set of three toggle switches labeled CONNECT-DISCONNECT A, B,

or C. These switches can connect the selected sequence control circuit to one common control (see figure 4-4B). If these three switches are in position DISCONNECT, each sequence control circuit is controlled by its corresponding set of power control switches. If toggle switches A and B are in position CONNECT, sequence control circuits A and B are placed on a common bus, and both can be controlled by one set of power on-off switches. When toggle switch C is in position CONNECT, it will tie the designated sequence control circuits to the second MDL display cabinet.

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 control panel C, there are two pushbutton switches labeled GEN CLEAR A and GEN CLEAR B. The domain of each of these switches depends on the position of the three CONNECT-DISCONNECT switches (explained above under POWER CONTROLS).

There is no direct clear switch located at the operator's console; however, the 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 LD SLCT 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.

The system can be cleared by means of the LOAD switch. When the LOAD switch is depressed at either the console or the maintenance panel, 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.

Processor Register Clear

A set of six pushbutton switches is provided for individually clearing registers A, B, C, X, Y and P of the data processor selected by the display select switch.

Input/Output Processor

The Input/Output Processor registers may be individually cleared with the switches listed below:

1. Switch D clears the data register.
2. Switch C clears the command register.
3. Switch T clears the tag register.
4. Switch TOD clears the time-of-day register.

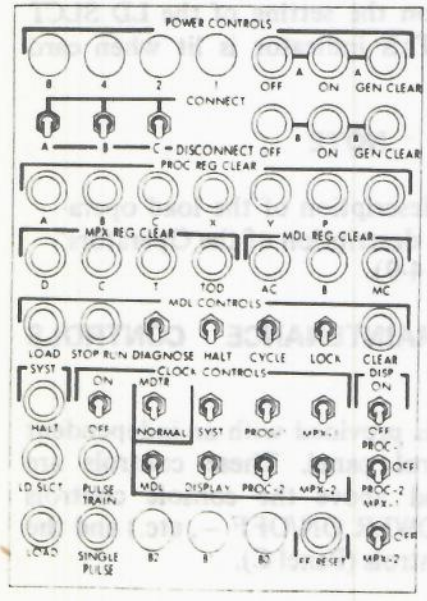


Figure 4-4A. Panel C General Controls

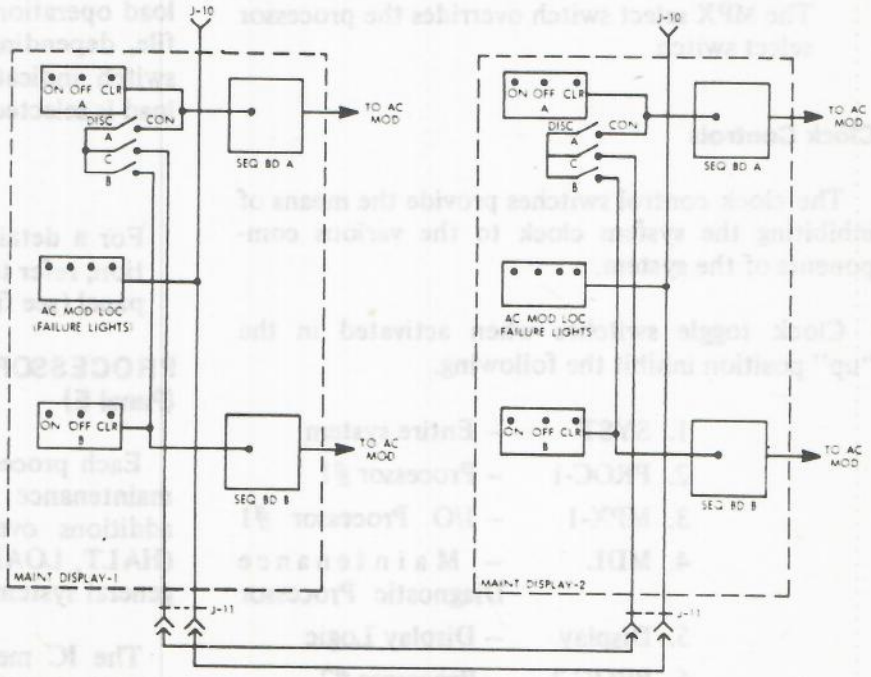


Figure 4-4B.

Figure 4-4. Power Control

MDL Register Clear

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

1. Switch MC clears the core address.
2. Switch B clears the TC no.
3. Switch AC clears the string no.

MDL Control Switches

This group of switches is used for loading and controlling the MDL.

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:

1. On-Off Switch: When this switch is in position ON, the display logic is enabled; when the switch is in position OFF, the display logic is disabled.
2. Processor select switch: This three-position toggle switch selects which of two processors is to be scanned by the MDL.
3. I/O Processor (MPX) select switch: This three-position toggle switch selects which of two I/O Processors is scanned by the MDL. The MPX select switch overrides the processor select switch.

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.

1. SYST – Entire system
2. PROC-1 – Processor #1
3. MPX-1 – I/O Processor #1
4. MDL – Maintenance Diagnostic Processor
5. Display – Display Logic
6. PROC-2 – Processor #2
7. MPX-2 – I/O Processor #2

Single Pulse Switch

This switch is used to produce a single clock pulse 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-nanosecond 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 MDL.

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 functions of these switches are the same as their corresponding switches at the console. The HALT switch is used to halt the system without clearing it. The LOAD switch is used to perform a load operation from either the card reader or disk file, depending upon the setting of the LD SLCT switch indicator. This 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 (see figure 4-9).

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 IC memory registers of the processor are not displayed by the display unit of the system; 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. Refer to figure 4-6, which shows a front view of Panel E.

Start Switch

The START switch is a pushbutton switch which functions to start a halted processor and to execute the next operator syllable pointed to by PSR, PIR, and PBR. This switch is active only when the clock of the processor is enabled and when this switch is depressed it generates a sequence complete level (SECL) to cause the execution of the next operator syllable to be initiated in the normal manner.

Conditional Halt Switch

This is a two-position toggle switch which enables 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 position "down" and functions to stop the data processor when in position"up" (off).

Stop Switches

The following set of stop switches enables 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 position "up" (off), causes the processor to stop after the execution of each operator syllable. It activates the INFL (inhibit fetch level).

INT-I SWITCH

When in position "up", the stop on internal interrupt switch (INT-I) causes the data processor to stop upon the occurrence of an internal interrupt condition. 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), when in position "up" causes the data processor to stop upon the occurrence of an external interrupt. 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 two-position toggle switches used to enable the STOP switches to function when the data processor is in control state or 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 when depressed, functions to clear the flip flops of the related data processor.

Local/Remote Switch

This is a two-position toggle switch which when placed in the LOCAL position, places the data processor in a local state. The processor unit functions normally when in the LOCAL state except for the following:

1. 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.
2. The facilities of the READ PROC REG switches are enabled.

ADJ (O, O) Switch

This is a pushbutton switch which activates the push-down stack register operator to cause all TOS registers to be stored in memory, thereby saving the contents of the A and B registers so that these registers may be used to subsequently manipulate the data processor's IC memory via the maintenance panel switches (READ-IC and WRITE-IC). The ADJ (O, O) switch is active only when the processor's clock is enabled.

Read IC Switch

This is a pushbutton switch which initiates a read processor register 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 clock of the processor is enabled.

READ IC OPERATION

To perform the read IC operation, do the following:

1. Adjust O, O.
2. Load the address in the B register.
3. Set BROF.
4. 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 by 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 OPERATION

To perform the write IC operation, do the following:

1. Adjust O, O.
2. Load the address in the B register.
3. Load the information to be written in the A register.
4. Set AROF and BROF.
5. Depress the WRITE IC pushbutton; the contents of the A register will be written in the cell addressed.

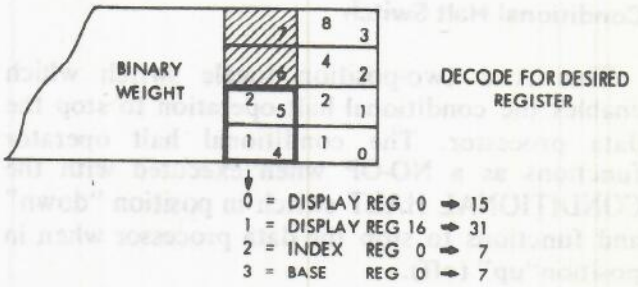
Read Proc Reg Switches

These switches enable the read out and display of the related processor register (IC memory register). The contents of the register 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 and their functions are listed below:

1. Switch S is the read S register switch.
2. Switch F is the read F register switch.
3. Switch PBR is the read PBR register switch.
4. Switch PIR is the read PIR register switch.
5. Switch BOSR is the read BOS register switch.
6. Switch LOSR is the read LOS register switch.

NOTE

These IC memories are displayed in the SM register.



Register Name	Usage	Decimal Address	Hexadecimal Address
DOO		0 ⇒	00 ⇒
D31	Display	31	1F
PIR	Program index	32	20
SIR	Source index	33	21
DIR	Destination index	34	22
TIR (BUF 3)	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 (BUF 2)	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

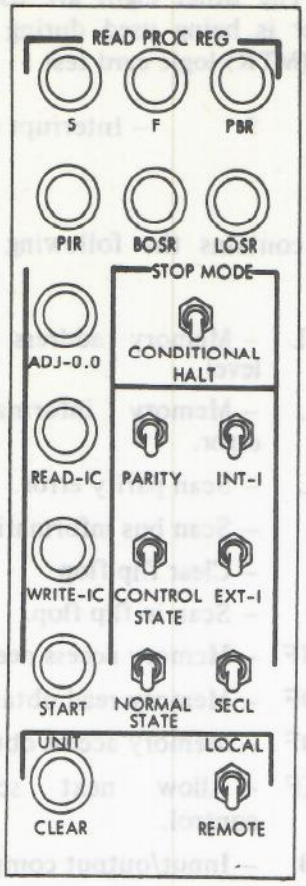


Figure 4-6. Panel E

Input/Output Processor Registers and Flip Flops

The Input/Output Processor registers and flip flops are displayed on Panel B as shown in figure 4-7. This panel is shared with the Processors for display mode.

Row B

This row contains the logical elements for MDL. Each flip flop may be used in one of two ways: I/O testing or data processor testing.

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
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.

FLIP FLOP	USE ON I/O TEST	USE ON MDL 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
LO1F, LO2F	Sequence count	Sequence count
MAOO ⇒ MAO9	Memory address	Memory address
BO1F ⇒ BO8F	Tape read control	Data
AO1F ⇒ A1Of	Character buffer word buffer	Command-data
CO1F ⇒ CO8F	Card address register	Card address 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 five MTRI flip flops, one for each pair of Tag registers.

Row F

This row contains the following I/O Processor control flip flops:

IC 1	⇒	8	Initiate count cycle for operational sequence flow.
KY 1	⇒	5	Key register 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
MINF	–		Minus level store
RDAF	–		Result descriptor access

Row G

This row contains the time-of-day register and the interrupt status bit flip flops.

TIME OF DAY O ⇒ 43 – This register contains 44 flip flops of which 36 are used for time-of-day. The other eight are used when the entire register is being used during maintenance test routines (MTR) logic card test.

IS O ⇒ 9 – Interrupt status bits.

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
SIO6	⇒ SI17

Input/Output Processor Maintenance Control Panel

Panel D (figure 4-8) is used for local maintenance operations with the I/O Processor. Four types of operations can be accomplished using this panel:

1. Reading and writing the I/O Processor scratch-pad memory.
2. Reading and writing main memory.
3. Executing I/O descriptors.
4. Logic card testing.

The requirements for these operations are two-fold: the I/O Processor Local/Remote switch must be in position LOCAL and the I/O Processor display mode must be active as well as system clock.

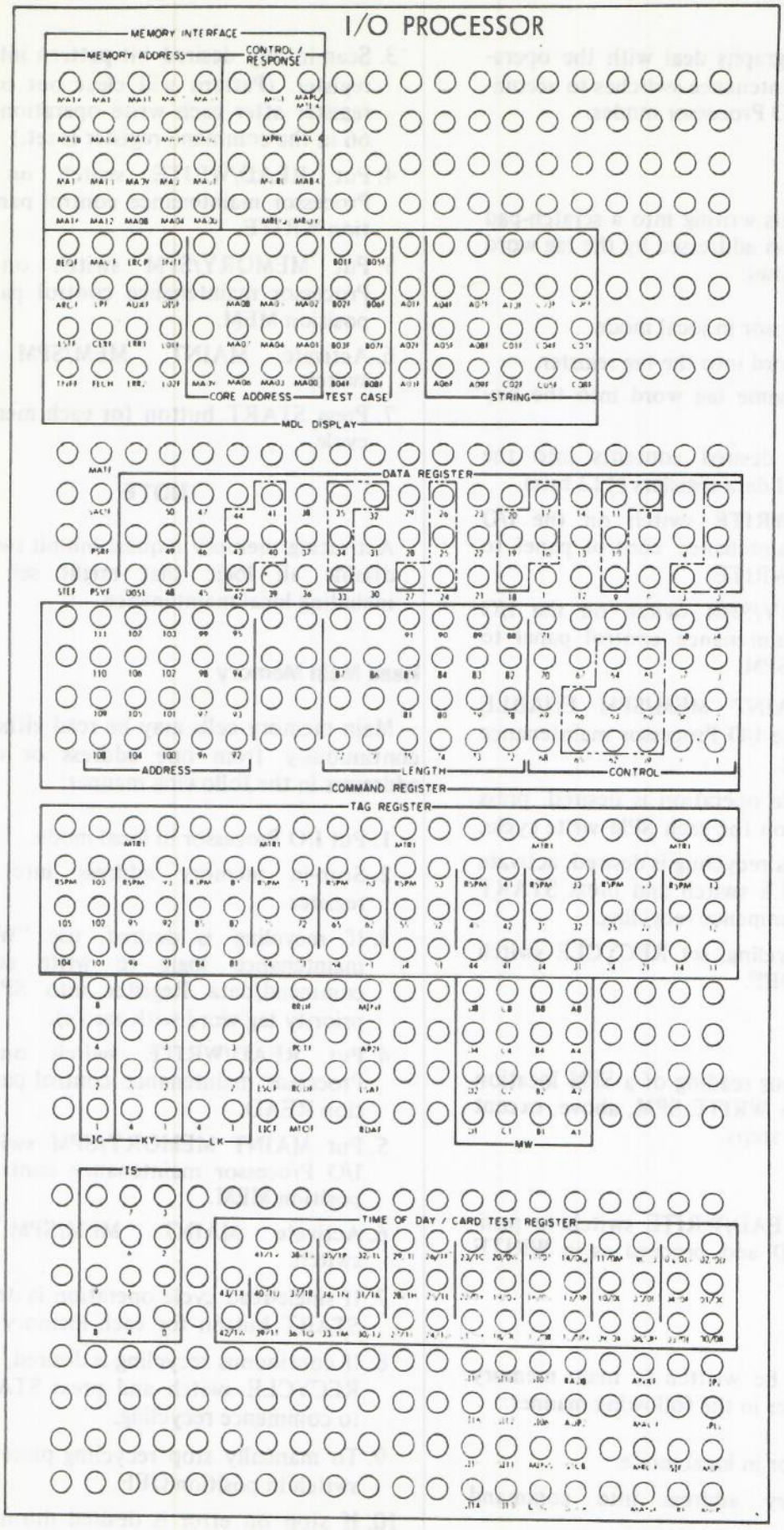


Figure 4-7. Input/Output Processor Display Panel B

The following paragraphs deal with the operational use of these maintenance switches to accomplish the above four I/O Processor modes.

Write SPM

Single or Continuous writing into a scratch-pad memory (SPM) location addressed by the tag word is accomplished as follows:

1. Put I/O Processor in local mode.
2. Scan-in tag word into the tag register.
3. Scan-in the same tag word into the key register.
4. Scan-in the desired contents into the command and data registers (112 bits).
5. Put READ/WRITE switch on the I/O Processor maintenance control panel to the position WRITE.
6. Put MEMORY/SPM switch on the I/O Processor maintenance control panel to the position SPM.
7. Activate MAINT MEM/SPM ENABLE switch on the I/O Processor maintenance control panel.
8. If single-cycle operation is desired, press START button for each SPM write cycle.
9. If continuous recycling is desired, activate the RECYCLE switch and press START button to commence recycling.
10. To stop recycling, set RECYCLE switch to position OFF.

Read SPM

Single or Continuous reading of a SPM location is accomplished as in WRITE SPM, above, except for the following two steps.

Step 4 – Omit

Step 5 – Put the READ/WRITE switch to position READ 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:

1. Put I/O Processor in local mode.
2. Scan-in memory address into command register.
3. Scan-in any desired bit pattern into the data register. (Pattern will clear out of the data register after each write operation unless bit 66 in the command register is set.)
4. Put READ/WRITE switch on the I/O Processor maintenance control panel to position WRITE.
5. Put MEMORY/SPM switch on the I/O Processor maintenance control panel to the position MEM.
6. Activate MAINT. MEM/SPM ENABLE switch.
7. 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:

1. Put I/O Processor in local mode.
2. Scan-in memory address into command register.
3. If recycling is desired, use "Write SPM" maintenance logic to write contents of command/data Register into SPM (highest priority tag word with zero's).
4. Put READ/WRITE switch on the I/O Processor maintenance control panel to position READ.
5. Put MAINT MEMORY/SPM switch on the I/O Processor maintenance control panel to position MEM.
6. Activate MAINT. MEM/SPM ENABLE switch.
7. If single-read cycle operation is desired, press START button for each memory read cycle.
8. If continuous recycling is desired, activate the RECYCLE switch and press START button to commence recycling.
9. To manually stop recycling place RECYCLE switch in position OFF.
10. 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.

11. Note that activating MEM INHIBIT REQUEST switch will disable all logic that might set MANF including local maintenance.
12. Activating the INHIBIT Mem ADRS 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:

1. Put I/O Processor in local mode.
2. Scan-in area and I/O descriptors into command/data registers. The specified unit designate field selects the channel on which the descriptor is to be executed.
3. 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.

4. Place MAINT MEM/SPM ENABLE switch in position OFF.
5. Place MAINT DESCRIPTOR ENABLE switch in position ENABLE.
6. 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:

1. Steps 1 through 5 are the same as the maintenance descriptor (single) procedure.
6. Activate RECYCLE switch.
7. 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.
8. To manually stop the recycling, set RECYCLE switch to position OFF.
9. If stop on error is desired during recycling, activate the ERROR STOP switch. Upon detection of a result descriptor error from the peripheral control 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.

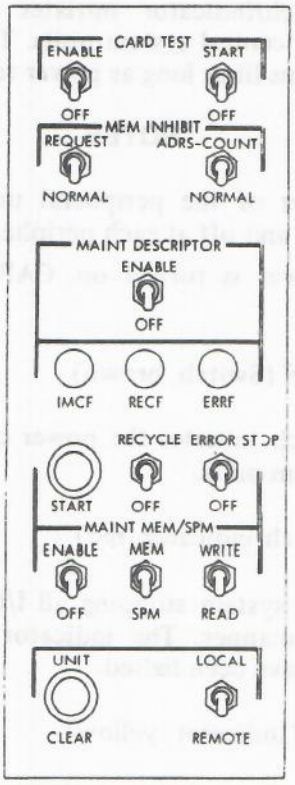


Figure 4-8. Panel D Input/Output Processor Maintenance Control Panel

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 I/O Processor backplane. The testing procedure is activated by putting the CARD TEST ENABLE switch to position "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 (figure 4-9) contains an operators panel and a visual message control center for communicating with the operating system. A total of eight 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.

NOTES

1. The power of the peripheral units must be turned on and off at each peripheral unit.
2. When power is turned on, CARD LOAD is selected.

POWER OFF (Switch, brown)

This switch initiates the power off cycle for all central system units.

HALT (Switch/indicator, red)

Halts 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 two-second run timers, in each processor. Each processor timer is triggered when that processor executes an interrogate peripheral unit status operator. The run indicator is lit when the timer in any processor which is in remote is ON. If all processors are in local mode, 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:

1. The load timer in the processor-interrupt controller is triggered to produce an 800-nanosecond (LSIG) signal which is sent to I/O Processor – A.
2. Address registers LOSR, BOSR, F, STKNR, and Display O are set to zero.
3. Register S is set to 8192.
4. PDR (program dictionary index) is set to a value of 4.
5. PIR (program index register) is set to a value of 1.
6. The processor is forced into an idle state to await an expected I/O finished interrupt.
7. The I/O Processor responds to the load signal by jamming the appropriate unit number into the command/data register. The I/O Processor sequence control logic is set to IC 02, and the card read cycle is started.
8. The information (a bootstrap program) on the EBCDIC punched card is read into the first twelve memory locations. This information contains tag fields. (Seven characters per word.)
9. At the end of the successful card read, the I/O Processor sends an I/O finish interrupt to the processor. It responds by entering a hardware interrupt handling procedure. Memory cell DO + 3 contains the PCW of the bootstrap program 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 first segments of disk memory. This type of an operation is used to bring the first portion of the operating system into core memory.

The same hardware functions take place as for card read except for the following:

1. A disk unit number is placed in the command/data register because the LOAD select switch selected a DISK LOAD.
2. The I/O finish interrupt reflects a disk operation instead of a card operation.
3. 8192 words are read instead of 12.

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.

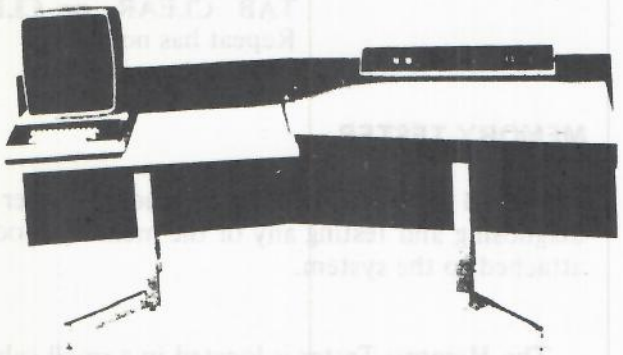


Figure 4-9. Operators Control Console

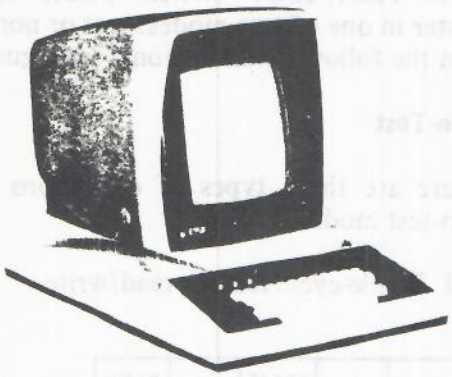


Figure 4-10. Visual Message Control Center

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.
⨂ ETX	End-of-text character. Places the end-of-text character at the cursor location.
↖ HOME	Causes the cursor to be moved to the home (upper left) position.
LINE ERASE	1. 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. 2. Line Erase will not function unless Erase Lock is depressed simultaneously with Line Erase.
↖ CLEAR	1. Unshifted – CLEAR erases all data on the screen except tab flags; and, with Forms Option, data bracketed by Shift-In/Shift-Out. 2. Shifted – CLEAR erases all data on the screen and all tab flags. 3. 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	1. 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.

2. 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 — Unshifted – TAB CLEAR causes the removal of the tab stop flag located at the cursor position in all lines.

↓ (Line Feed) — Line Feed (LF) moves the cursor down one line. When the cursor is in the bottom line, Line Feed causes it to reappear in the top line.

↑ (Reverse Line Feed) — Reverse Line Feed (RLF) 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 (BS) cursor one character. When the cursor is at left edge of page, backspace causes it to reappear at right edge of page in the same line.

→ (Forward Space) — Forward Space (FS) moves the cursor one space to the right. If the cursor is at right edge of page, Forward Space causes it to reappear at the left edge down shifted one line. If the cursor is located in last position of bottom line, Forward Space 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 other key will be repeated at a rate of about 15 Hertz. When depressed in conjunction with LOC, REC, XMIT, TAB CLEAR or CLEAR Repeat has no effect.

MEMORY TESTER

The B 6700 includes a Memory Tester for diagnosing and testing any of the memory modules attached to the system.

The Memory Tester is located in a small cabinet, with its display panel as shown in figure 4-12. The NON–TEST/TEST switch places the Memory Tester in one of two modes: test or non-test, which is in the following discussion. (See figure 4-13.)

Non-Test

There are three types of operations used in the non-test mode:

1. Single-cycle read or read/write.

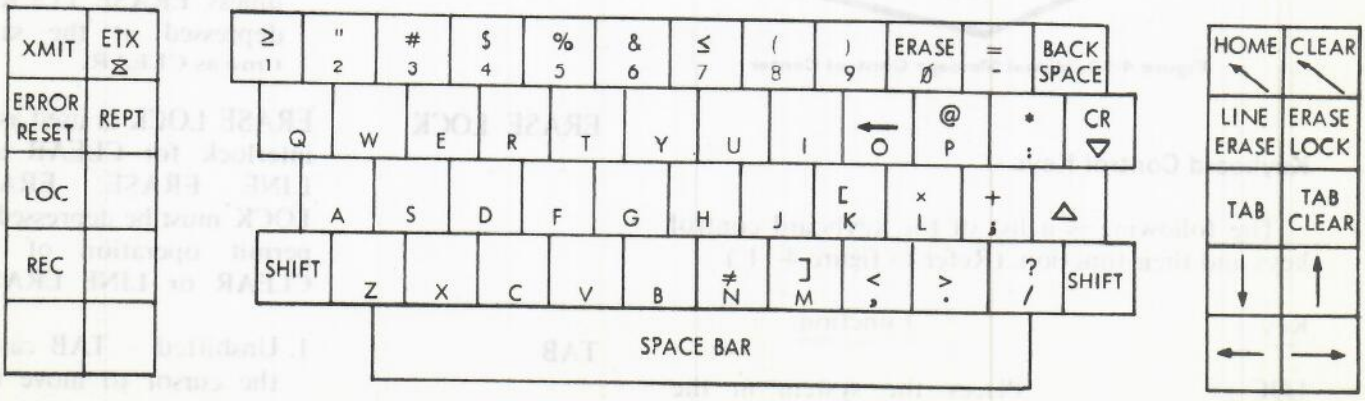


Figure 4-11. Keyboard Format

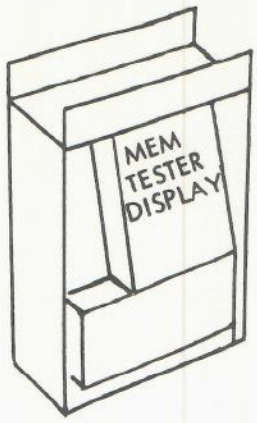


Figure 4-12. Memory Tester

2. Search memory(s) for specific data; search for equal or unequal.
3. Sample a given address for changes.

Test

The following operations are performed when the corresponding test pattern switches are in position "up" (on). None of the patterns selected checks for parity errors when the WRITE ONLY/NORM/READ ONLY switch is in position READ ONLY. More than one test pattern can be selected, and the pattern will run in the order given below:

1. Test-pattern MANUAL INSERT selected enables a fixed test pattern.
2. Test-pattern ALL–1 selected runs an all-"one" test.
3. Test-pattern ALL–0 selected runs an all-"zero" test.
4. Test-pattern CHECKERBOARD selected runs a checkerboard pattern, writing two 0's, then two 1's.
5. Test-pattern CHK'BD COMPL selected runs the checkerboard complement pattern test.
6. Test-pattern BIT COMPLEMENT selected runs the bit complement pattern test.
7. Test-pattern COMPL – BIT COMPL selected runs the complement bit complement pattern test.
8. Test-pattern WALKING–1 selected runs the full walking "one" pattern test.
9. Test-pattern WALKING–0 selected runs the full walking "zero" pattern test.
10. Test-pattern MEM CLEAR selected runs the memory clear pattern master reset test.

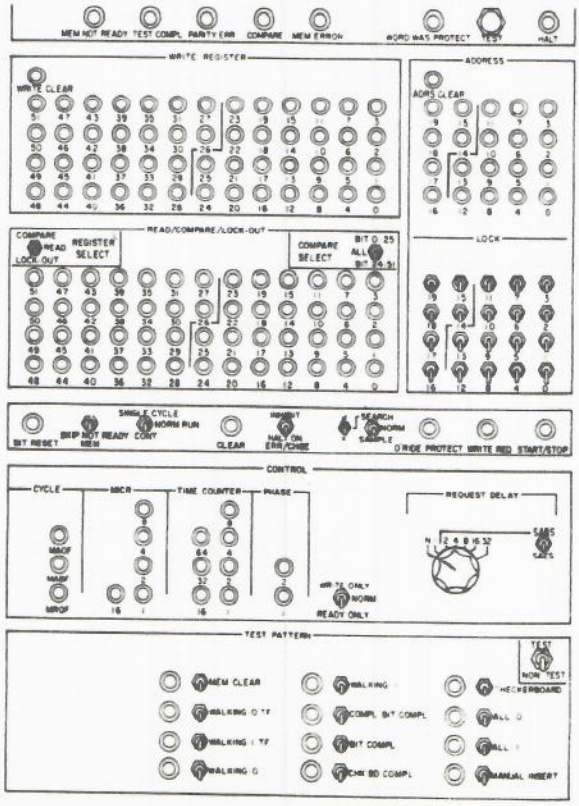


Figure 4-13. Memory Tester Panel

SECTION 5
SYSTEM CONCEPT

GENERAL

The B 6700 system consists of a maximum of three Processors, a maximum of three I/O Processors, Main Memory, a Memory Tester, one or more Power modules, an Operators Console, one or two Maintenance Diagnostic Processors (MDL), one or two Display Panels, one to six 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 (ZO 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.

The operators are grouped into 10 groups called the Operator Families (figure 5-1). The grouping of related operators into families minimizes the logic required in the processor. The 10 families of operators with a brief purpose for each are:

1. Family A OPS – Arithmetic Operators
2. Family B OPS – Logical Operators
3. Family C OPS – Subroutine Operators
4. Family D OPS – B 6700 Word Oriented Operators
5. Family E OPS – Scaling Operators
6. Families F,G,H, OPS – String Operators
7. Family J OPS – Value Call
8. 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. The Program Controller also selects and decodes the syllable to be executed, and furnishes this OP code to all the Family Controllers through the Z10 bus. The Program controller strobes the proper OP family, allowing that OP family to proceed through its logical steps of performing the function of that

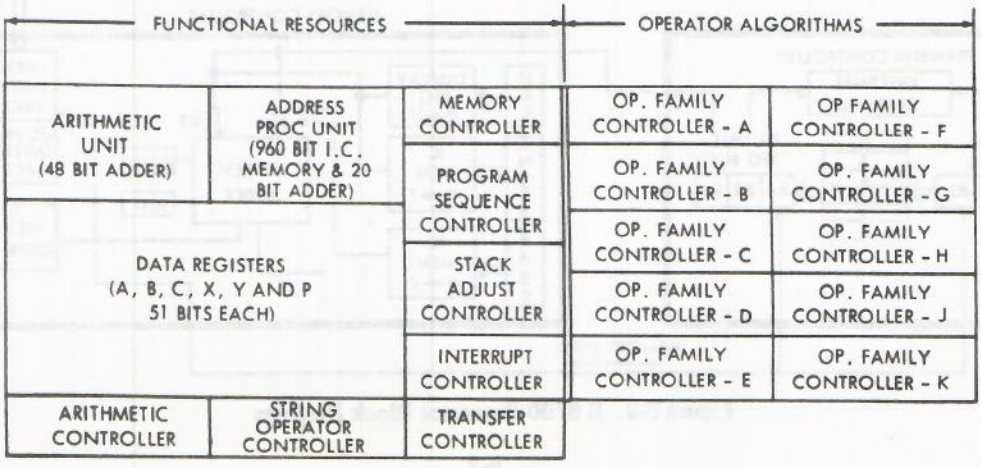


Figure 5-1. B 6700 Processor Organization

operator. At the completion of the operator, an 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 through 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 6700 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 I/O Processor.

The A and B registers are the Top of Stack registers, and 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:

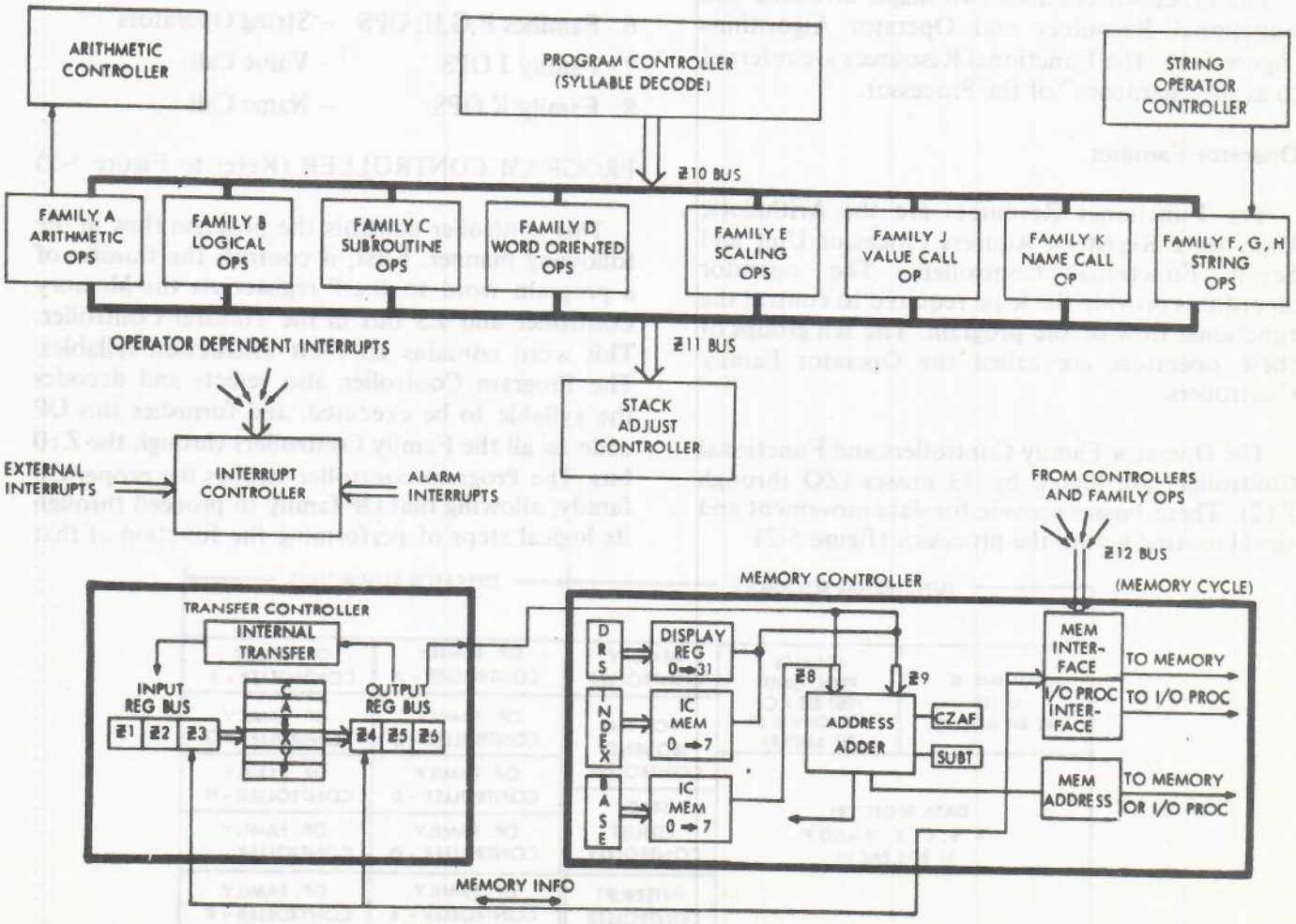


Figure 5-2. B 6700 Processor Block Diagram

1. A direct, full-word transfer path using the Z5 and Z2 busses.
2. 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.
3. A steering Network and Mask network providing a field displacement between stack registers using the Z6 and Z1 busses.
4. 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.
5. 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.
6. 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 as follows:

1. TOA (TOP OF APERTURE) – the highest bit position of the accepting field (highest bit of the window).
2. TOM (TOP OF MASK) the highest bit position to be inhibited n the transfer (lowest bit of the window).
3. DIS (DISPLACEMENT) – a right shift of the bits through the steering matrix.

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.

INTERRUPT CONTROLLER (Refer to figure 5-2)

The Interrupt Controller provides a method of 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 were not modified by the operator. Bit 23 identifies class 2 interrupts, where the values were changed by the operator before the interrupt.

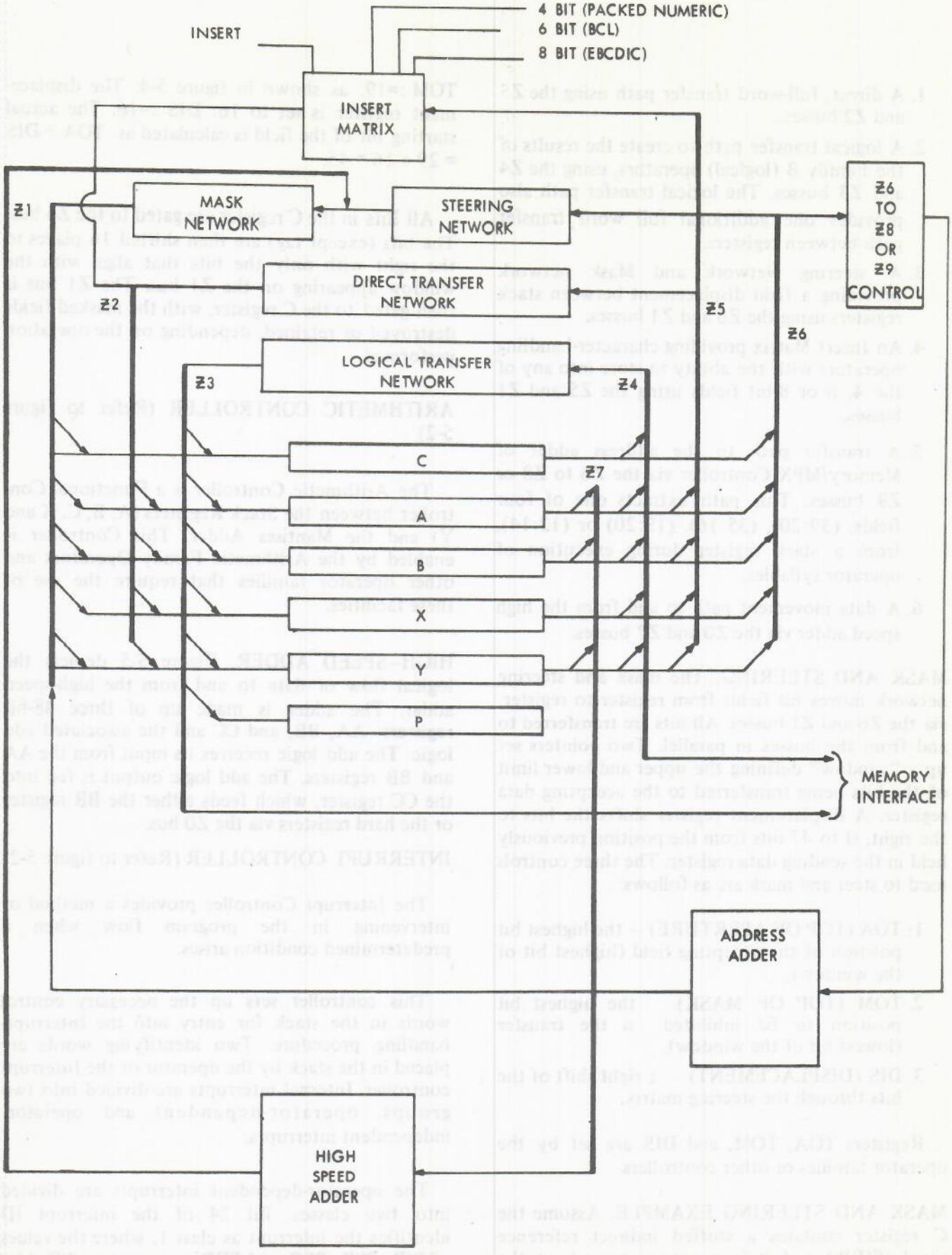


Figure 5-3. Internal Data Transfer Section

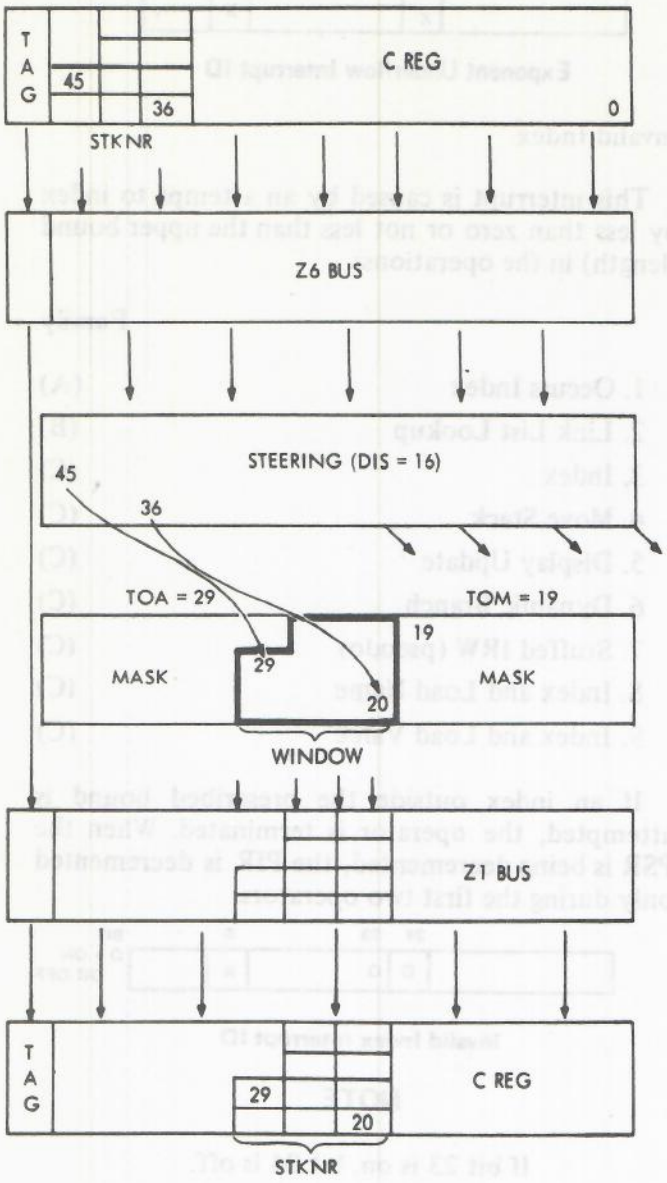


Figure 5-4. Mask and Steering

OPERATOR-DEPENDENT INTERRUPTS. These interrupt conditions are sensed by the operator and normally result in a premature termination of the operator under control of the logic of the operator. 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:

1. Memory Protect
2. Invalid Operand
3. Divide by Zero
4. Exponent Overflow
5. Exponent Underflow
6. Invalid Index
7. Integer Overflow
8. Bottom of Stack
9. Presence Bit
10. Sequence Error
11. Segmented Array
12. Programed Operator

Memory Protect.

This interrupt occurs under the following conditions:

1. A store, overwrite, read/lock or string transfer is attempted using a Data Descriptor that has the read only bit on (bit 43). The operation is

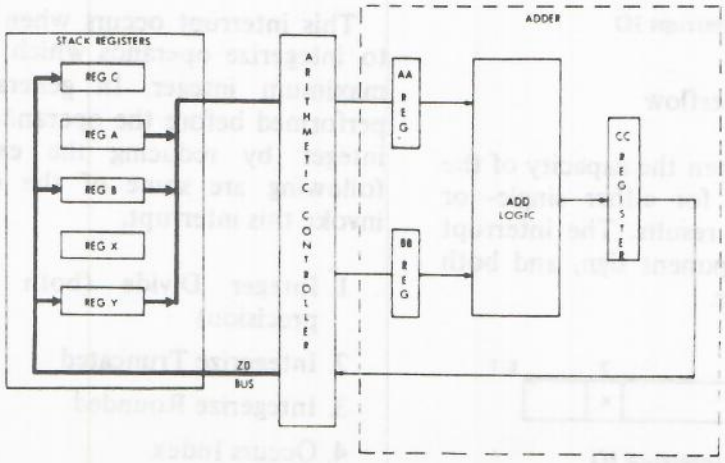
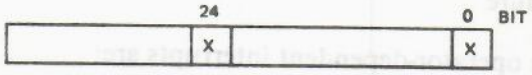


Figure 5-5. Arithmetic Control

terminated prior to the memory access, leaving the descriptor in the A register.

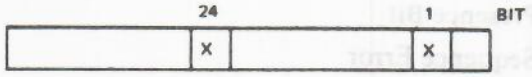
2. A store is attempted into a word in memory that has a tag field representing program code, RCW, MSCW, or Segment Descriptor. The memory write is aborted when bit 48 is detected in the "flashback" word that is placed into the C register. The operation is terminated leaving the original addressing word in the A register.



Memory Protect Interrupt ID

Invalid Operand

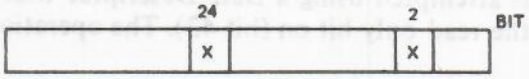
This interrupt occurs when operators attempt to use the wrong types of control words or data. When control words and data are accessed, they are checked to ensure that they meet the necessary requirements of the operator being executed. When the interrupt occurs, the operator is terminated prematurely.



Invalid Operand Interrupt ID

Divide by Zero

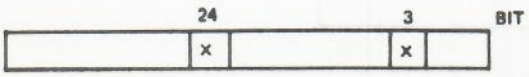
This interrupt results when a division operator is attempted with the divisor equal to zero. This interrupt terminates the operation prematurely, leaves the A register cleared, the interrupt ID in the B register, and PSR and PIR backed up to point to the initiating operator.



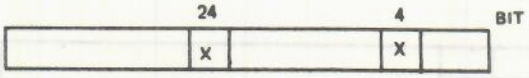
Divide by Zero Interrupt ID

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 interrupts clear the A register.



Exponent Overflow Interrupt ID



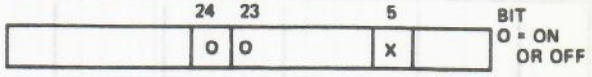
Exponent Underflow Interrupt ID

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
1. Occurs Index	(A)
2. Link List Lookup	(B)
3. Index	(C)
4. Move Stack	(C)
5. Display Update	(C)
6. Dynamic Branch	(C)
7. Stuffed IRW (pseudo)	(C)
8. Index and Load Name	(C)
9. Index and Load Value	(C)

If an index outside the prescribed bound is attempted, the operator is terminated. When the PSR is being decremented, the PIR is decremented only during the first two operators.



Invalid Index Interrupt ID

NOTE

If bit 23 is on, bit 24 is off.

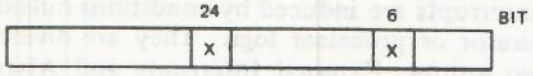
Integer Overflow

This interrupt occurs when an attempt is made to integerize operands which have a greater than maximum integer. In general, the checking is performed before the operand is converted into an integer by reducing the exponent field. The following are some of the operators that may invoke this interrupt.

1. Integer Divide (both single and double precision)
2. Integerize Truncated
3. Integerize Rounded
4. Occurs Index

5. Integerize Rounded, Double Precision

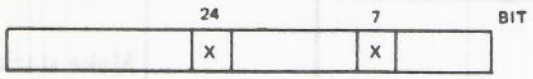
If the interrupt is invoked, the operator is terminated.



Integer Overflow Interrupt ID

Bottom of Stack

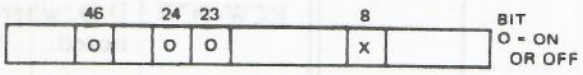
This interrupt is used to inform the Operating System that a Return or Exit Operator has caused the program stack to be returned to its base. If this condition arises, the operator will terminate with the last accessed RCW (Return Control Word) left in the A register.



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:

1. Data-Dependent
2. 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.

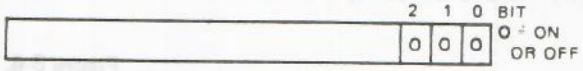
The P-bit procedure makes the non-present reference present prior to returning to the interrupted program. The PIR and PSR setting for the current operator are saved in the RCW for data-dependent presence-bit interrupts.

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 update and when the processor is trying to access 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.

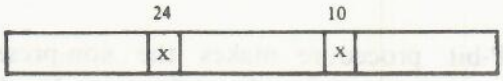
Program Restart. In order to restart some operators after a presence bit interrupt, it is necessary for the P-bit procedure to return either an IRW or Data Descriptor. The "RT-bit" in the presence bit ID (P1) indicates to the P-bit procedure whether to perform an exit or return operator when returning 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 and PIR), exit/return and "RT-bit" relationship to the various presence bit interrupt conditions.

Segmented Array

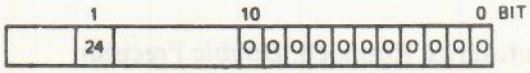
This interrupt is used by the string operators as an upper limit boundary detection. Arrays in main memory may be segmented into groups of 256 words each, bounded on both ends by memory link words. Each word read from memory during string operator executions is checked for the presence of bit 48 (memory protect). If the bit is on, the segmented-array interrupt is set. String operator interrupts leave a special parameter in the A register. This parameter 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 Interrupt ID

Programed Operator

This interrupt is used for the detection of invalid operator codes. Primary codes BC, E7, EF, F6, and F7 are detected and cause this 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.

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

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	(1) IRW (stuffed)	Int. I.D.	0	Exit	S_n (4)	Make stack or stack vector present.
		(2) IRW	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, or Move Stack

(3) RT bit is packed in the Int. I.D. (P_1)

(4) S_n indicates the PIR and PSR point to current operator syllable

(5) Move Stack operators

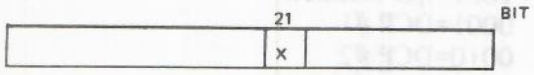
Figure 5-6. Presence Bit Interrupt

normally result in a momentary interruption of a program process which will be continued after handling or recording of 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 as follows:

1. Processor to Processor interrupt
2. Special Control interrupts
   a. Interval timer
   b. Stack overflow
3. I/O Processor interrupts
   a. I/O finish
   b. Data Communications
   c. General Control Adapter
   d. 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.

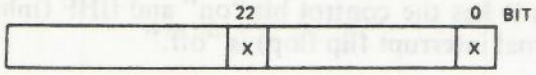


Processor to Processor Interrupt ID

This interrupt also is 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

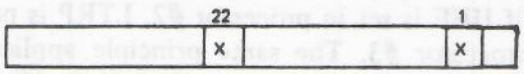
This interrupt is used for programmatic time slicing. The interval timer is activated by the SINT (Set Interval Timer) operator. The timer is set to the value of bits 10:11 of the B register and decrements every 512 microseconds until equal to zero. At this time, if the timer is still armed, the interrupt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

Input/Output Processor Interrupts

The Input/Output Processor interrupts may be handled by any system processor. A priority is established between I/O Processors and Processors to determine which Processor responds when an interrupt is present. This is necessary when multiple Processors and I/O Processors are present because they all share a common scan bus.

Since the scan bus allows for only two I/O Processor external interrupt paths (EIMA and EIMB), provisions have been made to allow I/O Processor C to use the interrupt path of I/O Processor B (EIMB). When I/O Processor C completes an I/O operation, an I/O finished interrupt is sent, via a separate coaxial cable, to the external interrupt card in I/O Processor B. I/O Processor B sends this interrupt to the appropriate processor by means of the I/O Processor B interrupt path, on the scan bus (EIMB). The processor that accepts the interrupt scans-in a read interrupt literal from I/O Processor B. The interrupt literal from I/O Processor B denotes to the designated processor that the external interrupt originated in I/O Processor C.

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 third clock pulse.

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

Priority Handling

The priority is established with left-to-right priority (LTRP) for I/O Processor A and right-to-left priority (RTLP) for I/O Processor B, which allows the appropriate I/O Processor to place its interrupt in the appropriate processor. Figure 5-7 is a hypothetical system configuration used for explanatory purposes.

LTRP is true for processor #1 and RTLP is true for processor #3. The priorities are passed to another processor when IIHF is set. If IIHF is set in processor #1, LTRP is passed to processor #2; and if IIHF is set in processor #2, LTRP is passed to processor #3. The same principle applies for RTLP, with the exception that it is true for processor #3 and is passed to processor #2. The priorities in a processor are reestablished when IIHF is reset in any processor. Only the processor that has LTRP true and IIHF false can accept interrupts from I/O Processor A, and only the processor that has RTLP true and IIHF false can accept interrupts from I/O Processor B.

Priority-Handling Example With IIHF Off

Assume I/O Processors A and B have I/O finished interrupts occurring at the same time and all three processors are in the normal state (IIHF "off"). LTRP will be true for processor #1, which gates the external interrupt from I/O Processor A to processor #1, and RTLP is true for processor #3, which gates the external interrupt from I/O Processor B to processor #3. Since only one processor can communicate on the scan bus at any given time, the processor with the scan bit on will have priority and communicates on the scan bus first.

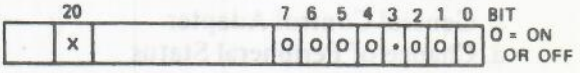
Priority Handling Example With IIHF On.

Assume I/O Processors A and B have I/O finished interrupts occurring at the same time, and processor #1 and processor #2 are both in control state (IIHF "on"). LTRP and RTLP will both be true for processor #2 and the external interrupts from I/O Processor A and I/O Processor B are both gated to processor #2. The hardware in the interrupt controller of all processors assigns I/O Processor A the highest priority; processor #2 will subsequently scan in the interrupt literal from I/O Processor A, while I/O Processor B will hold its interrupt since I/O Processor interrupts are not reset until a scan-in is performed. RTLP can still be passed to processor #1 or returned to processor #3 if either should return to normal state while processor #2 is still in control state; thus, each system processor is capable of handling external interrupts from any I/O Processor.

I/O Finish And Data Communications Interrupts

Both interrupts are handled by the Interrupt Controller as follows:

1. An SCNI (SCAN-IN) operator is forced into the Processor at the next SECL to read the interrupt literal into the B Register.
2. The normal operation of entry to the Interrupt Handling Procedure is then executed.



*LEFT OPEN FOR FURTHER EXPANSION

External Interrupt ID

NOTE

Bits 2:3 identify which I/O Processor the literal was read from.
I/O Processor A=001, I/O Processor B=010,
I/O Processor C=100.
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 that a special control device such as an Analog device, a plotter, or some machine being controlled by the system requires communication with the Processor.

Alarm Interrupts

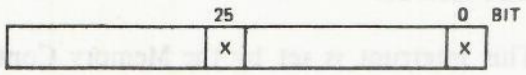
These interrupt conditions are not anticipated and inform the system of some detrimental change in environment. They normally result from either a programing error or hardware failure. The alarm interrupt conditions are recognized upon

occurrence by the interrupt controller. The interrupt controller assumes control of the machine, clears the activated operator family, marks the TOS registers full and activates the pseudo interrupt operator. In either case (programing error or hardware failure) the current operator is terminated prematurely. The alarm interrupts are:

1. Loop
2. Memory Parity
3. I/O Processor Parity
4. Invalid Address
5. Stack Underflow
6. 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. If the interrupt occurs, the ID remains in the B register, the A register is cleared and PIR is backed up.



Loop Interrupt ID

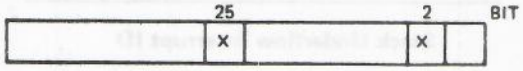
Memory Parity

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

I/O Processor Parity

This interrupt is the same as Memory Parity above, except that it is used for Processor/I/O Processor transfer.



I/O Processor Parity Interrupt ID

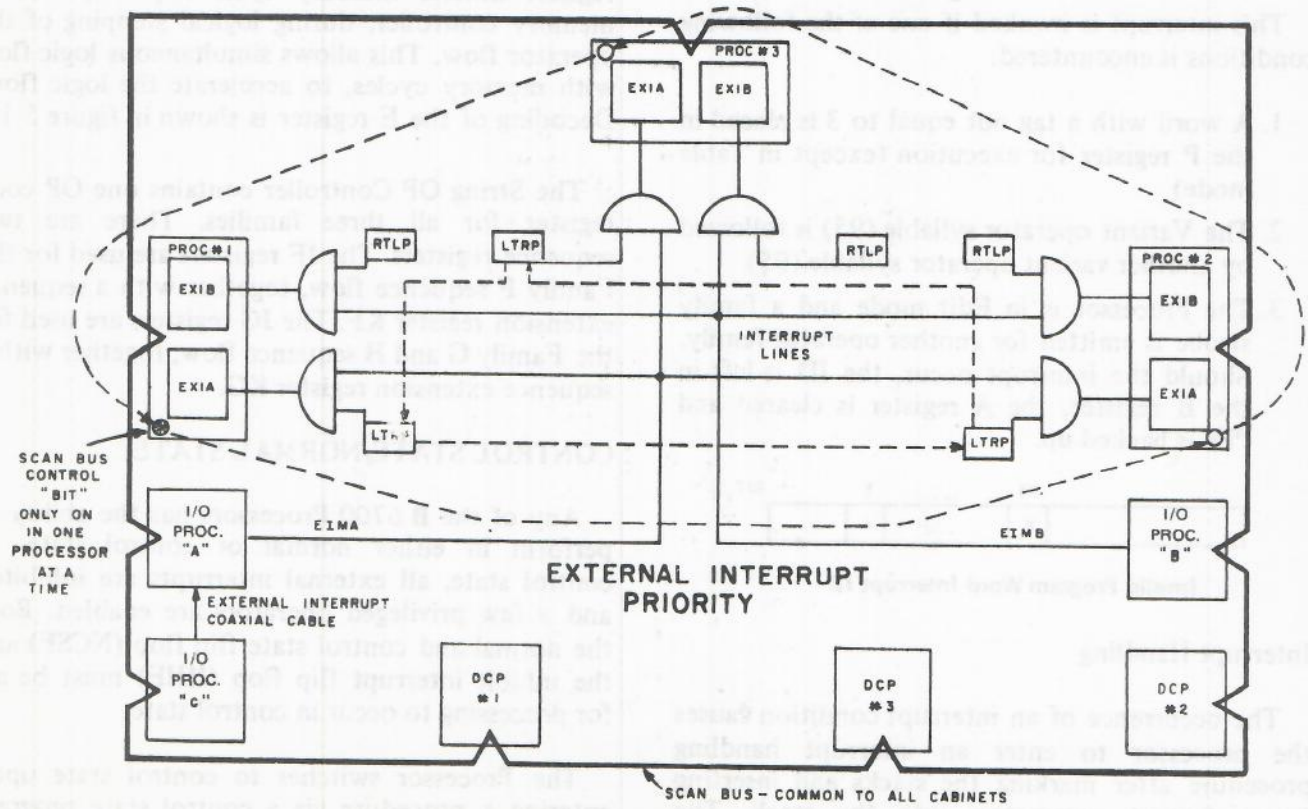
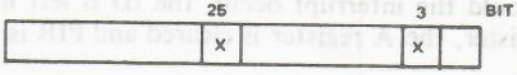


Figure 5-7. B 6700 Scan Bus Priority Control

Invalid Address

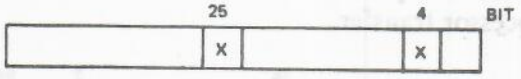
This interrupt is set by the Memory Controller when it fails to obtain an acknowledgement to a memory request within eight clock periods. This indicates that an attempt has been made to access a non-existent memory module. The Memory Controller initiates the interrupt and the Interrupt Controller leaves 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 F during stack adjustment. If this interrupt occurs, the ID remains in the B register, the A register is cleared and PIR backed up.

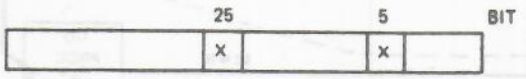


Stack Underflow Interrupt ID

Invalid Program Word

This interrupt is invoked if one of the following conditions is encountered:

1. A word with a tag not equal to 3 is placed in the P register for execution (except in Table mode).
2. The Variant operator syllable (95) is followed by another variant operator syllable (95).
3. 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, inserted into the stack as the interrupt condition are recognized, and 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 introducing an enter operator with an IRW pointing to DO + 3 at F + 1. The hardware expects to find a PCW at DO + 3; however, either an IRW or an IRW chain pointing to a PCW is a legitimate condition.

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 is by having the E register initiate memory cycle requests, via the memory controller, during logical stepping of the operator flow. This allows simultaneous logic flow with memory cycles, to accelerate the logic flow. Decoding of the E register 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 of the B 6700 Processors has the ability to perform in either normal or control state. In control state, all external interrupts are inhibited and a few privileged operators are enabled. Both the normal and control state flip flop (NCSF) and the inhibit interrupt flip flop (IIHF) must be set for processing to occur in control state.

The Processor switches to control state upon entering a procedure via a control state program

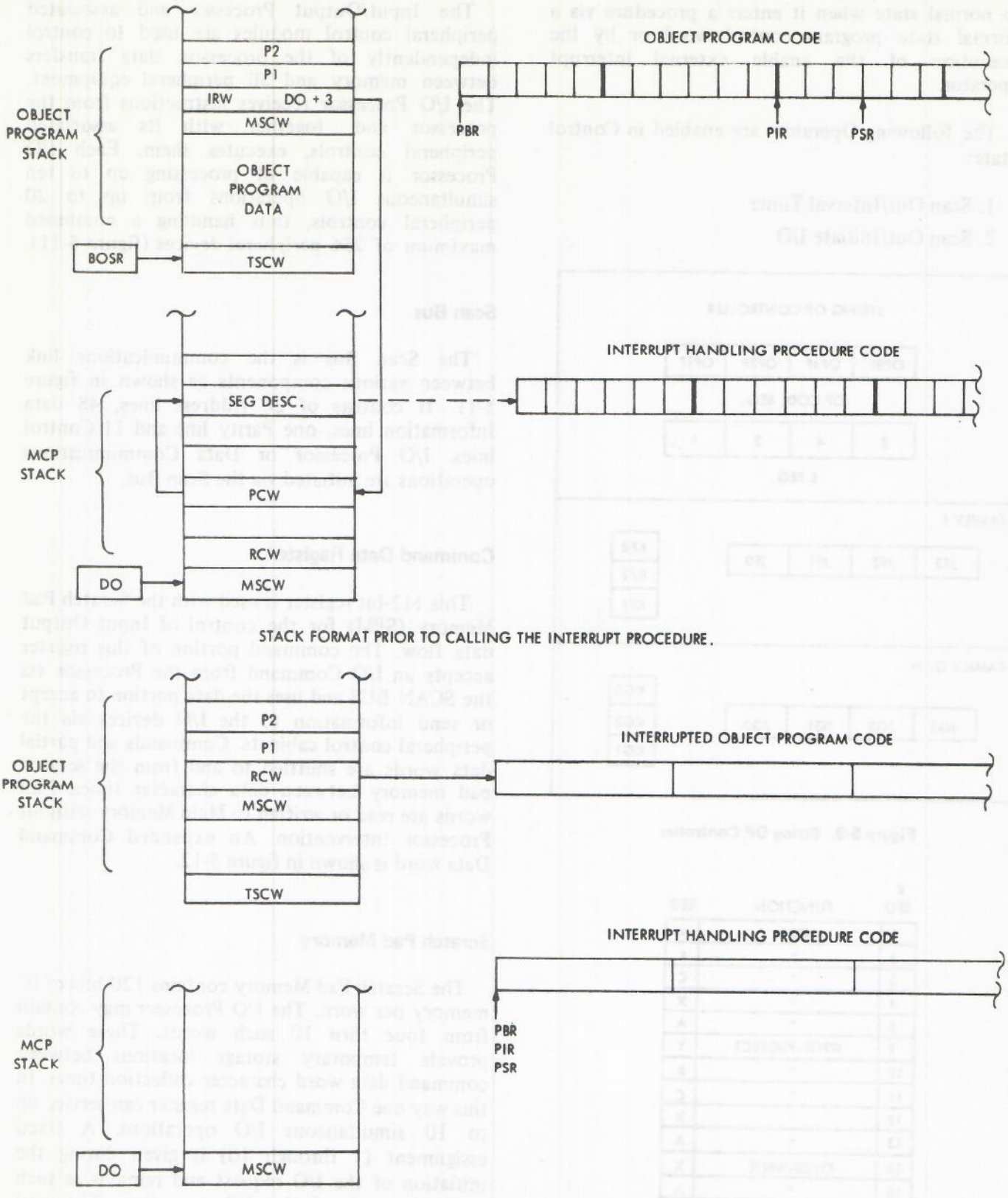


Figure 5-8. Stack Format

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

The following Operators are enabled in Control State:

1. Scan Out/Interval Timer
2. Scan Out/Initiate I/O

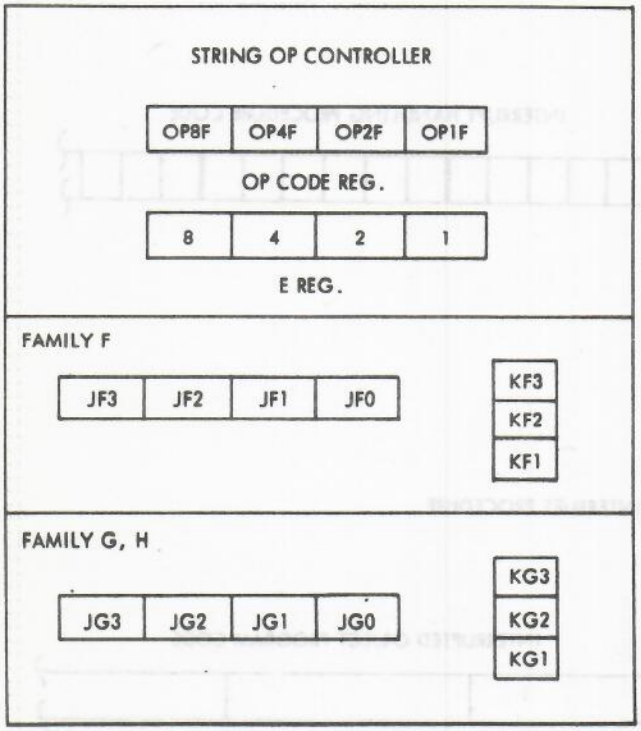


Figure 5-9. String OP Controller

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 PROCESSOR

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

Scan Bus

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

Command Data Register

This 113-bit register is used with the Scratch Pad Memory (SPM) 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 Memory contains 120 bits of IC memory per word. The I/O Processor may contain from four 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 shown in figure 5-12 will contain this assignment.

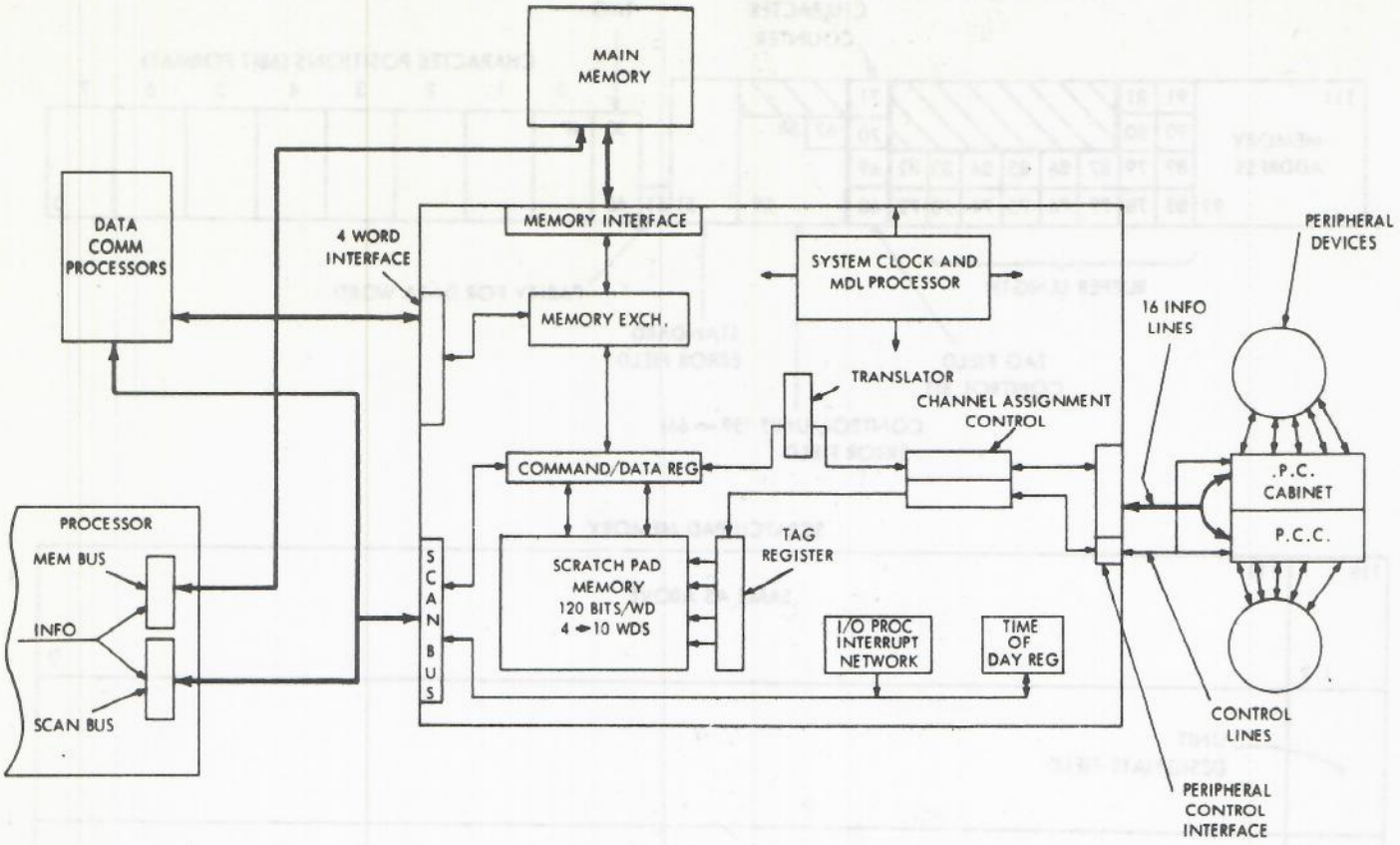


Figure 5-11. Input/Output Processor Block Diagram

Tag Register

The Tag Register (five flip flops per Scratch Pad Memory Word) associates a Scratch Pad Memory word with a specific I/O channel. This assignment is made when the I/O request is received from the Processor.

Memory Exchange

The Memory Exchange allows sharing of the Memory Interface lines between the I/O Processor and Data Communications Processors. The Memory Exchange has eight control lines, 20 address lines, 51 data lines and one parity line to the Memory interface.

Interrupt Network

The I/O Processor Interrupt Network informs the Processors of an interrupt condition in the I/O Processor. 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 (2.4μs) 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 with respect to system requirements.

Character Translator

Data flow between the I/O Processor and Peripheral devices is translated in one of three ways:

1. Direct (no translation in the I/O Processor)
2. Six-bit internal to BCL or vice versa
3. Eight-bit EBCDIC to BCL or vice versa

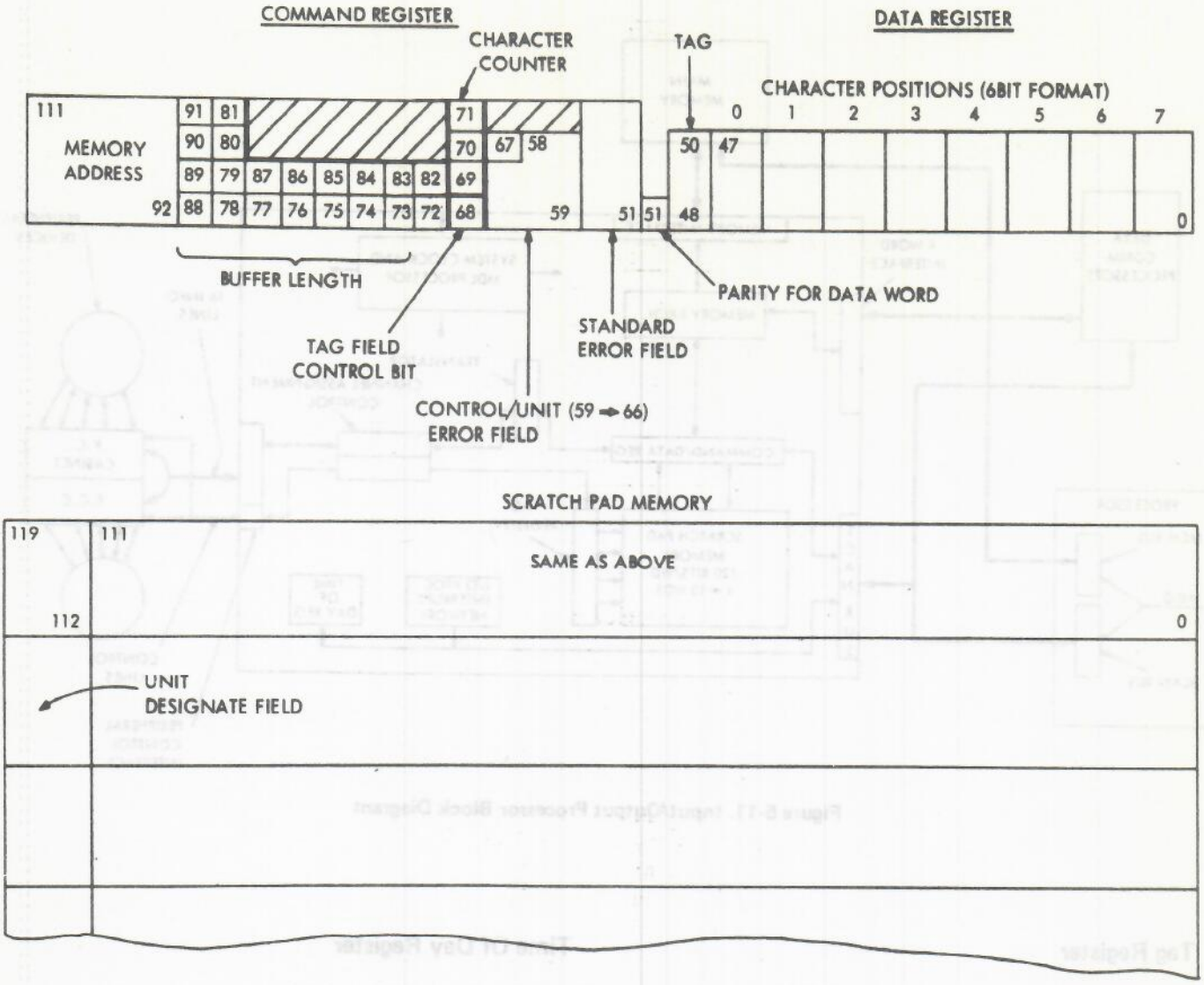


Figure 5-12. Command Data Register and Scratch Pad Memory

Peripheral Control Interface

The Peripheral Control Interface consists of 16 information 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:

1. BUSY/ – PCn
2. ARL – PCn (Access Request Level)
3. AGL – PCn (Access Granted Level)
4. CDL – PCn (Channel Designate Level)

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

Data Communications Interface

The Data Communications Interface consists of four, 20-wire cables sharing two word interfaces. Busses 2 and 4, 1 and 3 share the same memory request logic. Data Communications information is routed through the I/O Processor only to utilize the Memory Exchange of the I/O Processor.

System Clock Control and MDL Processor

The I/O Processor 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:

1. Processor:

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

2. I/O Processors: 5 megahertz, 25 nanosec pulse width
   1.67 megahertz, 25 nanosec pulse width
3. Memory:
   5 megahertz, 25 nanosec pulse width
4. Peripheral Control:
   1.67 megahertz 45, nanosec pulse width
5. Data Communications Processor:
   5 megahertz, 25 nanosec pulse width
6. MDL Processor:
   1.67 megahertz, 25 nanosec pulse width
   5 megahertz, 25 nanosec pulse width

MAINTENANCE DIAGNOSTIC PROCESSOR

The Maintenance Diagnostic Logic processor (MDL) is a main frame cabinet that consists of one I/O channel and has its own data processing capabilities. It is used for fault detection and isolation in the B 6700 Processors, I/O Processors, and 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 processes continuously in a loop under control of the display logic. It is used for indication and control of processor and I/O Processor flip flops.

DIAGNOSE MODE. In this mode the MDL Processor reads test cases from a tape unit, through 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 of cards may be read from the card reader: alpha, binary, and EBCDIC.

Alpha Card Read

Cards punched in the Alpha mode are decoded in the card reader from Card Code to six-bit BCL external code. The character is transmitted to the information register in the Card Reader Control in the Peripheral Control Cabinet. The information (one character) is held until the I/O Processor honors an access request and places the appropriate Scratch Pad Memory (SPM) word in its Command/Data register. I/O descriptor control bits 42 (translate) and 41 (six-or eight-bit) steer the character through the appropriate translator and place the character 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. When control bits 42 and 41 are equal to zero, this causes the translator, to be bypassed and causes a direct transfer of information into the Data Register. The information contained in one card column in strobed twice (once for each half of the card) and presented to the I/O Processor as two 6-bit characters. Tag read is not permitted in this mode.

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 I/O Processor 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 through 7 as shown in figure 5-13. The contents of the information register are decoded into EBCDIC code before being transferred over the information lines to the Data register. When six 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 permissible.

NOTE

Two other codes are available for use on the B 6700 system. They are ICT and BULL codes. Both are decoded by a special decoder in the Card Reader.

MEMORY AND INPUT/OUTPUT PROCESSOR CONTROLLER

The Memory Controller responds to 21 commands decoded from nine input lines. Figure 5-14 shows the four 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 set.

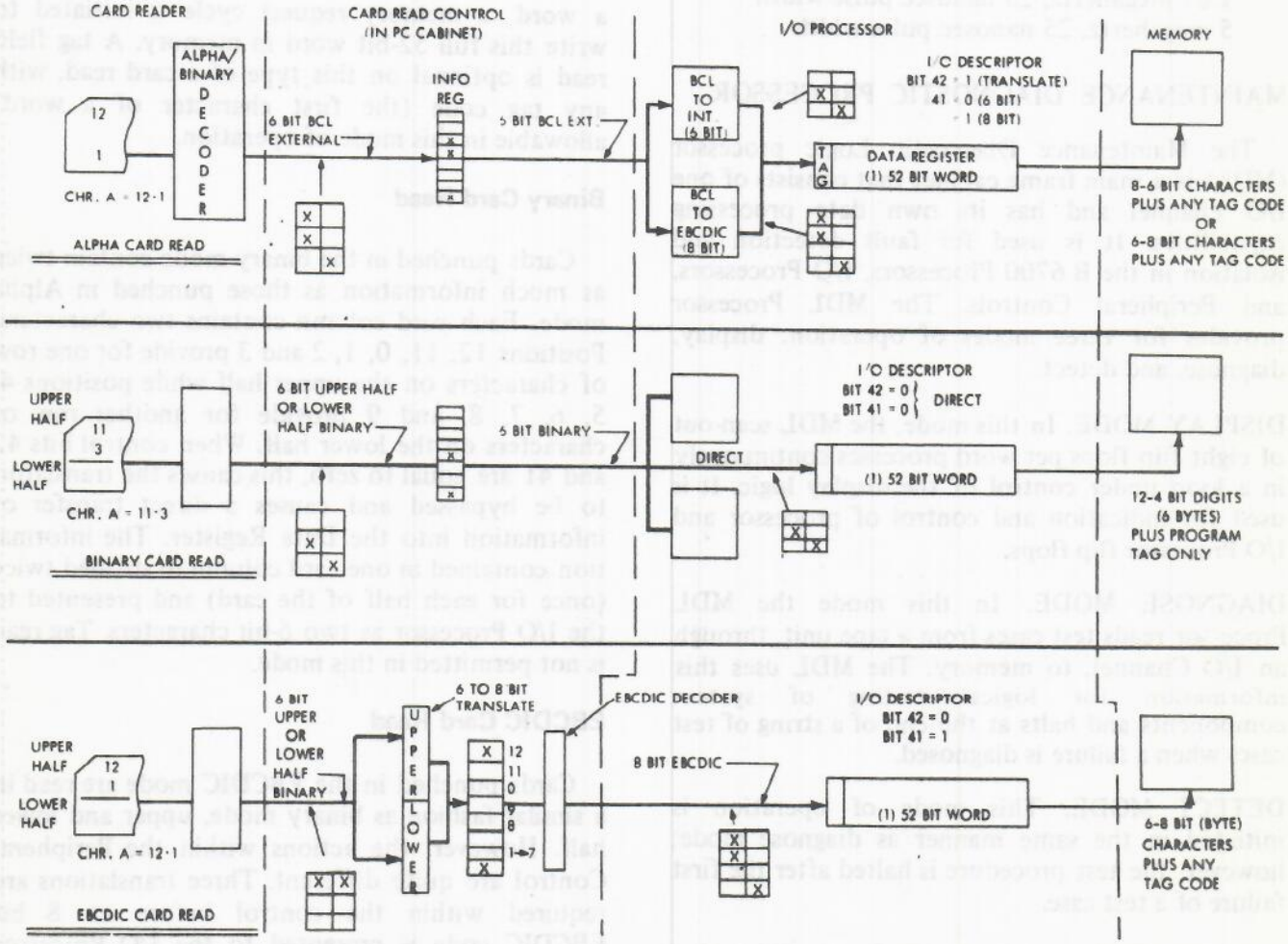
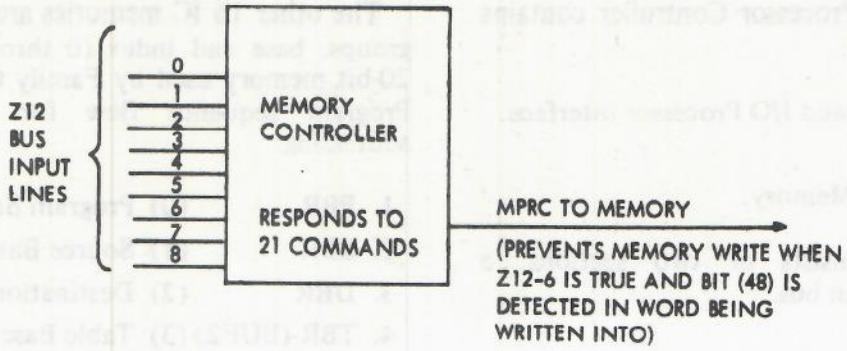


Figure 5-13. Data Information Flow



TYPE OF REQUESTING OPERATOR	MEMORY CONTROLLER FUNCTION	MEMORY CONTROLLER Z 12 LEVELS 8 7 6 5 4 3 2 1 0	PROCESSOR REGISTERS USED
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 and MVST operators exchange 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	A
		1 1 1	B
		1 1 1	C
		1 1 1	X
		1 1 1	Y
STORE OPERATORS	PROTECTED WRITE/READ ***	1 1	A
		1 1	B
		1 1	C
		1 1	X
		1 1	Y

NOTE

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

Figure 5-14. Memory Controller Decoding (cont)

The Memory/I/O Processor Controller contains the following sections:

1. B 6700 Memory and I/O Processor interface.
2. Address Adder.
3. 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 eight 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 through 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, one parity line and 11 control lines. It provides an asynchronous communication path between the B 6700 Processors and B 6700 I/O Processors or B 6700 Data Communications 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 Circuit (IC) 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 (DO 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 through 7). Each is a 20-bit memory used by Family Operator logic and Program sequence flow for base and index addressing:

1.	PBR	(0) Program Base
2.	SBR	(1) Source Base
3.	DBR	(2) Destination Base
4.	TBR (BUF2)	(3) Table Base
5.	S	(4) Top-Of-Stack Address
6.	SNR	(5) Stack Number
7.	PDR	(6) Program Dictionary
8.	TEMP	(7) Temporary Storage
9.	PIR	(0) Program Index
10.	SIR	(1) Source Index
11.	DIR	(2) Destination Index
12.	TIR (BUF3)	(3) Table Index
13.	LOSR	(4) Limit of Stack
14.	BOSR	(5) Base of Stack
15.	F	(6) Points to Top MSCW
16.	BUF	(7) Temporary Storage

MAIN MEMORY

Organization

Main memory in the B 6700 is organized so that any memory module can send information to, or receive information from all processors and all I/O Processors over any one of four information busses (see figure 5-15).

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; 200 nanoseconds later, the word is in the processor or I/O Processor 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.

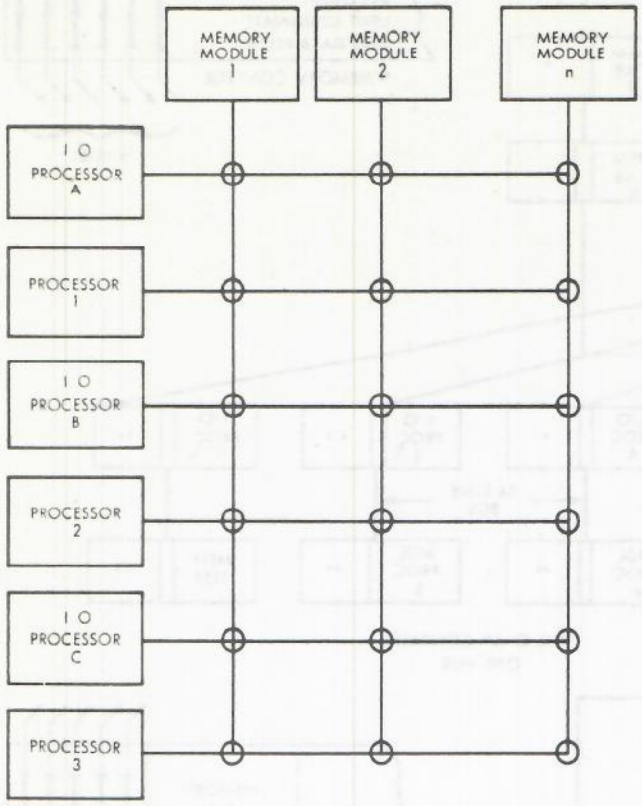


Figure 5-15. Memory Organization

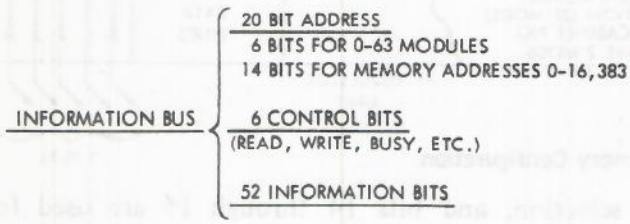


Figure 5-16. Information Transmission

Memory Protection

Memory protection prevents one program from affecting another by means of 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, during execution, cannot change program segments, data descriptors, or any program words or MCP tables.

Cabinet Configuration

The B 6700 Main Memory consists of one to 64 memory modules each of which contains 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. Also available are 65,536 word memory modules. (Refer to table 1-1.) The requestors are:

1. Processor #1, #2, or #3
2. I/O Processors A, B, or C
3. Memory Tester
4. MDL Processor A or B

Interface

The memory interface of the requesting unit 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. Note that the hubs within the requestors are all tied to the same address and information flow lines. Assume, for example, a Processor requesting access to memory module zero in cabinet zero. The Processor places the address and information on the busses. The address is decoded by all of the memory controls, but is only accepted by module zero 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.

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

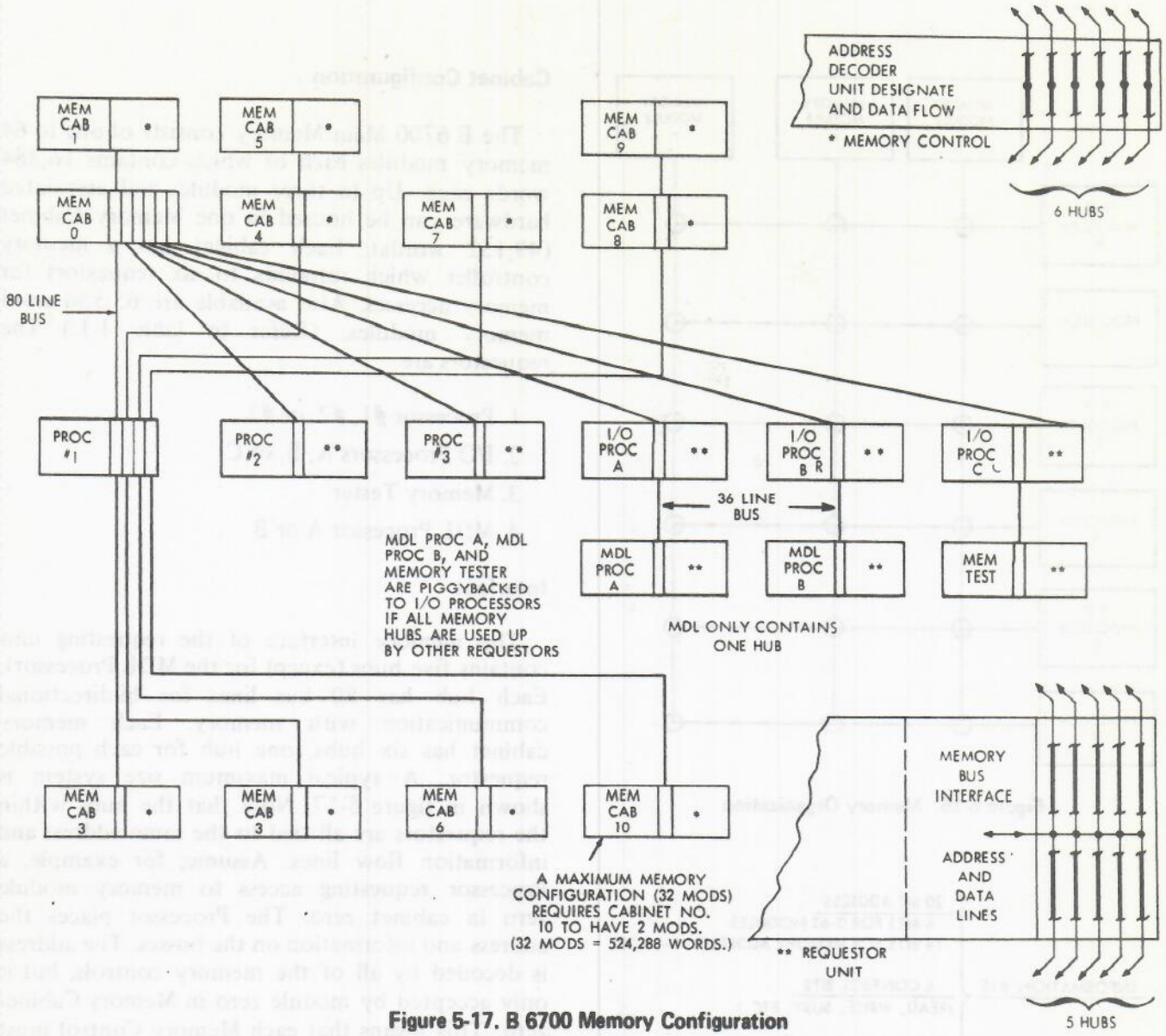


Figure 5-17. B 6700 Memory Configuration

Memory Registers

Each Memory module contains two core stacks, an 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, in which 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.

Memory Addressing

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

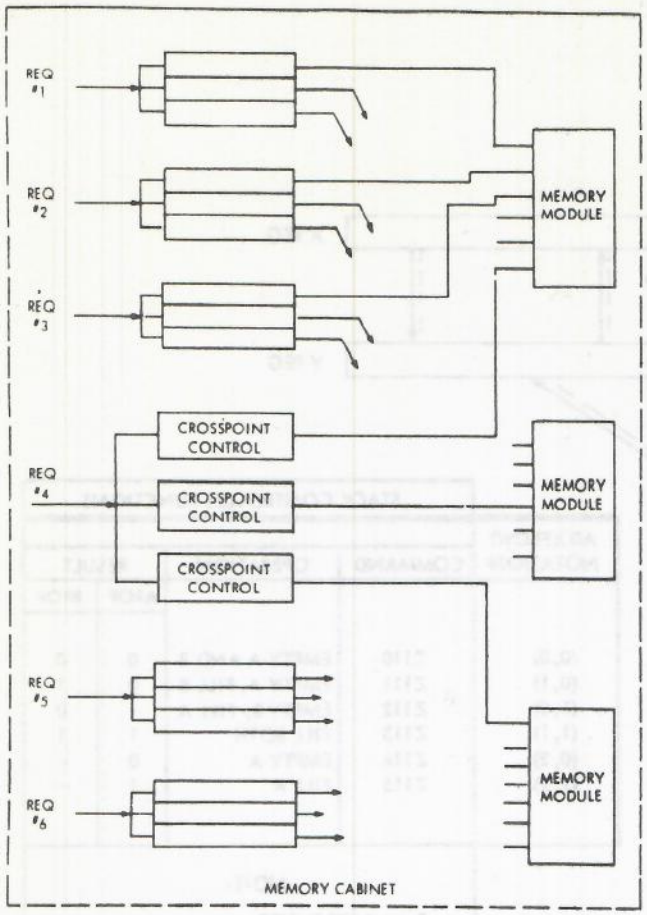


Figure 5-18. Memory Module Selection

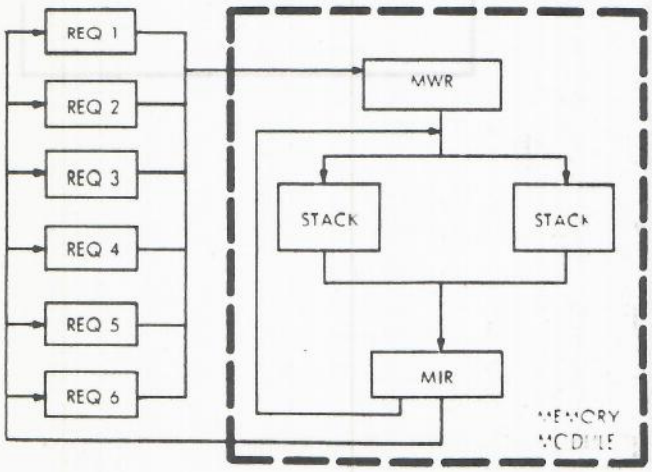


Figure 5-19. Memory Registers

module and word selection, 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

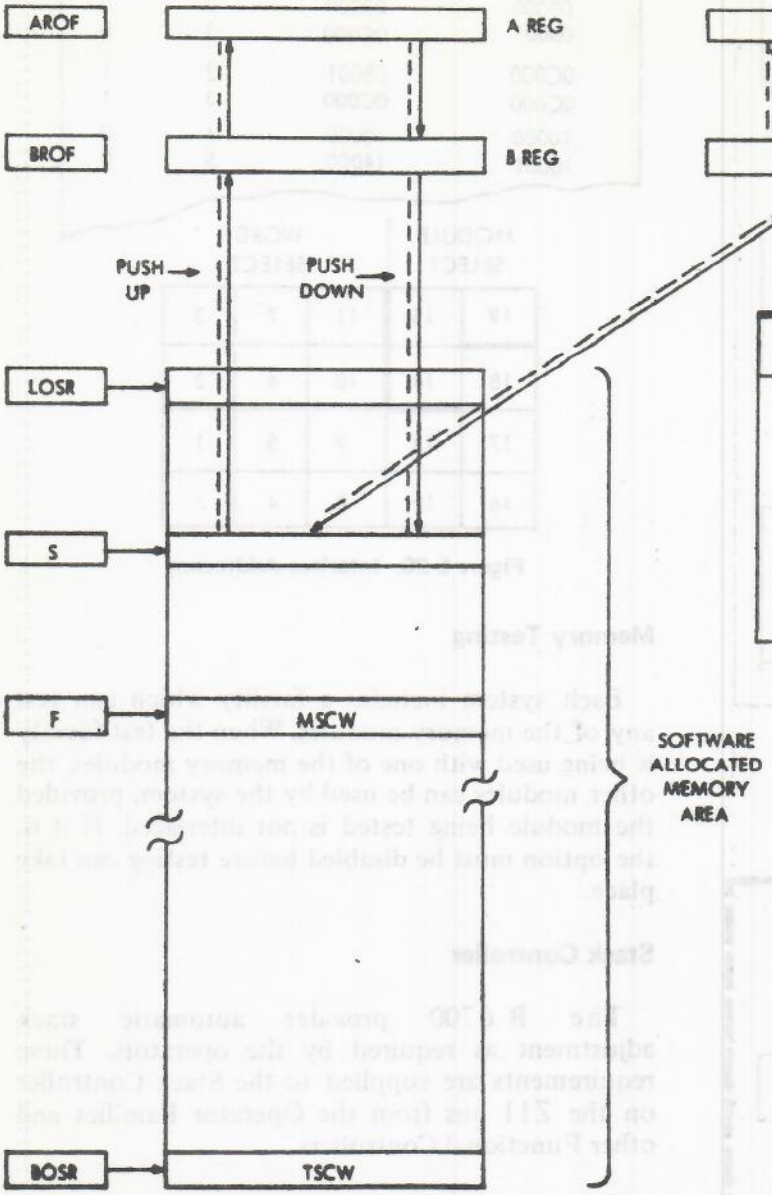
Each system includes a facility which can test any of the memory modules. When the test facility is being used with one of the memory modules, the other modules can be used by the system, provided the module being tested is not interlaced. If it is, the option must be disabled before testing can take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

Stack Controller

The B 6700 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 during both the 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.



ADJ(FLOW) NOTATION	STACK CONTROLLER FUNCTIONS			
	COMMAND	OPERATION	RESULT	
			AROF	BROF
(0,0)	Z110	EMPTY A AND B	0	0
(0,1)	Z111	EMPTY A, FILL B	0	1
(1,0)	Z112	EMPTY B, FILL A	1	0
(1,1)	Z113	FILL BOTH	1	1
(0,2)	Z114	EMPTY A	0	–
(1,2)	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.

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 eight-bit syllables. The first syllable of a program word is labeled syllable 0 and is formed by bits 47 through 40 (figure 6-1).

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).

Operation Types

Operations are grouped into three 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).

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 filled with 0's. 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. The operator is 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.

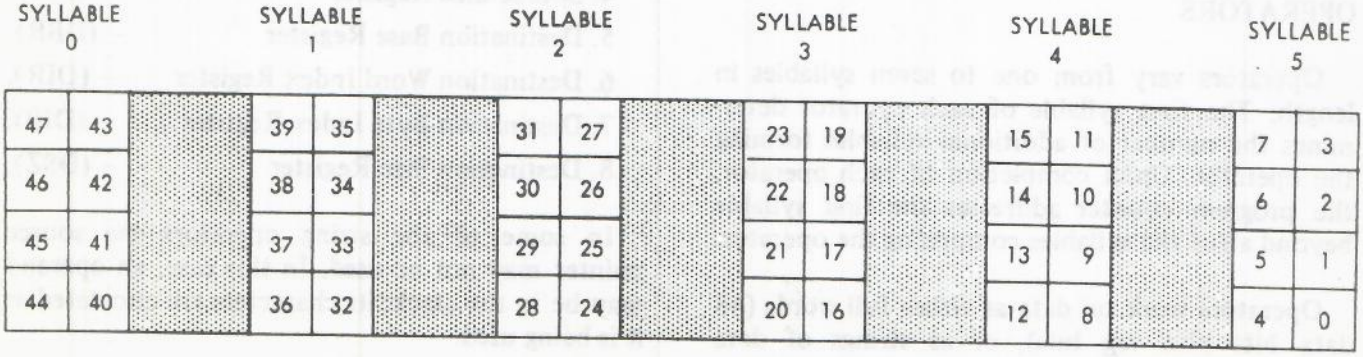


Figure 6-1. Program Word

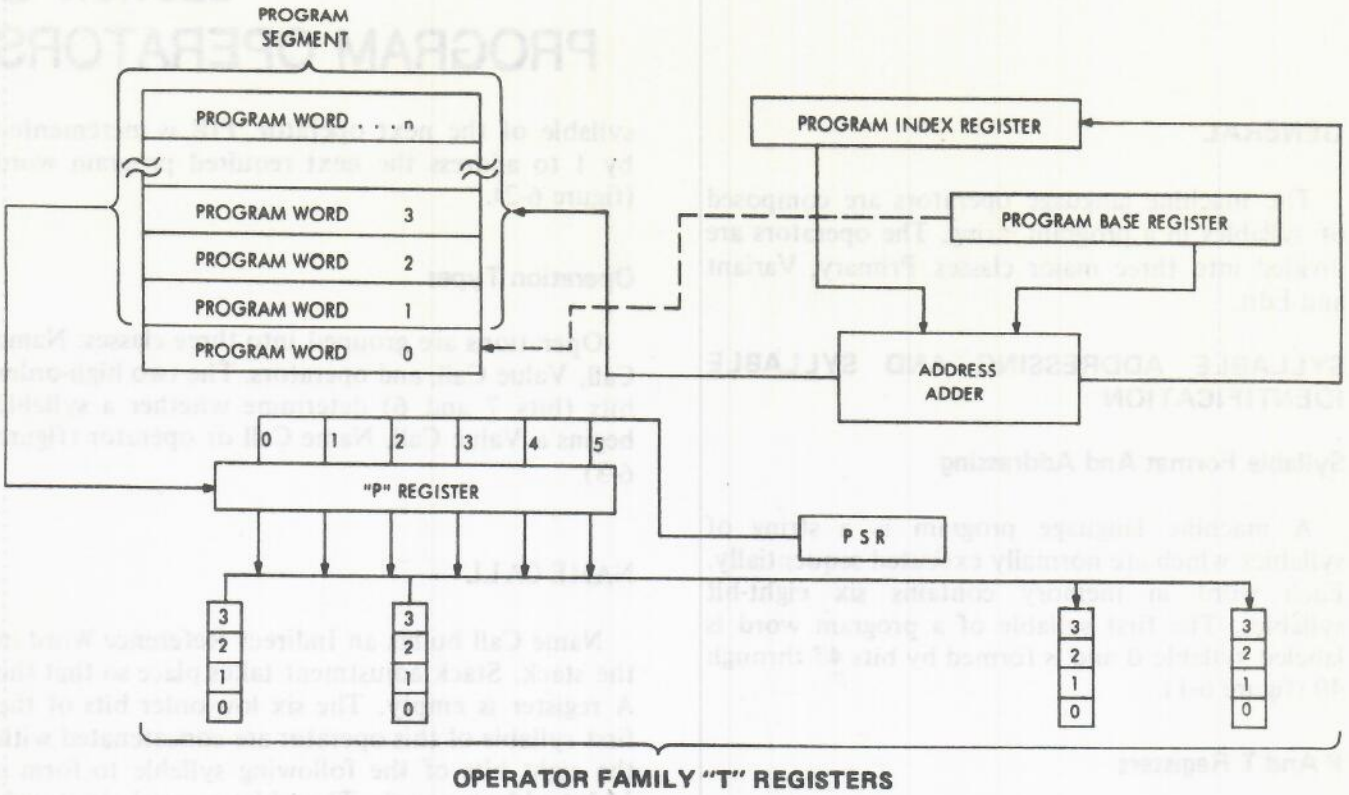


Figure 6-2. Program Word, Syllable Addressing

(BITS 7 and 6) Ident.	Syllable Type	No. 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 ⇒ 7	Performs the specified operation.

Figure 6-3. Syllable Decode Table

OPERATORS

Operators vary from one to seven 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 formatting data for input/output.

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

1. Source Base Register – (SBR).
2. Source Word Index Register – (SIR).
3. Source Byte Index Register – (SIB).
4. Source Size Register – (SSZ).
5. Destination Base Register – (DBR).
6. Destination Word Index Register – (DIR).
7. Destination Byte Index Register – (DIB).
8. 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, i.e., producing updated source and destination pointers and count. At completion of an operation the source and destination pointers are updated as follows:

1. If the source is an operand it remains in the stack.
2. 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.
3. 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 0, 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 0 and the other is not, then the size field of the non-zero descriptor is used.

If neither size field is equal to 0 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 0 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 0 through 1,048,576. The structure of the Word Data descriptor is illustrated in figure 6-4 and contains the following:

1. Bits 50:3, a tag of 101.
2. Bit 47:1, the presence bit, indicates the presence or absence of data in main memory. A 0 causes a presence bit interrupt whenever the descriptor is used by a processor to obtain non-present data. A 1 indicates that the data described is in main memory.
3. Bit 46:1, the copy bit. A 0 indicates that this is the original descriptor for the particular data area. A 1 indicates that this descriptor is a copy of the original descriptor (MOM).
4. Bit 45:1, the indexed bit. A 0 indicates that an indexing operation is required before the descriptor may be used to obtain data. A 1 indicates that indexing has already taken place and the index value is stored in bit positions 39:20 (Length/Index).
5. Bit 44:1, the segmented bit. A 0 indicates that the data is not segmented. A 1 indicates that the data is divided into segments.
6. Bit 43:1, the read-only bit. A 0 indicates that the data may be referenced for reading or writing. A 1 indicates that the area cannot be used for data storage.
7. Bits 42:2, a 0 indicates a word data descriptor.
8. Bit 40:1, the double-precision bit. A 0 indicates single-precision operands, a 1 indicates double-precision operands.
9. Bits 39:20, contain either the length of the memory area (If bit 45 = 0) or an index value (if bit 45 = 1). If bit 45 equals 0, the descriptor has not been indexed. This field is used for size checking during the indexing operation. If bit 45 equals 1, 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.

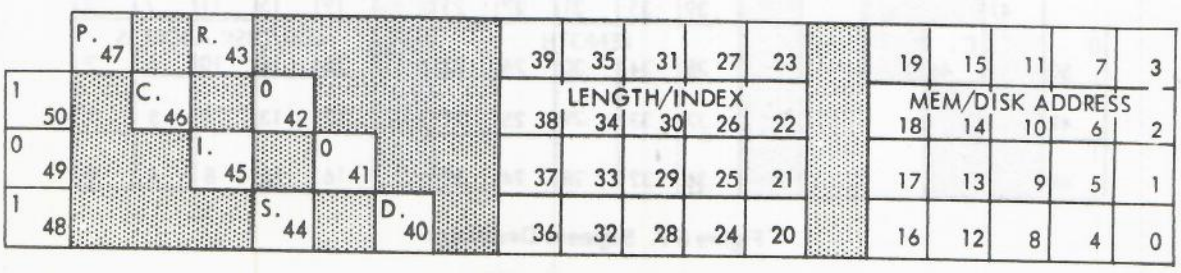


Figure 6-4. Word Data Descriptor

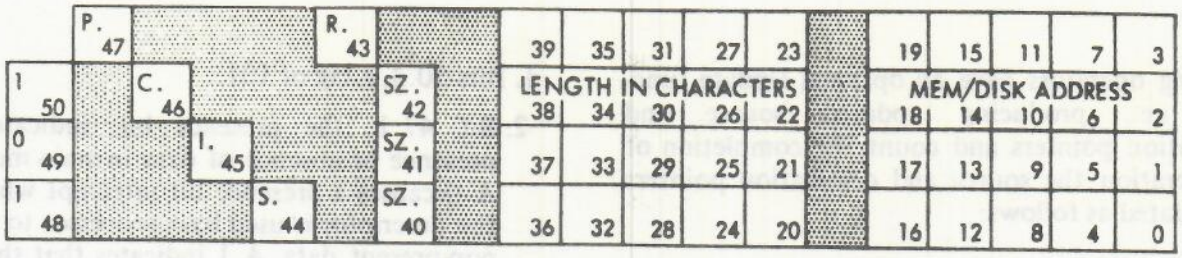


Figure 6-5. String Descriptor (Non-indexed)

10. Bits 19:20, contain either a main memory or disk address. If the presence bit (bit 47) equals 1, this field contains the memory address of data. If the presence bit equals 0 and the copy bit (bit 46) equals 0, this field contains the disk address of the data. If the presence bit equals 0 and the copy bit equals 1, 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 Descriptors 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:

1. Bits 50:3, a tag of 101.
2. Bit 47:1 the presence bit. A 0 causes a presence bit interrupt if the descriptor is used to access data. A 1 indicates the data is present in main memory.
3. Bit 46:1, the copy bit. A 0 indicates that this is the original descriptor for the particular data area. A 1 indicates that this descriptor is a copy of the original descriptor.
4. Bit 45:1, the indexed bit. A 0 indicates indexing is required. A 1 indicates that indexing has taken place and the word and character index are in the length/index field (see figure 6-6).
5. Bit 44:1, the segmented bit. A 0 indicates that the data area is not segmented. A 1 indicates that the date is segmented.
6. Bit 43:1, the read only bit. A 0 indicates that the data may be referenced for reading or writing. A 1 indicates that the data can be read only.
7. Bits 42:3, character size field. 100 indicates 8-bit bytes, 011 indicates 6-bit characters, and 010 indicates 4-bit digits.
8. Bits 39:20, contain either the length of the memory area (bit 45=0) or an index value (bit 45=1). When bit 45 equals 0, this field contains the length of the area in digits, characters or bytes. During indexing operations this field is used for size checking. When bit 45 is equal to 1, bits 39:4 contain a byte index and bits 35:16 contain a word index as illustrated in figure 6-6.

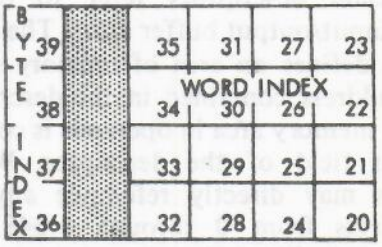


Figure 6-6. Byte/Word Index Field

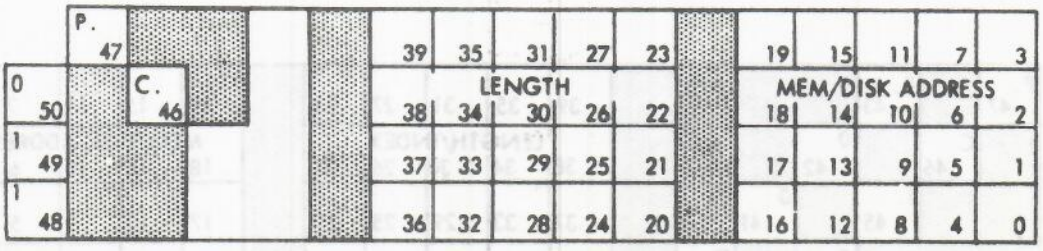


Figure 6-7. Segment Descriptor

9. Bits 19:20, contain either a main memory or a disk address. If the presence bit (bit 47) is 1, the field contains a memory address of the data. If both the presence bit and the copy bit (bit 46) are equal to 0, the field contains the disk address of the non-present data. If the presence bit is 0 and the copy bit is 1, 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:

1. Bits 50:3, a tag of 011.
2. Bit 47:1, the presence bit. A 0 indicates that the segment is absent from main memory.
3. Bit 46:1, the copy bit. A 0 bit indicates that this is the original segment descriptor. A 1 indicates that this is a copy of the original segment descriptor.
4. Bit 45:1, unused.
5. Bits 44:5, unused.

NOTE

Unused bits may be either 0 or 1.

6. Bits 39:20, specify the length of the program segment in words.
7. Bits 19:20 contain either the main memory address or the disk file address. If the presence bit (bit 47 equals 1, the field contains the main memory address of the program segment. If both the presence bit and the copy bit (bit 46) equal 0, the field contains the disk address of the non-present program segment. If the presence bit equals 0 and the copy bit equals 1, the field contains the absolute memory address of the original program segment descriptor.

MARK STACK CONTROL WORD

The Mark Stack Control Word (MSCW), together with the Return Control Word (RCW), 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:

1. Bits 50:3, a tag of 011.
2. Bit 47:1, the different-stack bit. A 0 indicates that the stack-number field refers to the current stack. A 1 indicates that the stack-number field refers to a different stack.
3. Bit 46:1, the environment bit. A 0 indicates an inactive MSCW, generated directly by the Mark Stack operator. The procedure entry has not been performed. A 1 denotes an active MSCW generated upon entry into a procedure, at which time the environment fields (stack number, displacement, value, and LL fields) are stored into the MSCW.
4. Bits 45:10, the stack-number field, contain the number of the stack from which the PCW was obtained at procedure-entry.
5. Bits 35:16, the displacement field, which, when added to the stack base address, locate the MSCW of the prior lexicographic level.
6. Bit 19:1, the value bit. A 0 indicates that the MSCW was generated during any operation that will be restarted from the beginning. A 1 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).
7. Bits 18:5, the LL field denote the lexicographical level at which the program will run when the procedure is entered.
8. Bits 13:14, denote the stack history. This field is used to locate in the stack, the preceding MSCW (i.e., the previous "F" register setting).

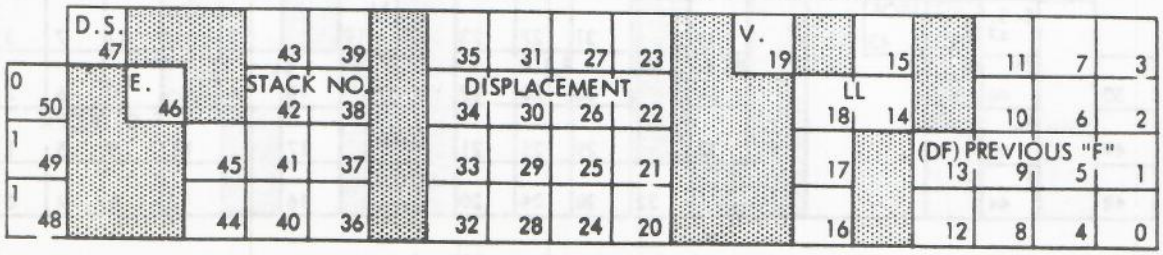


Figure 6-8. Mark Stack Control Word

PROGRAM CONTROL WORD

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

1. Bits 50:3, a tag of 111.
2. Bit 47:1, unused.
3. Bit 46:1, unused.
4. Bits 45:10, stack number containing the PCW.
5. Bits 35:3, define the program syllable within the word located by PIR.
6. Bits 32:13, an index to the Program Base Register.
7. Bit 19:1, normal state (0) or control state (1).
8. Bits 18:5, the level of the procedure being entered.
9. Bits 13:14, the segment descriptor index. Bits 12 through 0 specify the value to be added to the address located by either D-register 0 or 1. When bit 13 equals 0, D-register 0 is selected; when bit 13 equals 1, D-register 1 is selected.

RETURN CONTROL WORD

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

1. Bits 50:3, a tag of 011.
2. Bit 47:1, External Sign flip flop.
3. Bit 46:1, Overflow flip flop.
4. Bit 45:1, True/False flip flop.
5. Bit 44:1, Float flip flop.
6. Bit 43:1 is TFOF, True/False flip flop occupied flip flop.
7. Bits 35:3, the program syllable of the operator to be executed after return from the subroutine.
8. Bits 32:13, the PIR setting of the operator to be executed next in the calling routine.
9. Bit 19:1, a normal state (0) or control state (1) procedure.
10. Bits 18:5, the level of the calling procedure when the RCW was generated (at procedure entry).
11. Bits 13:14, the segment descriptor index. Bits 12 through 0 specify the value to be added to the address located by either D-register 0 or 1. When bit 13 = 0, D-register is selected; when bit 13 = 1, D register 1 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). References are relative to the D register specified by the address couple. The bit format of the IRW is shown in figure 6-11.

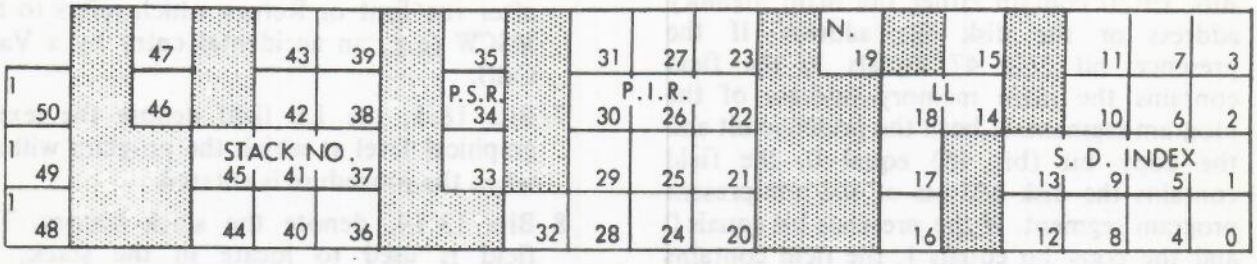


Figure 6-9. Program Control Word

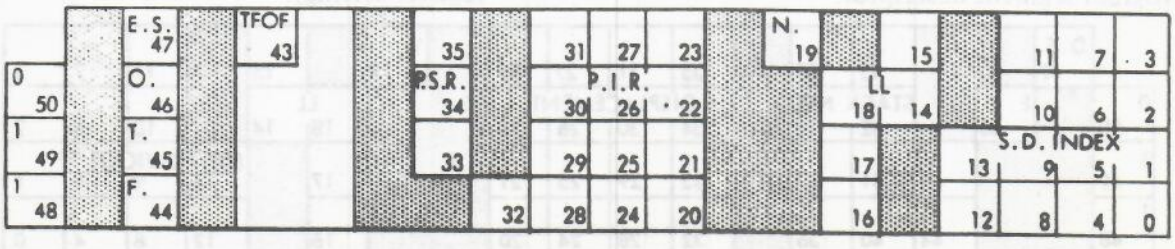


Figure 6-10. Return Control Word

STUFFED INDIRECT REFERENCE WORD

Reference to variables outside the current environment is accomplished by a Stuffed Indirect Reference Word. 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 index to the variable relative to the MSCW. The absolute memory location of the variable is formed by adding the contents of DISP and index to the base address of the referenced stack from the stack descriptor. The contents of the SIRW (with the exception of index) are dynamic and are accumulated as the program is executed. The stack number and DISP fields are entered into the SIRW by the Stuff Environment (STFF) operator. The bit format of the SIRW is shown in figure 6-12.

1. Bits 50:3, tag of 001.
2. Bit 47:1, unused.
3. Bit 46:1, the environment bit. A 1 indicates a Stuffed IRW. A 0 indicates an IRW.
4. Bits 45:10, stack number. When bit 46 equals 1, it specifies the number of the stack containing the address.
5. Bits 45:26, unused, when bit 46 equals 0.
6. Bits 35:16, displacement field. When bit 46 equals 1, this value added to the stack base address locates a Mark Stack Control Word.
7. Bits 19:6, unused.
8. Bits 12:13, index field. When bit 46 equals 1, the memory address is computed by adding the index field to the address of the MSCW specified by the stack number and displacement fields. Bit 13 is always 0.
9. Bits 13:14, when bit 46 equals 0, are divided into two functional fields (figure 6-13). Each field is variable in length. The first subfield, designated LL, selects one of the D registers. The second subfield is an index value which is added to the contents of the selected D register to form an absolute address. The lengths of the subfields 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

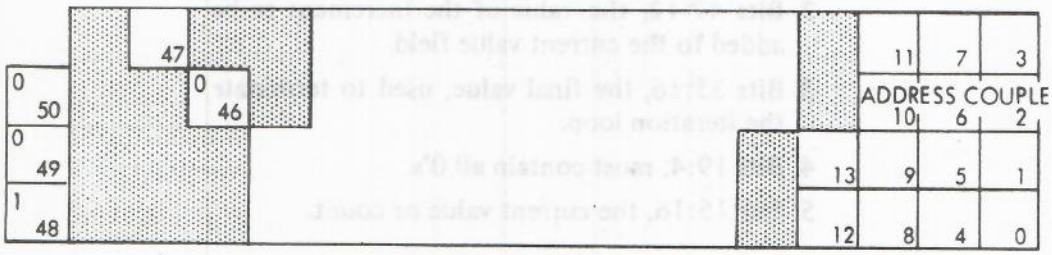


Figure 6-11. Indirect Reference Word

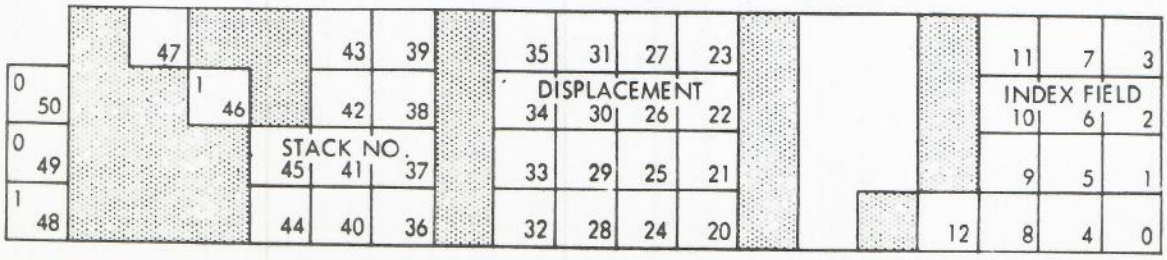


Figure 6-12. Stuffed Indirect Reference Word

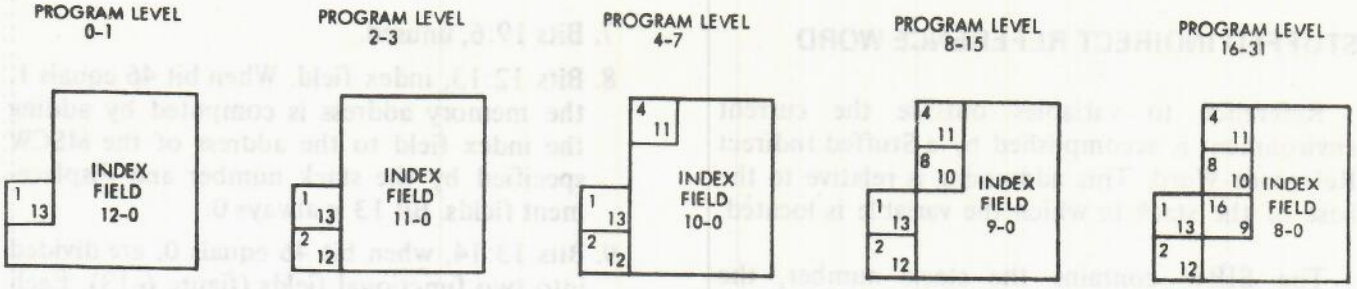


Figure 6-13. Program Level Bit Assignment

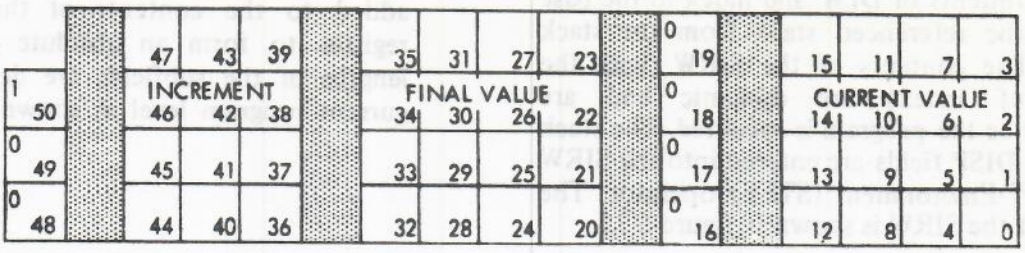


Figure 6-14. Step Index Word

NOTE

The bit order of the LL field is inverted.

STEP INDEX WORD

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

1. Bits 50:3, a tag of 100.
2. Bits 47:12, the value of the increment to be added to the current value field.
3. Bits 35:16, the final value, used to terminate the iteration loop.
4. Bits 19:4, must contain all 0's.
5. Bits 15:16, the current value or count.

SECTION 7
PRIMARY MODE OPERATORS

GENERAL

This section defines the functions of the primary operators. In each case, the name of the operator, corresponding mnemonic, and hexadecimal code are 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 the single-precision process invoked if both operands are of the single-precision type and the 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 is in the A register; the second word of the first operand occupies the X register. The first word of the second operand resides in 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 which use two integer operands yield an integer result if no overflow occurs. If one or both operands are non-integer, or if the result generates an overflow, the result is non-integer.

When an operator has been entered, 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 by 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-extension register containing the single-precision operand is set to all 0's. 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 1.

If the exponents are equal but the manitssa signs are unequal, the difference of the mantissas plus the appropriate sign are 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 0, the B register is set to all 0's.

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 7's 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 0, the B register is set to all 0's. If the A register mantissa is equal to 0, 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 remains 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 0, the B register is set to all 0's. If the mantissa of the A register is 0, 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 all 0's. 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 a 0 exponent for single precision and the exponent set to 13 for double precision. 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 stays in the B register.

If the mantissa of the B register is 0, the B register is set to all 0's. If the mantissa of the A register is 0, 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; 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 remains in the B register.

If the operand in the B register cannot 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.

Intergerize, 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 cannot be integerized, i.e., the exponent is greater than the number of the leading zeros in the operand, the integer overflow interrupt is set and the operation is terminated.

The operand is rounded, if necessary, by adding 1 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 normalized and set to a single-precision operand; or in the case of a data descriptor, the double-precision bit is set to 0.

If the word in the B register is a non-indexed, double-precision data descriptor, the double-precision bit is cleared to 0 and the length field multiplied by 2.

If the double-precision operand in the B register has an exponent greater than +63 after normalization, the exponent overflow interrupt is set. If the exponent is less than -63 after normalization, the exponent underflow interrupt 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 neither an operand nor a Data Descriptor, 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 interrupt is set. If the exponent is less than -63, the exponent underflow interrupt 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, the operand in the B register is rounded from the Y register, and the Y register is set to all 0's.

If a carry is developed during the rounding operation, the operand is adjusted and the new exponent is checked in the manner discussed in the preceding paragraph.

If the operand is a single-precision operand, it is normalized and no rounding occurs.

Set To Double-Precision (XTND) CE

The word in the B register is set to a double-precision operand and the Y register is set to all 0's. If a single-precision data descriptor is present in the B register, the double precision bit is set to 1.

If the word in the B register is a data descriptor with both the index bit and double-precision bit 0, the double-precision bit is set to 1 and the length field is divided by 2.

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 0's.

If the word in the B register is neither an operand nor a Data Descriptor, the invalid operand interrupt is set and the operation terminated.

LOGICAL OPERATORS

If only one of the operands LAND, LOR, or LEQV is in double-precision form, the other operand is considered as double-precision with the least significant 13 octades equal to all 0's.

Logical And (LAND) 90

Each bit of the B operand, except for the tag bits, is set to 1 where a 1 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 0. The tag of the B operand is not disturbed, unless the tag of the A operand specifies double-precision; in that case, the B operand tag is set to double-precision.

Logical Or (LOR) 91

Each bit position of the B operand except for the tag bits, is set to 1 if the corresponding bit position in either the A operand or the B operand is 1, otherwise, the bit is set to 0. 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

Each bit in the A operand is complemented except for the tag bits, which remain unchanged.

Logical Equivalence (LEQV) 93

Each bit of the B operand is set to 1, except for 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 for 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 that case, the B-register tag is set to double-precision.

RELATIONAL OPERATORS

The relational operators perform an algebraic comparison on the operands in the A register and the B register. The single precision result is left in the B register and the B register is marked full. The result is an operand in integer form with the value 1 if the relationship has been met or an operand with all information bits set to 0 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 0 set to 1 and all other information bits set to 0 is stored in the B register. Otherwise, a single-precision operand with all information bits set to 0 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 1. Otherwise, all bits in the B register are set to 0.

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 1. Otherwise, all bits in the B register are set to 0.

Equal (EQUL) 8C

If the operands in the B and A registers are algebraically equal, the B register is set to an integer form 1. Otherwise, all bits in the B register are set to 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 1. Otherwise, all bits in the B register are set to 0.

Less Than (LESS) 88

If the operand in the B register is algebraically less than the operand in the A register, the B register is set to an integer form 1. Otherwise, all the bits in the B register are set to zero.

Not Equal (NEQL) 8D

If the operand in the B register is not algebraically equal to the operand in the A register, the B register is set to an integer form 1. Otherwise, all the bits in the B register are cleared to 0.

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) AO

If the low order bit of the A register is 0, the Program Index Register (PIR) and Program Syllable Register (PSR) 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 PSR and the next 13 low order bits are placed in the PIR. 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 PIR and PSR 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 (BRFL), above.

Branch Unconditional (BRUN) A2

The PIR and PSR 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 (BRFL).

Dynamic Branch False (DBFL) A8

If the low order bit of the B register is 0 and the word in the A register is a Program Control Word (PCW) or an indirect reference to one, a branch is made to the specified syllable of that program segment.

If the low order bit of the B register is 0 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 0, PSR is set to 0, otherwise PSR is set to 011. The next higher order 20 bits are placed in the PIR. 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 1 and the word in the A register is a PCW, or an indirect reference to one, a branch is made to the specified syllable of the program segment.

If the low order bit of the B register is 1 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 (DBFL).

Dynamic Branch Unconditional (DBUN) AA

If the word in the A register is a PCW or an indirect reference to one, a branch is made 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 (DBFL).

Step And Branch (STBR) A4

The increment field of the step-index word (SIW) 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, the PIR and PSR are set from the next two syllables in the program string. Otherwise, the PIR and the PSR are advanced three syllables. The SIW 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

all 0's, the PIR and 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 the Variant and Edit modes.

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 the Variant and Edit modes.

Invalid Operator (NVLD) FF

This operator sets the invalid operand interrupt. This operator is also valid in Variant and Edit modes.

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 (IRW) or a Data Descriptor. If the A register contains an operand, a hardware interchange takes place so that the operand is transferred to 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 for double-precision) is stored. After the operand is stored, the A register and B register are marked empty and the operation is complete.

If the word addressed by the IRW is a Program Control Word, accidental procedure entry occurs. The spontaneously created Return Control Word (RCW) causes the Store Destructive (STOD) operator to be re-executed upon return from the procedure.

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 0, the presence bit interrupt is set. After the information has been made present, the operation is restarted.

Store Non-Destructive (STON) B9

This operator functions in virtually the same way as the STOD operator, however, at the completion of this operator, the BROF remains set, and the operand is retained in the B register.

Overwrite Destructive (OVRD) BA

This operator functions in the same way as the STOD operator, except that the OVRD operator overrides memory protection checks.

Overwrite Non-Destructive (OVRN) BB

This operator functions in the same way as the STON operator, except that the OVRN operator 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 Top-of-Stack 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(s) 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 all 0's 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 1, leaving all other bits set to 0. 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 bits 7:8 of the A register. The rest of the A register is set to all 0's. The A register is marked as full and the PSR 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 bits 15:16 of the A register. The rest of the register is set to all 0's. 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 all 0's. The A register is marked full, and the 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 intervening syllables are not executed.

Make Program Control Word (MPCW) BF

This operator performs a "Lit Call 48 Bits" (LT48) 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 1).

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 IRW 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 represents an array which is segmented, the index is partitioned into two portions by dividing it by the proper divisor which is determined by the type of data referenced by the descriptor, (double-precision word-128, single-precision word-256, four-bit digit-3072, six-bit character-2048, or eight-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 1, the index value in the B register is doubled. The balance of the operation is as described in the first paragraph of the description of this operator (INDX).

Index And Load Name (NXLN) A5

This operator performs an index operation; after the word in the A register has been indexed, the Data Descriptor pointed to by this word is brought into the A register. The copy bit (bit 46) of the Data Descriptor is set to 1 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 indexed 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 1, the invalid operand interrupt is set and the operation is terminated.

Index And Load Value (NXLV) AD

This operator performs an index operation; after the word in the A register has been 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 IRW 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, the latter is marked as a copy (copy bit [bit 46] is set to 1), 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 the time that the arithmetic operations are performed. 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 flip flop is set to 1.

Dynamic Scale Left (DSLF) C1

This operator performs virtually the same operation as the Scale Left (SCLF) operator; however, the scale factor is taken from the A register rather than from the program syllable following the operation syllable. The operand in the A register is integerized before scaling takes place.

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 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 (four-bit digits) and is left-justified. The 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 virtually the same operation as the Scale Right Save (SCRS) operator; however, the scale factor is obtained from the A register rather than from 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 using 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 it is used 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 0 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 0 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 1, it is set to 0; if it is a 0, the bit is set to 1.

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 is L.

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 exists 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 0's. The A register is marked empty and the B register is marked full.

This 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 0 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 0 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 the 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 0 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 0 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.

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, except that it overrides the memory protection checks.

Transfer Words, Overwrite Update (TWOU) DC

This operator performs a Transfer Words, Update operation, except that it overrides 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 comparison fails.

The first item in the stack is used as the delimiter. The second item in the stack, bits 19:20, 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 source and destination strings are checked for memory protection. The source character is then compared with the delimiter. The result of the comparison is set in the True/False flip flop (TFFF). If the condition is met the TFFF is set to 1, if it is not met it is set to 0.

If the number of characters transferred was equal to the repeat field the TFFF will remain set to 1. The A and B registers are marked empty and the operation is complete.

If the comparison fails, the TFFF is set to 0.

If the first operand in the stack is not a single-precision 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 comparison.

Transfer While Greater Or Equal, Update (TGEU) 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 operation.

Transfer While Equal, Destructive (TEQD) E4

This operator performs a Transfer While operation with the relation used in the comparison being equal.

Transfer While Equal, Update (TEQU) EC

This operator performs a Transfer While Equal operation. The source pointer, the destination pointer and count are updated at the conclusion of the operation.

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 operation.

Transfer While Less, Destructive (TLSD) EO

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 operation.

Transfer While Not Equal, Destructive (TNED) E5

This operator performs a Transfer While operation, using 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 operation.

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 and the destination pointer are updated at the conclusion of the operation.

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.

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 1, the segmented array interrupt is set and the operation is terminated.

COMPARE OPERATORS

The Compare Operators perform the specified comparison of two strings of data. The TFFF is conditioned by the results of the comparison.

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 TFFF is set to 1. If not, the TFFF is set to 0.

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.

Thus 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 TFFF 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 0, the TFFF 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 operation.

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 ≤ 0, the TFFF is set to 0.

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 operation.

Compare Characters Equal, Destructive (CEQD) F4

This operator performs the Compare Characters operation using the Equal comparison. If the repeat count ≤ 0, then TFFF is set to 1.

Compare Characters Equal, Update (CEQU) FC

This operator performs a Compare Characters Equal operation. The source pointer and destination pointer are updated at the conclusion of the operation.

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 ≤ 0, then TFFF is set to 0.

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 operation.

Compare Characters Less, Destructive (CLSD) F0

This operator performs the Compare Characters operation using the Less than comparison. If the repeat count ≤ 0, the TFFF is set to 0.

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 operation.

Compare Characters Not Equal, Destructive (CNED) F5

This operator performs the Compare Characters operation using the Not equal relation. If the repeat count ≤ 0, then TFFF is set to 0.

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 operation.

EDIT OPERATORS

Table Enter Edit, Destructive (TEED) D0

This operator is used to control edit micro-instuctions. These edit micro-instructions are contained in memory as a table and not as part of the normal program string. When this operator is entered, 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 single-precision 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 single-precision 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.

Table Enter Edit, Update (TEEU) D8

This operator performs a Table Enter Edit operation and updates the source pointer and destination pointer at the completion of the operation.

Execute Single Micro, Destructive (EXSD) D2

This operator performs the same function as the Table Enter Edit operator with the following exceptions: There is only one micro-operator and it follows this syllable. The first item in the stack is a single-precision operand that defines the length field.

Execute Single Micro, Update (EXSU) DA

This operator performs the same functions as an Execute Single Micro operator, except that it updates the source pointer and destination pointer at the completion of the operation.

Execute Single Micro, Single Pointer Update (EXPU) DD

This operator performs the same functions as an Execute Single Micro Update operator, except that one pointer is used as both source and destination pointer. The destination pointer is updated at the completion of the operation.

PACK OPERATORS

Pack, Destructive (PACD) D1

This operator packs data, addressed by the source pointer, into the top of the stack in four-bit (digit) format. The TFFF is set to 1 if the source data is negative. A negative number for an eight-bit (byte) format has a zone bit configuration of 1101 in the least significant byte. The six-bit (BCL) format for a negative number has a configuration of 10 in the least significant character position. The four-bit (digit) format has a 1101 configuration in the most-significant digit position. Data is right-justified as it is placed in the top-of-stack.

The operand in the top-of-the-stack is used as the length field. The second item is the source pointer. The operation then continues until the number of digits specified by the length/repeat field have been packed.

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 comprised of eight-bit bytes.

If the source data has the memory protect bit (bit 48) set to 1, 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 operation.

INPUT CONVERT OPERATORS

Input Convert, Destructive (ICVD) CA

This operator converts either six-bit BCL code, eight-bit EBCDIC or four-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 four-bit digit operand is converted to an operand of the equivalent binary value.

The sign of the converted operand is then set from the TFFF. If the converted operand is a single-precision operand, the TFFF is then set to 1. If the converted operand is a double-precision operand, the TFFF is set to 0.

At the completion of the operation the B register is marked full. The tag field is set to indicate either a single- or a double-precision operand.

If after being integerized, the item in the top-of-stack 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 operation.

Read True False Flip Flop (RTFF) DE

This operator places the status of the TFFF into the low-order bit position of the A register. The rest of the A register is set to all 0's. The A register is marked full at completion of this operation.

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

This operation 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 all 0's, marks the register full, and sets the Overflow flip flop to 0.

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, PCW, or an IRW.

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 1, the other half of the double-precision operand is brought to the X register.

If the word accessed is a non-indexed word data descriptor, the word is indexed using the second word in the stack for the index value. The word addressed by the non-indexed Data Descriptor is brought to the top-of-the-stack. If the double-processor bit (40) in the Data Descriptor is equal to 1, the other half of the double-processor operand is brought to the X register.

If the word accessed by the Data Descriptor is another indexed Data Descriptor, the word addressed by the Data Descriptor is brought to the top-of-the-stack, and one of the two above paragraphs is repeated.

If a Data Descriptor does not address an operand, SIRW, or a word descriptor or indexed string 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 (IRW) 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 IRW is another IRW, the operation continues as described above.

If the word accessed by the IRW is an indexed or non-indexed Data Descriptor, the operator proceeds as described above for Data Descriptors.

If the word accessed by the IRW is a Program Control Word (PCW), an accidental entry into the subroutine addressed by the PCW is initiated. A Mark Stack Control Word (MSCW) and a Return Control Word (RCW) are placed in the stack and an entry is made into the program. Upon completion of the program, a return operator will re-enter the flow value call at the label IRW, (figure 7-1).

Name Call (NAMC) 40 ⇒ 7F

This operator builds an IRW in the A register. The address couple is formed by concatenating the six low-order bits of the first syllable and the eight bits of the following syllable. The A register is marked full and the operation is complete.

Exit Operator (EXIT) A3

This operator returns to a calling procedure from a called procedure resetting all control registers from the RCW and the MSCW. The Exit operator does not return a value to the calling

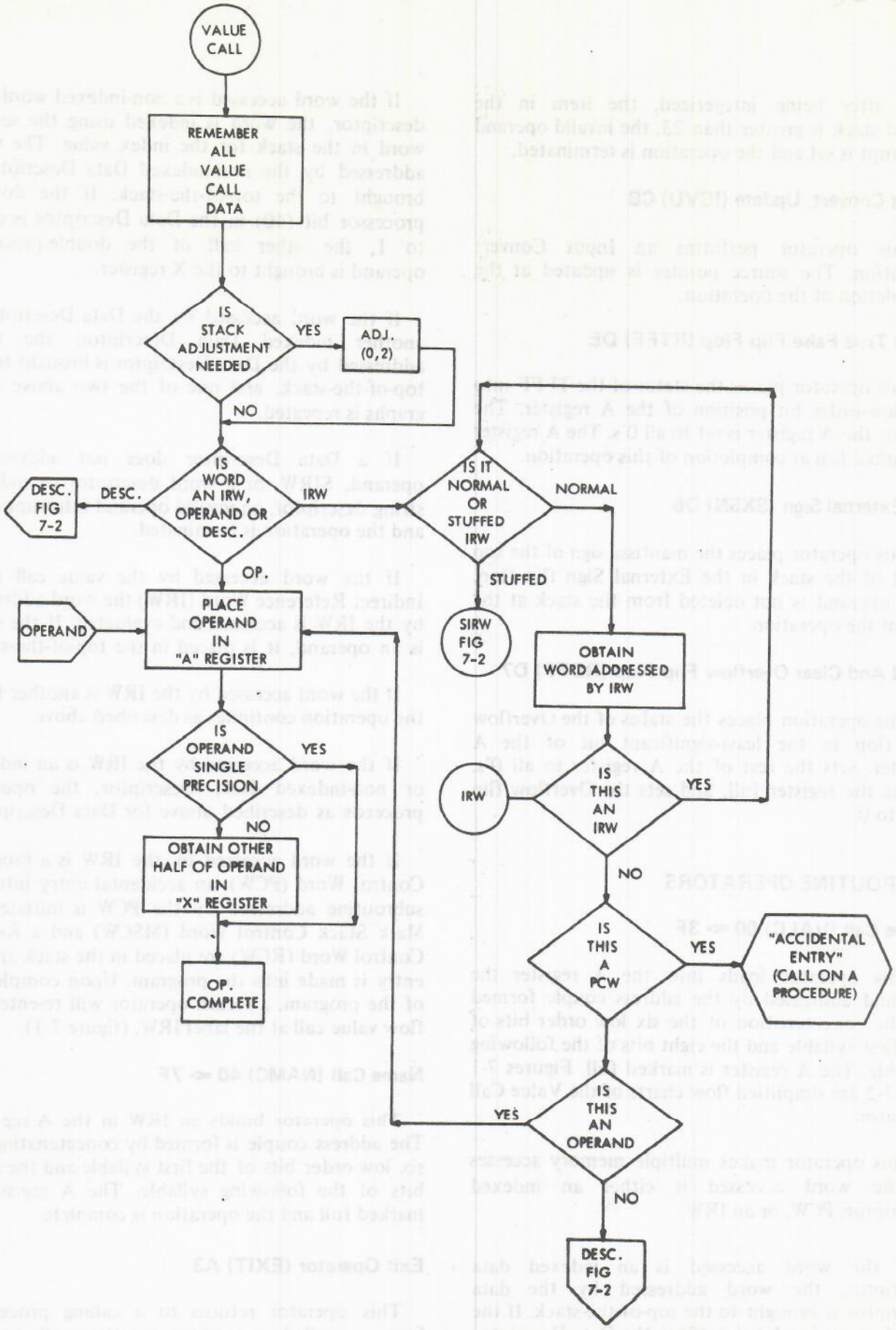


Figure 7-1. Flow of Value Call Operator

routine. Figure 7-3 shows a simplified flow chart of the Exit operator.

Return Operator (RETN) A7

This operator performs the same functions as 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 the VALC operator. 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 PCW. Figure 7-5 shows a simplified flow chart of the Enter operator.

The Enter operator accesses the IRW at F + 1, which points to the PCW. The operator then builds a RCW into the stack at F + 1.

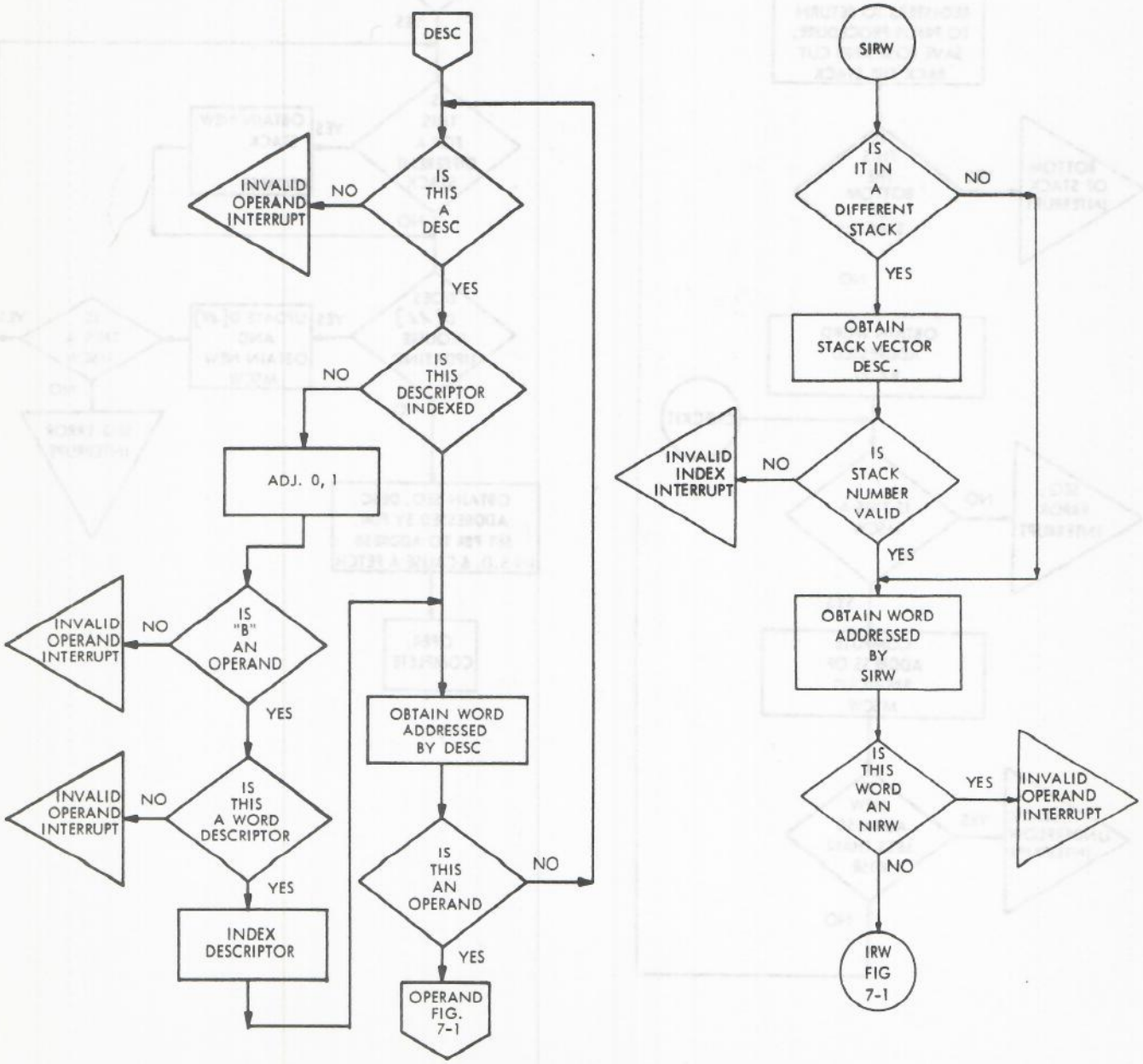


Figure 7-2. Flow of Value Call Operator (Cont)

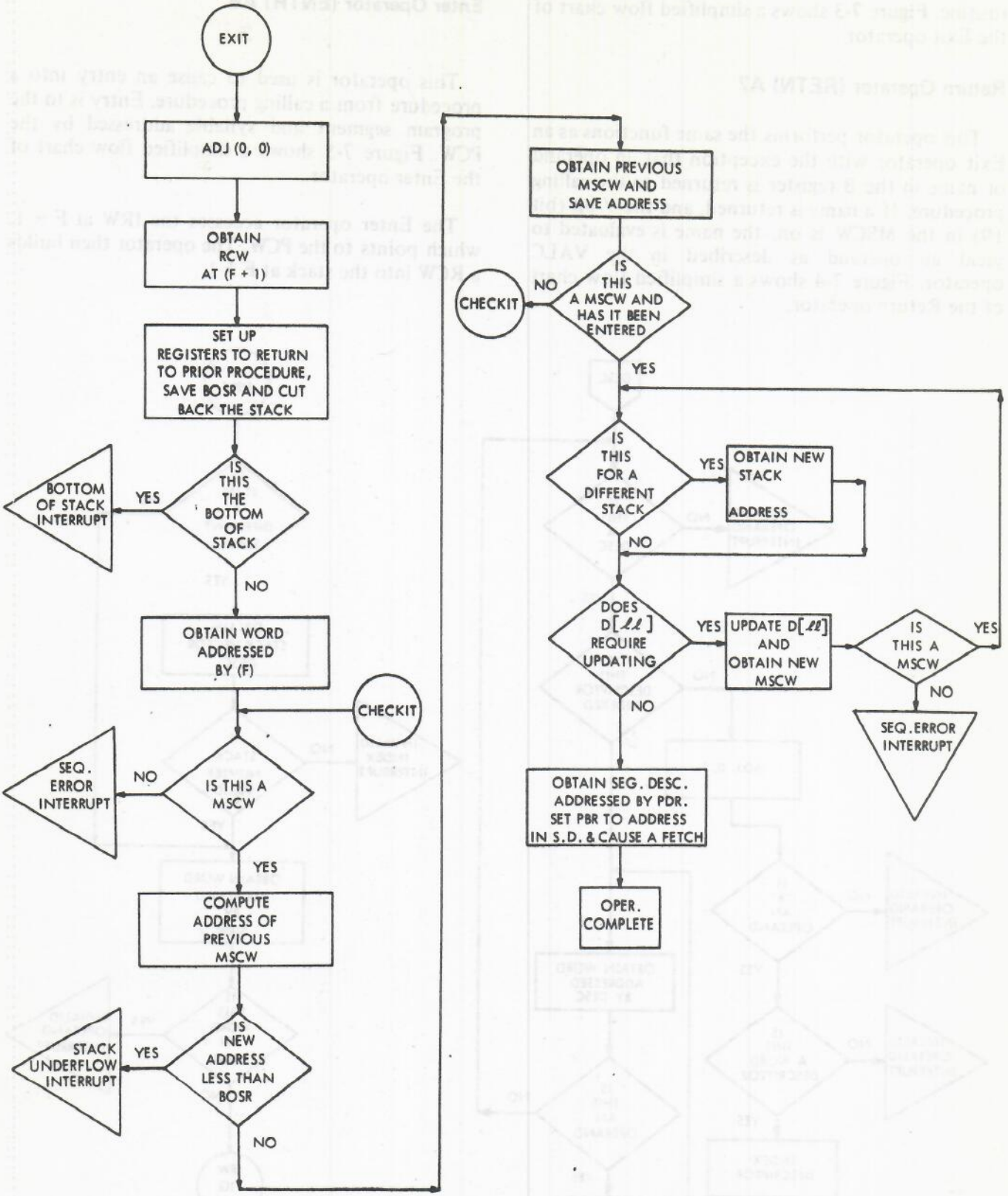


Figure 7-3. Flow of Exit Operator

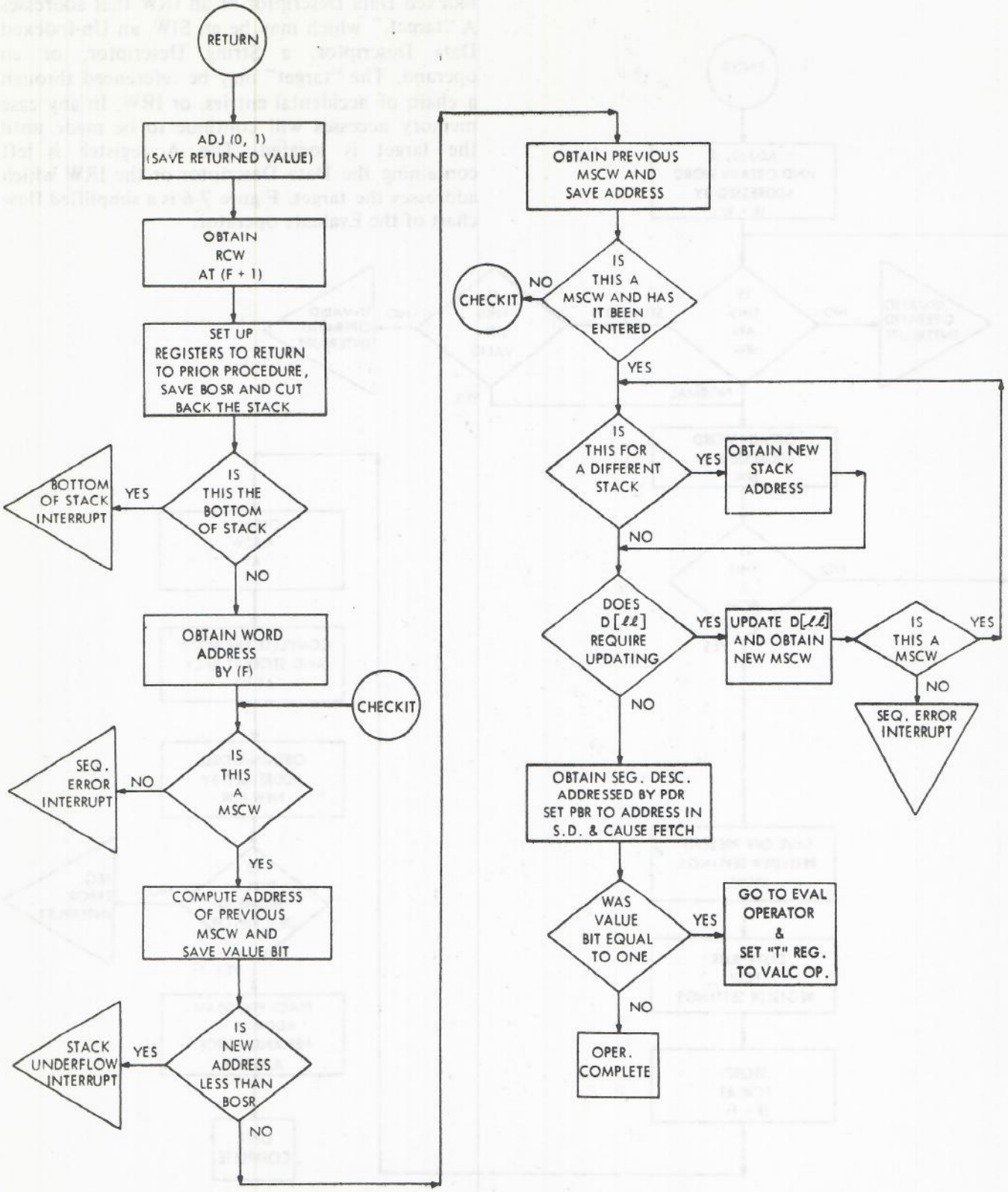


Figure 7-4. Flow of Return Operator

Evaluate (EVAL) AC

This operator loads the A register with an indexed Data Descriptor or an IRW that addresses A "target," which may be an SIW, an Un-Indexed Data Descriptor, a String Descriptor, or an operand. The "target" may be referenced through a chain of accidental entries, or IRW. 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 IRW which addresses the target. Figure 7-6 is a simplified flow chart of the Evaluate operator.

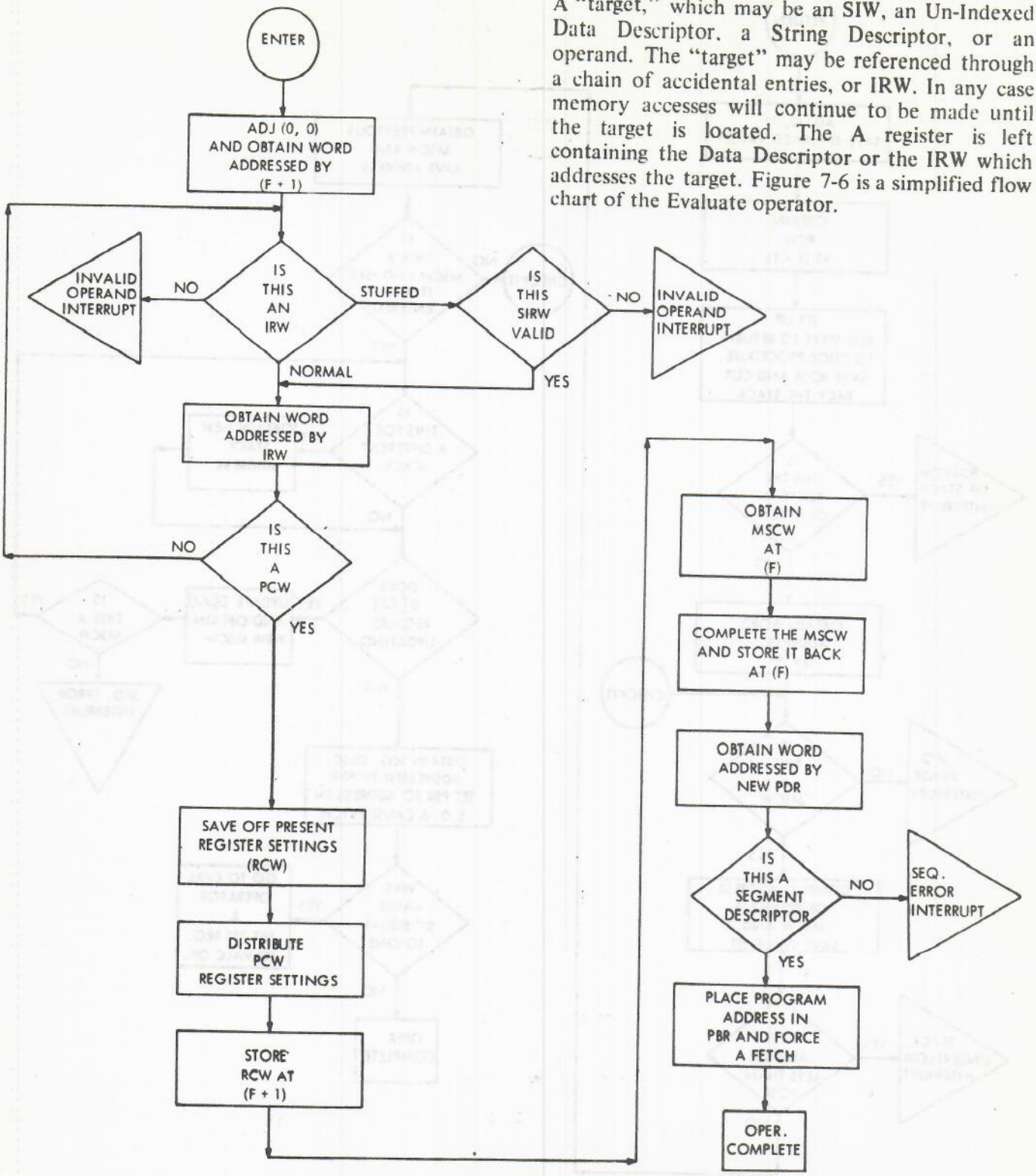
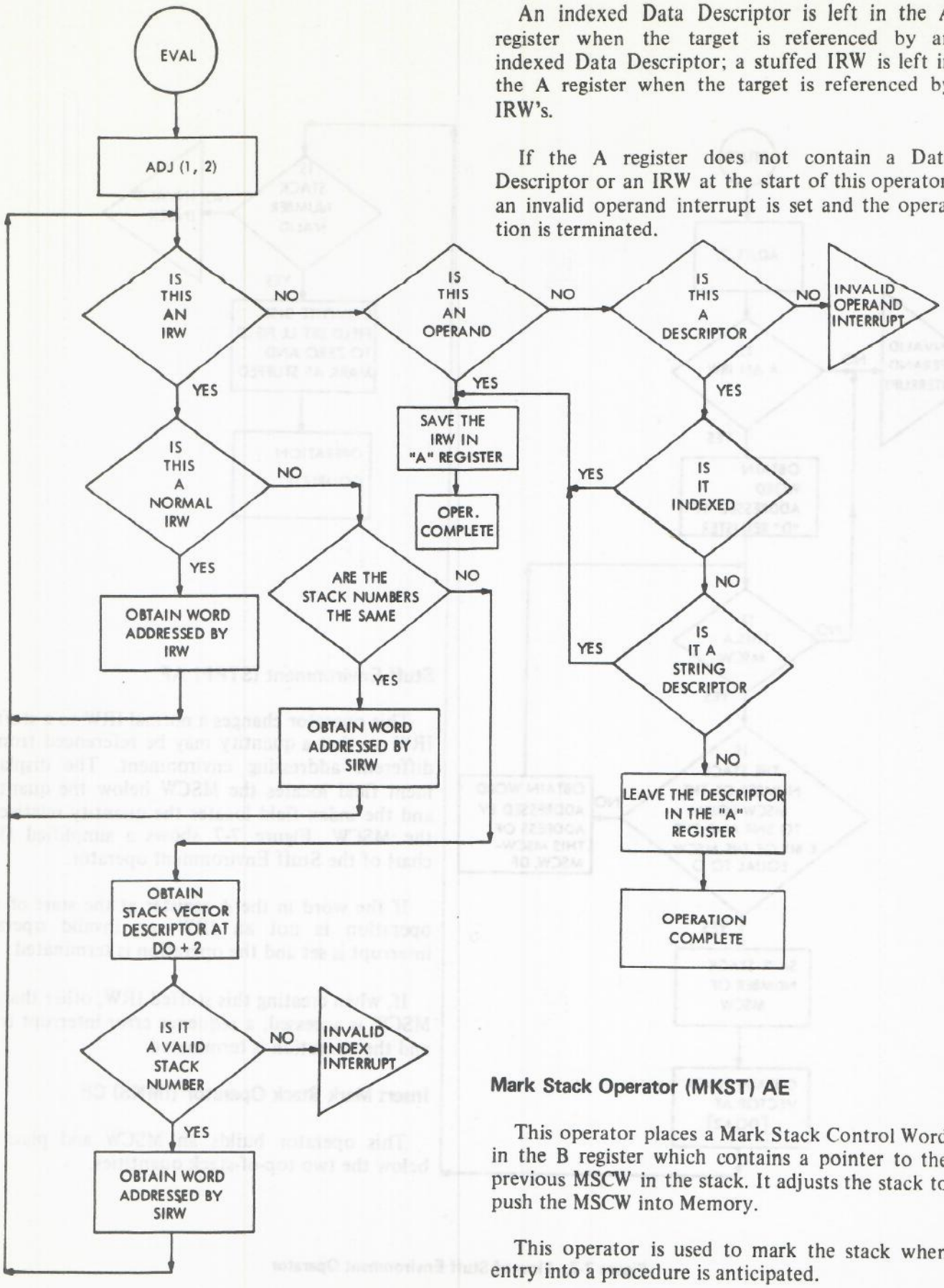


Figure 7-5. Flow of Enter Operator



An indexed Data Descriptor is left in the A register when the target is referenced by an indexed Data Descriptor; a stuffed IRW is left in the A register when the target is referenced by IRW's.

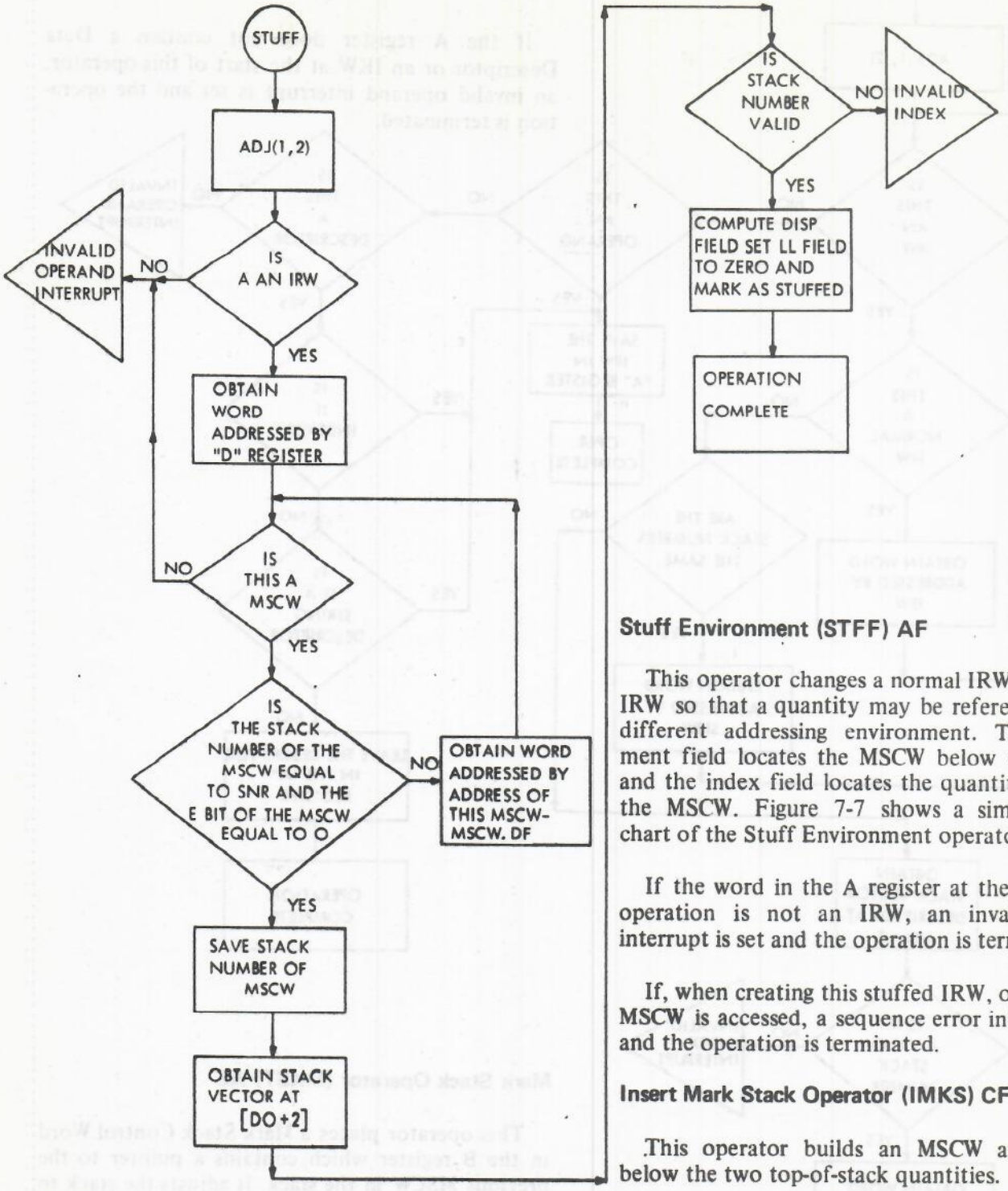
If the A register does not contain a Data Descriptor or an IRW at the start of this operator, an invalid operand interrupt is set and the operation is terminated.

Mark Stack Operator (MKST) AE

This operator places a Mark Stack Control Word in the B register which contains a pointer to the previous MSCW 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.

Figure 7-6. Flow of Evaluate Operator



Stuff Environment (STFF) AF

This operator changes a normal IRW to a stuffed IRW 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 operation is not an IRW, an invalid operand interrupt is set and the operation is terminated.

If, when creating this stuffed IRW, other than an MSCW is accessed, a sequence error interrupt is set and the operation is terminated.

Insert Mark Stack Operator (IMKS) CF

This operator builds an MSCW and places it below the two top-of-stack quantities.

Figure 7-7. Flow of Stuff Environment Operator

SECTION 8

VARIANT MODE OPERATION AND OPERATORS

GENERAL

Escape To 16-Bit Instruction (VARI) 95

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 EO through 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 operand which are placed in the A and the B registers; oth registers are marked full.

If the operand in the B register is a single-precision operand, the A register is set to all 0's 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 the program is restarted by an external interrupt. The Normal Control State flip flop (NCSF) and the Inhibit Interrupt flip flop (IIFF) are unconditionally set to allow external interrupts.

Set Interval Timer (SINT) 9545 (Control State Operator)

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. The processor is interrupted when the timer reaches 0 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 NCSF and the (IIHF) flip flops to 0's.

Disable External Interrupts (DEXI) 9547

This operator causes the processor to ignore external interrupts. This is accomplished by setting the IIHF to 1 and entering control state.

SCAN OPERATORS

The Scan operators communicate between the processor and the Input/Output, Data Communications or General Control Subsystems via the scan bus. The scan bus consists of 20 address lines, 12 control lines, and 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 given subsystem.

Parity is checked during transmission of both address and information and a scan-bus parity error interrupt is generated if the check fails.

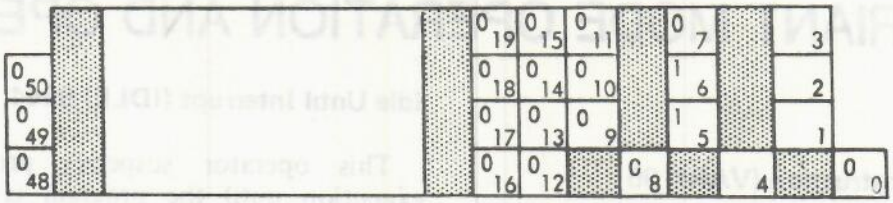


Figure 8-1. Read Time-Of-Day Function Word

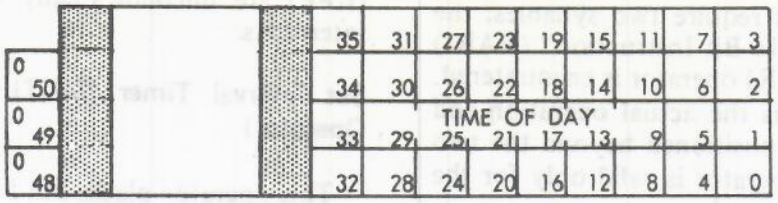


Figure 8-2. Time-of-Day Word

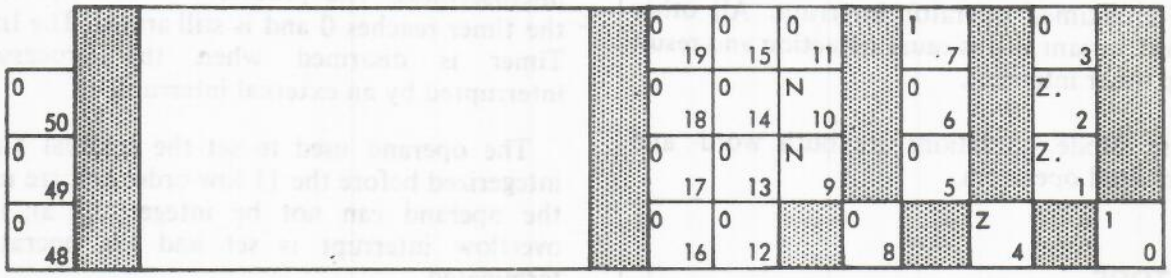


Figure 8-3. Read General Control Adapter Function Word

Scan In (SCNI) 954A

Scan In uses the A register to specify the type of input required and the I/O Processor that is to respond. The input data is placed in the B register. The A register is empty and the B register full at the completion of the operation.

Read Time-Of-Day Clock

This operation transfers the contents of the time-of-day register from the I/O processor to the B register. Note that if the system has more than one I/O processor, only one time-of-day clock is active. I/O Processor A responds when an I/O processor is not designated.

As this operation is initiated, the A register contains the function word shown in figure 8-1.

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 marked empty at the completion of this operation.

Read General Control Adapter

This operation places the contents of one of the four general control registers into the B register. Figure 8-3 shows the format of the function word present in the A register as the operation is initiated.

There are four General Control designations:

1. Z = 0001, GCA A
2. Z = 0010, GCA B
3. Z = 0100, GCA C
4. Z = 1000, GCA D

The N field is used to address or read one of four, 48-bit general control adapter registers. The following are these registers and their addresses:

1. N = 00, Input register.
2. N = 01, Interrupt mask.
3. N = 10, Interrupt register.
4. 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 I/O Processor specified. The A register contains the function word shown in figure 8-4.

I/O Processor designations are as follows:

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

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 undefined if the I/O processor has no result descriptor.

The result descriptor error field is divided into a STANDARD error field and unit error field. The STANDARD error field bit assignments are defined individually for each peripheral control:

1. Bit 0: Exception.
2. Bit 1: Attention.
3. Bit 2: Busy.
4. Bit 3: Not ready.
5. Bit 4: Descriptor error.
6. Bit 5: Memory address.
7. Bit 6: Memory parity error.
8. Bit 16: Memory protect.

The unit error field (U.N.) in figure 8-5 is the unit number field. The C.C. represents the character count field.

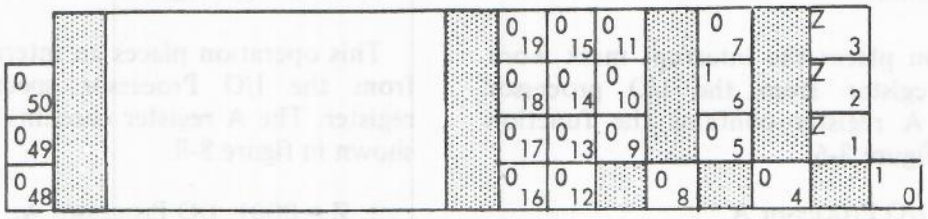


Figure 8-4. Read Result Descriptor Function Word

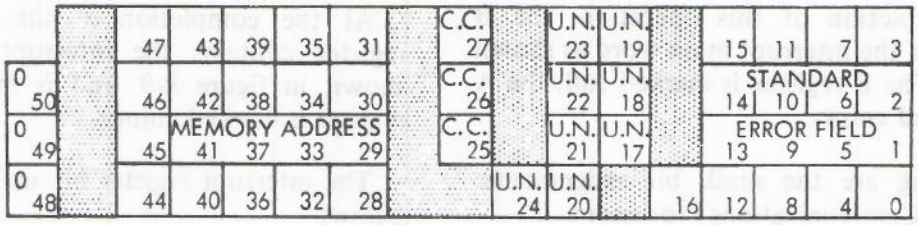


Figure 8-5. Result Descriptor

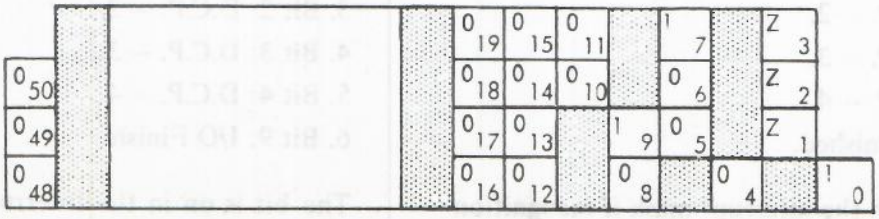


Figure 8-6. Read Interrupt Mask Function Word

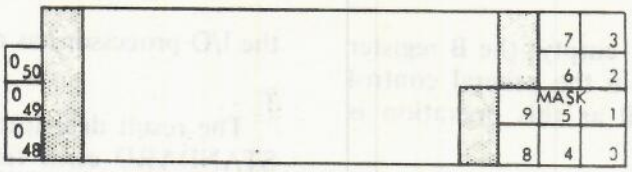


Figure 8-7. Interrupt Mask Word

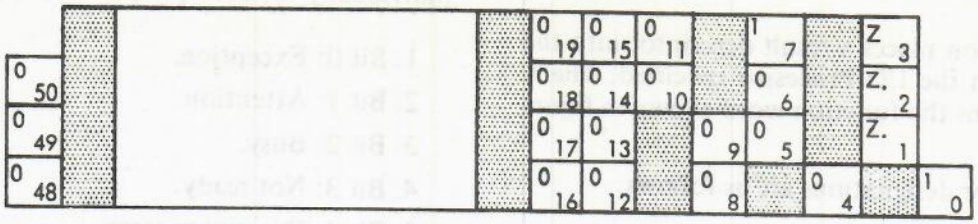


Figure 8-8. Read Interrupt Register Function Word

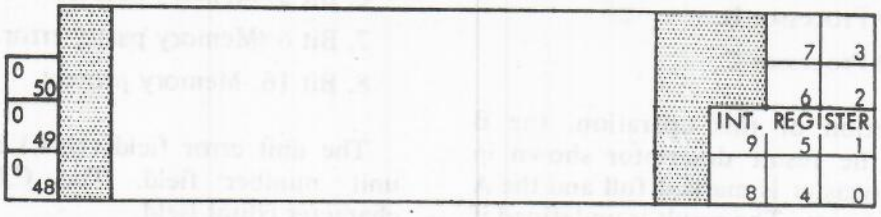


Figure 8-9. Interrupt Register Word

Read Interrupt Mask

This operation places the interrupt mask word into the B register from the I/O processor specified. The A register contains the function word shown in figure 8-6.

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

At the completion of this operation, the B register contains the interrupt mask word as shown in figure 8-7. The B register is marked full, the A register is marked empty.

The following are the mask bit assignments: (D.C.P. is Data Communications Processor)

1. Bit 0: Status change.
2. Bit 1: D.C.P. – 1.
3. Bit 2: D.C.P. – 2.
4. Bit 3: D.C.P. – 3.
5. Bit 4: D.C.P. – 4.
6. 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 from the I/O Processor specified into the B register. The A register contains the function word shown in figure 8-8.

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

At the completion of this operation, the B register contains the interrupt register word as shown in figure 8-9, and is marked full; the A register is marked empty.

The interrupt register bit assignments are shown below:

1. Bit 0: Status change.
2. Bit 1: D.C.P. – 1.
3. Bit 2: D.C.P. – 2.
4. Bit 3: D.C.P. – 3.
5. Bit 4: D.C.P. – 4.
6. Bit 9: I/O Finish.

The bit is on in the Interrupt Status Register if the interrupt is pending.

Read Interrupt Literal

This function places the interrupt literal word from the I/O Processor specified into the B register. The A register contains the function word shown in figure 8-10.

I/O Processor designations are as follows:

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

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.

The following are the interrupt literal bit assignments:

1. Bits 3:4, 0001 = I/O Processor A.
   0010 = I/O Processor B.
   0100 = I/O Processor C.
2. 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 Processor I/O finished.
   1111 = Status change.
   0000 = No external device.
   0101 = I/O Processor external interrupt.

Interrogate Peripheral Status

This operation places one of eight status vector words from one of the I/O Processors into the B register. A B 6700 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 displays the status of units 0 through 31, vector-word 1 the status of units 32 through 63, etc. The A register contains the function word shown in figure 8-12.

I/O Processor designations are as follows:

1. Bit 0: M = 0, All I/O Processors are to respond.
   M = 1, I/O Processor designated by Z to respond.
2. Bits 4:4: Z = 0001, I/O Processor A.

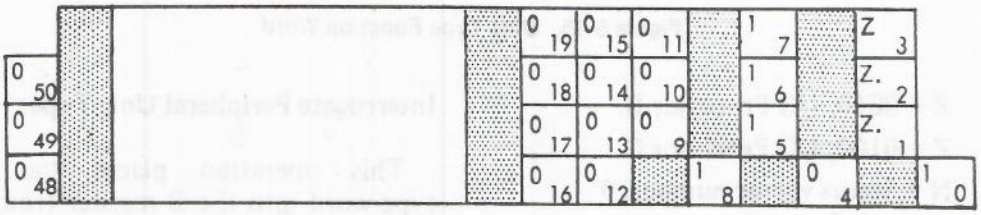


Figure 8-10. Read Interrupt Literal Function Word

Figure 8-11. Interrupt Literal Word

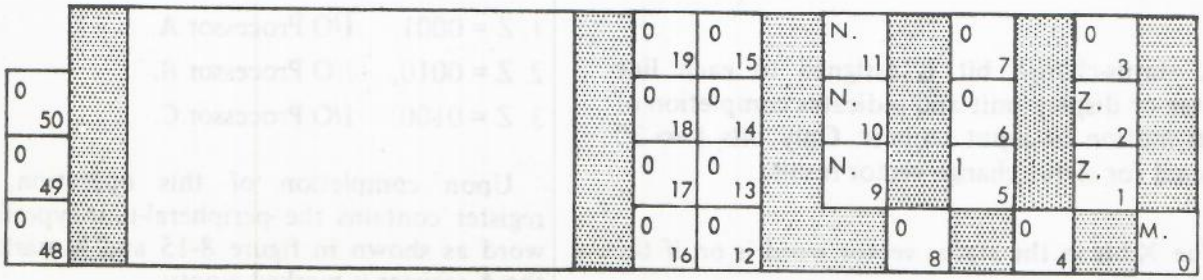


Figure 8-12. Interrogate Peripheral Status Function Word

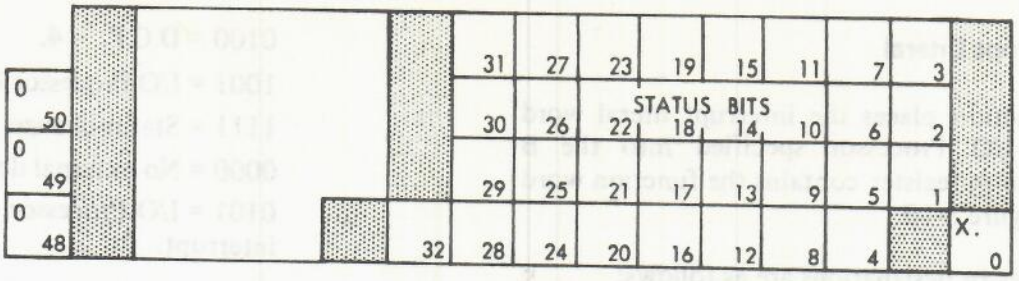


Figure 8-13. Status Vector Word

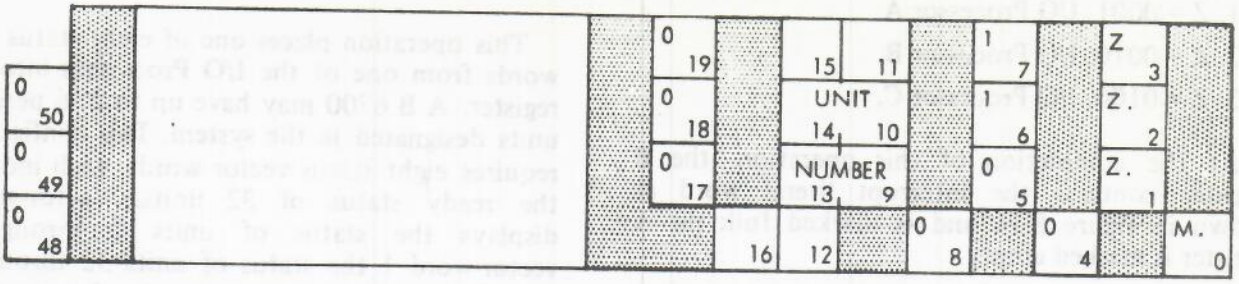


Figure 8-14. Interrogate Peripheral Unit Type Function Word

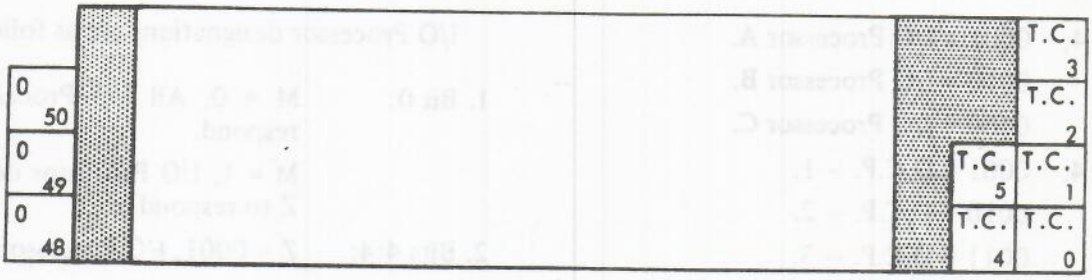


Figure 8-15. Unit Type Function Word

Z = 0010, I/O Processor B.

Z = 0100, I/O Processor C.

3. Bits 11:3: N = Status vector number, 0 through 7.

N = Status change vector, 8.

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.

A status-change bit is assigned to each line printer or display unit and indicates completion of paper-motion or input request. Only bits 1 ⇒ 30 are used for status change vector result.

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 I/O Processors. The A register contains the function word shown in figure 8-14.

1. M = 0, All I/O Processors to respond.
2. M = 1, I/O Processor designated by Z to respond.

When M = 1, the Z field MPX designations are:

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

Upon completion of this operation, the B register contains the peripheral-unit-type function word as shown in figure 8-15 and is marked full; the A register is marked empty.

The codes shown below identify the following units:

	Code	Unit
1.	00	No unit.
2.	01	Disk file.
3.	02	Display.
4.	04	Paper-tape reader.
5.	05	Paper-tape punch.
6.	06	Buffered line-printer, BCL drum.
7.	07	Unbuffered line-printer, BCL drum.
8.	09	Card reader.
9.	OB (11)	Card punch.
10.	OD (13)	Magnetic tape (7 channel). With status vector information.
11.	OE (14)	Magnetic tape (9 channel N.R.Z.). With status vector information.
12.	OF (15)	Magnetic tape (9 channel P.E.). With status vector information.
13.	1D (29)	Magnetic tape (7 channel). No status vector information.
14.	1E (30)	Magnetic tape (9 channel N.R.Z.). No status vector information.
15.	1F (31)	Magnetic tape (9 channel P.E.). No status vector information.
16.	26 (38)	Buffered line-printer, EBCDIC-subset drum.
17.	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 function word shown in figure 8-16.

Primary I/O Processor designations are as follows:

1. M = 0, All I/O Processors respond.
2. M = 1, I/O Processor designated by Z to respond.

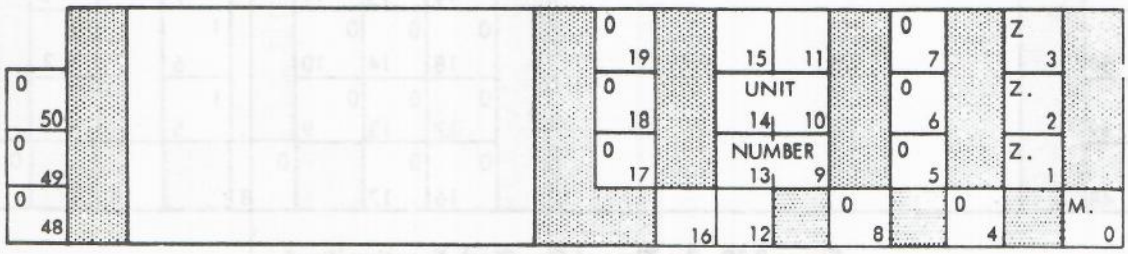


Figure 8-16. Interrogate I/O Path Function Word

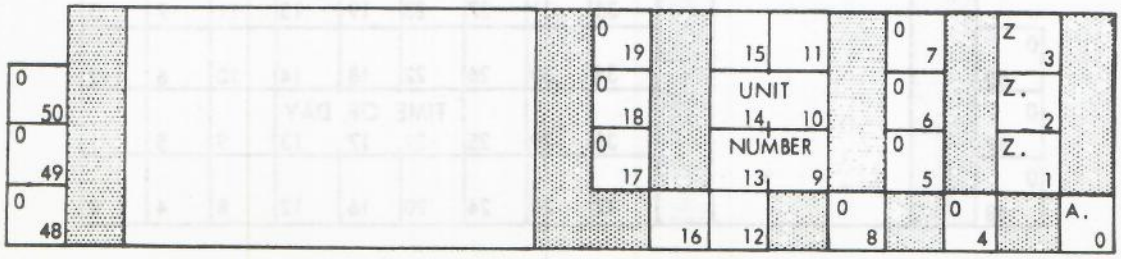


Figure 8-17. I/O Path Result Word

I/O Processor designations with M=1 are shown below:

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

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.

The A-bit indicates path availability:

1. A = 0, No path available.
2. A = 1, Path is available.

The Z field identifies the I/O processor when a path is available.

1. Z = 0001, Path is via I/O Processor A.
2. Z = 0010, Path is via I/O Processor B.
3. Z = 0011, Path is via either I/O Processor A or B.
4. Z = 0100, Path is via I/O Processor C.
5. Z = 0101, Path is via I/O Processor A and C.
6. Z = 0110, Path is via I/O Processor B and C.
7. Z = 0111, Path via all I/O Processors.

A data path consists of a data switching channel and a peripheral control.

Scan Out (SCNO) 954B

Scan-out places bits 0 through 19 of the top-of-stack word on the scan-bus address lines, and also places 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 I/O Processor (figure 8-19). The function word shown in figure 8-18 is in the A register. I/O Processor responds when an I/O Processor 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.

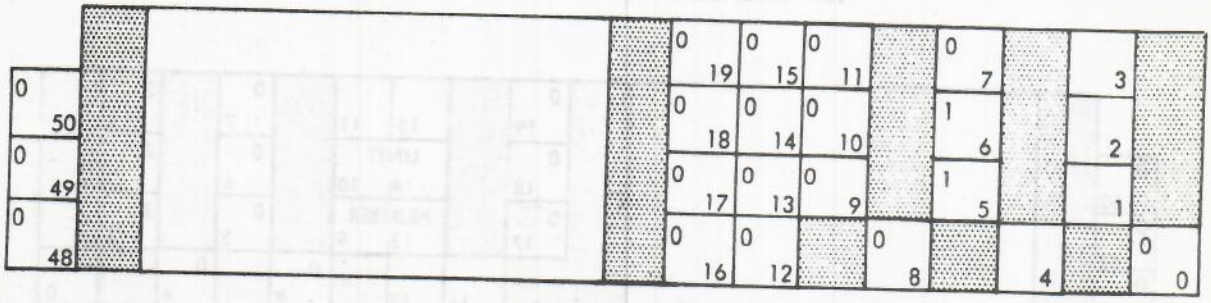


Figure 8-18. Set Time-of-Day Clock Function Word

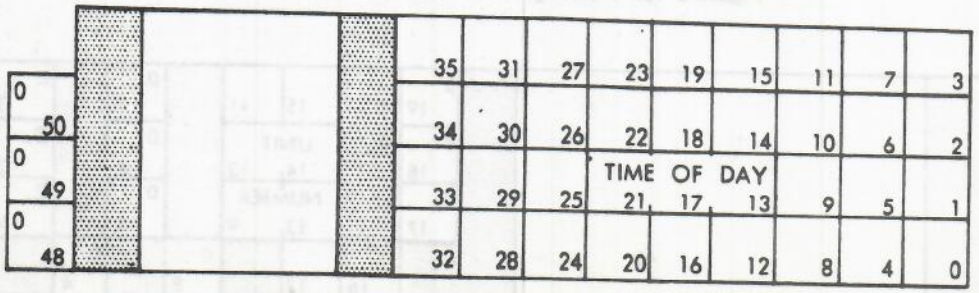


Figure 8-19. Time-of-Day Word

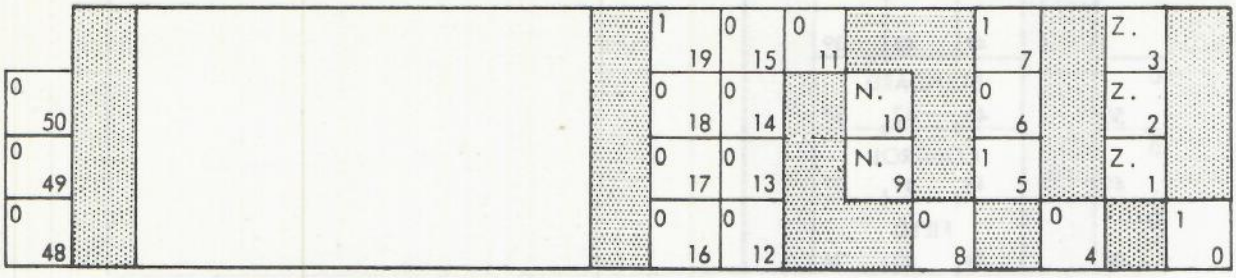


Figure 8-20. Set General Control Adapter Function Word

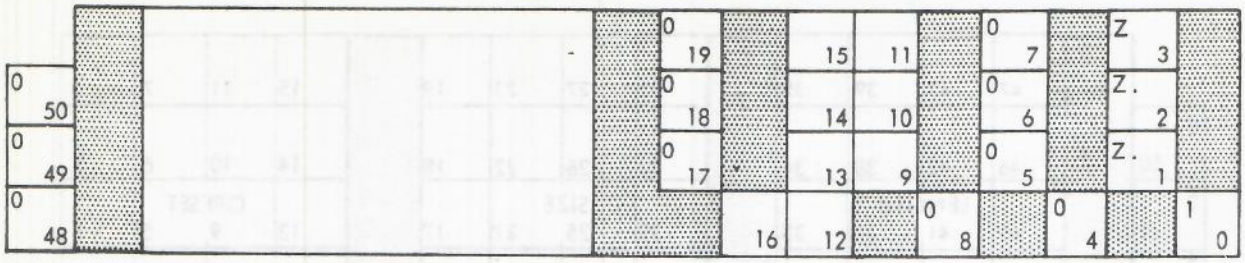


Figure 8-21. Initiate I/O Function Word

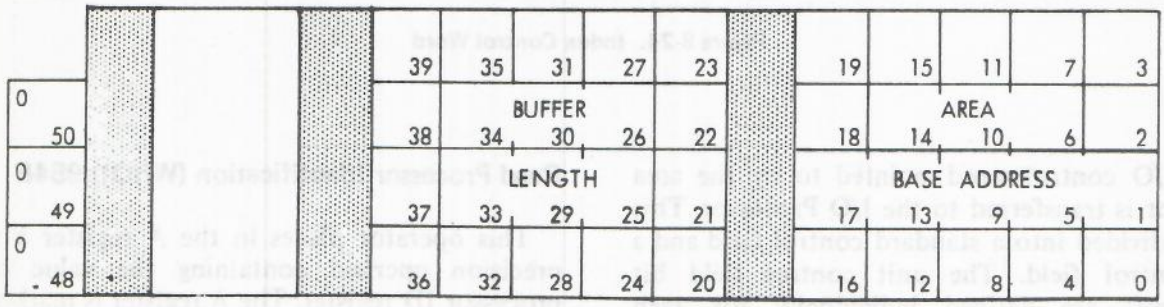


Figure 8-22. Area Descriptor

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 function word shown in figure 8-20, and the B register contains the output, the interrupt mask or the interrupt word.

I/O Processor designations are as follows:

1. Z = 0001, I/O Processor A.
2. Z = 0010, I/O Processor B.
3. Z = 0100, I/O Processor C.

Output, interrupt mask, or interrupt register designations are as follows:

1. N = 00, Output.
2. N = 01, Interrupt mask register.
3. N = 10, Interrupt register.

At the completion of this operation, 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 function word in the A register. The code word format is shown in figure 8-21.

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.

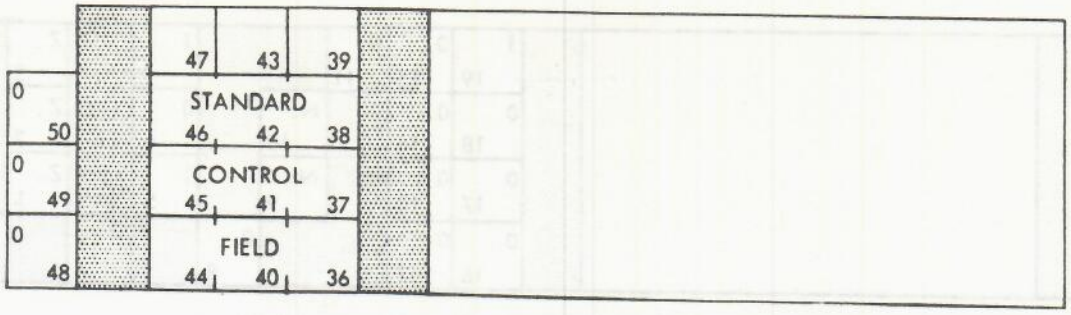


Figure 8-23. I/O Control Word

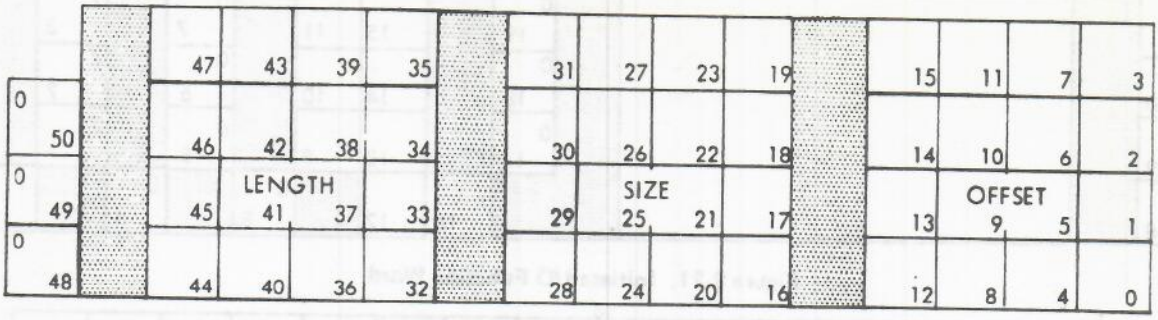


Figure 8-24. Index Control Word

The I/O control word pointed to by the area descriptor is transferred to the I/O Processor. This word is divided into a standard control field and a unit control field. The unit control field bit assignments are defined individually for each control. For additional information concerning unit control field bit assignments for each control, refer to section 5 of the B 6700 System Handbook.

Bit	Assignment	Bit=0	Bit=1
47	Reserved	– –	– –
46	Reserved	– –	– –
45	Attention	No	Yes
44	Read/write	write	read
43	Memory inhibit	No	Yes
42	Translate	No	Yes
41	Frame length	6-bit	8-bit
40	Memory protect	No	Yes
39	Backward transfer	No	Yes
38	Test	No	Yes
37-36	Tag field transfer	37=1	36=1
37-36	Store program tag	37=0	36=1
37-36	Store single-precision tag	37=0	36=0
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 A register is marked full.

Interrupt Other Processor (HEYU) 954F

This operator sets the processor interrupt flip flop of the other processor(s).

Occurs Index (OCRX) 9585

This operator places the following 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).

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 [47:16] is multiplied by the index value [15:16] 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.

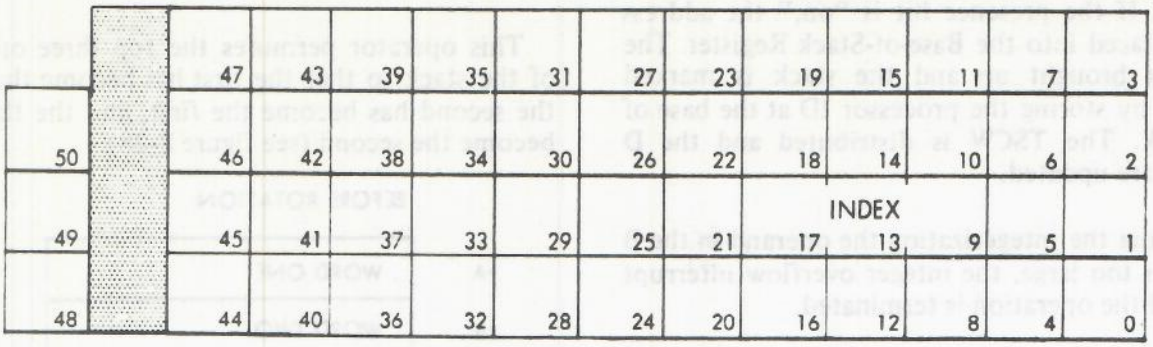
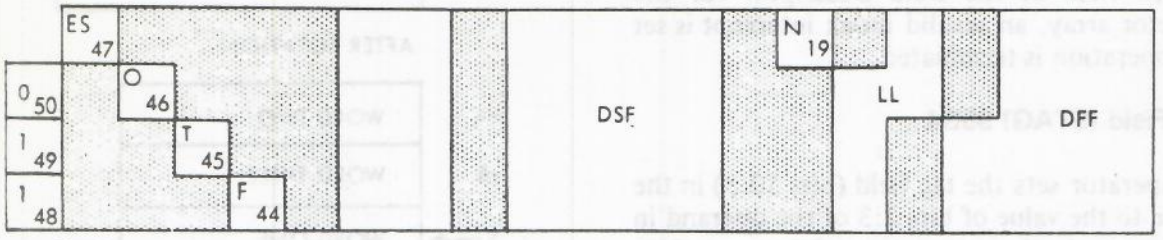


Figure 8-25. Index Word



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)

If either the ICW or the operand has a value of 0, the invalid index interrupt is set and the operation is terminated.

If the index value is less than 0 or greater than the SIZE field [31:16] of the ICW, the invalid index interrupt is set and the operation is terminated.

Integerized, Rounded, Double-Precision (NTGD) 9587

This operator creates from the operand in the B register a double-precision, rounded integer 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 all 0's.

The B register is marked full.

Move To Stack (MVST) 95AF

This operator causes the environment of the processor (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 (TSCW) (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 against the stack vector for invalid index. 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.

The Data Descriptor for the requested stack is accessed. If the presence bit is "on," the address field is placed into the Base-of-Stack Register. The TSCW 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 0 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 the second, the second has become the third, and the third has become the first (see figure 8-27).

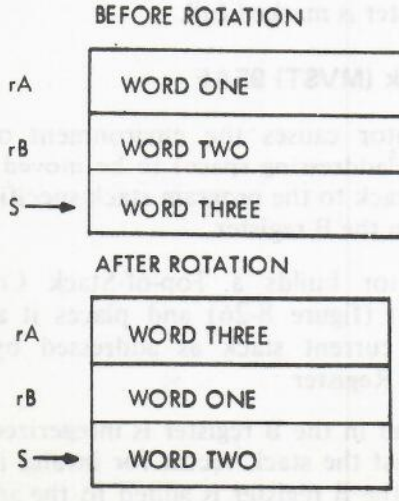


Figure 8-27. Stack Rotation Up

Rotate Stack Down (RSDN) 95B7

This operator permutes the top three operands of the stack so that the first has become the third, the second has become the first, and the third has become the second (see figure 8-28).

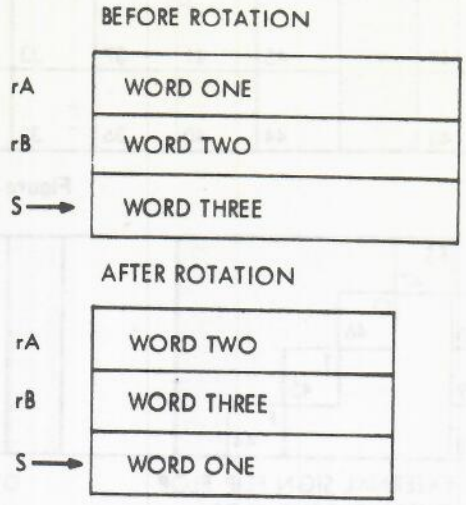


Figure 8-28. Stack Rotation Down

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:

Bits 5:2	=10	= Index register
Bits 2:3	= 0,	= PIR
	= 1,	= SIR
	= 2,	= DIR
	= 3,	= TIR, BUF 3
	= 4,	= LOSR
	= 5,	= BOSR
	= 6,	= F
	= 7,	= BUF
Bits 5:2	=11	= Base register
Bits 2:3	= 0,	= PBR
	= 1,	= IBR
	= 2,	= DBR
	= 3,	= TBR, BUF 2
	= 4,	= S
	= 5,	= SNR
	= 6,	= PDR
	= 7,	= TEMP

If bit 5 is 0, bits 4:5 select the D register equal to the binary value of the bits; i.e., bits 4:5 = 00101 select D register 5.

At the completion of this operation the A register contains the contents of the selected register, 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 low-order bits is the same as in the Read Processor Register operator (RPRR) discussed under the previous heading.

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 A register.

Count Binary Ones (CBON) 95BB

This operator counts the number of one-bits in the single-precision (double-precision) 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 neither 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 are read and checked for a value of 0 in the address (Link) portion of the word (0 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 described in 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 0, 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 0), the index bit (bit 45) is set and 1 is subtracted from the length field. If bit 45 is equal to 1, 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 1 and the operation is repeated. If the index field is equal to 0, 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 eight-bit bytes. At the start of the operator, the word in the A register defines the length of the operand in the B register; i.e., 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:

1. If the destination size is six-bit (BCL) format, the characters are transferred to the destination with the two zone bits set to 0.
2. If the destination size is eight-bit (EBCDIC) format, the bytes are transferred to the destination string with the four zone bits set to 1111.
3. If the destination size is 0, it is set to eight-bit format and handled as in (b) above.

Unpack Absolute, Update (UABU) 95D9

This operator performs an Unpack Absolute operation; at the completion of the operation, the destination pointer is updated and left in the stack.

Unpack Signed, Destructive (USND) 95DO

This operator performs an Unpack operation, plus an added function if the External Sign flip flop is set, then a zone of 10 is set in the last character for six-bit or a zone of 1101 is set in the last byte for eight-bit.

If the destination size is four-bit, the first digit position of the destination string is set to 1101 provided the External Sign flip flop is set. If the External Sign flip flop is 0, the first digit is set to 1100.

Unpack Signed, Update (USNU) 95D8

This operator performs an Unpack Signed operation; at the completion of the operation, the destination pointer is updated.

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 1, the relationship is true.

The operator uses the top four words in the stack 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; and 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 0 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, but 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 (TFFF) is set to 1 (true); otherwise, it is set to 0 (false).

Transfer While False, Destructive (TWFD) 95D2

This operator performs a Transfer While operation and tests for a zero bit in the table.

Transfer While False, Update (TWFU) 95DA

This operator performs a Transfer While False operation, but updates 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 (TFFF) is set to 1 (true); otherwise, it is set to 0 (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 four eight-bit characters. The table sizes are as follows:

1. Four-bit digits provide a 4-word table length.
2. Six-bit characters provide a 16-word table length.
3. Eight-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.

If all the characters have been scanned at the completion of this operation, TFFF is set to 1. If the scan was stopped by the delimiter test before the end of the string, the TFFF is set to 0.

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 also updates the count and the source pointer. The updated source pointer locates the character that stopped the scan. The number of characters not scanned is placed in the A register, and the register is 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 (SGEU) 95F9

This operator performs a Scan While Greater or Equal operation, but also 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, but also 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, but also updates the count and source pointer.

Scan While Less, Destructive (SLSD) 95FO

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, but also 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, but also 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 contains the value of 1, 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 the same as for all Scan While operators.

Scan While True, Update (SWTU) 95DD

This operator performs a Scan While True operation, but also 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 contains the value of zero.

Scan While False, Update (SWFU) 95DC

This operator performs a Scan While False operation, but also 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.

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 this micro-operator is 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 this micro-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 this micro-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 micro-operator syllable.

If this micro-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) DO

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 equals 0 and the numeric portion of the source characters equals zero, the insert character is moved to the destination string.

If the Float flip flop equals 0, 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 equals 0 and the numeric portion of the character in the source string equals 0, the first-insert character is transferred to the destination string.

If the Float flip flop equals 0 and the numeric portion of the character in the source string is not 0 the Float flip flop is set. If the External Sign flip

flop equals 1, the second-insert character is transferred to the destination string. If the External Sign flip flop equals 0, the third-insert character is transferred to the destination string. The numeric version of the source character is then transferred.

If the Float flip flop equals 1, 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 0.

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 contains a 0 and the External Sign flip flop is 1.

If the Float flip flop contains a 0 and the External Sign flip flop also equals 0, then the character in the third syllable of this operator is transferred.

If the Float flip flop contains a 1, then it is reset and no characters are transferred.

Insert Unconditional (INSU) DC

This micro-operator places an insert character into the destination string for the number of times specified by the repeat value. When entered by a Table Enter Edit operator, the repeat value 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 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 string. The length of the string is given by the repeat value from the table or the stack.

If the Float flip flop contains a 0, the first insert character is inserted into the destination string.

If the Float flip flop contains a 1, 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 1.

If the External Sign flip flop is equal to 0, this operator places in the destination string the character defined by the third syllable of this operator.

Insert Overpunch (INOP) D8

If the External Sign flip flop is equal to 1, 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 0, 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 PROCESSOR AND PERIPHERAL CONTROLS

GENERAL

The internal processing speed of the B 6700 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 independently of the processor by the I/O processor. A maximum of three I/O processors may be attached to a B 6700, each one capable of processing up to 12 I/O operations simultaneously, from any of 128 peripheral devices.

OPERATION

A peripheral control bus extends from the I/O processor 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.

Any processor can initiate an operation on any I/O processor in a three processor/three I/O processor configuration, by executing a Scan Out instruction. This instruction transfers a function word and a data word to an I/O processor. If the function word specifies an Initiate I/O operation, then the data word is an Area Descriptor. The I/O processor fetches the I/O Control Word located at the Area Base Address (from the Area Descriptor) and initiates the peripheral operation. Upon completion of this operation, the I/O Finish Interrupt is set. The Result Descriptor is returned when the processor executes a Read Result Descriptor command.

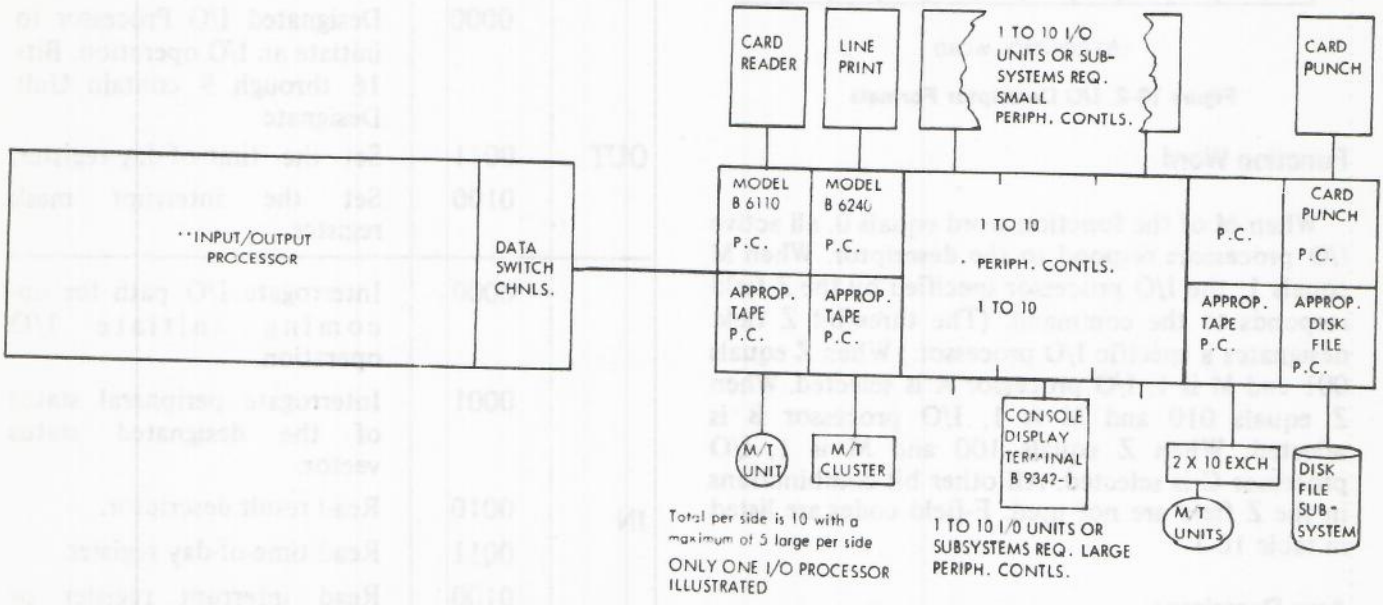
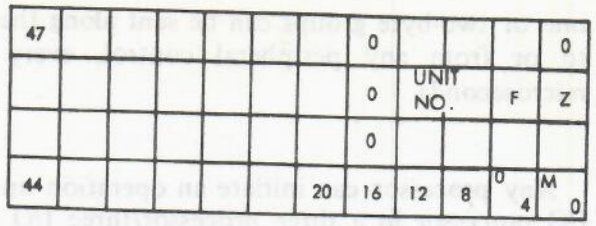


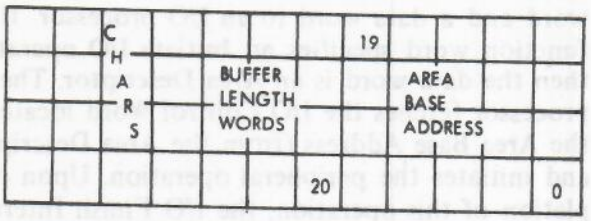
Figure 10-1. Input/Output Subsystem

DESCRIPTOR FORMATS

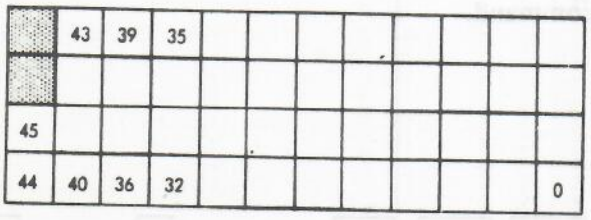
The formats of the function word, area descriptor, and I/O control word, respectively, are illustrated in figure 10-2.



FUNCTION WORD



AREA DESCRIPTOR



I/O CONTROL WORD

Figure 10-2. I/O Descriptor Formats

Function Word

When M of the function word equals 0, all active I/O processors respond to the descriptor. When M equals 1, the I/O processor specified by the Z field responds to the command. (The three-bit Z field designates a specific I/O processor.) When Z equals 001 and M is 1, I/O processor A is selected. When Z equals 010 and M is 1, I/O processor B is selected. When Z equals 100 and M is 1, I/O processor C 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 through 0, the unit control field, are unique for each peripheral control. Bits 45 through 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	I/O Processor Operation
	0000	Designated I/O Processor to initiate an I/O operation. Bits 16 through 9 contain Unit Designate.
OUT	0011	Set the time-of-day-register.
	0100	Set the interrupt mask register.
	0000	Interrogate I/O path for upcoming initiate I/O operation.
	0001	Interrogate peripheral status of the designated status vector.
IN	0010	Read result descriptor.
	0011	Read time-of-day register.
	0100	Read interrupt register or interrupt mask register.
	0110	Interrogate peripheral unit type.
	1111	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, are 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

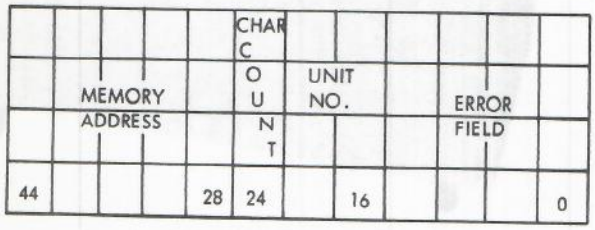


Figure 10-3. Result Descriptor Format

PERIPHERAL UNITS AND ASSOCIATED PERIPHERAL CONTROLS

Up to 256 I/O devices may be attached to a dual or triple I/O processor system. These devices communicate with the I/O processor through a maximum of 20 peripheral controls. One peripheral control cabinet houses 10 controls, five large and five small. Table 10-2 lists the peripheral controls available, excluding the magnetic tape and disk file controls which are listed separately.

Console

The Console Control Center (figure 10-4) includes the Operator Display Terminal, which allows the operator to communicate with the system. The B 6341 Control connects the Console Control Center and the I/O processor. Up to eight Operator Display Terminals may be included in a system. Figures 10-5 and 10-6 depict the result descriptor and the I/O Control word for the Single Line Control.

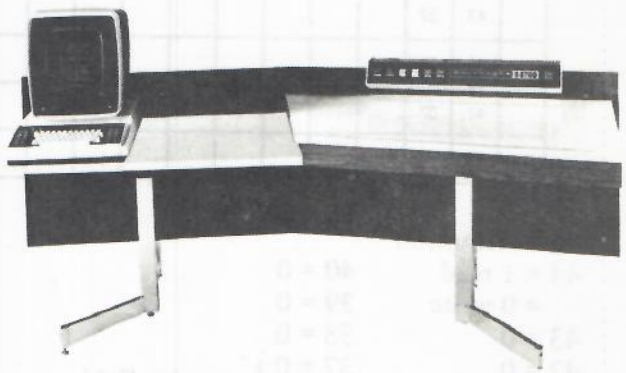


Figure 10-4. Console Control Center

Table 10-2. Peripherals and Controls

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 6212	Small	Card Punch Control
B 9220	100 CPS Paper Tape Punch	B 6220	Small	Paper Tape Punch Control
B 9242-11	860 LPM Printer (120 Prt. Pos.)	B 6240	Small	Line Printer Control
B 9243-11	1100 LPM Printer (120 Prt. Pos.)	B 6240	Small	Line Printer Control
B 9342-1	Operator Display Terminal	B 6341	Large	Operator Display Control

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 – B 9342-11
15 Time out
16 Memory protect error (read only)
24 : 8 Unit designate
27 : 3 Character count
47 :20 Memory address

Figure 10-5. Single Line Control Result Descriptor

	43	39									
	42	38									
	41	37									
44	40	36									

45 = Attention
44 = 1 read
= 0 write
43 = 0
42 = 0
41 = 1 8 bit
40 = 0
39 = 0
38 = 0
37 = 0 } tag-bit field
36 = 0 }

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. The card reader converts alpha card code to BCL, which is then converted into internal BCL or EBCDIC by translators in the I/O Processor. EBCDIC card code is converted to internal EBCDIC by the B 6110 card reader control. When binary punched cards are read 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									

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
41 = 0
37 = 0

37 tag bit
36 field

Figure 10-8. Card Read I/O Control Word

47					27					7	
									10	6	
					25		17		9		
				28	24		16		8		0

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 6212 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.

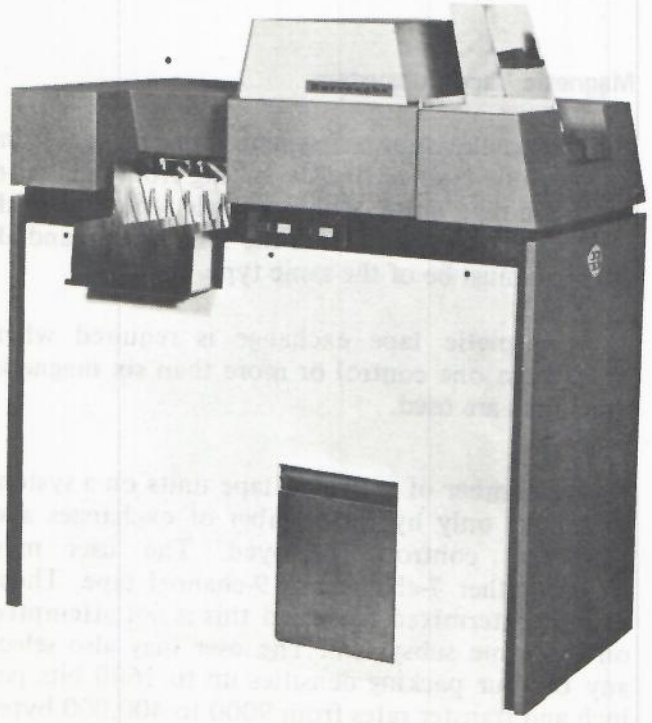


Figure 10-10. Card Punch

	42	38									
	41	37									
44		36	32								

44 = 0
38 = 0
32 = 1 Auxiliary stacker
37 tag bit
36 = 0 field

BCL
42 = 1
41 = 0 (6-bit internal frame size)

Binary
42 = 0
41 = 0
37 = 0

EBCDIC
42 = 0
41 = 1 (8-bit internal frame size)

Figure 10-11. Card Punch I/O Control Word

47					27					7	
									10	6	
					25		17				
				28	24						0

6:7 Standard Error Field
7 Punch Check
7 & 10 Memory Access Error
24:8 Unit Designate
27:3 Character Count
41:20 Memory Address

Figure 10-12. Card Punch Result Descriptor

Line Printers

Two basic line printers (figure 10-13) are available for use on the B 6700 system. The B 9242-11 prints 860 lines per minute (LPM) and the B 9243-11, 1100 LPM. Both printers are available with either 120 or 132 print positions. OCR printers are also available with printing speeds of 725 LPM and 900 LPM. All printers have vertical skipping and end-of-page formatting controlled by a punched paper tape and include the forms self-align feature. The B 6240 Line Printer Control connects the printer to the I/O Processor. Translators in the I/O Processor 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 the printer result descriptor.

Figure 10-13. Line Printer

	43		35	31							
	42	38	34	30							
	41	37	33								
44		36	32								

44 = 0

43 = 0 Print

= 1 Space – Inhibit data transfer

42 = 1 Translate to BCL

41 = 0 6 bit } Internal frame size

= 1 8 bit }

38 = 0

37 } tag bit

36 = 0 } field

35:4 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

47					27				7		
									6		
					25		17		9		
				28	24			12	8		0

6:7 Standard error field

8:2 Bit transfer error

10:1 Print check

11:1 Low paper

12:1 End of page

24:8 Unit designate

27:3 Character count

47:20 Memory address

Figure 10-15. Line Printer Result Descriptor

Magnetic Tape Subsystem

A magnetic tape subsystem can include from one to four tape controls servicing from one to 16 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 or 9-channel tape. These 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 9000 to 400,000 bytes per second.

A choice of physical construction may be made between 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.

Figure 10-16. Free-Standing Magnetic Tape Units

Figure 10-17. Cluster Tape Unit

Table 10-3
Available Magnetic Tape Subsystems

Style	Description
Magnetic Tape Units	
B 9381-12	18 KB Cluster, 2 Station, NRZ, 9-Channel, 800 BPI
B 9381-13	18 KB Cluster, 3 Station, NRZ, 9-Channel, 800 BPI
B 9381-14	18 KB Cluster, 4 Station, NRZ, 9-Channel, 800 BPI
B 9381-22	36 KB Cluster, 2 Station, NRZ, 9-Channel, 800 BPI
B 9381-23	36 KB Cluster, 3 Station, NRZ, 9-Channel, 800 BPI
B 9381-24	36 KB Cluster, 4 Station, NRZ, 9-Channel, 800 BPI
B 9382-12	36 KB Cluster, 2 Station, PE, 9-Channel, 1600 BPI
B 9382-13	36 KB Cluster, 3 Station, PE, 9-Channel, 1600 BPI
B 9382-14	36 KB Cluster, 4 Station, PE, 9-Channel, 1600 BPI
B 9382-22	72 KB Cluster, 2 Station, PE, 9-Channel, 1600 BPI
B 9382-23	72 KB Cluster, 3 Station, PE, 9-Channel, 1600 BPI
B 9382-24	72 KB Cluster, 4 Station, PE, 9-Channel, 1600 BPI
B 9383-12	18/36 KB Cluster, 2 Station, NRZ/PE, 9-Channel, 800/1600 BPI
B 9383-13	18/36 KB Cluster, 3 Station, NRZ/PE, 9-Channel, 800/1600 BPI
B 9383-14	18/36 KB Cluster, 4 Station, NRZ/PE, 9-Channel, 800/1600 BPI

Table 10-3 (Cont'd.)
Available Magnetic Tape Subsystems

Style	Description
Magnetic Tape Units (cont.)	
B 9383-22	36/72 KB Cluster, 2 Station, NRZ/PE, 9-Channel, 800/1600 BPI
B 9383-23	36/72 KB Cluster, 3 Station, NRZ/PE, 9-Channel, 800/1600 BPI
B 9383-24	36/72 KB Cluster, 4 Station, NRZ/PE, 9-Channel, 800/1600 BPI
B 9391	18/50/72 KC, Free-Standing Unit, 7-Channel, 200/556/800 BPI
B 9392	72 KB, Free-Standing Unit, 9-Channel, 800 BPI
B 9393-1	144 KB, Free-Standing Unit, 9-Channel, 1600 BPI
B 9393-3	240 KB, Free-Standing Unit, 9-Channel, 1600 BPI
B 9394-1	24/66/96 KC, Free-Standing Unit, 7-Channel, 200/556/800 BPI
B 9394-2	96 KB, Free-Standing Unit, 9-Channel, 800 BPI
B 9495-5	320 KB, Free-Standing Unit, 9-Channel, 1600 BPI
B 9495-6	400 KB, Free-Standing Unit, 9-Channel, 1600 BPI
Magnetic Tape Subsystem Controls, Exchanges and Features	
B 6381-11	18/36 KB NRZ Control, 9-Channel (For B 9381-12, 13, 14, 22, 23, 24)
B 6381-12	36/72 KB PE Control, 9-Channel (For B 9382-12, 13, 14, 22, 23, 24)
B 6381-14	18/36 KB DUAL NRZ Control, 9-Channel (Includes 2 Controls and 2 x 8 Exchange) (For B 9381-12, 13, 14, 22, 23, 24)
B 6381-15	36/72 KB DUAL PE Control, 9-Channel (Includes 2 Controls and 2 x 8 Exchange) For B9382-12, 13, 14, 22, 23, 24)
B 6381-16	DUAL NRZ/PE Control, 9-Channel (Includes 2 Controls and 2 x 8 Exchange) (For B 9383-12, 13, 14, 22, 23, 24)
B 6391-3	72KC Control, 7-Channel, (For B 9391)
B 6391-4	96KC Control, 7-Channel, (For B 9394-1)
B 6393-1	72KB Control, 9-Channel, (For B 9392)
B 6393-2	144/240 KB Control, 9-Channel, (For B 9393-1,-3)
B 6393-3	96KB Control, 9-Channel, (For B 9394-2)
B 6395-5	320/400 KB Dual Control, 9-Channel, (For B 9495-5, -6)
B 6490	2 x 10 Exchange (For B 9391, B 9392, B 9394-1, -2)
B 6492	4 x 16 Exchange (For B 9391, B 9392, B 9394-1, -2)
B 6493-1	1 x 8 Common Electronics Exchange (For B 9393-1, -2)
B 6493-2	2 x 8 Common Electronics Exchange (For B 9393-1, -2)
B 6495-1	Basic Electronics/Exchange, 2 x 8 (For B 9495 Series Only)
B 6495-2	Electronics/Exchange Extension, up to 4 x 16 (For B 6495-1)
B 6680-1	7-Channel NRZ Control Adapter (1 required per 7-Channel Port) (For B 6381-11, 14)
B 9989-1	7-Channel NRZ Station Adapter (For B 9381-12, 13, 14, 22, 23, 24)

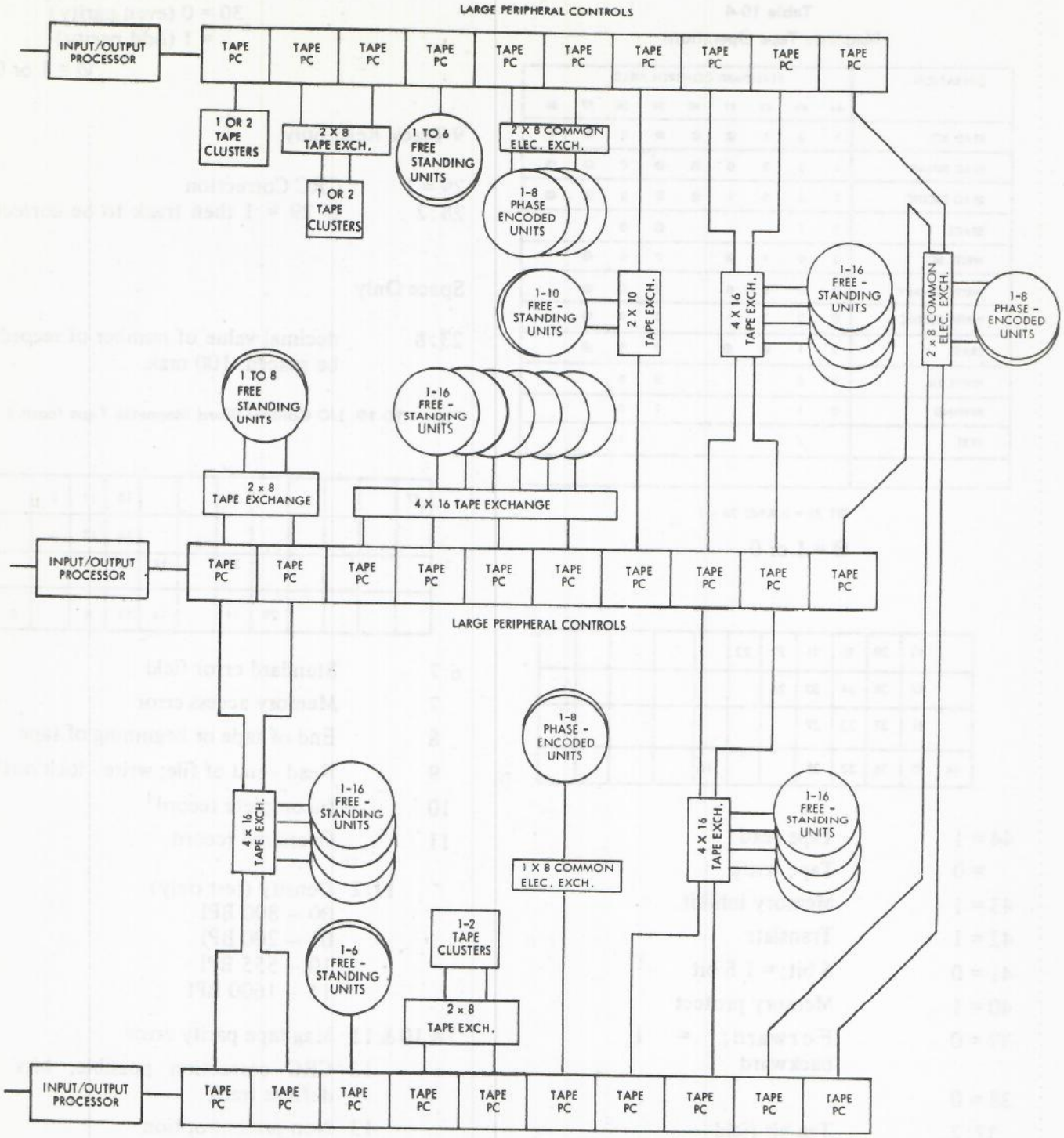


Figure 10-18. Magnetic Tape Configuration

Figure 10-19 shows the B 6700 magnetic tape I/O control word used to depict the various types of magnetic tape operations listed in table 10-4. When an operation is finished, the result descriptor returned is shown in figure 10-20.

Table 10-4
Magnetic Tape Operations

OPERATION	STANDARD CONTROL FIELD								
	44	43	42	41	40	39	38	37	36
READ BCL	1	0	1	Ø	Ø	Ø	0	Ø	Ø
READ BINARY	1	0	0	0	Ø	Ø	0	Ø	Ø
READ EBCDIC	1	0	0	1	Ø	Ø	0	Ø	Ø
SPACE	1	1				Ø	0		
WRITE BCL	0	0	1	Ø		0	0	Ø	
WRITE BINARY	0	0	0	0		0	0	Ø	
WRITE EBCDIC	0	0	0	1		0	0	Ø	
ERASE	0	1	Ø	Ø		0	0	Ø	
WRITE TM	0	0				0	0		
REWIND	0	1				1	0		
TEST							1		

BIT 35 = 0 AND 34 = 1

Ø = 1 or 0

	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 BPI
- 1111 1600 BPI (9-track only)
- 000Ø Unit-selected density

30 = 0 (even parity)
= 1 (odd parity)
Ø = 1 or 0

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.)

47					27			15	11	7	
								14	10	6	
					25		17	13	9		
				28	24		16	12	8		0

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

11:2 Density (test only)
00 – 800 BPI
01 – 200 BPI
10 – 555 BPI
11 – 1600 BPI

7&10&11 Mag tape parity error
12 CRC correction possible, bits 15:3 defines track
13 Non-present option
14 Unit is in a rewind when bit 12 is off

Figure 10-20. Magnetic Tape Result Descriptor

15	Six-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 Memory Systems

The Disk File Memory Systems are extremely high-speed, modular, random information storage systems. 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 6700 system. The exchanges involved are located within the auxiliary cabinets that are attached to the peripheral control cabinets. All of the disk file controls are the large size controls; therefore, they must be located only in positions 0 through 4 in the peripheral control cabinet.

The various types of disk file memory systems 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.

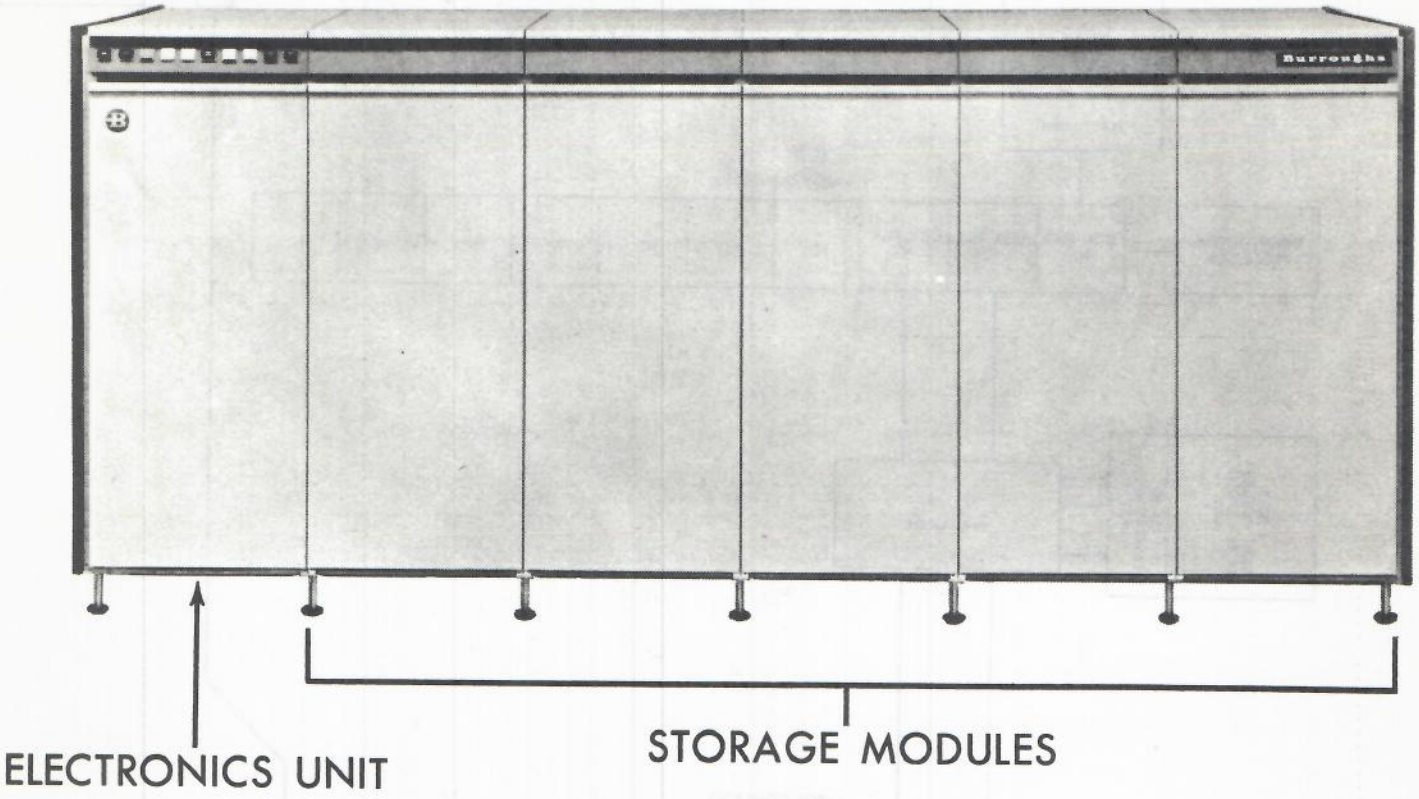


Figure 10-21. Basic Disk File Subsystem

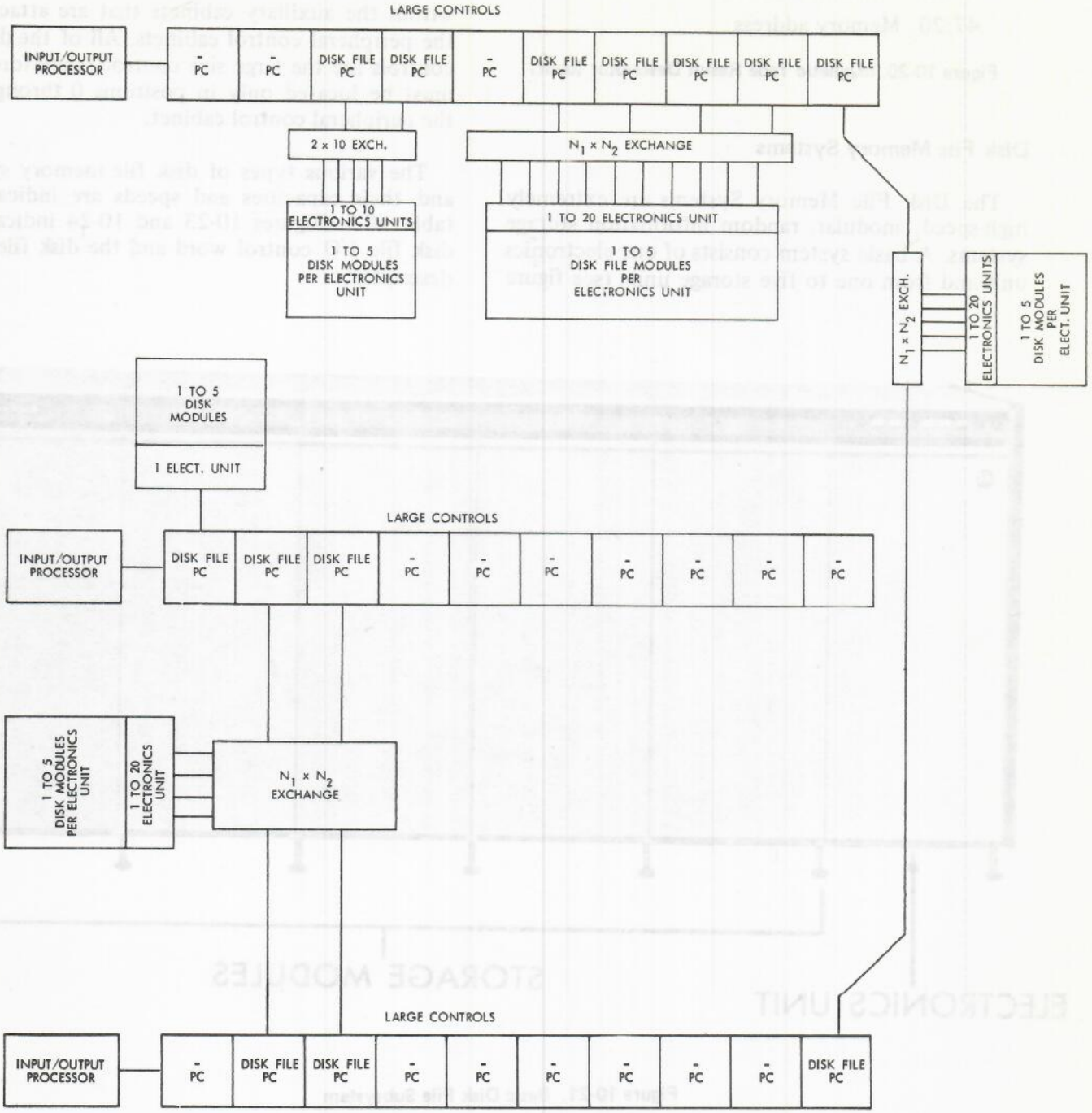
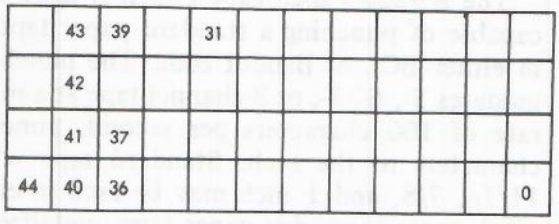


Figure 10-22. Disk File Configurations

Table 10-5
Disk File Memory System Types

Style	Description
Head-Per-Track Disk Files	
B 9379-20	20 Million Byte, 23 ms Disk File (Includes 1 DFEU)
B 9379-21	20 Million Byte, 23 ms Increment for B 9379-20 (4 max. per B 9379-20)
B 9379-30	20 Million Byte, 40 ms Disk File (Includes 1 DFEU)
B 9379-31	20 Million Byte, 40 ms Increment for B 9379-30 (4 max. per B 9379-30)
Head Per-Track Memory Banks	
B 9375-1	100 Million Byte, 23 ms (Includes 1 DFEU)
B 9375-2	20 Million Byte, 23 ms Increment for B 9375-1
B 9375-4	100 Million Byte, 40 ms (Includes 1 DFEU)
B 9375-5	20 Million Byte, 40 ms Increment for B 9375-4
Disk File Electronics Units	
B 9371-8	Optional Additional DFEU for B 9379-20, B 9375-1
B 9371-9	Optional Additional DFEU for B 9379-30, B 9375-4
Disk File Controls, Exchanges and Features	
B 6373	Disk File Control
B 6471	N1 x N2 Disk File Exchange (Up to 4 x 20)
B 6471-5	Control Adapter (N1 Side, up to 4 per B 6471)
B 6471-6	EU Adapter (N2 Side, up to 20 per B 6471)
B 6471-7	Exchange Extension (For over 10 DFEU's)
B 6473	1 x 2 Disk Exchange
Disk File Optimizer and Features	
B 6375	Basic Disk File Optimizer (DFO) (Includes 8 Words of DFO Memory)
B 6675-1	DFO Memory Increment of 8 Words (32 Words Maximum)
B 9971-11	DFSU Adapter for DFO (1 required per DFSU controlled by DFO)



44 = 1, 43 = 0 } Disk file read

44 = 1, 43 = 1 } Read check

44 = 0, 43 = 0 } Write

42 = 0 No translation
41 = 1 8-bit characters
40 = 1 Memory protect
39 = 1 Maintenance segment
37, 36 } Tag bit / Field
31:24 Disk file address (decimal)

Figure 10-23. Disk File I/O Control Word

47					27			15	11	7	
										6	
					25		17		9		
				28	24		16		8		0

6:7 Standard error field

7 Memory access error or read error or write lockout

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

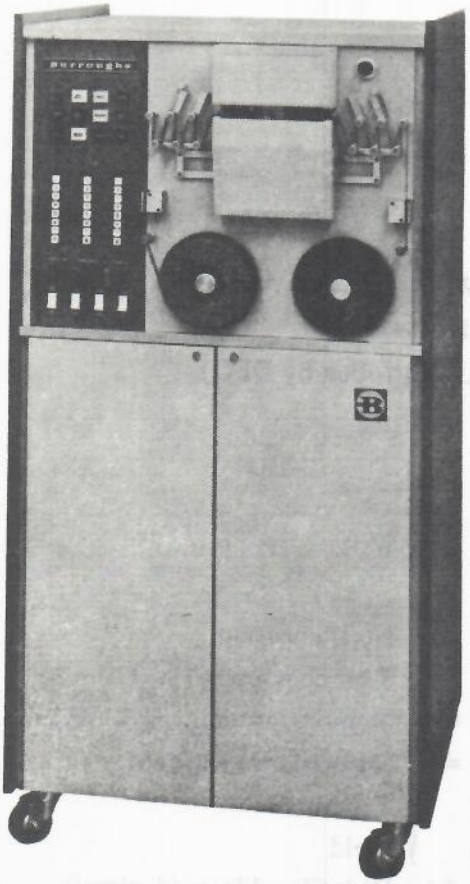


Figure 10-25. B 9120 Paper Tape Reader

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.

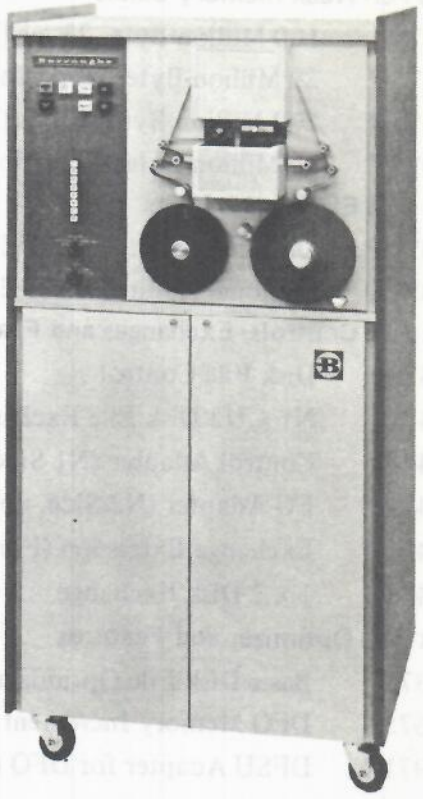


Figure 10-26. B 9220 Paper Tape Punch

The B 9220 Paper Tape Punch (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 10 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. 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 10-27 indicates the paper tape control word and the various paper tape operations possible on the B 6700. Figure 10-28 indicates the paper tape result descriptor.

	43	39	35								
	42	38	34								
		37									
44		36									

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 bit
00 - 7 bit information plus 1 parity bit
01 - 6 bit information plus 1 parity bit

	44	43	42	41	40	39	38	37	36	35	34
READ BCL	1	0	1	0	0	0	0	0	0	0	1
READ BINARY	1	0	0	0	0	0	0	0	0	0	0
WRITE BCL	0	0	1	0	0	0	0	0	0	0	1
WRITE BINARY	0	0	0	0	0	0	0	0	0	0	0
PUNCH LEADER	0	1		0	0	0	0	0	0		
FWD SPACE	1	1		0		0	0	0			
BKWD SPACE	1	1		0		1	0	0			
REWIND	0	1				1	0				

Figure 10-27. Paper Tape I/O Control Word and Operations

47					27					7	
									10	6	
					25		17		9		
				28	24		16		8		0

6:7 Standard error field
7 Memory access error or tape read parity 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

Disk-Pack Drive Memory System

The Magnetic Actuator Disk-Pack Drive Memory Systems are extremely high-speed, modular, random information storage systems. A basic disk-pack drive memory subsystem includes the disk-pack drive controller, dual disk-pack drive, and the interconnecting cables. (See figure 10-29.)

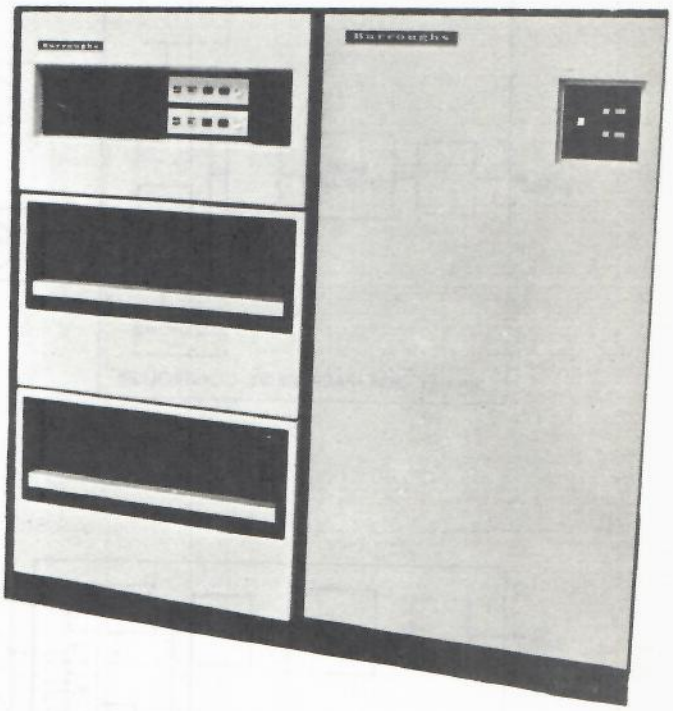


Figure 10-29.
Disk-Pack Drive and Disk-Pack Drive Controller

The controller acts upon I/O instructions from the B 6700 I/O processor, powers the disk-pack drive, and transfers information between disk-pack drives and the B 6700 I/O processor. The controller performs the operation specified by the OP code (and variants) of the I/O descriptor, and, at the completion of the operation, generates a result descriptor which contains operation and/or error status information.

The disk-pack drive controller with single access capabilities may be used with eight disk-pack spindles (four dual drives) in a one-by-eight configuration, or two groups of eight disk-pack spindles (eight dual drives) in a one-by-16 configuration. Selection of each group is determined by a variant in the I/O descriptor. The disk-pack drive controller with dual access capability may be used in a two-by-eight configuration in which the disk-pack drive controller contains two internal control units. This allows the I/O processor to execute two simultaneous operations (two reads, two writes, or a read and a write). This configuration can be expanded to a two-by-16 configuration. See figure 10-30 for a subsystem block diagram.

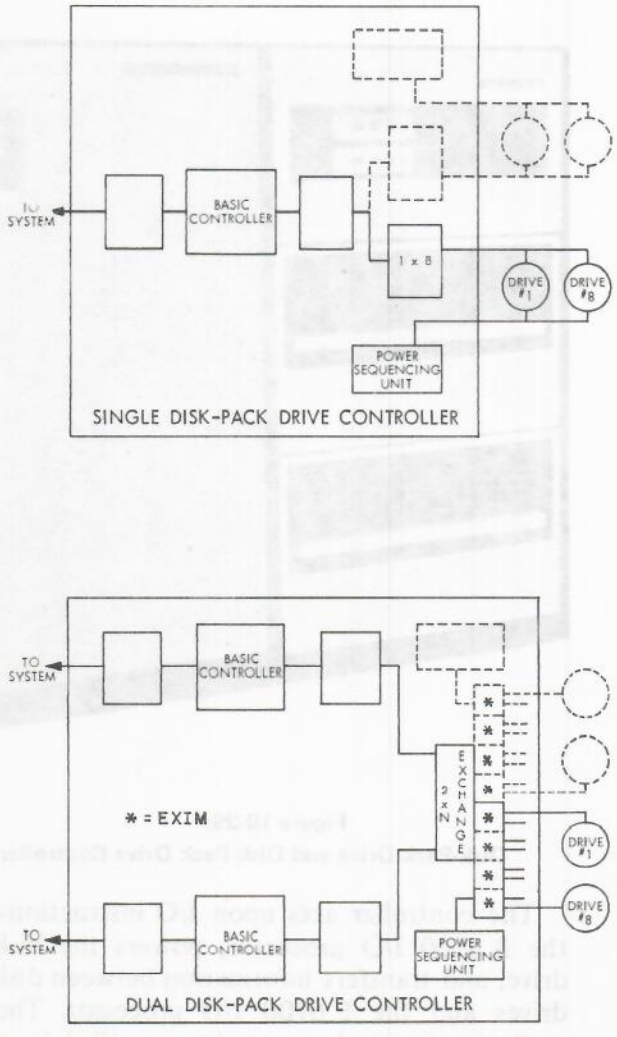


Figure 10-30. Disk-Pack Subsystem Block Diagram

Each disk pack contains 11 disks and 20 recording surfaces, each surface accessed by an individual arm from the actuator. Each disk surface contains 406 tracks. (See figure 10-31 for details of the recording surfaces.)

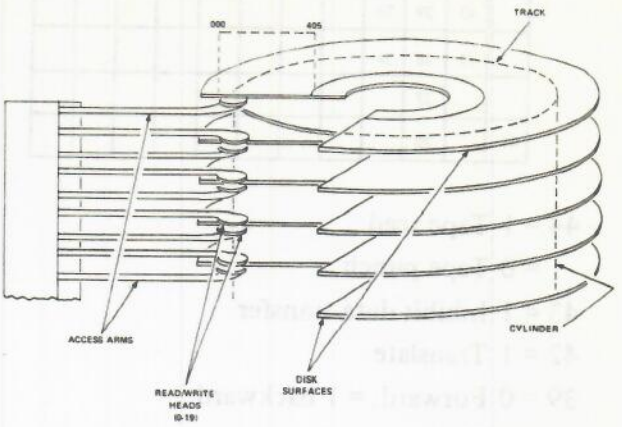


Figure 10-31. Disk-Pack Recording Surfaces

The data transfer is bit-serial. The maximum byte capacity, transfer rate, and other pertinent information for the various disk-pack subsystems are presented in table 10-6. Figures 10-32 and 10-33 delineate the disk pack I/O control word and the disk-pack result descriptor, respectively.

Table 10-6.

Disk-Pack Subsystem Characteristics

DISK PACK DRIVE STYLE NO.	DESCRIPTION	AVERAGE ACCESS TIME (MS)	AVERAGE LATENCY (MS)	STORAGE CAPACITY PER DISK-PACK DRIVE		DATA TRANSFER RATE	PACK DATA		
				MULTI-SECTOR MODE*	FULL TRACK MODE*		MAX. RECORDING DENSITY (BPI)	TRACK DENSITY (TPI)	DISK-PACK STYLE NO.
B 9484-3	Dual drive with single access disk-pack drive controller (B 6380-1)	30	12.5	95.5	121.0	312.5 KB	2200	200	B 9974-1
B 9485-3	Dual drive with simultaneous access disk-pack drive controller (B 6380-2)	30	12.5	95.5	121.0	312.5 KB	2200	200	B 9974-1
B 9486-3	Dual drive add on increment without disk-pack drive controller	30	12.5	95.5	121.0	312.5 KB	2200	200	B 9974-1
B 9484-4	Dual drive wtih single access disk-pack drive controller (B 6383-1)	30	12.5	174.4	242.0	625.0 KB	4400	200	B 9974-4
B 9485-4	Dual drive with simultaneous access disk-pack drive controller (B 6383-2)	30	12.5	174.4	242.0	625.0 KB	4400	200	B 9974-4
B 9486-4	Dual drive add on increment without disk-pack drive controller	30	12.5	174.4	242.0	625.0 KB	4400	200	B 9974-4
B 9486-45	Add on increment without disk-pack drive controller	30	12.5	87.2	121.0	625.0 KB	4400	200	B 9974-4

*Million eight-bit bytes

	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24
WRITE	0	0	0	1	0	0	0	Ø	0	0	F_1	V_8	V_4	V_2	V_1	0	0	S_8	S_4	S_2	S_1
READ	1	0	0	1	Ø	0	0	Ø	Ø	0	F_1	V_8	V_4	V_2	V_1	0	0	S_8	S_4	S_2	S_1
TEST	0	0	0	0	0	0	1	0	0	0	F_1	0	0	0	0	0	0	V_8	V_4	V_2	V_1
INITIALIZE	0	0	0	1	0	0	0	0	0	1	F_1	V_8	V_4	V_2	V_1	0	0	S_1	S_4	S_2	S_1
VERIFY	1	0	0	1	Ø	0	0	0	0	1	F_1	V_8	V_4	V_2	V_1	0	0	S_1	S_4	S_2	S_1
RELOCATE	0	1	0	1	0	0	0	0	0	0	F_1	N_8	N_4	N_2	N_1	0	0	S_8	S_4	S_2	S_1

F=0 standard format (33 sectors per track) F=1 single sector per track format

WRITE:

- $S_1 = 1$ Initiate conditional seek
- $S_1 = 0$ Initiate unconditional seek
- $S_2 = 1$ Disable automatic restore function following a seek run condition
- $S_4 = 1$ Execute a parity check on all sectors written upon completion of write operation
- $V_4 = 1$ Reserved for file protect memory (FPM)
- $V_8 = 1$ Enable EBCDIC-ASCII translator

READ:

- $S_1 = 1$ Initiate conditional seek
- $S_1 = 0$ Initiate unconditional seek
- $S_4 = 1$ Read binary address field only into memory address specified by begin memory address (A).
- $S_4 = 0$ Initiate normal read
- $S_2 = 1$ Disable automatic restore function following a seek error condition
- $S_8 = 1$ Disable error correction
- $V_1 = 1$, $V_2 = 1$ Reserved for FPM
- $V_8 = 1$ Enable EBCDIC-ASCII translator.

TEST:

- $V_1 = 1$ Power down (take off line) the selected drive for pack removal

INITIALIZE:

- $V_4 = 1$ Write test data pattern in each sector as specified at the begin address
- $V_1 = 0$, $V_2 = 0$ Initialize entire pack
- $V_1 = 1$ Initialize designated cylinder
- $V_2 = 1$ Initialize designated track

VERIFY:

- $V_1 = 0$, $V_2 = 0$ Verify entire pack and terminate on first error encountered
- $V_1 = 1$ Verify and report all errors within the designated cylinder
- $V_1 = 0$, $V_2 = 1$ Verify and report all errors within the designated track
- $V_4 = 1$ Verify data bits by comparing with 16-bit data pattern beginning at the begin address
- $V_8 = 1$, $V_4 = 0$ Verify test data pattern within an initialize
- $S_2 = 1$ Disable automatic restore function following a seek error condition

RELOCATE:

- $N = 1 \Rightarrow 5$ Spare sector in 28 through 32 on the designated cylinder

Figure 10-32. Disk-Pack I/O Control Word (IOCW)

Result Descriptor

A result descriptor is generated by the controller at the completion of each I/O operation. This descriptor is stored in a fixed location of reserved memory dependent on the I/O channel that is being used.

NOTE

An automatic restore function (restore heads to cylinder 000) is normally performed on all I/O operations when either a seek time-out or seek error condition occurs.

The format of the result descriptor is:

Bits	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

The bit assignments for the result descriptor are:

BITS SET	DESCRIPTION
1	Operation complete
2	Exception condition
3	Disk-pack not ready or "unsafe"
3, 4	Control cleared during operation
4	Data error (data error on read, or memory parity error on write)
4, 5	Memory access error
4, 5, 9	Transmission parity error
4, 8	Memory interface parity error
4, 9	Speed error
4, 10	Reserved (FPM)
5	Address parity error
5, 6	First action with drive
5, 7	Write lockout
5, 8	Sector time-out (see note above)
5, 9	Address position error (verify)
5, 10	Reserved (FPM)
6	Drive seeking
6, 7	Invalid command descriptor
6, 8	Seek initiated
7, 8	Single bit error correction
8	Seek error (see note above)
8, 9	Seek time-out
9	Disk-pack drive busy (time-out)
9, 10	Reserved (FPM)
10, 11, 12	Disk-pack drive identification during test operations
15, 16	Processor identification (test)
13, 14, 15, 16	Unit designate (all operations except test)

Figure 10-33. Disk-Pack Result Descriptor Format

SECTION 11

B 6700 DATA COMMUNICATIONS SYSTEM

GENERAL

The B 6700 Data Communications System is comprised of one or more of each of the following units:

1. Data Communications Processor (DCP).
   Each B 6700 I/O Processor accommodates up to four DCP's through the word interfaces. The word interfaces provide access to the B 6700 main memory.

2. 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 DCP. It is also possible to connect an Adapter Cluster between two DCP's. This allows the Adapter Cluster to be serviced from either DCP.

3. 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 6700 Data Communications System can service a maximum of 2048 communications lines. A typical system configuration with only two processors and two I/O processors illustrated is shown in figure 11-1.

DATA COMMUNICATIONS PROCESSOR (DCP)

The Data Communications Processor (DCP) 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.

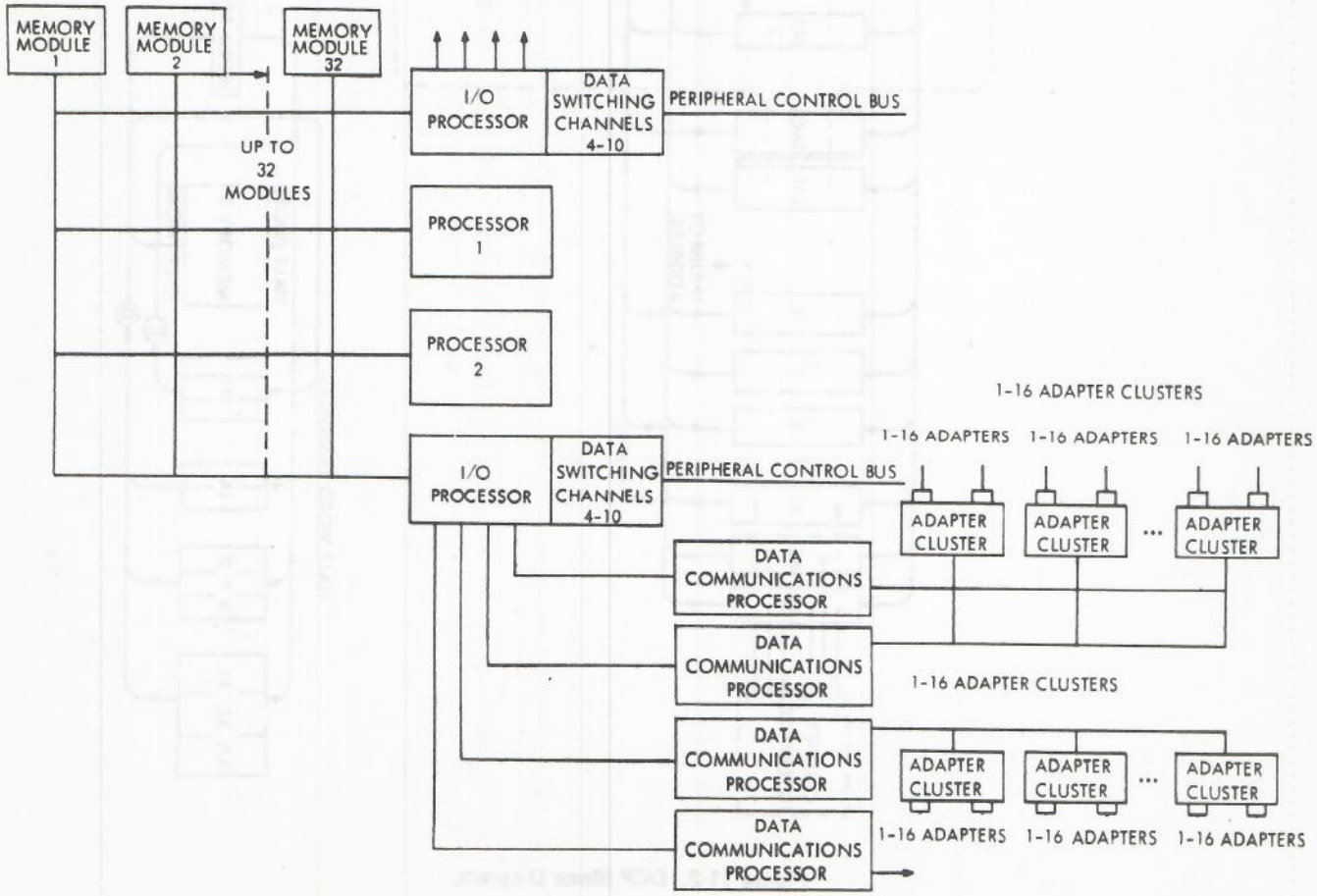


Figure 11-1. B 6700 System Configuration Including Data Communications.

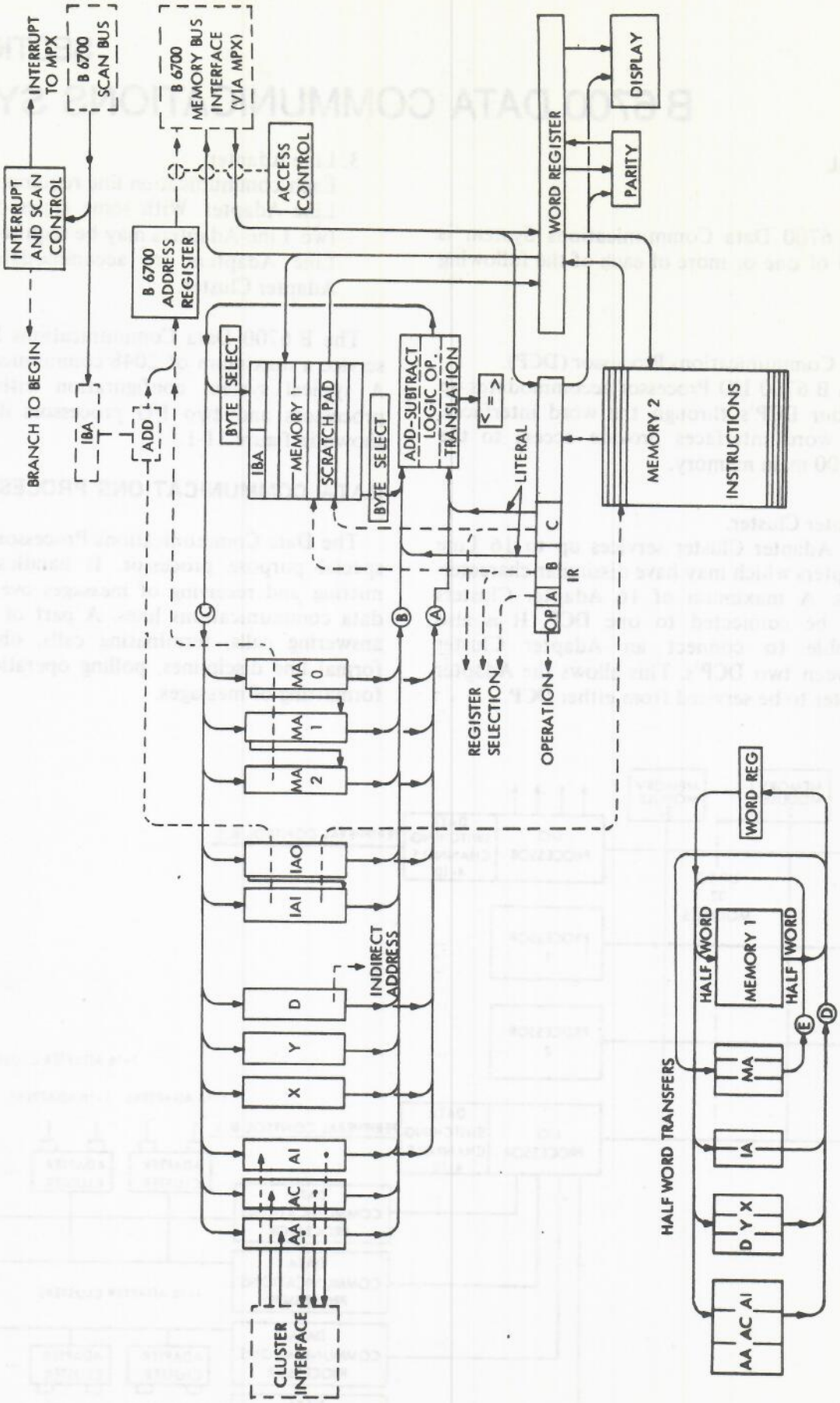


Figure 11-2. DCP Block Diagram.

The DCP is a stored program computer which obtains its program instructions either from B 6700 main memory or from an optional local memory. Through the use of the local memory the throughput of the DCP is significantly increased due to the reduction in instruction fetch time.

If the optional local memory is not present, the DCP shares the B 6700 system main memory with the other units of the B 6700. Memory allocation for the DCP is controlled by the B 6700 Master Control Program. Data exchanges occur when the B 6700 processor initiates a DCP operation and when the DCP finishes an operation, i.e., I/O complete signal from the DCP.

The internal form of the DCP is shown in figure 11-2. The DCP is an elementary micro-programed processor. Two-address and three-address instructions, operating on eight-bit bytes, are used by the DCP. The byte organization fits into a basic half-word (three byte) structure permitting efficient half-word transfers within the DCP. The functions of the DCP are accomplished with a small array of intercommunicating registers, a simple arithmetic-logical unit and an eight-word scratch pad memory.

For complete information on all DCP registers and memories, refer to the Data Communications Processor Reference Manual (form 1054384).

ADAPTER CLUSTER

The Adapter Cluster is the interface between the DCP and the data-communication Line Adapters. Each Adapter Cluster services up to 16 Line Adapters. Data transmission rates of from 45.5 to 4800 BPS are handled simultaneously by the Adapter Cluster.

Figure 11-3 shows a block diagram of the Adapter Cluster. The Adapter Cluster basic functions are:

1. Line termination: scanning, clocking and temporary storage.

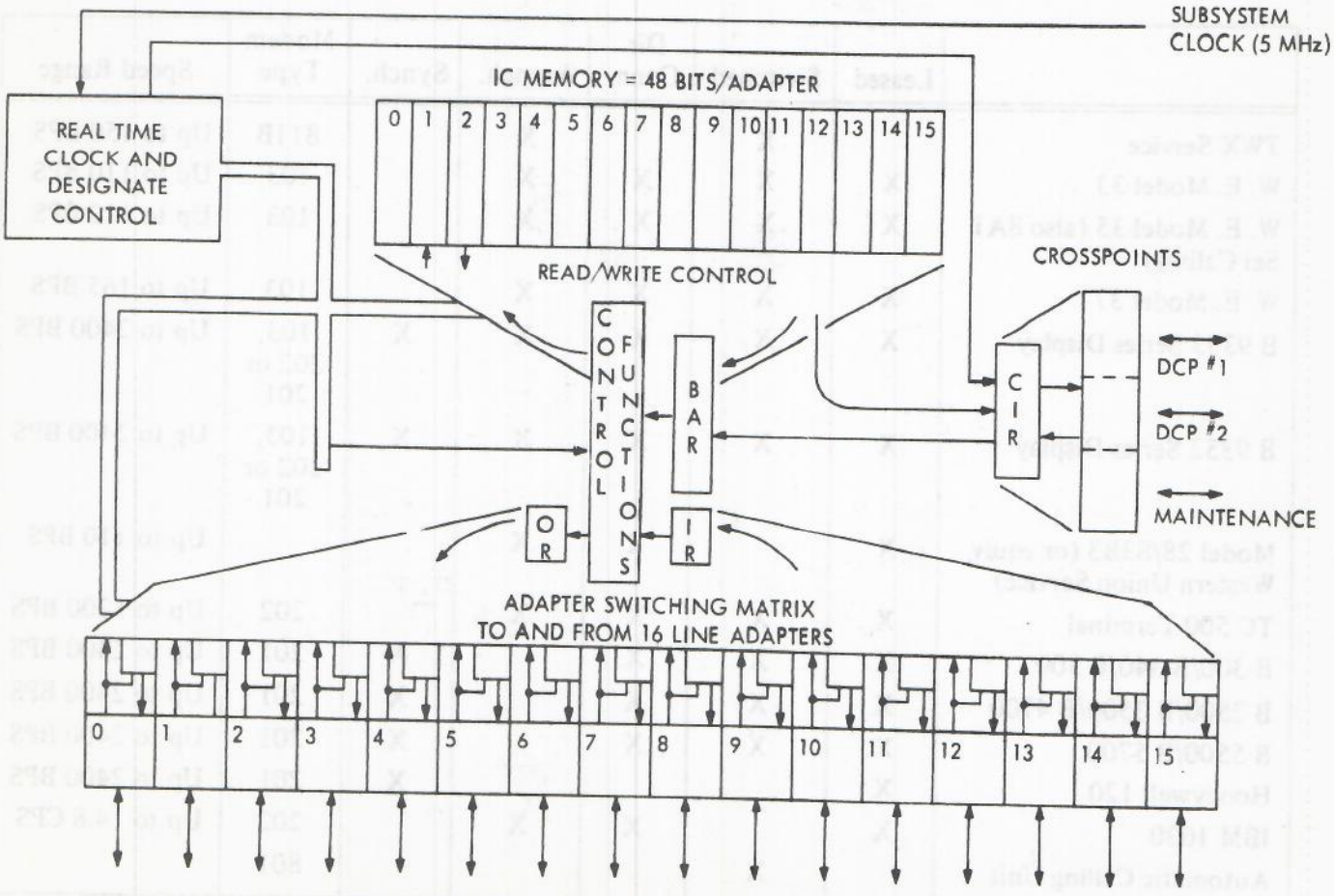


Figure 11-3. Adapter Cluster.

2. Character assembly and disassembly.
3. Synchronization attainment and maintenance.
4. Timer operation to maintain line discipline.
5. Some character recognition. (Mainly synchronization characters for the various line disciplines.)
6. Information exchange with one or two DCP's.

The Adapter Cluster functions in a manner that makes it appear transparent to most characters and message formats. However, as stated in item (5) above it does recognize the synchronization 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 DCP 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 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 (form 1054384).

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 9353 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
B 300/B 340/B 500	X	X	X		X	201	Up to 2400 BPS
B 2500/B 3500/B 4700	X	X	X		X	201	Up to 2400 BPS
B 5500/B 5700	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	

SECTION 12
DISK FILE OPTIMIZER

GENERAL

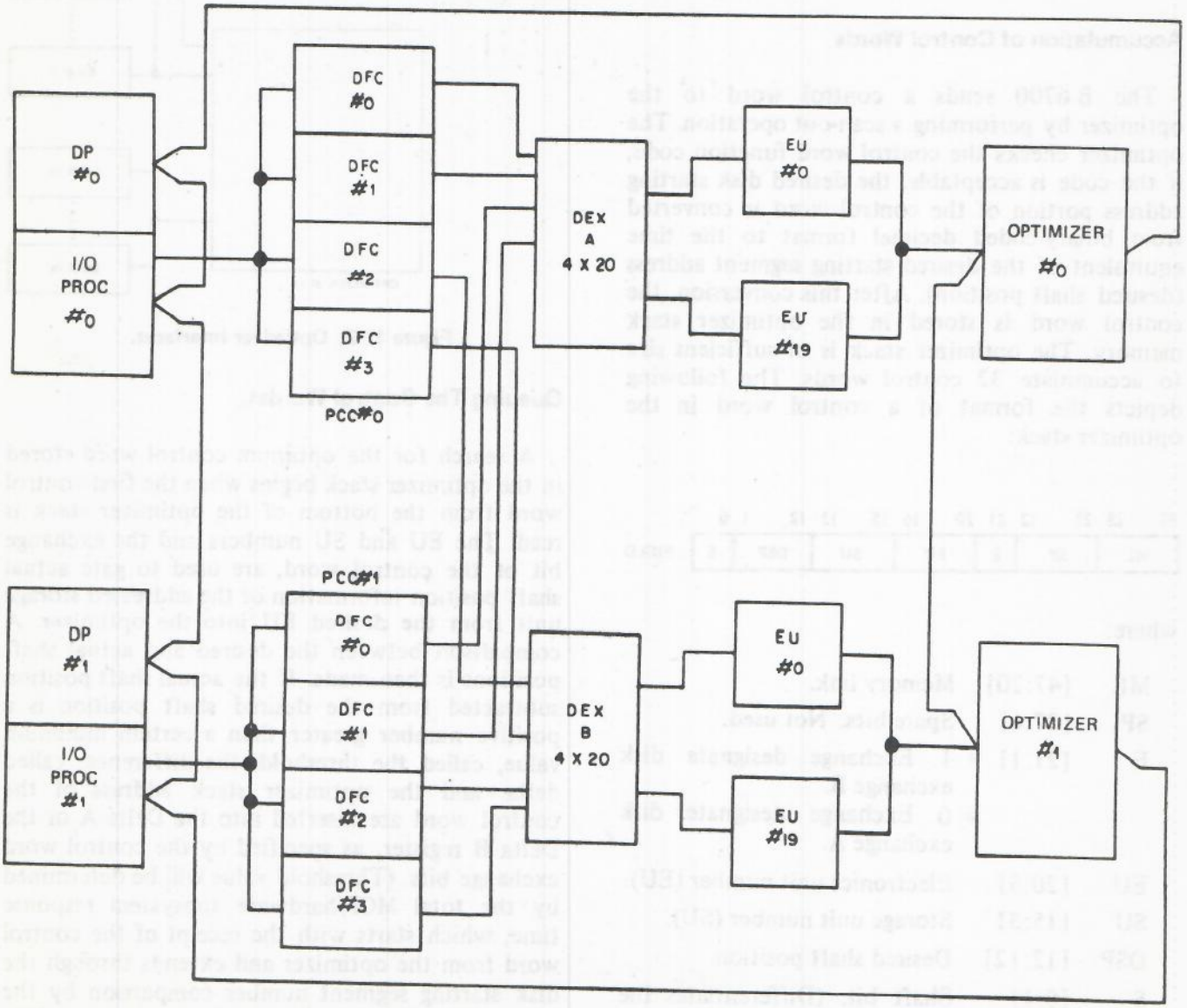
This section describes the functional characteristics for the Disk File Optimizer used with the B 6700 Information Processing Systems. The optimizer functions to optimize the transfer of information between a processor of the B 6700 System and its associated disk file subsystem in order to improve the transfer rate. The optimizer communicates with the B 6700 via the scan bus, and with the disk file subsystem directly, or indirectly via another optimizer. Figure 12-1 shows the relationship of the optimizer to the other units in the B 6700 System.

FUNCTIONAL CHARACTERISTICS

Functional Performance Characteristics

The basic functions of the optimizer are:

1. Accumulate control words from the B 6700 with each control word representing a disk transfer operation.
2. Select the optimum control word from among the accumulated control words to minimize access time to the disk file (queuing).
3. Transmit the optimum control word to the B 6700 upon request.



NOTE: For illustrative purposes only, a two processor, two I/O processor system is shown.

Figure 12-1. The Optimizer in the B 6700 System.

The function of the optimizer is to optimize access time for the disk file controllers sharing the B 6700. Figure 12-2 depicts all of the devices that interface with the optimizer.

Components

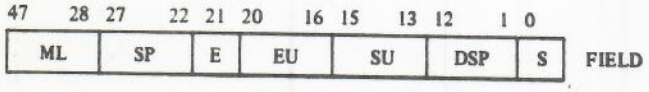
The optimizer consists of the following functional units:

1. Input/Output (I/O) Interface Unit.
2. Disk Address Unit.
3. Queuing Unit.
4. MDL Interface Unit.

OPERATIONAL CHARACTERISTICS

Accumulation of Control Words

The B 6700 sends a control word to the optimizer by performing a scan-out operation. The optimizer checks the control word function code; if the code is acceptable, the desired disk starting address portion of the control word is converted from binary-coded decimal format to the time equivalent of the desired starting segment address (desired shaft position). After this conversion, the control word is stored in the optimizer stack memory. The optimizer stack is of sufficient size to accumulate 32 control words. The following depicts the format of a control word in the optimizer stack:



where:

ML	[47:20]	Memory link.
SP	[27:6]	Spare bits. Not used.
E	[21:1]	= 1 Exchange designate disk exchange B. = 0 Exchange designate disk exchange A.
EU	[20:5]	Electronics unit number (EU).
SU	[15:3]	Storage unit number (SU).
DSP	[12:12]	Desired shaft position.
S	[0:1]	Shaft bit. (Differentiates the two shafts of a storage unit, when applicable.)

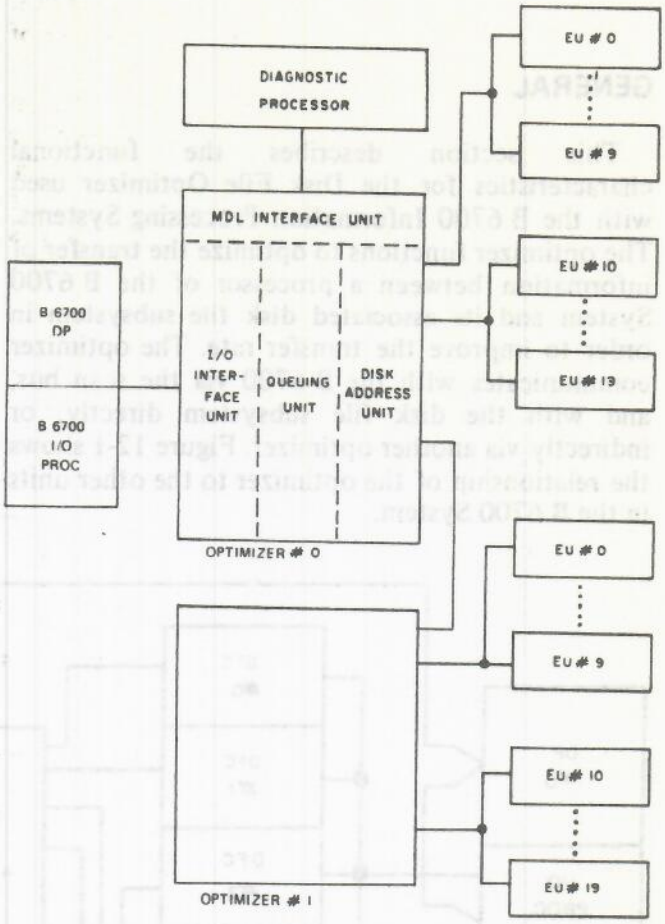


Figure 12-2. Optimizer Interfaces.

Queuing The Control Words

A search for the optimum control word stored in the optimizer stack begins when the first control word from the bottom of the optimizer stack is read. The EU and SU numbers and the exchange bit of the control word, are used to gate actual shaft position information of the addressed storage unit from the desired EU into the optimizer. A comparison between the desired and actual shaft positions is then made. If the actual shaft position subtracted from the desired shaft position is a positive number greater than a certain minimum value, called the threshold, the difference, called delta, and the optimizer stack address of the control word are inserted into the Delta A or the Delta B register, as specified by the control word exchange bits. (Threshold value will be determined by the total MCP/hardware subsystem response time, which starts with the receipt of the control word from the optimizer and extends through the disk starting segment number comparison by the controller.) The other control words stored in the optimizer stack undergo the same process

sequentially from the bottom of the stack. As each word is processed, if a delta is generated which is smaller than the contents of the appropriate Delta register, and greater than the threshold, the smaller delta replaces the larger one in the register. The optimizer continuously scans through its stack of accumulated control words, when it is not engaged in control word transfers at the I/O processor interface. A complete pass through the list of accumulated control words constitutes a stack scan cycle.

Stack Operation

A control word suitable for execution by the disk system (a queued control word) is not sent to the I/O processor until the optimizer has completed at least one full scan through its memory stack since the last scan-in or scan-out operation. If, during the middle of a scan cycle, a scan-in operation (referencing an exchange) occurs, the optimizer restarts the scan cycle at the beginning and does not send a queued control word (referencing the same exchange) until at least one full scan cycle has been completed. If, however, during the middle of a scan cycle, a scan-out operation (referencing an exchange) occurs, the optimizer continues that scan cycle and does not send a queued control word (referencing the same exchange) until at least that scan cycle has been completed.

Stack Erasure And Compression

When a control word is transmitted to the I/O processor, this word is erased from the optimizer stack. Erasure is accomplished by transferring the word at the top of the optimizer stack into the location marked for erasure. Th op-of-the-stack pointer is then decremented by on

Optimizer Dump

The optimizer has the capability of executing a read top-of-the-stack instruction. The Optimizer Dump function is implemented by successive applications of this instruction. When the function code defines a read top-of-stack instruction, this then becomes the next operation. Then, whether the referenced EU is busy or not, the command word located at the top of the optimizer stack is transmitted to the I/O processor, unless a malfunction is to be reported.

Degraded Mode Operation

If one optimizer or I/O processor fails, optimizing responsibility for the disk file subsystem will be assumed by the remaining optimizer and I/O processor. Thus, the following situation may prevail:

1. Access to any disk file will still be possible via the remaining optimizer/I/O processor pair (optimizer or I/O processor failure).
2. The remaining optimizer will be able to queue jobs involving any of the disks (optimizer failure).
3. The remaining I/O processor will continue to be able to transfer control words to and from either optimizer (I/O processor failure).

EU Conflict Resolution

Logic exists to prevent both optimizers of an optimizer pair from accessing the same EU bus simultaneously. The optimizer-to-optimizer signals required to implement this conflict resolution are described later in this section.

INTERFACE REQUIREMENTS

Interface With The I/O Processor

The I/O processor communicates with the optimizer for the following reasons:

1. To send a control word to the optimizer (store control word request).
2. To request a control word selected by the optimizing process to govern the execution of the next disk transfer (optimized control word request).
3. To receive an optimized control word from the top of the optimizer stack (top-of-the-stack control word request).
4. To clear the optimizer stack (clear-the-stack request).

The I/O processor interface communications are accomplished with a scan-out sequence for output operations and with a scan-in sequence for input operations.

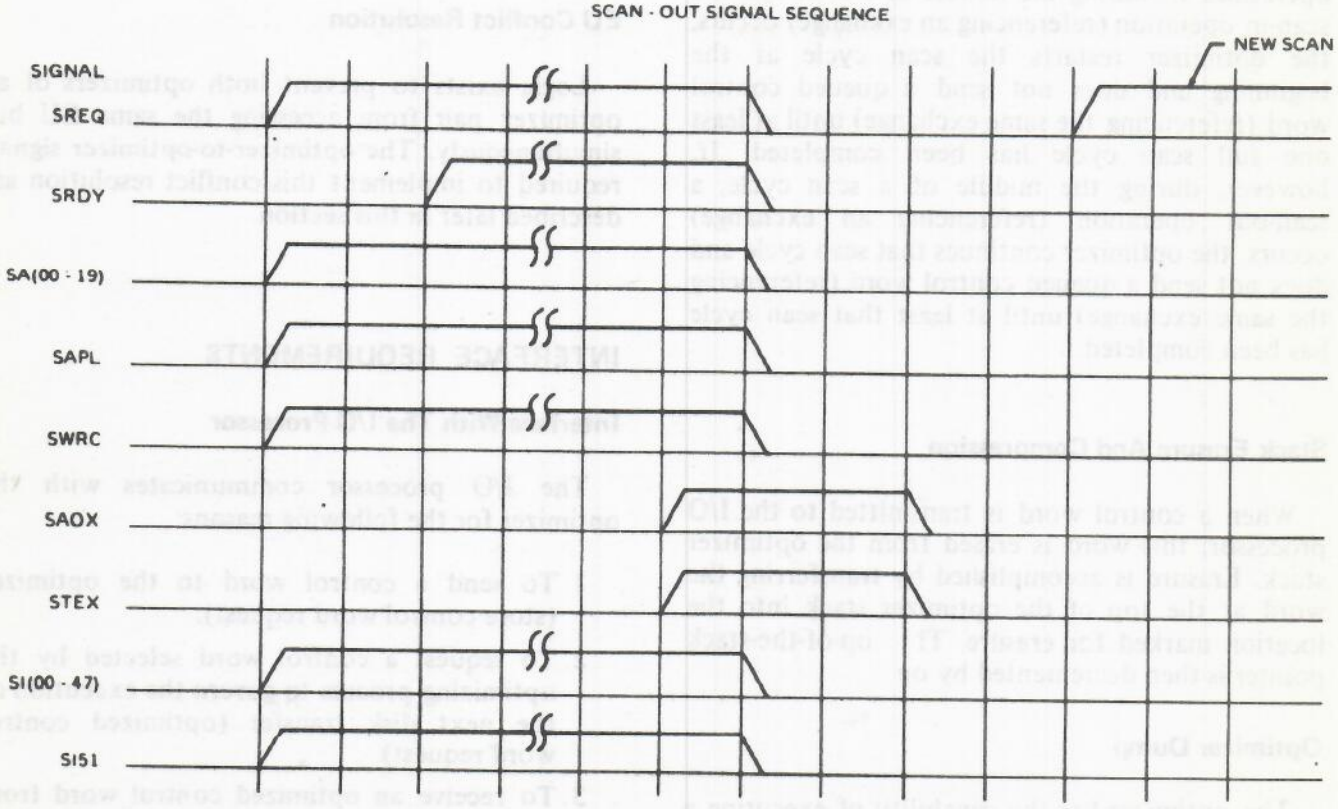
Control Word

Each control word sent to the optimizer contains the following information:

1. Desired Disk Starting Address. Eight bits of the desired disk starting address are used to define the desired exchange and EU. The remaining 26 bits are used to define the desired SU, shaft (if applicable), face, zone, track and segment.
2. Function Code. The function code, together with the Scan Write Control (SWRC) signal, is used to define one of the operations.
3. Memory Link. The memory link points to an address in main memory wherein the disk operation is defined. This address is returned to the I/O processor and identifies the next disk operation to be performed.

Scan-Out

When the MCP has a control word for the optimizer, the MCP initiates the scan-out sequence by making the Scan Write Control (SWRC) line come true, and it then sends a Scan Request (SREQ) signal to the optimizer as indicated in figure 12-3. If the optimizer stack is not full, the optimizer responds by making its Scan Ready (SRDY) signal come true. At this time, the information being sent to the optimizer is available on the interface lines: 20 bits are transferred on the Scan Address SA (00 thru 19) lines, and 48 bits are transferred on the Scan Information SI (00 thru 47) lines. Two odd parity bits accompany the signals received from the I/O processor: Scan Address Parity Level (SAPL) for signals SA (00 thru 19), SREQ, and SWRC; and Scan Information Parity Bit (SI51) for signals SI (00 thru 47). The optimizer indicates receipt of the signal by making



NOTE

a. SREQ must be off at least one clock between scans (after the fall of SOAX).

b. SRDY must be held on for at least one clock after SAOX is turned on.

c. Since the system in asynchronous, the processor may recognize SAOX and drop its signals later than shown.

Figure 12-3. Scan-Out Signal Sequence.

the Scan Access Obtained (SAOX) signal come true. If the optimizer detects a parity error during transmission of the control word, it will make the Scan Transmission Error (STEX) signal come true.

Scan-In

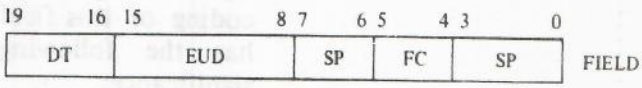
When the MCP requests a control word from the optimizer, the MCP initiates the scan-in sequence by keeping the Scan Write Control (SWRC) line low, while sending a Scan Request (SREQ) signal to the optimizer, as indicated in figure 12-4. The optimizer responds by raising its Scan Ready (SRDY) signal. At this time, control information is transferred to the optimizer over the 20 Scan Address lines SA (00 thru 19), and a Parity signal (SAPL) is sent to the optimizer to maintain odd parity on signals SA (00 thru 19), SREQ, and SWRC. The optimizer responds by generating a Scan-In word, the contents of which are determined by the status controls, together with an odd parity signal for this word (SI51) and raises the Scan Access Obtained (SAOX) signal to inform the I/O processor that the control word is available on the interface lines. In addition, if the optimizer detected a parity error during the transmission of control information over the 20 SA (00 thru 19) lines, it makes the Scan Transmission Error (STEX) signal come true at this time.

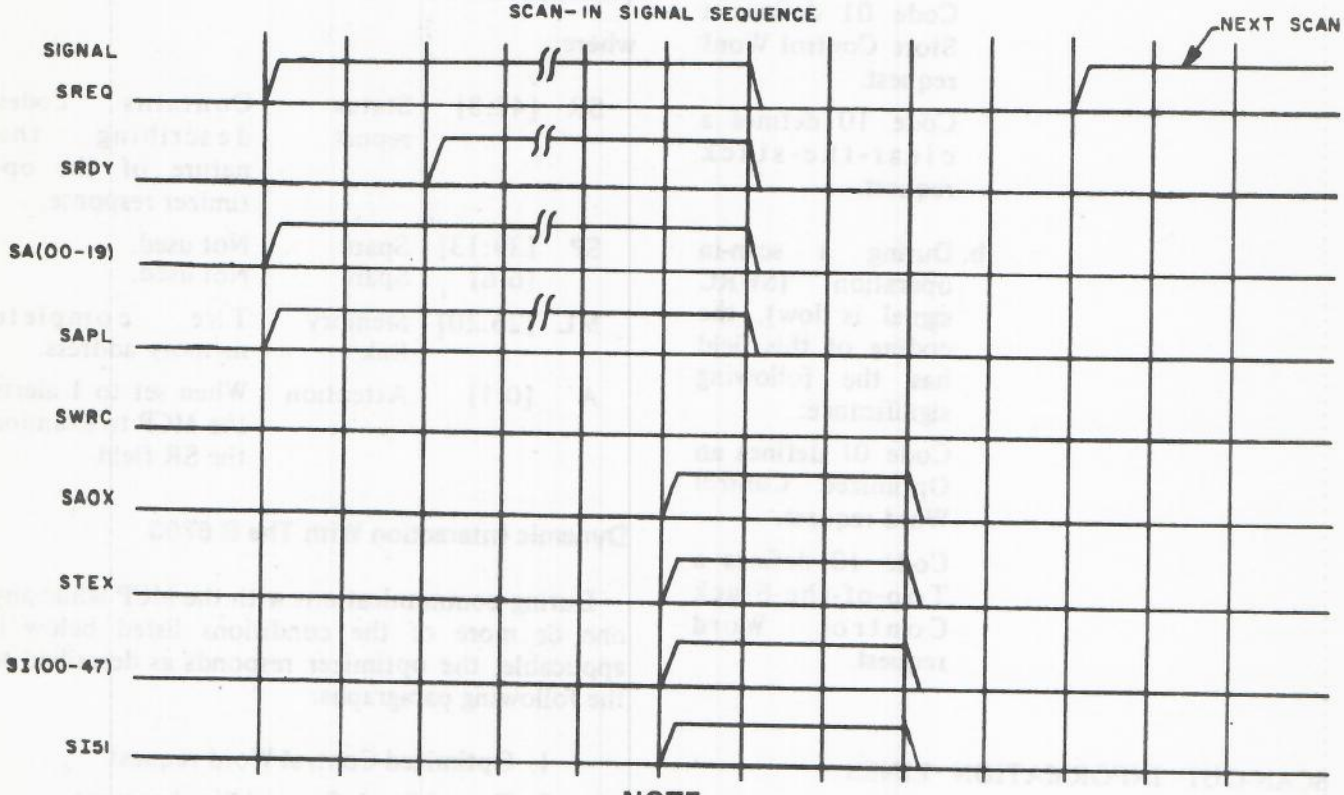
Scan Bus Data Format

Data is transferred on the scan bus over the uni-directional (I/O processor to optimizer) Scan Address lines and over the bi-directional Scan Information lines.

SCAN ADDRESS LINES (SA)

The following is the B 6700 Scan Address line word format:





NOTE

a. SREQ must be off at least one clock between scans (after the fall of SOAX).

b. SRDY must be held on for at least one clock after SAOX is turned on.

c. Since the system is asynchronous, the processor may recognize SAOX and drop its signals later than shown.

Figure 12-4. Scan-In Signal Sequence.

where:

DT	[19:4]	Device Type	Code 1001 selects the optimizer.
EUD	[15:8]	Electronic Unit Designate	Defines the exchange and the EU number associated with the job.
SP	[7:2]	Spare	Not used.
	[3:4]	Spare	Not used.
FC	[5:2]	Function Code	Defines the operation requested by the I/O processor as follows:

a. During a scan-out operation (SWRC signal is high), the coding of this field has the following significance:

Code 01 defines a Store Control Word request.

Code 10 defines a clear-the-stack request.

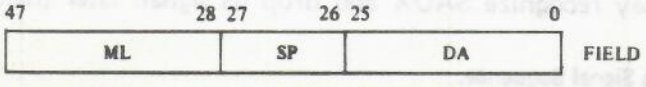
b. During a scan-in operation (SWRC signal is low), the coding of this field has the following significance:

Code 01 defines an Optimized Control Word request.

Code 10 defines a Top-of-the-Stack Control Word request.

SCAN-OUT INFORMATION LINES

The word format of the B 6700 scan-out information lines is as follows:

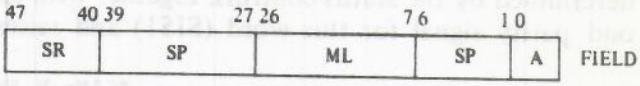


where:

ML	[47:20]	Memory link	Defines the complete memory address. Bit 47 is the most-significant bit.
SP	[27:2]	Spare	Not used.
DA	[25:26]	Disk Address	Defines the six BCD characters plus the two expansion bits of the desired disk starting address, not including the desired EU or desired disk file exchange (DEX).

SCAN-IN INFORMATION LINES (SI)

The word format of the B 6700 scan-in information lines is as follows:



where:

SR	[47:8]	Status report	Contains codes describing the nature of the optimizer response.
SP	[39:13]	Spare	Not used.
	[6:6]	Spare	Not used.
ML	[26:20]	Memory link	The complete memory address.
A	[0:1]	Attention	When set to 1 alerts the MCP to examine the SR field.

Dynamic Interaction With The B 6700

During communications with the MCP while any one or more of the conditions listed below is applicable, the optimizer responds as described in the following paragraphs:

1. Optimized Control Word request.
2. Top-of-Stack Control Word request.
3. Store Control Word request.
4. Clear-the-Stack request.
5. First stack scan cycle incomplete.
6. Arithmetic Address Converter (AAC) busy.

7. No access to the Optimizer Exchange (OEX).
8. SU not available.
9. Optimizer Stack (OS).
10. Control Word (CW) not available.
11. Scan bus parity error.
12. OS parity error.
13. Disk address error.
14. OS full.

OPTIMIZED CONTROL WORD REQUEST

If the MCP requests a queued control word (referencing an exchange) during a scan-in operation and the optimizer has an optimized control word (referencing the same exchange) ready for transmission, the optimizer responds with the memory location of that control word and an appropriate status code. (The optimizer obtains the memory location by reading a control word from the OS, from the location determined by the appropriate Delta register at the topmost filled position of the memory stack and the stack scan-information (SI) register, DAR or DBR.)

TOP-OF-STACK CONTROL WORD REQUEST

If the MCP requests the control word located at the topmost filled position of the OS and the stack is not empty, the optimizer responds with the memory location of that control word and an appropriate status code.

STORE THE CONTROL WORD REQUEST

If the MCP requests a control word be loaded into the OS (via the scan-out operation), the optimizer responds by accepting that control word, sends the DA portion of the control word to the AAC section for processing (unless unable to do so because of an error condition), and then stores the control word in the OS in the format indicated above.

CLEAR-THE-STACK REQUEST

If the MCP requests the OS be cleared (via a scan-out operation), the optimizer sets TSR to the position indicating an empty optimizer stack location, thus effectively erasing the OS. All error-detecting flip flops previously set are reset.

FIRST STACK SCAN CYCLE INCOMPLETE

If the MCP initiates an Optimized Control Word request (referencing an exchange) prior to the completion of a full stack scan cycle since the last scan bus operation (referencing the same exchange) with that optimizer, the optimizer answers the SREQ signal of the I/O processor with a SRDY signal, and then waits until completion of a scan cycle before sending an Optimized Control Word accompanied by an SAOX signal.

ARITHMETIC ADDRESS CONVERTER (AAC) BUSY

If the MCP performs a scan-out operation while the AAC section of the optimizer is still busy converting the control word it received during the previous scan-out operation, the optimizer responds to the SREQ of the I/O processor signal with an SRDY signal. The optimizer then waits until completion of the address conversion before accepting the new control word along with the SAOX signal.

NO ACCESS TO OEX

If the optimizer does not receive a strobe (accompanying the time equivalent of angular shaft position) from an EU, in response to a request for shaft position information, the optimizer immediately stops optimizing and responds in the following manner. If the next scan bus operation is a Store Control Word request, the optimizer will process the new control word in the normal manner and load it into the optimizer stack. Subsequent Store Control Word requests (if any) are similarly processed. A Clear-the-Stack request is also honored, if it occurs. However, when a request for an Optimized Control Word or a Top-of-Stack Control Word takes place (and a Clear-the-Stack request has not occurred), the optimizer responds with the memory location of the control word associated with the No-Access-To-OEX error and an appropriate status code. Optimizing then resumes.

SU NOT AVAILABLE

If, during optimizing, in response to a request for shaft position information, the optimizer does not receive an SU Ready level, the optimizer immediately halts and the following occurs. If the next scan bus operation is a Store Control Word request, the optimizer processes the new control

word in the normal manner and loads it into the optimizer stack. Subsequent Store Control Word requests (if any) are similarly processed. A Clear-the-Stack request is also honored, if it occurs. However, when a request for an Optimized Control Word or a Top-of-the Stack Control Word takes place (and a Clear-the-Stack request has not occurred), the optimizer responds with the memory location of the control word associated with the SU Not Available error and an appropriate status code. Optimizing then resumes.

OPTIMIZER STACK (OS) EMPTY

If the MCP requests an optimized control word, referencing an exchange, and the OS does not contain control words referencing the same exchange; or if the MCP requests the control word from the top-of-the-stack and the OS is completely empty (contains no control words), the optimizer responds with a memory location of all 0's and an appropriate status code.

CONTROL WORD NOT AVAILABLE

If, after completion of a full stack scan cycle, the MCP requests an optimized control word (referencing an exchange), and one is unavailable for transmission because all of the control words stored in the OS (referencing the requested exchange) reference EU's which are busy, the optimizer returns a memory link of all 0's and an appropriate status code (unless an error condition is to be reported).

SCAN BUS PARITY ERROR

If, during a scan-out operation, the optimizer detects a parity error on either the Scan Address lines or the Scan Information lines, the optimizer responds with an SRDY signal and then an SAQX signal accompanied by an STEX signal. The optimizer loads the received control word into the OSR and ignores it; that is, the new control word will not be sent to the AAC for processing nor will it be loaded into the optimizer stack. If, during a Scan-In operation, the optimizer detects a parity error on the Scan Address lines, the optimizer responds with a SRDY signal, then an SAOX signal accompanied by a STEX signal, and by a memory link of all 0's.

OPTIMIZER STACK (OS) PARITY ERROR

The optimizer generates a parity bit for each control word loaded into the OS. Parity is checked whenever a control word is read from the OS. If an OS parity error is detected, optimizing is halted immediately and the optimizer responds in the following manner: If the next scan bus operation is a scan-out, the optimizer will process the new control word in the normal manner and load it into the OS. Subsequent scan-out operations, if any, are similarly processed. A Clear-the-Stack request will be honored, if it occurs. However, when a request for an Optimized Control Word or Top-of-Stack Control Word takes place (and a Clear-the-Stack request has not occurred), the optimizer responds with the memory link of the control word associated with the error and an appropriate status code. Optimizing then resumes.

NOTE

The memory link may not be valid at this time.

DISK ADDRESS ERROR

The AAC section of the optimizer checks if the EUD or DA portions of the last received control word contains one of the following error conditions:

1. The Electronic Unit Designate (EUD) or Disk Address (DA) portion of the control word is not properly coded (BCD), either when it is received by the AAC or subsequently during the conversion process.
2. During conversion, it is determined that the number of SU's exceeds five, or the number of faces exceeds eight, or the number of tracks exceeds 50.

Then, the optimizer immediately halts address conversion and optimizing. Consequently this control word is not loaded into the optimizer stack, and the optimizer responds in the following manner: If the next scan bus operation is a Store Control Word request, the optimizer will not respond (that is, the SRDY signal is low). A Clear-the-Stack request will be honored, if it occurs. However, when a request for an Optimized Control Word takes place (and a Clear-the-Stack request has not occurred), the optimizer responds with the memory link of the control word associated with the Disk Address Error and an appropriate status code.

OPTIMIZER STACK FULL

If the MCP inaugurates a Store Control Word request scan-out operation (by sending the SREQ and SWRC signals with the appropriate function code), and the optimizer stack is full, the optimizer does not respond; that is, it keeps the SRDY signal low.

DISK INTERFACE

The disk file subsystem (DFS) consists of electronics units (EU's) and their associated storage units (SU's). Each optimizer has the capability to communicate directly with up to 20 EU's associated with one disk file exchange (by means of two 10-EU busses); each optimizer can also communicate indirectly with up to 20 EU's associated with another disk file exchange via another optimizer, as depicted in figure 12-5. In normal operation, each optimizer is restricted to direct communication with its associated 20 EU's but each optimizer does have the ability to access all 40 EU's (on a pair of disk exchanges), if necessary.

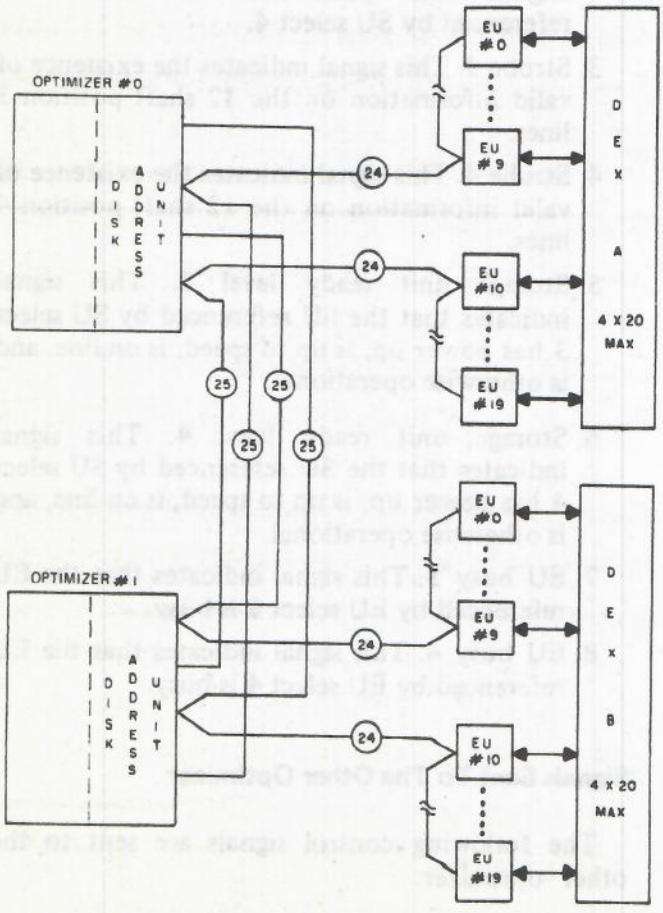


Figure 12-5. The Disk File Subsystem (DFS) Interface.

The following signals, which the optimizer sends and receives on the disk file subsystem interface, are also shown in figure 12-6:

1. Signals sent directly to the DFS.
2. Signals sent to the DFS via the other optimizer.
3. Control signals sent to the other optimizer.
4. Signals received directly from the DFS.
5. Signals received from the DFS via the other optimizer.

Signals Sent Directly To The Disk File Subsystem

The following signals are sent directly to the disk file subsystem:

1. Select 1. This signal enables communication between the optimizer and the first set of 10 EU's on the disk file exchange normally associated with this optimizer. The EU's use this signal to gate out information from the desired SU to the optimizer.
2. Select 2. This signal enables communication between the optimizer and the second set of 10 EU's on the exchange normally associated with this optimizer. The EU's use this signal to gate out information from the desired SU to the optimizer.
3. EU select 1. These signals are transmitted over four lines to define one of 10 EU's designated by select 1.
4. SU select 1. These signals are transmitted over four lines to define one of five SU's in the EU referenced by EU select 1, and one of two shafts, when applicable.
5. EU select 2. These signals are transmitted over four lines to define one of 10 EU's designated by select 2.
6. SU select 2. These signals are transmitted over four lines to define one of five SU's in the EU referenced by EU select 2, and one of two shafts, when applicable.

Signals Received Directly From The Disk File Subsystem

The following signals are received directly from the disk file subsystem:

1. Shaft position 1. This is the output of a 12-bit

counter containing the time equivalent of the angular shaft position of the desired SU referenced by SU select signal 1.

2. Shaft position 2. This is the output of a 12-bit counter containing the time-equivalent of the angular shaft position of the desired SU referenced by SU select signal 2.

3. Strobe 1. This signal indicates the existence of valid information on the 12 shaft position 1 lines.

4. Strobe 2. This signal indicates the existence of valid information on the 12 shaft position 2 lines.

5. Storage unit ready level 1. This signal indicates that the SU referenced by SU select 1 has power up, is up to speed, is on-line, and is otherwise operational.

6. Storage unit ready level 2. This signal indicates that the SU referenced by SU select 2 has power up, is up to speed, is on-line, and is otherwise operational.

7. EU busy 1. This signal indicates that the EU referenced by EU select 1 is busy.

8. EU busy 2. This signal indicates that the EU referenced by EU select 2 is busy.

Signals Sent To The Disk File Subsystem Via The Other Optimizer

The following signals are sent to the disk file subsystem via the other optimizer of an optimizer pair:

1. Select 3. This signal enables communication between the optimizer and the first set of 10 EU's on the DFX not normally associated with this optimizer, via the other optimizer. The EU's use this signal to gate out information from the desired SU to the optimizer.

2. Select 4. This signal enables communication between the optimizer and the second set of 10 EU's on the disk file exchange not normally associated with this optimizer, via the other optimizer. The EU's use this signal to gate out information from the desired SU to the optimizer.

3. EU select 3. These signals are transmitted over four lines and define one of 10 EU's designated by select 3.

4. SU select 3. These signals are transmitted over four lines and define one of five SU's in the EU referenced by select 3, and one of two shafts, when applicable.

5. EU select 4. These signals are transmitted over four lines and define one of 10 EU's designated by select 4.

6. SU select 4. These signals are transmitted over four lines and define one of five SU's in the EU referenced by select 4, and one of two shaft, when applicable.

Signals Received From The Disk File Subsystem Via The Other Optimizer

The following signals are received from the disk file subsystem via the other optimizer:

1. Shaft position 3. This is the output of a 12-bit counter containing the time equivalent of angular shaft position of the desired SU referenced by SU select 3.

2. Shaft position 4. This is the output of a 12-bit counter containing the time equivalent of angular shaft position of the desired SU referenced by SU select 4.

3. Strobe 3. This signal indicates the existence of valid information on the 12 shaft position 3 lines.

4. Strobe 4. This signal indicates the existence of valid information on the 12 shaft position 4 lines.

5. Storage unit ready level 3. This signal indicates that the SU referenced by SU select 3 has power up, is up to speed, is on-line, and is otherwise operational.

6. Storage unit ready level 4. This signal indicates that the SU referenced by SU select 4 has power up, is up to speed, is on-line, and is otherwise operational.

7. EU busy 3. This signal indicates that the EU referenced by EU select 3 is busy.

8. EU busy 4. This signal indicates that the EU referenced by EU select 4 is busy.

Signals Sent To The Other Optimizer

The following control signals are sent to the other optimizer:

1. Access request. This signal requests access

to an EU normally associated with the other optimizer.

2. Access granted. This signal enables the other optimizer to access an EU not normally associated with it, if the bus to the requested EU is not being used.
3. Shaft position 1. Identical to the signals described above.
4. Shaft position 2. Identical to signals described above.
5. Strobe 1. Identical to signals described above.
6. Strobe 2. Identical to signals described above.
7. Storage unit ready level 1. Identical to signals described above.
8. Storage unit ready level 2. Identical to signals described above.
9. EU busy 1. Identical to signals described above.
10. EU busy 2. Identical to signals described above.

Signals Received From The Other Optimizer

The following control signals are received from the other optimizer:

1. Access granted. This signal enables the optimizer to access an EU not normally associated with it, if the bus to the requested EU is not being used by the optimizer normally associated with it (the other optimizer).
2. Access request. This signal, from the other optimizer, requests access to an EU not normally associated with it.
3. Select 1. This signal indicates a request to raise the signal described above.
4. Select 2. This signal indicates a request to raise the signal described above.
5. EU select 1. This signal indicates a request to raise the signal described above.
6. SU select 1. This signal indicates a request to raise the signal described above.
7. EU select 2. This signal indicates a request to raise the signal described above.
8. SU select 2. This signal indicates a request to raise the signal described above.

FUNCTIONAL UNITS

The optimizer consists of the components specified earlier. Figure 12-6 is a block diagram of the optimizer. It details the sections, (other than the MDL interface unit) which compose each functional unit, and illustrates their inter-relationship.

I/O Interface Unit

The I/O Interface Unit communicates with the MCP; it accepts control words from the MCP and returns control words and status reports to the MCP. The following sections are included in this unit:

1. Drivers and receivers (DR and RX).
2. Scan bus controls.
3. Control word (CW) checker.
4. Status controls.

DRIVERS (DR) AND RECEIVERS (RX)

The lines involved in the optimizer/B 6700 interface constitute the scan bus. The scan bus lines were discussed previously. The DR and RX sections provide the optimizer with the capability of driving and receiving all of the optimizer/B 6700 interface signals.

SCAN BUS CONTROLS

The receipt, processing and transmission of the control signals of the optimizer/B 6700 interface is performed under the supervision of the scan bus controls.

CONTROL WORD (CW) CHECKER

The CW Checker examines the scan interface lines in order to determine if the scan operation is addressed to the optimizer. If so, the CW Checker then checks to see if a scan parity error exists.

STATUS CONTROLS

The Status Controls store information defining optimizer response to the request at the B 6700 interface, and load the Status Report (SR) field of the Scan-In word with a code to describe the response. The Status Controls monitor the conditions listed below, load the Memory Link (ML) field of the Scan-In word with the information indicated below, and set the indicated bit of the SR field:

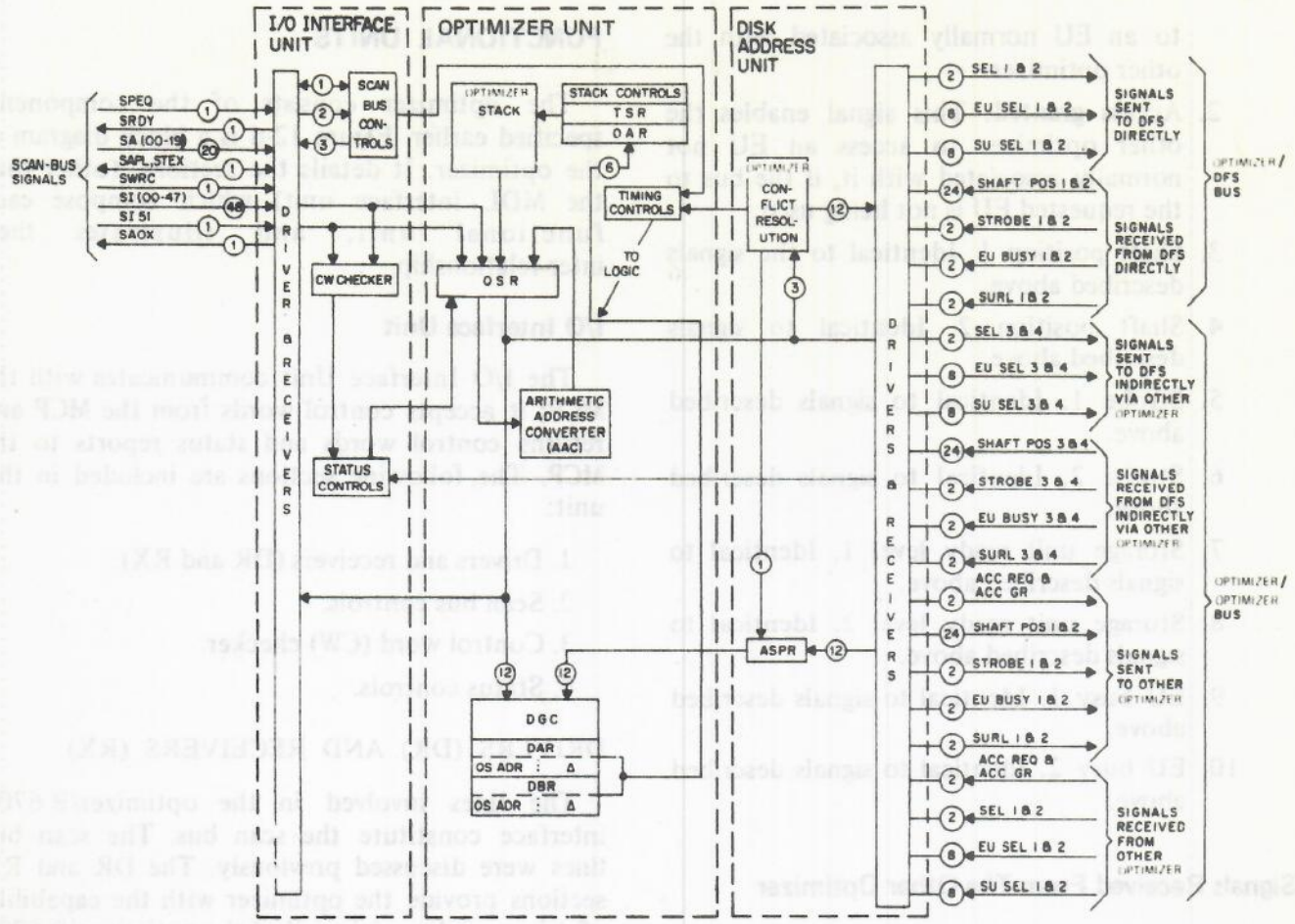


Figure 12-6. Optimizer Block Diagram With Interface Signals (Omitting The Maintenance Diagnostic Interface)

Status Condition	Memory Link	Status Report
No access to OEX	ACW*	47
SU not available	ACW*	46
OS parity error	ACW*	45
Disk address error	ACW*	44
Optimized control word	ACW*	43
Top-of-stack control word	ACW*	42
Stack empty	Zeros	41
Control word not available	Zeros	40

*Associated Control Word – control word associated with the generation of the status report.

Disk Address Unit

The Disk Address Unit activates the Address Select lines to indicate the desired EU and SU, and receives shaft position and control signal information from the selected EU. This unit includes the following:

1. Drivers and receivers (DR and RX).
2. Electronics units conflict resolution.
3. Actual shaft position register (ASPR).

DRIVERS AND RECEIVERS

Capability to address and receive signals from up to 20 EU's directly, and up to 20 EU's indirectly, is provided by the Drivers and Receivers discussed above.

EU CONFLICT RESOLUTION

The EU Conflict Resolution previously defined is used in the EU Conflict Resolution discussed above.

ACTUAL SHAFT POSITION REGISTERS (ASPR)

The time-equivalent of the angular shaft position of the desired SU of the desired EU is stored in the ASPR.

Optimizing Unit

The Optimizing Unit is capable of accumulating up to 32 control words and selecting the best one in terms of minimum access time. This unit includes the following:

1. Arithmetic address converter (AAC).
2. Optimizer Stack (OS).
3. Optimizer Stack register (OSR).
4. Stack Controls. Top-of-the Stack register (TSR), and Optimizer Address register (OAR).
5. Delta Generator and Comparator (DGC).
6. Delta A register (DAR) and Delta B register (DBR).
7. Timing controls.

ARITHMETIC ADDRESS CONVERTER (AAC)

The AAC accepts the 26 bits of the disk address made available to the optimizer during a Scan-Out operation. This information is in BCD format and consists of the desired disk starting address other than the desired exchange number (A or B) and the desired EU number. The AAC converts this information into the desired SU number, and a 12-bit binary number representing the desired starting segment number in terms of the time-equivalent of angular shaft position of the desired SU. The AAC performs the conversion by successively subtracting, from the number obtained from the OSR, constants obtained from a configuration card which defines the type of the SU and from parameter cards which define the storage capabilities of the SU type. (The optimizer can accommodate disk systems consisting of a mix of up to four SU types: IC-3, IC-4, IC-5, and II-B; however, all SU's attached to a given EU must be of the same type.) In this manner, the AAC calculates the desired segment number. The segment number is then converted into the corresponding shaft position, expressed in segments of time, taking into account the SU type and the zone of the disk address. After conversion, the SU number (three bits) and the shaft position information (12 bits) are loaded into the OSR along with the EU number (five bits) and exchange bit.

OPTIMIZER STACK

The optimizer stack provides storage for up to 32 control words. The disk starting address portion of these control words is compatible in format with the addresses received from the SU's.

OPTIMIZER STACK REGISTER (OSR)

The OSR acts as a link to the optimizer stack. Control words to be written into the stack are first loaded into the OSR.

Control words stored in the stack may, when desired, be read into the OSR. The OSR also acts as the link to the scan bus by receiving and transmitting the data interchanged on that bus.

STACK CONTROLS (TSR AND OAR)

The stack controls and the TSR and OAR selections supervise writing into or reading from the optimizer stack, indicate the extent to which the stack is occupied, and find the stack location of the current interest.

STACK CONTROLS. Overall supervision of writing into or reading from the optimizer stack is performed by the stack controls.

TOP-OF-THE-STACK REGISTER (TSR). The TSR indicates the extent to which the optimizer stack is occupied by registering the topmost position of the stack which is occupied. As a control word is added to the optimizer stack, the TSR is incremented by 1. Whenever a control word is erased from the stack, the TSR is decremented by 1.

OPTIMIZER ADDRESS REGISTER (OAR). The OAR points to the optimizer stack location currently being used.

DELTA GENERATOR AND COMPARATOR (DGC)

The DGC accepts the desired shaft position from the OSR and the actual shaft position from the

ASPR. It then generates a delta, compares this delta with the delta stored in the appropriate Delta register, DAR or DBR, and stores the smaller of the two deltas in the proper Delta register, DAR or DBR. The DGC erases a stored delta when it becomes obsolete.

DELTA A REGISTER AND DELTA B REGISTER (DAR AND DBR)

The DAR contains the best control word (in terms of minimum access time) referencing Exchange A. Similarly, the DBR contains the best control word referencing Exchange B. In either case, a control word is referenced by storing its optimizer stack address and its delta, in the DAR and DAB respectively. Each Delta register has a flag associated with it, indicating that an optimized control word is available.

TIMING CONTROLS

The Timing Controls of the optimizing unit provide the overall basic timing coordination for consistent operation and initiate operation of the various functional units at the proper time.

APPENDIX A
OPERATORS, ALPHABETICAL LIST

NAME	MNEMONIC	HEXA-DECIMAL CODE	PAGE
ADD	ADD	80	7-1
BIT RESET	BRST	9E	7-9
BIT SET	BSET	96	7-9
BRANCH FALSE	BRFL	A0	7-5
BRANCH TRUE	BRTR	A1	7-5
BRANCH UNCONDITIONAL	BRUN	A2	7-5
CHANGE SIGN BIT	CHSN	8E	7-9
COMPARE CHARACTERS EQUAL DESTRUCTIVE	CEQD	F4	7-13
COMPARE CHARACTERS EQUAL, UPDATE	CEQU	FC	7-13
COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED	F1	7-13
COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU	F9	7-13
COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD	F2	7-12
COMPARE CHARACTERS GREATER, UPDATE	CGTU	FA	7-13
COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED	F3	7-13
COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU	FB	7-13
COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD	F0	7-13
COMPARE CHARACTERS LESS, UPDATE	CLSU	F8	7-13
COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED	F5	7-13
COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU	FD	7-13
CONDITIONAL HALT (all modes)	HALT	DF	7-6
COUNT BINARY ONES	CBON	95BB	8-13
DELETE TOP OF STACK	DLET	B5	7-6
DISABLE EXTERNAL INTERRUPT	DEXI	9547	8-1
DIVIDE	DIVD	83	7-2
DUPLICATE TOP OF STACK	DUPL	B7	7-6
DYNAMIC BIT RESET	DBRS	9F	7-9
DYNAMIC BIT SET	DBST	97	7-9
DYNAMIC BRANCH FALSE	DBFL	A8	7-5
DYNAMIC BRANCH TRUE	DBTR	A9	7-5
DYNAMIC BRANCH UNCONDITIONAL	DBUN	AA	7-5
DYNAMIC FIELD INSERT	DINS	9D	7-10
DYNAMIC FIELD ISOLATE	DISO	9B	7-10
DYNAMIC FIELD TRANSFER	DFTR	99	7-10
DYNAMIC SCALE LEFT	DSLF	C1	7-8
DYNAMIC SCALE RIGHT FINAL	DSRF	C7	7-9
DYNAMIC SCALE RIGHT ROUND	DSRR	C9	7-9
DYNAMIC SCALE RIGHT SAVE	DSRS	C5	7-8

NAME	MNEMONIC	HEXA-DECIMAL CODE	PAGE
DYNAMIC SCALE RIGHT TRUNCATE	DSRT	C3	7-8
ENABLE EXTERNAL INTERRUPTS	EEXI	9546	8-1
END EDIT (edit mode)	ENDE	DE	9-3
END FLOAT (edit mode)	ENDF	D5	9-2
ENTER	ENTR	AB	7-17
EQUAL	EQUL	8C	7-4
ESCAPE TO 16-BIT INSTRUCTION	VARI	95	8-1
EVALUATE	EVAL	AC	7-20
EXCHANGE	EXCH	B6	7-6
EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU	DD	7-14
EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD	D2	7-14
EXECUTE SINGLE MICRO, UPDATE	EXSU	DA	7-14
EXIT	EXIT	A3	7-15
EXTENDED MULTIPLY	MULX	8F	7-2
FIELD INSERT	INSR	9C	7-10
FIELD ISOLATE	ISOL	9A	7-10
FIELD TRANSFER	FLTR	98	7-9
GREATER THAN	GRTR	8A	7-4
GREATER THAN OR EQUAL	GREQ	89	7-4
IDLE UNTIL INTERRUPT	IDLE	9544	8-1
INDEX	INDX	A6	7-7
INDEX AND LOAD NAME	NXLN	A5	7-7
INDEX AND LOAD VALUE	NXLV	AD	7-7
INPUT CONVERT, DESTRUCTIVE	ICVD	CA	7-14
INPUT CONVERT UPDATE	ICVU	CB	7-15
INSERT CONDITIONAL (edit mode)	INSC	DD	9-2
INSERT DISPLAY SIGN (edit mode)	INSG	D9	9-2
INSERT MARK STACK	IMKS	CF	7-22
INSERT OVERPUNCH (edit mode)	INOP	D8	9-3
INSERT UNCONDITIONAL (edit mode)	INSU	DC	9-2
INTEGER DIVIDE	IDIV	84	7-2
INTEGERIZE, ROUNDED	NTGR	87	7-3
INTEGERIZE, TRUNCATED	NTIA	86	7-3
INTEGERIZE, ROUNDED DOUBLE-PRECISION	NTGD	9587	8-11
INTERRUPT OTHER PROCESSORS	HEYU	954F	8-10
INVALID OPERATOR (all modes)	NVLD	FF	7-6
LEADING ONE TEST	LOG2	958B	8-11
LINKED LIST LOOKUP	LLLU	95BD	8-13
LESS THAN	LESS	88	7-4
LESS THAN OR EQUAL	LSEQ	8B	7-4
LIT CALL ONE	ONE	B1	7-7

NAME	MNEMONIC	HEXA-DECIMAL CODE	PAGE
LIT CALL ZERO	ZERO	B0	7-7
LIT CALL 8 BITS	LT8	B2	7-7
LIT CALL 16 BITS	LT16	B3	7-7
LIT CALL 48 BITS	LT48	BE	7-7
LOAD	LOAD	BD	7-8
LOAD TRANSPARENT	LODT	95BC	8-13
LOGICAL AND	LAND	90	7-4
LOGICAL EQUAL	SAME	94	7-4
LOGICAL EQUIVALENCE	LEQV	93	7-4
LOGICAL NEGATE	LNOT	92	7-4
LOGICAL OR	LOR	91	7-4
MAKE PROGRAM CONTROL WORD	MPCW	BF	7-7
MARK STACK	MKST	AE	7-21
MASKED SEARCH FOR EQUAL	SRCH	95BE	8-13
MOVE CHARACTERS (edit mode)	MCHR	D7	9-1
MOVE NUMERIC UNCONDITIONAL (edit mode)	MVNU	D6	9-1
MOVE TO STACK	MVST	95AF	8-11
MOVE WITH FLOAT (edit mode)	MFLT	D1	9-1
MOVE WITH INSERT (edit mode)	MINS	D0	9-1
MULTIPLY	MULT	82	7-2
NAME CALL	NAMC	40 ⇒ 7F	7-15
NO OPERATION (all modes)	NOOP	FE	7-6
NOT EQUAL	NEQL	8D	7-5
OCCURS INDEX	OCRX	9585	8-10
OVERWRITE DESTRUCTIVE	OVRD	BA	7-6
OVERWRITE NON-DESTRUCTIVE	OVRN	BB	7-6
PACK DESTRUCTIVE	PACD	D1	7-14
PACK UPDATE	PACU	D9	7-14
PUSH DOWN STACK REGISTERS	PUSH	B4	7-6
READ AND CLEAR OVERFLOW FLIP FLOP	ROFF	D7	7-15
READ PROCESSOR IDENTIFICATION	WHOI	954E	8-10
READ PROCESSOR REGISTER	RPRR	95B8	8-12
READ TAG FIELD	RTAG	95B5	8-12
READ TRUE/FALSE FLIP FLOP	RTFF	DE	7-15
READ WITH LOCK	RDLK	95BA	8-13
REMAINDER DIVIDE	RDIV	85	7-2
RESET FLOAT (edit mode)	RSTF	D4	9-2
RETURN	RETN	A7	7-17
ROTATE STACK DOWN	RSDN	95B7	8-12
ROTATE STACK UP	RSUP	95B6	8-12
SCALE LEFT	SCLF	C0	7-8
SCALE RIGHT FINAL	SCRF	C6	7-8

NAME	MNEMONIC	HEXA-DECIMAL CODE	PAGE
SCALE RIGHT ROUNDED	SCRR	C8	7-9
SCALE RIGHT SAVE	SCRS	C4	7-8
SCALE RIGHT TRUNCATE	SCRT	C2	7-8
SCAN IN	SCNI	954A	8-2
SCAN OUT	SCNO	954B	8-8
SCAN WHILE EQUAL, DESTRUCTIVE	SEQD	95F4	8-15
SCAN WHILE EQUAL, UPDATE	SEQU	95FC	8-15
SCAN WHILE FALSE, DESTRUCTIVE	SWFD	95D4	8-16
SCAN WHILE FALSE, UPDATE	SWFU	95DC	8-16
SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED	95F1	8-15
SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU	95F9	8-15
SCAN WHILE GREATER, DESTRUCTIVE	SGTD	95F2	8-15
SCAN WHILE GREATER, UPDATE	SGTU	95FA	8-15
SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED	95F3	8-15
SCAN WHILE LESS OR EQUAL, UPDATE	SLEU	95FB	8-15
SCAN WHILE LESS, DESTRUCTIVE	SLSD	95F0	8-15
SCAN WHILE LESS, UPDATE	SLSU	95F8	8-16
SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED	95F5	8-16
SCAN WHILE NOT EQUAL, UPDATE	SNEU	95FD	8-16
SCAN WHILE TRUE, DESTRUCTIVE	SWTD	95D5	8-16
SCAN WHILE TRUE, UPDATE	SWTU	95DD	8-16
SET DOUBLE TO TWO SINGLES	SPLT	9543	8-1
SET EXTERNAL SIGN	SXSN	D6	7-15
SET INTERVAL TIMER	SINT	9545	8-1
SET PROCESSOR REGISTER	SPRR	95B9	8-13
SET TAG FIELD	STAG	95B4	8-12
SET TO DOUBLE-PRECISION	XTND	CE	7-3
SET TO SINGLE-PRECISION, ROUNDED	SNGL	CD	7-3
SET TO SINGLE-PRECISION, TRUNCATED	SNGT	CC	7-3
SET TWO SINGLES TO DOUBLE	JOIN	9542	8-1
SKIP FORWARD DESTINATION CHARACTERS (edit mode)	SFDC	DA	9-2
SKIP FORWARD SOURCE CHARACTERS (edit mode)	SFSC	D2	9-2
SKIP REVERSE DESTINATION CHARACTERS (edit mode)	SRDC	DB	9-2
SKIP REVERSE SOURCE CHARACTERS (edit mode)	SRSC	D3	9-2
STEP AND BRANCH	STBR	A4	7-5
STORE DESTRUCTIVE	STOD	B8	7-6
STORE NON-DESTRUCTIVE	STON	B9	7-6
STRING ISOLATE	SISO	D5	7-12
STUFF ENVIRONMENT	STFF	AF	7-22
SUBTRACT	SUBT	81	7-1

NAME	MNEMONIC	HEXA-DECIMAL CODE	PAGE
TABLE ENTER EDIT, DESTRUCTIVE	TEED	D0	7-13
TABLE ENTER EDIT, UPDATE	TEEU	D8	7-14
TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND	E6	7-12
TRANSFER UNCONDITIONAL, UPDATE	TUNU	EE	7-12
TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD	E4	7-11
TRANSFER WHILE EQUAL, UPDATE	TEQU	EC	7-12
TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED	E1	7-11
TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU	E9	7-11
TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD	E2	7-11
TRANSFER WHILE GREATER, UPDATE	TGTU	EA	7-11
TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED	E3	7-12
TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD	95D2	8-14
TRANSFER WHILE FALSE, UPDATE	TWFU	95DA	8-14
TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD	95D3	8-14
TRANSFER WHILE TRUE, UPDATE	TWTU	95DB	8-14
TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU	EB	7-12
TRANSFER WHILE LESS, DESTRUCTIVE	TLSD	E0	7-12
TRANSFER WHILE LESS, UPDATE	TLSU	E8	7-12
TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED	E5	7-12
TRANSFER WHILE NOT EQUAL, UPDATE	TNEU	ED	7-12
TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD	D4	7-11
TRANSFER WORDS OVERWRITE UPDATE	TWOU	DC	7-11
TRANSFER WORDS, DESTRUCTIVE	TWSD	D3	7-10
TRANSFER WORDS, UPDATE	TWSU	DB	7-11
TRANSLATE	TRNS	95D7	8-15
UNPACK ABSOLUTE, DESTRUCTIVE	UABD	95D1	8-14
UNPACK ABSOLUTE, UPDATE	UABU	95D9	8-14
UNPACK SIGNED, DESTRUCTIVE	USND	95D0	8-14
UNPACK SIGNED, UPDATE	USNU	95D8	8-14
VALUE CALL	VALC	00 ⇒ 3F	7-15

APPENDIX B
OPERATORS, NUMERICAL LIST

PRIMARY MODE

HEXA-DECIMAL CODE	NAME	MNEMONIC	PAGE
00 => 3F	VALUE CALL	VALC	7-15
40 => 7F	NAME CALL	NAMC	7-15
80	ADD	ADD	7-1
81	SUBTRACT	SUBT	7-1
82	MULTIPLY	MULT	7-2
83	DIVIDE	DIVD	7-2
84	INTEGER DIVIDE	IDIV	7-2
85	REMAINDER DIVIDE	RDIV	7-2
86	INTEGERIZE, TRUNCATED	NTIA	7-3
87	INTEGERIZE, ROUNDED	NTGR	7-3
88	LESS THAN	LESS	7-4
89	GREATER THAN OR EQUAL	GREQ	7-4
8A	GREATER THAN	GRTR	7-4
8B	LESS THAN OR EQUAL	LSEQ	7-4
8C	EQUAL	EQUL	7-4
8D	NOT EQUAL	NEQL	7-5
8E	CHANGE SIGN BIT	CHSN	7-9
8F	EXTENDED MULTIPLY	MULX	7-2
90	LOGICAL AND	LAND	7-4
91	LOGICAL OR	LOR	7-4
92	LOGICAL NEGATE	LNOT	7-4
93	LOGICAL EQUIVALENCE	LEQV	7-4
94	LOGICAL EQUAL	SAME	7-4
95	ESCAPE TO 16-BIT INSTRUCTION	VARI	8-1
96	BIT SET	BSET	7-9
97	DYNAMIC BIT SET	DBST	7-9
98	FIELD TRANSFER	FLTR	7-9
99	DYNAMIC FIELD TRANSFER	DFTR	7-10
9A	FIELD ISOLATE	ISOL	7-10
9B	DYNAMIC FIELD ISOLATE	DISO	7-10
9C	FIELD INSERT	INSR	7-10
9D	DYNAMIC FIELD INSERT	DINS	7-10
9E	BIT RESET	BRST	7-9
9F	DYNAMIC BIT RESET	DBRS	7-9
A0	BRANCH FALSE	BRFL	7-5
A1	BRANCH TRUE	BRTR	7-5
A2	BRANCH UNCONDITIONAL	BRUN	7-5
A3	EXIT	EXIT	7-15
A4	STEP AND BRANCH	STBR	7-5
A5	INDEX AND LOAD NAME	NXLN	7-7

PRIMARY MODE

HEXA-DECIMAL CODE	NAME	MNEMONIC	PAGE
A6	INDEX	INDX	7-7
A7	RETURN	RETN	7-17
A8	DYNAMIC BRANCH FALSE	DBFL	7-5
A9	DYNAMIC BRANCH TRUE	DBTR	7-5
AA	DYNAMIC BRANCH UNCONDITIONAL	DBUN	7-5
AB	ENTER	ENTR	7-17
AC	EVALUATE DESCRIPTOR	EVAL	7-20
AD	INDEX AND LOAD VALUE	NXLV	7-7
AE	MARK STACK	MKST	7-21
AF	STUFF ENVIRONMENT	STFF	7-22
B0	LIT CALL ZERO	ZERO	7-7
B1	LIT CALL ONE	ONE	7-7
B2	LIT CALL 8 BITS	LT8	7-7
B3	LIT CALL 16 BITS	LT16	7-7
B4	PUSH DOWN STACK REGISTERS	PUSH	7-6
B5	DELETE TOP OF STACK	DLET	7-6
B6	EXCHANGE	EXCH	7-6
B7	DUPLICATE TOP OF STACK	DUPL	7-6
B8	STORE DESTRUCTIVE	STOD	7-6
B9	STORE NON-DESTRUCTIVE	STON	7-6
BA	OVERWRITE DESTRUCTIVE	OVRD	7-6
BB	OVERWRITE NON-DESTRUCTIVE	OVRN	7-6
BD	LOAD	LOAD	7-8
BE	LIT CALL 48 BITS	LT48	7-7
BF	MAKE PROGRAM CONTROL WORD	MPCW	7-7
C0	SCALE LEFT	SCLF	7-8
C1	DYNAMIC SCALE LEFT	DSLF	7-8
C2	SCALE RIGHT TRUNCATE	SCRT	7-8
C3	DYNAMIC SCALE RIGHT TRUNCATE	DSRT	7-8
C4	SCALE RIGHT SAVE	SCRS	7-8
C5	DYNAMIC SCALE RIGHT SAVE	DSRS	7-8
C6	SCALE RIGHT FINAL	SCRF	7-8
C7	DYNAMIC SCALE RIGHT FINAL	DSRF	7-9
C8	SCALE RIGHT ROUNDED	SCRR	7-9
C9	DYNAMIC SCALE RIGHT ROUND	DSRR	7-9
CA	INPUT CONVERT, DESTRUCTIVE	ICVD	7-14
CB	INPUT CONVERT, UPDATE	ICVU	7-15
CC	SET TO SINGLE-PRECISION, TRUNCATED	SNGT	7-3
CD	SET TO SINGLE-PRECISION, ROUNDED	SNGL	7-3
CE	SET TO DOUBLE-PRECISION	XTND	7-3
CF	INSERT MARK STACK	IMKS	7-22
D0	TABLE ENTER EDIT, DESTRUCTIVE	TEED	7-13

PRIMARY MODE

HEXA-DECIMAL CODE	NAME	MNEMONIC	PAGE
D1	PACK DESTRUCTIVE	PACD	7-14
D2	EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD	7-14
D3	TRANSFER WORDS, DESTRUCTIVE	TWSD	7-10
D4	TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD	7-11
D5	STRING ISOLATE	SISO	7-12
D6	SET EXTERNAL SIGN	SXSN	7-15
D7	READ AND CLEAR OVERFLOW FLIP FLOP	ROFF	7-15
D8	TABLE ENTER EDIT, UPDATE	TEEU	7-14
D9	PACK UPDATE	PACU	7-14
DA	EXECUTE SINGLE MICRO, UPDATE	EXSU	7-14
DB	TRANSFER WORDS, UPDATE	TWSU	7-11
DC	TRANSFER WORDS OVERWRITE UPDATE	TWOU	7-11
DD	EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU	7-14
DE	READ TRUE/FALSE FLIP FLOP	TRFF	7-15
DF	CONDITIONAL HALT	HALT	7-6
E0	TRANSFER WHILE LESS, DESTRUCTIVE	TLSD	7-12
E1	TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED	7-11
E2	TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD	7-11
E3	TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED	7-12
E4	TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD	7-11
E5	TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED	7-12
E6	TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND	7-12
E8	TRANSFER WHILE LESS, UPDATE	TLSU	7-12
E9	TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU	7-11
EA	TRANSFER WHILE GREATER, UPDATE	TGTU	7-11
EB	TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU	7-12
EC	TRANSFER WHILE EQUAL, UPDATE	TEQU	7-12
ED	TRANSFER WHILE NOT EQUAL, UPDATE	TNEU	7-12
EE	TRANSFER UNCONDITIONAL, UPDATE	TUNU	7-12
F0	COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD	7-13
F1	COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED	7-13
F2	COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD	7-12
F3	COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED	7-13
F4	COMPARE CHARACTERS EQUAL, DESTRUCTIVE	CEQD	7-13
F5	COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED	7-13
F8	COMPARE CHARACTERS LESS, UPDATE	CLSU	7-13

PRIMARY MODE

HEXA-DECIMAL CODE	NAME	MNEMONIC	PAGE
F9	COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU	7-13
FA	COMPARE CHARACTERS GREATER, UPDATE	CGTU	7-13
FB	COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU	7-13
FC	COMPARE CHARACTERS EQUAL, UPDATE	CEQU	7-13
FD	COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU	7-13
FE	NO OPERATION	NOOP	7-6
FF	INVALID OPERATOR	NVLD	7-6

VARIANT MODE

9542	SET TWO SINGLES TO DOUBLE	JOIN	8-1
9543	SET DOUBLE TO TWO SINGLES	SPLT	8-1
9544	IDLE UNTIL INTERRUPT	IDLE	8-1
9545	SET INTERVAL TIMER	SINT	8-1
9546	ENABLE EXTERNAL INTERRUPTS	EEXI	8-1
9547	DISABLE EXTERNAL INTERRUPTS	DEXI	8-1
954A	SCAN IN	SCNI	8-2
954B	SCAN OUT	SCNO	8-8
954E	READ PROCESSOR IDENTIFICATION	WHOI	8-10
954F	INTERRUPT OTHER PROCESSORS	HEYU	8-10
9585	OCCURS INDEX	OCRX	8-10
9587	INTEGERIZE, ROUNDED, DOUBLE-PRECISION	NTGD	8-11
958B	LEADING ONE TEST	LOG2	8-11
95AF	MOVE TO STACK	MVST	8-11
95B4	SET TAG FIELD	STAG	8-12
95B5	READ TAG FIELD	RTAG	8-12
95B6	ROTATE STACK UP	RSUP	8-12
95B7	ROTATE STACK DOWN	RSDN	8-12
95B8	READ PROCESSOR REGISTER	RPRR	8-12
95B9	SET PROCESSOR REGISTER	SPRR	8-13
95BA	READ WITH LOCK	RDLK	8-13
95BB	COUNT BINARY ONES	CBON	8-13
95BC	LOAD TRANSPARENT	LODT	8-13
95BD	LINKED LIST LOOKUP	LLLU	8-13
95BE	MASKED SEARCH FOR EQUAL	SRCH	8-13
95D0	UNPACK SIGNED, DESTRUCTIVE	USND	8-14
95D1	UNPACK ABSOLUTE, DESTRUCTIVE	UABD	8-14
95D2	TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD	8-14
95D3	TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD	8-14
95D4	SCAN WHILE FALSE, DESTRUCTIVE	SWFD	8-16
95D5	SCAN WHILE TRUE, DESTRUCTIVE	SWTD	8-16
95D7	TRANSLATE	TRNS	8-15

VARIANT MODE

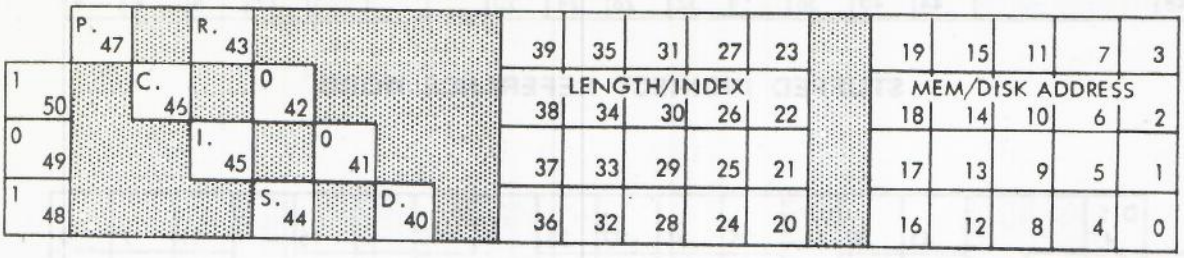
HEXA-DECIMAL CODE	NAME	MNEMONIC	PAGE
95D8	UNPACK SIGNED, UPDATE	USNU	8-14
95D9	UNPACK ABSOLUTE, UPDATE	UABU	8-14
95DA	TRANSFER WHILE FALSE, UPDATE	TWFU	8-14
95DB	TRANSFER WHILE TRUE, UPDATE	TWTU	8-14
95DC	SCAN WHILE FALSE, UPDATE	SWFU	8-16
95DD	SCAN WHILE TRUE, UPDATE	SWTU	8-16
95DF	CONDITIONAL HALT	HALT	7-6
95F0	SCAN WHILE LESS, DESTRUCTIVE	SLSD	8-15
95F1	SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED	8-15
95F2	SCAN WHILE GREATER, DESTRUCTIVE	SGTD	8-15
95F3	SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED	8-15
95F4	SCAN WHILE EQUAL, DESTRUCTIVE	SEQD	8-15
95F5	SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED	8-16
95F8	SCAN WHILE LESS, UPDATE	SLSU	8-16
95F9	SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU	8-15
95FA	SCAN WHILE GREATER, UPDATE	SGTU	8-15
95FB	SCAN WHILE LESS OR EQUAL, UPDATE	SLEU	8-15
95FC	SCAN WHILE EQUAL, UPDATE	SEQU	8-15
95FD	SCAN WHILE NOT EQUAL, UPDATE	SNEU	8-16
95FE	NO OPERATION	NOOP	7-6
95FF	INVALID	NVLD	7-6

EDIT MODE

D0	MOVE WITH INSERT	MINS	9-1
D1	MOVE WITH FLOAT	MFLT	9-1
D2	SKIP FORWARD SOURCE CHARACTERS	SFSC	9-2
D3	SKIP REVERSE SOURCE CHARACTERS	SRSC	9-2
D4	RESET FLOAT	RSTF	9-2
D5	END FLOAT	ENDF	9-2
D6	MOVE NUMERIC UNCONDITIONAL	MVNU	9-1
D7	MOVE CHARACTERS	MCHR	9-1
D8	INSERT OVERPUNCH	INOP	9-3
D9	INSERT DISPLAY SIGN	INSG	9-2
DA	SKIP FORWARD DESTINATION CHARACTERS	SFDC	9-2
DB	SKIP REVERSE DESTINATION CHARACTERS	SRDC	9-2
DC	INSERT UNCONDITIONAL	INSU	9-2
DD	INSERT CONDITIONAL	INSC	9-2
DE	END EDIT	ENDE	9-3
DF	CONDITIONAL HALT	HALT	7-6
FE	NO OPERATION	NOOP	7-6
FF	INVALID	NVLD	7-6

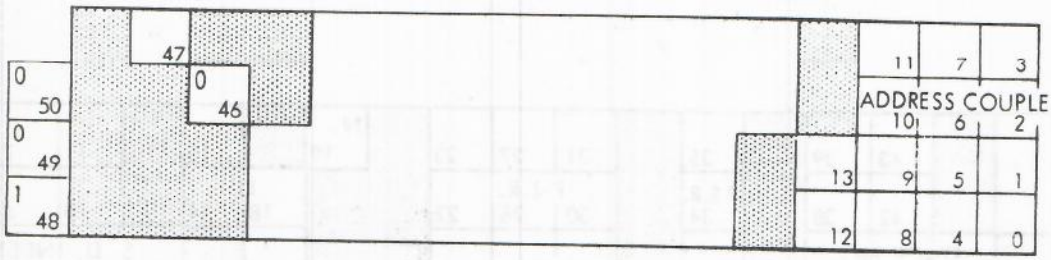
APPENDIX C

CONTROL WORD FORMATS

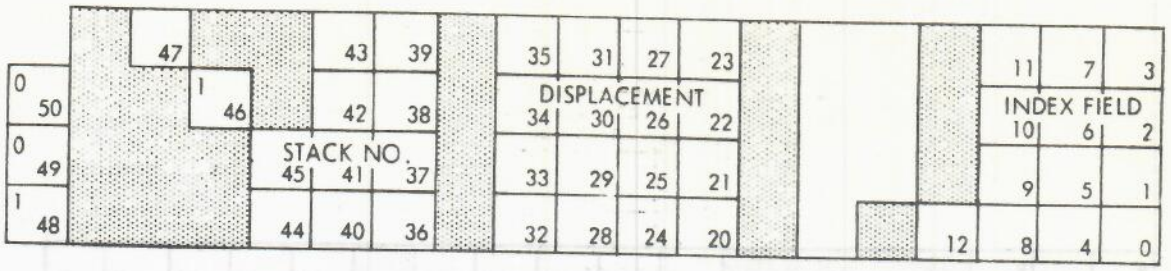


DATA DESCRIPTOR

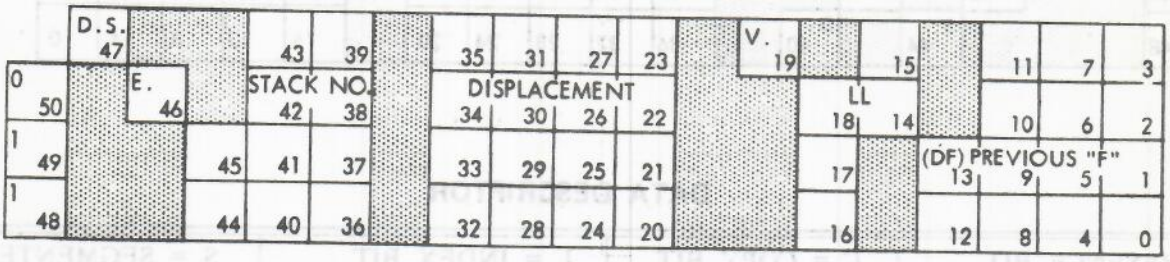
P = PRESENCE BIT 1 = PRESENT IN MAIN MEMORY 0 = NOT PRESENT IN MAIN MEMORY	C = COPY BIT 1 = A COPY 0 = ORIGINAL	I = INDEX BIT 1 = INDEXED 0 = NON INDEXED	S = SEGMENTED BIT 1 = AREA SEGMENTED 0 = NOT SEGMENTED
READ ONLY BIT 1 = READ ONLY 0 = READ/WRITE	42 & 41 MUST = 00 FOR DATA DESC.	D = DOUBLE-PRECISION BIT 1 = DOUBLE-PRECISION DATA 0 = SINGLE-PRECISION DATA	



NORMAL INDIRECT REFERENCE WORD

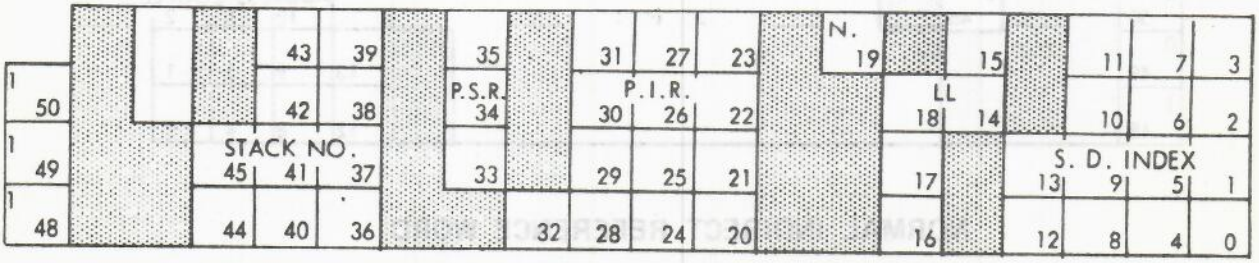


STUFFED INDIRECT REFERENCE WORD



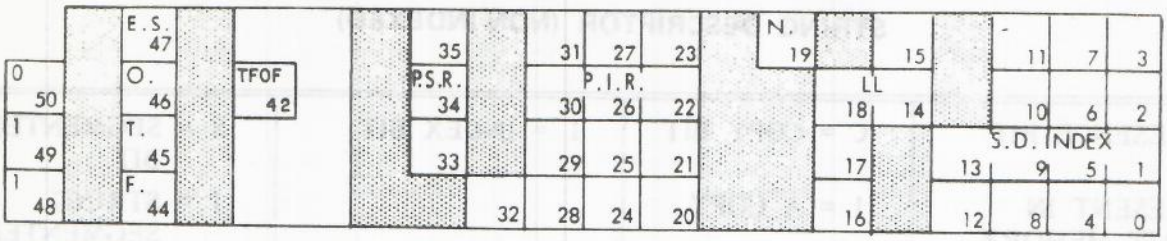
MARK STACK CONTROL WORD

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



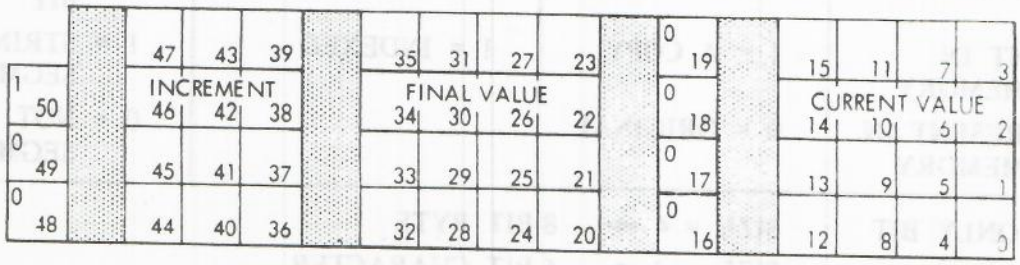
PROGRAM CONTROL WORD

N = NORMAL/CONTROL STATE FF SD = SEGMENT DESCRIPTOR
1 = CONTROL STATE
0 = NORMAL STATE

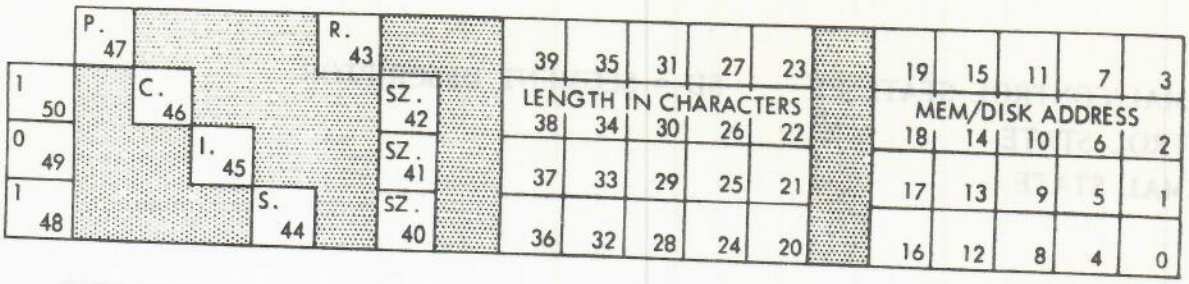


RETURN CONTROL WORD

E.S. = EXTERNAL SIGN BIT 1 = NEGATIVE 0 = POSITIVE	0 = OVERFLOW FF 1 = OVERFLOW 0 = NO OVERFLOW	T = TRUE/FALSE FF 1 = TRUE 0 = FALSE TFOF = TRUE/FALSE FF OCCUPIED FF 1 = TFFF VALID 0 = TFFF NOT DETERMINED
F = FLOAT FF 1 = FLOAT 0 = NO FLOAT	N = NORMAL/CONTROL FF 1 = CONTROL STATE 0 = NORMAL STATE	

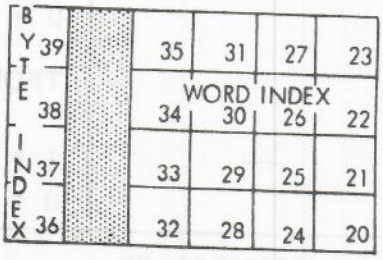


STEP INDEX WORD



STRING DESCRIPTOR (NON-INDEXED)

P = PRESENCE BIT 1 = PRESENT IN MAIN MEMORY 0 = NOT PRESENT IN MAIN MEMORY	C = COPY BIT 1 = A COPY 0 = ORIGINAL	I = INDEX BIT 0 = NON-INDEXED	S = SEGMENTED BIT 1 = STRING SEGMENTED 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		

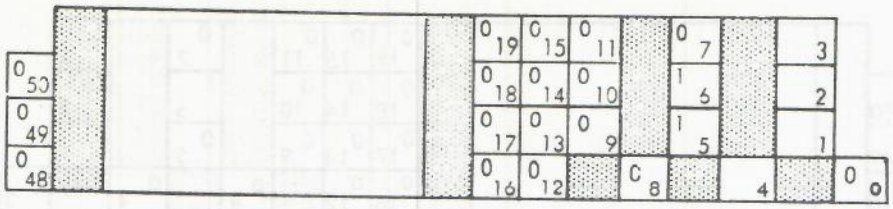


STRING DESCRIPTOR (INDEXED)

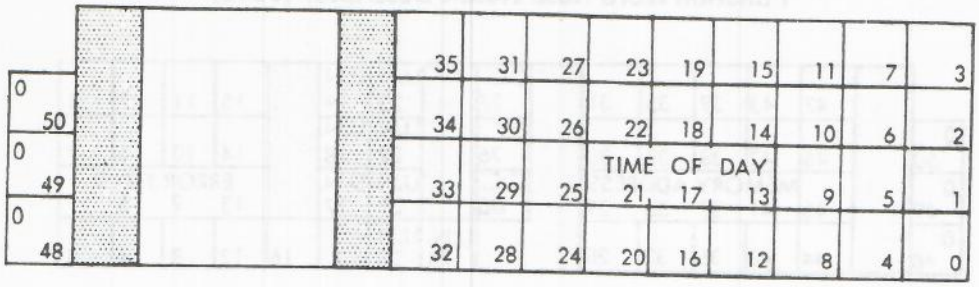
P = PRESENCE BIT 1 = PRESENT IN MAIN MEMORY 0 = NOT PRESENT IN MAIN MEMORY	C = COPY BIT 1 = A COPY 0 = ORIGINAL	I = INDEX BIT 1 = INDEXED	S = SEGMENTED BIT 1 = STRING SEGMENTED 0 = NOT SEGMENTED
R = READ ONLY BIT 1 = READ ONLY 0 = READ/WRITE	SIZE = 4 ⇒ 8-BIT BYTE SIZE = 3 ⇒ 6-BIT CHARACTER SIZE = 2 ⇒ 4-BIT DIGIT		

APPENDIX D
SCAN FUNCTION CODE WORDS

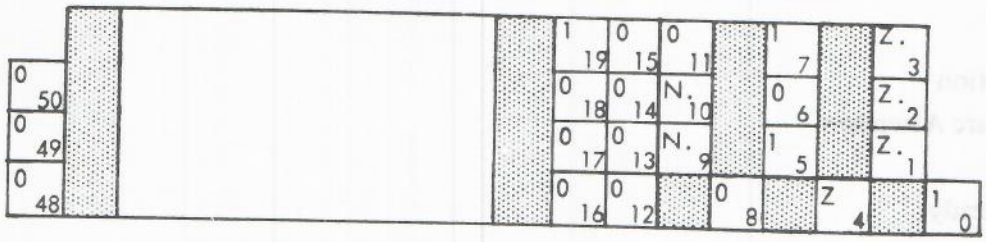
(SCAN IN)



Function Word Read Time of Day Clock (0011)



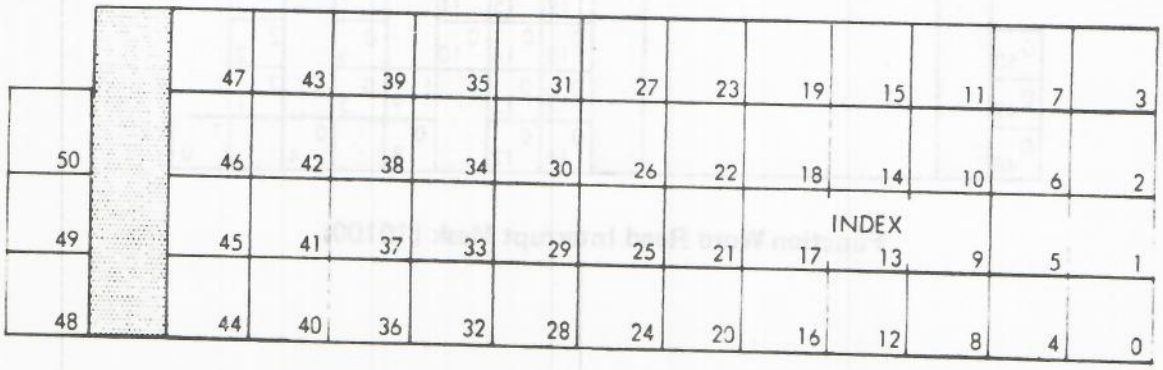
Time of Day (Binary) Word Returned



Function Word Read General Control Adapter (0101)

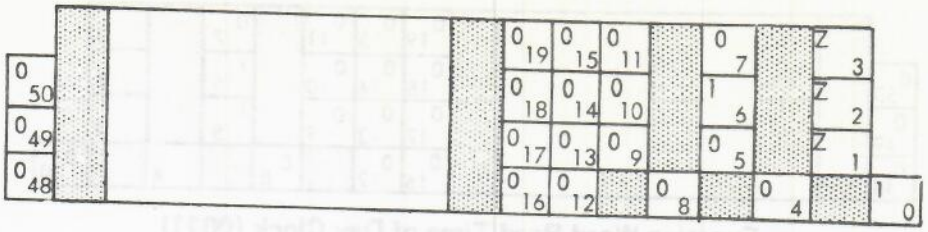
Z = 0001, GCA A is to respond
Z = 0010, GCA B is to respond
Z = 0100, GCA C
Z = 1000, GCA D

N = 00, Read GCA Input Register
N = 01, Read GCA Interrupt Mask Register
N = 10, Read GCA Interrupt Register
N = 11, Read GCA Output Register

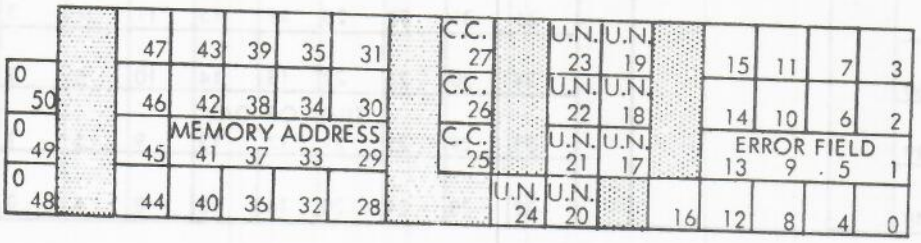


a. G.C.A. Register Word Returned
b. G.C.A. Register Word Sent To I/O Processor

(SCAN IN) (Cont'd.)

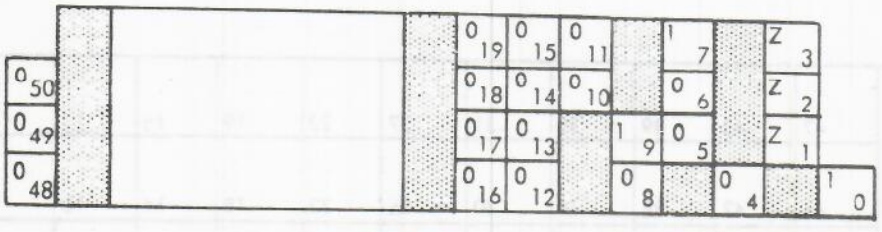


Function Word Read Result Descriptor (0010)



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 I/O Processor section)

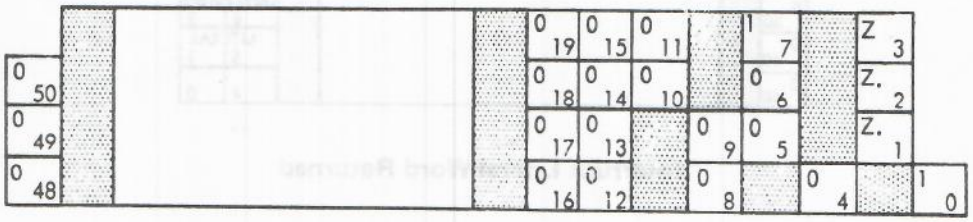


Function Word Read Interrupt Mask (10100)

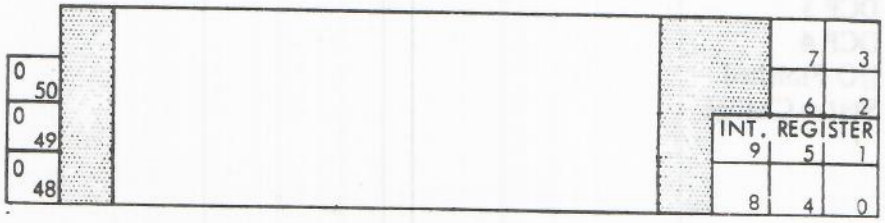
(SCAN IN) (Cont'd.)

Interrupt Mask Word Returned

Bit 9 = I/O Processor I/O Finish
Bit 1 = Data Communications Processor 1
Bit 2 = Data Communications Processor 2
Bit 3 = Data Communications Processor 3
Bit 4 = Data Communications Processor 4
Bit 0 = Status Change



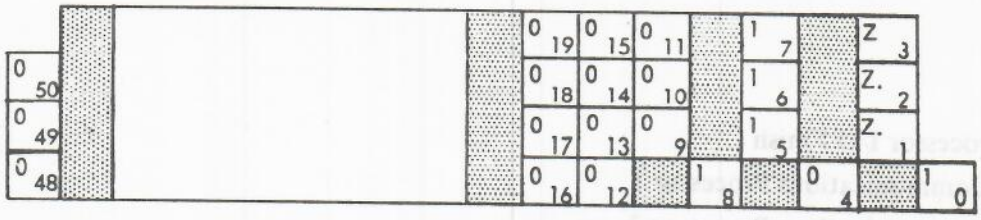
Function Word Read Interrupt Register (0100)



Interrupt Register Word Returned

(SCAN IN) (Cont'd.)

Bit 9 = I/O Processor I/O Finish
Bit 1 = Data Communications Processor 1
Bit 2 = Data Communications Processor 2
Bit 3 = Data Communications Processor 3
Bit 4 = Data Communications Processor 4
Bit 0 = Status Change Interrupt



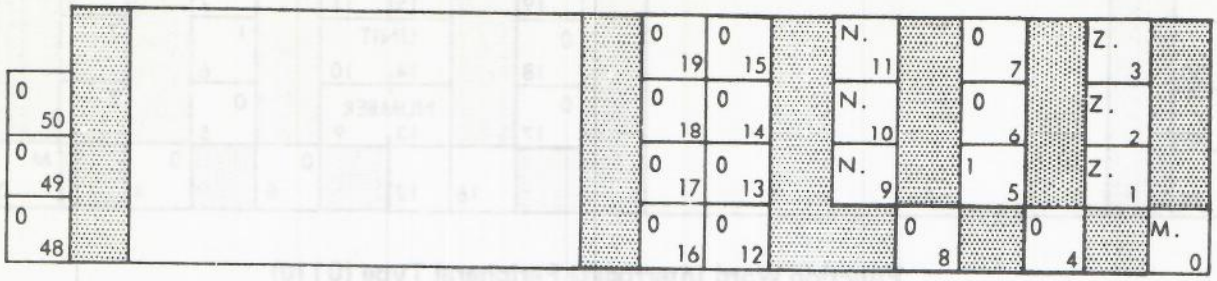
Function Word Read Interrupt Literal (1111)

Interrupt Literal Word Returned

Bits 2:3 = 001 = I/O Processor A
010 = I/O Processor B
100 = I/O Processor C

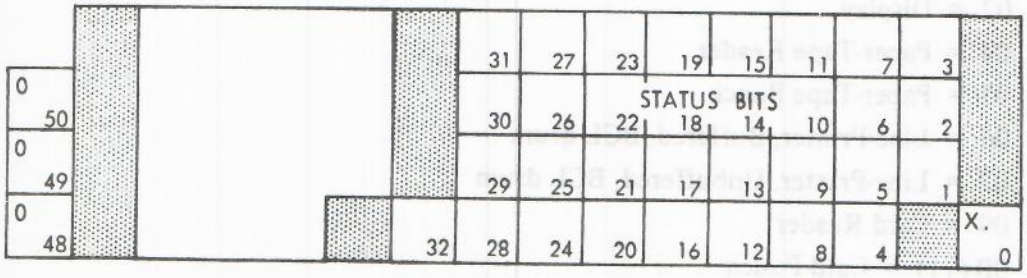
Bits 7:4 = 0001 = DCP 1
0010 = DCP 2
0011 = DCP 3
0100 = DCP 4
1001 = I/O Finished
1111 = Status Change

(SCAN IN) (Cont'd.)



Function Word Interrogate Peripheral Status (0001)

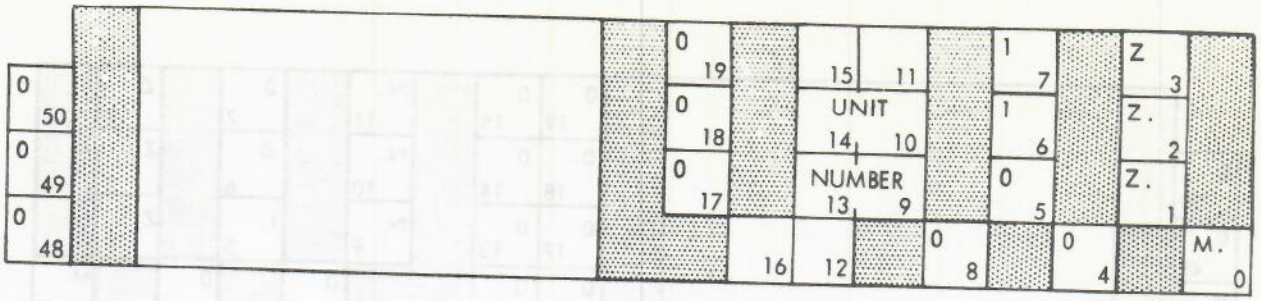
M = 0 = All I/O Processors to respond
M = 1 = I/O Processor designated by Z to respond
Z = 001 = Designates I/O Processor A
Z = 010 = Designates I/O Processor B
Z = 100 = Designates I/O Processor C
N = 0 ⇒ 7 Status Vector Number (in Binary)
N = 8 Status Change Vector



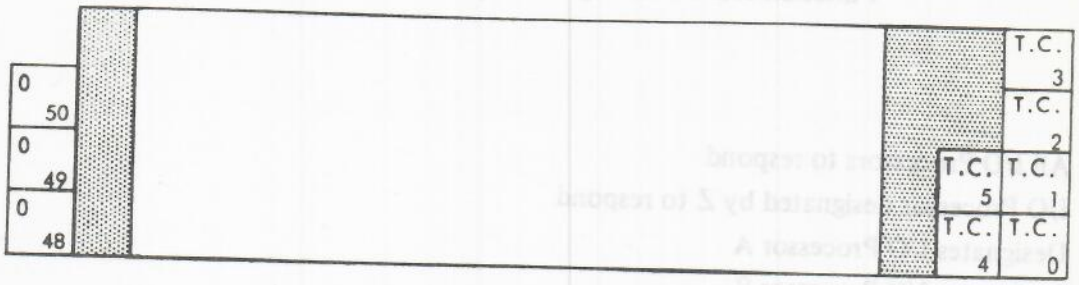
Unit Status Word Returned

X = 0 = Status word not present
X = 1 = Status word present

(SCAN IN) (Cont'd.)



Function Word Interrogate Peripheral Type (0110)

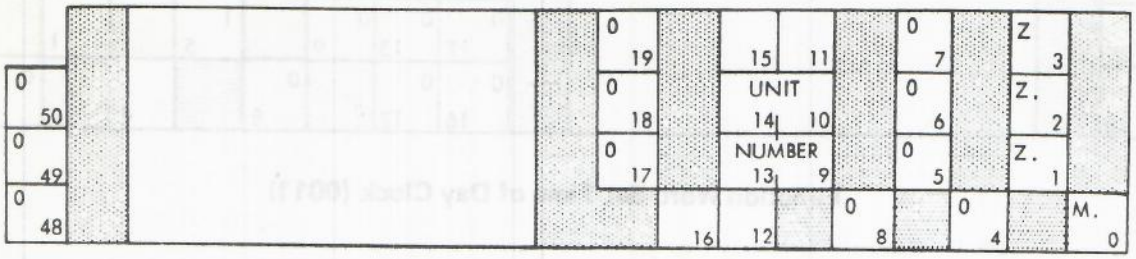


Unit Type Word Returned

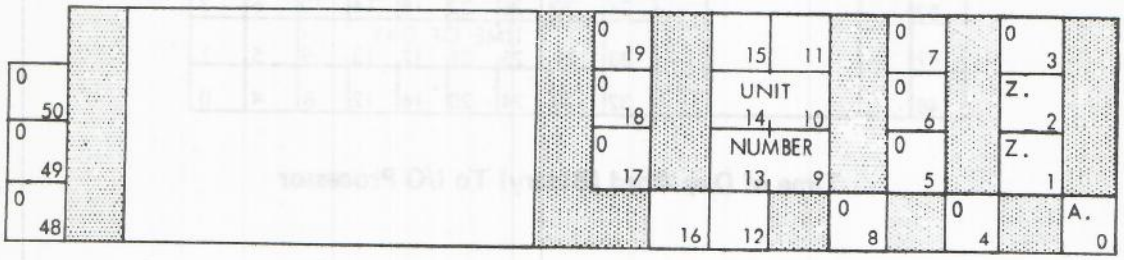
Type Code = 00 = No Unit
01 = Disk File
02 = Display
04 = Paper-Tape Reader
05 = Paper-Tape Punch
06 = Line-Printer, Buffered, BCL drum
07 = Line-Printer, Unbuffered, BCL drum
09 = Card Reader
0B (11) = Card Punch
0D (13) = Magnetic Tape (7-track) } Exchange
0E (14) = Magnetic Tape (9-track NRZ) } Exchange
0F (15) = Magnetic Tape (9-track P.E.) } Exchange
1D (29) = Magnetic Tape (7-track) } Serial or Cluster
1E (30) = Magnetic Tape (9-track NRZ) } Serial or Cluster
1F (31) = Magnetic Tape (9-track P.E.) } Serial or Cluster

(SCAN IN) (Cont'd.)

Type Code = 26 (38) = Line-Printer, Buffered, EBCDIC drum
(Cont'd.) 27 (39) = Line-Printer, Unbuffered, EBCDIC drum



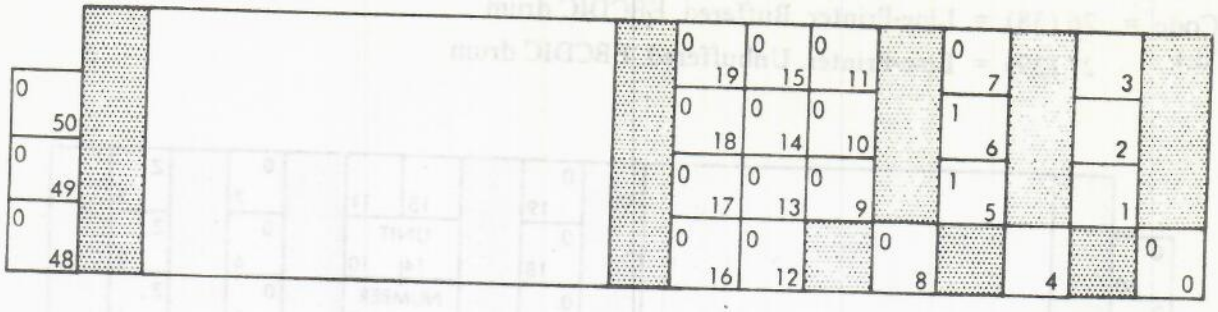
Function Word Interrogate Input/Output Path (0000)



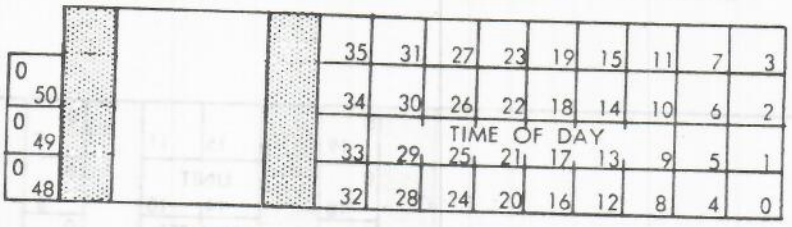
Input/Output Path Word Returned

A = 0 = No Path Available
A = 1 = Path is Available
Z = 001 = Path via I/O Processor A
Z = 010 = Path via I/O Processor B
Z = 011 = Path via Either I/O A or B
Z = 100 = Path via I/O Processor C
Z = 101 = Path via I/O Processor A and C
Z = 110 = Path via I/O Processor B and C
Z = 111 = Path via all I/O Processors

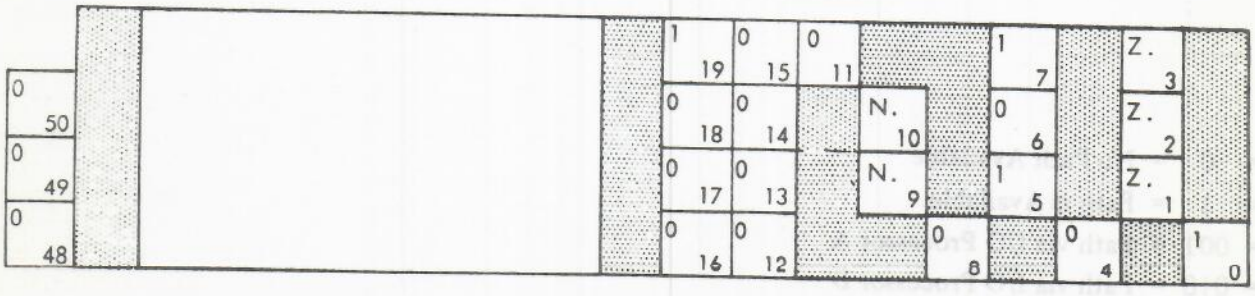
(SCAN OUT)



Function Word Set Time of Day Clock (0011)



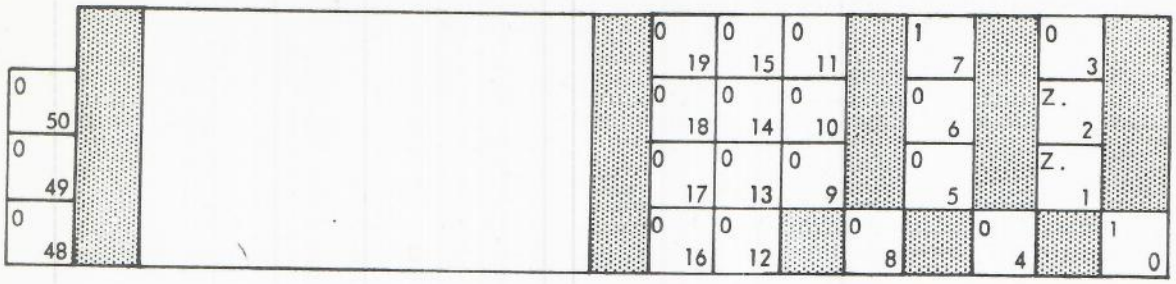
Time of Day Word (Binary) To I/O Processor



Function Word Set General Control Adapter (0101)

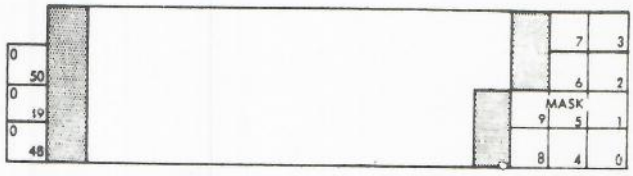
(SCAN OUT) (Cont'd.)

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

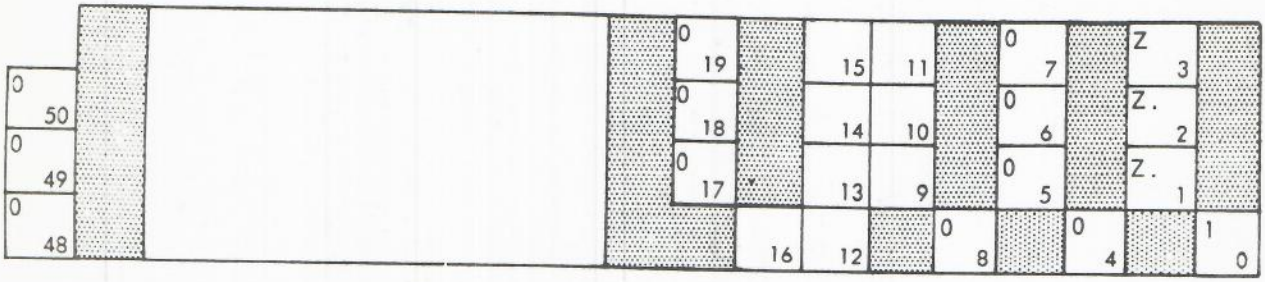


Function Word Set Interrupt Mask (0100)

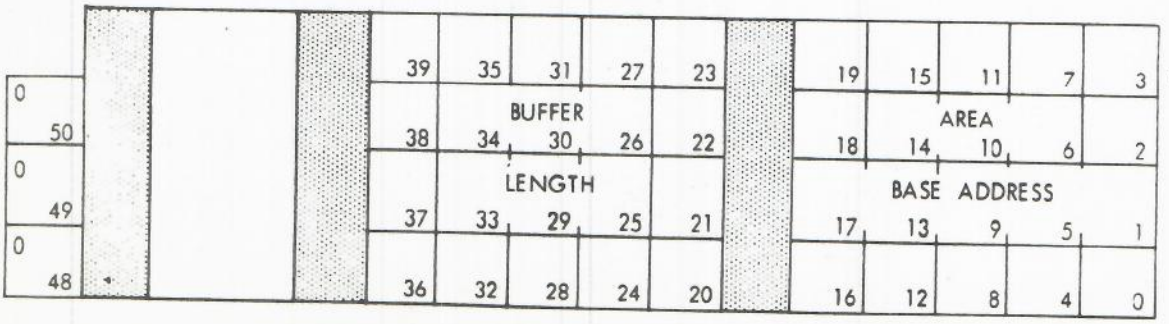
Bit 9 = I/O Processor
Bit 1 = Data Communications Processor 1
Bit 2 = Data Communications Processor 2
Bit 3 = Data Communications Processor 3
Bit 4 = Data Communications Processor 4
Bit 0 = Status Change Interrupt



Interrupt Mask Word Sent To Multiplexor



Function Word Initiate I/O (0000)



Area Descriptor Word Sent To I/O Processor

DATA REPRESENTATION

EBCDIC GRAPHIC	BCL	DECIMAL VALUE	EBCDIC INTERNAL	HEX. GRAPHIC	EBCDIC CARD CODE	BCL CARD CODE	OCTAL	BCL INTERNAL	BCL EXTERNAL
BLANK		64	0100 0000	40	No Punches	No Punches	60	11 0000	01 0000
[		74	0100 1010	4A	12 8 2	12 8 4	33	01 1011	11 1100
.		75	0100 1011	4B	12 8 3	12 8 3	32	01 1010	11 1011
<		76	0100 1100	4C	12 8 4	12 8 6	36	01 1110	11 1110
(		77	0100 1101	4D	12 8 5	12 8 5	35	01 1101	11 1101
+		78	0100 1110	4E	12 8 6				11 1010
|	←	79	0100 1111	4F	12 8 7	12 8 7	37	01 1111	11 1111
&		80	0101 0000	50	12	12	34	01 1100	11 0000
]		90	0101 1010	5A	11 8 2	0 8 6	76	11 1110	01 1110
$		91	0101 1011	5B	11 8 3	11 8 3	52	10 1010	10 1011
*		92	0101 1100	5C	11 8 4	11 8 4	53	10 1011	10 1100
)		93	0101 1101	5D	11 8 5	11 8 5	55	10 1101	10 1101
;		94	0101 1110	5E	11 8 6	11 8 6	56	10 1110	10 1110
¬	≤	95	0101 1111	5F	11 8 7	11 8 7	57	10 1111	10 1111
-		96	0110 0000	60	11	11	54	10 1100	10 0000
/		97	0110 0001	61	0 1	0 1	61	11 0001	01 0001
,		107	0110 1011	6B	0 8 3	0 8 3	72	11 1010	01 1011
%		108	0110 1100	6C	0 8 4	0 8 4	73	11 1011	01 1100
_	≠	109	0110 1101	6D	0 8 5	0 8 2	74	11 1100	01 1010
>		110	0110 1110	6E	0 8 6	8 6	16	00 1110	00 1110
?		111	0110 1111	6F	0 8 7	*	14	00 1100	00 0000
:		122	0111 1010	7A	8 2	8 5	15	00 1101	00 1101
#		123	0111 1011	7B	8 3	8 3	12	00 1010	00 1011
@		124	0111 1100	7C	8 4	8 4	13	00 1011	00 1100
'	≥	125	0111 1101	7D	8 5	8 7	17	00 1111	00 1111
=		126	0111 1110	7E	8 6	0 8 5	75	11 1101	01 1101
"		127	0111 1111	7F	8 7	0 8 7	77	11 1111	01 1111
(+)PZ	+	192	1100 0000	CO	12 0	12 0	20	01 0000	11 1010
A		193	1100 0001	C1	12 1	12 1	21	01 0001	11 0001
B		194	1100 0010	C2	12 2	12 2	22	01 0010	11 0010
C		195	1100 0011	C3	12 3	12 3	23	01 0011	11 0011
D		196	1100 0100	C4	12 4	12 4	24	01 0100	11 0100
E		197	1100 0101	C5	12 5	12 5	25	01 0101	11 0101
F		198	1100 0110	C6	12 6	12 6	26	01 0110	11 0110
G		199	1100 0111	C7	12 7	12 7	27	01 0111	11 0111
H		200	1100 1000	C8	12 8	12 8	30	01 1000	11 1000
I		201	1100 1001	C9	12 9	12 9	31	01 1001	11 1001
(!)MZ	MULT ×	208	1101 0000	D0	11 0	11 0	40	10 0000	10 1010
J		209	1101 0001	D1	11 1	11 1	41	10 0001	10 0001
K		210	1101 0010	D2	11 2	11 2	42	10 0010	10 0010
L		211	1101 0011	D3	11 3	11 3	43	10 0011	10 0011
M		212	1101 0100	D4	11 4	11 4	44	10 0100	10 0100
N		213	1101 0101	D5	11 5	11 5	45	10 0101	10 0101
O		214	1101 0110	D6	11 6	11 6	46	10 0110	10 0110
P		215	1101 0111	D7	11 7	11 7	47	10 0111	10 0111

*All other codes

DATA REPRESENTATION

EBCDIC GRAPHIC	BCL	DECIMAL VALUE	EBCDIC INTERNAL	HEX. GRAPHIC	EBCDIC CARD CODE	BCL CARD CODE	OCTAL	BCL INTERNAL	BCL EXTERNAL
Q		216	1101 1000	D8	11 8	11 8	50	10 1000	10 1000
R		217	1101 1001	D9	11 9	11 9	51	10 1001	10 1001
¢		224	1110 0000	E0	0 8 2				00 0000
S		226	1110 0010	E2	0 2	0 2	62	11 0010	01 0010
T		227	1110 0011	E3	0 3	0 3	63	11 0011	01 0011
U		228	1110 0100	E4	0 4	0 4	64	11 0100	01 0100
V		229	1110 0101	E5	0 5	0 5	65	11 0101	01 0101
W		230	1110 0110	E6	0 6	0 6	66	11 0110	01 0110
X		231	1110 0111	E7	0 7	0 7	67	11 0111	01 0111
Y		232	1110 1000	E8	0 8	0 8	70	11 1000	01 1000
Z		233	1110 1001	E9	0 9	0 9	71	11 1001	01 1001
0		240	1111 0000	F0	0	0	00	00 0000	00 1010
1		241	1111 0001	F1	1	1	01	00 0001	00 0001
2		242	1111 0010	F2	2	2	02	00 0010	00 0010
3		243	1111 0011	F3	3	3	03	00 0011	00 0011
4		244	1111 0100	F4	4	4	04	00 0100	00 0100
5		245	1111 0101	F5	5	5	05	00 0101	00 0101
6		246	1111 0110	F6	6	6	06	00 0110	00 0110
7		247	1111 0111	F7	7	7	07	00 0111	00 0111
8		248	1111 1000	F8	8	8	10	00 1000	00 1000
9		249	1111 1001	F9	9	9	11	00 1001	00 1001

NOTES

1. EBCDIC 0100 1110 also translates to BCL 11 1010.
2. EBCDIC 1100 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.
3. 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.
4. The remaining 189 EBCDIC codes are translated to BCL 00 0000 (? code).
5. The EBCDIC graphics and BCL graphics are the same except as follows:

BCL	EBCDIC
≥	' (single quote)
x (multiply)	!
≤	¬ (not)
≠	¯ (underscore)
←	

B 6700 EBCDIC/HEX CARD CODE

	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					•	$	,	#									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 +	-	0 -	0 -	9 0	9 -	9 0 -	9 0 -		

B 6700 EBCDIC/HEX CARD CODE

Use of the B 6700 EBCDIC/HEX Card Code Chart.

1. Locate the desired EBCDIC graphic code within the table.
2. The two-part Hexadecimal Code is read as follows:
   a. The first part is found in the vertical column above or below the desired EBCDIC code.
   b. The second part is found in the horizontal row either to the right or left of the desired EBCDIC code.

   (1) Examples:

   SYN = 32

   F = C6
3. The two-part Card Code is found in the same manner as HEX (2) except the zone and numeric bits are read from the very outer portion of the table.
   a. Examples:

   SYN = 9 2

   F = + 6
   b. The card code exceptions to the above procedure are enclosed in heavy lines on the chart and are defined below:

   (1) 00 = + 0981 (NUL)
   (2) 10 = + -981 (DLE)
   (3) 20 = - 0981
   (4) 30 = + -0981
   (5) 40 = BLANK
   (6) 50 = + (&)
   (7) 60 = - (-)
   (8) 70 = + - 0
   (9) CO = + 0 ({) ($\dot{\bar{0}}$)
   (10) DO = - 0 (}) ($\bar{0}$)
   (11) EO = 0 82 (\)
   (12) FO = 0 (0)
   (13) 61 = 0 1 (/)
   (14) E1 = -09 1
   (15) 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

HEXADECIMAL-DECIMAL CONVERSION TABLE

	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

	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

	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

	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

	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

	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

	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

	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

Wherever There's
Business There's

Burroughs

1058633 5-72 Printed in U. S. America