Burroughs B6500

Information Processing Systems

CHARACTERISTICS MANUAL

Burroughs
B 6500
INFORMATION PROCESSING SYSTEMS
CHARACTERISTICS MANUAL

Burroughs Corporation
Detroit, Michigan 48232

Printed in U. S. America 6-69 1032315

AA 950739 AA 32496

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

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Sales Technical Services, Burroughs Corporation, 6071 Second Avenue, Detroit, Michigan 48232.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

The Burroughs B 6500 Information Processing System is a medium-to-large scale system incorporating monolithic integrated circuitry, with a memory size of up to 3,145,728 bytes (524,288 words) of high-speed memory, with 1.2 microsecond or 600 nanosecond cycle times; dual input/output (I/O) multiplexor capability allowing up to 20 simultaneous operations; data communications software for remote computing and file manipulation; and, the ability to handle up to 256 peripheral units, including up to 36 billion bytes of disk storage.

A unique hardware design, developed from years of successful experience with the B 5500, has resulted in the parallel design of the B 6500 hardware and supporting software. Where traditionally hardware was designed prior to software development, parallel design assures that the hardware contains all necessary logic for efficient software packages, which in turn optimizes hardware capabilities. The B 6500 design affords a general "re-entrant" technique which permits multiple users to share a common object program. In addition, the system further expands the use of hardware stack organization used in the B 5500.

To solve dynamic storage allocation problems, the B 6500 system employs and expands upon the Burroughs descriptor method of segmentation, first used on the B 5500, in lieu of some form of fixed-sized "paging" technique. 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 on the B 6500. This part of the stack is used for multiple executions of the same program, thus eliminating many of the bookkeeping functions required to implement Master Control Program (MCP) re-entrancy.

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

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

This manual contains a description of the B 6500 characteristics and is intended to provide background technical information for those directly associated with data processing.

SECTION 1

SYSTEM DESCRIPTION

GENERAL

This section describes the main components of the B 6500 System. System design is based on program-independent modularity, the ability to process programs on available equipment without reprograming or recompiling, while, at the same time, making efficient use of that equipment. Machine language of the B 6500 is designed specifically to accept the common compiler languages of ALGOL, COBOL, and FORTRAN. The system automatically handles memory assignments, input/output assignments, program segmentation, and subroutine linkage, thus eliminating arduous programing tasks and reducing the likelihood of error. Programs are debugged and maintained at the source language level.

The Master Control Program (MCP) provides for over-all system coordination and control, reducing operator intervention to a minimum. By controlling the sequence of processing, initiating all input/output operations, and providing automatic handling procedures to meet virtually all processing conditions, the MCP continually obtains maximum use of system components. This centralized control permits changes in scheduling, system configuration, and program size to be readily accommodated.

SYSTEM ORGANIZATION

Generally, computer systems are organized around a central system which controls memory accesses, establishes I/O priority, etc. In the B 6500 this central control function has been disbursed throughout the entire system (see figure 1-1), permitting component removal without destruction of the system's ability to perform. The MCP recognizes a component omission and adjusts its task accordingly. Such "graceful degradation" allows operation to continue while corrections are made. Conversely, the B 6500 can be expanded on-site to accommodate mounting work-loads without the need for reprograming or recompiling. The MCP notes the larger hardware configuration and takes immediate advantage of the new environment, reallocating memory, processor, I/O, and peripheral resources to the object program.

Processor

The B 6500 can accommodate either one or two processors, both of which can access any portion of total memory. The B 6500 operates at a clock frequency of either 2.5 megacycles (model B 6503) or 5.0 megacycles (models B 6504 and B 6506). Following are some of the processor features:

a. Program code cannot be modified while in residence.

b. A hardware stack mechanism provides for efficient handling of temporary storage and subroutine requirements.

c. Control bits added to each word provide for efficient MCP or hardware action, depending upon the control bit configuration.

d. Memory protection (preventing one program from affecting another) is provided for by a combination of hardware and software features. Hardware features include automatic detection of a program attempt to index beyond an assigned data area and the use of control bits, which prevent a user program from changing program segments, data descriptors, segment descriptors, memory links, MCP tables, etc. The software sets the control bits of those words which are not available to the object program.

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

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

An aggressive hardware method of detecting and servicing system interrupts contributes to the ability of the B 6500 to process a mix of independent programs in an efficient manner. Under the constant, automatic management of the MCP, multiprocessing is the normal mode of operation. With one processor in the configuration, multiprograming (interleaved programing) is the method employed. Dual-processor B 6500 Systems combine both multiprograming and parallel processing. (The ability to multiprogram, parallel process, or both has been defined as multiprocessing.)

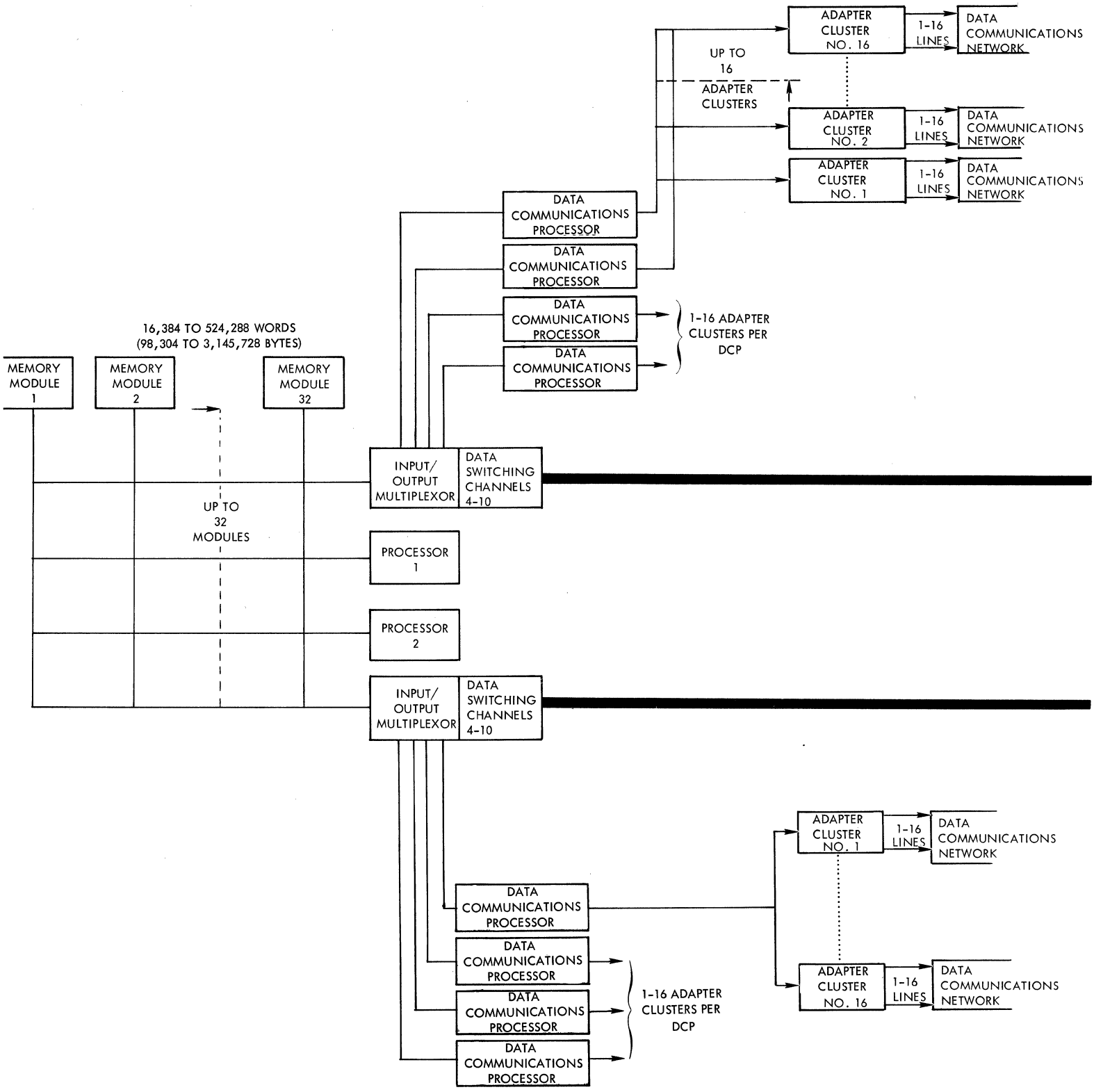
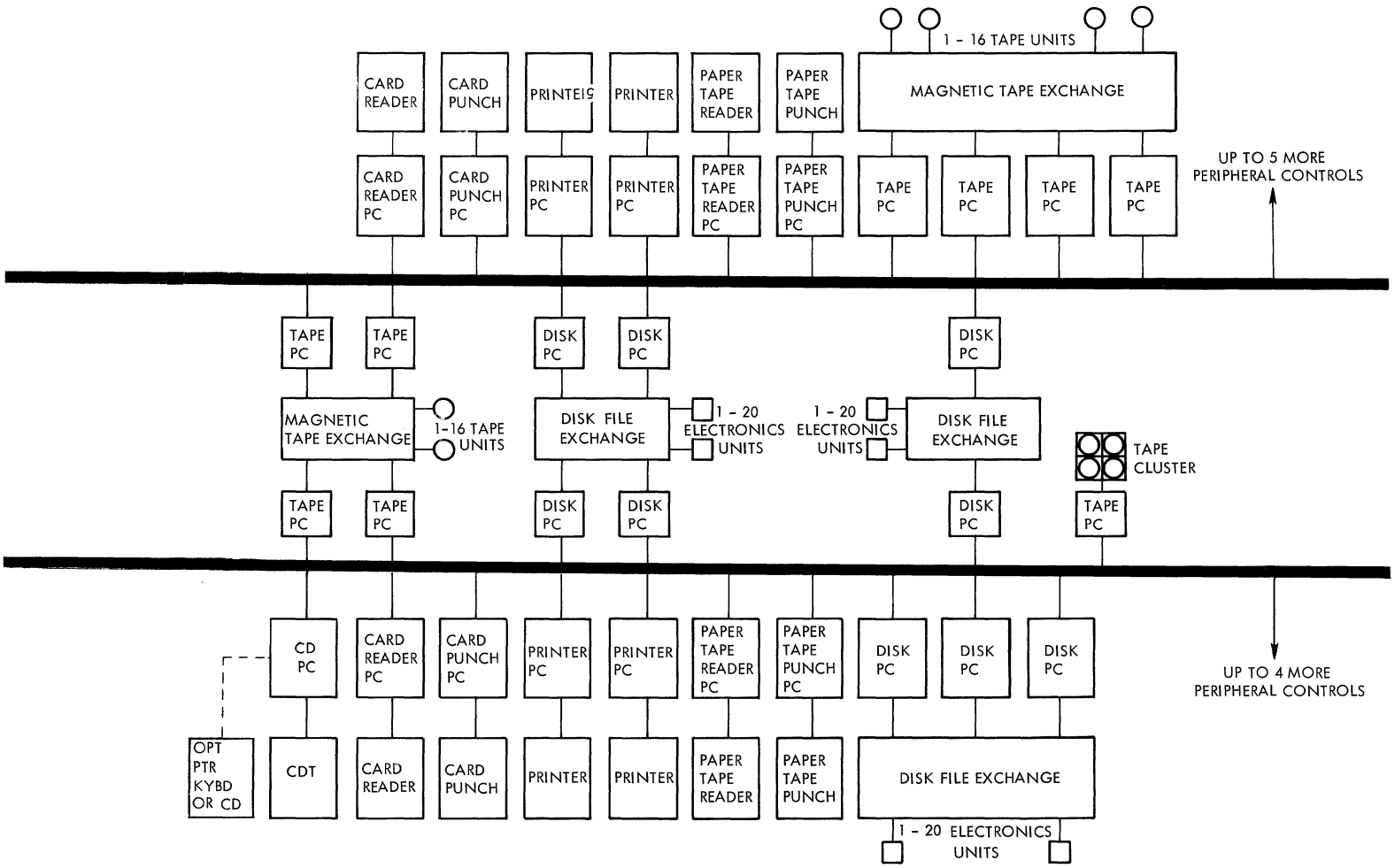


Figure 1-1. B 6500 Representative Configuration


1 - 16 TAPE UNITS
CARD READER
CARD PUNCH
PRINTER
PRINTER
PAPER TAPE READER
PAPER TAPE PUNCH
MAGNETIC TAPE EXCHANGE
CARD READER PC
CARD PUNCH PC
PRINTER PC
PRINTER PC
PAPER TAPE READER PC
PAPER TAPE PUNCH PC
TAPE PC
TAPE PC
TAPE PC
TAPE PC
UP TO 5 MORE PERIPHERAL CONTROLS
TAPE PC
TAPE PC
DISK PC
DISK PC
DISK PC
MAGNETIC TAPE EXCHANGE
1-16 TAPE UNITS
DISK FILE EXCHANGE
1 - 20 ELECTRONICS UNITS
1 - 20 ELECTRONICS UNITS
DISK FILE EXCHANGE
TAPE CLUSTER
TAPE PC
TAPE PC
DISK PC
DISK PC
DISK PC
TAPE PC
CD PC
CARD READER PC
CARD PUNCH PC
PRINTER PC
PRINTER PC
PAPER TAPE READER PC
PAPER TAPE PUNCH PC
DISK PC
DISK PC
DISK PC
UP TO 4 MORE PERIPHERAL CONTROLS
OPT PTR KYBD OR CD
CDT
CARD READER
CARD PUNCH
PRINTER
PRINTER
PAPER TAPE READER
PAPER TAPE PUNCH
DISK FILE EXCHANGE
1 - 20 ELECTRONICS UNITS

Main Memory

Main memory is expandable from one to eight modules on a B 6503 processor system, and from one to 32 modules on B 6504 or B 6506 processor systems. Each memory module contains 16,384 (52-bit) words, permitting a current maximum of 524,288 words of memory. Provision is made for future expansion beyond one million words of storage.

Each memory word consists of 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. A 20-bit binary addressing scheme provides for addressing all 524,288 words of memory. Main memory cycle time is 600 nanoseconds for B 6506 processor systems, and 1.2 microseconds for B 6503 and B 6504 processor systems.

Each memory cabinet includes a memory test facility used for fault detection and isolation of the three memory modules contained within the cabinet. When the unit test facility is being used to check out one of the modules, the others can continue in use by the system.

The 8-bit Extended Binary Coded Decimal Interchange Code (EBCDIC) is the primary internal code of the B 6500, with the system also able to accept information recorded in 4-bit and 6-bit formats. Because many data communications devices transmit in American Standard Code for Information Interchange (ASCII), provision is made for direct processing of ASCII code as well.

Second Level Memory

Second level memory for the B 6500 consists of Burroughs head-per-track disk file subsystems. Program and data segments stored on disk are automatically transferred to main memory as required, by action of the MCP. Through the use of an optional disk file exchange and controls, up to four transfer operations to or from each disk file subsystem can occur simultaneously. Multiple disk file subsystems totaling up to 36 billion bytes of storage can be attached to a B 6500 System.

Input/Output Multiplexor

The Input/Output Multiplexor and associated peripheral control modules are used to control the transfer of data between memory and all peripheral equipment, independent of the processor. The multiplexor receives instructions from the processor and, together with its associated peripheral controls, executes these instructions. One or two multiplexors may be attached to a B 6500. Each multiplexor is capable of processing up to ten simultaneous I/O operations from up to 20 peripheral controls, handling a combined maximum of 256 peripheral devices.

Each multiplexor provides three separate and independent units:

a. Data switching channels which provide the necessary linkage between the peripheral device (excluding data communications) and main memory.
b. Data communications processors which permit interfacing of remote devices to the B 6500.
c. A real time adapter which permits interfacing of real time devices such as wind tunnels and rocket stands.

The number of available data switching channels determines the number of simultaneous I/O operations that can be performed, that is, the channels float, being assigned to peripherals upon demand for service and released back to the multiplexor for reassignment upon receipt of an I/O complete.

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. Large controls utilize a two character buffer and small controls a one character buffer. Each multiplexor can accommodate up to ten large and ten small controls.

The maximum configuration using two multiplexors (20 controllers per multiplexor) can be further expanded through the use of disk file and magnetic tape exchanges. Figure 1-2 illustrates a possible magnetic tape subsystem. Figure 1-3 illustrates a possible disk file subsystem.

Data Communications Processor

Up to four Data Communications Processors (DCP) can be connected to an I/O multiplexor. From one to sixteen adapter clusters can be attached to each DCP, and from one to 16 remote lines can be attached to each adapter cluster, providing control for up to 256 remote lines. Since a single B 6500 may have up to eight DCPs, a maximum of 2,048 remote lines can be serviced by a single system.

Real Time Adapter

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

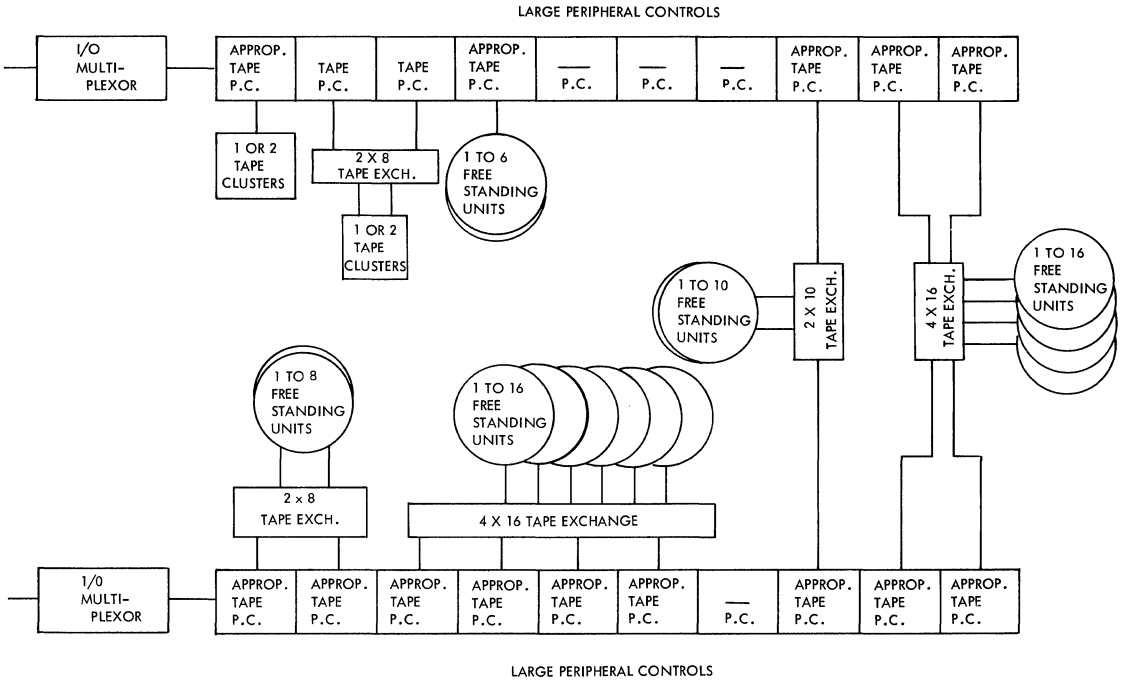


Figure 1-2. Possible Magnetic Tape Subsystem

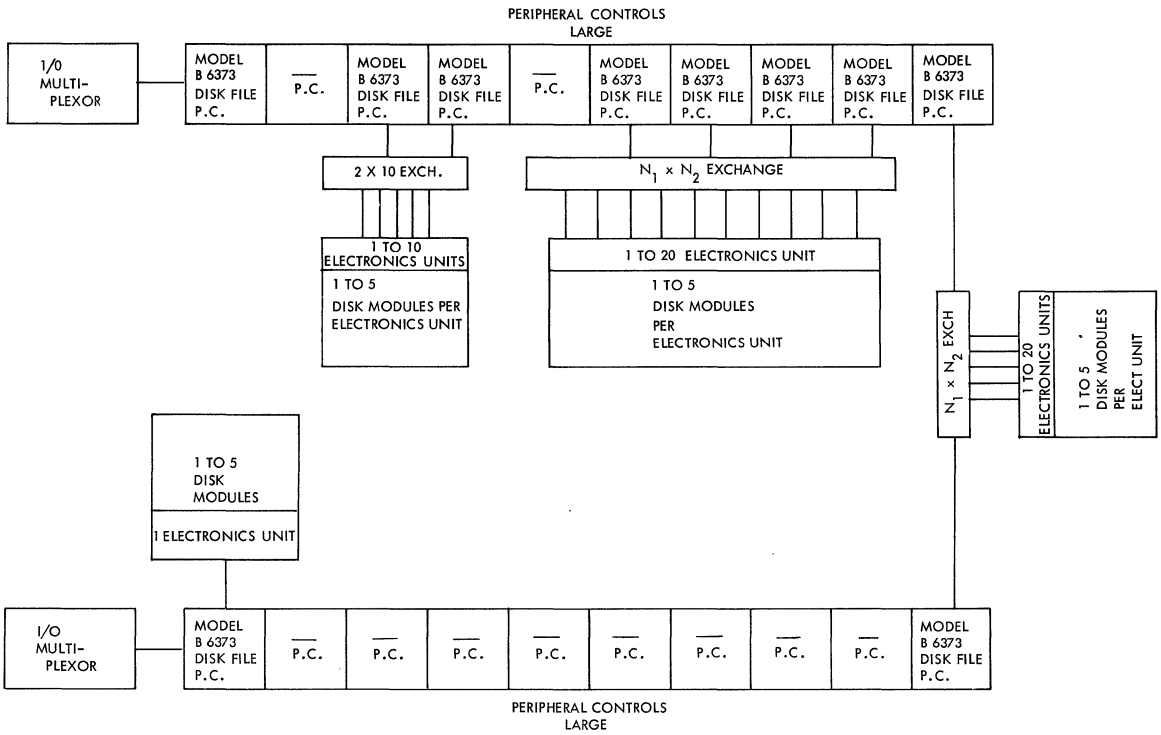


Figure 1-3. Possible Disk File Subsystem

SECTION 2

MAIN MEMORY

GENERAL

Main memory cycle time for the B 6500 can be either 600 nanoseconds or 1.2 microseconds. All processor models can accommodate from one to 32 memory modules, with the exception of the B 6503 which can accommodate a maximum of eight modules. Each module consists of 16,384 words, with each word having 48 information bits, three control bits, and a parity bit.

MEMORY ORGANIZATION

Main memory in the B 6500 is organized so that any memory module can send information to, or receive information from both processors and both I/O multiplexors over any one of four information busses (see figure 2-1).

As a word transfers along a buss, it is "inspected" by the circuitry in each module to determine whether it should go to that particular module. This arrangement eliminates the need for a central control to establish a linkage directing the word to the proper module.

Information is transmitted along the buss in parallel, as illustrated in figure 2-2.

Each module examines six bits of each address to determine if they apply to that particular module. If so, the module sets the linkage to receive the word. Two-hundred nanoseconds after the memory cycle is initiated, the module grants access. In another 200 nanoseconds, the word is available from memory, and 200 nanoseconds later the word is available in the processor or I/O multiplexor

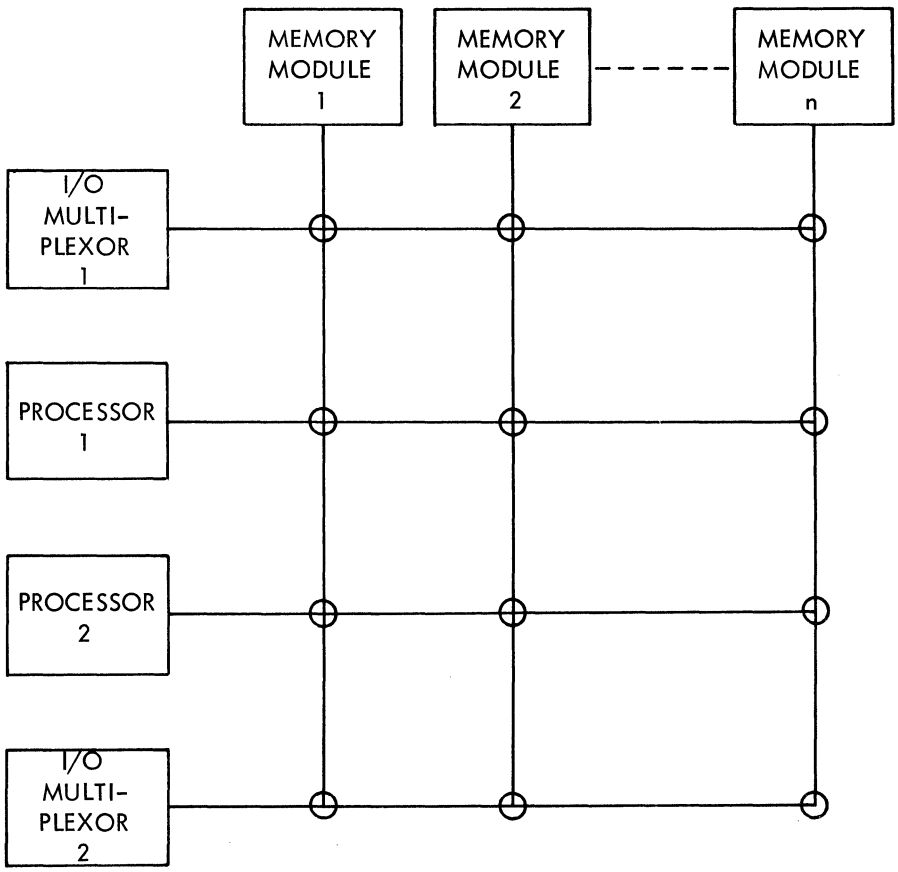


Figure 2-1. Memory Organization

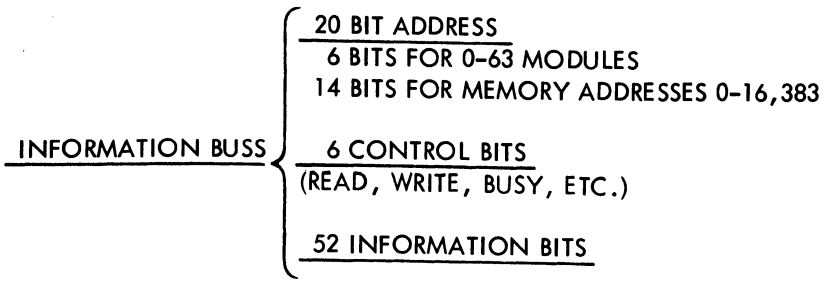


Figure 2-2. Information Transmission

register. Operation of each memory module is independent of the operation of any other memory module. Memory cycles can occur simultaneously within any four modules. Virtually all of memory can be overlayed, and all code is re-entrant.

MEMORY PROTECTION

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

Using the memory Read With Lock operation, the information read from memory is interchanged with the word in the top of the stack. All data and address transfers are parity checked.

MEMORY PACKAGING

Up to three memory modules of 16,384 words each, the memory power supply, and the memory exchange equipment associated with each module are housed in a single cabinet. All of the memory modules in the cabinet can be interlaced, with odd addresses serviced by one module and even addresses serviced by another. This feature can be enabled or disabled by field engineering. Each system includes a test facility which can exercise any of the memory modules. When the test facility is being used with one of the memory modules, the other modules can be used by the system, provided the module being tested is not interlaced. If it is, the option must be disabled before testing can take place.

SECOND LEVEL MEMORY

Burroughs unique head-per-track disk file subsystem provides the user with virtually unlimited expansion capability. The 20 to 60 millisecond average access time of the various disk file models permits extremely large programs and data segments to be stored on the disk and brought into main memory by the MCP only when required. Additional information on the disk file subsystem is provided in Section 6 of this manual.

SECTION 3

PROCESSOR

GENERAL

All B 6500 processors are parallel machines and operate at a clock frequency of either 2.5 or 5.0 megacycles. Processors with different clock rates cannot be intermixed on the same system. The processor operates in the binary mode, but has extensive string manipulation capabilities that allow it to handle 4-bit, 6-bit, and 8-bit characters.

Extensive interrupt facilities are provided to allow the intervention of the Master Control Program for resource allocation and object program error detection. Each processor can handle its own interrupts without disturbing the other. The processors may also share the processing of external interrupts generated by input/output operations occurring on either multiplexor. The command structure in conjunction with a stack provides for the implementation of string notation and the automatic linking of subroutines.

The MCP for the B 6500 makes use of two types of 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 routine to provide extremely accurate timing information and also can be read by application programs. This clock is associated with the multiplexor and runs continuously, even when the processors are halted. The interval timer is a clock (one associated with each processor), the purpose of which is to provide a predetermined timed interrupt for "time-slicing," loop hang-up, etc. This interval can range from 512 microseconds to one second, in 512 microsecond intervals.

PROCESSOR STATES

The processor can operate in either of two states: control state under the MCP or normal state for user programs and certain MCP functions. In a dual-processor system each processor handles its own internal interrupts. Either processor may handle external interrupts. Both processors may be in control state at the same time.

Control State

Entry into control state occurs when the processor is started as a result of certain interrupt conditions, procedure entry or exit, or disable interrupt operations. In control state, the processor can execute privileged instructions not available in normal state and external interrupts are inhibited. Exit from control into normal state occurs whenever the MCP initiates a normal state procedure, exits back to a normal state procedure or executes an Enable External Interrupts operator. After an interrupt the user program return may or may not be to the program in process when the interrupt occurred.

Normal State

Normal state excludes use of privileged instructions required by the MCP, permits hardware detection of invalid operators, and enforces memory protect and security facilities. Exit from normal state occurs as a result of an interrupt condition or by a call to a control state procedure; e.g., to initiate I/O. Many MCP functions can be run in normal state.

INTERRUPT SYSTEM

For internal interrupts, each processor in a B 6500 System is provided with a private, internal interrupt network. Internal interrupts associated with a processor are fed directly into this network and are stacked local to that processor. External interrupts, on the other hand, may be serviced by either processor.

A processor responds to an interrupt by entering a subroutine. The interrupt causes the processor to take the following action:

a. Mark the stack.
b. Insert an Indirect Reference Word into the stack, which links to a reserved location of the program stack where a link to the MCP interrupt routine has been stored.
c. Push all pertinent registers into the stack.
d. Insert into the stack an integer value defining the interrupt.

e. Insert a second parameter into the stack, giving address information about interrupt.
f. Enable an Enter operator.

The subroutine mechanism then causes an entry to an MCP routine to process the specific interrupt. The MCP reactivates the interrupted object program by returning through the normal subroutine mechanism.

Internal (Processor Dependent) Interrupts

SYLLABLE DEPENDENT PROCESSOR INTERRUPTS.

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

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

Within a processor, no more than one syllable dependent interrupt is set at any one time.

SYLLABLE INDEPENDENT PROCESSOR INTERRUPTS.

Following is a list of syllable-independent interrupts:

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

External Interrupts

External interrupts are fed into the processor interrupt system. If the interrupt network is disabled on one processor, the external interrupt signal is routed to the second processor, since both processors in a dual-processor system are able to respond and process external interrupts independently and simultaneously. The ability of either processor to handle interrupts is made possible through the use of a distributed interrupt network and the ability of the system to have both processors in control state at the same time. The activities of two processors in control state are coordinated (interlocked) through the use of the Read With Lock mechanism. Each processor is capable of processing its own interrupts without disturbing the other processor. If both processors are handling interrupts, additional interrupts are stacked 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 a processor acknowledges an external interrupt and enters the interrupt sequence.

MEMORY PROTECTION

Bit 48 of each word in main memory is a memory protect bit. This bit is ON in all program words, indirect reference words, data descriptors, program descriptors, main memory storage links, and processor generated control words. With the exception of Stack adjustment and the Over-write operation, an attempt by a processor to write into a location containing a word that has the memory protect bit set ON will cause a memory protect interrupt to be set in the processor interrupt register.

INFORMATION REPRESENTATION

Each processor word contains 48 information bits, three control bits, and one parity bit (figure 3-1).

Bit positions 50 through 48 are control bits which are used to identify descriptors, provide memory protection, describe the type of data, and perform other control functions. These control bits are inaccessible to normal state (user) programs.

It is impossible to store into a word which has a value of 1 in bit position 48 except by use of the Overwrite Store operator. The various combinations of bits 50, 49, and 48 and their functions are as follows:

a. 000 – single precision operand.
b. 010 – double precision operand.
c. 110 – software control word.
d. 001 – indirect reference word.

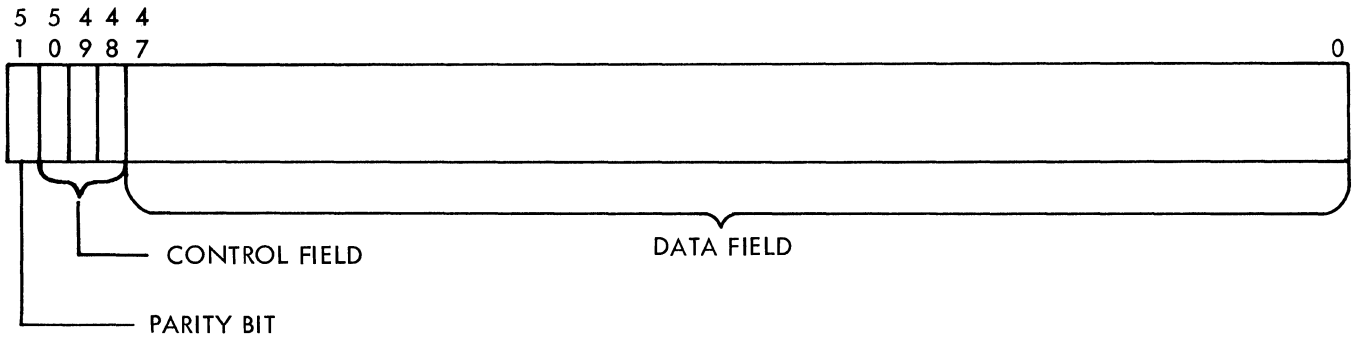


Figure 3-1. Basic Word Structure

e. 011 – storage links, code, segment descriptors, mark stack control words (partial or complete), return control words.
f. 101 – data descriptors and string descriptors.
g. 111 – program control words.
h. 100 – step index word.

Operands

SINGLE PRECISION.

The structure of a single precision operand (data word) is illustrated in figure 3-2:

a. Bits 50-48, a control field of 000.
b. Bit 47, unused.
c. Bit 46, designates whether the integer and/or mantissa is positive (bit 46 = 0) or negative (bit 46 = 1).
d. Bit 45, designates whether the exponent is positive (bit 45 = 0) or negative (bit 45 = 1), if the exponent is other than zero.
e. Bits 44-39, contain the exponent.
f. Bits 38-0, contain the mantissa.

The exponent is a binary number and, with its associated sign, is an octal scale factor for the mantissa. The exponent is used for automatic scaling of the mantissa when performing arithmetic, comparison, and integer operations. The range of the exponent is ± 63. The magnitude of the operand is obtained by multiplying the integer value of the mantissa by eight raised to the signed power of the exponent.

$$V = (\pm M \times 8^{\pm E})$$

Where V is the value of the number, ±M is the signed value of the mantissa, and ±E is the signed value of the exponent. The range of numbers that can be expressed is:

$$[(8^{+13} - 1) \times 8^{+63}] \rightarrow [1 \times 8^{-51}] \text{ and } 0$$

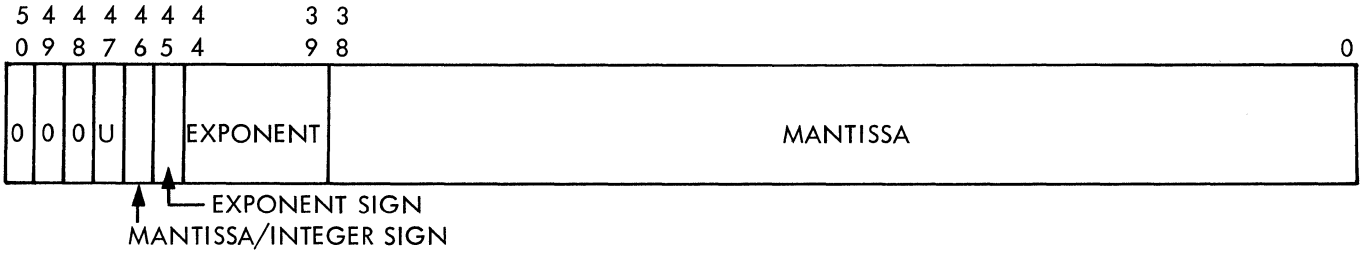


Figure 3-2. Single Precision Operand

DOUBLE PRECISION.

A double precision operand (figure 3-3) is comprised of two 48 bit words. The first word of the operand is identical to the single precision operand except for a control field of 010 which indicates to the processor that this is only one-half of a pair of words. Bits 38-0 of the first word represents the most significant portion of the mantissa and bits 38-0 of the second word the least significant portion of the mantissa. The exponent for a double precision operand is expressed by using bits 47-39 of word two as the most significant portion and bits 44-39 of word one as the least significant portion.

The range of numbers that can be expressed is as follows:

$$[(8^{+13} - 1) \times 8^{+32,767}] \rightarrow [1 \times 8^{-32,755}] \text{ and } 0$$

All arithmetic operators in the processor automatically perform properly for operands which identify themselves as double precision.

Descriptors

Descriptors are words used to locate data and program areas in memory, and to describe these areas for control purposes. Descriptors are the only words containing absolute addresses which can be used by a normal state program; however, the normal state program cannot alter them.

DATA DESCRIPTORS.

Data descriptors are used for referring to data areas, including input/output buffer areas. The data descriptor defines an area of memory starting at the base address contained in the descriptor. The size of the memory area in number of words is contained in the length field of the descriptor. Data is normally referred to by a Value Call syllable or Evaluate Name syllable. Data descriptors may directly reference any memory word address from 0 through 524,288.

The structure of the data descriptor is illustrated in figure 3-4 and contains the following:

a. Bits 50-48, a control field of 101.

b. Bit 47, a presence bit, indicates the presence or absence of data. A 0-bit causes a presence bit interrupt whenever the descriptor is accessed by a program to obtain non-present data. A 1-bit indicates that the data requested by a program is in memory.

c. Bit 46, a copy bit. A 0-bit indicates that this is the original descriptor for the particular data area. A 1-bit indicates that this descriptor is a copy of the original descriptor.

d. Bit 45, an index bit. A 0-bit indicates an indexing operation will be performed when the descriptor is addressed to obtain data. A 1-bit indicates that indexing has already taken place and the index value is stored in bit positions 39 through 20 (Length/Index).

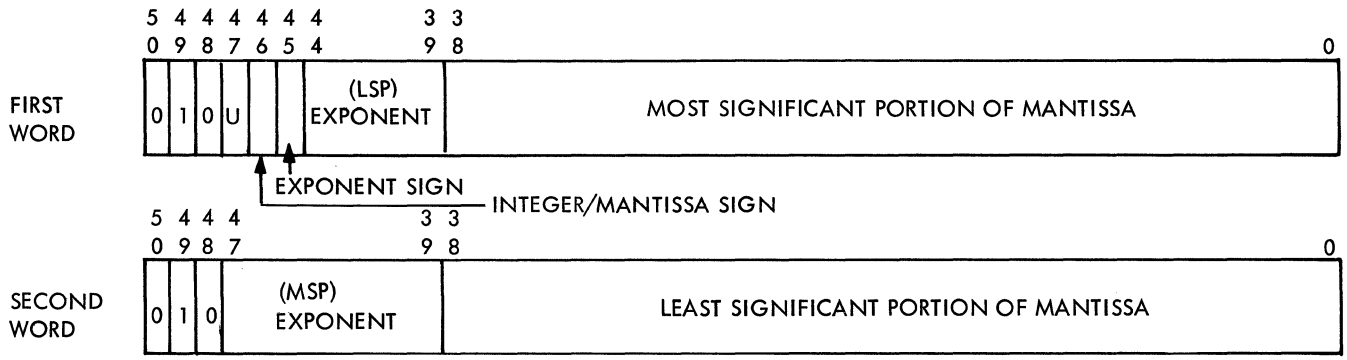


Figure 3-3. Double Precision Operand

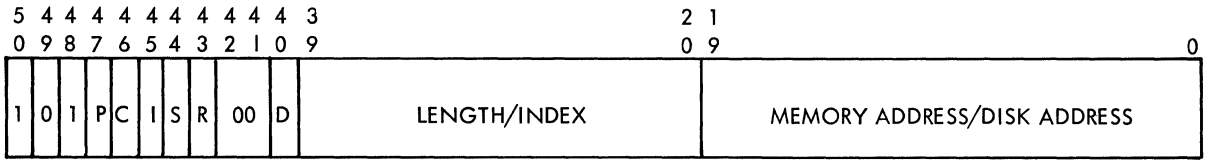


Figure 3-4. Data Descriptor

e. Bit 44, a segmented bit. A 0-bit indicates that the data is not segmented. A 1-bit indicates that the data area is segmented (segment size is 256 words).

f. Bit 43, a read-only bit. A 0-bit indicates that the data may be referenced for reading or writing. A 1-bit indicates a read-only condition.

g. Bits 42-41, must equal zero for data descriptors.

h. Bit 40, a double precision bit. A 0-bit indicates single precision, a 1-bit indicates double precision.

i. Bits 39-20, contain either the length of the memory area (bit 45 = 0) or an index value (bit 45 = 1). If bit 45 equals 0, it indicates that the descriptor has not been indexed. This field can be used to perform size checking before the indexing operation. If bit 45 equals 1, it indicates that the descriptor has been indexed. For a double precision operation, the index is multiplied by two prior to index size checking. It is the result of this multiplication that is stored in the index field.

j. Bits 19-0, 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 will contain the disk address of non-present data. If the presence bit equals 0 and the copy bit equals 1, this field will contain the absolute memory address of the original descriptor.

STRING DESCRIPTOR.

The string descriptor (figure 3-5) is used to refer to data areas organized as 4-, 6-, or 8-bit characters. The descriptor defines an area of memory starting at the base address contained in the descriptor. The size of the memory area is defined by the length field. The string descriptor contains the following information:

a. Bits 50-48, a control field of 101.

b. Bit 47, a presence bit. A 0-bit indicates that a presence bit interrupt will occur if the descriptor is addressed to obtain non-present data. A 1-bit indicates no interrupt will occur because the data requested is present in main memory.

c. Bit 46, a copy bit. A 0-bit indicates that this is the original descriptor for the particular data area. A 1-bit indicates that this descriptor is a copy of the original descriptor.

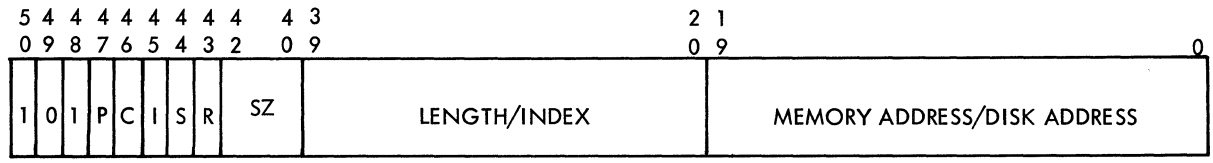


Figure 3-5. String Descriptor

d. Bit 45, an index bit. A 0-bit indicates an indexing operation will be performed when the descriptor is addressed to obtain data. A 1-bit indicates that indexing has already taken place and the index value is stored in bit positions 39 through 20 (Length/Index).
e. Bit 44, a segment bit. A 0-bit indicates that the data area is not segmented; a one-bit indicates that the data area is segmented (segment is 256 words).
f. Bit 43, a read-only bit. A 0-bit indicates that the data may be referenced for reading and writing. A 1-bit indicates that the data can be read only.
g. Bits 42-40. 100 indicates an 8-bit character; 011 indicates a 6-bit character; 010 indicates a 4-bit character.
h. 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 will contain the length of the area, in 4-, 6-, or 8-bit characters. The descriptor will not have been indexed. This field is used to perform size checking prior to indexing operations. When indexing is required, it is performed as follows. Prior to indexing, the index is divided by 6 for 8-bit data, by 8 for 6-bit data, and by 12 for 4-bit data. The result of this division is stored in the word index field (see figure 3-6). The remainder of this division is stored in the byte index field.

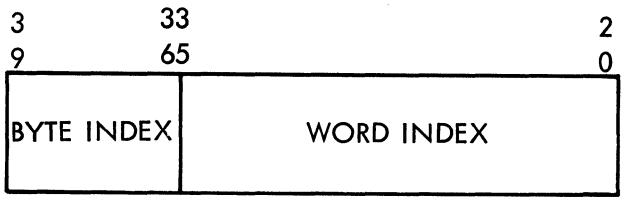


Figure 3-6. Byte/Word Index Field

When bit 45 equals 1, bits 39-20 contain a byte/word index in the format illustrated in figure 3-6. The descriptor will have been indexed.

i. Bits 19-0, contain either a main memory or disk address. If the presence bit (bit 47) equals 1, the field will contain a memory address of data. If both the presence bit and the copy bit (bit 46) equal 0, the field will contain the disk address of non-present data. When the presence bit equals 0 and the copy bit equals 1, the field will contain the absolute memory address of the original descriptor.

SEGMENT DESCRIPTOR.

The segment descriptor (figure 3-7) is used to locate a program segment and contains the following information:

a. Bits 50-48, a control field of 011.
b. Bit 47, a presence bit. A 0-bit causes a presence bit interrupt if the descriptor is addressed to obtain a non-present segment. A 1-bit does not generate an interrupt, indicating that the segment requested is present in main memory.
c. Bit 46, a copy bit. A 0-bit indicates that this is the original segment descriptor. A 1-bit indicates that this is a copy of the original segment descriptor.
d. Bits 45-40, unused.
e. Bits 39-20, specify the length of the program segment in words.
f. Bits 19-0, contain either the main memory address or the disk file address. If the present bit (bit 47) equals 1, the field will contain the main memory address of the program segment. If both the presence bit and the copy bit (bit 46) equal 0, the field will contain the disk address of the non-present program segment. If the presence bit equals 0 and the copy bit equals 1, the field will contain the absolute memory address of the original program segment descriptor.

Mark Stack Control Word

The Mark Stack Control Word (MSCW), along with the Program Control Word/Return Control Word, provides a linking mechanism for tracing the history of previous control register settings through the stack. The MSCW is placed in the stack as a result of executing a Mark Stack operator. The MSCW is organized as illustrated in figure 3-8 and provides the following data:

a. Bits 50-48, a control field of 011.
b. Bit 47, a different stack bit. A 0-bit indicates that the MSCW refers to the current stack. A 1-bit indicates that the MSCW refers to a different stack.
c. Bit 46, an environment bit. A 0-bit indicates that an inactive MSCW was generated directly by the Mark Stack operator and that the corresponding procedure entry has not been

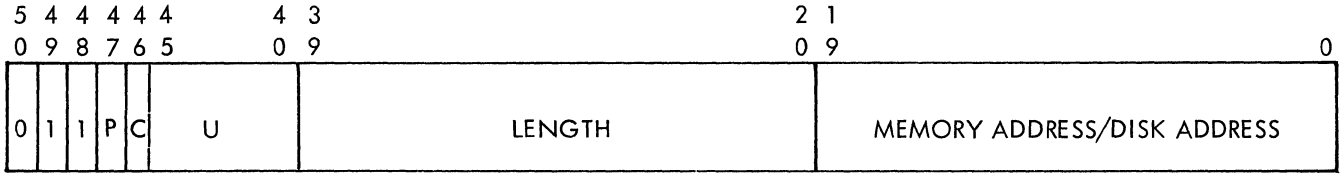


Figure 3-7. Segment Descriptor

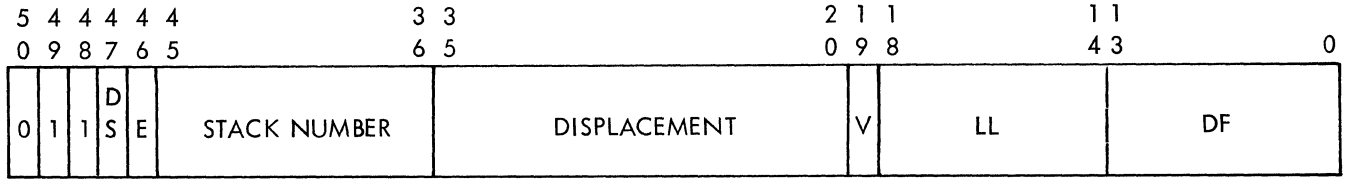


Figure 3-8. Mark Stack Control Word

performed. A 1-bit denotes an active MSCW generated upon entry into a procedure, at which time the environment fields (stack number, displacement, and value fields) are stored into the Mark Stack Control Word.

d. Bits 45-36, a stack number field which contains the number of the stack in use.

e. Bits 35-20, a displacement field, which, when added to the stack base address, locates a Mark Stack Control Word.

f. Bit 19, a value bit. A 0-bit indicates that the MSCW was generated during an operation that will be restarted from the beginning. A 1-bit indicates that the operator must continue after the Exit or Return which refers to this MSCW (e.g. an accidental entry on a Value Call).

g. Bits 18-14, denotes the level of the procedure being entered.

h. Bits 13-0, denotes the stack history. This field is used to indicate where in the stack the preceding MSCW is located (i.e. the previous "F" register setting).

Program Control Word/Return Control Word

The Program Control Word (PCW)/Return Control Word (RCW) and the Mark Stack Control Word are used for entry into a procedure. The organization of the PCW/RCW is illustrated in figure 3-9 and contains the following:

a. Bits 50-48, a control field of either 111 to denote a PCW or 011 to denote an RCW.

b. Bits 45-36, stack number for a PCW/RCW: Bit 47, for external sign. Bit 46, for overflow. Bit 45, for True/False Flip-Flop. Bit 44, for Float Flip-Flop.

c. Bits 35-33, define the program syllable within the word located by PIR.

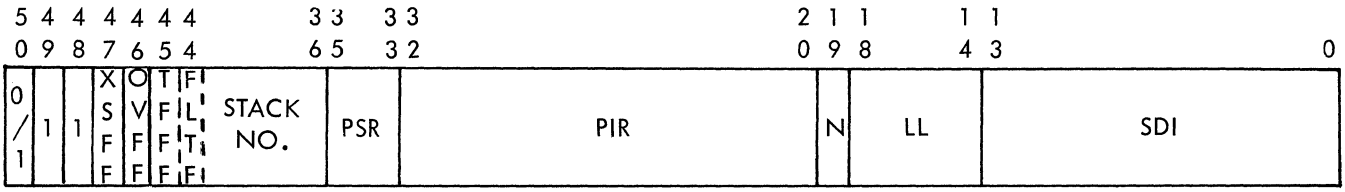


Figure 3-9. Program Control Word/Return Control Word

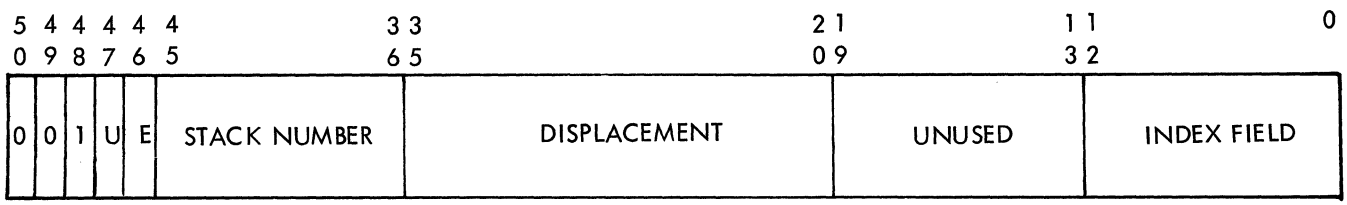


Figure 3-10. (Stuffed) Indirect Reference Word

d. Bits 32-20, used as an index to the Base Program Register.
e. Bit 19, denotes the presence of either a normal state program (bit = 0) or a control state program (bit = 1).
f. Bits 18-14, denote either the level of the procedure being entered when the word is used as a PCW or the level of the old procedure when an RCW is generated at the time of procedure entry.
g. Bits 13-0, a segment descriptor index. Bits 12 through 0 specify the value to be added to the address located by either display register 0 or 1. When bit 13 equals 0, display register 0 is selected; when bit 13 equals 1, display register 1 is selected.

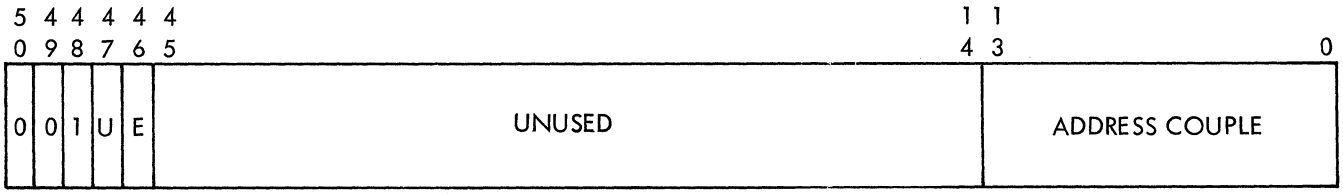


Figure 3-11. (Normal) Indirect Reference Word

Indirect Reference Word

The Indirect Reference Word (IRW) provides addressing information for the execution of a subsequent operator; e.g., Evaluate Descriptor, Value Call, Store, etc. The IRW is placed in the stack by a Name Call operator. The IRW has two forms: Stuffed (figure 3-10) and Normal (figure 3-11).

a. Bits 50-48, a control field of 001.
b. Bit 47, unused.
c. Bit 46, an environment bit. A 1-bit indicates a Stuffed IRW. A 0-bit indicates a Normal IRW.
d. Bits 45-36, stack number. When bit 46 equals 1, specifies the stack number to be used by the subsequent operator.
e. Bits 45-14, when bit 46 equals 0, unused.
f. Bits 35-20, displacement field. When bit 46 equals 1, this value added to the stack base address locates a Mark Stack Control Word.
g. Bits 12-0, index field. When bit 46 equals 1, the index value is added to the contents of the display register specified by the Mark Stack Control Word.
h. Bits 13-0, when bit 46 equals 0, it is subdivided into two functional fields. Each field is variable in length. The first subfield, designated LL, is used to select one of 32 display registers. The second subfield contains an index value which is added to the contents of the selected display register to form an absolute address. The length of the subfields are defined by the current program level as follows:

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

NOTE

The bit order of the LL field is inverted.

INSTRUCTION FORMAT

Each B 6500 instruction (operator) is from one to 12 syllables in length; each syllable contains 8 bits. Syllables are executed sequentially from left to right. When the program word has been exhausted, the next sequential program word is fetched from memory. There are three types of syllables:

a. Operators – one to 12 syllables.
b. Value Calls – two syllables.
c. Name Calls – two syllables.

The two high-order bits determine whether a syllable is an Operator, a Value Call, or Name Call, as follows:

Identification	Syllable Type	Function
1X	Operator	Performs the specified operation
00	Value Call	Brings an operand into the stack
01	Name Call	Places the name into the stack

(X = makes no difference what the value is)

REGISTERS

Information Registers

The information registers (A, B, C, X, Y, and P) are 51-bit registers which communicate both with memory and each other. A, B, X, and Y are the active top of the stack registers; P holds the current program word. C is a scratch pad register for the processor.

Address Registers

There are forty-eight 20-bit registers provided in two integrated chip (I.C.) memories. These two I.C. memories are associated with an address adder so that a word in one I.C. memory may be incremented by a word in the other I.C. memory and the result used to address main memory.

STACK MECHANISM

The stack is an area of memory assigned to a program which serves to provide basic program and data references associated with the program, as well as a facility for the temporary storage of data and dynamic program history. When a program is activated, high-speed registers A, B, X, and Y are linked to the program's stack memory area. This linkage is established by the S register which contains an address which points to the last word stored in the stack memory area. The top of the stack registers function to extend the program's stack memory into a quick access environment for data manipulation.

Data is brought into the stack through the top of the stack registers. The content of the S register is increased by 1 as each word is pushed from the top of the stack registers into the stack memory area and is decreased by 1 as a word is withdrawn from the stack memory area and placed in the top of the stack registers. As a result, the S register continually points to the last word placed in the program's stack memory area.

The allocation of memory in figure 3-12 shows three areas set aside as stacks for multiprograming (interleaved processing) of three different jobs. Since the stack is used as a flexible work area, its absolute location is unimportant. The stack in use at any given moment is associated with the A, B, X, and Y registers, with the memory address at the top of the stack recorded in the S register. When the processor switches from one job to another, any information held in the A, B, X and Y registers is automatically stored in the memory stack prior to transfer of control. The current setting of the S register is always preserved for re-entry to the job. The Master Control Program controls the processing sequence of various jobs and allocates stack memory space according to the number and the size of programs in memory at any one time. Switching from job to job, and therefore from stack to stack, is controlled by the Master Control Program.

The stack memory area assigned to a program is set by the stack base address and the stack limit register. Both stack overflow and underflow cause a program interrupt.

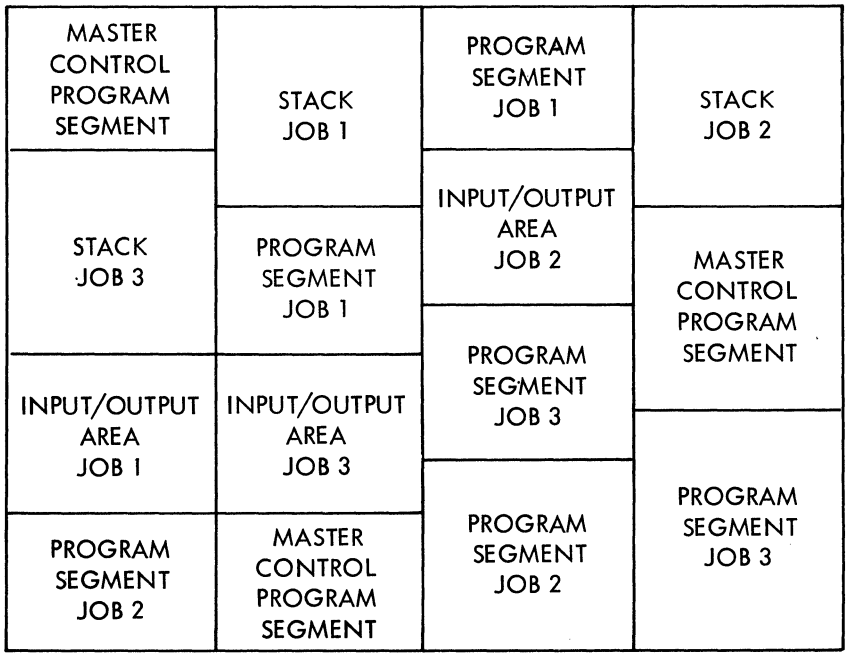


Figure 3-12. Example Allocation Of Main Memory

Stack Adjustment

The contents of the top of the stack registers are maintained automatically by the processor hardware, in accordance with the environmental demands of the current operator. When the current operator brings data into the stack, the top of the stack registers are adjusted to accommodate the incoming data, and, where necessary, the registers themselves are pushed into the stack memory area. The S register is brought from the stack memory area to the top of the stack registers as required.

Cactus Stack

An additional feature of the stack mechanism is the cactus stack structure, which provides for parallel computation algorithms, as required for many simulation languages. Since the MCP controls parallel processes, it is possible to place the segment dictionary descriptors in the common trunk of a "tree" of stacks for those jobs which are identical. For example, if five FORTRAN compilations are being carried out at one time, the stacks for those compilations would be organized as shown in figure 3-13.

All programs are entered and exited through the segment descriptors common in the base stack; all references to those descriptors are relative. Entrance to, or removal of any given program segment from memory, therefore, is achieved by changing the presence bit in that segment descriptor. No stack search of any kind is required. This is automatically reflected in all five copies of the example in figure 3-13. Re-entrance is provided automatically for all programs, not compilers alone, since compilers are themselves object programs.

OPERATORS (INSTRUCTIONS)

Operators may be 1 to 12 syllables in length; each syllable is 8 bits in length. The first syllable of each operator determines the number of additional syllables associated with that operator. Upon completion of each operator, the program counter points beyond all of the syllables comprising the operator.

Following is a discussion of the types of available operators, including a list of specific operators in each category. A detailed description of individual operators will be provided in a subsequent reference publication.

Invalid Instruction Detection

Detection of an invalid instruction terminates the operator and a programed operator interrupt is set in the processor interrupt register. Invalid instructions are detected by the following methods:

a. Testing for unassigned operator codes.
b. Testing for any value other than 011 in bit positions 50 through 48 of a program word (an attempt to execute something which is not code).

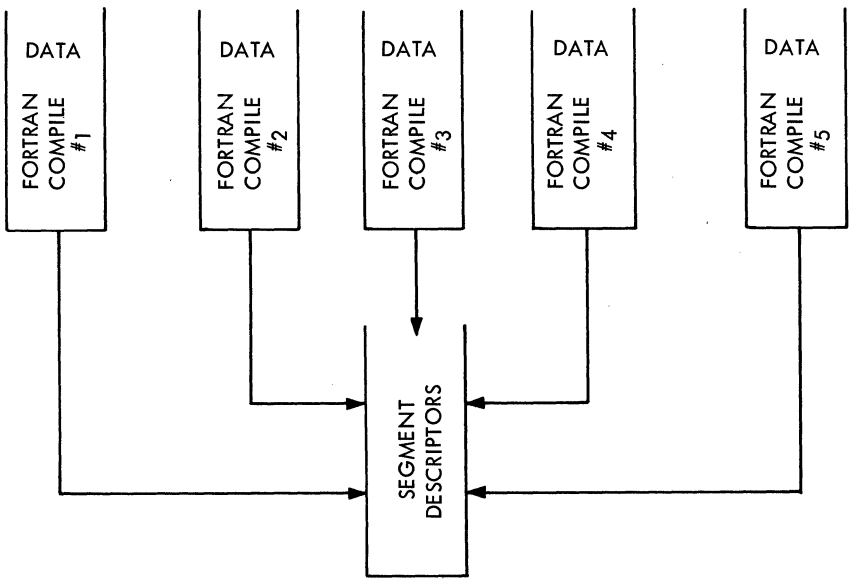


Figure 3-13. Cactus Stack Structure

c. Testing for an invalid operand function.

Mark Stack Operator

This operator inserts a mark into the stack for subsequent use by an Enter operator. The mark placed in the stack is in the form of a Mark Stack Control Word. The Mark Stack operator is normally used when an entry to a procedure is anticipated. The normal sequence of events to enter a procedure is as follows:

a. Mark the stack (with a Mark Stack Control Word).
b. Insert an indirect reference to a Program Control Word.
c. Insert some parameters if any are to be passed to the procedure.
d. Execute an Enter operator which will in turn cause entry into the program segment by the Program Control Word. A Return Control Word is stored in the stack.

Insert Mark Stack Operator

This operator inserts a Mark Stack Control Word below the two top of the stack items.

Name Call Operator

The Name Call operator (two syllables) builds an Indirect Reference Word and places it into the stack. Stack adjustment takes place so that the A register is empty. The six low-order bits of the first syllable of this operator are concatenated with the eight bits of the following syllable to form a 14-bit address couple. The address couple is placed, right-justified, into the A register, with the remainder of the A register set to ZERO. The control field of the A register is set to 001 and the register is marked as being full.

Value Call Operator

This operator (two syllables) loads into the top of the stack the operand referenced by the address couple contained in the operator syllable. If the referenced operand is at the end of a chain of Indirect Reference Words or descriptors (data or program), multiple memory accesses are made until the operand is located. The operand is then placed into the top of the stack registers (A, B, X or Y). The operand may be either single or double precision, causing either one or two words to be loaded into the stack. Presence bit action, in the form of a presence bit interrupt, is enabled whenever a non-present descriptor is accessed. In addition, subroutine entry is enabled if a Program Control Word is accessed without using the standard mechanism for subroutine entry, i.e., accidental entry. The standard mechanism consists of marking the stack, passing parameters, and then calling on the Program Control Word.

Evaluate Descriptor

The Evaluate Descriptor loads the top of the stack with a data descriptor which points to the referenced operand. This operand may be referenced either through a chain of Indirect Reference Words or descriptors. In either case, multiple memory accesses may be made until the operand is located. The top of the stack is then replaced with either the descriptor or the indirect Reference Word which points to the operand. A descriptor is left in the stack when the operand is referenced by a descriptor; an Indirect Reference Word is left in the stack when the operand is referenced by an Indirect Reference Word. The Evaluate Descriptor operator would be executed only after the top of the stack has been loaded with either a descriptor or Indirect Reference Word. An invalid operand interrupt is set if the top of the stack word is not a descriptor or Indirect Reference Word when an Evaluate Descriptor operator is to be executed.

Enter Operator

The Enter operator causes an entry to a procedure to occur. The sequence of events to enter a procedure are:

a. Mark the stack.
b. Insert an indirect reference to a Program Control Word.
c. Insert parameters if any are to be passed to the procedure.
d. Execute an Enter operator.

The Enter operator causes entry into the program segment located by the Program Control Word. A Return Control Word is stored in the stack.

Exit Operator

This operator causes a return from a called procedure to a calling procedure when a called procedure is not required to return a result.

Return Operator

This operator causes a return from a called procedure to a calling procedure when the called

procedure is required to return a result. The Return operator will leave either an operand or a descriptor in the stack depending upon whether the procedure was called by a Value Call operator or an Evaluate Descriptor operator. A single or double precision operand is left in the top of the stack if the procedure was entered by a Value Call operator. A descriptor (data or program) is left in the top of the stack if procedure entry was accomplished by an Evaluate Descriptor operator.

Branch Operators

Branch instructions function to break the normal sequence of serial instruction fetches. Branching may be either relative to the base address of the current program segment or to an absolute memory address. Branch operators may be executed on a conditional or unconditional basis. Conditional branch operators test the low-order bit of the second word in the stack to determine whether or not to branch. Unconditional Branch operators always cause a branch. The Branch operator repetoire consists of:

a. Branch Unconditional.
b. Branch on True.
c. Branch on False.
d. Dynamic Branch Unconditional.
e. Dynamic Branch True.
f. Dynamic Branch False.
g. No Operation.
h. Conditional Halt.
i. Invalid Operator.

Step And Branch Operator

The Step and Branch operator causes the addition of the increment field to the current value field of the Step Index Word addressed by the top of stack word. Then test the current value field for greater than the final value field. If greater, then execute the next two syllables from the program string. Otherwise, branch three program syllables and continue in sequence. Replace the Step Index Word in the stack or memory.

Arithmetic Operators

The Arithmetic operators require two operands in the top of the 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

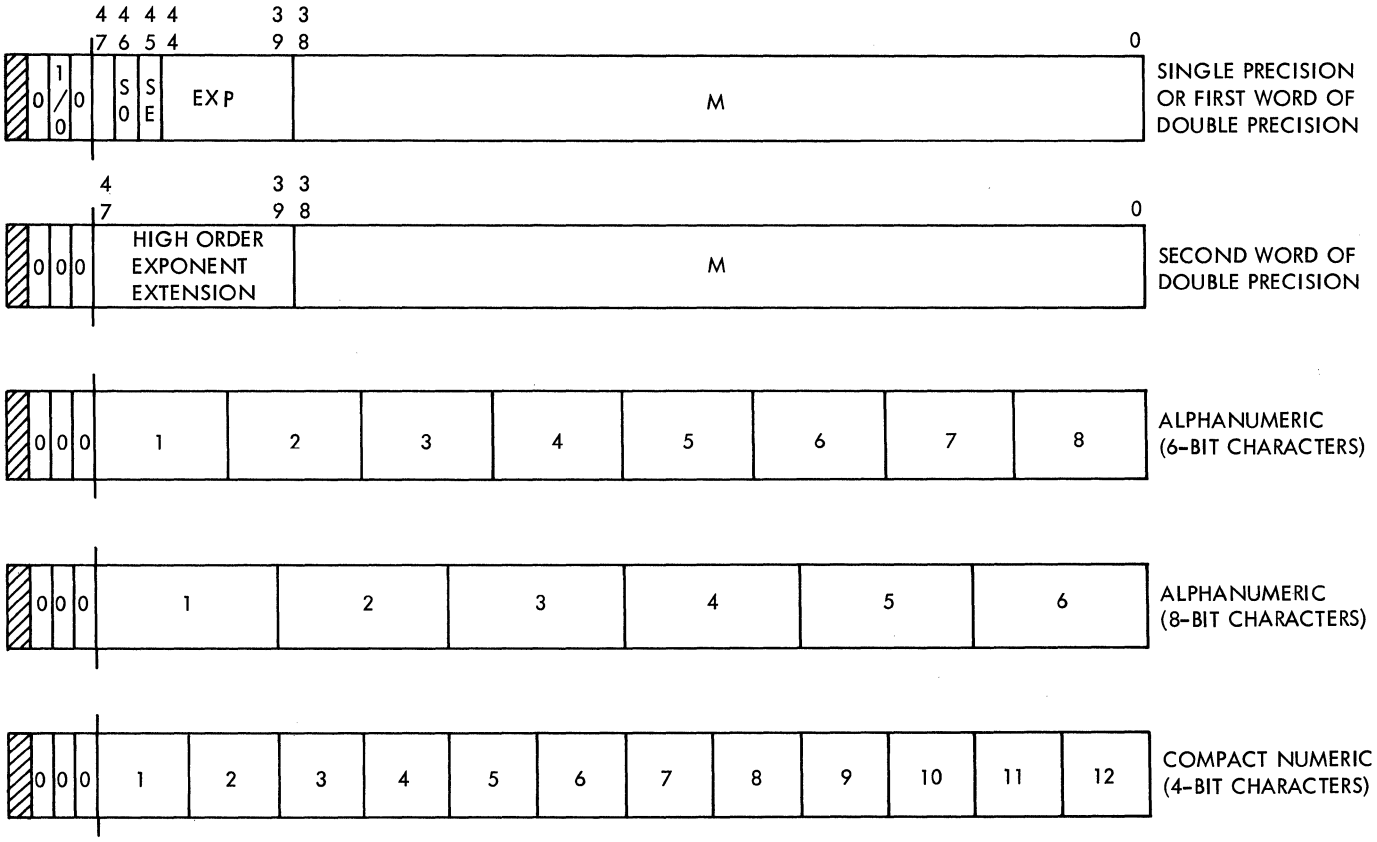


Figure 3-14. Operand Formats

process adapts automatically to the data environment, with single precision process invoked if both operands are of the single precision type and a double precision process invoked if either operand is of the double precision type.

The formats for an operand are illustrated in figure 3-14.

Each double precision operand occupies two words. The second word of the operand is an extension of the mantissa of the first word of an operand; i.e., the mantissa of the first word of an operand may be an integer but the mantissa of the second word is always a fraction. When in the stack, the first operand occupies the top two words of the stack, and the second operand occupies the third and fourth words of the stack. Therefore, double precision arithmetic processes operate on four words in the stack as opposed to two for single precision processes. Double precision arithmetic leaves a two-word result in the top of the stack.

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

During the execution of SP(DP) arithmetic operators, the exponents of the operands are carried to 7(16) bits. At the completion of the execution of an arithmetic operator, the exponent is reduced to 6(15) bits. If the exponent of the result requires more than 6(15) bits, the appropriate interrupt bit (exponent overflow or underflow) is set in the Processor Interrupt Register.

The following arithmetic operators are provided in the B 6500.

a. Add.
b. Subtract.
c. Multiply.
d. Divide.
e. Integer Divide.
f. Remainder Divide.
g. Integerize, Truncated.
h. Integerize, Rounded.
i. Integerize, Rounded, Double Precision.
j. Set to Single Precision, Truncated.
k. Set to Single Precision, Rounded.
l. Set to Double Precision.
m. Set Double to Two Singles.
n. Set Two Singles to Double.
o. Extended Multiply.

Logical Operators

Logical operators use the two top items in the stack except for logical negate which uses only the top stack item. The result of the logical operation leaves the result in the top of the stack. The Logical operators consist of:

a. Logical AND.
b. Logical OR.
c. Logical Equivalence.
d. Logical Negate.

Relational Operators

The Relational operators perform algebraic or logical comparisons on the top two values in the stack. The operands are removed from the stack and the result of the comparison is placed in the top of the stack. The result is a single precision operand with the least significant bit set to ONE if the relation is TRUE, or all information bits set to zero if the relation is FALSE. The Relational operators are:

a. Greater Than.
b. Greater Than or Equal.
c. Equal.
d. Less Than or Equal.
e. Less Than.
f. Not Equal.
g. Logical Equal.

Index and Load Operators

The Index and Load operators cause the system to generate the address of an indexed item, or generate an address and load the specified item into the top of the stack. They consist of the following:

a. Index.
b. Index and Load Name.
c. Index and Load Value.
d. Load.
e. Load Transparent.

Stack Operators

The function of the Stack operators is to manipulate items in the stack. These operators can exchange, rotate, duplicate, or delete items at the top of the stack. They include:

a. Exchange.
b. Rotate Stack Down.
c. Rotate Stack Up.
d. Duplicate Top of Stack.
e. Delete Top of Stack.
f. Push Down Stack Registers.

Store Operators

The Store operators cause information in the stack to be stored into the specified address in memory. The address is generated from information in the stack. A special Store operator allows the MCP to bypass the hardware memory protection checks. The Store operators consist of:

a. Store Destructive.
b. Store Non-Destructive
c. Overwrite Destructive.
d. Overwrite Non-Destructive.

Transfer Operators

The Transfer operators cause the system to transfer any field of bits from one word in the stack to any field of another word in the stack. These operators include:

a. Field Transfer.
b. Dynamic Field Transfer.
c. Field Isolate.
d. Dynamic Field Isolate.
e. Field Insert.
f. Dynamic Field Insert.
g. String Isolate.

Literal Call Operators

The Literal Call operators place the designated value into the top of the stack. The value can be any value within the range of the system up to a full 48-bit word. The Literal operators are:

a. Lit Call Zero.
b. Lit Call One.
c. Lit Call 8 Bits.
d. Lit Call 16 Bits.
e. Lit Call 48 Bits.

Bit Operators

The Bit operators are concerned with a specified bit in the top of the stack. A bit may be set or reset, or a word in the top of the stack can be examined to find either the first bit ON or the number of bits that are ON. The Bit operators are:

a. Bit Set.
b. Dynamic Bit Set.
c. Bit Reset.
d. Dynamic Bit Reset.
e. Change Sign Bit.
f. Count Binary Ones.
g. Leading One Test.

String Operators

The String operators are used to transfer, scan, compare, and translate strings of data. In addition, a set of micro-operators provides a means of formatting data for output.

STRING TRANSFER OPERATOR.

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. The String Transfer operators include:

a. Transfer Words.
b. Transfer Words, Update.
c. Transfer Unconditional, Destructive.
d. Transfer Unconditional, Update.
e. Transfer Words Overwrite Destructive.
f. Transfer Words Overwrite, Update.

SCAN WHILE OPERATORS.

The Scan While operators cause a scan of a string of characters, which continues until the specified scan is satisfied. The top words of the stack specify the field length, the delimiter character, and the source pointer. The TRUE/FALSE flip-flop will indicate the result of the scan operation. The following comprise the Scan While operator repetoire:

a. Scan While Greater.
b. Scan While Greater, Update.
c. Scan While Greater or Equal.
d. Scan While Greater or Equal, Update.
e. Scan While Equal.
f. Scan While Equal, Update.
g. Scan While Less or Equal.

h. Scan While Less or Equal, Update.
i. Scan While Less.
j. Scan While Less, Update.
k. Scan While Not Equal.
l. Scan While Not Equal, Update.
m. Scan While True.
n. Scan While True, Update.
o. Scan While False.
p. Scan While False, Update.

COMPARE OPERATORS.

The Compare operators perform the specified compare of two strings of data. The top three items in the stack specify the number of characters to compare and the locations of the two data areas. The TRUE/FALSE flip-flop will indicate the result of the compare. The Compare operators are:

a. Compare Characters Greater.
b. Compare Characters Greater, Update.
c. Compare Characters Greater or Equal.
d. Compare Characters Greater or Equal, Update.
e. Compare Characters Equal.
f. Compare Characters Equal, Update.
g. Compare Characters Less or Equal.
h. Compare Characters Less or Equal, Update.
i. Compare Characters Less.
j. Compare Characters Less, Update.
k. Compare Characters Not Equal.
l. Compare Characters Not Equal, Update.

TRANSLATE OPERATOR.

The Translate operator causes data to be moved and translated from a source string to a destination area. The top four items in the stack specify the number of characters translated and the locations of the translation table, source string, and destination areas.

PACK OPERATORS.

The Pack operators cause BCL or EBCDIC numeric characters to be packed, in the top of the stack, as 4-bit numeric information. The top two items in the stack contain the number of digits to pack and the location of the source data. The pack operators include:

a. Pack, Destructive.
b. Pack, Update.

UNPACK OPERATORS.

The Unpack operators cause packed numeric information (4-bit digits) to be moved to a destination area as Burroughs Common Language (BCL) or EBCDIC numeric characters. The top three items in the stack contain the number to unpack, the packed data word, and the location of the destination area. The Unpack operators include:

a. Unpack Absolute.
b. Unpack Absolute, Update.
c. Unpack Signed.
d. Unpack Signed, Update.

INPUT CONVERT OPERATORS

The Input Convert operators cause the specified number of characters in a source string to be converted into a single or double precision integer. The two top of the stack items specify the number of characters to convert and the location of the source string. The Input Convert operators are:

a. Input Convert.
b. Input Convert, Update.

EDIT OPERATORS.

The Edit operators cause data to be transferred from a source string to a destination area under control of a string of Edit micro-operators. The Edit micro-operators may be in the program string or in a separate table. The top of the stack contains the locations of the source string, destination area, and edit table, as required. The Enter operator set consists of:

a. Table Enter Edit.
b. Table Enter Edit, Update.
c. Execute Single Micro, Single Pointer Update.
d. Execute Single Micro.
e. Execute Single Micro, Update.

MOVE MICRO-OPERATORS.

The Move Micro-operators cause data to be transferred as a result of one of the Edit operators discussed above. These Move operators are:

a. Move Characters.
b. Move Numeric Unconditional

SKIP MICRO-OPERATORS.

The Skip Micro-operators are used to alter the location pointers of a source or destination area, either forward or reverse, by a specified number of characters. The Skip operators consist of:

a. Skip Forward Source Characters.
b. Skip Reverse Source Characters.
c. Skip Forward Destination Characters.
d. Skip Reverse Destination Characters.

INSERT MICRO-OPERATORS.

The Insert Micro-operators cause the specified character to be inserted into the destination string as many times as required. These operators are:

a. Insert Unconditional.
b. Insert Conditional.
c. Insert Display Sign.
d. Insert Overpunch.
e. End Insert.
f. End Float.
g. Move With Insert.
h. Move With Float.
i. End Edit.

SCALE OPERATORS.

Higher-level languages, such as COBOL, require integer arithmetic. The Scale operators provide the means of aligning decimal points prior to performing the arithmetic operations. In addition, the Scale Right operators provide the capability for converting from binary to decimal. The Scale operators are:

a. Scale Left.
b. Dynamic Scale Left.
c. Scale Right Save.
d. Dynamic Scale Right Save.
e. Scale Right Truncate.
f. Dynamic Scale Right Truncate.
g. Scale Right Round.
h. Dynamic Scale Right Round.
i. Scale Right Final.
j. Dynamic Scale Right Final.

OCCURS INDEX OPERATOR.

The Occurs Index operator uses the two top words in the stack to create an index into a table. The index is checked to ensure that it is valid for the table size.

SPECIAL FLIP-FLOP OPERATORS.

The function of these operators is to generate a value in the top of the stack as a result of the specific flip-flop setting, or to change the setting of the flip-flop. These special flip-flop operators are:

a. Set External Sign.
b. Read TRUE/FALSE Flip-Flop.
c. Read and Clear Overflow Flip-Flop.
d. Reset Float.

Software Operators

Additional operators have been included to perform specific functions required by the MCP. This set includes:

a. Masked Search for Equal.
b. Linked List Lookup.
c. Set Tag Field.
d. Read Tag Field.
e. Set Interval Timer.
f. Read With Lock.
g. Move To Stack.
h. Stuff Environment.
i. Read Processor Identification.
j. Enable External Interrupt.
k. Idle Until Interrupt.
l. Interrupt Other Processors.
m. Read Processor Register.
n. Set Processor Register.
o. Scan In.
    1) Read Time of Day Clock.
    2) Read General Control Adapter.
    3) Read Result Descriptor.
    4) Read Interrupt Mask.
    5) Read Interrupt Register.
    6) Read Interrupt Literal.
    7) Interrogate Peripheral Status.
    8) Interrogate Peripheral Unit Type.
    9) Interrogate I/O Path.
p. Scan Out.
    1) Set Time of Day Clock.
    2) Set General Control Adapter.
    3) Set Interrupt Mask.
    4) Initiate I/O.
q. Make Program Control Word.

SECTION 4

INPUT/OUTPUT MULTIPLEXOR AND PERIPHERAL CONTROLS

GENERAL

The internal processing speed of the B 6500 is complemented by equally powerful input/output (I/O) hardware to achieve a well-balanced computing system. Transfer of all data between memory and all peripheral devices is controlled by the I/O multiplexor, independent of the processor. One or two of these multiplexors may be attached to a B 6500, each one capable of processing up to ten I/O operations simultaneously, from any of 256 peripheral devices.

OPERATION

A peripheral control (P.C.) buss extends from the multiplexor to the various peripheral devices. Attached along this buss are from one to 20 peripheral controls (figure 4-1). Information in one- or two-byte groups can be sent along the buss to or from any peripheral control every 1.2 microseconds.

Either processor can initiate an I/O operation on either multiplexor, in a two processor/two multiplexor configuration, by executing an Initiate I/O command. This command transfers a Unit Number Word and an Area Descriptor to the multiplexor via the scan buss. The multiplexor then fetches the I/O Control Word located at the Area Base Address of the Area Descriptor and initiates the peripheral operation. Upon its completion, the I/O Finish Interrupt is set. The Result Descriptor is returned when either processor executes a Read Result Descriptor command.

Following is an example of what occurs upon initiation of a card read command. Each peripheral control contains a buffer – a one-character buffer for some devices and a two-character buffer for other devices. Once the physical read operation has been initiated and the peripheral control buffer filled, the peripheral control signals the multiplexor that it is ready for service. For an 800 card per minute card reader, this occurs approximately

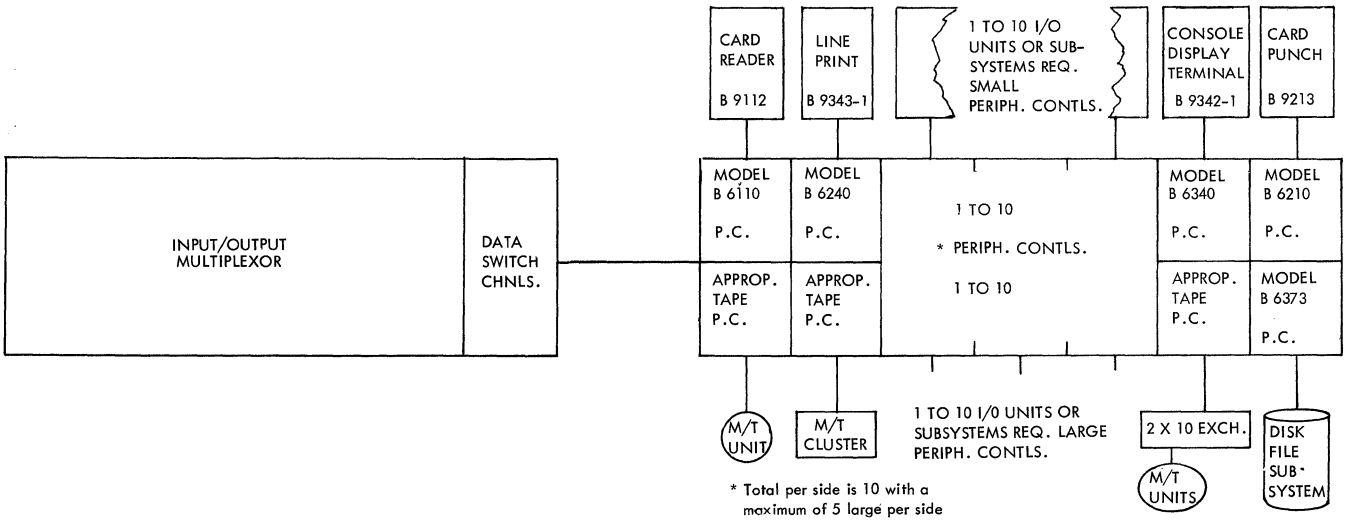


Figure 4-1. Input/Output Subsystem

once every millisecond. The multiplexor recognizes the request for service from the peripheral control and selects the proper data switching channel to receive the information. The data switching channel then properly positions the characters within a word. A character transmission is completed in 1.2 microseconds. When the final character of the word has been transmitted, the multiplexor places the entire word on the memory buss from which it is picked up by the proper module. This sequence is repeated for each I/O device requesting service. The number of devices which can be simultaneously serviced in this manner is limited only by the number of data switching channels within the multiplexor, which can range from four to ten per multiplexor.

DESCRIPTOR FORMATS

The formats of the Unit Number Word, Area Descriptor, and I/O Control Word, respectively, are illustrated in figure 4-2.

Unit Number Word

When M of the unit number word equals 0, all active multiplexors respond to the descriptor. When M equals 1, the multiplexor specified by the Z field responds to the command. (The 4-bit Z field designates a specific multiplexor.) When Z equals 0001 and M is 1, multiplexor A is selected. When Z equals 0010 and M is 1, multiplexor B is selected. Only the low-ordered bit that is on selects a multiplexor.

Area Descriptor

The area base address specifies the base address of the memory area. Buffer length indicates the size of the area defined.

I/O Control Word

The I/O Control Word is divided into a standard control field and a unit control field. Bits 35 – 0, the unit control field, are unique for each peripheral control. Bits 47 – 36, the standard control field, are defined as follows:

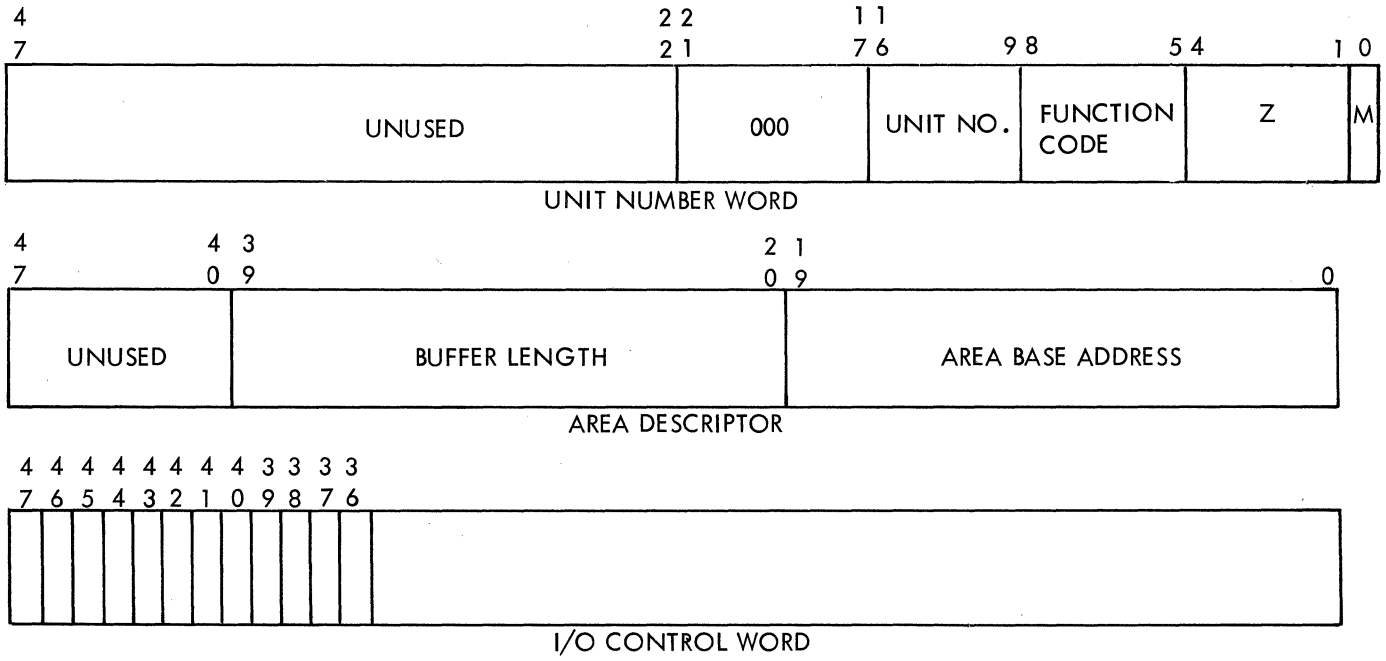


Figure 4-2. I/O Descriptor Formats

Bit	Assignment	Bit = 0	Bit = 1
47	Reserved		
46	Reserved		
45	Attention	No	Yes
44	Memory Read/Write	Read	Write
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 = 0
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

Result Descriptor

The format of the Result Descriptor is shown in figure 4-3.

Bits 47 – 28 indicate the final memory address at which the I/O operation terminated. Bits 16 – 0, the error field, is subdivided into a standard error field and a unit error field. The unit error field bit assignments, bits 16 – 7, are unique for each peripheral control. The standard error field bit assignments, bits 6 – 0, are as follows:

Bit	Assignment
6	Memory Parity Error
5	Memory Address Error
4	Descriptor Error
3	Not Ready
2	Busy
1	Software Attention
0	Software Attention/ Hardware Exception

PERIPHERAL CONTROLS

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

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

Magnetic Tape Control

A magnetic tape subsystem can include from one to four tape controls and 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. The available controls are:

a. B 6381-1 – used with B 9381-2, -3, -4 36 KB cluster units.
b. B 6381-2 – used with B 9382-2, -3, -4 72 KB cluster units.
c. B 6381-3 – used with B 9380-2, -3, -4 cluster units.
d. B 6391-3 – used with B 9391 72 KC free-standing tape unit.
e. B 6391-4 – used with B 9394-1 96 KC free-standing tape unit.
f. B 6393-1 – used with B 9392 72 KB free-standing tape unit.
g. B 6393-2 – used with B 9393 144 KB free-standing tape unit.
h. B 6393-3 – used with B 9394-2 96 KB free-standing tape unit.

MAGNETIC TAPE EXCHANGES.

A magnetic tape exchange is required when either more than one control or more than six magnetic

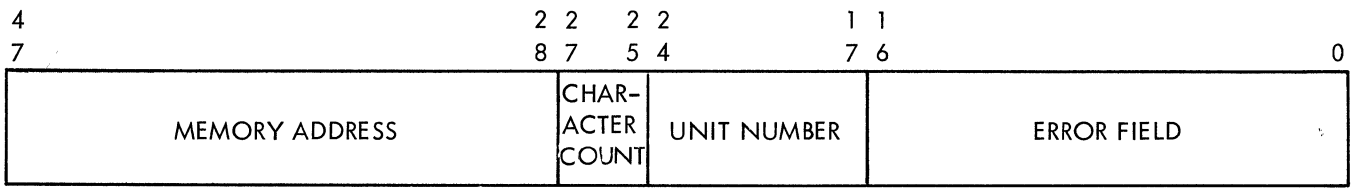


Figure 4-3. Result Descriptor Format

tape units are used. The available tape exchanges are:

a. B 6480 – 2 x 8 exchange used with B 9380-2, -3, -4.
b. B 6481 – 2 x 8 exchange used with B 9381-2, -3, -4, B 9382-2, -3, -4.
c. B 6490 – 2 x 10 exchange used with B 9391, B 9392, B 9394-1, -2.
d. B 6491 – 2 x 8 exchange used with B 9393.
e. B 6492 – 4 x 16 exchange used with B 9391, B 9392, B 9394-2.

Card Reader Control

The B 6110 Card Reader Control can be used with either the B 9111 or B 9112 Card Readers. The card reader reads BCL, binary, EBCDIC, ICT, or BULL card codes. These card codes are converted to BCL or EBCDIC internal code by translators in the multiplexor or card reader control. On a binary read, no code translation is effected.

Card Punch Control

The B 6210 Card Punch Control is used with the B 9213 Card Punch, which can punch either BCL, binary, EBCDIC, ICT, or BULL card codes. BCL internal code can be punched out as BCL, ICT, or BULL card code through the use of optional translators in the card punch control. The card punch control includes an EBCDIC internal to EBCDIC card code translator as a standard feature. If EBCDIC internal is required to be punched as BCL card code, or as one of the others, the translation is done between internal codes in the multiplexor. When punching binary, the contents of each column are divided into two 6-bit characters. The upper six bits are punched from the first 6-bit character received, and the lower six bits from the next 6-bit character.

Line Printer Control

The B 6240 Line Printer Control is used with the B 9245-2 and B 9245-3, or the B 9242-1 and B 9243-1 when using B 9943 Printer Memory. The B 6242 Line Printer Control is used with the B 9242-1 and B 9243-1 Line Printers without printer memory. EBCDIC 8-bit internal code or BCL 6-bit internal code is transmitted as BCL code by translators in the multiplexor. Print drums with either EBCDIC or BCL graphics can be used.

Paper Tape Reader Control

The B 6120 Paper Tape Reader Control is used with the B 9120 Paper Tape Reader, which accommodates either BCL or binary tape. BCL tape is converted to BCL internal code or EBCDIC by translators in the multiplexor. No conversion occurs when reading binary tape.

Paper Tape Punch Control

The B 6220 Paper Tape Punch Control is used with the B 9220 Paper Tape Punch, which accommodates either BCL or binary tape. BCL internal code or EBCDIC is transmitted to the punch control as BCL internal code by the translators in the multiplexor. BCL internal code is then converted to BCL paper tape code by a translator in the punch control.

Disk File Control

The B 6373 Disk File Control is used with several different models of Burroughs head-per-track disk file subsystem. The various models available are discussed in Section 6, Peripheral Components.

DISK FILE EXCHANGE.

From one to four disk file controls can be used to form a multiple control subsystem, more than one of which can be attached to the B 6500. Disk subsystems may be attached to a maximum of 18 of the 20 large peripheral controls. One large control must be reserved for a magnetic tape unit and one for the console display, as required by the MCP. A B 6471 Basic Disk File Exchange N_1 x N_2 (up to 4 x 20 with appropriate adapters) is required if it is desirable to attach more than one electronics unit to a peripheral control. For example, with a 2 x 10 exchange, ten electronics units can be attached to the system through the use of only two peripheral controls. Each electronics unit can handle up to five modules of four disks each.

Console Display Control

The B 6340 Console Display Control permits B 9342-1 Console Display and up to 7 more B 9342-1 Console Displays or B 9342-2 optional Printer/Keyboards to be connected to a B 6500 System.

SECTION 5

DATA COMMUNICATIONS PROCESSOR

GENERAL

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

The B 6350 Data Communications Processor (DCP) is a small special purpose computer which contains sufficient registers, logic, and translation ability to perform all basic functions associated with sending and receiving data. Up to four DCP's can be connected to an I/O Multiplexor, with each DCP capable of accommodating from one to 256 communications lines (figure 5-1). On a two Multiplexor system, this provides a B 6500 with the ability to service 2048 data communications lines.

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

a. B 6650-1 with the following characteristics:
   1) Direct or modem connect.
   2) Asynchronous.
   3) Up to 600 BPS.
   4) Two wire or 100 series type modem using RS 232 defined interface.
   5) Serial by bit transmission.
   6) Half-duplex mode.

b. B 6650-2 with the following characteristics:
   1) Direct or modem connect.
   2) Asynchronous.
   3) Up to 1800 BPS.
   4) Two wire or 202 series type modem using RS 232 defined interface.
   5) Serial by bit transmission.
   6) Half-duplex mode.

c. B 6650-3 with the following characteristics:
   1) Modem connect.
   2) Synchronous.
   3) Up to 2400 BPS.
   4) 201 series type modem using RS 232 defined interface.

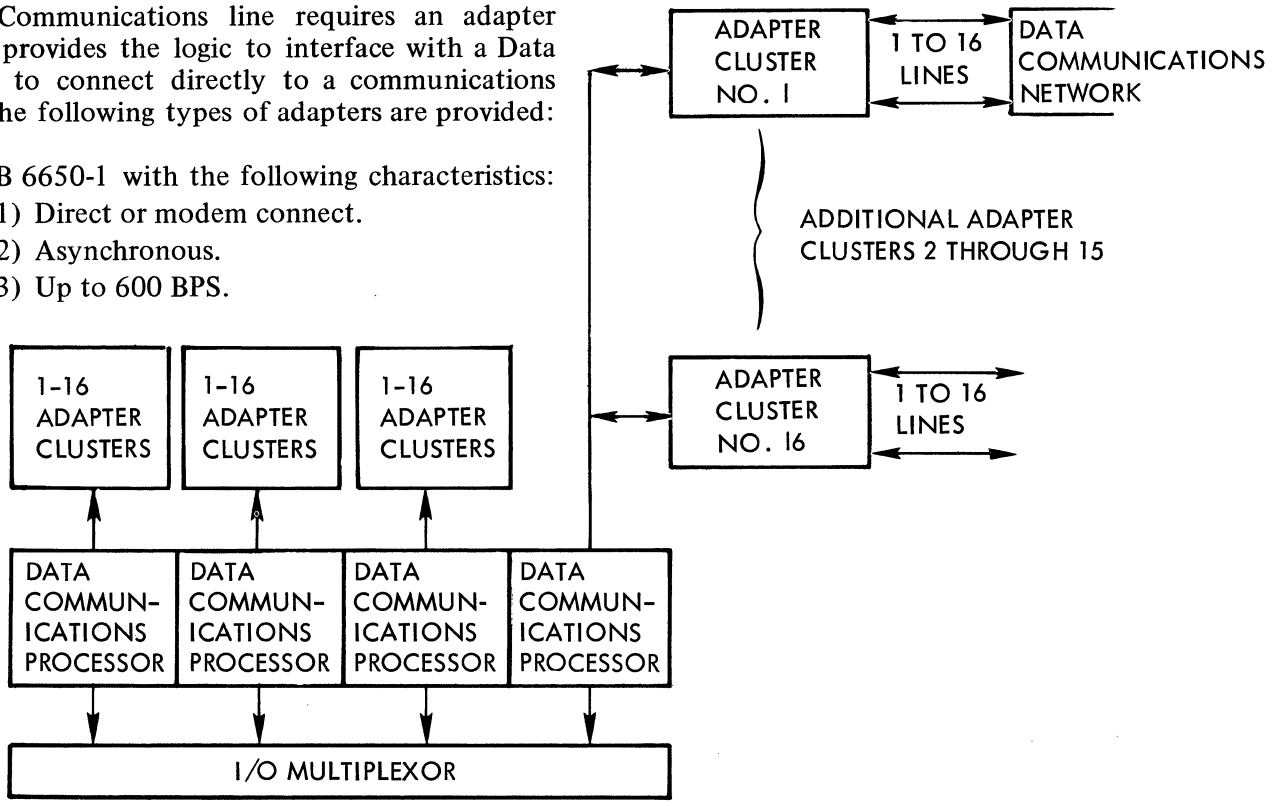


Figure 5-1. Organization of Data Communications Processor Remote Lines

5) Serial by bit transmission.
6) Half-duplex mode.

d. B 6650-4 with the same characteristics as the -3 except the speed is up to 4800 BPS.
e. B 6650-5 with the same characteristics as the -3 except the speed is up to 9600 BPS.
f. B 6650-6 Touch-Tone* Telephone Input.
g. B 6650-7 Audio Response.
h. B 6650-8 Automatic Dial Out.

*Registered Trade Mark of AT & T Co.

These adapters provide the facilities for attaching the various remote terminals listed in Table 5-1 to a B 6500.

OPERATION

When a data communications command is initiated by the processor, the I/O descriptor is sent to the multiplexor which directs it to the proper data communications processor (figure 5-2).

The DCP contains a set of arithmetic registers, 8 words of scratchpad memory and may or may not contain 1024 52-bit words of local DCP memory. Based on the descriptor received, the DCP reads the control codes and the data associated with that message from the main memory one word at a time. It then attaches its character buffer, through the proper Adapter Cluster, and sends the character it has fetched out to the proper line adapter. This procedure is continued until the End-of-Message code has been received.

On incoming messages, the line adapter signals the data communications processor through the Adapter Cluster, and the above procedure is reversed. The DCP generates an input request interrupt and the MCP data handler routine directs the incoming characters to a location in memory set aside for that particular line adapter.

A maximum of 256 lines can be tied to a single DCP. The DCP can readily handle a situation in which all lines have 10 character per second Teletypes. (Data would be received at the rate of

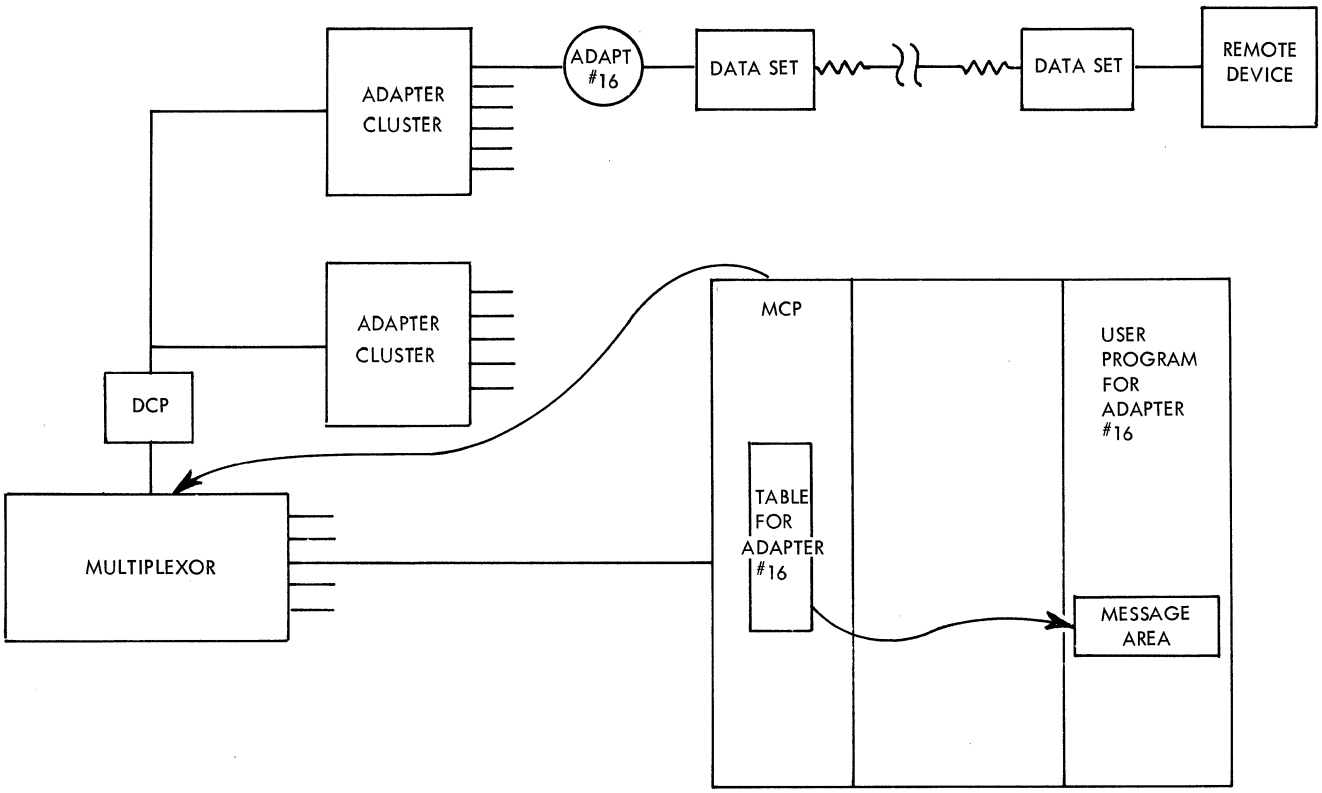


Figure 5-2. Typical Data Communications Message Flow

TABLE 5-1

B 6500 Remote Terminal Characteristics

Terminal	Leased	Switched	Direct Connect	Asynchronous	Synchronous	Modem Type	Speed Range
TWX Service		X		X		103,811B	Up to 150 BPS.
Western Electric Model 33	X	X	X	X		103	Up to 110 BPS.
Western Electric Model 35[1]	X	X	X	X		103,816A	Up to 110 BPS.
Western Electric Model 37	X	X	X	X		103	Up to 165 BPS.
Touch Tone/Audio Response	X	X		Parallel		403	Up to 10 CPS.
B 9351 Series Input and Display	X	X	X	X	X	103,202, or 201	Up to 2400 BPS.
B 9352 Series Input and Display	X	X	X	X	X	103,202, or 201	Up to 2400 BPS.
Model 28/83B3 (or equivalent Western Union Service)[2]	X		X	X			Up to 110 BPS.
TC 500 Terminal	X	X	X	X		202	Up to 1200 BPS.
B 300/B 500 Systems	X	X	X		X	201	Up to 2400 BPS.
B 2500/B 3500 Systems	X	X	X		X	201	Up to 2400 BPS.
B 5500 Systems	X	X	X		X	201	Up to 2400 BPS.
Honeywell 120 System	X				X	201	Up to 2400 BPS.
IBM 1030	X		X	X		202	Up to 14.8 CPS.
Automotive Calling Unit		X				801	

1. Also 8A1 Selective Calling Service or Western Electric equivalent.
2. Speed for 83B3 service is 74.6 bps.

one character per 400 microseconds.) The variety of conditions under which data communications can be used makes it difficult to determine the peak load under typical conditions. Each DCP operates independently of all others; up to eight DCPs, four to a multiplexor, can be attached to a B 6500.

Each Adapter Cluster has two input/output connectors, each of which can connect to a data communications processor, thereby permitting the Adapter Cluster to be handled by two DCPs. See figure 1-1 for a typical configuration. This type of configuration provides a backup facility in the case of failure of one DCP.

Because the DCP can be programed and can execute code stored in main memory or local DCP memory, it can perform many of the functions previously wired into the line adapter. The DCP can handle the line discipline for a given data set and perform the various control code and line delimiter functions for the specific device programmatically. For example, a remote location may have two different types of computers, each requiring the same model data set, but each with a different set of control codes. With the DCP, both computers can be serviced over the same line with the same line adapter.

PROGRAM REQUIREMENTS

Scan Function

The Scan Function, which scans all lines for activity, can be stored in the Data Communications Processor or stored in main memory.

Service Program

When the Scan Function finds a line which requires attention, either a program stored in the DCP or a program stored in main memory is used to service that line. Typically, this requires a character to be stored in, or read from main memory. The main memory address is stored in DCP Scratch Pad memory. The character being serviced is checked to determine if any special action is required; i.e., delimiter detection or parity check.

Polling Program

Some terminal types such as the B 9352 series input and display, may require a Polling Program which is either a program stored in instruction memory or a program stored in main memory. If a program stored in instruction memory is used, the general registers can be used for temporary storage, thus allowing the same program to service many channels. Polling by this means does not require main memory cycles.

SECTION 6

PERIPHERAL COMPONENTS

GENERAL

Peripheral components are units that provide input and output facilities. For the B 6500 System they operate independent of the processor, but always under control of the MCP through the multiplexor and associated peripheral control. Up to a maximum of 256 I/O devices may be attached to a 2 multiplexor system, where each magnetic tape unit or station on a tape quad and each electronics unit in a disk file subsystem is considered a device. These 256 devices are totally independent of the up to 1024 remote lines which can also be on-line through the Data Communications Processors. This section provides a general discussion of the peripheral components that may be used with B 6500 Systems.

DESK CONSOLE

The desk console is the operation center of the B 6500 System providing the switches and indicator lights for operator control. Included in the console control center is the console display terminal, the communications link between the system and operator, by means of which instructions and certain information is displayed to the operator. Up to 960 characters (40 data lines of 24 characters each) can be displayed at one time. Through this unit the operator may signal a reply or request information from the MCP concerning programs in process or operating conditions. If provision for hard copy of display information is desired, an optional printer/keyboard can be added. As an alternative to using the slow typewriter, hard copy can also be obtained on the line printer.

Because of the comprehensiveness of the console display terminal operation, and because the speed of printer/keyboard is not adequate to keep up with the system operation, hard copy can also be obtained through a line printer as required.

DISK FILES

The B 6500 Disk File Subsystem (figure 6-1) is an extremely high-speed, modular, random information storage system. A basic system consists of one electronics unit and from one to five storage modules, each storage module having a capacity of

Figure 6-1. Disk File Electronics Unit with one Disk File Module attached

18 or 20 million bytes to provide a maximum of 100 million bytes of storage on a single electronics unit. Other models are available (Data Memory Banks, figure 6-2) to allow B 6500 disk file storage up to 36 billion bytes.

Disk modules are available with average access times ranging from 20 to 60 milliseconds. Each storage module contains four disks, each with two recording surfaces, divided into a number of circular tracks. Each track has an individual, fixed read/write head, operated by electronic switching. Each track is divided into segments of 180 bytes. A sequential 7-digit decimal address scheme provides ease in record addressing, file organization, and file maintenance. Each disk in the module is provided

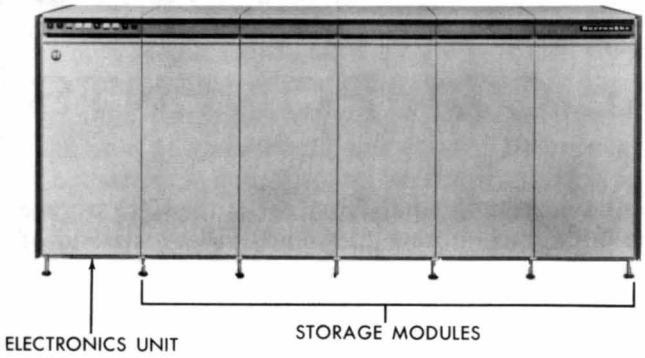


Figure 6-2. Data Memory Bank

with an individual lockout switch to prevent writing; however, a locked out disk may still be read. Automatic checking is provided to assure valid data recording.

Table 6-1 provides a list of the various disk files available with the B 6500 System.

MAGNETIC TAPE

The B 6500 affords considerable flexibility in the area of magnetic tape handling devices, with the number of units available on a given system limited only by the number of exchanges and peripheral controls employed. The user may choose either 7-channel BCL tape or 9-channel EBCDIC tape, which may be intermixed, provided this is not attempted on the same subsystem. The user also may select any of four packing densities up to 1600 bits per inch (BPI) and transfer rates from 9,000 to 144,000 characters per second.

A choice of physical construction is also furnished. The user may choose a free-standing device (figure 6-3) which houses one magnetic tape unit per cabinet or the cluster device (figure 604) which provides up to four tape-handling devices per cabinet.

The magnetic tape units are capable of reading and spacing in either a forward or reverse direction.

Table 6-2 documents the packing densities and transfer rates available with either type of tape unit.

TABLE 6-1

Disk File Storage

Electronics Unit Model	Storage Unit Model	Disk File Module Description
B 9371-7	B 9372-11	10.87 million bytes, 20 milliseconds average access time, five modules per electronics unit
	B 9375-10	Data Memory Bank – 100 million bytes, 23 milliseconds average access time
B 9371-8	B 9376-10	Additional 20 million byte increments
	B 9375-12	Data Memory Bank – 100 million bytes, 40 milliseconds average access time.
B 9371-9	B 9376-12	Additional 20 million byte increments.
	B 9375-13	Data Memory Bank – 100 million bytes, 60 milliseconds average access time
B 9371-10	B 9376-13	Additional 20 million byte increments

* Data Memory Banks include electronics units; however, optional additional units may be ordered using stated model numbers.

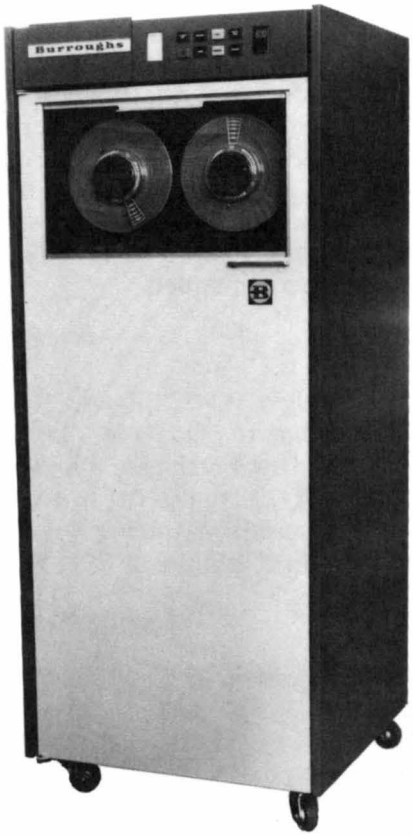


Figure 6-3. Free-Standing Magnetic Tape Unit

Figure 6-4. Clustered Tape Unit

CARD READERS

Card readers (figure 6-5) available for use with a B 6500 System can read 51-, 60-, 66-, or

TABLE 6-2

Magnetic Tape Unit Characteristics

Model	Channels	Speed (IPS)	Density (BPI)	Transfer rate
Clustered				
B 9380-2, -3, -4	7	45	800/556/200	36/25/9 KC
B 9381-2, -3, -4	9	45	800/200 *	36 KB
B 9382-2, -3, -4	9	45	1600	72 KB
Free-Standing				
B 9391	7	90	800/556/200	72/50/18 KC
B 9392	9	90	800/200 *	72/18 KB
B 9393	9	90	1600	144 KB
B 9394-1	7	120	800/556/200	96/66/24 KC
B 9394-2	9	120	800/200 *	96/24 KB

* These Tape Units require their peripheral controls to have an optional adapter to permit 200 BPI operation.

80-column punched cards at speeds of either 800 (B 9111) or 1400 (B 9112) cards per minute. The card readers use an immediate access clutch to provide optimum operation. Card columns are read serially by photo-electric sensing, with the read circuitry automatically monitored between card cycles. Invalid character detection is provided during card reading. Cards may be added or removed while the reader is operating. Both the input hopper and the output stacker have a capacity of 2400 cards. Optional features include the ability to read 40-column Treasury Checks and round holes in Postal Money Orders. Cards with any corner cut, those that have been verified (notched on the right), and cards of varying thickness are acceptable; the latter, however, must be consistent during any one run. The number of card readers on a given system is limited only by the number of available peripheral controls.

Figure 6-5. Card Reader

LINE PRINTERS

Three line printers (figure 6-6) are available for use with B 6500 Systems. The B 9242-1 contains the full BCL character (64) set, prints at a speed of 860 lines per minute, and provides 120 print positions. The B 9243-1 prints at a speed of 1100 lines per minute, provides 120 print positions, and has a 44 character set. The B 9245-2 and B 9245-3 print at a speed of 315 lines per minute. The -2 printer contains 120 print positions, whereas the -3 printer contains 132 positions. Each has vertical skipping and end-of-page formatting controlled by a punched paper tape. The normal paper slew rate is 25 inches per second, with an optional high-speed slew of 75 inches per second for skips of more than 1.17 inches. With the high-speed slew option selected, use of the B 9949 Power Forms Stacker option is recommended.

Figure 6-6. Line Printer

CARD PUNCH

The B 9213 Card Punch (figure 6-7), used with the B 6500 System, punches at a maximum rate of 300 cards per minute. While waiting for a Punch command, cards can remain in the punch station indefinitely without being damaged. Pre-punched cards may be used, but previously punched columns cannot be repunched. Punching is verified by an echo check using solid state circuitry. An immediate access clutch provides demand feeding. Cards may be loaded or unloaded while the unit is in operation. The card hopper capacity is 1000 cards; the three card stackers (primary, auxiliary, error) have a capacity of 1200 cards each. Stacker selection is accomplished programmatically.

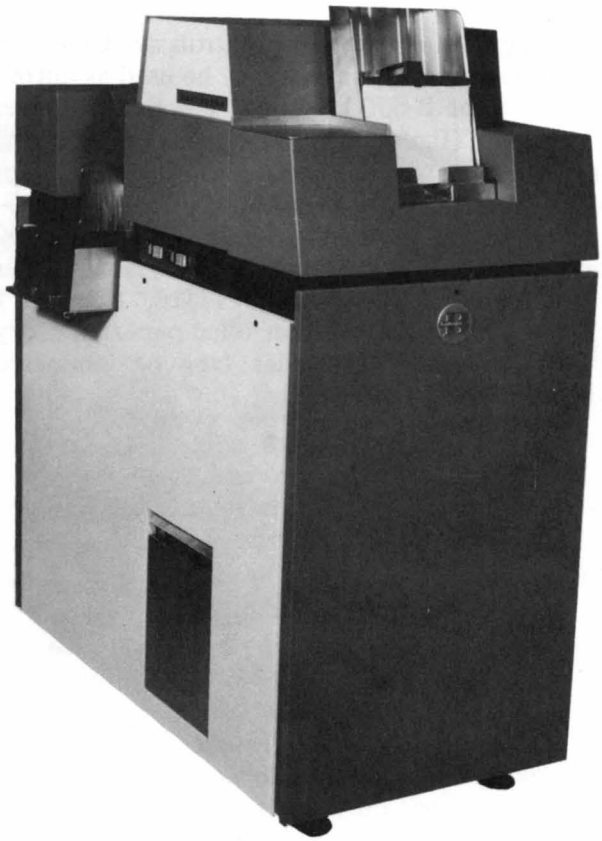


Figure 6-7. Card Punch

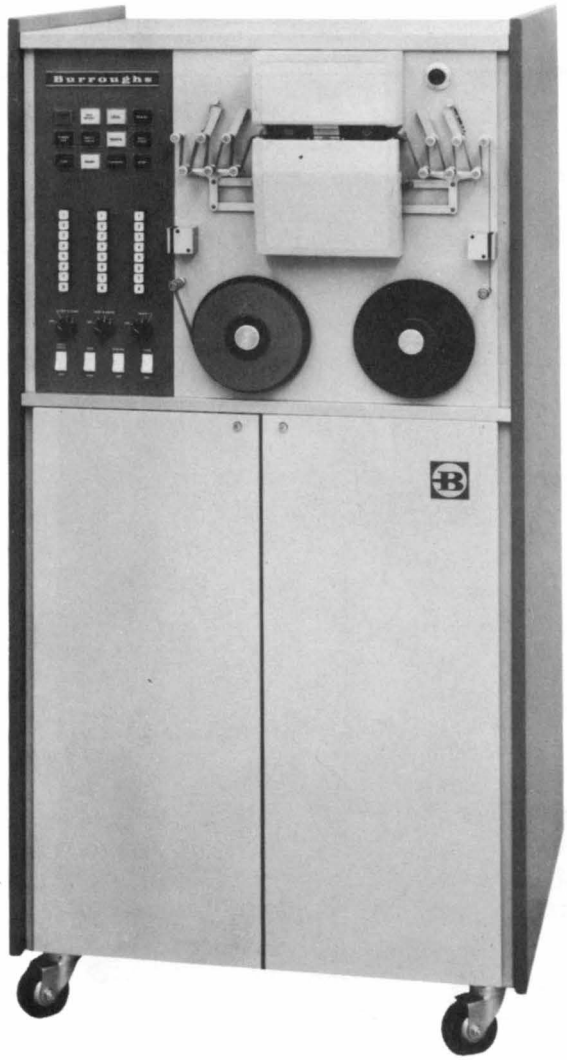


Figure 6-8. Paper Tape Reader

PAPER TAPE READER

The B 9120 Paper Tape Reader (figure 6-8), used with the B 6500 System, is capable of reading punched paper tape at a maximum rate of 1000 characters per second and metalized Mylar or fanfold tape at a maximum rate of 500 characters per second. Baudot and BCL to EBCDIC code translation is automatic; all other codes are read directly into memory and must be translated programmatically. The reader can accommodate 5-, 6-, 7-, or 8 channel tape as selected by the console operator.

Tape widths of 11/16, 7/8, or 1 inch are interchangeable. The number of paper tape readers is controlled by the number of peripheral controls available.

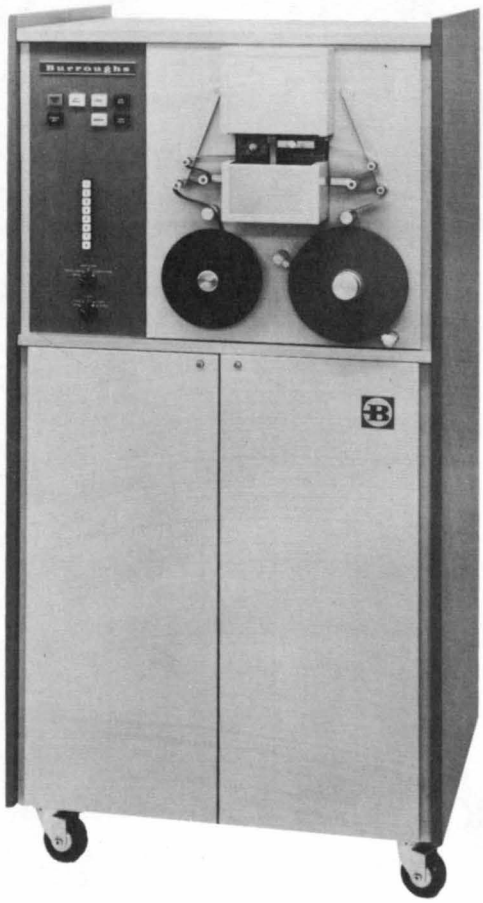


Figure 6-9. Paper Tape Punch

PAPER TAPE PUNCH

Depending on the peripheral controls available, one or more paper tape punches may be used as output devices for the B 6500 System. The B 9220 Paper Tape Punch (figure 6-9) is capable of punching standard paper tape format in either BCL or Baudot code. The punch will accommodate 5-, 6-, 7- or 8 channel tape at a maximum rate of 100 characters per second, punching ten characters per inch. Standard tape widths of 11/16, 7/8, and 1 inch may be punched. Either oiled paper tape, dry paper tape, metalized Mylar tape or laminated Mylar tape may be used.

SECTION 7

SOFTWARE

GENERAL

The B 6500 offers a wide range of software systems. Overseeing system operation is the Master Control Program (MCP) which consists of a special set of routines to provide control over scheduling, dynamic storage allocation of memory, handling of all input/output operations, standard error handling procedures, multiprograming and parallel processing, and the handling of all system interrupts. At the user level, three problem-oriented languages are available: COBOL for the solution of business oriented applications and ALGOL and FORTRAN which are employed primarily in the solution of scientific problems. In addition, the systems afford a variety of application programs covering the areas of: scientific systems, industrial management systems, financial management systems, and advanced information management systems.

Successful use of a higher-level language (ESPOL) for the implementation of the B 5500 MCP has led Burroughs Corporation to the development of more powerful language for implementation of the B 6500 MCP. With it, it is possible to describe the functions of the MCP in a natural, rigorous manner. It is possible to talk about queues, parallel processing, and interlocks as though they were simple variables.

The B 6500 MCP is organized in an hierarchical fashion (see figure 7-1). First is the central, or hard core, section, which acts to merge the system hardware and software. The design criteria for this section are similar to those used by hardware logic designers. The hard core is a small hardware routine used to minimize hardware requirements of other portions of the software, such as required for memory addressing, physical input/output operations, and system security. Conversely, it fields the hardware signals generated by interrupts and causes the appropriate routines of the next section to be executed.

The second area is the resident MCP which remains in primary memory to handle the major task of the MCP: resource allocation relative to primary memory, secondary memory, I/O devices, processors, and time. The third area of the MCP includes routines which reside in the system's disk memory, to be brought into primary memory as required. Functions performed by these routines include library maintenance, file control, job scheduling, control cards, printer backup, and card reader backup. These utility routines are part of the generalized library system for the B 6500.

Burroughs provides the user with a complete set of utility routines. However, since no one standard algorithm is suitable for all customers, this portion of the MCP is highly modular and precisely described relative to interface requirements. This permits a customer to change the algorithms in this section without fear of affecting unknown areas of the first two levels of the MCP. For example, it is possible to implement a different control card format simply by changing the control card program module.

MASTER CONTROL PROGRAM (MCP) FEATURES

The features of the MCP encompass three general areas: computer functions, automatic system assignment and coordination, and multiprocessing.

Computer Functions

The MCP controls all computer functions except those involving physical handling of such items as tape, cards, paper, etc. It provides supervisory control instructions to the operator so that he may make major decisions in directing the processing, with all other control handled automatically by the MCP. The operator is always able to communicate with the system when the need arises. The MCP can analyze error conditions and provide a standard course of action where a specific course of action is not provided in the object program.

Automatic System Assignment and Coordination

Certain operations which are the responsibility of the programmer with other computer systems are performed automatically with the B 6500. For example, the MCP is able to assign memory areas

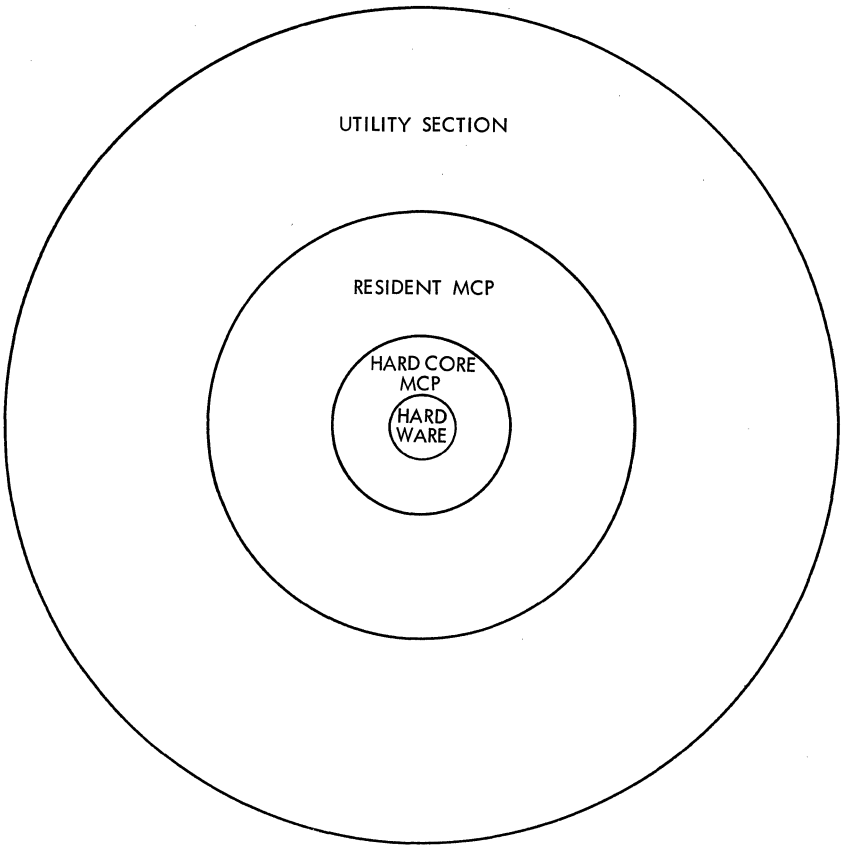


Figure 7-1. Organization of B 6500 MCP

of program segments, input/output areas and the program stack. Input and output files are identified automatically and assigned to physical units. Standard tape-handling procedures provide for labeling, end-of-reel, and end-of-file conditions.

Multiprocessing

The MCP determines the optimum sequence and combination for the processing of a batch of jobs, adjusting its schedule when new or higher priority jobs are introduced, and dynamically allocating memory to gain maximum efficiency in processing a reordered job sequence. The MCP also controls the execution of program segments and initiates all input/output operations. Because of its ability to oversee all programs in process, it helps to maximize efficiency of input/output device operation.

ELEMENTS OF THE MASTER CONTROL PROGRAM

Following are some of the special-purpose routines included in the MCP that anticipate and provide for various operating circumstances:

a. An executive routine which coordinates the operation of the MCP by determining the type of control function required and transferring control to the proper routine, and supervises input/output operations and execution of all program segments in memory. When an interrupt occurs, this routine determines the cause and either notifies the operator or takes action to rectify the situation or to continue processing.

b. A scheduling routine which evaluates the priority and equipment requirements of a batch of programs, scheduling and rescheduling to maintain efficient and continuous processing.

c. An environment control routine that dynamically allocates memory and assigns input/output devices according to the needs of the object programs.

d. An exception condition routine which provides standard error-handling procedures.

Entry to Control State

The processor transfers to the control state in response to an interrupt, which may result from operator communication with the system, developments in the program being executed (e.g., initiation of an I/O operation), hardware malfunction, etc. For example, an interrupt is generated whenever an I/O operation is terminated in order to notify the MCP that a data switching channel is free. The MCP then initiates a new operation, to achieve maximum system utilization. When an interrupt condition occurs, control is transferred to the MCP upon completion of the current program syllable. The contents of the arithmetic and control registers are then stored in the program stack; the executive routine determines the cause of the interrupt; and, control is transferred to the appropriate MCP routine. When the interrupt condition has been serviced, control is returned to an object program, with the registers restored from the appropriate program stack.

In the following paragraphs the elements of the MCP are examined in greater detail, along with the function and interrelation of each routine.

Executive Routine

An MCP initialization routine is loaded into the system when the LOAD switch on the console is depressed. This in turn loads the required portions of the MCP executive routine into memory. The initialization routine is then released from memory. As the coordinating member of the MCP, the executive routine has six basic functions:

a. To initiate all input/output operations.
b. To analyze all interrupt conditions and provide an appropriate course of action.
c. To maintain control over all programs.
d. To control the use of other routines in the MCP.
e. To maintain an operation log.
f. To maintain an internal physical system description.

INPUT/OUTPUT OPERATIONS

In order to initiate and coordinate all I/O operations, the executive routine maintains several tables in memory. Information about all the input/output descriptors for all programs in memory is recorded in these tables. Access through these tables enables the executive routine to reference a particular I/O descriptor when an I/O operation is to be executed. Thus the executive routine can evaluate the status of any descriptor at any time. The executive routine, then, has constantly updated information on the location and status of all I/O descriptors.

INITIATION OF AN INPUT/OUTPUT OPERATION. MCP action on an I/O descriptor causes the operation to be executed. When an I/O operation is being executed, the descriptor is set to indicate that the buffer area referenced is not available to the object program. When an object program has processed all the data in an input area or filled an output area with information, it requests an I/O operation of the MCP by executing a read or write statement. The executive routine can then initiate the operation specified by executing the I/O descriptor when an input/output path to the specific device becomes available. Processing is interrupted only long enough for the executive routine to initiate the specified operation, after which the data switching channel proceeds independently of the processor. Control is then returned to a program segment of one of the programs in the mix.

Since simultaneous input/output operations can be executed on the B 6500, it is possible that several descriptors may be in the tables maintained by the MCP. Therefore, the executive routine must continually update the I/O tables whenever I/O operations are initiated or completed.

Programs with a preponderance of I/O operations frequently use multiple read or write areas so that there is always storage for data being read in or for results waiting to be written out. Thus processing is not delayed while necessary data is loaded or while results are cleared from the output area. Multiple read and write areas are normally used in a regular sequence, and the I/O descriptors that reference these areas are linked so that each I/O descriptor is executed in sequence.

COMPLETION OF AN INPUT/OUTPUT OPERATION. When the specified operation has been executed, the multiplexor generates an I/O finish interrupt. The MCP obtains the result descriptor from the multiplexor. This descriptor indicates whether or not the operation was completed successfully and provides information regarding the type of operation performed, unit designation, end-of-file or end-of-reel condition, and presence of parity or other errors. There are two possible courses of action:

a. With the successful completion of the operation, the I/O descriptor just processed is restored to indicate that the buffer is available. If another descriptor is ready for processing, the executive routine initiates the specified operation before transferring control to a program. Control is returned to a program segment of one of the programs in the mix (refer to the Schedule Routine).
b. Unsuccessful completion of the operation results in the executive routine examining the result descriptor to determine the cause of failure and taking corrective action. For example, if a parity error occurred on a tape read, the tape would be automatically re-read several times. Persistent failure would result either in discontinuation of the program to which that tape is attached or bypassing the information if such action is specified in the object program.

   When an output operation is unsuccessful, the MCP retains the output information, thus allowing for additional attempts at transmission.

An end-of-reel condition indicated by the external-result descriptor causes the executive routine to locate the next input file or output tape and to provide the required tape-handling procedures. In the case of an end-of-file condition, the executive routine refers to the object program for specific instructions.

HANDLING INTERRUPT CONDITIONS.

When any interrupt condition occurs, control is transferred automatically to the executive routine, which ascertains the cause of interrupt and initiates appropriate program action. During compilation, certain operators which cause interrupts may be generated into the object program to provide for necessary MCP operations. Their use in connection with I/O operations has just been described. Interrupts may also indicate a need for additional program segments or memory space or the completion of a program.

Other interrupts are associated with standard program checks that determine such arithmetic conditions as exponent overflow and underflow, integer overflow, or divide by zero. The object program may specify the action to be taken when these interrupts occur, otherwise the MCP will follow a standard procedure.

Interrupt conditions may be classified as either processor-dependent or processor-independent. The type of interrupt determines the order of executive routine response. In addition, there are hardware failure interrupts, such as memory parity errors, which cause the console operator to be notified of an equipment failure.

CONTROL OF PROGRAM SEGMENTS.

When processing of a program segment is interrupted, the B 6500 automatically stores the address of the next syllable to be executed. Thus after the interrupt condition has been satisfied, control can be transferred to the proper point in the program. If control is transferred to a non-present program segment, a presence bit interrupt is generated, requiring the MCP to read that program segment into core memory.

OTHER MASTER CONTROL PROGRAM ROUTINES.

The executive routine loads and transfers control to the schedule, environment control, and error routines when the need arises. If, for example, the console operator requests a revision of job priorities, the executive routine responds by loading the schedule routine. When error conditions arise, the executive routine determines the type of error and selects and loads the appropriate routine to handle the situation.

MAINTENANCE OF AN OPERATION LOG.

A record is maintained of processing time for each program, including the time the job was started, elapsed running time, and the actual processor time. Thus at any time during processing, the console operator may query the MCP to obtain the elapsed running time for a job. When the total processing time is known, the operator can then determine the amount of time needed to complete the job. Such information is valuable if a new job is to be introduced into the system or if priority changes require a re-ordering of the current processing sequence. The MCP records the log for each job on disk.

SUMMARY.

The executive routine coordinates the functions of the Master Control Program, responds to all interrupt conditions, maintains control over program segments and other MCP routines, maintains a constant system description and operations log, and provides for system operator communication.

The Schedule Routine

This routine relieves the programmer and operator of scheduling and ensures an effective multiprocessing environment. The operator loads program parameter cards as inputs to the schedule routine and the executive routine loads the schedule routine to memory. The schedule routine:

a. Determines the sequence of jobs to be run, and in a multiprocessing environment, determines the best program mix. Priority ratings, system requirements of each object program, and the present system configuration are considered.
b. Reschedules job sequence whenever a higher priority job is introduced. (Adjusting a job sequence during a production run is called dynamic rescheduling.)
c. Relays information about the jobs scheduled to the environment control routine.

Program Backlog Table

To perform the above functions, the schedule routine develops a program backlog table of all programs in the mix. The elements of this table are formulated from priority ratings provided by the operator and the program parameters produced by the compiler.

When all parts of the program backlog table have been developed, it contains, in order of priority, job identification, input/output requirements, memory requirements, file descriptions, processing status (running or finished), time started, time completed, and input/output time. This table, maintained in memory throughout processing by the executive routine, is used by the environment control routine for memory allocation and input/output unit assignments.

Program-Finish Conditions

When a job is completed, the environment control routine scans the control tables and adjusts all entires pertaining to that job. The next program to be run is located in the program backlog table. The environment control routine makes unit and memory assignments as described, advises the operator of file requirements, loads and initializes the program stack and the base location tables, loads the first segment, and transfers control to this segment.

Changing the Schedule

If a new job with a higher priority rating is introduced into the system, the schedule routine alters the program backlog table.

PROBLEM-ORIENTED LANGUAGES

Three problem-oriented programing languages are available with a Burroughs B 6500 System: COBOL for the solution of business oriented problems, ALGOL and FORTRAN for the solution of scientific problems as well as those in other areas of application.

Problem-oriented languages allow the programmer to state a problem in a manner directly adapted to the given situation, business problems in business-like statements, scientific problems in algebraic-type equations. Because such languages are largely machine independent, program modification and program exchange between users of different systems are accomplished with relative ease. The B 6500 leads itself to compiler implementation. For example, the hardware Edit command, within a single instruction, includes all of the editing functions required for COBOL without being limited to them.

COBOL

COBOL for the B 6500 is the CODASYL 68, (COBOL-68) language. B 6500 COBOL is an extended version which provides the user with the complete facilities of the B 6500 hardware/software system without resorting to machine or assembly language. In particular, automatic program segmentation, automatic peripheral assignment, dynamic multiprocessing, the inclusion of debugging language statements allow a high degree of sophistication in program design. B 6500 COBOL continues this sophistication through the coding and debugging stages of application development.

A program written in COBOL, the source program, is accepted as input to the B 6500 COBOL Compiler. The Compiler first verifies that the COBOL rules have been followed and then generates an object program in executable machine language. Due to the speed of compilation, no object deck is required; instead, the object program is placed on disk. It may also be dumped on magnetic tape for back-up storage. Should program changes become necessary, the source deck is corrected and a new compilation run made; thus

the source deck always reflects the object program being executed. Compilation speed on the B 6500 is 5000 card images per minute.

ALGOL

Burroughs Extended ALGOL is based on the definitive "Revised Report on the Algorithmic Language ALGOL 60" (Communications of the ACM, Vol. 6, No. 1, January, 1963). It permits the programmer to exercise close control over data transmission and manipulation to any desired degree.

ALGOL compilation speed on the B 6500 is 10,000 cards per minute.

FORTRAN

B 6500 FORTRAN is compatible with IBM 7094 FORTRAN IV version 13. A great deal of programing flexibility is permitted the user due to the increased difficulty in enforcing restrictions on level 13 Fortran. Fortran programs written for the B 5500 can be compiled on the B 6500. Compilation speed of Fortran programs is comparable to that of ALGOL.

Compiling a Source Program

A program written in a compiler language is converted from this source language into the internal machine language by the applicable compiler. The object program is compiled under any one of the following options:

a. Object program compiled with immediate execution (compile and go under control of the MCP).
b. Object program compiled to the Disk File Library for future execution.
c. Object program compiled for a syntax check only (no execution of the object program).
d. Various source language input and output options can be specified during the compilation as directed by the programmer.

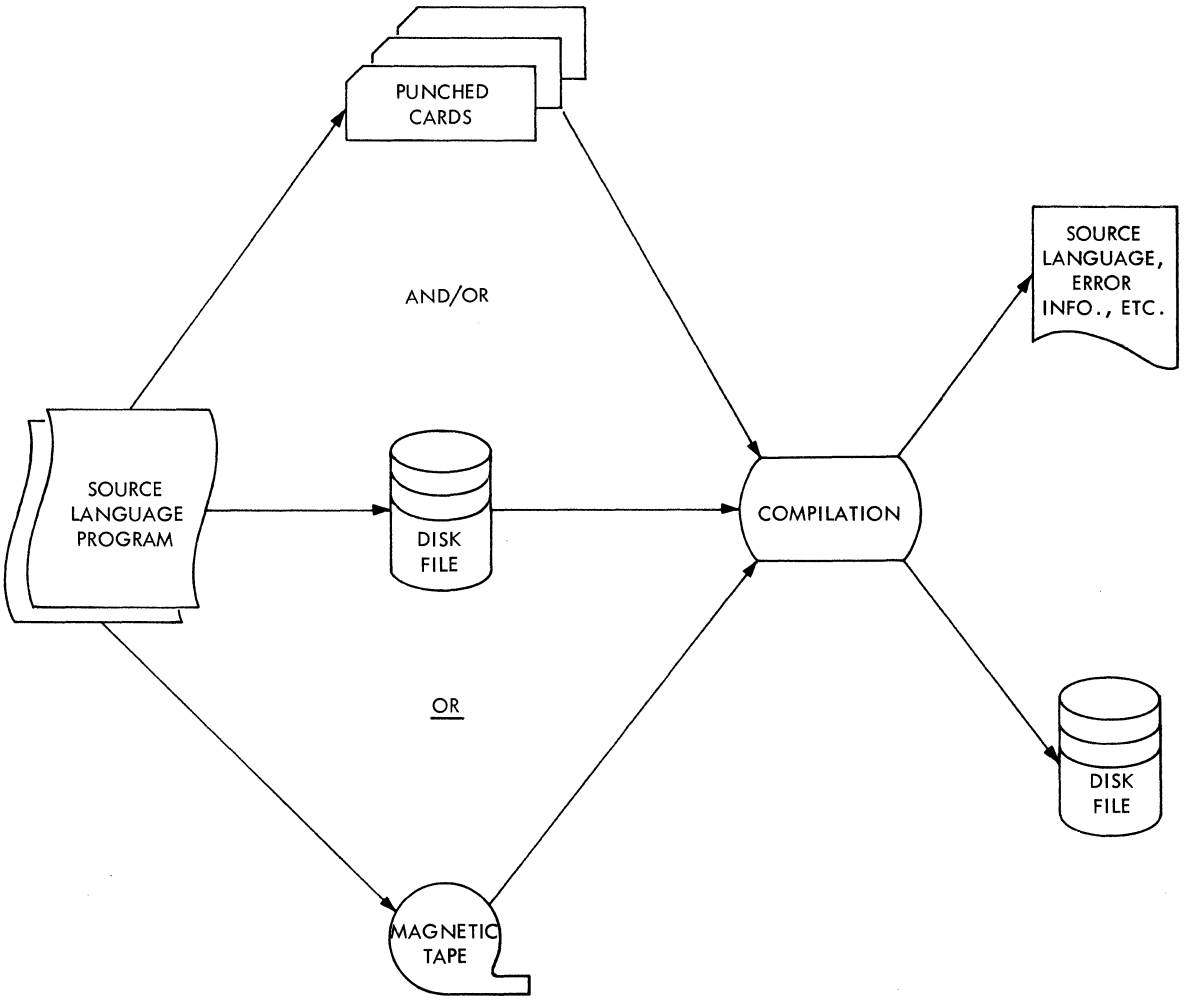


Figure 7-2. Compilation of a Source Program

The compiled program consists of:

a. A program parameter record containing program memory requirements, peripheral equipment requirements, and information about the segment dictionary.
b. File information blocks containing information related to each file.
c. The main program with one or more program segments.

Figure 7-2 illustrates the transformation from the source to object program.

APPLICATION PROGRAMS

The Burroughs B 6500 offers a wide range of applications programs for business, finance, and science. Major application areas include:

a. Scientific Systems.
b. Industrial Management Systems.
c. Financial Management Systems.
d. Advanced Information Management Systems.

Scientific Systems

Scientific Systems for the B 6500 include an Advanced Mathematical Programing System, a Statistical Programing System, and Simulation Techniques.

ADVANCED MATHEMATICAL PROGRAMING SYSTEM.

This system utilizes the latest techniques to provide:

a. Unlimited variables.
b. Unique control language.
c. Matrix generator.
d. Sophisticated report writer.
e. "Crash" algorithm.
f. Decomposition algorithm.
g. Full and mixed integer capability.
h. Non-linear algorithm.
i. Extensive parametric capability.
j. Extensive sensitivity capability.

STATISTICAL PROGRAMING.

The B 6500 is provided with an Advanced Business Statistics System which encompasses such areas as time series analysis, regression and correlation analysis, factor analysis, and variance analysis.

SIMULATION TECHNIQUES.

For those engaged in designing simulation experiments, the B 6500 provides two powerful languages, DYNAMO and SIMULA.

DYNAMO. DYNAMO[1] is an easy-to-use, special-purpose compiler that translates mathematical models from simple algebraic relationships into tabulated and plotted results. The models may represent any "servo-mechanism" system, whether social, biological, or mechanical.

DYNAMO accepts equations and checks them for consistency, proper equation formulation, and keypunch accuracy. If the equations check, object code is produced. Output from DYNAMO is in the form of graphic plots.

SIMULA. SIMULA (Simulation Language) provides a method of precisely describing "discrete event systems." Designed by the Norwegian Computing Center in Oslo, SIMULA is currently being used for analysis of nerve networks, communications systems, traffic flow, production systems, etc.

SIMULA is an extension of ALGOL 60 and contains ALGOL 60 as a subset. The extension includes new basic concepts, new statements which operate upon these concepts, and a set of library procedures. In addition, certain ALGOL statements have been extended to operate upon the new concepts.

SIMULA also contains extensive list processing facilities.

Industrial Management Systems

The Industrial Management Systems provide application programs for:

a. Inventory Management.
b. Production Management.

[1] DYNAMO was developed by the Massachusetts Institute of Technology.

INVENTORY MANAGEMENT.

Inventory Management was one of the first areas of the business process to be brought under examination. The objective was to develop a valid technique for handling inventory problems and to provide for its implementation. To meet requirements for a scientific inventory management system, Burroughs offers B 6500 users BICS and ACT!ON.

BURROUGHS INVENTORY CONTROL SYSTEM (BICS). BICS is a set of computer programs representing the latest techniques in the application of computer technology to the existing body of inventory management theory. The BICS objective is to produce a man-computer team which will significantly increase management control over general business inventories. To an organization not currently employing a scientific inventory management system, BICS, written entirely in COBOL, represents a development value estimated conservatively at $100,000. This system is provided at no additional cost to B 6500 users as part of Burroughs continuing program of management assistance.

ACT!ON SYSTEM. Burroughs B 6500 ACT!ON System is a generalized manufacturing inventory accounting and control system which integrates manufacturing planning, computer programs. Burroughs computers, and a system of machine language "transacting" documents to effect improved business management.

With the ACT!ON System, significant manufacturing transactions are transmitted to a set of data processing records maintained in computer files. These files are subject to continual review by highly refined computer programs, which, following the exception principle, immediately signal variance from management standards. Every reasonable precaution has been incorporated to minimize material shortage. The system tells what materials are needed and when they are to be ordered. When material orders are delayed, reminders are issued. When conditions alter the schedule, reschedule messages maintain material balance. When material receipt is late, ACT!ON messages are issued. A final warning is provided to minimize the possibility of out-of-stock conditions.

With ACT!ON, production is not bound by out-of-date status schedules. Major rescheduling efforts may be handled with ease; production planning elements are rescheduled independently; production and material requirement schedules are maintained in balance with the servo-mechanism value of the system. Today, manufacturing personnel employ a new set of tools which result in:

a. Minimum shortages.
b. Surplus elimination.
c. Schedule Maintenance.
d. Cost reduction.
e. Increased profits.

PRODUCTION MANAGEMENT.

Studies show that new users of data processing systems usually implement only financial and administrative applications. With experience, however, users find that big savings can be realized by computerizing plant and distribution operations. The prime objective of any production organization is to complete and deliver a product on schedule without increased inventory, extensive overtime, or excessive machine idle time.

To assist the users of B 6500 Systems, Burroughs offers several management tools. Among these are PROMIS (Project Oriented Management Information System), and Production Accounting System.

PROJECT ORIENTED MANAGEMENT INFORMATION SYSTEM (PROMIS). While PERT techniques are management tools for the planning and control of projects, PROMIS is designed to provide the communications vehicle through which planning is accomplished and a project instrumented for control. PROMIS is designed to overcome difficulties encountered with existing network systems. With use of fast random access storage and list processing techniques, information can be maintained through the use of data communications and remote stations, eliminating the time lag inherent in batch processing systems.

PROMIS is built on the data base concept and allows simultaneous maintenance of many networks, using different calendars, date options, and computation features. The latter include "percent complete" and "time now" options. Three types of slack are maintained.

Time estimates may be given in hours, days, shifts, weeks, or months and may be intermixed throughout a network. Time estimates may also be expressed as work-content and rates of work (i.e., 300 man-hours at 16 man-hours per day), Holidays and work patterns are automatically considered.

Directed dates and schedule dates are allowed to impose external constraints on activities or events.

Reporting is accomplished through a report generator which allows specification of sort keys, editing, and suppression of fields. The reports may be sorted and edited by levels, fragments, or special report flags, as well as by standard flags. Such extensive editing capabilities allow the use of exception reporting, with remote inquiry provided for immediate response.

A new element has been developed known as a "prerequisite" which allows constraints to be defined which contain no resources and have no duration such as equipment deliveries, budget approvals, etc. Prerequisites are specified as constraints for "events."

Comments may be added to any element of the system. Comments consist of alpha-numeric information describing commitments, methods of work, resources or any explanation of events, activities, networks, or prerequisites. Any number of comments may be associated with any element of the system.

PRODUCTION ACCOUNTING SYSTEM. The Production Accounting System maintains job or lot control from the work floor using remote input for real-time update. Also included is a product configuration control, allowing several types of explosion and implosion inquiries. Documents are created for maintaining material and tool control as well as the job or lot control. Analysis programs are provided for machine and work center efficiency reporting. The system allows for remote inquiry and updating, product routing and standards, and provides the capability for efficient reporting by job or lot.

Financial Systems

A full line of financial systems are available with the B 6500. Among these are High-Volume Paper Processing Systems for MICR and OCR documents. Also available is an Integrated Banking System which cross-indexes all files that apply to an application, and then cross-references these files with files of all other applications on the system. The result is a complete customer profile.

Advanced Information Management System

The Advanced Information Management System is a data-base system providing a total systems approach to management problems, through the use of Burroughs head-per-track disk file for rapid response to inquiries and data communication facilities to permit information exchange between home office and field locations.

The above discussion of application programs represents but a cross-section of those provided for the B 6500.

APPENDIX A OPERATORS,ALPHABETICAL LIST

NAME	MNEMONIC	HEXADECIMAL CODE
ADD	ADD	P80
BIT RESET	BRST	P9E
BIT SET	BSET	P96
BRANCH FALSE	BRFL	PA0
BRANCH TRUE	BRTR	PA1
BRANCH UNCONDITIONAL	BRUN	PA2
CHANGE SIGN BIT	CHSN	P8E
COMPARE CHARACTERS EQUAL, DESTRUCTIVE	CEQD	PF4
COMPARE CHARACTERS EQUAL, UPDATE	CEQU	PFC
COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED	PF1
COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU	PF9
COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD	PF2
COMPARE CHARACTERS GREATER, UPDATE	CGTU	PFA
COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED	PF3
COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU	PFB
COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD	PF0
COMPARE CHARACTERS LESS, UPDATE	CLSU	PF8
COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED	PF5
COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU	PFD
CONDITIONAL HALT	HALT	UDF
COUNT BINARY ONES	CBON	VBB
DELETE TOP OF STACK	DLET	PB5
DISABLE EXTERNAL INTERRUPTS	DEXI	V47
DIVIDE	DIVD	P83
DUPLICATE TOP OF STACK	DUPL	PB7
DYNAMIC BIT RESET	DBRS	P9F
DYNAMIC BIT SET	DBST	P97
DYNAMIC BRANCH FALSE	DBFL	PA8
DYNAMIC BRANCH TRUE	DBTR	PA9
DYNAMIC BRANCH UNCONDITIONAL	DBUN	PAA
DYNAMIC FIELD INSERT	DINS	P9D
DYNAMIC FIELD ISOLATE	DISO	P9B
DYNAMIC FIELD TRANSFER	DFTR	P99
DYNAMIC SCALE LEFT	DSLF	PC1
DYNAMIC SCALE RIGHT FINAL	DSRF	PC7
DYNAMIC SCALE RIGHT ROUND	DSRR	PC9
DYNAMIC SCALE RIGHT SAVE	DSRS	PC5
DYNAMIC SCALE RIGHT TRUNCATE	DSRT	PC3
ENABLE EXTERNAL INTERRUPTS	EEXI	V46
END EDIT	ENDE	EDE
END FLOAT	ENDF	ED5
ENTER	ENTR	PAB
EQUAL	EQUL	P8C
ESCAPE TO 16-BIT INSTRUCTION	VARI	P95
EVALUATE DESCRIPTOR	EVAL	PAC

APPENDIX A (cont.)

NAME	MNEMONIC	HEXADECIMAL CODE
EXCHANGE	EXCH	PB6
EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU	PDD
EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD	PD2
EXECUTE SINGLE MICRO, UPDATE	EXSU	PDA
EXIT	EXIT	PA3
EXTENDED MULTIPLY	MULX	P8F
FIELD INSERT	INSR	P9C
FIELD ISOLATE	ISOL	P9A
FIELD TRANSFER	FLTR	P9B
GREATER THAN	GRTR	P8A
GREATER THAN OR EQUAL	GREQ	P89
IDLE UNTIL INTERRUPT	IDLE	V44
INDEX	INDX	PA6
INDEX AND LOAD NAME	NXLN	PA5
INDEX AND LOAD VALUE	NXLV	PAD
INPUT CONVERT, DESTRUCTIVE	ICVD	PCA
INPUT CONVERT, UPDATE	ICVU	PCB
INSERT CONDITIONAL	INSC	EDD
INSERT DISPLAY SIGN	INSG	ED9
INSERT MARK STACK	IMKS	PCF
INSERT OVERPUNCH	INOP	ED8
INSERT UNCONDITIONAL	INSU	EDC
INTEGER DIVIDE	IDIV	P84
INTEGERIZE, ROUNDED	NTGR	P87
INTEGERIZE, ROUNDED, DOUBLE PRECISION	NTGD	V87
INTEGERIZE, TRUNCATED	NTIA	P86
INTERRUPT OTHER PROCESSORS	HEYU	V4F
INVALID OPERATOR	NVLD	UFF
LEADING ONE TEST	LOG2	V8B
LESS THAN	LESS	P88
LESS THAN OR EQUAL	LSEQ	P8B
LINKED LIST LOOKUP	LLLU	VBD
LIT CALL ONE	ONE	PB1
LIT CALL ZERO	ZERO	PB0
LIT CALL 16 BITS	LT16	PB3
LIT CALL 48 BITS	LT48	PBE
LIT CALL 8 BITS	LT8	PB2
LOAD	LOAD	PBD
LOAD TRANSPARENT	LODT	VBC
LOGICAL AND	LAND	P90
LOGICAL EQUAL	SAME	P94
LOGICAL EQUIVALENCE	LEQV	P93
LOGICAL NEGATE	LNOT	P92
LOGICAL OR	LOR	P91
MAKE PROGRAM CONTROL WORD	MPCW	PBF

APPENDIX A (cont.)

NAME	MNEMONIC	HEXADECIMAL CODE
MARK STACK	MKST	PAE
MASKED SEARCH FOR EQUAL	SRCH	VBE
MOVE CHARACTERS	MCHR	ED7
MOVE NUMERIC UNCONDITIONAL	MVNU	ED6
MOVE TO STACK	MVST	VAF
MOVE WITH FLOAT	MFLT	ED1
MOVE WITH INSERT	MINS	ED0
MULTIPLY	MULT	P82
NAME CALL	NAMC	P40=> 7F
NO OPERATION	NOOP	UFE
NOT EQUAL	NEQL	P8D
OCCURS INDEX	OCRX	V85
OVERWRITE DESTRUCTIVE	OVRD	PBA
OVERWRITE NON-DESTRUCTIVE	OVRN	PBB
PACK DESTRUCTIVE	PACD	PD1
PACK UPDATE	PACU	PD9
PUSH DOWN STACK REGISTERS	PUSH	PB4
READ AND CLEAR OVERFLOW FLIP-FLOP	ROFF	PD7
READ PROCESSOR IDENTIFICATION	WHOI	V4E
READ PROCESSOR REGISTER	RPRR	VB8
READ TAG FIELD	RTAG	VB5
READ TRUE/FALSE FLIP-FLOP	RTFF	PDE
READ WITH LOCK	RDLK	VBA
REMAINDER DIVIDE	RDIV	P85
RESET FLOAT	RSIF	ED4
RETURN	RETN	PA7
ROTATE STACK DOWN	RSDN	VB7
ROTATE STACK UP	RSUP	VB6
SCALE LEFT	SCLF	PC0
SCALE RIGHT FINAL	SCRF	PC6
SCALE RIGHT ROUND	SCRR	PC8
SCALE RIGHT SAVE	SCRS	PC4
SCALE RIGHT TRUNCATE	SCRT	PC2
SCAN IN	SCNI	V4A
SCAN OUT	SCNO	V4B C
SCAN WHILE EQUAL, DESTRUCTIVE	SEQD	VF4
SCAN WHILE EQUAL, UPDATE	SEQU	VFC
SCAN WHILE FALSE, DESTRUCTIVE	SWFD	VD4
SCAN WHILE FALSE, UPDATE	SWFU	VDC
SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED	VF1
SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU	VF9
SCAN WHILE GREATER, DESTRUCTIVE	SGTD	VF2
SCAN WHILE GREATER, UPDATE	SGTU	VFA
SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED	VF3
SCAN WHILE LESS OR EQUAL, UPDATE	SLEU	VFB

APPENDIX A (cont.)

NAME	MNEMONIC	HEXADECIMAL CODE
SCAN WHILE LESS, DESTRUCTIVE	SLSD	VF0
SCAN WHILE LESS, UPDATE	SLSU	VF8
SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED	VF5
SCAN WHILE NOT EQUAL, UPDATE	SNEU	VFD
SCAN WHILE TRUE, DESTRUCTIVE	SWTD	VD5
SCAN WHILE TRUE, UPDATE	SWTU	VDD
SET DOUBLE TO TWO SINGLES	SPLT	V43
SET EXTERNAL SIGN	SXSN	PD6
SET INTERVAL TIMER	SINT	V45 C
SET PROCESSOR REGISTER	SPRR	VB9
SET TAG FIELD	STAG	VB4
SET TO DOUBLE PRECISION	XTND	PCE
SET TO SINGLE PRECISION, ROUNDED	SNGL	PCD
SET TO SINGLE PRECISION, TRUNCATED	SNGT	PCC
SET TWO SINGLES TO DOUBLE	JOIN	V42
SKIP FORWARD DESTINATION CHARACTERS	SFDC	EDA
SKIP FORWARD SOURCE CHARACTERS	SFSC	ED2
SKIP REVERSE DESTINATION CHARACTERS	SRDC	EDB
SKIP REVERSE SOURCE CHARACTERS	SRSC	ED3
STEP AND BRANCH	STBR	PA4
STORE DESTRUCTIVE	STOD	PB8
STORE NON-DESTRUCTIVE	STON	PB9
STRING ISOLATE	SISO	PD5
STUFF ENVIRONMENT	STFF	PAF
SUBTRACT	SUBT	P81
TABLE ENTER EDIT, DESTRUCTIVE	TEED	PD0
TABLE ENTER EDIT, UPDATE	TEEU	PD8
TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND	PE6
TRANSFER UNCONDITIONAL, UPDATE	TUNU	PEE
TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD	PE4
TRANSFER WHILE EQUAL, UPDATE	TEQU	PEC
TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD	VD2
TRANSFER WHILE FALSE, UPDATE	TWFU	VDA
TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED	PE1
TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU	PE9
TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD	PE2
TRANSFER WHILE GREATER, UPDATE	TGTU	PEA
TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED	PE3
TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU	PEB
TRANSFER WHILE LESS, DESTRUCTIVE	TLSD	PE0
TRANSFER WHILE LESS, UPDATE	TLSU	PE8
TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED	PE5
TRANSFER WHILE NOT EQUAL, UPDATE	TNEU	PED
TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD	VD3
TRANSFER WHILE TRUE, UPDATE	TWTU	VDB

APPENDIX A (cont.)

NAME	MNEMONIC	HEXADECIMAL CODE
TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD	PD4
TRANSFER WORDS OVERWRITE UPDATE	TWOU	PDC
TRANSFER WORDS, DESTRUCTIVE	TWSD	PD3
TRANSFER WORDS, UPDATE	TWSU	PDB
TRANSLATE	TRNS	VD7
UNPACK ABSOLUTE, DESTRUCTIVE	UABD	VD1
UNPACK ABSOLUTE, UPDATE	UABU	VD9
UNPACK SIGNED, DESTRUCTIVE	USND	VD0
UNPACK SIGNED, UPDATE	USNU	VD8
VALUE CALL	VALC	P00=> 3F

NOTES:

The hexadecimal value of the code is preceded by a P, V, U, or E. All codes preceded by a P (primary) are one syllable long. All codes preceded by a V (variant) are two syllables long, the first syllable being hexadecimal 95 (Escape to 16 Bit Instruction). All codes preceded by a U (unconditional) can be used as one syllable codes, or as two syllable codes when preceded by hexadecimal 95. All codes preceded by an E (edit) are Edit micro-operators.

Name Call and Value Call are an exception to the above and are always two syllables.

If the hexadecimal code is followed by a C, the operator is only valid in control state. In the case of Scan Out, only some functions are restricted to control state.

APPENDIX B OPERATORS,NUMERICAL LIST

HEXADECIMAL CODE	NAME	MNEMONIC
UDF	CONDITIONAL HALT	HALT
UFE	NO OPERATION	NOOP
UFF	INVALID OPERATOR	NVLD
P00=>3F	VALUE CALL	VALC
P40=>7F	NAME CALL	NAMC
P80	ADD	ADD
P81	SUBTRACT	SUBT
P82	MULTIPLY	MULT
P83	DIVIDE	DIVD
P84	INTEGER DIVIDE	IDIV
P85	REMAINDER DIVIDE	RDIV
P86	INTEGERIZE, TRUNCATED	NTIA
P87	INTEGERIZE, ROUNDED	NTGR
P88	LESS THAN	LESS
P89	GREATER THAN OR EQUAL	GREQ
P8A	GREATER THAN	GRTR
P8B	LESS THAN OR EQUAL	LSEQ
P8C	EQUAL	EQUL
P8D	NOT EQUAL	NEQL
P8E	CHANGE SIGN BIT	CHSN
P8F	EXTENDED MULTIPLY	MULX
P90	LOGICAL AND	LAND
P91	LOGICAL OR	LOR
P92	LOGICAL NEGATE	LNOT
P93	LOGICAL EQUIVALENCE	LEQV
P94	LOGICAL EQUAL	SAME
P95	ESCAPE TO 16-BIT INSTRUCTION	VARI
P96	BIT SET	BSET
P97	DYNAMIC BIT SET	DBST
P98	FIELD TRANSFER	FLTR
P99	DYNAMIC FIELD TRANSFER	DFTR
P9A	FIELD ISOLATE	ISOL
P9B	DYNAMIC FIELD ISOLATE	DISO
P9C	FIELD INSERT	INSR
P9D	DYNAMIC FIELD INSERT	DINS
P9E	BIT RESET	BRST
P9F	DYNAMIC BIT RESET	DBRS
PA0	BRANCH FALSE	BRFL
PA1	BRANCH TRUE	BRTR
PA2	BRANCH UNCONDITIONAL	BRUN
PA3	EXIT	EXIT
PA4	STEP AND BRANCH	STBR
PA5	INDEX AND LOAD NAME	NXLN
PA6	INDEX	INDX
PA7	RETURN	RETN

APPENDIX B (cont.)

HEXADECIMAL CODE	NAME	MNEMONIC
PA8	DYNAMIC BRANCH FALSE	DBFL
PA9	DYNAMIC BRANCH TRUE	DBTR
PAA	DYNAMIC BRANCH UNCONDITIONAL	DBUN
PAB	ENTER	ENTR
PAC	EVALUATE DESCRIPTOR	EVAL
PAD	INDEX AND LOAD VALUE	NXLV
PAE	MARK STACK	MKST
PAF	STUFF ENVIRONMENT	STFF
PB0	LIT CALL ZERO	ZERO
PB1	LIT CALL ONE	ONE
PB2	LIT CALL 8 BITS	LT8
PB3	LIT CALL 16 BITS	LT16
PB4	PUSH DOWN STACK REGISTERS	PUSH
PB5	DELETE TOP OF STACK	DLET
PB6	EXCHANGE	EXCH
PB7	DUPLICATE TOP OF STACK	DUPL
PB8	STORE DESTRUCTIVE	STOD
PB9	STORE NON-DESTRUCTIVE	STON
PBA	OVERWRITE DESTRUCTIVE	OVRD
PBB	OVERWRITE NON-DESTRUCTIVE	OVRN
PBD	LOAD	LOAD
PBE	LIT CALL 48 BITS	LT48
PBF	MAKE PROGRAM CONTROL WORD	MPCW
PC0	SCALE LEFT	SCLF
PC1	DYNAMIC SCALE LEFT	DSLF
PC2	SCALE RIGHT TRUNCATE	SCRT
PC3	DYNAMIC SCALE RIGHT TRUNCATE	DSRT
PC4	SCALE RIGHT SAVE	SCRS
PC5	DYNAMIC SCALE RIGHT SAVE	DSRS
PC6	SCALE RIGHT FINAL	SCRF
PC7	DYNAMIC SCALE RIGHT FINAL	DSRF
PC8	SCALE RIGHT ROUND	SCRR
PC9	DYNAMIC SCALE RIGHT ROUND	DSRR
PCA	INPUT CONVERT, DESTRUCTIVE	ICVD
PCB	INPUT CONVERT, UPDATE	ICVU
PCC	SET TO SINGLE PRECISION, TRUNCATED	SNGT
PCD	SET TO SINGLE PRECISION, ROUNDED	SNGL
PCE	SET TO DOUBLE PRECISION	XTND
PCF	INSERT MARK STACK	IMKS
PD0	TABLE ENTER EDIT, DESTRUCTIVE	TEED
PD1	PACK DESTRUCTIVE	PACD
PD2	EXECUTE SINGLE MICRO, DESTRUCTIVE	EXSD
PD3	TRANSFER WORDS, DESTRUCTIVE	TWSD
PD4	TRANSFER WORDS OVERWRITE DESTRUCTIVE	TWOD
PD5	STRING ISOLATE	SISO

APPENDIX B (cont.)

HEXADECIMAL CODE	NAME	MNEMONIC
PD6	SET EXTERNAL SIGN	SXSN
PD7	READ AND CLEAR OVERFLOW FLIP-FLOP	ROFF
PD8	TABLE ENTER EDIT, UPDATE	TEEU
PD9	PACK UPDATE	PACU
PDA	EXECUTE SINGLE MICRO, UPDATE	EXSU
PDB	TRANSFER WORDS, UPDATE	TWSU
PDC	TRANSFER WORDS OVERWRITE UPDATE	TWOU
PDD	EXECUTE SINGLE MICRO, SINGLE POINTER UPDATE	EXPU
PDE	READ TRUE/FALSE FLIP-FLOP	RTFF
PE0	TRANSFER WHILE LESS, DESTRUCTIVE	TLSD
PE1	TRANSFER WHILE GREATER OR EQUAL, DESTRUCTIVE	TGED
PE2	TRANSFER WHILE GREATER, DESTRUCTIVE	TGTD
PE3	TRANSFER WHILE LESS OR EQUAL, DESTRUCTIVE	TLED
PE4	TRANSFER WHILE EQUAL, DESTRUCTIVE	TEQD
PE5	TRANSFER WHILE NOT EQUAL, DESTRUCTIVE	TNED
PE6	TRANSFER UNCONDITIONAL, DESTRUCTIVE	TUND
PE8	TRANSFER WHILE LESS, UPDATE	TLSU
PE9	TRANSFER WHILE GREATER OR EQUAL, UPDATE	TGEU
PEA	TRANSFER WHILE GREATER, UPDATE	TGTU
PEB	TRANSFER WHILE LESS OR EQUAL, UPDATE	TLEU
PEC	TRANSFER WHILE EQUAL, UPDATE	TEQU
PED	TRANSFER WHILE NOT EQUAL, UPDATE	TNEU
PEE	TRANSFER UNCONDITIONAL, UPDATE	TUNU
PF0	COMPARE CHARACTERS LESS, DESTRUCTIVE	CLSD
PF1	COMPARE CHARACTERS GREATER OR EQUAL, DESTRUCTIVE	CGED
PF2	COMPARE CHARACTERS GREATER, DESTRUCTIVE	CGTD
PF3	COMPARE CHARACTERS LESS OR EQUAL, DESTRUCTIVE	CLED
PF4	COMPARE CHARACTERS EQUAL, DESTRUCTIVE	CEQD
PF5	COMPARE CHARACTERS NOT EQUAL, DESTRUCTIVE	CNED
PF8	COMPARE CHARACTERS LESS, UPDATE	CLSU
PF9	COMPARE CHARACTERS GREATER OR EQUAL, UPDATE	CGEU
PFA	COMPARE CHARACTERS GREATER, UPDATE	CGTU
PFB	COMPARE CHARACTERS LESS OR EQUAL, UPDATE	CLEU
PFC	COMPARE CHARACTERS EQUAL, UPDATE	CEQU
PFD	COMPARE CHARACTERS NOT EQUAL, UPDATE	CNEU
V42	SET TWO SINGLES TO DOUBLE	JOIN
V43	SET DOUBLE TO TWO SINGLES	SPLT
V44	IDLE UNTIL INTERRUPT	IDLE
V45	SET INTERVAL TIMER	SINT
V46	ENABLE EXTERNAL INTERRUPTS	EEXI
V47	DISABLE EXTERNAL INTERRUPTS	DEXI
V4A	SCAN IN	SCNI
V4B	SCAN OUT	SCNO
V4E	READ PROCESSOR IDENTIFICATION	WHOI
V4F	INTERRUPT OTHER PROCESSORS	HEYU

APPENDIX B (cont.)

HEXADECIMAL CODE	NAME	MNEMONIC
V85	OCCURS INDEX	OCRX
V87	INTEGERIZE, ROUNDED, DOUBLE PRECISION	NTGD
V88	LEADING ONE TEST	LOG2
VAF	MOVE TO STACK	MVST
VB4	SET TAG FIELD	STAG
VB5	READ TAG FIELD	RTAG
VB6	ROTATE STACK UP	RSUP
VB7	ROTATE STACK DOWN	RSDN
VB8	READ PROCESSOR REGISTER	RPRR
VB9	SET PROCESSOR REGISTER	SPRR
VBA	READ WITH LOCK	RDLK
VBB	COUNT BINARY ONES	CBON
VBC	LOAD TRANSPARENT	LODT
VBD	LINKED LIST LOOKUP	LLLU
VBE	MASKED SEARCH FOR EQUAL	SRCH
VD0	UNPACK SIGNED, DESTRUCTIVE	USND
VD1	UNPACK ABSOLUTE, DESTRUCTIVE	UABD
VD2	TRANSFER WHILE FALSE, DESTRUCTIVE	TWFD
VD3	TRANSFER WHILE TRUE, DESTRUCTIVE	TWTD
VD4	SCAN WHILE FALSE, DESTRUCTIVE	SWFD
VD5	SCAN WHILE TRUE, DESTRUCTIVE	SWTD
VD7	TRANSLATE	TRNS
VD8	UNPACK SIGNED, UPDATE	USNU
VD9	UNPACK ABSOLUTE, UPDATE	UABU
VDA	TRANSFER WHILE FALSE, UPDATE	TWFU
VDB	TRANSFER WHILE TRUE, UPDATE	TWTU
VDC	SCAN WHILE FALSE, UPDATE	SWFU
VDD	SCAN WHILE TRUE, UPDATE	SWTU
VF0	SCAN WHILE LESS, DESTRUCTIVE	SLSD
VF1	SCAN WHILE GREATER OR EQUAL, DESTRUCTIVE	SGED
VF2	SCAN WHILE GREATER, DESTRUCTIVE	SGTD
VF3	SCAN WHILE LESS OR EQUAL, DESTRUCTIVE	SLED
VF4	SCAN WHILE EQUAL, DESTRUCTIVE	SEQD
VF5	SCAN WHILE NOT EQUAL, DESTRUCTIVE	SNED
VF8	SCAN WHILE LESS, UPDATE	SLSU
VF9	SCAN WHILE GREATER OR EQUAL, UPDATE	SGEU
VFA	SCAN WHILE GREATER, UPDATE	SGTU
VFB	SCAN WHILE LESS OR EQUAL, UPDATE	SLEU
VFC	SCAN WHILE EQUAL, UPDATE	SEQU
VFD	SCAN WHILE NOT EQUAL, UPDATE	SNEU
ED0	MOVE WITH INSERT	MINS
ED1	MOVE WITH FLOAT	MFLT
ED2	SKIP FORWARD SOURCE CHARACTERS	SFSC
ED3	SKIP REVERSE SOURCE CHARACTERS	SRSC
ED4	RESET FLOAT	RSTF

APPENDIX B (cont.)

HEXADECIMAL CODE	NAME	MNEMONIC
ED5	END FLOAT	ENDF
ED6	MOVE NUMERIC UNCONDITIONAL	MVNU
ED7	MOVE CHARACTERS	MCHR
ED8	INSERT OVERPUNCH	INOP
ED9	INSERT DISPLAY SIGN	INSG
EDA	SKIP FORWARD DESTINATION CHARACTERS	SFDC
EDB	SKIP REVERSE DESTINATION CHARACTERS	SRDC
EDC	INSERT UNCONDITIONAL	INSU
EDD	INSERT CONDITIONAL	INSC
EDE	END EDIT	ENDE

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: _____ FORM: _____

_____ DATE: _____

_____

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

cut along dotted line

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

STAPLE

FOLD DOWN SECOND FOLD DOWN

Postage Will Be Paid by Addressee

No Postage Stamp Necessary If Mailed in the United States

BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
6071 Second Avenue
Detroit, Michigan 48232

attn: Sales Technical Services
Systems Documentation

FOLD UP FIRST FOLD UP

Wherever There's
Business There's

Burroughs

1032315 6-69 Printed in U. S. America