Burroughs

MEDIUM SYSTEMS

B 3700/B 4700

CENTRAL PROCESSOR

REFERENCE MANUAL

AA828689 AA995115 AA120798

Burroughs Corporation

Detroit, Michigan 48232

$2.00

Printed in U.S. America

12-72

1063799


Burroughs B700
Burroughs B3700

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF TABLES

B 3700/B 4700 CENTRAL PROCESSOR

INTRODUCTION

The B 3700 system and the even more powerful B 4700 system constitute Burroughs response to the EDP equipment users' demand for more efficient, faster, and more sophisticated data-processing systems. The B 3700 and B 4700 systems are character-oriented toward business and data communications applications. They consist of highly efficient modular hardware/program products, plus a wide range of exceptional peripheral devices that fulfill the requirements involved in solving the data gathering and immediate response requirements that confront data processing installations today and in the near future.

It is the purpose of this manual to acquaint the reader with the Central Processor and associated components offered by Burroughs Corporation which are applicable to the B 3700 and B 4700 systems.

SYSTEM DESCRIPTION

B 3700

The B 3700 is characterized by a 3 MHz clock speed and a memory cycle time of **667** nanoseconds. Memory sizes range from 100,000 to 300,000 bytes of solid state (IC) memory. A unique method of overlapping memory cycles has been implemented in the B 3700, to enhance the processor's response characteristics during high I/O activity periods. Due to this feature, an I/O bandpass of 3 million bytes per second is possible.

The B 3700 can accommodate up to 20 concurrent I/O channels, 10 for high speed peripheral subsystems and 10 for low speed peripheral devices. The I/O subsystem can be expanded to accommodate integrated data communication controls which will accommodate up to 36 lines.

B 4700

The B 4700 system has a higher clock rate (4 MHz) than does the B 3700. The B 4700 has the capability to handle 20 standard I/O channels, or up to 56 channels for data communication applications. The memory subsystem may contain up to one million directly addressable digits of high-speed core memory having a cycle time of 500 nanoseconds.

Both systems can utilize all medium systems peripheral devices as well as the full mix of medium systems controls, exchanges and adapters.

Power for the system, excluding peripheral devices, is derived from one central power supply, which contains all the required regulator and sensing circuitry. These supplies are internally compensated so as to reduce to a minimum any power fluctuation to the system. Several detectors are utilized to sense fault conditions such as low input voltage, dropped cycles, or input voltage that is too high. In addition, temperature sensors are employed to warn users of excessive room temperature.

Variable Field Length Floating Point (Optional)

This adapter is available for those users who require the Floating Point commands, OP 80 through 83. Installation of this adapter enables the user to perform real-arithmetic operation, with up to 100 digits of accuracy. Basically, the function of this adapter is to handle the alignment and normalization of operands as generally required in real-arithmetic operations.

B 3700/B 4700 SYSTEM CONFIGURATIONS

The configurations for the B 3700 and B 4700 systems are described in table 1, which lists various cabinets and operating characteristics. Figure 1 shows various B 4700 configurations, and figure 2 shows the typical B 3700 equipment configurations.

DATA REPRESENTATION

The basic unit of data used in the B 3700 and B 4700 processors is the bit, which represents

the presence or absence of a specific piece of information. By assigning particular values to specific bits, these bits can be combined to form larger, more efficient units of information. Because of the overall system orientation toward COBOL and the business applications by the majority of users, analysis indicated that four bits would be the optimum size of this unit of information, designated as the digit. By assigning binary weights of 1, 2, 4, and 8 to each of the bits, the combined value can represent any number from 0 through 15. The digits shown below illustrate how various bit combinations are used to represent numerical values:

8 Bit	1	0	1	1
4 Bit	0	1	1	0
2 Bit	0	0	0	1
1 Bit	1	1	0	1
	(A)	(B)	(C)	(D)

Table 1. B 3700/B 4700 Processor Styles

SYSTEM	FEATURES										
	Clock Rate	Number of Processors	Number of I/O Channels	Data Transfer Rate Per System (Bytes Per Second) (c)	Minimum Memory Per System (KB)	Maximum Memory Per System (KB)	Maximum Memory for Combined Systems (KB)	Number of Disk Controls (a)	File Protect Memory	Auxiliary Cabinet
B 3771	3MHz	1	8	3M	100	300	N/A	1	No	No
B 3772	3MHz	2	18	3M	100	300	600	2	Yes	Yes
B 4704	4MHz	1	8	2M	150	500	N/A	1	No	No
B 4711	4MHz	1	10	2M	150	500	N/A	1	(b)	(b)
B 4712	4MHz	2	18	2M	150	500	1,000	2	Yes	Yes
B 4713	4MHz	3	26	2M	150	500	1,500	3	Yes	Yes
B 4714	4MHz	4	34	2M	150	500	2,000	4	Yes	Yes
B 4708	4MHz	1	8	4M	150	500	N/A	1	No	No
B 4731	4MHz	1	10	4M	150	500	N/A	1	(b)	(b)
B 4732	4MHz	2	18	4M	150	500	1,000	2	Yes	Yes
B 4733	4MHz	3	26	4M	150	500	1,500	3	Yes	Yes
B 4734	4MHz	4	34	4M	150	500	2,000	4	Yes	Yes

NOTES

(a) Minimum.
(b) Included if multiple disk file controls.
(c) Amounts shown are millions of bytes per second per system.

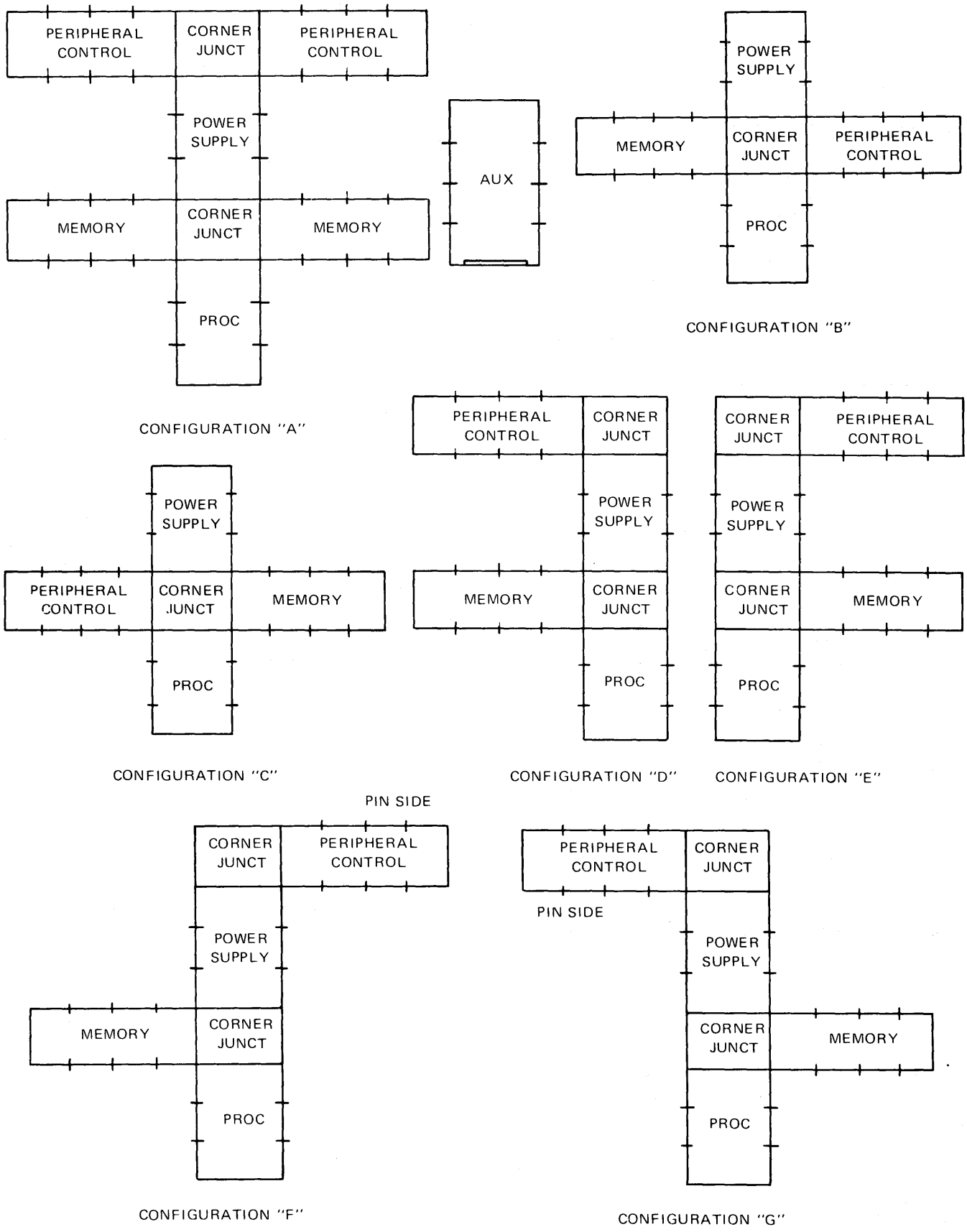
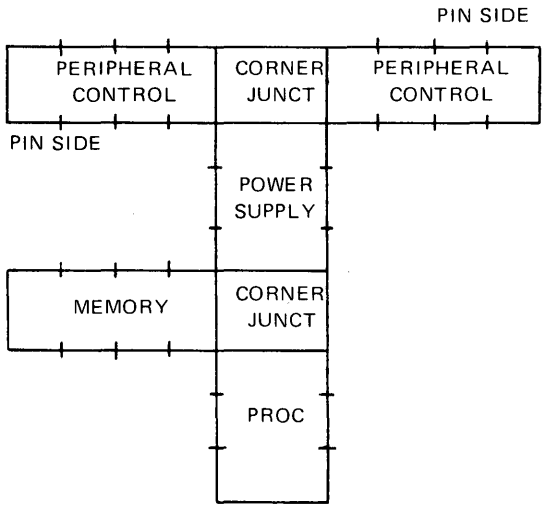
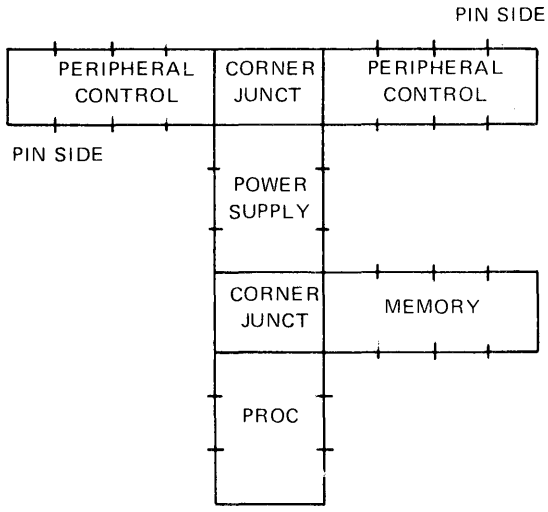


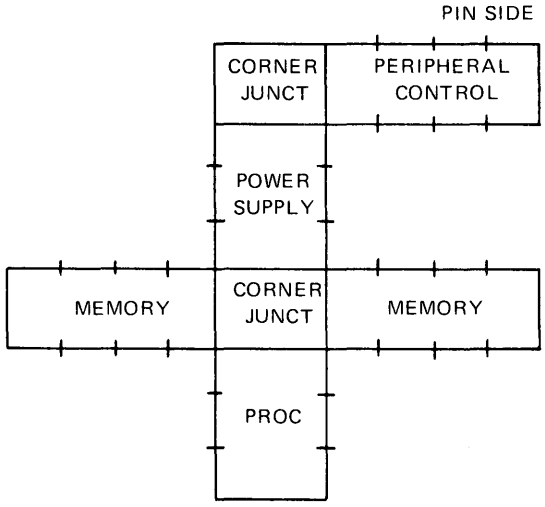
Figure 1. Typical B 4700 Configurations (Sheet 1 of 2)



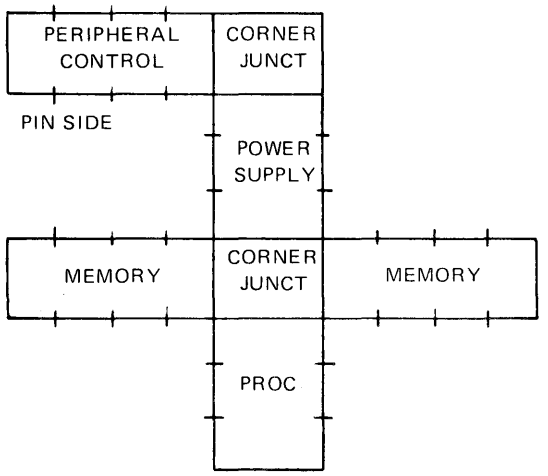
CONFIGURATION "H"



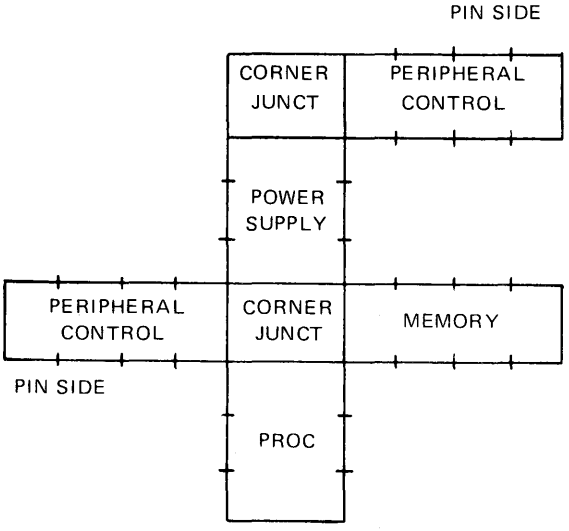
CONFIGURATION "I"



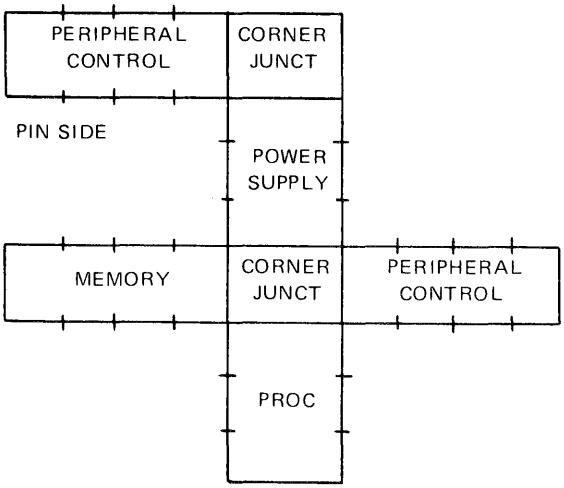
CONFIGURATION "J"



CONFIGURATION "K"



CONFIGURATION "L"



CONFIGURATION "M"

Figure 1. Typical B 4700 Configuration (Sheet 2 of 2)

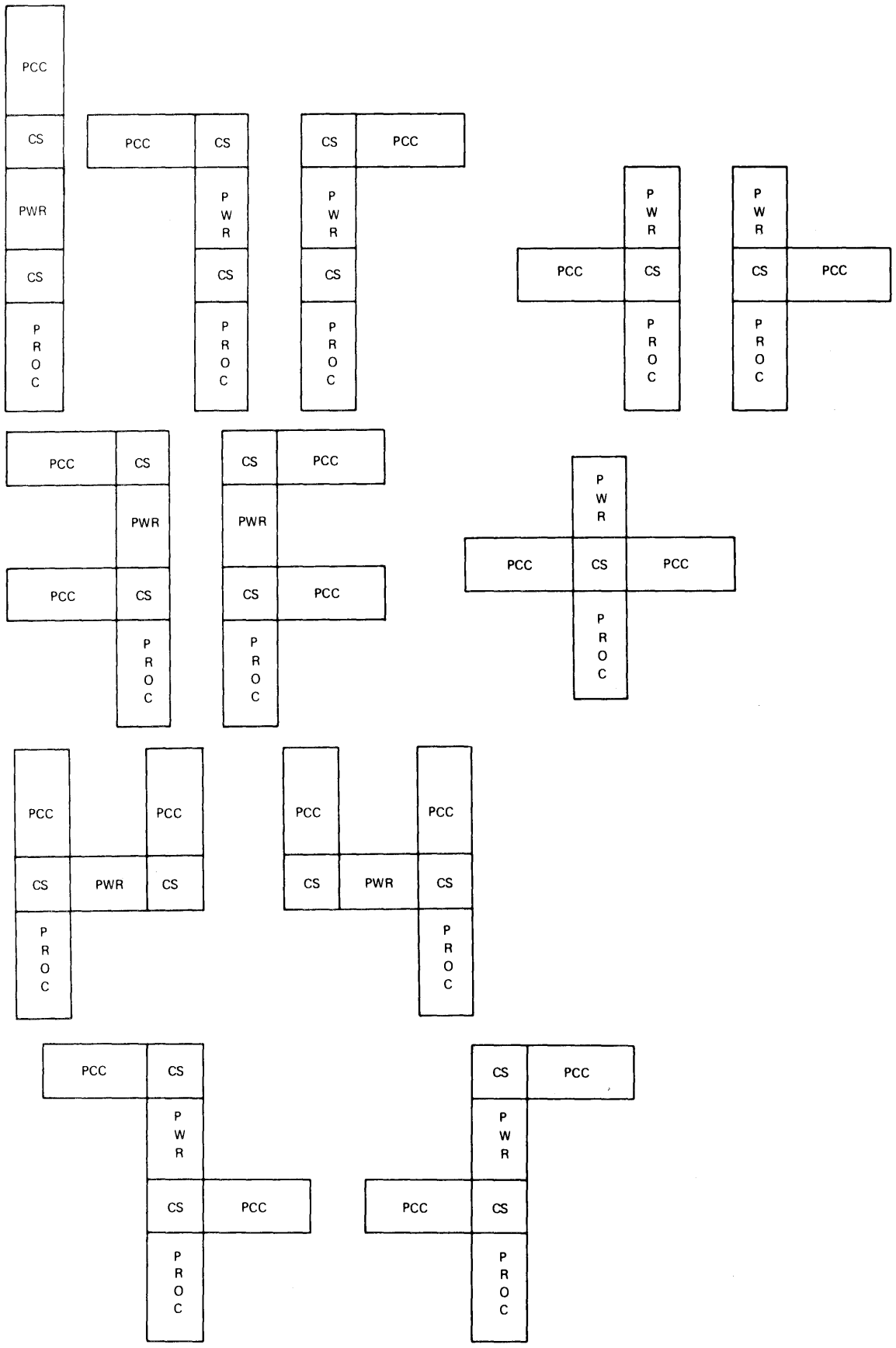


Figure 2. Typical B 3700 Configurations

In these examples, the presence of a bit is indicated by a 1 in the particular cell, and the absence of the bit is represented by a 0. In A above, both the 8-bit and the 1-bit are present, so the digit value is 9. The next digit, B, has the 4 and 1 bit present, representing a decimal 5. The last two columns, C and D, represent decimals 12 and 11, respectively, and as such cannot be represented as a normal decimal digit.

These special cases of digits having numeric values greater than 9_{10} are called undigits and are represented either by alpha characters or by a slash through the units portion of the decimal value as shown below:

$DIGIT_{10}$	REPRESENTED AS:	
	IN PRINT	ON CONSOLE DISPLAY
10	A	Ø
11	B	⁄
12	C	⁄2
13	D	⁄3
14	E	⁄4
15	F	⁄5

The digit is the basic unit addressable by the processor, and is the smallest unit of data available to the user.

As previously stated, the digit can only represent values of from 0 to 15, or 16 discrete states. Since this is not enough to represent all the possibilities of numeric digits, alphabetic characters and special symbols used in the data processing field, two digits are combined to form the unit of data called the byte, which has the ability to represent a total of 16 x 16, or 256 discrete values.

The byte, shown below, can be used to represent any alphanumeric or special character in either EBCDIC or ASCII.

8	8
4	4
2	2
1	1

A byte is often referred to as a character; however, this is not entirely accurate. The byte is the measure of a quantity of data, which may or may not represent a character; however, a character is a specific array of bits that conveys some usable intelligence.

A word, as used in B 3700 and B 4700 systems, consists of four contiguous digits, beginning and ending at specific locations in memory. As shown below, the four digits are identified as A, B, C, and D, with each bit of each digit being identified according to its binary value. Words must begin at MOD 4 addresses; that is, addresses that are evenly divisible by four.

		DIGITS			
		A	B	C	D
Bits	8	A8	B8	C8	D8
	4	A4	B4	C4	D4
	2	A2	B2	C2	D2
	1	A1	B1	C1	D1

Because many instructions use word-oriented data fields, the ability to address to a word level, and subsequently manipulate these large data blocks, rather than repetitively handling smaller units, affords a higher efficiency level and a significant increase in system throughput.

The type of data to be used is specified by the instruction as being either unsigned numeric, signed numeric, or alpha. In the case of numeric data, four-bit digits are used and, in the case

of signed numeric, a sign digit is placed before the data field. If the instruction specified alpha information, the data field is treated as groups of 8-bit bytes.

PROCESSOR INSTRUCTION FORMAT

Processor instructions may vary in length from 4 to 24 digits and have from none to three addresses as shown below:

a. OP VV

b. OP AAAA

c. OP AAAAAA

d. OP AFBF AAAAAA BBBBBB

e. OP AFBF AAAAAA BBBBBB CCCCCC

In the above example, OP indicates the operation code, VV indicates variant digits, AF and BF represent the A and B field variant digits, and A, B, and C represent the addresses of the respective data fields. In all cases, each letter represents one digit of instruction information.

Operation Code

The first two digits of any instruction always represent the operation (OP) code. This OP code is indicative of the length of the instruction and, more importantly, informs the processor of the type of action to take on the specified data.

Variant Digits

These digits, when used in instructions such as shown in the previous example, give the processor more detailed information on how to handle the specified data.

AF, BF Variants

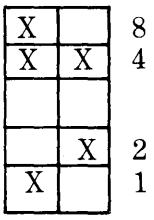
Generally, these digits specify the length of the "A" data field and the "B" data field, respectively. Since both AF and BF are 2-digit fields, each may specify a length of 00 to 99, with 00 representing a length of 100. In some instructions, AF and BF may be combined to represent lengths of from 0000 to 9999, with 0000 representing a value of 10,000. The significance of the value of the AF and BF variants is determined by the contents of the "A" field and "B" field, respectively.

Indirect Field Length

Another function available within the variant fields, when their use within an instruction specifies field length, is that of indirect field length. When the two high-order bits (8 and 4) of the most significant digit of AF or BF are ON, they indicate an indirect field length. The two low-order bits (2 and 1) plus the entire least significant digit indicate the tens and units digits of an address where the field may be found. This address must be an even number. For example:

Indirect field length is indicated.

Tens position of the address where the field can be found.



Units position of the address where the field length can be found.

NOTE

X indicates the bit is ON. Blank indicates OFF.

In this example, the variant reflects that an indirect address contains the field length of the instruction being executed and that the length of the field is to be found at base relative address 00016.

While it is logically possible to use indirect field length to any depth, it is obvious that the number of addresses generated by the technique is limited. The two low-order bits of the high-order digit may represent values of 0, 1, 2, or 3. Since the address must be even, the range may be from 00000 to 00038, base relative. This technique can be very useful in cases where many instructions in a program refer to the same field, and that field is variable in length. The field length may be modified in only one location. All instructions referring to that field will be so modified without the necessity of any other code change.

Literal

Another use of the AF variant is to indicate that the A syllable of the B 3700 and B 4700 Systems command does not contain an address, but instead contains literal data to be used directly by the command. This option is indicated by the 8-bit and the 2-bit, in the most significant digit of the AF field, being ON, with the 4-bit being OFF. A literal, as with any data, may be represented as 4-bit digits, either signed or unsigned, or as 8-bit characters. The 1-bit of the most significant digit of AF and the 8-bit of the least significant digit are used to indicate the internal representation to be used for the literal and are called controller bits which represent the following 2-bit values:

a. 0 — unsigned numeric (4-bit mode).

b. 1 — signed numeric (4-bit mode).

c. 2 — alphanumeric (8-bit mode).

d. 3 — unused.

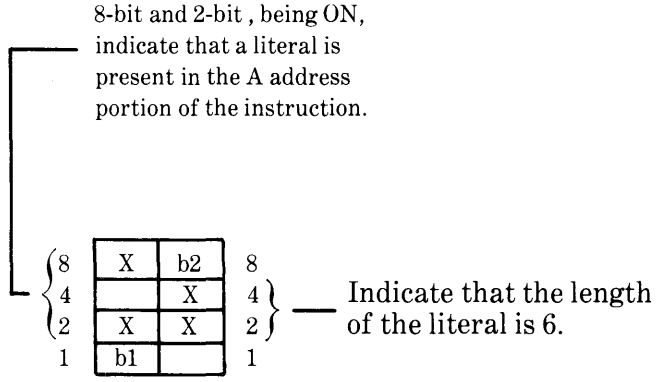
The remaining three bits of the low-order digit of AF specify the length of the literal. Since the A syllable is six digits in length, the following lengths are maximum:

a. Unsigned numeric — six 4-bit digits.

b. Signed numeric — five 4-bit digits plus a sign 4-bit digit.

c. Alphanumeric — three 8-bit (byte) characters.

For example:



In the example, b1 and b2 both being OFF indicate an unsigned numeric representation. The A syllable contains a literal of six unsigned numeric digits.

All field lengths specified for signed numeric fields reflect the length, excluding the sign. In floating-point instructions, field length reflects the length of the mantissa only.

Addresses

Typically, data field addresses have six digits, the most significant of which is used to flag such functions as indexing, indirect addressing, or extended addressing. The remaining five (or in the case of extended addressing, six) digits constitute the base relative address of the data field.

INDEXING

The eight-and four-bits of the most-significant digit of the address are used to specify indexing with any one of the three programmable index registers available to each program. The condition of these bits indicates which index register is to be used, as shown below:

8 Bit	4 Bit	
0	0	No indexing
0	1	Index Register 1 (I x 1)
1	0	Index Register 2 (I x 2)
1	1	Index Register 3 (I x 3)

If indexing is specified, the value of the index register is algebraically added to the address field prior to storing the address in address memory.

INDIRECT ADDRESSING

When the two low-order bits (2 and 1) of the most-significant digit of the address are both ON, the following five digits are considered to be the address of an address. In this case, the new address is read, checked for indexing or additional indirect addressing, and is stored in address memory only after all indirect addressing has been eliminated.

EXTENDED ADDRESS

If the second digit of an address is a binary 12 (undigit 2), the following six digits constitute an address having the capability to address full memory (1,000,000 digits). Figure 3 illustrates the extended address and its flag compared to a normal unextended address.

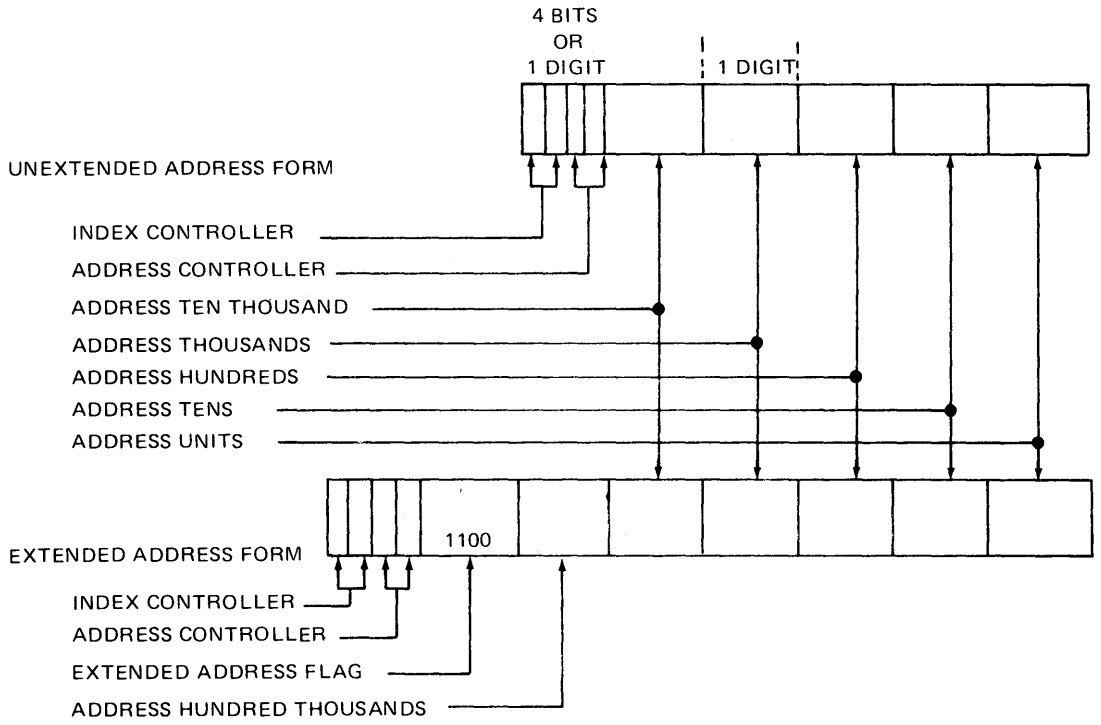


Figure 3. Extended Address

All capabilities assigned to normal addresses, including indexing, indirect addressing, and literal capability, apply to extended addresses.

INTERRUPTS

In order to make most efficient use of available system resources, an interrupt scheme had to be designed into the B 3700/B 4700 systems. As a result, the processor has been freed from time-consuming monitoring tasks and allowed to function with as little interference as possible. When some event does occur that requires processor action, the Interrupt flag is set, and the operating system examines the cause of the interrupt and determines the action to be taken.

Interrupts may occur while the processor is in either normal or control state. Their handling is somewhat different under the different conditions.

Control State Interrupts

When the processor is operating in control state, there may be interrupt conditions generated, but no automatic interrupt handling takes place at the completion of an instruction. Rather, at the time that the MCP attempts to return to normal state, automatic branching to control location 000094 will occur so that these interrupts will be handled by the MCP.

If the processor is operating in the zero-base state, the following processor conditions will cause the processor to branch to control location 000094:

a. Memory parity error interrupt.

b. Address error interrupt.

c. Instruction time out interrupt.

d. All non-assigned operator codes.

e. Operator codes requiring options which are not present.

f. Invalid halts.

Since the processor is in the zero-base state, any of the above interrupts indicates that a hardware or an operating system error exists. A privileged instruction interrupt cannot occur because all of the instructions are executable in the zero-base state. The clock interrupt condition occurs in the same manner as in normal state, but no automatic branching occurs in control state.

Normal State Interrupts

When certain operational conditions occur within the processor while executing instructions in the normal state, the following logic occurs:

a. If either an operator interrupt, a timer interrupt, or an I/O interrupt occurs, the interrupt will generate a result descriptor and store the descriptor in the reserved memory location.

b. The interrupt will set the interrupt flag and store the program return point and logical register settings.

c. An automatic branch is taken to the address specified by the contents of reserved memory location 000094. The processor operational mode is changed from normal state to control state, zero base.

The branch to the control state is a branch to the operating system — the Master Control Program. It is the function of the operating system to determine the type of interrupt and the course of action that is to be followed. The interrupt system, as explained, is automatic and is an integral part of the processor hardware system.

LOGICAL UNITS

Functionally, the processor may be divided into four broad areas: a logical or data section, an arithmetic section, a control section, and an addressing section. Each of these areas is subdivided into smaller, more specialized functional modules, and each is intimately associated with the other sections.

Logic Section

The first of these major areas, the logic section, consists of the data comparison circuitry, the overflow flag, several data registers and the busses required to transfer data throughout the system.

COMPARISON FLIP-FLOPS

Two hardware flip-flops make up the comparison logical unit. These two flip-flops have the following four combinations (0 = OFF, 1 = ON):

a. 00 — cleared.

b. 01 — greater or high comparison result.

c. 10 — less or low comparison result.

d. 11 — zero or equal comparison result.

These four states represent the result of executing an instruction that affects these logical units. The cleared state indicates that there is no comparison result existing. The instructions that set a result into the comparison flip-flops, which are displayed on the appropriate console indicators, are:

a. All arithmetic instructions.

b. All floating point instructions.

c. Compare instructions.

d. Move Numeric instruction.

e. Move Alphanumeric instruction.

f. The bit test instructions.

g. The logical instructions.

h. The scan instructions.

i. The Edit instruction.

j. The Scan Result Descriptor instruction.

k. Search instructions.

Any branching that is done on the basis of the comparison flip-flops does not change their status. Only another instruction that affects them can change their status. When entering control state by means of the Branch Communicate instruction or the automatic interrupt system, the status of the comparison flip-flops is stored. The 2-bit and 1-bit of the same character that stores the overflow flip-flop status will contain the status of the comparison flip-flops. The flip-flops are then cleared before branching to control state. When return is made to the normal state, the Branch Reinstate instruction will restore the comparison flip-flops from the character in reserved memory. Similarly, when entering a subroutine, the Enter instruction stores the comparison flip-flops in the 2-bit and 1-bit of the specified character in the memory stack. This same character contains the overflow flip-flops status. When leaving the subroutine with the Exit instruction, the comparison flip-flops are restored from this character in the stack.

OVERFLOW FLIP-FLOP

The overflow flip-flop is a hardware logical unit that indicates the data field of a move or arithmetic instruction is exceeded. If an overflow condition is detected, the command is executed but the data is not affected. The overflow flip-flop is not cleared at the beginning of an arithmetic operation, therefore, it indicates overflow that has occurred any time before or during a series of arithmetic operations or other interjected nonarithmetic operations. Instructions that can create an overflow condition are:

a. Arithmetic instructions except Multiply.

b. Floating point arithmetic instructions.

c. Move Numeric.

d. Move Alphanumeric.

e. Fixed Length Arithmetic Instructions.

Overflow cannot occur during a Multiply instruction since the receiving field is always large enough to contain the product. In all cases except the floating point instructions, overflow results when the receiving field cannot contain the sending field. With floating point instructions, the overflow can also be caused by an out-of-range exponent. The overflow flip-flop is turned OFF by the Branch On Overflow instruction. Once cleared, it can be turned ON if the conditions arise while executing those instructions that may turn it ON. There are two ways by which the present setting of the overflow flip-flop may be stored in the 4-bit of reserved location 000076, after which the flip-flop is cleared. These are:

a. Branch Communicate.

b. Automatic interrupt feature.

The overflow flip-flop is restored from the reserved memory location by the Branch Reinstate instruction. In addition, the object program may branch to some subroutine and the overflow flip-flop setting must be retained and restored at the conclusion of the subroutine execution, prior to continuing in the object program. The Enter instruction will store the flip-flop setting into the four bits of the reserved character location in the memory stack, and then clear the overflow flip-flop. The Exit instruction will restore the overflow flip-flop setting from this reserved location.

DATA REGISTERS

Both the B 3700 and the B 4700 make extensive use of internal data registers so as to allow a constant flow of fully buffered data, thus eliminating adverse effects to system performance. Various instructions use these buffer registers to hold arithmetic operands, data read from or to be written into memory, or data upon which comparisons are to be made. When used in arithmetic operations, some of these data buffers have the ability to automatically form the complement of the number they contain, thus reducing the arithmetic cycle times.

BUSSING

In order to rapidly and efficiently move data from point to point within the processor, all registers and buffers must have some sort of interconnection scheme. By using a polynodal bussing network and the appropriate control signals, it is possible to transfer data into and out of registers without error or loss, all within a few billionths of a second. This concept is used extensively throughout the B 3700 and B 4700 processors. The speed of data transfers using this bussing technique is such that in many instances the only delay is that imposed by the propagation time of an electrical impulse through the system interconnections. Dependent upon the OP code, the processor generates the proper control signals to switch data from a selected source onto the busses, and then into the destination register.

Arithmetic Section

This area of the processor consists of the variable field length arithmetic unit or adder, the fixed field length arithmetic unit, and the floating point adapter. The Floating Point adapter is optional with either system. Variable field length arithmetic capability is standard on both processors.

ADDER

The processor uses an adder that accumulates two fields from the most significant to the least significant digit positions. Reverse addition, as incorporated in B 3700 and B 4700 systems, has the advantage of detecting an overflow condition prior to altering the receiving field for the result. The principle used in this type of adder is illustrated by the flow chart in figure 4 and the five examples in figure 5. If the data fields are signed, sign manipulation takes place prior to the addition since they are the most significant digits.

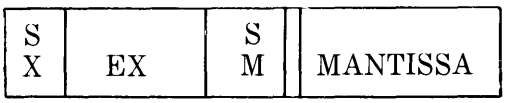
In figure 5, there is no manipulation of signs shown. The addition of two data fields is only shown to present the technique of a left-to-right adder.

In example 1, as each set of digits is added, no carry is generated and no nines are produced. As each new result is generated by the adder, the previous result is stored in the result field. In example 2, an overflow condition is immediately detected because of a carry on the first digit addition. In example 3, the first five nines are not stored until the result equal to eight is generated. The eight is retained until the carry is generated when adding the last digits, at which time the eight is made into a nine and stored. The following nines are then stored as zeros in the receiving field, and then the final digit (8) is stored. Nothing is stored in example 4 until the final digits are added. The receiving field in example 5 remains unchanged even though an overflow condition is not detected until the final digits are added. This is due to the result being contained in the nine's counter of the adder.

FLOATING POINT ARITHMETIC

In order to represent very large or very small numbers using decimal notation, a great many digits must be used. To simplify calculations, "powers of ten," or standard notation, has long been used. For example, the decimal value of 1,000,000 could be shown in standard notation as 1×10^9. Numbers in this format can be manipulated using normal algebraic procedures far more simply than if they were shown in decimal format.

Floating point arithmetic slightly modifies this standard notation to a format more usable in computers. In floating point format, the number 1,000,000 becomes +09+000001 when the field length specifies six digits. The general format for any floating point number is shown below,



Where SX is the one-digit sign of the exponent, EX is the two-digit exponent, and SM is the one-digit mantissa sign. The mantissa may be from 1 to 100 digits in length, as specified by the appropriate field variant (AF or BF). Both the exponent and the mantissa are integers.

Floating point arithmetic has some restrictions concerning zeros, their representation and use. The only correct way to represent a zero is with an exponent of -99 and a mantissa of all zeros,

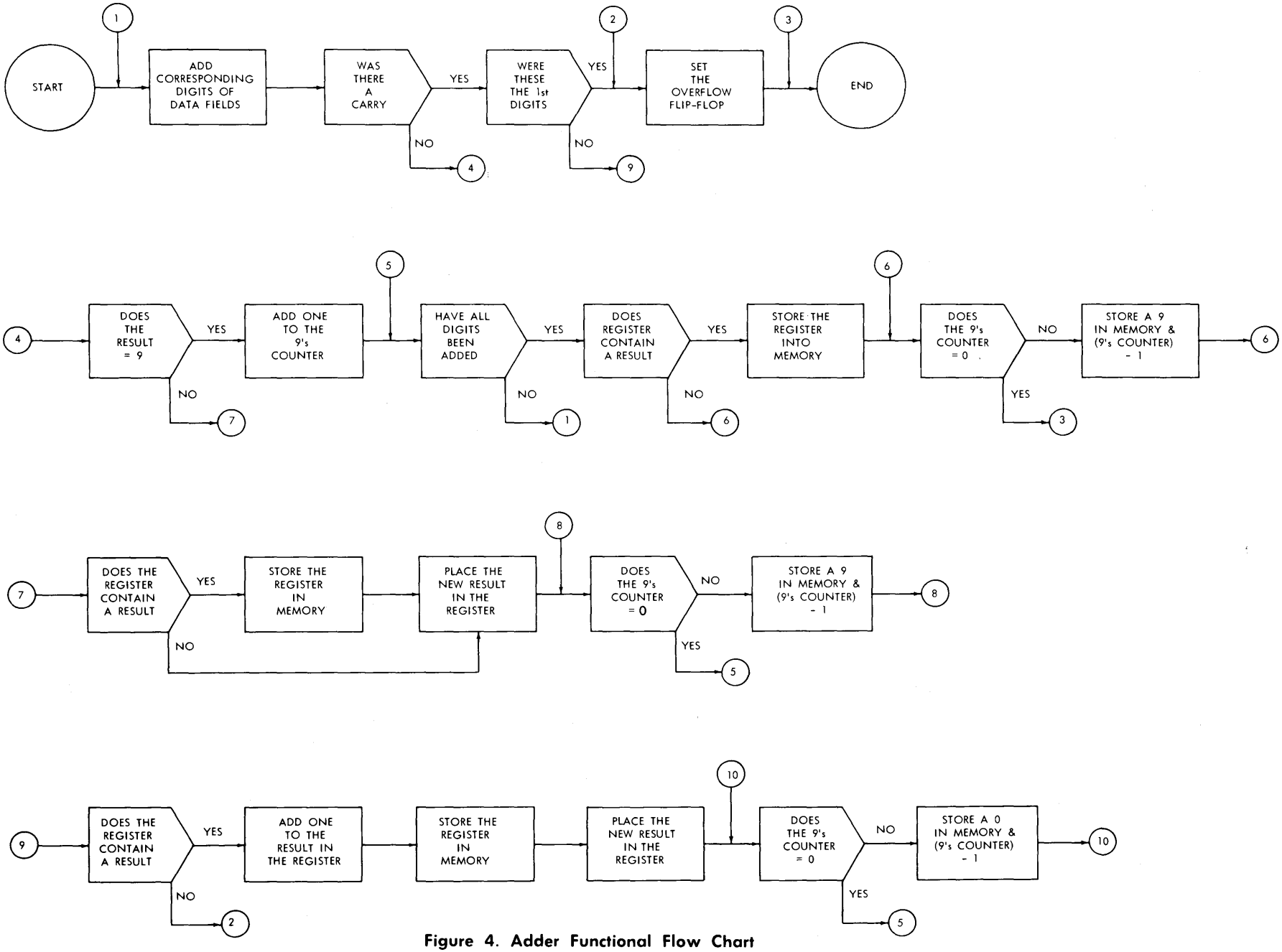


Figure 4. Adder Functional Flow Chart

EXAMPLE 1:

```
        12343
        12343
RESULT  24686
```

EXAMPLE 2:

```
            12345
            92345
RESULT 1:   0 + CARRY
FINAL RESULT: OVERFLOW FLIP-FLOP SET
```

EXAMPLE 3:

```
            876543229
            123455779
RESULT 1:   999998998 + CARRY
FINAL RESULT: 999999008
```

EXAMPLE 4:

```
            0876543
            0123457
RESULT 1:   0999990 + CARRY
FINAL RESULT: 1000000
```

EXAMPLE 5:

```
            876543
            123457
RESULT 1:   999990 + CARRY
FINAL RESULT: OVERFLOW FLIP-FLOP SET
```

Figure 5. Adder Examples

since this is the smallest value the notation can represent. Leading zeros in non-zero mantissas are acceptable in addition or subtraction; however, in multiplication or division they will cause the entire mantissa to be treated as a zero and yield erroneous results.

FLOATING POINT ADAPTER (OPTIONAL)

The purpose of the floating point adapter is to manipulate the exponents and mantissas so that the numbers are aligned properly. This means that the exponent must either be decremented and the mantissa multiplied by 10 for each decrement of the exponent, or the exponent incremented and the mantissa divided by 10 for each time the exponent is incremented. The net effect of these manipulations does not change the value of the number, but it does position the digits properly so that they may be added or subtracted.

In addition to the alignment function for addition and subtraction, this adapter also performs a normalizing function for multiplication and division.

Normalization (elimination of leading zeros) is accomplished by examining the leading digit of the product or quotient and, if zero, multiplying the result field by 10 with a corresponding decrement of the exponent.

FIXED LENGTH ARITHMETIC UNIT

This portion of the processor makes use of several innovative techniques in arithmetic data manipulation; these include word-wide transfers into and out of the various arithmetic registers, multiple shift capabilities, automatic overflow compensation, and a full word multiply capability. This unit consists of several major elements and their associated controls and interconnections as described below and illustrated in figure 6.

ACCUMULATOR, EXPONENT REGISTER, EXTENSION REGISTER. These three elements work in conjunction to hold the complete arithmetic operand. The Exponent Register (SX) contains the exponent sign (XS), the exponent (XA), and the mantissa sign (MS) of the operand contained in the accumulator. Typically, the accumulator holds the mantissa or some intermediate stage of the arithmetic operation in progress which will form the mantissa at the completion of the operation. The Extension Register (E), is used to extend the accuracy of the accumulator one additional digit or, in some cases, to reduce the chance of arithmetic overflow prior to normalization.

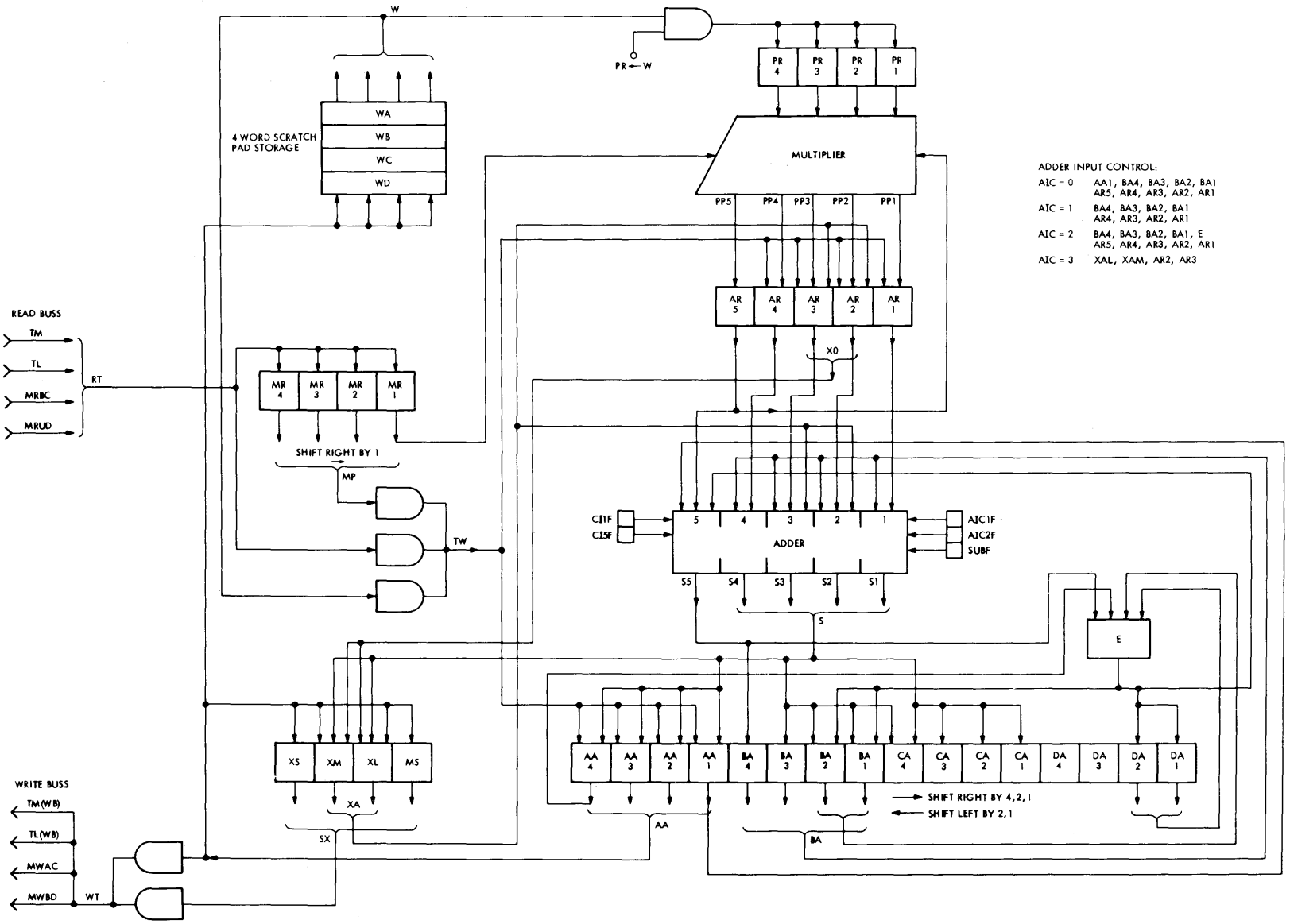


Figure 6. Fixed Length Arithmetic Unit

ADDER. The adder used in the fixed length arithmetic unit provides the capability to add or subtract two groups of five digits each and provides a decimally corrected result. As shown in figure 7, the adder consists of four major areas: input gating, hexadecimal adder, carry logic, and decimal correction logic.

The input gating logic selects the two registers which are to be added or subtracted. In most cases, the inputs will be the B word of the accumulator and the AR register. Other combinations may call for the inclusion of the E register or perhaps the addition of the operand exponents.

The hexadecimal adder and its associated carry logic work together to provide parallel five-digit addition. Since these are decimally oriented systems, hexadecimal results must be converted to decimal formats. The carry logic partially provides for this conversion by handling any carry from sums greater than nine and propagating carries through intermediate carry generating sums.

Decimal correction logic is used when the sum of an addition is greater than nine. As shown below, then 6 is added to the hexadecimal sum, the final sum is the correct decimal value:

```
 9
+8
--
11  <-- hexadecimal sum
+6
--
17  <-- corrected decimal sum
```

The decimal correction logic is, in effect, a second stage of the adder which forces the addition of a 6 to the hexadecimal result.

SCRATCHPAD MEMORY. Four words of high-speed scratchpad memory are provided for temporary operand storage during the execution of the instruction. Data may be shifted from the accumulator into any of the four words, or from the scratchpad to one of the arithmetic registers.

ARITHMETIC REGISTER. This register, designated AR in figure 6, is most commonly used as one of the inputs to the five-digit adder. It is a full five-digit register and may receive data from the multiplier, the MR register, or directly from memory. This register has no shift capability.

MULTIPLIER. The multiplier makes use of two registers, the four-digit PR and the shiftable four-digit MR, along with read-only memory look-up tables and adder circuitry. The basic algorithm for the multiply operation is as follows:

1. Multiply all four digits of PR by the least-significant digit of MR. This is done by using the digit values in PR and MR as addresses in the ROM look-up tables.

2. Add the 4 two-digit products to form 1 five-digit partial product.

3. Shift MR right one digit.

4. Multiply, add, and shift as in 1, 2, and 3 until all digits have been multiplied.

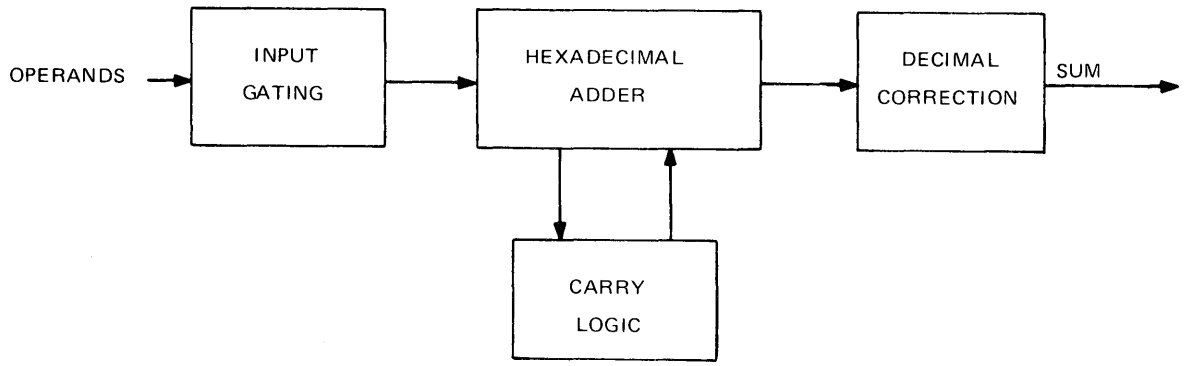


Figure 7. Fixed Length Adder

5. After each step 2, the partial products must be added, thus building the final product.

Control Section

This section of the processor is used to control and monitor internal system operation and to generate information for use by the operating system regarding processor status. Included in these control elements are the Normal flip-flop, the Interrupt flip-flop, result descriptors, Mode flip-flop, and the Timer.

The operation of the B 3700/B 4700 systems is controlled by the MCP. This control program, which resides in memory starting at address zero, performs all of its functions with the base register equal to zero and the processor in control state. With the base register equal to zero the following operational conditions exist:

a. Privileged instructions are valid.

b. Such processor errors as a memory parity error, memory address error, invalid instruction, and instruction time-out will cause the operation of the system to be stopped.

When in control state, the operation of the processor is not sensitive to the setting of the Interrupt flip-flop. An interrupt will be ignored until it is programmatically sensed by the control program.

All object programs are loaded into memory and initiated by the control program. At the time an object program is loaded into memory, it is allocated an area of memory by the control program. Each time an object program is initiated, the base and limit registers are set to bound the area of memory that was allocated to the program (see figure 8).

Also, the processor is put into what is referred to as normal state. Under these conditions (Base $\neq$ 0 and normal state) the following operational features exist:

a. Except for certain special conditions, all processor accesses to memory are restricted to the program's allocated area of memory.

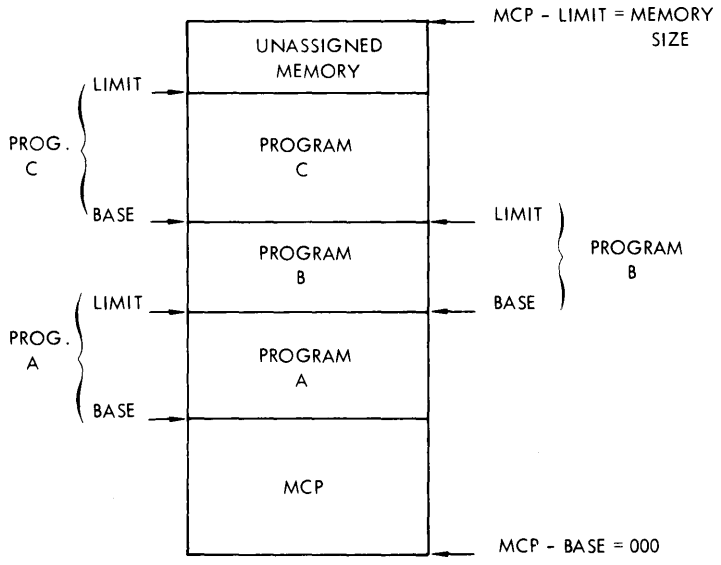


Figure 8. Memory Allocation

b. All privileged instructions are invalid, due to Base $\neq$ 0.

c. With the Base $\neq$ 0, processor errors will result in storing a processor result descriptor and an immediate branch to the control program.

d. Because the processor is in normal state, the operation of the processor is sensitive to the Interrupt flip-flop. If set, then the processor forces a branch to the control program.

Item a above indicates that under certain conditions the processor can access areas of memory that are outside of the Base/Limit register settings. These conditions exist only when it is necessary to access certain areas of the control program's reserved memory. The following list indicates these conditions:

a. Result descriptor storage, when it is necessary to store a processor result descriptor into locations 80 through 83.

b. NI (next instruction address) storage by all multiply and divide instructions into locations 88 through 93.

c. The accessing of the Halt Execution digit in location 77 by the Halt instructions.

d. The storage of pertinent information into locations 64 through 76 when the object program branches to the control program.

INTERRUPTS

The interrupt feature of the system is used to indicate to the control program when it must respond to the systems needs. Interrupts can indicate either normal operating procedures or error conditions that may occur.

Interrupts can be separated into two types: processor interrupts and I/O interrupts. Processor interrupts are a result of some action initiated by the processor. I/O interrupts are those that result from the completion of any I/O operation.

There is one interrupt flip-flop, located in the processor, that is used to indicate when any interrupt condition exists. In order to determine the type of interrupt, processor or I/O, a result descriptor is stored every time an interrupt condition exists. With every I/O result descriptor (R/D) written into memory the interrupt flip-flop is set.

RESULT DESCRIPTORS

The processor and each I/O channel have a specific area of memory in which to store its R/D. These areas are located within the control program's area of reserved memory. The processor R/D is always stored in the four digits starting at memory location 80.

The I/O R/D's are stored in memory starting at location 100 with a R/D located every 20th memory address. For example; channel 00 R/D is stored in 100, channel 01 R/D is stored in 120, channel 02 is stored in 140, etc.

To determine the location of the R/D for any I/O channel, multiply the channel number by 20, then add 100, (Ch#x 20 + 100=R/D location). For example: the R/D location for I/O channel six can be determined by:

$$(6 \times 20) + 100 = 220$$

(6 = Ch#; 220 = Location of Ch #6 R/D)

The result descriptors are used to describe some operation. Each R/D is 16 bits in length, located at a Mod 4 address (80, 100, etc.), therefore taking up one memory word. Referencing figure 9, the two most significant bits of the R/D are the control bits that indicate the status of the result descriptor in memory. The most significant bit of the result descriptor, bit number 1, is referred to as the operation complete bit. Each time a result descriptor is written into memory, the operation complete bit is set to indicate to the control program that the R/D location contains a valid result descriptor.

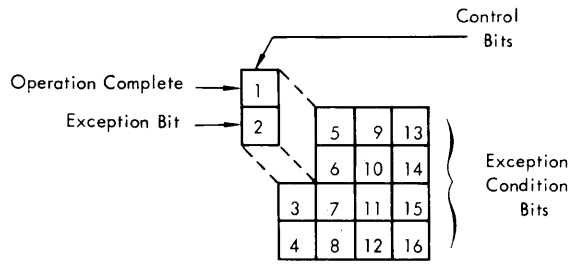


Figure 9. R/D Format

The second bit is referred to as the Exception bit. If set, it indicates that some exception condition is indicated by the remaining bits of the descriptor, 3 through 16.

When a result descriptor is written into its respective location in reserved memory of the control program, the two most significant bits are set to indicate the following:

BITS	INDICATION
1 * $\overline{2}$	Operation complete, with no exception condition.
1 * 2	Both control bits ON indicate a R/D with an exception condition being described with the R/D.

The exception conditions described are those conditions which may occur as a result of some operation.

PROCESSOR RESULT DESCRIPTOR

The processor result descriptor bits and the conditions described are shown in figure 10. Bits 4 through 8 describe error conditions which can occur during the operation of the processor.

Bits 9 and 10 are not considered error conditions. Each exception condition is described below.

R/D BIT	CONDITION DESCRIBED
1	OPERATION COMPLETE
2	EXCEPTION BIT
3	N/A
4	INVALID I/O
5	INVALID INSTRUCTION
6	MEMORY PARITY ERROR
7	MEMORY ADDRESS ERROR
8	INSTRUCTION TIME-OUT
9	TIMER
10	OPERATOR
11 ⇒ 16	N/A

Figure 10. Processor Result Descriptor

INVALID I/O. The Invalid I/O is indicated by bit 4. This type of an error indicates that the control program attempted to initiate an input or an output operation that was invalid. An invalid I/O is the result of the following:

a. An attempt to initiate an I/O operation on a channel that does not contain a control; that is, an unused or nonexistent channel.

b. An attempt to initiate an I/O Descriptor (instruction for the I/O control) that is not valid for this type of control;i.e., card read instruction to a magnetic tape control, etc.

c. An attempt to initiate an I/O channel that is busy; that is, it is presently performing an I/O operation.

d. The addresses in the I/O Descriptor are invalid.

When an invalid I/O is detected, the processor will set the Interrupt flip-flop and store a result descriptor with bits 1, 2, and 4 set.

INVALID INSTRUCTION. An invalid instruction error is detected while the processor is in a fetch cycle. This error is due to the following:

a. An attempt to fetch a non-existent OP code. This also includes optional instructions (floating point instructions) when the option is not present.

b. An attempt to fetch a privileged instruction with the base register unequal to zero.

c. A Halt instruction is executed when the Halt Execution Digit in absolute address 77 (reserved memory of the control program) indicates that halts are invalid.

d. Attempting to Branch Communicate to an invalid location in the control program.

An invalid instruction error will result in the storage of a processor result descriptor with bits 1, 2, and 5 set.

MEMORY PARITY ERROR. All accesses to memory will check the information in the memory word accessed for correct parity. If a parity error exists and the memory access was initiated by the processor, a processor R/D is stored with bits 1, 2, and 6 set.

ADDRESS ERROR. An address error will result in a processor R/D with bits 1, 2, and 7 set. This type of an error is caused by the following:

a. Base/Limit error — due to an attempt by the processor to access an area of the memory that is outside of the area specified by the base and limit registers; i.e., an address that is less than the base or that is equal to or greater than the limit register.

b. Non-synced addresses — odd addresses when even addresses are specified, or mod 2 addresses when mod 4 addresses are specified.

INSTRUCTION TIME-OUT. The processor allows itself 250 ms to perform either a fetch or an execute cycle. If it takes longer than 250 ms,

it is assumed that the fetch or execute will not properly terminate itself; therefore, a processor R/D is stored with bits 1, 2, and 8 set.

TIMER. The timer interrupt, which is not considered an error, is caused by word G of address memory, the first timer word, being equal to or greater than the second timer word, H. This results in a processor R/D stored with bits 1, 2, and 9 set and the interrupt flip-flop set.

The frequency at which the timer interrupt will occur is determined by the control program. Normally, the MCP will set address memory word H to a value that will give a timer interrupt once every second.

OPERATOR INTERRUPT. The operator interrupt will set the interrupt flip-flop and store a R/D with bits 1, 2, and 10 set. This interrupt is caused by the operator pressing the OI key on the console to indicate to the control program that operator intervention is desired.

NORMAL/CONTROL STATE OPERATION

The processor operates in one of four states, with the Base register equal to or unequal to zero:

1. Base = 0 & control state
2. Base ≠ 0 & normal state
3. Base = 0 & normal state
4. Base ≠ 0 & control state

The control program functions with the Base = 0 and control state. Under these conditions the privileged instructions are valid and the operation of the processor is not affected by the interrupt flip-flop. If an interrupt occurs, it is detected programmatically by the control program.

Object programs are initiated by the control program. They operate with the processor in normal state with the base register unequal to zero. If an interrupt should occur, the processor completes the instruction presently being

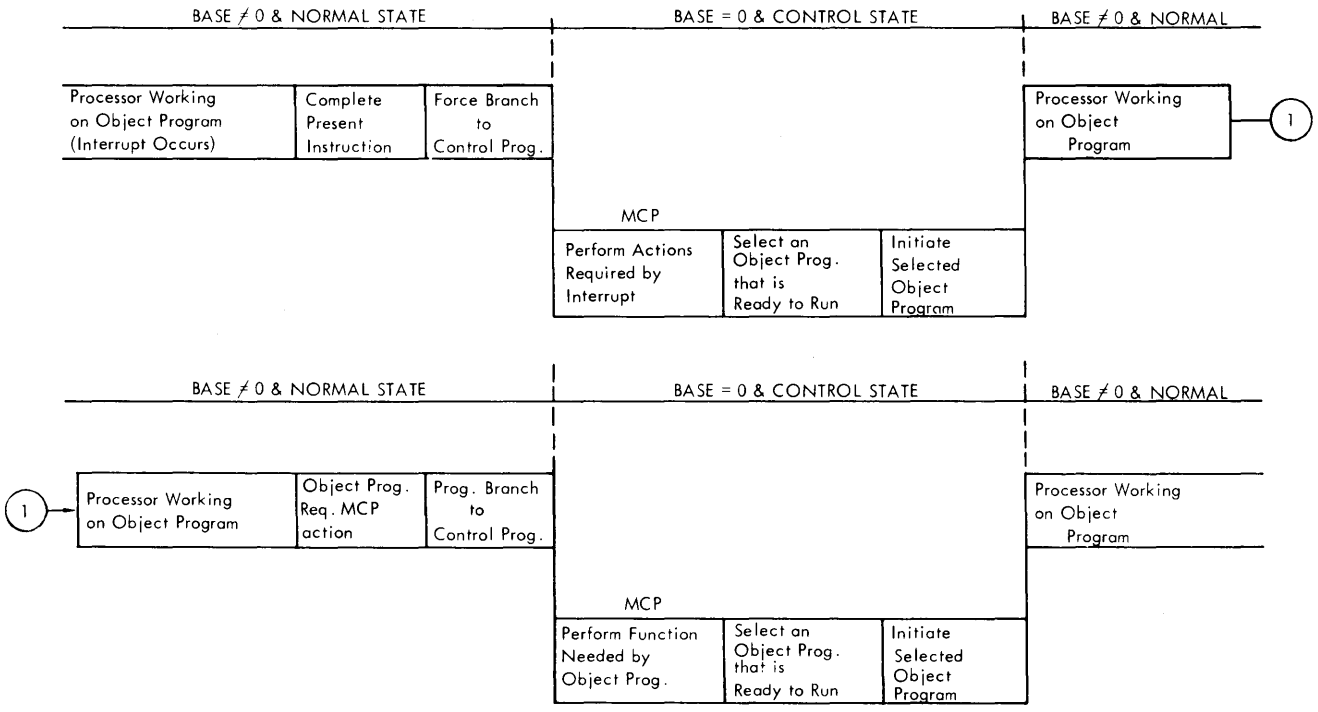


Figure 11. Operational Block Diagram

executed, then branches to the control program (see figure 11).

The control program will perform the necessary actions required by the interrupt and any other programmatic housekeeping chores that are needed at this time.

It will then select an object program that is ready to run and initiate the object program on the processor in normal state.

An object program may require that certain functions be performed by the control program, such as initiation of I/O operations. In this case, the object program will take a branch to a predetermined location in the control program, allowing the appropriate actions to be taken for performance of the indicated function.

After performing the necessary function, the control program will select an object program that is ready to be run and initiate this object program in normal state.

In order to facilitate proper communication between the control program and the object program, the processor makes use of two instructions and an area of the control program's reserved memory. The instructions used are the Branch Reinstate (BRE) and the Branch Communicate (BCT). The reserved memory utilized is located in absolute address 64 through 76 of the control program.

The reserved memory portion utilized is shown in figure 12. This area will contain the absolute address of the next instruction to be executed in the object program (aaaaaa), in locations 64 through 69. The object program's base (BBB) and limit (LLL) register settings are contained in locations 70 through 72 and 73 through 75, respectively. In location 76 is the COM information to be set into the processor when an object program is initiated. The COM digit in 76 consists of:

8 bit — ASCF (Mode FF)

4 bit — OVF (Overflow FF)

2 bit — COMLF

1 bit — COMHF

Prior to the initiation of an object program, the control program must put into locations 64 through 76 of reserved memory the information that pertains to the program being initiated.

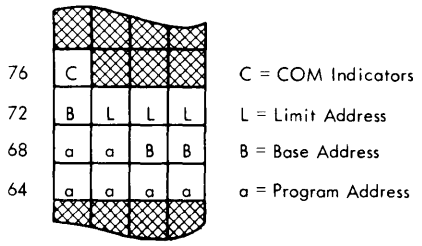


Figure 12. Interrupt and BRE Reserved Memory

OP REGISTER

This two-digit register holds the OP code of the instruction presently being executed and is used in conjunction with the logic and sequence counters to develop the micro-operation necessary to execute the instruction.

LOGIC COUNTER-SEQUENCE COUNTER

These two counters, each having the ability to count from 0 through 15 working in conjunction with the OP code of the instruction being executed, are used to generate the micro-operators required to complete the execution of the instruction.

Addressing Section

The addressing scheme used in the B 3700 and B 4700 systems is designed to yield optimum performance with the MCP's data-handling techniques. The ability to index or indirect address previously discussed, along with the data protection offered by the use of base and limit registers, enables the MCP to more efficiently use the memory available.

Prior to describing how the MCP makes use of the hardware, some of the addressing registers will be described.

ADDRESS REGISTER

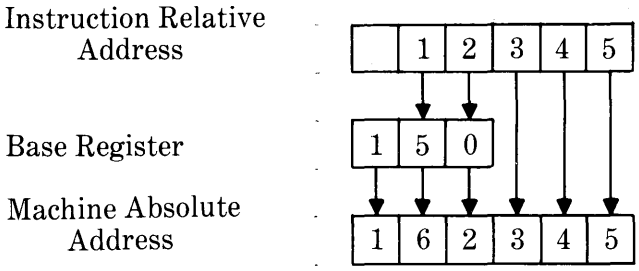
This register will contain the six-digit absolute address of the data or instruction in memory. Typically, this address is assembled during the fetch of the instruction and is placed into the address register at the end of the fetch. Associated with the address register is circuitry to enable address comparison and modification. Addresses, either for the processor or I/O devices, are placed into address memory by way of the address register.

BASE-LIMIT REGISTERS

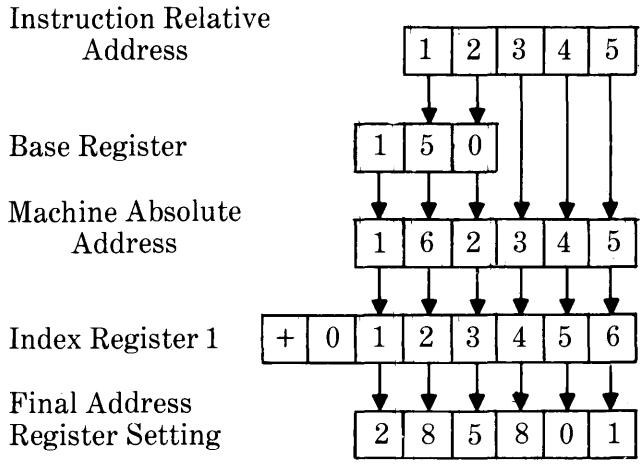
These two 3-digit registers delineate the boundaries of a program in memory. When a program is loaded, the MCP automatically assigns a base and limit address, which, when added to the relative addresses within the program, indicates the absolute addresses involved with that program. The MCP may reassign base and limit values to move programs around in memory in order to create a large area of unassigned memory from several smaller areas.

ADDRESSING TECHNIQUES

Addresses within all B 3700 and B 4700 systems object programs are compiled and executed as being base relative to zero. This means that each program is created and executed with the assumption of zero being the object program's beginning point. When a program is assembled or compiled, it is assigned a five-digit relative address, starting at zero and continuing upward as far as necessary, depending on the size of the program. During execution of an instruction, each address contained in the instruction is automatically incremented by the three-digit base register, at no cost in execution time. The combination of the base register and the relative addresses creates a machine absolute location. This is a hardware capability and allows the MCP to assign a program to any contiguous area of memory large enough to contain the program, and to subsequently relocate the program when necessary, by merely changing the base register. For example, assume the MCP loads a program starting at location 150000; the setting of the base register would be 150 as shown below:



Assume further that the program has indicated that an instruction is to be indexed by the contents of index register 1. The final absolute location would be derived as shown below:



Consider a system having 150,000 bytes of memory and using an MCP employing 25,000 bytes. It is evident that there are 125,000 bytes left for user jobs.

Supposing that the user wishes to have the following jobs running.

JOB #1	37K BYTES
JOB #2	12K BYTES
JOB #3	8K BYTES

JOB #4	14K BYTES
JOB #5	32K BYTES
JOB #6	26K BYTES

It is evident that Job #6 cannot be run, since Jobs 1 through 5 require 103K bytes, leaving only 22K available. However, when Job #3 is completed, the MCP will reassign the base and limit addresses of Jobs 4 and 5, so as to push them down in memory and generate 30K of contiguous memory. Job #6 is then called in and execution begins.

ADDRESS MEMORY

Address memory is a group of eight 6-digit registers used primarily to hold preassembled addresses. Each six-digit register is called an address memory word and serves a specific function in processor operation. By preassembling the addresses, and storing them beforehand, the processor is able to execute instructions in far less time than otherwise would be possible.

These eight words are designated A through H and serve the following functions:

A. Contain the A address of the instruction in the OP register.

B. Contain the B address of the instruction in the OP register.

C. Contain the C address of the instruction in the OP register. The C word may also be used for data during some arithmetic operations.

D. Contain the address of the next instruction to be executed.

E. General purpose; used for temporary storage of addresses and data during processor operation.

F. Contain the address of the instruction in the OP register.

G. Contain the first timer word which is counted up at a 1KHz rate.

H. Contain the second timer word, which is programmably set. The G and H words are compared and, when equal, cause a timer interrupt.

I/O ADDRESS MEMORY

In order to facilitate I/O operations, the assembled begin and end addresses for the I/O data are stored in address memory similar to that used by the processor. Since each channel requires a full word for both the begin and end addresses, two word locations are reserved for each channel.

I/O ADDRESS REGISTER

Outputs from, and inputs to, the I/O address memory are handled by the I/O address register. By using a separate register for I/O activity, the processor's address register is freed to perform other functions during periods of high I/O activity.

Memory Control Section

The memory control section is used by the processor as an interface to the memory subsystem and, as such, has the circuitry required to decode both data and addresses, generate and check parity, correct errors, and control the timing signals sent to the memory.

When a request for a memory write is made, memory control decodes the address to select the proper area in memory. At the same time, the timing generators are enabled to gate the decoded address and data to their proper locations. As the data to be written goes through memory control, the bit count is determined; and, if it is even, a parity bit is generated which is written into memory along with the data.

If this is to be a read operation, the address must be decoded, as was done with the write, and the timing circuitry must also be enabled, to gate the information from memory into the data register. As the data is moving into the registers, it is checked for the correct parity and, if incorrect, the processor is notified of the error.

TRANSLATOR

Included within memory control is a translator which converts BCL or, optionally, ASCII data

from peripheral devices to EBCDIC data for use in the central system. By using the bit configuration of the data as addresses to integrated circuit ROM look-up tables, "on the fly" translation can be done with no loss of time. The decision as to whether to translate or not is made by the I/O control, dependent upon the media being handled.

ERROR CORRECTION

This feature enables the correction of single bit errors that would otherwise cause the job to be aborted. It involves the implementation of an additional five bits to the memory word which, in combination, can specify a particular bit position in the word. These additional bits, called hamming bits, are not available for use by the programmer.

As shown in the representation of a memory word below, the hamming bits (Hn) occupy those positions which can be specified by a power of two; i.e., positions 1, 2, 4, 8, and 16.

BIT POSITION	1	2	3	4	5	6	7	8	9	10	11
BIT NAME	H1	H2	A8	H4	A4	A2	A1	H8	B8	B4	B2

BIT POSITION	12	13	14	15	16	17	18	19	20	21	22
BIT NAME	B1	C8	C4	C2	H16	C1	D8	D4	D2	D1	P

During writing, a hamming bit is used to generate even parity across specified bit positions. The hamming bits generated by a write are stored in memory along with the data and parity bits.

In figure 13, the data written was F3D6, and the data read out was FBD6, due to picking up the B8 bit. The bit name, position and the data written are in the first three columns. The X's indicate those data bits that the particular hamming bit monitors. Remember that the hamming bit is used to maintain even parity over those bits it monitors. Since there are seven bits ON out of the ten which are monitored by HB1, HB1 is turned ON to achieve even parity. HB2 is OFF, since there are six bits on in the positions HB2 monitors. Following this same logic, it can be seen that HB4 will be ON, HB8 OFF, and HB16 ON. Since the bit count, including the hamming bits, is even, the parity bit is turned on to achieve overall odd parity.

During a read, the previously written hamming bits are exclusively ORed with those generated by the read. If the resultant value is zero, and parity is odd, the data read is correct. If the resultant value is not zero, and parity is even, the data has a single error and is correctable. If the resultant value is not zero and parity is odd, the data read has a double error which, while detectable, is not correctable. In either error situation, the processor is notified of the error. Overall parity is such that the total bit count of data and hamming bits is odd.

In those cases where single-bit errors are indicated, the resultant value of the exclusive ORing process specifies the position of the incorrect bit, which is then complemented. Error detection and correction are accomplished without increasing the memory cycle time.

OPERATOR CONSOLE AND DISPLAY

Operator Display

The operator display consists of the following:

1. A numeric tube display of two groups of six digits representing the following combinations:

LEFT DISPLAY	RIGHT DISPLAY
OP AF BF	INSTRUCTION ADDRESS
MEMORY ADDRESS	MEMORY INFORMATION
PROGRAM ADDRESS	BASE LIMIT

Each group is appropriately identified by a legend which is illuminated only when the numeric tube display is operating in that particular mode.

Each digit position is capable of displaying all 16 bits of the extended number set.

A halt, whether programmed or manual, displays "OP-AF-BF" and "Instruction Address". An 8-digit instruction (Address Branch-Exit-Halt Branch) displays its OP code on the left and the instruction address on the right.

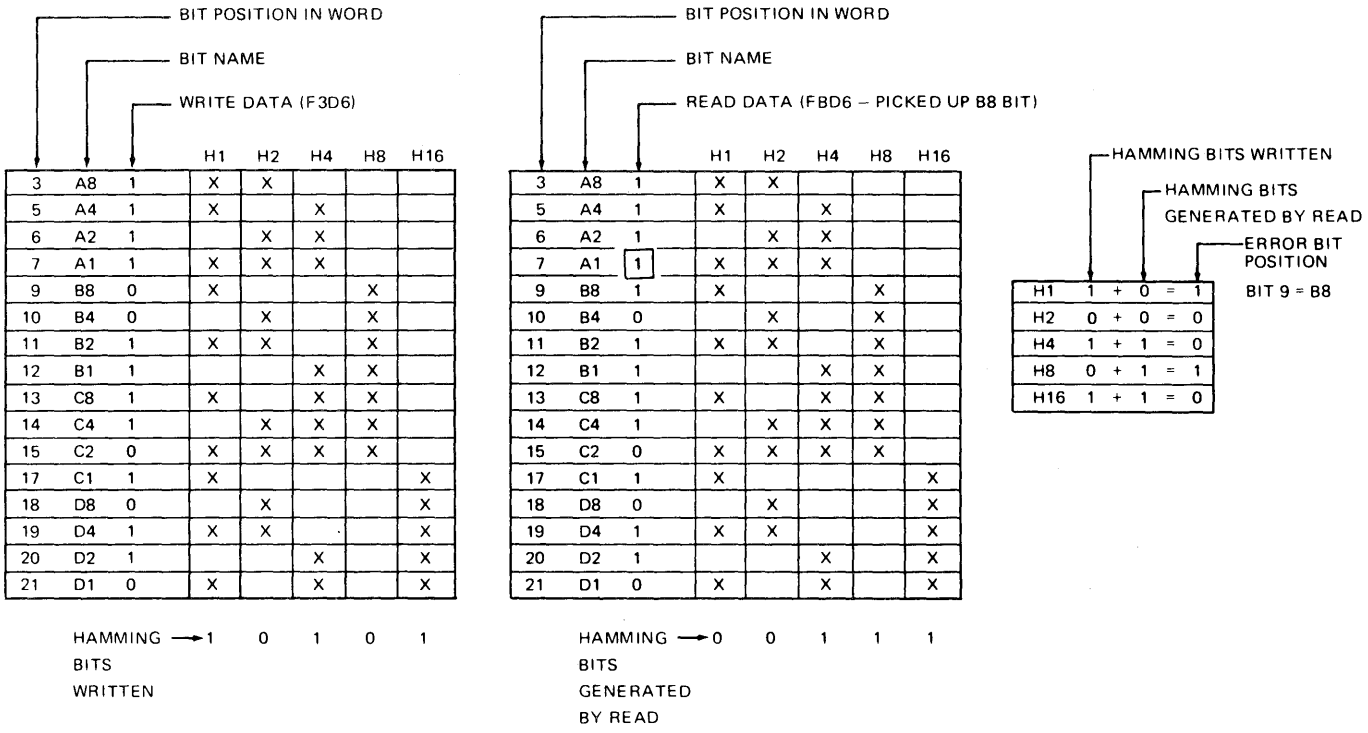


Figure 13. Error Correction

When indirect field length is specified in the AF or BF fields, the display in the AF or BF positions is the resultant field length and not the bit combination specifying indirect field and its address.

When a literal is specified in the AF field, the bit combination specifying the literal operand and the bit combination specifying the controller are cleared to zero before display. Subsequent use of the "A" key displays the address of the operand and not the literal. (See description of the OP and the A keys.)

2. Channel indicators (00 to 19) which light when the corresponding I/O channel is busy.

3. The following legends which are illuminated only when the corresponding condition exists. ("Program" lights for an invalid OP or an invalid address, "Check" lights for memory parity error, "Program" and "Check" light for an instruction time out.)

NORMAL	ASCII	INTERRUPT
LOW	EQUAL	HIGH
PROGRAM	CHECK	OVERFLOW
	TEST	OVERHEAT

Console Controls

The console controls consist of keys used to enter information into registers or memory, keys used to control the display of information, one key to stop and start the processor, one key for terminate, and keys for power on and power off. All keys are appropriately labeled. Key names and related functions are given below:

CLR (clear)	Sets nearly all flip-flops in the processor, I/O controls, central control, and memory control to the false or cleared state, with the exception of the limit register which is set to memory size. This key is active only when the processor is in a stopped condition and when all I/O control operations are completed.
TERM (Terminate)	Performs the same function as the CLEAR key except this key is always active and provides an absolute means of handling the processor which otherwise may be unhaltable because of a hardware or program failure. The halt is immediate. No display is presented.
STOP/RUN	Causes the processor to halt if running or to start if halted. The legends just above the key indicate the appropriate state. The STOP switch causes a stop only at the end of the instruction being executed. The keyboard is inactive until all the I/O operations are complete.
OI (Operator Interrupt)	When depressed, this key alters the function of the STOP-RUN bar. If the processor is running, depressing of the STOP-RUN bar causes the Interrupt flip-flop to be set and stores a processor result descriptor. There is no halt. If the processor is in a stopped state, depression of the STOP-RUN bar starts the processor.
ON	The power ON key initiates the sequence up of the power supply cabinet which provides all power for the central system. System controlled auxiliary cabinets will also be powered on by the sequence initiated by the power ON switch. However, local controlled auxiliary cabinets and all peripheral units must be powered up individually. A delay inhibits the use of the power ON switch for a minimum of 10 seconds after the power OFF button has been pressed (even if the system is in a powered off state with circuit breakers on at the time that power OFF is pressed).
OFF	This switch powers off the central system and system controlled auxiliary cabinets. Power OFF unlatches the overvoltage and undervoltage detection circuits. Therefore, if either an overvoltage or an undervoltage causes a power down sequence, power OFF must be pressed before the power ON switch is active. Moreover, the delay referred to in the power ON switch description above is triggered. This delay allows time for certain capacitors in the power supplies to discharge and their circuits to stabilize before another power on sequence may begin. The power OFF switch also clears most flip-flops in the central system and, thereby, unconditionally terminates all logical operations before the power down sequence begins.
SI (Single Instruct)	Causes current instruction to be executed and the next to be fetched and displayed. The first depression after a halt causes a fetch and display only.

OP	Causes OP-AF-BF and address of current instruction (the "F" word of address memory) to be displayed. Subsequent use of the keyboard enters the digits (0..9, 10..15) into the OP-AF-BF register. The entry of the first digit blanks the remaining display. The entered digits are shifted from right to left in the left-hand group of six NIXIE® tubes.
A, B, C	Depression of any one of these three keys causes the current contents of the applicable word of address memory (A, B, or C) to be displayed in the right-hand group of NIXIE® tubes. The OP-AF-BF register contents also are displayed in the left-hand group of the NIXIE® tubes.
PA (Program Address)	Causes the program address of the next instruction (the "D" word of address memory) and the base and limit register settings to be displayed. Subsequent use of the keyboard enters an address into the program address register (the "D" word of ADM). The entry of the first digit blanks the remaining portion of the left-hand display. the entered digits are shifted from right to left in the left-hand group of NIXIE® tubes.
BM (Base Modify)	When depressed, and in conjunction with other appropriate keys, causes memory address, program address, or instruction address to be displayed base relative in lieu of absolute. Therefore, while the base modify function is active, the value in the base register is subtracted from all addresses before they are displayed. Conversely, the base register value is added to all addresses entered via the keyboard while the base modify function is active. Therefore, base relative addresses must be entered under these circumstances. Pressing the BM key a second time unlatches the function.
CS (Control State)	This key acts as a switch; when CS is on and the processor is in normal state, depression of the SI key causes execution of a single instruction. If the execution of that single instruction causes a return to control state with this key active, all control state instructions are executed continuously; the processor halts only when normal state is reinstated. Thus, the execution of a communicate instruction consists of the execution of all control state instructions associated with that communicate.
AD (Address)	Causes the memory address register and the information contained at that address to be displayed. Subsequent use of the keyboard enters an address into the memory address register. The entry of the first digit blanks the remaining display. The entered digits are shifted from right to left. Depending on the mode of the address displayed in the left-hand NIXIE® tubes, 1, 2, or 4 digits of memory information are displayed in the right-hand NIXIE® tubes. Refer to READ key.
WR (Write)	This key is active only subsequent to the depression of the AD key and permits insertion of information to memory via the keyboard. The SKIP key can be used. The memory address is advanced with each entry. The digit to be changed appears left-justified. When a digit is entered, that digit appears right-justified and the new digit to be changed appears left-justified.
SKIP	This key is active only subsequent to the depression of the WR key. This key is used to skip over a digit position when writing into memory without changing the digit.

®Registered Burroughs Trademark

READ	This key is active only subsequent to the depression of the AD key. This key causes the next digit (or group of digits) to be read from memory and displayed. The memory address is advanced by four once it is synchronized to 000000 + 4N (mod 4).
KEYBOARD	The group of 16 keys from 0 to $\overline{5}$ representing the 16 hexadecimal numbers 0 through 15. The keyboard is active in the STOP state and inactive in the RUN state.
LD (Load)	Initiates a program load into memory from a preselected peripheral. Refer to load function.

Load Function

Two types of load commands are available. The first type, called Universal Load, permits operator selection of the input media. The second type, called Normal Load, restricts the selection of the input media to a particular peripheral. This peripheral selection can be changed by a simple field change made by the Field Engineer.

UNIVERSAL LOAD

Manual depression of the clear button causes the processor to be cleared. The operator then inserts, via the keyboard, eight digits into memory beginning at location 000000. The first two digits form a two-digit channel number, and the next six digits form the first syllable of an I/O descriptor acceptable to the peripheral control attached to the channel selected.

Operation code 66 is entered into the OP register from the keyboard by the entry 660000.

Manual depression of the RUN button causes the processor to go into a pseudo initiate I/O cycle in which the channel number and the first syllable of an I/O descriptor are obtained from memory starting at address 000000.

The begin and end addresses for the I/O descriptor are set to 001000 and 001400, respectively; and segment number, in the case of a disk, is set to zero. Having completed the initiate I/O cycle, the processor idles until the interrupt flip-flop is set signifying completion of the I/O operation. When the interrupt flip-flop is set, the processor transfers the absolute address of the pertinent result descriptor to IX1 and clears the result descriptor area to zero after transferring the most-significant digit of the result descriptor to a temporary storage register, BBA. If no exception bit (BBA4F) is present, the processor strips the most-significant 4-bit digits from sequential characters beginning at address 1000 and compresses the field to 100 4-bit digits, which are stored back into memory address 1000 through 1099. A branch is then taken to location 001000.

If the peripheral control returned an exception bit, the operation is automatically retried.

An invalid descriptor causes the processor to halt.

NORMAL LOAD

A load button is on the console and a function providing the eight-digit channel number and the first six digits of an I/O descriptor for insertion in location 000000 is provided. The eight digits that are inserted can be changed by a Field Engineer.

Depression of the load button causes these digits to be written into memory starting at location 000000 after which the sequence described under universal load takes place.

CENTRAL SYSTEM POWER

All voltages utilized in both the B 3700 and the B 4700 systems, with the exception of those developed in the B 4700 memory, are developed and controlled in one central power cabinet. These voltages are developed by a converter-to-inverter type system, i.e., one that first converts the commercial input power to 160 VDC, then runs the 160 VDC through an inverter to develop a 160 V peak-to-peak, 2 KHz square wave, which is then used as a source to the various regulators.

The regulators are both series and shunt types, generally employing on-board voltage sensing, and, in some cases, remote overcurrent detection.

The voltages used in the processor are shown in figure 14 and are described in the following paragraphs.

+4.75, −2, DC Common

These three voltages are the primary logic supply voltage. The 4.75 volts is used as the logic TRUE level, -2 volts is the logic FALSE level, and DC COMMON is the reference line for these and all other logic voltages in the system.

+170 Volts

This voltage is used only in the NIXIE® tube portion of the display panel. Because of their construction, NIXIE® tubes require this voltage to ionize the gas surrounding the selected bar.

±12 Volts and ±10 Volts

These voltages are used for biasing and driving the transistors used in the console and display circuit.

30 Volts AC

This is a sampled portion of the source voltage which is used to monitor input voltage fluctuations.

Air Sense

This line comes from the AIR LOSS-TEMPERATURE sense circuitry in the processor and is used to drop system power if there is a loss of airflow to the processor or if ambient temperature becomes too high.

Power ON/OFF

These lines come from the ON and OFF buttons on the operator console and enable the operator to bring up or drop system power.

+4 Volts, +22 Volts

These voltages, used only on B 3700 systems, are used to drive the IC memory components.

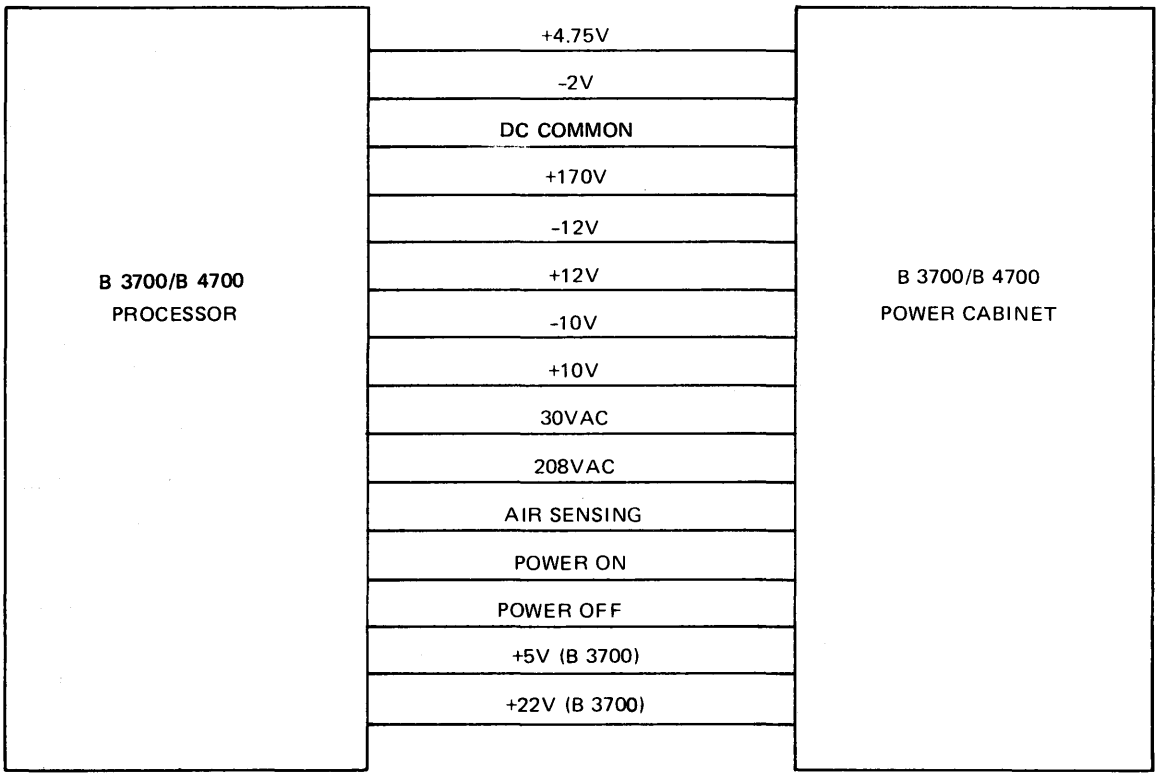


Figure 14. Processor Voltages

APPENDIX A

B 3700/B 4700 INSTRUCTION SET

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
INC	01	aa	bb	2	c	c	(A+B) → B
ADD	02	aa	bb	3	c	c	(A+B) → C
DEC	03	aa	bb	2	c	c	(B-A) → B
SUB	04	aa	bb	3	c	c	(B-A) → C
MPY	05	aa	bb	3	c	N.A.	(AxB) → C
DIV	06	aa	bb	3	c	c	(B/A) → C Remainder in B
MVD	08	OV	00	3	N.A.	N.A.	Move words A→B, increment A and B add. (until B add C add) Decrement A and B add, then move backward A→B (until B add = C add)
MVL	09	nn	ii	3	N.A.	N.A.	B→A, C→B, A→C nn = length of A,B,C
MVA	10	aa	bb	2	c	c	A→B
MVN	11	aa	bb	2	c	c	A→B
MVW	12	nn	nn	2	N.A.	N.A.	A→B, nnnn = Number of words
MVC	13	nn	nn	2	N.A.	N.A.	A→B, Clear A, nnnn = Number of words
MVR	14	aa	nn	2	N.A.	N.A.	A→B, move A field nn times
TRN	15	nn	nn	3	N.A.	N.A.	Translate A with table in B and move in C B add must be multiple of 1000 nnnn = Number of char or digit to translate
SDE	16	aa	bb	2	c	N.A.	Com H - Char/Digit - No compare (AN ≠ Bn)

If answer is:
= 0 COMP = E
> 0 COMP = H
< 0 COMP = L

B 3700/B 4700 INSTRUCTION SET (Cont)

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
SDU	17	aa	bb	2	c	N.A.	Com E - Char/Digit - Compare (An = Bn)
SZE	18	aa	bb	2	c	N.A.	Com L - 1st Char/ Digit - Compare ($A_n = B_1$)
SZU	19	aa	bb	2	c	N.A.	Field Length into B + 38
NOP	20	-	-	1	N.A.	N.A.	No Significant Action
LSS	21	-	-	1	N.A.	N.A.	BR. to A add IF COM = LOW
EQL	22	-	-	1	N.A.	N.A.	BR. to A add IF COM = EQUAL
LEQ	23	-	-	1	N.A.	N.A.	BR. to A add IF COM = LOW or EQUAL
GTR	24	-	-	1	N.A.	N.A.	BR. to A add IF COM = HIGH
NEQ	25	-	-	1	N.A.	N.A.	BR. to A add IF COM ≠ EQUAL
GEQ	26	-	-	1	N.A.	N.A.	BR. to A add IF COM = EQUAL or HIGH
BUN	27	-	-	1	N.A.	N.A.	BR. to A add anytime
OVF	28	-	-	1	N.A.	N.A.	BR. to A add IF OVF IS SET
HBR	29	-	-	1	N.A.	N.A.	Exam ADD 77 And Halt unless: Bit 1 = 1 Ignore Halt in Nor Bit 2 = 1 Ignore Halt in Cont. Bit 4 = 1 Invalid inst. unless ignore Bit 8 = 1 Not significant

B 3700/B 4700 INSTRUCTION SET (Cont)

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
BCT	30	AA	AA	0	0	0	ABS (64→76)←NI, BA, LA, COM: BR. to AAAA (Indirect Address) IF AAAA Address is not 5nnnnn DO 300094 ON B 4700 inv inst gives 300094
NTR	31	nn	nn	1	0	0	Stack←NI, IX3, COM, nnnn PAR IX3←Sa (Stack Address)
EXT	32	-	-	1	c	c	B+40←IX3, (IX3, COM)←Stack Branch to A Address
BST	33	aa	mm	1	c	N.A.	Set Bits per mask (mm) in aa units of a field
BRT	34	aa	mm	1	c	N.A.	Reset Bits per mask (mm) in aa units of a field
SLL	37	aa	nn	2	c	N.A.	Search Link List A compared to B+BF IF = Store B Address in IX1 IF≠ Go to B Add When B=0 Set COMH A CONTROLLER 00 Four Bit Mode 01 Reserved 10 8-Bit Mode 11 Indirect Add AF=Length of Key BF = OffSet From B A=Key Address B 1st list entry

B 3700/B 4700 INSTRUCTION SET (Cont)

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
SLD	38	aa	nn	2	c	N.A.	Search Link Delink As above, but IX1 = B Add IX2 = Previous B Add
SEA	34	aa	nn	3	c	N.A.	aa = Length of A & B nn = B Field Increment c Address = End cc = 0 Search for = cc = 1 Search for Lower cc = 2 Search for Lowest
BZT	40	aa	mm	1	c	N.A.	m = Mark A Bit/ = M bit→COM E Else
BOT	41	aa	mm	1	c	N.A.	m = Mark A Bit = M Bit→COME Else→COMH
AND	42	aa	bb	3	c	N.A.	(C Bit←1) IF (A Bit*B bit)
ORR	43	aa	bb	3	c	N.A.	(C Bit←1) IF (A Bit + bit)
NOT	44	aa	bb	3	c	N.A.	(C Bit←1) IF (A Bit*B bit)1+(A bit/* B bit)
CPA	45	aa	bb	2	c	N.A.	Collating Sequence
CPN	46	aa	bb	2	c	N.A.	Algebraic Compare
SMF	47	mi	ii	0	N.A.	N.A.	m=0 set EBCD1C; m=1 set ASCII
HBK	48	ii	mm	0	N.A.	N.A.	m = (Base+46) then Execute Halt as per 77ABS
EDT	49	aa	bb	3	c	0	C←A Format as per B
IAD	50	--	--	1	C.IF OVF	c	ACC + A
IAS	51	--	--	1	c	c	ACC + A → A
ISU	52	--	--	1	C.IF OVF	c	ACC - A
ISS	53	--	--	1	c	c	ACC - A → A
IMU	54	--	--	1	C.IF OVF	c	AX ACC
IMS	55	--	--	1	c	c	AX ACC → A

B 3700/B 4700 INSTRUCTION SET (Cont)

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
IMT	57	--	--	1	C.IF OVF	c	AC = 0 A + 1 →A/ AC = 1, A-1 → A
ILD	58	--	--	1	N.A.	N.A.	A → ACC
IST	59	--	--	1	c	N.A.	ACC → A
CKB	60	dd	in	N.A.	N.A.	N.A.	dd = Information, n bits - 8=1 (READ) 4 = 1 Base Mod, 1 = 1 (Complement)
ULD	66	ii	ii	N.A.	N.A.	N.A.	Univ. Load, ABS ADD Zero = CCOPVVV (I/O Channel & I/O DESC.)
RAA	70	--	--	1	C.IF OVF	c	ACC + A
RAS	71	--	--	1	c	c	ACC + A →
RSU	72	--	--	1	C.IF OVF	c	ACC - A
RSS	73	--	--	1	c	c	ACC -A→A
RMU	74	--	--	1	C.IF OVF	c	AXACC
RMS	75	--	--	1	c	c	AXACC → A
RDV	76	--	--	1	C.IF OVF	c	ACC + A
RDS	77	--	--	1	c	c	ACC ÷ A →A
RLD	78	--	--	1	N.A.	N.A.	A → ACC
RST	79	--	--	1	c	N.A.	ACC → A
FAD	80	aa	bb	3	c	c	C ← B + A
FSU	81	aa	bb	3	c	c	C ← B - A
FMP	82	aa	bb	3	c	c	C ← B X A
FDV	83	aa	bb	3	c	c	C ← B/A (Remainder in B)

B 3700/B 4700 INSTRUCTION SET (Cont)

MNE	OP	AF	BF	# OF ADD	COM IND	OVF	REMARKS
ACM	84	D_3D_4	NA	0	C.IF OVF	c	D_3D_4 0 i Normalize 1 i Convert Real to Integer 2 i Set Mantissa sign to + 3 i Set Mantissa sign to - 4 i Complement Mantissa Sign 5 i Clear ACC to -99 + 0 6 n Increment exponent by n 7 n Decrement exponent by n
IBA	85	ii	ii	0	N.A.	N.A.	BRANCH TO 003000
BRE	90	Ni	ii	0	c	c	(NI, BA, LA, COM)←(64→76) abs. N=0 set NOR; AFB=1 Set Trace F/F Br. to NI
SRD	91	AA	AA	0	c	N.A.	Examine R/D's starting at AAAA
RAD	92	im	cc	1	N.A.	N.A.	cc = Chan, # ; A--Begin (lm=0) A←End (m=1) Write A & B (m=9)
II0	94	ii	cc	1	N.A.	N.A.	cc=Chan, # ; A Add locates I/O DESC
RDT	95	ii	ii	1	N.A.	N.A.	A←G
RCT	96	ii	ii	1	N.A.	N.A.	A←G; g←O
STT	97	ii	ii	1	N.A.	N.A.	H←A
SNE	98	LS	CT	0			Snap Execute } L=LC S=SC CT=Iteration count
SNF	99	LS	CT	0			Snap Fetch }

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: MEDIUM SYSTEMS
B 3700/B 4700
CENTRAL PROCESSOR
Reference Manual

FORM: 1063799
DATE: 12-72

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

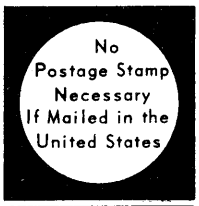
GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

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

cut along dotted line

STAPLE

FOLD DOWN SECOND FOLD DOWN



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