

5

Dept Exp. Psychology
Walter Reed Army Institute for Research
Forest Glen Section

pb 250
REFERENCE MANUAL

pb Packard Bell Computer

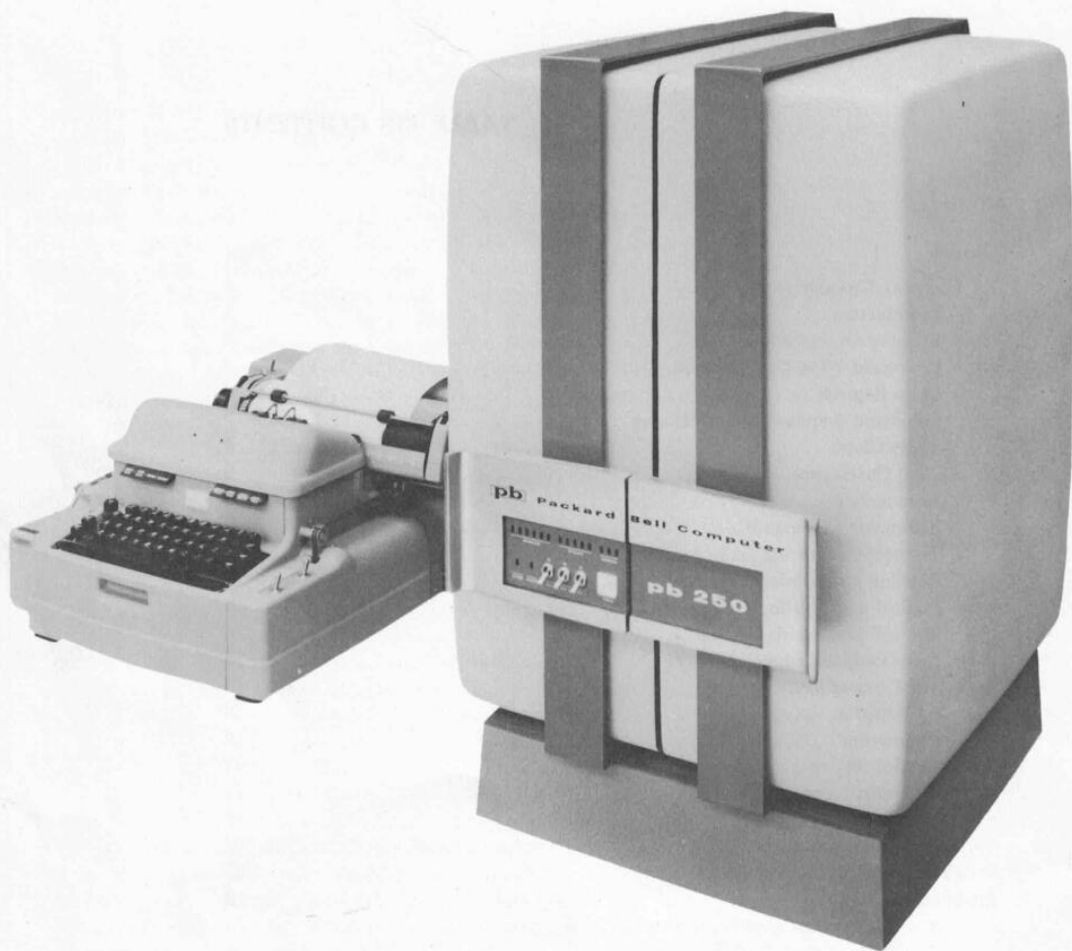
Reference Manual

PB 250

General Purpose Digital Computer

TABLE OF CONTENTS

| <i>Chapter</i> | <i>Page</i> |
|--|-------------|
| 1. GENERAL CHARACTERISTICS | 5 |
| Introduction | 5 |
| Memory Organization | 5 |
| Command Word Configuration | 6 |
| Index Register | 6 |
| Command Sequencing and Timing | 7 |
| Parity Check | 8 |
| 2. PB 250 COMMANDS | 9 |
| Introduction | 9 |
| Arithmetic Commands | 9 |
| Transfer Commands | 12 |
| Loading and Storing Commands | 13 |
| Logical and Shifting Commands | 15 |
| Control Commands | 16 |
| Input-Output Commands | 16 |
| 3. INPUT-OUTPUT SYSTEM | 19 |
| Introduction | 19 |
| Flexowriter | 19 |
| Control Switches | 21 |
| Computer Console | 21 |
| Flexowriter Input | 22 |
| Flexowriter Output | 22 |
| 4. FUNCTIONAL COMPUTER LOGIC | 25 |
| APPENDICES | 26 |
| A. Binary-Octal Number Systems | 26 |
| B. Table of Powers of 2 | 28 |
| C. Octal-Decimal Integer Conversion Table | 29 |
| D. Octal-Decimal Fraction Conversion Table | 33 |
| E. Command List | 36 |



PB 250 General Purpose Digital Computer.

1. GENERAL CHARACTERISTICS

Introduction

The Packard Bell PB 250 is a high-speed, completely solid-state general purpose digital computer in which both the data and the commands required for computation are stored in a homogeneous memory. The storage medium is a group of nickel steel magnetostrictive lines along which acoustical pulses are propagated. At one end of each of these lines is a writing device for translating electrical energy into acoustical energy. At the other end of each line is a reading device for translating acoustical energy back into electrical signals. By rewriting the stored information as it is read, information continuously circulates without alteration except for alterations which result from the execution of the computer program.

The PB 250 provides a repertoire of more than 50 commands flexible enough to permit coding of a very broad range of scientific and engineering problems. Double precision commands are provided for operating upon large numbers. Commands to normalize and scale numbers facilitate floating point operation. Square root, and variable length multiplication and division operations are available in the command list. Other features include input-output buffering, and a large number of optional peripheral units such as punched card equipment, tape handlers, shaft encoders, photo readers, and analog-to-digital and digital-to-analog converters. A magnetic core memory of 16,384 words can be linked with the computer, providing an input-output rate of up to 85,000 words per second.

Memory Organization

The memory of the basic PB 250 contains ten lines, numbered octally from 00 through 11, which hold both data and instructions. Each long line, 01 through 11, contains 256 (decimal), or 400 (octal), locations, also called sectors, that are numbered 000 through 377. *Note: All sector and line numbers are given in octal notation throughout this manual.* Since the information

in any location can be either data or a command, the generic term "word" is used to cover both. The location of any word is specified by a line and sector number, and these together are called an address. Line 00 is a 16-word Fast Access Line. Since line 00 is 1/16 the length of a long word line, any unit of information contained in it is available 16 times during each complete circulation of the 256-word lines. Thus, any word in the Fast Access Line can be identified by one of 16 sector addresses. For example, sector 000 of the line 00 can be identified by the following addresses: 00000, 02000, 04000, 06000, 36000.

Fifty-three additional lines, each of which may have from one to 256 words, can be added. These lines are numbered 10 through 36 and 40 through 77. Line number 37 is used for the Index Register. If all of the additional lines are used, and if all hold 256 words, the memory capacity of the PB 250 is extended to 15,888 words.

Commands can be executed only from lines 00 through 07; these lines are therefore designated "Command Lines."

Data Word Configuration

Every number stored in the PB 250 is represented by a series of pulses which correspond to a series of zeros and ones that are the digits of the binary number system. The term "binary digit" is usually contracted to the word "bit." (A discussion of binary numbers may be found in appendix A.)

A number stored in a location in the PB 250 consists of twenty-one bits that represent magnitude and a twenty-second bit to indicate sign. A negative number has a one in position zero, whereas a positive number has a zero in position zero. Negative numbers are expressed in their 2's complement form. (A discussion of complementary arithmetic may be found in appendix A.)



Figure 1.—Data Word Configuration

These 22 positions are sufficient to represent a 6-digit decimal number. Larger numbers may easily be represented by using the double precision features of the computer.

Arithmetic Registers

Three arithmetic registers, A, B, and C, are provided for arithmetic operations and information manipulation. Each register has exactly the same format as a memory location, including the sign, and all are available to the programmer. Double precision commands treat A and B as a double-length register; information may be interchanged between A, B, and C. The contents of a register may be tested for non-positive values or compared against the contents of any memory location. A record may be kept in one register of operations performed on the others. The specific functions of these registers are described in the discussion of the commands in Chapter 2.

Stored Program Concept

The computer is capable of performing certain operations, or "obeying commands," and more than 50 such commands are available to the programmer for problem solving. A set of commands, or instructions, is known as a program and is stored in memory, one command per word; thus the term "stored program." Commands may specify or "address" memory locations, manipulate data, call or read in information from external sources, or even operate on other commands.

Each command has a specific numeric code, but for greater convenience a three-letter mnemonic code has been devised. For example, the command DIVIDE has the numeric code 31, and the mnemonic DIV. A program to accept mnemonic OP codes can be devised which will cause the computer to substitute internally the proper binary representation for an alphabetic code. The code, whether numeric or mnemonic, is referred to as the operation code or, for brevity, the OP code. Numeric OP codes are always specified in octal notation.

Command Word Configuration

As previously mentioned, information in any memory location may be either data or a command. When the information is a command, it has a definite configuration, or format, as illustrated below.

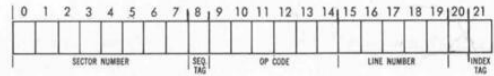


Figure 2.—Command Word Configuration.

Each subdivision, or field, of the command word is uniquely identified. The subdivisions are the sector number, Sequence Tag, OP code, line number, and Index Tag fields. There will be frequent references in subsequent discussions to the address field of a command. Although the address is made up of a sector and a line number, these numbers are not contiguous in the command format. The address field, however, will be considered as a single entity. The address 03204 refers to sector 032, line 04. The contents of the address field in a command do not always designate a memory location. For example, the shifting commands use the address field to indicate the number of places to shift.

The Sequence Tag field may contain either a one or a zero, and its use is detailed in the section entitled "Command Sequence and Timing" in this chapter.

The OP code field contains a numeric code which specifies one of the PB 250 commands.

The Index Tag field may contain either a one or a zero. When a one is placed in this field, the contents of the Index Register are used; a zero in the field indicates no use of that register.

Bit position 20 contains a one only when referring to a line number of 40 or greater.

Index Register

The Index Register stores a line number for use with commands which have an Index Tag of one. When used, the contents of the Index Register replace the line number of the address in the command. This replacement is made during the reading of the command, *but does not change the command as it stands in memory*. For example, if the contents of the Index Register are 01, then in the execution of the following program step:

| OP CODE | ADDRESS | INDEX TAG |
|---------|---------|-----------|
| ADD | 03204 | 1 |

the computer will add the contents of location 03201, instead of 03204.

Line number 37 is reserved to designate the Index Register. Addresses 00037 through 37737 all apply to this register, and bit positions 16 through 20 are the useful positions for the line number. Thus, a STA into line 37, any sector, places bits 16 through 20 of A into the Index Register, bits 16 through 20.

Command Sequencing and Timing

The PB 250 reads and executes commands from the circulating command lines. The words of the long lines are read serially in sector number sequence (000, 001, 002 - - - 376, 377, 000, 001 - - -). The time for each word to pass through a reading device is 12 microseconds; therefore, the time for all 256 words of a long line is 3072 microseconds. The commands are read and executed in numerical order from a given line starting in sector 000 of line 01 (00001, 00101, 00201, - - -). The performance of each command involves four phases:

| | |
|-----------|----------------------------|
| Phase I | Wait to read next command. |
| Phase II | Read next command. |
| Phase III | Wait to execute command. |
| Phase IV | Execute command. |

For example, a command in 00001 to store A in 03004 will be read (Phase II) in sector 000, held for execution (Phase III) in sectors 001 through 027, executed (Phase IV) in sector 030, and held while waiting to read the next command (Phase I) in sectors 031 through 000. Phase II will follow in sector 001 to read the next command in 00101.

There are four classes of commands in which the nature of Phase IV differs. A tabulation showing into which class each command falls is provided in Table 1.

CLASS 1.

In this class of commands, execution always follows the reading of the command by skipping Phase III. The sector number in the command is used to designate the first sector number in which Phase IV is discontinued. This class of commands consists of all those which require an extended interval of execution such as block transfer, shifting, and multiplication. The execution time for this class of command varies with the required duration. For example, block trans-

fers require 12 microseconds per word, shifting requires 12 microseconds per bit, and multiplication requires 12 microseconds per multiplier bit.

CLASS 2.

In this class of commands, execution is always completed in the sector specified by the sector number of the command. This class consists of all one-sector operations such as load, store, add,

TABLE 1.—COMMAND CLASSIFICATION.

| CLASS 1.—Executed between command location and address sector number. | | |
|---|-----|------|
| HALT | HLT | (00) |
| MERGE A INTO C | MAC | (00) |
| NORMALIZE AND DECREMENT | NAD | (20) |
| LEFT SHIFT AND DECREMENT | LSD | (21) |
| RIGHT SHIFT AND INCREMENT | RSI | (22) |
| SCALE RIGHT AND INCREMENT | SAI | (23) |
| NO OPERATION | NOP | (24) |
| INTERCHANGE A AND M | IAM | (25) |
| MOVE LINE X TO LINE 7 | MLX | (26) |
| SQUARE ROOT | SQR | (30) |
| DIVIDE | DIV | (31) |
| DIVIDE REMAINDER | DVR | (31) |
| MULTIPLY | MUP | (32) |
| SHIFT B RIGHT | SBR | (33) |
| WRITE OUTPUT CHARACTER | WOC | (6X) |
| PULSE TO SPECIFIED UNIT | PTU | (70) |
| MOVE COMMAND LINE BLOCK | MCL | (71) |
| BLOCK SERIAL OUTPUT | BSO | (72) |
| BLOCK SERIAL INPUT | BSI | (73) |

| CLASS 2.—Executed in address sector number. | | |
|---|-----|------|
| INTERCHANGE A AND C | IAC | (01) |
| INTERCHANGE B AND C | IBC | (02) |
| LOAD A | LDA | (05) |
| LOAD B | LDB | (06) |
| LOAD C | LDC | (04) |
| STORE A | STA | (11) |
| STORE B | STB | (12) |
| STORE C | STC | (10) |
| ADD | ADD | (14) |
| SUBTRACT | SUB | (15) |
| EXTEND BIT PATTERN | EBP | (40) |
| GRAY TO BINARY | GTB | (41) |
| AND M & C | AMC | (42) |
| CLEAR A | CLA | (45) |
| CLEAR B | CLB | (43) |
| CLEAR C | CLC | (44) |
| AND OR COMBINED | AOC | (46) |
| EXTRACT FIELD | EXF | (47) |
| DISCONNECT INPUT UNIT | DIU | (50) |
| READ TYPEWRITER KEYBOARD | RTK | (51) |
| READ PAPER TAPE | RPT | (52) |
| READ FAST UNIT | RFU | (53) |
| LOAD A FROM INPUT BUFFER | LAI | (55) |
| COMPARE A AND M | CAM | (56) |
| CLEAR INPUT BUFFER | CIB | (57) |

| CLASS 3.—Executed in address sector number and following sector. | | |
|--|-----|------|
| ROTATE | ROT | (03) |
| LOAD DOUBLE PRECISION | LDP | (07) |
| STORE DOUBLE PRECISION | STD | (13) |
| DOUBLE PRECISION ADD | DPA | (16) |
| DOUBLE PRECISION SUBTRACT | DPS | (17) |

| CLASS 4.—Executed between command location and address sector number. | | |
|---|-----|------|
| TRANSFER UNCONDITIONALLY | TRU | (37) |
| TRANSFER IF A NEGATIVE | TAN | (35) |
| TRANSFER IF B NEGATIVE | TBN | (36) |
| TRANSFER IF C NEGATIVE | TCN | (34) |
| TRANSFER ON OVERFLOW | TOF | (75) |
| TRANSFER ON EXTERNAL SIGNAL | TES | (77) |

and clear. All commands of this class require 12 microseconds to execute.

CLASS 3.

Class 3 is an extension of Class 2 to handle double precision operations. As in Class 2, execution always starts in the sector specified by the sector number of the command but the execution phase is always extended into the following sector. All commands of this class require 24 microseconds to execute.

CLASS 4.

Class 4 consists of commands for conditional and unconditional transfer of control. The condition for a conditional transfer is tested in Phase II and, if the condition is met, the next command is read from the line and sector number specified by the command. If the condition is not met, the command directly following the transfer of control command is read. A conditional transfer where the condition is not met, thus requires no execution time. The unconditional transfer selects the next command with no restrictions. The execution time when control is transferred is 12 microseconds per sector for the interval between the transfer of control command and the next command.

With commands stored in sequential sectors, the indicated command sequence will proceed at the rate of one instruction per $(3072 + 12)$ microseconds. To provide for a higher computation rate, a Sequence Tag of one may be used in bit position 8 of commands in Classes 1, 2, and 3. The use of this option will dic-

tate that the next command will be read in the sector directly following the end of the execution phase. For example, a command in 00001 to store A in 03004 will be followed by the command in 03101 if the Sequence Tag is a one. The use of the Sequence Tag will increase the computation rate from 300 commands per second to a rate approaching 40,000 commands per second.

When the 16-word line, line 00, is used as the command line, the command sequence will be improved. For example, a command read from location 00100 to store A in 03004 will be executed in sector 030 and the next normal command in location 00200 will be read in sector 042. If the Sequence Tag is one in the command, the next command will be read in sector 031 and will be, therefore, the command in location 01100.

Parity Check

The arithmetic registers and each memory word carry an additional position for an even parity check. This position is not under program control and need not concern the programmer in the design and coding of his problem. The parity check is generated during the execution of the STORE and MOVE commands and is tested when loading the arithmetic registers, during adding and subtracting operations, and when reading commands.

Computation will stop on a parity error, and may be restarted by clearing the parity flip-flop with the BREAKPOINT SWITCH and the ENABLE SWITCH of the typewriter.

2. PB 250 COMMANDS

Introduction

The commands described in this chapter have been grouped according to function. Mnemonics are derived from or are abbreviations of the command names. For example, DIV is derived from DIVIDE, STA from STORE A, and LSD from LEFT SHIFT AND DECREMENT.

Certain conventions are employed in this and subsequent chapters to facilitate the illustration of the various computer functions. A program step is shown as follows:

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
|----------|---------|---------|

The Location is the memory location in which the command is stored. The OP Code refers to the operation, such as LOAD A or STORE C, and in this field the mnemonic code is used for convenience. The Address contains the operand. The command SUBTRACT 23204, which is located at 22101, is designated by:

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 22101 | SUB | 23204 |

The letters A, B, and C refer to the arithmetic registers; the letter M symbolizes a memory location. Parentheses around A, B, C, or M indicate the contents of that register or word. For example, (A) indicates the contents of A, (M) the contents of M, and (AB) the contents of A and B as one register.

In cases where there might be confusion as to whether a number is binary, decimal, or octal, a subscript will be used to indicate which number base applies. For instance, 10110_2 indicates a binary number, 496210_{10} a decimal number, and 37110_8 an octal number. The numeric code for commands is expressed in octal.

Arithmetic Commands

| | | |
|---------------------------|-----|------|
| Add | ADD | (14) |
| Subtract | SUB | (15) |
| Double Precision Add | DPA | (16) |
| Double Precision Subtract | DPS | (17) |
| Square Root | SQR | (30) |
| Divide | DIV | (31) |
| Divide Remainder | DVR | (31) |
| Multiply | MUP | (32) |
| Clear A | CLA | (45) |
| Clear B | CLB | (43) |
| Clear C | CLC | (44) |
| Gray To Binary | GTB | (41) |
| Compare A and M | CAM | (56) |

In the use of the commands ADD, SUBTRACT, DOUBLE PRECISION ADD, and DOUBLE PRECISION SUBTRACT, the resulting value may exceed the capacity of the arithmetic registers. In this event, the Overflow Switch is turned on. The command TRANSFER ON OVERFLOW is used to test for this condition.

| | | |
|------------|------------|-------------|
| ADD | Add | (14) |
|------------|------------|-------------|

The contents of M, the specified address, are added to the contents of the A register. This sum replaces the contents of A; the contents of M are unaffected. Overflow is possible.

| | | |
|------------|-----------------|-------------|
| SUB | Subtract | (15) |
|------------|-----------------|-------------|

The contents of M, the specified address, are subtracted from the contents of the A register. The result re-

places the contents of A; the contents of M are unaffected. Overflow is possible.

Double Precision Operations

Double precision numbers must be stored in consecutive words and the specified address is the lower ordered address. For example, if the specified memory location is 03404, the double precision number is stored in memory locations 03404 and 03504. The location 03404 contains the Least Significant Word (LSW) and 03504 contains the Most Significant Word (MSW). In double precision operations, the A and B registers are combined and treated as one register.

DPA Double Precision Add (16)

The contents of the word pair starting at M, the specified address, are added to the contents of the combined A and B registers. The result replaces the contents of A and B. The word pair at M is not affected. Position 0 of the B register does not act as a sign but is a part of the number and any carry from position 0 propagates into position 21 of the A register. Overflow is possible.

DPS Double Precision Subtract (17)

The contents of the word pair starting at M, the specified address, are subtracted from the contents of the combined A and B registers. The result replaces the contents of A and B. The word pair at M is not affected. Position 0 of the B register does not act as a sign but is a part of the number and any carry from position 0 propagates into position 21 of the A register. Overflow is possible.

SQR Square Root (30)

The argument is (AB), the square root appears in B and the remainder in A. The C register takes part in the operation and its contents are replaced by the square root. This result in C will always be scaled as the full root but it will differ from (B) in the least significant bit computed. If only A is loaded with the argument, B should be cleared or it may influence the least significant bit of the computed root.

The address field of the SQUARE ROOT command is not used to specify the location of an operand, but contains an address number, N, which specifies the

first sector number following the completion of the operation. The SQUARE ROOT command is a variable length operation and, as such, the programmer may specify a quantity, S, which is the number of bits of the root that are to be developed, starting with the half unit bit. The programmer should determine if the argument is positive before performing a SQUARE ROOT operation. If the binary point is always considered as being to the right of the sign bit, and S is $(25)_s$, then the full root is formed in B with the binary point to the right of the sign bit. N is determined from S in the following manner:

$$N_s = \text{The sector number of the command}_s + S_s + 1_s$$

For example: It is desired to generate a full root of 21 bits plus sign in B and the SQUARE ROOT command is in 00104.

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 00104 | SQR | 02700 |

If a Sequence Tag of one is used, the next command is read from 02704. With a Sequence Tag of zero the next command is read from 00204.

When S is $(12)_s$, the half unit bit is formed in bit 12 of B, the one-fourth unit bit is formed in bit 13 of B, etc. Bit 11 of B will be a zero, and bits originally in bits 10 through 20 of B are moved to bits 0 through 10 of B as part of the remainder in A. The result in C will be: a plus sign in bit 0, the first nine bits of the root in bits 1 through 9, and non-useful data in bits 10 through 21. The line number of the address field *must* be 00.

DIV Divide (31)

The dividend is (AB) and the divisor is (C). The quotient appears in B, the remainder in A, and (C) are unaffected. If only A is loaded with a dividend, B should be cleared or it may influence the least significant bit of the quotient.

The address field of the DIVIDE command is not used to specify the location of an operand, but contains an address number, N, which specifies the first sector number following the completion of division. The line number of the address field *must* be 00.

The DIVIDE command is a variable length operation and, as such, the programmer may specify a quantity, S, which is the number of bits of the quotient, including a units bit, that are to be developed. The sign bit is always developed in addition to the S bits. If the binary point is always considered as being to the right of the sign bit, and S is $(26)_s$, then the full quotient is formed in B with the binary point to the right of the sign bit (the sign bit is discarded, and

replaced by the units bit). If S is $25)_s$, one-half of the quotient is formed in B with the binary point to the right of the sign bit. N is determined from S in the following manner:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

For example: It is desired to obtain the full quotient in B and the DIVIDE command is located in 05605.

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 05605 | DIV | 10500 |

If a Sequence Tag of one is used, the next command is read from 10505. With a Sequence Tag of zero the next command is read from 05705.

When S is $12)_s$, the sign of the quotient in B is formed in bit 11, the units bit of the quotient is formed in bit 12, the half unit bit of the quotient is formed in bit 13, etc. The original contents of B in bits 10 through 20 are moved to bits 0 through 10 of B as part of the remainder in A.

DVR Divide Remainder (31)

To obtain the least significant half of a double length quotient, store the first quotient obtained in B and perform the DIVIDE REMAINDER command again with S equal to $26)_s$ to divide the remainder in A by (C). The second result in B may then be joined directly to the first quotient to form a normal double length number. The line number of the address field should be 01.

MUP Multiply (32)

The multiplier must be loaded into the B register and the multiplicand into the C register. The computer clears the A register and puts the product in the combined A and B registers; C is unaffected. The most significant portion of the product will appear in A. The address field of the MULTIPLY command is not used to specify the location of an operand but contains an address number, N , which specifies the first sector number following the completion of multiplication. The MULTIPLY command is a variable length operation and, as such, the programmer may specify a quantity, S , which is the number of bits, starting from the least significant end, of the multiplier to operate on the multiplicand. If the binary point is always considered to be to the right of the sign, and

S is $26)_s$, then the full product is formed in A and B with the binary point to the right of the sign bit in A. Note that the sign of B is counted as a multiplier bit. If S is $27)_s$, one-half of the product is formed in A and B with the binary point to the right of the sign bit in A. N is determined from S in the following manner:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

For example: It is desired to obtain the full product and the MULTIPLY command is located in 01103.

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 01103 | MUP | 04000 |

If a Sequence Tag of one is used, the next command is read from 04003. With a Sequence Tag of zero the next command is read from 01203.

The last bit used of the multiplier is treated as the sign of the multiplier. When S is $26)_s$ or $27)_s$, the normal sign is used from bit 0 of B. When S is $12)_s$, bits 13 through 21 of B are used for the multiplier, with bit 12 used for the sign. Bits 0 through 11 of B are moved to 10 through 21, with the bit in 10 repeated in 9. The product is formed in A and in bits 0 through 8 of B. The line number of the address *must* be 00.

CLA Clear A (45)

Each bit in the A register is set to zero, including the sign position.

CLB Clear B (43)

Each bit in the B register is set to zero, including the sign position.

CLC Clear C (44)

Each bit in the C register is set to zero, including the sign position.

GTB Gray To Binary (41)

The GRAY TO BINARY command sends the binary representation of a Gray-coded number in A to A. The result in A is correct if its sign is positive. If its sign is negative, the inverse of the result in A should be used.

For example: It is desired to convert the Gray-coded number in the A register to binary.

Before execution of GTB, (A) are Gray 5, + 0000006. After execution of GTB, (A) are + 0000005.

It should be noted that if the contents of the A register are negative, after executing GTB, the one's complement of the correct result is obtained.

This command will also aid in parity tests on input data. If, after this command is given, the sign of A is negative, A had an odd number of ones in bits 1 through 21.

CAM Compare A and M (56)

The contents of A are compared with the contents of M and, if the two are identical, the Overflow Switch is turned on. If not, the Overflow Switch will be turned off. In either case, (A) and (M) are unaltered and command execution continues in the regular manner. All 22 positions of A and M are compared.

Transfer Commands

| | | |
|------------------------------------|------------|-------------|
| Transfer Unconditionally | TRU | (37) |
| Transfer If A Negative | TAN | (35) |
| Transfer If B Negative | TBN | (36) |
| Transfer If C Negative | TCN | (34) |
| Transfer On Overflow | TOF | (75) |
| Transfer On External Signal | TES | (77) |

TAN Transfer If A Negative (35)

If the contents of the A register are negative, the computer will take its next command from the specified address, which may be in any command line. If (A) are not negative, the next sequential command is executed. A Sequence Tag of zero is required.

TBN Transfer If B Negative (36)

If the contents of the B register are negative, the computer will take its next command from the specified address, which may be in any command line. If (B) are not negative, the next sequential command is executed. A Sequence Tag of zero is required.

TCN Transfer If C Negative (34)

If the contents of the C register are negative, the computer will take its next command from the specified address, which may be in any command line. If (C) are not negative, the next sequential command is executed. A Sequence Tag of zero is required.

TRU Transfer Unconditionally (37)

The computer will take its next command from the specified address, which may be in any command line. A Sequence Tag of one is required.

TOF Transfer On Overflow (75)

An overflow results from generating a number too large for the capacity of the arithmetic registers, specifically from the ADD, SUBTRACT, DOUBLE PRECISION ADD, and DOUBLE PRECISION SUBTRACT commands. When an overflow occurs, the Overflow Switch is turned on. The command COMPARE A AND M will also turn the Overflow Switch on if (A) are equal to (M). After execution of the command SQUARE ROOT, the Overflow Switch is turned off.

The TRANSFER ON OVERFLOW command will cause the computer to take its next command from the specified address if the Overflow Switch is on, and *then turn the switch off*. If the Overflow Switch is not on, the next sequential command is executed. Transfer may be to any command line. A Sequence Tag of zero is required for conditional transfer. A Sequence Tag of one provides an unconditional transfer and turns the Overflow Switch off, if on.

TES Transfer On External Signal (77)

This command will cause the computer to take its next command from the specified address upon sensing a signal from a source external to the computer. The nature of this signal is specified by the line number portion of the address. In the standard PB 250, line numbers 25 through 37 are used to specify the following input signals:

Lines 25 - 32: Arbitrary input signals.

Line 33: Magnetic tape reader clock input signal.

Line 34: Photo tape reader sprocket input signal.

Line 35: BREAKPOINT SWITCH input signal.

Line 36: Typewriter or paper tape reader character input complete signal.

Line 37: Typewriter not ready for an output character signal.

Line numbers 00 through 24 will provide additional input selectors which may be obtained as an option for additional arbitrary input signals. Since the line number of the address is reserved for signal specification, the effected transfer can be only to some sector in the same line as the TRANSFER ON EXTERNAL SIGNAL command.

For example:

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 02206 | TES | 02736 |

If a transfer is effected, the computer will take the next command from location 02736. If no transfer is effected, the next command will be executed from 02306. The Sequence Tag should always be zero for this command.

Loading and Storing Commands

| | | |
|-------------------------|-----|------|
| Load A | LDA | (05) |
| Load B | LDB | (06) |
| Load C | LDC | (04) |
| Load Double Precision | LDP | (07) |
| Interchange A and C | IAC | (01) |
| Interchange B and C | IBC | (02) |
| Interchange A and M | IAM | (25) |
| Rotate | ROT | (03) |
| Store A | STA | (11) |
| Store B | STB | (12) |
| Store C | STC | (10) |
| Store Double Precision | SDP | (13) |
| Move Command Line Block | MCL | (71) |
| Move Line X To Line Y | MLX | (26) |

LDA Load A (05)

The A register is cleared and the contents of M, the specified address, are read into the A register. The contents of M are not affected.

For example: It is desired to load the A register with the contents of memory location M.

| | (A) | (M) |
|-----------------------------|----------|----------|
| Before execution of command | 0423361 | -0644551 |
| After execution of command | -0644551 | -0644551 |

LDB Load B (06)

The B register is cleared and the contents of M, the specified address, are read into the B register. The contents of M are not affected.

LDC Load C (04)

The C register is cleared and the contents of M, the specified address, are read into the C register. The contents of M are not affected.

LDP Load Double Precision (07)

Both the A and B registers are cleared. The contents of M, the specified address, are read into the B register and the contents of the next memory location are read into the A register. The contents of the two memory locations are not affected.

IAC Interchange A and C (01)

The contents of the A register are loaded into the C register and the contents of the C register are loaded into the A register.

For example:

| | (A) | (C) |
|---|----------|----------|
| Before execution of INTERCHANGE A AND C | 1012556 | -3543255 |
| After execution of INTERCHANGE A AND C | -3543255 | 1012556 |

IBC Interchange B and C (02)

The contents of the B register are loaded into the C register and the contents of the C register are loaded into the B register.

IAM Interchange A and M (25)

This command interchanges the information in the line designated by the line field of the address with the information in the A register. The interchange starts in the sector following the IAM command and continues up to, but not including, the address sector number. This command results in a one-word precession of the information in the designated line. The information originally in the A register is entered into the first sector and is replaced by the information in the last sector.

ROT Rotate (03)

The contents of the A register are loaded into the C register, the contents of the C register are loaded into the B register, and the contents of the B register are loaded into the A register.

For example:

| | (A) | (B) | (C) |
|----------------------------|---------|----------|----------|
| Before execution of ROTATE | 0213405 | 1333624 | -2453201 |
| After execution of ROTATE | 1333624 | -2453201 | 0213405 |

This command requires two sectors of execution, thus an optimized ROT command would appear as follows:

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 00203 | ROT | 00300 |

with a one in bit position 8. The next command to be executed would be located in 00503.

STA Store A (11)

The contents of the A register are stored in M, the specified address. The previous contents of M are lost and the contents of the A register are not affected.

For example: Store the contents of the A register in memory location M.

| | (A) | (M) |
|-----------------------------|---------|----------|
| Before execution of STORE A | 0021213 | -6000457 |
| After execution of STORE A | 0021213 | 0021213 |

STB Store B (12)

The contents of the B register are stored in M, the specified address. The previous contents of M are lost and the contents of the B register are not affected.

STC Store C (10)

The contents of the C register are stored in M, the specified address. The previous contents of M are lost and the contents of the C register are not affected.

STD Store Double Precision (13)

This command operates on both the A and B registers. The contents of the B register are stored in M, the specified address, and the contents of the A register are stored in the memory location following M. For

example, if the specified address is 00004, the contents of B are stored in 00004 and the contents of the A register are stored in 00104. The contents of A and B are not affected and the previous contents of 00004 and 00104 are lost.

Some discussion on double precision is in order. A double precision number consists of two words, or 44 bits. The commands functioning in the double precision mode will operate on two words and treat A and B as one register, where A is the Most Significant Word (MSW) and B is the Least Significant Word (LSW). These commands are STORE DOUBLE PRECISION, LOAD DOUBLE PRECISION, DOUBLE PRECISION ADD, and DOUBLE PRECISION SUBTRACT.

MCL Move Command Line Block (71)

The contents of the first word following the MCL command and all subsequent words on that line, up to but not including the address sector number, are copied into the corresponding sector positions of the address line number.

MLX Move Line X To Line 7 (26)

This command transfers information from the line designated by the line field of the address (X) to line 07. The transfer starts in the sector following the MLX command and continues up to, but not including, the address sector number.

Logical and Shifting Commands

| | | |
|---------------------------|-----|------|
| Extend Bit Pattern | EBP | (40) |
| And M & C | AMC | (42) |
| Merge A Into C | MAC | (00) |
| And Or Combined | AOC | (46) |
| Extract Field | EXF | (47) |
| Normalize And Decrement | NAD | (20) |
| Left Shift And Decrement | LSD | (21) |
| Right Shift And Increment | RSI | (22) |
| Scale Right And Increment | SAI | (23) |
| Shift B Right | SBR | (33) |

EBP Extend Bit Pattern (40)

Starting from the right, each position of M is checked; if the position contains a zero, the corresponding position in A is unaffected. If the position contains a one, the corresponding position of A is changed so that it

is the same as the bit written to its immediate right. The (M) are unaffected. All 22 positions of A and M take part in this operation.

For example:

```
(M)      1 1 1 0 0 0 1 1 1 0 0 0
(A)      0 1 0 1 0 1 0 1 0 0 0 1
Result (in A register) 1 1 1 1 0 1 0 0 0 0 0 1
```

AMC And M & C (42)

A one is placed in each of those bit positions of B where there are ones in the corresponding positions of both C and M. Zeros are placed in all other positions of B. Neither (C) nor (M) are altered.

For example:

```
(M)      1 1 0 0
(C)      1 0 1 0
Result (in B register) 1 0 0 0
```

All 22 positions of M, B, and C take part in this operation.

MAC Merge A Into C (00)

A one is placed in each of those bit positions of C where there is a one in the corresponding positions of A or C, or in both; thus, a logical OR is accomplished. Zeros are placed in all other positions of C.

For example:

```
(A)      0 1 0 0 1 1 1 0 0 1 1 1
(C)      1 0 0 0 0 1 1 0 0 0 1 1
Result (in C register) 1 1 0 0 1 1 1 0 0 1 1 1
```

All 22 positions of A and C take part in this operation. The address field of this command is not used to specify the location of an operand but *must* contain an address number which is the sector number of the command + 1.

AOC And Or Combined (46)

Symbolically, this command is MC or $\bar{M}B$ with the result appearing in B. (The notation \bar{M} refers to the logical operation of negation.) For each one in M, the bit in the corresponding position of C is copied into B. For each zero in M, the bit in the corresponding position of B is preserved. All 22 positions of M, B, and C take part in this operation.

For example:

```
(M)      1 1 1 1 0 0 0 0
(C)      1 1 0 0 1 0 1 0
(B)      0 1 0 1 1 1 0 0
Result (in B register) 1 1 0 0 1 1 0 0
```

EXF Extract Field (47)

Symbolically, this command is $\bar{M}B$ with the result appearing in B. For each one in M, a zero is put in the corresponding position in B. For each zero in M, the bit in the corresponding position of B is preserved.

For example:

```
(M)      1 1 1 0 0 0
(B)      1 1 0 1 0 1
Result (in B register) 0 0 0 1 0 1
```

All 22 positions of M and B take part in this operation.

NAD Normalize And Decrement (20)

The address field of the NORMALIZE AND DECREMENT command is not used to specify the location of an operand, but contains an address number, N, which specifies the first sector number following the completion of execution. In executing this command, the (AB) are shifted left until one of two conditions is met:

1. $(AB) \geq \frac{1}{2}$ or $(AB) < -\frac{1}{2}$ (considering the binary point between the sign and position 1).
2. (AB) have been shifted S positions (where S is selected by the programmer).

The (C) are decremented by one for each position shifted. The sign (position 0) of A does not move, but position 0 of B takes part in the shifting. The vacated positions of B are filled with zeros. The programmer should select S large enough so as not to inhibit proper normalization. S is used in determining N in the following manner:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

For example: It is desired to normalize a number occupying the combined A and B registers. Choosing S equal to 53)_s allows for normalizing every possible number in AB but still terminates the operation if the (AB) equal zero.

| LOCATION | OP CODE | ADDRESS |
|----------|---------|---------|
| 02206 | NAD | 07600 |

If a Sequence Tag of one is used, the next command is read from 07606. With a Sequence Tag of zero the next command is read from 02306.

LSD Left Shift And Decrement (21)

The (AB) are shifted left for S positions, S being determined by the programmer. The (C) are decremented by one for each position shifted. Bits shifted past position 1 of A are lost and zeros fill the vacated right end of B. Position 0 (the sign) of A is not moved, but position 0 of B takes part in the shifting. The address field of this command is not used to specify the location of an operand, but contains an address number, N, which is determined by:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

RSI Right Shift And Increment (22)

The (AB) are shifted right for S positions, S being determined by the programmer. The (C) are incremented by one for each position shifted. The bit in the sign position of A is copied into the vacated positions of A. The bits shifted past position 21 of B are lost. Position 0 (the sign) of A is not moved but position 0 of B takes part in the shifting. The address field of this command is not used to specify the location of an operand but contains an address number, N, which is determined by:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

SAI Scale Right And Increment (23)

The (AB) are shifted right and the (C) are incremented by one for each position shifted. The operation continues until one of the two conditions is met:

1. $(C) \geq 0$.
2. (AB) are shifted S positions (where S is selected by the programmer).

The bit in the sign position of A is copied into the vacated positions of A. Position 0 (the sign) of A is not moved, but position 0 of B takes part in the shifting. S should be selected so as not to inhibit the scaling. The address field of this command is not used to specify the location of an operand but contains an address number, N, which is determined by:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

SBR Shift B Right (33)

This command functions in the same way as RSI with the following exceptions:

1. The C register is not affected by this command.
2. The A register is cleared after shifting right one position.

Bits shifted past position 21 of the B register are lost. The address field of this command is not used to specify the location of an operand but contains an address number, N, which is determined by:

$$N)_s = \text{The sector number of the command})_s + S)_s + 1)_s.$$

Control Commands

| | | |
|---------------------|------------|-------------|
| Halt | HLT | (00) |
| No Operation | NOP | (24) |

| | | |
|------------|-------------|-------------|
| HLT | Halt | (00) |
|------------|-------------|-------------|

This command stops computation and turns on the parity error indicator light on the console. To continue execution of the program, the ENABLE SWITCH and the BREAKPOINT SWITCH on the Flexowriter must be depressed. This will turn off the parity error indicator and, upon release of the ENABLE SWITCH, the program will continue.

The address field of this command is not used to specify the location of an operand, but *must* contain an address number other than the sector number of the command + 1.

| | | |
|------------|---------------------|-------------|
| NOP | No Operation | (24) |
|------------|---------------------|-------------|

This command causes the computer to continue in regular command sequence. Memory and registers are not affected.

Input-Output Commands

| | | |
|---------------------------------|------------|-------------|
| Disconnect Input Unit | DIU | (50) |
| Read Typewriter Keyboard | RTK | (51) |
| Read Paper Tape | RPT | (52) |
| Read Fast Unit | RFU | (53) |
| Load A From Input Buffer | LAI | (55) |
| Clear Input Buffer | CIB | (57) |
| Write Output Character | WOC | (6X) |
| Pulse to Specified Unit | PTU | (70) |
| Block Serial Input | BSI | (73) |
| Block Serial Output | BSO | (72) |

DIU Disconnect Input Unit (50)

The Input Buffer, which is the register for accepting input characters, is disabled. The INDICATING LIGHT of the Flexowriter, if on, is turned off.

RTK Read Typewriter Keyboard (51)

The INDICATING LIGHT on the Flexowriter is turned on, the typewriter keyboard is energized, and the Input Buffer is enabled to accept a character input. After a typewriter key has been depressed and the loading of the Input Buffer has been completed, the Flexowriter sends a signal to the computer. Upon receiving this signal, the computer de-energizes the keyboard and turns the INDICATING LIGHT off. The command TRANSFER ON EXTERNAL SIGNAL with an address line number of 36 is used to determine if the computer has received the Flexowriter signal. It is necessary to execute this command for each input character.

RPT Read Paper Tape (52)

This command causes the Flexowriter paper tape reader to read a character and enables the Input Buffer to accept this character. After the Input Buffer has been loaded, the Flexowriter sends a signal to the computer. The command TRANSFER ON EXTERNAL SIGNAL with an address line number of 36 is used to determine if the computer has received the Flexowriter signal. It is necessary to execute this command for each character input.

RFU Read Fast Unit (53)

This command will enable the Input Buffer to accept input from a fast unit such as a photo reader or a magnetic tape handler. The PULSE TO SPECIFIED UNIT command is used to select, start, and stop the selected input medium. This command differs from the other READ commands in that it is not self-disabling. The DISCONNECT INPUT UNIT command must be used to terminate the operation.

LAI Load A From Input Buffer (55)

The capacity of the Input Buffer is any character up to eight bits. This command will load the contents of the Input Buffer into positions 14 through 21 of the A register under control of a Format Word, or "mask." The Format Word must be in the same line as the LOAD A FROM INPUT BUFFER instruction, and its location is specified by the sector number of the address. Positions 0 through 13 of A are affected if the Format Word is not properly constructed. The Format Word functions as follows: In those positions of the words where there are ones, the corresponding positions bits of the Input Buffer Register are transferred to the corresponding positions of A. No other positions of A are altered. After the transfer of information to A, the Input Buffer is cleared.

CIB Clear Input Buffer (57)

Zero bits are placed in all positions of the Input Buffer. Execution will occur during the sector number of the address.

WOC Write Output Character (6X)

A single character up to eight bits may be sent to an output unit. The character is incorporated into the command and occupies positions 12 through 19 (the line number field and the right-most three positions of the OP code field). The X in the numerical code (6X) is thus determined by the output character. The output medium is determined by the command line from which the command is executed. The output is to the typewriter when the command is executed from line 05, and to the punch when executed from line 06.

Prior to executing the next command, the computer must be delayed to provide sufficient time for the execute phase of the designated output device. To accomplish this delay a predetermined number, in effect a delay number, must be loaded into C. When output begins, the (C) are decremented until they are negative, after which the next command is read. The (C) are decremented once each sector. A detailed outline of Flexowriter output is available in Chapter 3.

PTU Pulse To Specified Unit (70)

This command produces a specified combination of signals on five output signal lines and an "activate" signal on a sixth line. These signals are used to start and stop equipment external to the computer. The line number of the address field specifies the combination of signals, and the sector number defines the first sector number following the execution. The activate signal is presented in the sectors between the command location and the sector number.

BSI Block Serial Input (73)

This command loads information directly into memory at the rate of 0.5 microseconds per bit. Input information is presented to the computer in the form of a series of bits, normally from some external shift register. The shifting operation in the external register must be under computer clock control. A Format Block determines when a bit will be accepted from the input device. This Format Block is formed by the binary configuration of the information contained in that portion of the command line which begins with the sector following the BLOCK SERIAL INPUT command and continues up to, but not including, the address sector number in the command. The information entering the computer will be loaded into the line specified by the address in the command. It will occupy those positions of this line that correspond with one bits in the Format Block. Positions of this data line that correspond with zero bits in the Format Block will be loaded with zeros.

The following example illustrates the functioning of the BLOCK SERIAL INPUT command. The source from which information enters the computer will be called the External Register (ER). The line containing the BLOCK SERIAL INPUT command and the Format Block will be called the Command And Format Line (C & FL). The line receiving the data from (ER) will be called the Data Line (DL).

For example:

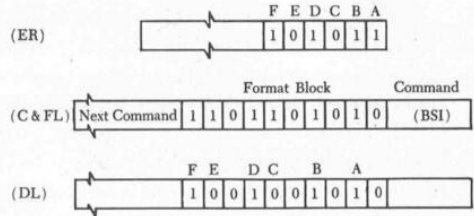


Figure 3.—Functioning of BLOCK SERIAL INPUT Command.

BSO Block Serial Output (72)

The BLOCK SERIAL OUTPUT command operates in a manner which is effectively the reverse of the BLOCK SERIAL INPUT (73) command. That is, the information in the Data Line is shifted into the External Register whenever a one appears in the Format Block. Nothing is done with the information in those positions of the Data Line which correspond to zero bits in the Format Word. For details of this command, reference is made to the description of the BLOCK SERIAL INPUT (73) command. Memory and registers are unaffected by this command.

3. INPUT-OUTPUT SYSTEM

Introduction

The PB 250 has three separate input-output systems. The first processes up to an eight-bit character from the Flexowriter, high-speed paper tape reader, or magnetic tape handlers. High-speed character input is described in the manuals for the various fast equipments. The second system is incorporated in the BLOCK SERIAL commands (72) and (73). The third system is used for control rather than for numerical information and uses the PULSE TO SPECIFIED UNIT and TRANSFER ON EXTERNAL SIGNAL commands (70) and (77).

Flexowriter

A Model FL Flexowriter is used as the control unit for the PB 250. This machine is also used to prepare, duplicate, and read tapes. The Flexowriter can be used on-line (Flexowriter under control of computer), or off-line (Flexowriter under control of operator). The general appearance and operation of the Flexowriter are similar to a standard electric typewriter. (See diagram of Flexowriter Keyboard in Figure 6.) Such features as space lever, paper release lever, platen knobs, margin release lever, ribbon position lever, margin and tab stops, and type guide, are used in exactly the same manner as for a standard typewriter. This section will be primarily concerned with the on-line mode of operation.

The Flexowriter prepares tape by punching coded holes across the width of the tape. The punched tape

is prepared manually from the keyboard (off-line) or by output commands from the computer (on-line), and is described as having channels and characters. The channels run lengthwise along the tape while characters are across its width. The PB 250 system uses six channels of an eight-channel tape; the code used in this tape is pictured in Figure 5. The Tape Reader can read and type out the information contained on a coded tape. The Flexowriter type format is illustrated in Figure 4; the upper line shows the upper case characters and the lower line the lower case characters.

When the Flexowriter is under the control of the computer, its Paper Tape Reader, Typewriter Keyboard, and Paper Tape Punch are used as input-output devices by the computer. Each typewriter character has a specific code, which is sent to the computer as a pattern of six bits.

The command READ PAPER TAPE will cause the Tape Reader to read a single character and load it into the Input Buffer of the computer. The command READ TYPEWRITER KEYBOARD will turn on the INDICATING LIGHT of the Flexowriter, after which a typewriter key must be depressed in order to load the Input Buffer. The INDICATING LIGHT is turned off by the loading operation.

The command WRITE OUTPUT CHARACTER provides information to either the typewriter or the punch. It is possible to prepare a tape with as many as eight channels by this command and read such a tape back into the computer with the READ PAPER TAPE command.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | π | √ | = | [|] | Ω | & | * | (|) | ? | " | : | / | . | , | |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | + | - | ' | ; | \$ | . | , |

Figure 4.—Flexowriter Characters.

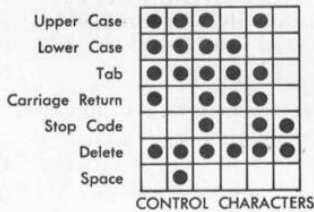
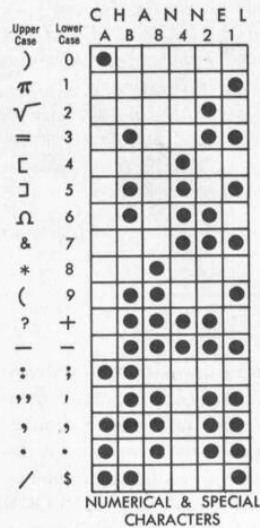
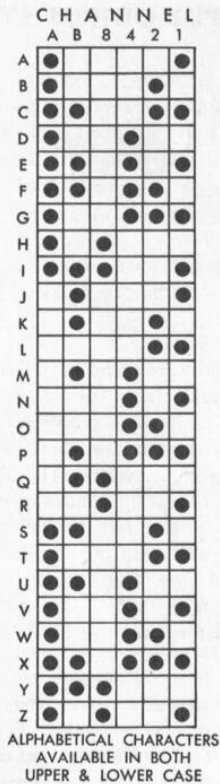


Figure 5.—Flexowriter Code

Multiple Operations

With commands READ PAPER TAPE, READ TYPEWRITER KEYBOARD, and WRITE OUTPUT CHARACTER the five kinds of multiple simultaneous operations shown below are possible with the Flexowriter.

Keyboard

The keyboard of the Flexowriter is similar to a standard typewriter keyboard and may serve many of the same purposes. The numeric, alphabetic, and symbolic keys require no explanation other than the alpha-

TABLE 2.—SIMULTANEOUS OPERATING MODES.

| Number | Type In Data | Read Input Tape | Punch Output Tape | Type Output Data | Read Output Data Into Computer for Verification | Use Type in Operation to Verify each Output Character | Compute |
|--------|--------------|-----------------|-------------------|------------------|---|---|---------|
| 1 | | X | X | | | | X |
| 2 | | X | | X | | | X |
| 3 | X | | X | | | | X |
| 4 | | | X | | X | | X |
| 5 | | | | X | | X | X |

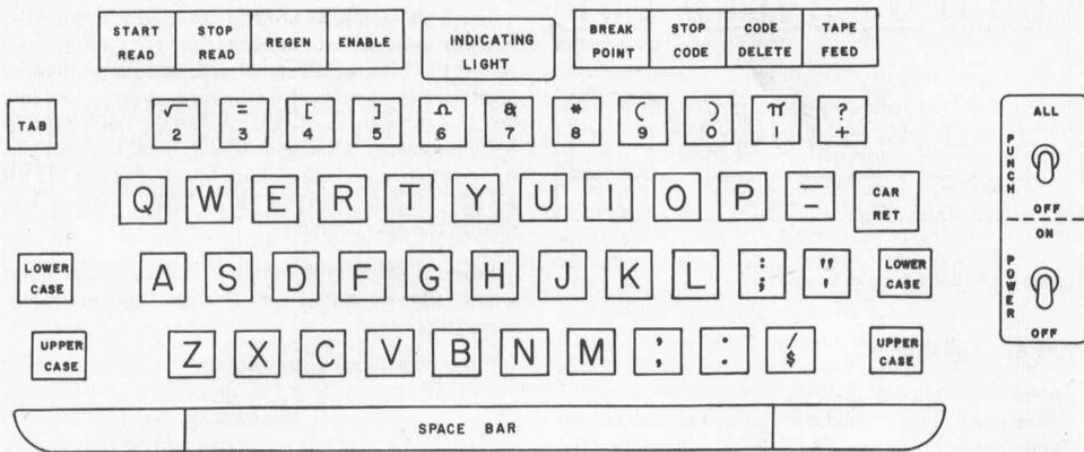


Figure 6.—Flexowriter Keyboard

numeric code for the computer as given in Figure 5. The TAB KEY, CAR RET (carriage return), LOWER CASE, UPPER CASE, and SPACE BAR are self-explanatory and are analogous to controls on a standard typewriter. The REGEN SWITCH, when depressed, permits exact duplication of the tape which is in the Flexowriter Tape Reader.

Control Switches

Certain switches and keys on the Flexowriter are used to control the computer.

ENABLE SWITCH:

1. Interrupts computation.
2. Conditions the use of other switches and keys of the Flexowriter.

BREAKPOINT SWITCH:

1. Sends signal to computer which may be tested by the command TRANSFER ON EXTERNAL SIGNAL.
 2. With ENABLE SWITCH will clear parity flip-flop (indicated by PARITY light on).
- I Key: With ENABLE SWITCH will cause the computer to execute the command in memory location 00001.
- C Key: With ENABLE SWITCH will cause the computer to cycle by one command.

The Computer Console

The console of the PB 250, shown in Figure 7, is composed of lights and switches arranged in two rows. The top row has three sets of lights: six for OPERATION, five for OPERAND, and three for COMMAND. OPERATION specifies which OP code is being executed, i.e., ADD, LOAD A, etc. Using 1 to indicate a light on, and 0 for a light off, the pattern 001100 represents the command ADD (Command 14 in octal). OPERAND specifies the line number portion of the address, and COMMAND indicates from which command line the command is being executed.

On the second row are the O'FLOW light, PARITY light, FILL switch, TEST switches, and POWER button. The O'FLOW light is on if an overflow has occurred. The PARITY light indicates a parity check error. Computation may be started by depressing the ENABLE and BREAKPOINT SWITCHES on the Flexowriter to clear the parity flip-flop. The FILL switch is used for loading a "bootstrap loading program" and the method is outlined below. The TEST switches are utilized by the Field Service Representative for maintenance of the system. The POWER button is an alternating type, turning the power to the computer on or off.

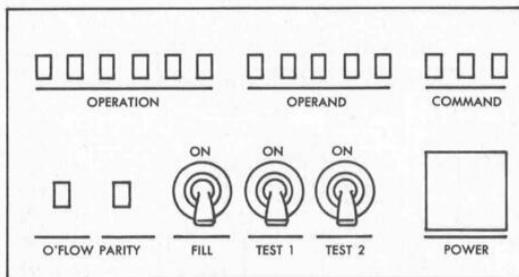


Figure 7.—PB 250 Computer Console

Loading The Computer

To start the computer after power is applied, a special one-channel service routine tape is inserted in the Tape Reader. To load the one-channel tape, the FILL switch on the front of the computer must be set on until the Tape Reader stops (about 30 seconds). When the Tape Reader stops, the service routine will be loaded with its first command in location 00001, and the parity flip-flop will be set to block computation. The ENABLE SWITCH on the Flexowriter must be depressed while the BREAKPOINT SWITCH and then the "I" key are operated to clear the parity flip-flop and set the computer to obey the first command. When the Flexowriter ENABLE SWITCH is released, the computer uses the service routine to continue loading the memory.

Flexowriter Input

The following description assumes a familiarity with the commands READ TYPEWRITER KEYBOARD, READ PAPER TAPE, LOAD A FROM INPUT BUFFER, and TRANSFER ON EXTERNAL SIGNAL, which were discussed in the preceding chapter.

The Input Buffer of the computer receives the input from the Flexowriter and can accept up to an eight-bit character. This entry is logically accumulative for each bit of the character, requiring that the buffer be cleared before each input. The Input Buffer is enabled to accept an input by either READ TYPEWRITER KEYBOARD or READ PAPER TAPE. The single character sent by the reader, or provided by the depressed typewriter key, is loaded into the buffer and upon completion of buffer loading the computer is signaled by the Flexowriter. This action requires a period

of time, during which it is possible to execute a large number of commands.

The command TRANSFER ON EXTERNAL SIGNAL is used to determine if the computer has received the Flexowriter signal and, upon a positive indication, the contents of the buffer may be loaded into the A register by the LOAD A FROM INPUT BUFFER command. After approximately 60 milliseconds, it is possible to execute another READ TYPEWRITER KEYBOARD or READ PAPER TAPE command to read in the next character.

Program steps to accomplish this character input are as follows (Dotted lines indicate intervening program steps):

| OP CODE | ADDRESS |
|---------|----------------------------|
| RPT | |
| ... | |
| ... | |
| TES | Address of LAI instruction |
| ... | |
| LAI | Address of Format Word |

It is possible, after executing READ TYPEWRITER KEYBOARD, that a typewriter key is not depressed. In anticipation of such an event, the programmer may provide commands to "wait" a period of time and then go on with computation. In this case, the commands DISCONNECT INPUT UNIT and CLEAR INPUT BUFFER should be used to turn off the Flexowriter INDICATING LIGHT and clear the buffer.

Flexowriter Output

As with the input system, the output system is capable of transmitting characters with any number of bits up to a maximum of eight. In the case of outputs, however, the character sent to the output unit is part of the WRITE OUTPUT CHARACTER command. This is illustrated in the following figure of the command word structure.

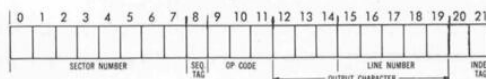


Figure 8.—Position of output character in command word.

The three right-hand bits of the OP code and the five left-hand bits of the line address contain the code for the output character. For outputs to the typewriter, the output command must be located in line 05, whereas for outputs to the punch, the output command must be located in line 06.

Just prior to the execution of the command WRITE OUTPUT CHARACTER, a "delay" number must be loaded into the C register. This delay number is used to provide the output unit with a sufficiently long signal. The value of the delay number is related to the amount of time required by a particular output device. For the Flexowriter the delay numbers are:

| | |
|------------------|-----------------------|
| Paper Tape Punch | 0002424) _s |
| Typewriter | 0003232) _s |

During execution of the output command, the delay number is decremented, and execution is completed after the (C) become negative. The next command to be executed will be located in the next sequential location of the command line if there is no sequence bit, and from the addressed sector of the command line if there is a sequence bit. The time necessary to execute the output command is (12 x delay number) micro-

seconds. The output cycle, then, consists of two commands. The first loads the delay number into C; the second, the WRITE OUTPUT CHARACTER command (with the character incorporated in the command), activates the output unit.

In order to keep the Flexowriter operating at maximum speed, an output command must be available every 105 milliseconds. For the punch an execute phase of 15 milliseconds is required, leaving 90 milliseconds for computation before the next output command is required; for the typewriter an execute phase of 20 milliseconds is required, leaving 85 milliseconds for computation. In either case, a maximum of between 3,000 and 4,000 commands can be executed between consecutive output operations while operating the Flexowriter or punch at maximum speed of approximately 10 characters per second. To prevent typing an output character before the Flexowriter has completed the previous character, the TRANSFER ON EXTERNAL SIGNAL command with an address line number of 37 should be used to test the readiness of the typewriter. In the case of the punch, however, no signal is available, and an adequate delay must be built into the program itself.

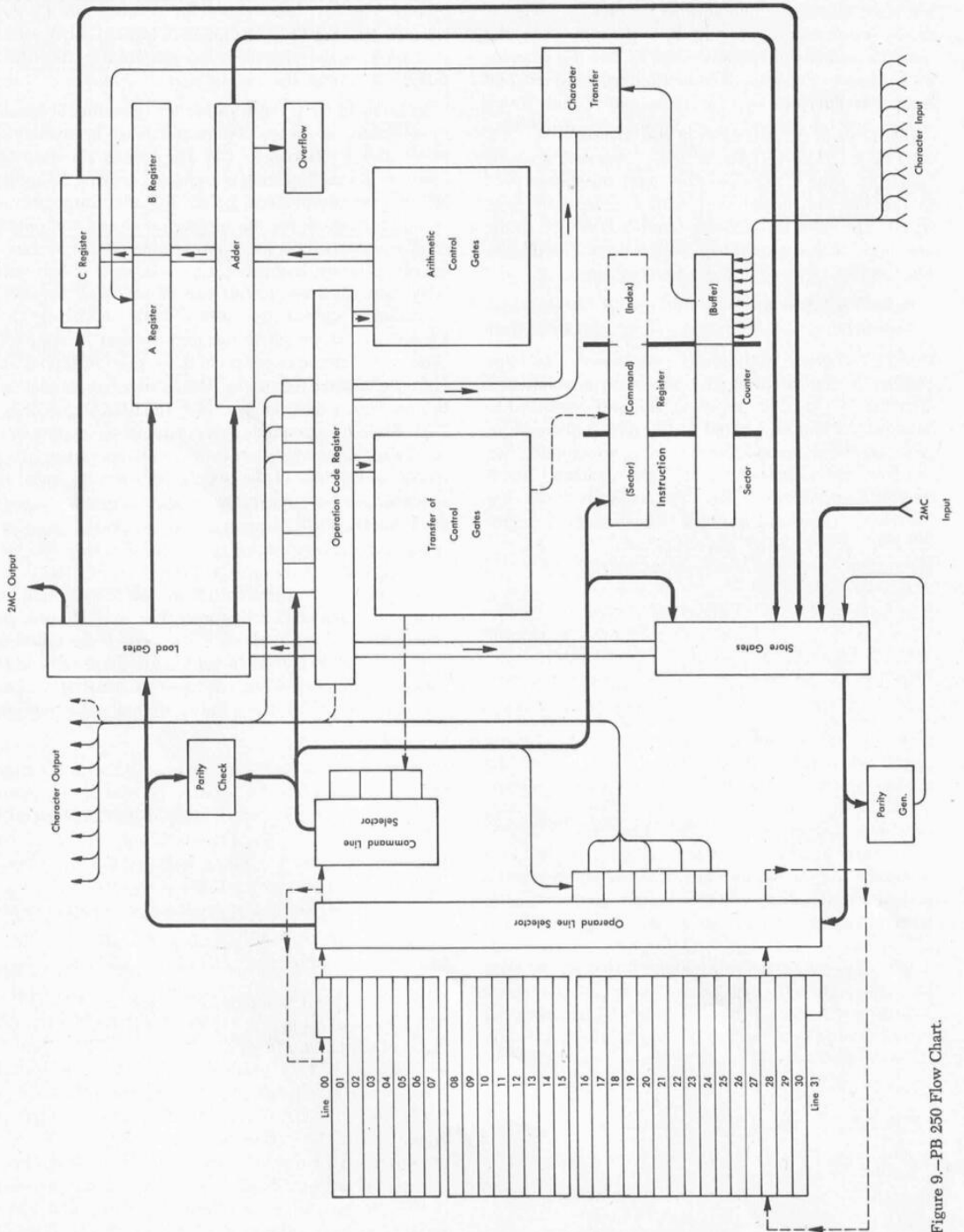


Figure 9.—PB 250 Flow Chart.

4. FUNCTIONAL COMPUTER LOGIC

The flow of information is built around the operation of the circulating memory lines and registers as indicated in the Flow Chart, Figure 9. The program commands are obtained through the Command Line Selector which is set to one of the seven long lines or to the Fast Access Line. As each command is read from a command line, its component fields are set into the proper storage elements to control the execution of the command. Depending upon the Index Tag of the command, either the line number of the address or the line number in the Index Register is entered into the Operand Line Selector. The OP code and Sequence Tag are entered into the Operation Code Register, while the sector number of the address is entered into the Instruction Register. As the command is read, the next command sector number in the Instruction Register is advanced by one count.

Depending on the command read, the computer goes through a wait to execute phase, an execute phase, and a wait to read next command phase. The timing for the execute phase is provided by comparing the address sector number in the Instruction Register with the sector numbers in the Sector Counter. The timing for the read command phase is provided by comparing the next command sector number in the Instruction Register with the sector numbers in the Sector Counter.

For load commands, the OP code routes the data from the Operand Line Selector into one of the circulating registers, A, B, or C. The flow for a 2 mc serial output follows a similar path. For store commands, the OP code selects one of the registers as a source of data

to be stored through the Operand Line Selector. A 2 mc serial input also follows this type of path. A Parity Code is generated with each store command and checked with each load command.

Addition and subtraction into the A register or the A and B register pair is accomplished by routing the data through the Adder. These operations are checked for Overflow. For other arithmetic operations, such as multiplication, the OP code conditions the Arithmetic Control Gates to execute the prescribed operations on the contents of A, B, and C.

For several OP codes, a transfer of control may be conditional, depending upon the signs of A, B, or C or an Overflow or upon external inputs. If the conditions of the transfer are met, the address in the command is used to change the Command Line Selector as well as the next command sector number in the Instruction Register.

An eight-bit Buffer is provided in the Sector Counter for input characters. These may be transferred upon command to the A register. An eight-bit character output is provided by the Operand Line Selector and three stages of the Operation Code Register. An activate signal is obtained from the Operation Code Register to indicate the transmission of each output character.

A load or store command with an address indicating line 37)8 operates on the Index Register portion of the Instruction Register. The data in this portion of the Instruction Register is called into the Operand Selector by an Index Tag of one in the command.

APPENDIX A

Binary-Octal Number Systems

Numerical Systems

Any number can be represented as the sum of a group of terms, having the form $a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \dots + a_1 b^1 + a_0 b^0$, where $b > 1$ and $0 \leq a_i \leq (b-1)$. The integer b is called the base, or radix of the particular numerical system, while a represents the range of numerical values in that system.

Decimal System

The numerical system of radix 10 is called the decimal system. In this case, numerical values are specified by combining powers of ten in the form of $a_n (10^n) + a_{n-1} (10^{n-1}) + a_{n-2} (10^{n-2}) + \dots + a_1 (10^1) + a_0 (10^0)$. The usual practice, when writing decimal numbers, is to omit the powers of ten and write out only the values of a . For example, consider the decimal number 1875. This number actually represents $1 (10^3) + 8 (10^2) + 7 (10^1) + 5 (10^0)$ but for the sake of convenience is merely written as 1875, with the position of the particular decimal digit indicating with which power of ten the digit is associated.

Binary System

The PB 250 operates in the binary or radix 2 mode; hence, to understand the operation of the computer, an understanding of binary arithmetic is essential.

Here, numerical values are specified by combining powers of two in the form $a_n (2^n) + a_{n-1} (2^{n-1}) + a_{n-2} (2^{n-2}) + \dots + a_1 (2^1) + a_0 (2^0)$. As before, the usual practice when writing binary numbers is to omit the powers of two and write out only the values of the a terms. For example, consider the binary number 1011. This number actually represents $1 (2^3) + 0 (2^2) + 1 (2^1) + 1 (2^0)$ but for the sake of convenience is merely written as 1011, with the position of the particular binary digit (or bit) indicating with which power of two the digit is associated.

Binary-to-Decimal Conversion

The relationship between binary and decimal notation becomes obvious if, in the above binary example, the

powers of two are actually added up, giving a result of $8 + 0 + 2 + 1 = 11)_{10}$; therefore, $(1011)_2 = 11)_{10}$.

NOTE

The notation $)_{10}$ indicates that the particular number is in the radix 10, or decimal system. The notation $)_2$ indicates radix 2, or binary system, and $)_8$ indicates radix 8, or octal system.

Any decimal number can be represented in binary by the summation of appropriate powers of two. Table 3 shows how any decimal number up to 12 can be written in binary form. The relation between the number of bits, n , and the highest decimal digit, N , that these bits can represent is $N = (2^n - 1)$. Thus, in the 4-bit Table 3, the highest decimal value possible is $(2^4 - 1)$ or $15)_{10}$ which would appear in Table 3 as $1111)_2$.

TABLE 3.—DECIMAL NUMBER EQUIVALENTS

| Decimal | Binary | | | |
|---------|--------|-------|-------|-------|
| | 2^3 | 2^2 | 2^1 | 2^0 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |

Decimal-to-Binary Conversion

One method of converting decimal numbers to binary is to divide the decimal number repeatedly by two, ignoring the remainder each time, until the decimal number is reduced to one or zero. Then, reading from the end, the successive remainders will be the binary representation of the original decimal number.

Octal Notation

As the decimal number increases in magnitude, its

binary representation becomes rather long and inconvenient to handle; hence, the octal system (radix 8) is used as a convenient shorthand notation for binary numbers.

Binary-to-Octal Conversion

Since $2^3 = 8$, it can be seen that three binary digits are represented by one octal digit. The binary-to-octal conversion is performed by merely grouping the binary number into 3-bit units, starting from the binary point, and interpreting each unit individually. For instance, the number 101011010_2 becomes

$$\begin{array}{ccc} \underbrace{101} & \underbrace{011} & \underbrace{010} \\ 5 & 3 & 2 \end{array}$$

or 532_8 . Conversely, it can be seen that any octal number can be translated to binary by simply writing the binary equivalent of each digit. For instance, the number 612_8 becomes

$$\begin{array}{ccc} \underbrace{6} & \underbrace{1} & \underbrace{2} \\ 110 & 001 & 010 \end{array}$$

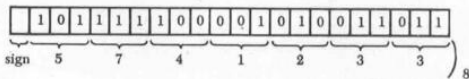
or 110001010_2 .

PB 250 Data Word

A data word in the PB 250 consists of 21 bits plus sign; therefore, the highest number that can be stored in one memory word is $(2^{21} - 1)$ or

- a) $111\ 111\ 111\ 111\ 111\ 111\ 111_2$
- b) $7, 777, 777_8$
- c) $2, 097, 151_{10}$

A typical data word might appear as shown below.
(Note the convenience of the octal notation.)



Binary Complementary Arithmetic

Certain computer operations, such as subtraction or the manipulation of negative numbers, are performed in the computer by using the complement of the particular number. An understanding of complementary

arithmetic is therefore important as an aid in understanding computer operation.

The 1's complement of a binary number is defined as the number that must be added to the original number to give a result consisting of all 1's. The 1's complement is obtained by simply inverting, i.e., by changing all 1's to 0's and changing all 0's to 1's in the given binary number. For example, the 1's complement of 1010110_2 would be 0101001_2 .

The 2's (or "true") complement of a binary number is formed by first finding the 1's complement of the number and then adding 1 to the least significant bit position.

For example, the 2's complement of 1010110_2 would be the 1's complement (0101001_2) plus 1, or 0101010_2 .

Note how negative numbers are manipulated using complementary arithmetic. For example, add 111_2 to -010_2 . Replacing the negative number by its 2's complement makes the problem $111 + 110$. Adding (neglecting overflow), gives 101_2 , which is the correct answer.

Similar examples are indicated below in decimal, binary, and complemented binary forms. The complemented binary form has a leading zero, to indicate positive numbers, which becomes a leading one when complemented for negative numbers. A negative answer appears in complemented form with a leading one.

| | Decimal | Binary | Complemented Binary |
|----|---------|--------|---------------------|
| a) | +12 | +1100 | 0 1100 |
| | -04 | -0100 | 1 1100 |
| | <hr/> | <hr/> | <hr/> |
| | +08 | +1000 | 0 1000 |
| b) | +10 | +1010 | 0 1010 |
| | -10 | -1010 | 1 1010 |
| | <hr/> | <hr/> | <hr/> |
| | +00 | +0000 | 0 0000 |
| c) | +12 | +1100 | 0 1100 |
| | -14 | -1110 | 1 0010 |
| | <hr/> | <hr/> | <hr/> |
| | -02 | -0010 | 1 1110 |

APPENDIX B

Table of Powers of 2

| 2^n | n | 2^{-n} |
|-------------------|-----|---|
| 1 | 0 | 1.0 |
| 2 | 1 | 0.5 |
| 4 | 2 | 0.25 |
| 8 | 3 | 0.125 |
| 16 | 4 | 0.062 5 |
| 32 | 5 | 0.031 25 |
| 64 | 6 | 0.015 625 |
| 128 | 7 | 0.007 812 5 |
| 256 | 8 | 0.003 906 25 |
| 512 | 9 | 0.001 953 125 |
| 1 024 | 10 | 0.000 976 562 5 |
| 2 048 | 11 | 0.000 488 281 25 |
| 4 096 | 12 | 0.000 244 140 625 |
| 8 192 | 13 | 0.000 122 070 312 5 |
| 16 384 | 14 | 0.000 061 035 156 25 |
| 32 768 | 15 | 0.000 030 517 578 125 |
| 65 536 | 16 | 0.000 015 258 789 062 5 |
| 131 072 | 17 | 0.000 007 629 394 531 25 |
| 262 144 | 18 | 0.000 003 814 697 265 625 |
| 524 288 | 19 | 0.000 001 907 348 632 812 5 |
| 1 048 576 | 20 | 0.000 000 953 674 316 406 25 |
| 2 097 152 | 21 | 0.000 000 476 837 158 203 125 |
| 4 194 304 | 22 | 0.000 000 238 418 579 101 562 5 |
| 8 388 608 | 23 | 0.000 000 119 209 289 550 781 25 |
| 16 777 216 | 24 | 0.000 000 059 604 644 775 390 625 |
| 33 554 432 | 25 | 0.000 000 029 802 322 387 695 312 5 |
| 67 108 864 | 26 | 0.000 000 014 901 161 193 847 656 25 |
| 134 217 728 | 27 | 0.000 000 007 450 580 596 923 828 125 |
| 268 435 456 | 28 | 0.000 000 003 725 290 298 461 914 062 5 |
| 536 870 912 | 29 | 0.000 000 001 862 645 149 230 957 031 25 |
| 1 073 741 824 | 30 | 0.000 000 000 931 322 574 615 478 515 625 |
| 2 147 483 648 | 31 | 0.000 000 000 465 661 287 307 739 257 812 5 |
| 4 294 967 296 | 32 | 0.000 000 000 232 830 643 653 869 628 906 25 |
| 8 589 934 592 | 33 | 0.000 000 000 116 415 321 826 934 814 453 125 |
| 17 179 869 184 | 34 | 0.000 000 000 058 207 660 913 467 407 226 562 5 |
| 34 359 738 368 | 35 | 0.000 000 000 029 103 830 456 733 703 613 281 25 |
| 68 719 476 736 | 36 | 0.000 000 000 014 551 915 228 366 851 806 640 625 |
| 137 438 953 472 | 37 | 0.000 000 000 007 275 957 614 183 425 903 320 312 5 |
| 274 877 906 944 | 38 | 0.000 000 000 003 637 978 807 091 712 951 660 156 25 |
| 549 755 813 888 | 39 | 0.000 000 000 001 818 989 403 545 856 475 830 078 125 |
| 1 099 511 627 796 | 40 | |
| 2 199 023 255 552 | 41 | |
| 4 398 046 511 104 | 42 | |
| 8 796 093 022 208 | 43 | |

APPENDIX C

Octal-Decimal Integer Conversion Table

0000 0000
to to
0777 0511
(Octal) (Decimal)

Octal Decimal

10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 0000 | 0000 | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 |
| 0010 | 0008 | 0009 | 0010 | 0011 | 0012 | 0013 | 0014 | 0015 |
| 0020 | 0016 | 0017 | 0018 | 0019 | 0020 | 0021 | 0022 | 0023 |
| 0030 | 0024 | 0025 | 0026 | 0027 | 0028 | 0029 | 0030 | 0031 |
| 0040 | 0032 | 0033 | 0034 | 0035 | 0036 | 0037 | 0038 | 0039 |
| 0050 | 0040 | 0041 | 0042 | 0043 | 0044 | 0045 | 0046 | 0047 |
| 0060 | 0048 | 0049 | 0050 | 0051 | 0052 | 0053 | 0054 | 0055 |
| 0070 | 0056 | 0057 | 0058 | 0059 | 0060 | 0061 | 0062 | 0063 |
| 0100 | 0064 | 0065 | 0066 | 0067 | 0068 | 0069 | 0070 | 0071 |
| 0110 | 0072 | 0073 | 0074 | 0075 | 0076 | 0077 | 0078 | 0079 |
| 0120 | 0080 | 0081 | 0082 | 0083 | 0084 | 0085 | 0086 | 0087 |
| 0130 | 0088 | 0089 | 0090 | 0091 | 0092 | 0093 | 0094 | 0095 |
| 0140 | 0096 | 0097 | 0098 | 0099 | 0100 | 0101 | 0102 | 0103 |
| 0150 | 0104 | 0105 | 0106 | 0107 | 0108 | 0109 | 0110 | 0111 |
| 0160 | 0112 | 0113 | 0114 | 0115 | 0116 | 0117 | 0118 | 0119 |
| 0170 | 0120 | 0121 | 0122 | 0123 | 0124 | 0125 | 0126 | 0127 |
| 0200 | 0128 | 0129 | 0130 | 0131 | 0132 | 0133 | 0134 | 0135 |
| 0210 | 0136 | 0137 | 0138 | 0139 | 0140 | 0141 | 0142 | 0143 |
| 0220 | 0144 | 0145 | 0146 | 0147 | 0148 | 0149 | 0150 | 0151 |
| 0230 | 0152 | 0153 | 0154 | 0155 | 0156 | 0157 | 0158 | 0159 |
| 0240 | 0160 | 0161 | 0162 | 0163 | 0164 | 0165 | 0166 | 0167 |
| 0250 | 0168 | 0169 | 0170 | 0171 | 0172 | 0173 | 0174 | 0175 |
| 0260 | 0176 | 0177 | 0178 | 0179 | 0180 | 0181 | 0182 | 0183 |
| 0270 | 0184 | 0185 | 0186 | 0187 | 0188 | 0189 | 0190 | 0191 |
| 0300 | 0192 | 0193 | 0194 | 0195 | 0196 | 0197 | 0198 | 0199 |
| 0310 | 0200 | 0201 | 0202 | 0203 | 0204 | 0205 | 0206 | 0207 |
| 0320 | 0208 | 0209 | 0210 | 0211 | 0212 | 0213 | 0214 | 0215 |
| 0330 | 0216 | 0217 | 0218 | 0219 | 0220 | 0221 | 0222 | 0223 |
| 0340 | 0224 | 0225 | 0226 | 0227 | 0228 | 0229 | 0230 | 0231 |
| 0350 | 0232 | 0233 | 0234 | 0235 | 0236 | 0237 | 0238 | 0239 |
| 0360 | 0240 | 0241 | 0242 | 0243 | 0244 | 0245 | 0246 | 0247 |
| 0370 | 0248 | 0249 | 0250 | 0251 | 0252 | 0253 | 0254 | 0255 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 0400 | 0256 | 0257 | 0258 | 0259 | 0260 | 0261 | 0262 | 0263 |
| 0410 | 0264 | 0265 | 0266 | 0267 | 0268 | 0269 | 0270 | 0271 |
| 0420 | 0272 | 0273 | 0274 | 0275 | 0276 | 0277 | 0278 | 0279 |
| 0430 | 0280 | 0281 | 0282 | 0283 | 0284 | 0285 | 0286 | 0287 |
| 0440 | 0288 | 0289 | 0290 | 0291 | 0292 | 0293 | 0294 | 0295 |
| 0450 | 0296 | 0297 | 0298 | 0299 | 0300 | 0301 | 0302 | 0303 |
| 0460 | 0304 | 0305 | 0306 | 0307 | 0308 | 0309 | 0310 | 0311 |
| 0470 | 0312 | 0313 | 0314 | 0315 | 0316 | 0317 | 0318 | 0319 |
| 0500 | 0320 | 0321 | 0322 | 0323 | 0324 | 0325 | 0326 | 0327 |
| 0510 | 0328 | 0329 | 0330 | 0331 | 0332 | 0333 | 0334 | 0335 |
| 0520 | 0336 | 0337 | 0338 | 0339 | 0340 | 0341 | 0342 | 0343 |
| 0530 | 0344 | 0345 | 0346 | 0347 | 0348 | 0349 | 0350 | 0351 |
| 0540 | 0352 | 0353 | 0354 | 0355 | 0356 | 0357 | 0358 | 0359 |
| 0550 | 0360 | 0361 | 0362 | 0363 | 0364 | 0365 | 0366 | 0367 |
| 0560 | 0368 | 0369 | 0370 | 0371 | 0372 | 0373 | 0374 | 0375 |
| 0570 | 0376 | 0377 | 0378 | 0379 | 0380 | 0381 | 0382 | 0383 |
| 0600 | 0384 | 0385 | 0386 | 0387 | 0388 | 0389 | 0390 | 0391 |
| 0610 | 0392 | 0393 | 0394 | 0395 | 0396 | 0397 | 0398 | 0399 |
| 0620 | 0400 | 0401 | 0402 | 0403 | 0404 | 0405 | 0406 | 0407 |
| 0630 | 0408 | 0409 | 0410 | 0411 | 0412 | 0413 | 0414 | 0415 |
| 0640 | 0416 | 0417 | 0418 | 0419 | 0420 | 0421 | 0422 | 0423 |
| 0650 | 0424 | 0425 | 0426 | 0427 | 0428 | 0429 | 0430 | 0431 |
| 0660 | 0432 | 0433 | 0434 | 0435 | 0436 | 0437 | 0438 | 0439 |
| 0670 | 0440 | 0441 | 0442 | 0443 | 0444 | 0445 | 0446 | 0447 |
| 0700 | 0448 | 0449 | 0450 | 0451 | 0452 | 0453 | 0454 | 0455 |
| 0710 | 0456 | 0457 | 0458 | 0459 | 0460 | 0461 | 0462 | 0463 |
| 0720 | 0464 | 0465 | 0466 | 0467 | 0468 | 0469 | 0470 | 0471 |
| 0730 | 0472 | 0473 | 0474 | 0475 | 0476 | 0477 | 0478 | 0479 |
| 0740 | 0480 | 0481 | 0482 | 0483 | 0484 | 0485 | 0486 | 0487 |
| 0750 | 0488 | 0489 | 0490 | 0491 | 0492 | 0493 | 0494 | 0495 |
| 0760 | 0496 | 0497 | 0498 | 0499 | 0500 | 0501 | 0502 | 0503 |
| 0770 | 0504 | 0505 | 0506 | 0507 | 0508 | 0509 | 0510 | 0511 |

1000 0512
to to
1777 1023
(Octal) (Decimal)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 1000 | 0512 | 0513 | 0514 | 0515 | 0516 | 0517 | 0518 | 0519 |
| 1010 | 0520 | 0521 | 0522 | 0523 | 0524 | 0525 | 0526 | 0527 |
| 1020 | 0528 | 0529 | 0530 | 0531 | 0532 | 0533 | 0534 | 0535 |
| 1030 | 0536 | 0537 | 0538 | 0539 | 0540 | 0541 | 0542 | 0543 |
| 1040 | 0544 | 0545 | 0546 | 0547 | 0548 | 0549 | 0550 | 0551 |
| 1050 | 0552 | 0553 | 0554 | 0555 | 0556 | 0557 | 0558 | 0559 |
| 1060 | 0560 | 0561 | 0562 | 0563 | 0564 | 0565 | 0566 | 0567 |
| 1070 | 0568 | 0569 | 0570 | 0571 | 0572 | 0573 | 0574 | 0575 |
| 1100 | 0576 | 0577 | 0578 | 0579 | 0580 | 0581 | 0582 | 0583 |
| 1110 | 0584 | 0585 | 0586 | 0587 | 0588 | 0589 | 0590 | 0591 |
| 1120 | 0592 | 0593 | 0594 | 0595 | 0596 | 0597 | 0598 | 0599 |
| 1130 | 0600 | 0601 | 0602 | 0603 | 0604 | 0605 | 0606 | 0607 |
| 1140 | 0608 | 0609 | 0610 | 0611 | 0612 | 0613 | 0614 | 0615 |
| 1150 | 0616 | 0617 | 0618 | 0619 | 0620 | 0621 | 0622 | 0623 |
| 1160 | 0624 | 0625 | 0626 | 0627 | 0628 | 0629 | 0630 | 0631 |
| 1170 | 0632 | 0633 | 0634 | 0635 | 0636 | 0637 | 0638 | 0639 |
| 1200 | 0640 | 0641 | 0642 | 0643 | 0644 | 0645 | 0646 | 0647 |
| 1210 | 0648 | 0649 | 0650 | 0651 | 0652 | 0653 | 0654 | 0655 |
| 1220 | 0656 | 0657 | 0658 | 0659 | 0660 | 0661 | 0662 | 0663 |
| 1230 | 0664 | 0665 | 0666 | 0667 | 0668 | 0669 | 0670 | 0671 |
| 1240 | 0672 | 0673 | 0674 | 0675 | 0676 | 0677 | 0678 | 0679 |
| 1250 | 0680 | 0681 | 0682 | 0683 | 0684 | 0685 | 0686 | 0687 |
| 1260 | 0688 | 0689 | 0690 | 0691 | 0692 | 0693 | 0694 | 0695 |
| 1270 | 0696 | 0697 | 0698 | 0699 | 0700 | 0701 | 0702 | 0703 |
| 1300 | 0704 | 0705 | 0706 | 0707 | 0708 | 0709 | 0710 | 0711 |
| 1310 | 0712 | 0713 | 0714 | 0715 | 0716 | 0717 | 0718 | 0719 |
| 1320 | 0720 | 0721 | 0722 | 0723 | 0724 | 0725 | 0726 | 0727 |
| 1330 | 0728 | 0729 | 0730 | 0731 | 0732 | 0733 | 0734 | 0735 |
| 1340 | 0736 | 0737 | 0738 | 0739 | 0740 | 0741 | 0742 | 0743 |
| 1350 | 0744 | 0745 | 0746 | 0747 | 0748 | 0749 | 0750 | 0751 |
| 1360 | 0752 | 0753 | 0754 | 0755 | 0756 | 0757 | 0758 | 0759 |
| 1370 | 0760 | 0761 | 0762 | 0763 | 0764 | 0765 | 0766 | 0767 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 1400 | 0768 | 0769 | 0770 | 0771 | 0772 | 0773 | 0774 | 0775 |
| 1410 | 0776 | 0777 | 0778 | 0779 | 0780 | 0781 | 0782 | 0783 |
| 1420 | 0784 | 0785 | 0786 | 0787 | 0788 | 0789 | 0790 | 0791 |
| 1430 | 0792 | 0793 | 0794 | 0795 | 0796 | 0797 | 0798 | 0799 |
| 1440 | 0800 | 0801 | 0802 | 0803 | 0804 | 0805 | 0806 | 0807 |
| 1450 | 0808 | 0809 | 0810 | 0811 | 0812 | 0813 | 0814 | 0815 |
| 1460 | 0816 | 0817 | 0818 | 0819 | 0820 | 0821 | 0822 | 0823 |
| 1470 | 0824 | 0825 | 0826 | 0827 | 0828 | 0829 | 0830 | 0831 |
| 1500 | 0832 | 0833 | 0834 | 0835 | 0836 | 0837 | 0838 | 0839 |
| 1510 | 0840 | 0841 | 0842 | 0843 | 0844 | 0845 | 0846 | 0847 |
| 1520 | 0848 | 0849 | 0850 | 0851 | 0852 | 0853 | 0854 | 0855 |
| 1530 | 0856 | 0857 | 0858 | 0859 | 0860 | 0861 | 0862 | 0863 |
| 1540 | 0864 | 0865 | 0866 | 0867 | 0868 | 0869 | 0870 | 0871 |
| 1550 | 0872 | 0873 | 0874 | 0875 | 0876 | 0877 | 0878 | 0879 |
| 1560 | 0880 | 0881 | 0882 | 0883 | 0884 | 0885 | 0886 | 0887 |
| 1570 | 0888 | 0889 | 0890 | 0891 | 0892 | 0893 | 0894 | 0895 |
| 1600 | 0896 | 0897 | 0898 | 0899 | 0900 | 0901 | 0902 | 0903 |
| 1610 | 0904 | 0905 | 0906 | 0907 | 0908 | 0909 | 0910 | 0911 |
| 1620 | 0912 | 0913 | 0914 | 0915 | 0916 | 0917 | 0918 | 0919 |
| 1630 | 0920 | 0921 | 0922 | 0923 | 0924 | 0925 | 0926 | 0927 |
| 1640 | 0928 | 0929 | 0930 | 0931 | 0932 | 0933 | 0934 | 0935 |
| 1650 | 0936 | 0937 | 0938 | 0939 | 0940 | 0941 | 0942 | 0943 |
| 1660 | 0944 | 0945 | 0946 | 0947 | 0948 | 0949 | 0950 | 0951 |
| 1670 | 0952 | 0953 | 0954 | 0955 | 0956 | 0957 | 0958 | 0959 |
| 1700 | 0960 | 0961 | 0962 | 0963 | 0964 | 0965 | 0966 | 0967 |
| 1710 | 0968 | 0969 | 0970 | 0971 | 0972 | 0973 | 0974 | 0975 |
| 1720 | 0976 | 0977 | 0978 | 0979 | 0980 | 0981 | 0982 | 0983 |
| 1730 | 0984 | 0985 | 0986 | 0987 | 0988 | 0989 | 0990 | 0991 |
| 1740 | 0992 | 0993 | 0994 | 0995 | 0996 | 0997 | 0998 | 0999 |
| 1750 | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 |
| 1760 | 1008 | 1009 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 |
| 1770 | 1016 | 1017 | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 |

Octal-Decimal Integer Conversion Table

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 2000 | 1024 | 1025 | 1026 | 1027 | 1028 | 1029 | 1030 | 1031 |
| 2010 | 1032 | 1033 | 1034 | 1035 | 1036 | 1037 | 1038 | 1039 |
| 2020 | 1040 | 1041 | 1042 | 1043 | 1044 | 1045 | 1046 | 1047 |
| 2030 | 1048 | 1049 | 1050 | 1051 | 1052 | 1053 | 1054 | 1055 |
| 2040 | 1056 | 1057 | 1058 | 1059 | 1060 | 1061 | 1062 | 1063 |
| 2050 | 1064 | 1065 | 1066 | 1067 | 1068 | 1069 | 1070 | 1071 |
| 2060 | 1072 | 1073 | 1074 | 1075 | 1076 | 1077 | 1078 | 1079 |
| 2070 | 1080 | 1081 | 1082 | 1083 | 1084 | 1085 | 1086 | 1087 |
| 2100 | 1088 | 1089 | 1090 | 1091 | 1092 | 1093 | 1094 | 1095 |
| 2110 | 1096 | 1097 | 1098 | 1099 | 1100 | 1101 | 1102 | 1103 |
| 2120 | 1104 | 1105 | 1106 | 1107 | 1108 | 1109 | 1110 | 1111 |
| 2130 | 1112 | 1113 | 1114 | 1115 | 1116 | 1117 | 1118 | 1119 |
| 2140 | 1120 | 1121 | 1122 | 1123 | 1124 | 1125 | 1126 | 1127 |
| 2150 | 1128 | 1129 | 1130 | 1131 | 1132 | 1133 | 1134 | 1135 |
| 2160 | 1136 | 1137 | 1138 | 1139 | 1140 | 1141 | 1142 | 1143 |
| 2170 | 1144 | 1145 | 1146 | 1147 | 1148 | 1149 | 1150 | 1151 |
| 2200 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 |
| 2210 | 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 |
| 2220 | 1168 | 1169 | 1170 | 1171 | 1172 | 1173 | 1174 | 1175 |
| 2230 | 1176 | 1177 | 1178 | 1179 | 1180 | 1181 | 1182 | 1183 |
| 2240 | 1184 | 1185 | 1186 | 1187 | 1188 | 1189 | 1190 | 1191 |
| 2250 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 |
| 2260 | 1200 | 1201 | 1202 | 1203 | 1204 | 1205 | 1206 | 1207 |
| 2270 | 1208 | 1209 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 |
| 2300 | 1216 | 1217 | 1218 | 1219 | 1220 | 1221 | 1222 | 1223 |
| 2310 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 | 1230 | 1231 |
| 2320 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 |
| 2330 | 1240 | 1241 | 1242 | 1243 | 1244 | 1245 | 1246 | 1247 |
| 2340 | 1248 | 1249 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 |
| 2350 | 1256 | 1257 | 1258 | 1259 | 1260 | 1261 | 1262 | 1263 |
| 2360 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1270 | 1271 |
| 2370 | 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 3000 | 1536 | 1537 | 1538 | 1539 | 1540 | 1541 | 1542 | 1543 |
| 3010 | 1544 | 1545 | 1546 | 1547 | 1548 | 1549 | 1550 | 1551 |
| 3020 | 1552 | 1553 | 1554 | 1555 | 1556 | 1557 | 1558 | 1559 |
| 3030 | 1560 | 1561 | 1562 | 1563 | 1564 | 1565 | 1566 | 1567 |
| 3040 | 1568 | 1569 | 1570 | 1571 | 1572 | 1573 | 1574 | 1575 |
| 3050 | 1576 | 1577 | 1578 | 1579 | 1580 | 1581 | 1582 | 1583 |
| 3060 | 1584 | 1585 | 1586 | 1587 | 1588 | 1589 | 1590 | 1591 |
| 3070 | 1592 | 1593 | 1594 | 1595 | 1596 | 1597 | 1598 | 1599 |
| 3100 | 1600 | 1601 | 1602 | 1603 | 1604 | 1605 | 1606 | 1607 |
| 3110 | 1608 | 1609 | 1610 | 1611 | 1612 | 1613 | 1614 | 1615 |
| 3120 | 1616 | 1617 | 1618 | 1619 | 1620 | 1621 | 1622 | 1623 |
| 3130 | 1624 | 1625 | 1626 | 1627 | 1628 | 1629 | 1630 | 1631 |
| 3140 | 1632 | 1633 | 1634 | 1635 | 1636 | 1637 | 1638 | 1639 |
| 3150 | 1640 | 1641 | 1642 | 1643 | 1644 | 1645 | 1646 | 1647 |
| 3160 | 1648 | 1649 | 1650 | 1651 | 1652 | 1653 | 1654 | 1655 |
| 3170 | 1656 | 1657 | 1658 | 1659 | 1660 | 1661 | 1662 | 1663 |
| 3200 | 1664 | 1665 | 1666 | 1667 | 1668 | 1669 | 1670 | 1671 |
| 3210 | 1672 | 1673 | 1674 | 1675 | 1676 | 1677 | 1678 | 1679 |
| 3220 | 1680 | 1681 | 1682 | 1683 | 1684 | 1685 | 1686 | 1687 |
| 3230 | 1688 | 1689 | 1690 | 1691 | 1692 | 1693 | 1694 | 1695 |
| 3240 | 1696 | 1697 | 1698 | 1699 | 1700 | 1701 | 1702 | 1703 |
| 3250 | 1704 | 1705 | 1706 | 1707 | 1708 | 1709 | 1710 | 1711 |
| 3260 | 1712 | 1713 | 1714 | 1715 | 1716 | 1717 | 1718 | 1719 |
| 3270 | 1720 | 1721 | 1722 | 1723 | 1724 | 1725 | 1726 | 1727 |
| 3300 | 1728 | 1729 | 1730 | 1731 | 1732 | 1733 | 1734 | 1735 |
| 3310 | 1736 | 1737 | 1738 | 1739 | 1740 | 1741 | 1742 | 1743 |
| 3320 | 1744 | 1745 | 1746 | 1747 | 1748 | 1749 | 1750 | 1751 |
| 3330 | 1752 | 1753 | 1754 | 1755 | 1756 | 1757 | 1758 | 1759 |
| 3340 | 1760 | 1761 | 1762 | 1763 | 1764 | 1765 | 1766 | 1767 |
| 3350 | 1768 | 1769 | 1770 | 1771 | 1772 | 1773 | 1774 | 1775 |
| 3360 | 1776 | 1777 | 1778 | 1779 | 1780 | 1781 | 1782 | 1783 |
| 3370 | 1784 | 1785 | 1786 | 1787 | 1788 | 1789 | 1790 | 1791 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 2400 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 | 1286 | 1287 |
| 2410 | 1288 | 1289 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 |
| 2420 | 1296 | 1297 | 1298 | 1299 | 1300 | 1301 | 1302 | 1303 |
| 2430 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1310 | 1311 |
| 2440 | 1312 | 1313 | 1314 | 1315 | 1316 | 1317 | 1318 | 1319 |
| 2450 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 |
| 2460 | 1328 | 1329 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 |
| 2470 | 1336 | 1337 | 1338 | 1339 | 1340 | 1341 | 1342 | 1343 |
| 2500 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1350 | 1351 |
| 2510 | 1352 | 1353 | 1354 | 1355 | 1356 | 1357 | 1358 | 1359 |
| 2520 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 |
| 2530 | 1368 | 1369 | 1370 | 1371 | 1372 | 1373 | 1374 | 1375 |
| 2540 | 1376 | 1377 | 1378 | 1379 | 1380 | 1381 | 1382 | 1383 |
| 2550 | 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1390 | 1391 |
| 2560 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 | 1398 | 1399 |
| 2570 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 |
| 2600 | 1408 | 1409 | 1410 | 1411 | 1412 | 1413 | 1414 | 1415 |
| 2610 | 1416 | 1417 | 1418 | 1419 | 1420 | 1421 | 1422 | 1423 |
| 2620 | 1424 | 1425 | 1426 | 1427 | 1428 | 1429 | 1430 | 1431 |
| 2630 | 1432 | 1433 | 1434 | 1435 | 1436 | 1437 | 1438 | 1439 |
| 2640 | 1440 | 1441 | 1442 | 1443 | 1444 | 1445 | 1446 | 1447 |
| 2650 | 1448 | 1449 | 1450 | 1451 | 1452 | 1453 | 1454 | 1455 |
| 2660 | 1456 | 1457 | 1458 | 1459 | 1460 | 1461 | 1462 | 1463 |
| 2670 | 1464 | 1465 | 1466 | 1467 | 1468 | 1469 | 1470 | 1471 |
| 2700 | 1472 | 1473 | 1474 | 1475 | 1476 | 1477 | 1478 | 1479 |
| 2710 | 1480 | 1481 | 1482 | 1483 | 1484 | 1485 | 1486 | 1487 |
| 2720 | 1488 | 1489 | 1490 | 1491 | 1492 | 1493 | 1494 | 1495 |
| 2730 | 1496 | 1497 | 1498 | 1499 | 1500 | 1501 | 1502 | 1503 |
| 2740 | 1504 | 1505 | 1506 | 1507 | 1508 | 1509 | 1510 | 1511 |
| 2750 | 1512 | 1513 | 1514 | 1515 | 1516 | 1517 | 1518 | 1519 |
| 2760 | 1520 | 1521 | 1522 | 1523 | 1524 | 1525 | 1526 | 1527 |
| 2770 | 1528 | 1529 | 1530 | 1531 | 1532 | 1533 | 1534 | 1535 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 3400 | 1792 | 1793 | 1794 | 1795 | 1796 | 1797 | 1798 | 1799 |
| 3410 | 1800 | 1801 | 1802 | 1803 | 1804 | 1805 | 1806 | 1807 |
| 3420 | 1808 | 1809 | 1810 | 1811 | 1812 | 1813 | 1814 | 1815 |
| 3430 | 1816 | 1817 | 1818 | 1819 | 1820 | 1821 | 1822 | 1823 |
| 3440 | 1824 | 1825 | 1826 | 1827 | 1828 | 1829 | 1830 | 1831 |
| 3450 | 1832 | 1833 | 1834 | 1835 | 1836 | 1837 | 1838 | 1839 |
| 3460 | 1840 | 1841 | 1842 | 1843 | 1844 | 1845 | 1846 | 1847 |
| 3470 | 1848 | 1849 | 1850 | 1851 | 1852 | 1853 | 1854 | 1855 |
| 3500 | 1856 | 1857 | 1858 | 1859 | 1860 | 1861 | 1862 | 1863 |
| 3510 | 1864 | 1865 | 1866 | 1867 | 1868 | 1869 | 1870 | 1871 |
| 3520 | 1872 | 1873 | 1874 | 1875 | 1876 | 1877 | 1878 | 1879 |
| 3530 | 1880 | 1881 | 1882 | 1883 | 1884 | 1885 | 1886 | 1887 |
| 3540 | 1888 | 1889 | 1890 | 1891 | 1892 | 1893 | 1894 | 1895 |
| 3550 | 1896 | 1897 | 1898 | 1899 | 1900 | 1901 | 1902 | 1903 |
| 3560 | 1904 | 1905 | 1906 | 1907 | 1908 | 1909 | 1910 | 1911 |
| 3570 | 1912 | 1913 | 1914 | 1915 | 1916 | 1917 | 1918 | 1919 |
| 3600 | 1920 | 1921 | 1922 | 1923 | 1924 | 1925 | 1926 | 1927 |
| 3610 | 1928 | 1929 | 1930 | 1931 | 1932 | 1933 | 1934 | 1935 |
| 3620 | 1936 | 1937 | 1938 | 1939 | 1940 | 1941 | 1942 | 1943 |
| 3630 | 1944 | 1945 | 1946 | 1947 | 1948 | 1949 | 1950 | 1951 |
| 3640 | 1952 | 1953 | 1954 | 1955 | 1956 | 1957 | 1958 | 1959 |
| 3650 | 1960 | 1961 | 1962 | 1963 | 1964 | 1965 | 1966 | 1967 |
| 3660 | 1968 | 1969 | 1970 | 1971 | 1972 | 1973 | 1974 | 1975 |
| 3670 | 1976 | 1977 | 1978 | 1979 | 1980 | 1981 | 1982 | 1983 |
| 3700 | 1984 | 1985 | 1986 | 1987 | 1988 | 1989 | 1990 | 1991 |
| 3710 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 |
| 3720 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 |
| 3730 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 |
| 3740 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 |
| 3750 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 |
| 3760 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 |
| 3770 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 |

2000 1024
to to
2777 1535
(Octal) (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

3000 1536
to to
3777 2047
(Octal) (Decimal)

Octal-Decimal Integer Conversion Table

4000 to 4777
(Octal) | 2048 to 2559
(Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 4000 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 |
| 4010 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 |
| 4020 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 |
| 4030 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 |
| 4040 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 |
| 4050 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 |
| 4060 | 2096 | 2097 | 2098 | 2099 | 2100 | 2101 | 2102 | 2103 |
| 4070 | 2104 | 2105 | 2106 | 2107 | 2108 | 2109 | 2110 | 2111 |
| 4100 | 2112 | 2113 | 2114 | 2115 | 2116 | 2117 | 2118 | 2119 |
| 4110 | 2120 | 2121 | 2122 | 2123 | 2124 | 2125 | 2126 | 2127 |
| 4120 | 2128 | 2129 | 2130 | 2131 | 2132 | 2133 | 2134 | 2135 |
| 4130 | 2136 | 2137 | 2138 | 2139 | 2140 | 2141 | 2142 | 2143 |
| 4140 | 2144 | 2145 | 2146 | 2147 | 2148 | 2149 | 2150 | 2151 |
| 4150 | 2152 | 2153 | 2154 | 2155 | 2156 | 2157 | 2158 | 2159 |
| 4160 | 2160 | 2161 | 2162 | 2163 | 2164 | 2165 | 2166 | 2167 |
| 4170 | 2168 | 2169 | 2170 | 2171 | 2172 | 2173 | 2174 | 2175 |
| 4200 | 2176 | 2177 | 2178 | 2179 | 2180 | 2181 | 2182 | 2183 |
| 4210 | 2184 | 2185 | 2186 | 2187 | 2188 | 2189 | 2190 | 2191 |
| 4220 | 2192 | 2193 | 2194 | 2195 | 2196 | 2197 | 2198 | 2199 |
| 4230 | 2200 | 2201 | 2202 | 2203 | 2204 | 2205 | 2206 | 2207 |
| 4240 | 2208 | 2209 | 2210 | 2211 | 2212 | 2213 | 2214 | 2215 |
| 4250 | 2216 | 2217 | 2218 | 2219 | 2220 | 2221 | 2222 | 2223 |
| 4260 | 2224 | 2225 | 2226 | 2227 | 2228 | 2229 | 2230 | 2231 |
| 4270 | 2232 | 2233 | 2234 | 2235 | 2236 | 2237 | 2238 | 2239 |
| 4300 | 2240 | 2241 | 2242 | 2243 | 2244 | 2245 | 2246 | 2247 |
| 4310 | 2248 | 2249 | 2250 | 2251 | 2252 | 2253 | 2254 | 2255 |
| 4320 | 2256 | 2257 | 2258 | 2259 | 2260 | 2261 | 2262 | 2263 |
| 4330 | 2264 | 2265 | 2266 | 2267 | 2268 | 2269 | 2270 | 2271 |
| 4340 | 2272 | 2273 | 2274 | 2275 | 2276 | 2277 | 2278 | 2279 |
| 4350 | 2280 | 2281 | 2282 | 2283 | 2284 | 2285 | 2286 | 2287 |
| 4360 | 2288 | 2289 | 2290 | 2291 | 2292 | 2293 | 2294 | 2295 |
| 4370 | 2296 | 2297 | 2298 | 2299 | 2300 | 2301 | 2302 | 2303 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 4400 | 2304 | 2305 | 2306 | 2307 | 2308 | 2309 | 2310 | 2311 |
| 4410 | 2312 | 2313 | 2314 | 2315 | 2316 | 2317 | 2318 | 2319 |
| 4420 | 2320 | 2321 | 2322 | 2323 | 2324 | 2325 | 2326 | 2327 |
| 4430 | 2328 | 2329 | 2330 | 2331 | 2332 | 2333 | 2334 | 2335 |
| 4440 | 2336 | 2337 | 2338 | 2339 | 2340 | 2341 | 2342 | 2343 |
| 4450 | 2344 | 2345 | 2346 | 2347 | 2348 | 2349 | 2350 | 2351 |
| 4460 | 2352 | 2353 | 2354 | 2355 | 2356 | 2357 | 2358 | 2359 |
| 4470 | 2360 | 2361 | 2362 | 2363 | 2364 | 2365 | 2366 | 2367 |
| 4500 | 2368 | 2369 | 2370 | 2371 | 2372 | 2373 | 2374 | 2375 |
| 4510 | 2376 | 2377 | 2378 | 2379 | 2380 | 2381 | 2382 | 2383 |
| 4520 | 2384 | 2385 | 2386 | 2387 | 2388 | 2389 | 2390 | 2391 |
| 4530 | 2392 | 2393 | 2394 | 2395 | 2396 | 2397 | 2398 | 2399 |
| 4540 | 2400 | 2401 | 2402 | 2403 | 2404 | 2405 | 2406 | 2407 |
| 4550 | 2408 | 2409 | 2410 | 2411 | 2412 | 2413 | 2414 | 2415 |
| 4560 | 2416 | 2417 | 2418 | 2419 | 2420 | 2421 | 2422 | 2423 |
| 4570 | 2424 | 2425 | 2426 | 2427 | 2428 | 2429 | 2430 | 2431 |
| 4600 | 2432 | 2433 | 2434 | 2435 | 2436 | 2437 | 2438 | 2439 |
| 4610 | 2440 | 2441 | 2442 | 2443 | 2444 | 2445 | 2446 | 2447 |
| 4620 | 2448 | 2449 | 2450 | 2451 | 2452 | 2453 | 2454 | 2455 |
| 4630 | 2456 | 2457 | 2458 | 2459 | 2460 | 2461 | 2462 | 2463 |
| 4640 | 2464 | 2465 | 2466 | 2467 | 2468 | 2469 | 2470 | 2471 |
| 4650 | 2472 | 2473 | 2474 | 2475 | 2476 | 2477 | 2478 | 2479 |
| 4660 | 2480 | 2481 | 2482 | 2483 | 2484 | 2485 | 2486 | 2487 |
| 4670 | 2488 | 2489 | 2490 | 2491 | 2492 | 2493 | 2494 | 2495 |
| 4700 | 2496 | 2497 | 2498 | 2499 | 2500 | 2501 | 2502 | 2503 |
| 4710 | 2504 | 2505 | 2506 | 2507 | 2508 | 2509 | 2510 | 2511 |
| 4720 | 2512 | 2513 | 2514 | 2515 | 2516 | 2517 | 2518 | 2519 |
| 4730 | 2520 | 2521 | 2522 | 2523 | 2524 | 2525 | 2526 | 2527 |
| 4740 | 2528 | 2529 | 2530 | 2531 | 2532 | 2533 | 2534 | 2535 |
| 4750 | 2536 | 2537 | 2538 | 2539 | 2540 | 2541 | 2542 | 2543 |
| 4760 | 2544 | 2545 | 2546 | 2547 | 2548 | 2549 | 2550 | 2551 |
| 4770 | 2552 | 2553 | 2554 | 2555 | 2556 | 2557 | 2558 | 2559 |

5000 to 5777
(Octal) | 2560 to 3071
(Decimal)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 5000 | 2560 | 2561 | 2562 | 2563 | 2564 | 2565 | 2566 | 2567 |
| 5010 | 2568 | 2569 | 2570 | 2571 | 2572 | 2573 | 2574 | 2575 |
| 5020 | 2576 | 2577 | 2578 | 2579 | 2580 | 2581 | 2582 | 2583 |
| 5030 | 2584 | 2585 | 2586 | 2587 | 2588 | 2589 | 2590 | 2591 |
| 5040 | 2592 | 2593 | 2594 | 2595 | 2596 | 2597 | 2598 | 2599 |
| 5050 | 2600 | 2601 | 2602 | 2603 | 2604 | 2605 | 2606 | 2607 |
| 5060 | 2608 | 2609 | 2610 | 2611 | 2612 | 2613 | 2614 | 2615 |
| 5070 | 2616 | 2617 | 2618 | 2619 | 2620 | 2621 | 2622 | 2623 |
| 5100 | 2624 | 2625 | 2626 | 2627 | 2628 | 2629 | 2630 | 2631 |
| 5110 | 2632 | 2633 | 2634 | 2635 | 2636 | 2637 | 2638 | 2639 |
| 5120 | 2640 | 2641 | 2642 | 2643 | 2644 | 2645 | 2646 | 2647 |
| 5130 | 2648 | 2649 | 2650 | 2651 | 2652 | 2653 | 2654 | 2655 |
| 5140 | 2656 | 2657 | 2658 | 2659 | 2660 | 2661 | 2662 | 2663 |
| 5150 | 2664 | 2665 | 2666 | 2667 | 2668 | 2669 | 2670 | 2671 |
| 5160 | 2672 | 2673 | 2674 | 2675 | 2676 | 2677 | 2678 | 2679 |
| 5170 | 2680 | 2681 | 2682 | 2683 | 2684 | 2685 | 2686 | 2687 |
| 5200 | 2688 | 2689 | 2690 | 2691 | 2692 | 2693 | 2694 | 2695 |
| 5210 | 2696 | 2697 | 2698 | 2699 | 2700 | 2701 | 2702 | 2703 |
| 5220 | 2704 | 2705 | 2706 | 2707 | 2708 | 2709 | 2710 | 2711 |
| 5230 | 2712 | 2713 | 2714 | 2715 | 2716 | 2717 | 2718 | 2719 |
| 5240 | 2720 | 2721 | 2722 | 2723 | 2724 | 2725 | 2726 | 2727 |
| 5250 | 2728 | 2729 | 2730 | 2731 | 2732 | 2733 | 2734 | 2735 |
| 5260 | 2736 | 2737 | 2738 | 2739 | 2740 | 2741 | 2742 | 2743 |
| 5270 | 2744 | 2745 | 2746 | 2747 | 2748 | 2749 | 2750 | 2751 |
| 5300 | 2752 | 2753 | 2754 | 2755 | 2756 | 2757 | 2758 | 2759 |
| 5310 | 2760 | 2761 | 2762 | 2763 | 2764 | 2765 | 2766 | 2767 |
| 5320 | 2768 | 2769 | 2770 | 2771 | 2772 | 2773 | 2774 | 2775 |
| 5330 | 2776 | 2777 | 2778 | 2779 | 2780 | 2781 | 2782 | 2783 |
| 5340 | 2784 | 2785 | 2786 | 2787 | 2788 | 2789 | 2790 | 2791 |
| 5350 | 2792 | 2793 | 2794 | 2795 | 2796 | 2797 | 2798 | 2799 |
| 5360 | 2800 | 2801 | 2802 | 2803 | 2804 | 2805 | 2806 | 2807 |
| 5370 | 2808 | 2809 | 2810 | 2811 | 2812 | 2813 | 2814 | 2815 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 5400 | 2816 | 2817 | 2818 | 2819 | 2820 | 2821 | 2822 | 2823 |
| 5410 | 2824 | 2825 | 2826 | 2827 | 2828 | 2829 | 2830 | 2831 |
| 5420 | 2832 | 2833 | 2834 | 2835 | 2836 | 2837 | 2838 | 2839 |
| 5430 | 2840 | 2841 | 2842 | 2843 | 2844 | 2845 | 2846 | 2847 |
| 5440 | 2848 | 2849 | 2850 | 2851 | 2852 | 2853 | 2854 | 2855 |
| 5450 | 2856 | 2857 | 2858 | 2859 | 2860 | 2861 | 2862 | 2863 |
| 5460 | 2864 | 2865 | 2866 | 2867 | 2868 | 2869 | 2870 | 2871 |
| 5470 | 2872 | 2873 | 2874 | 2875 | 2876 | 2877 | 2878 | 2879 |
| 5500 | 2880 | 2881 | 2882 | 2883 | 2884 | 2885 | 2886 | 2887 |
| 5510 | 2888 | 2889 | 2890 | 2891 | 2892 | 2893 | 2894 | 2895 |
| 5520 | 2896 | 2897 | 2898 | 2899 | 2900 | 2901 | 2902 | 2903 |
| 5530 | 2904 | 2905 | 2906 | 2907 | 2908 | 2909 | 2910 | 2911 |
| 5540 | 2912 | 2913 | 2914 | 2915 | 2916 | 2917 | 2918 | 2919 |
| 5550 | 2920 | 2921 | 2922 | 2923 | 2924 | 2925 | 2926 | 2927 |
| 5560 | 2928 | 2929 | 2930 | 2931 | 2932 | 2933 | 2934 | 2935 |
| 5570 | 2936 | 2937 | 2938 | 2939 | 2940 | 2941 | 2942 | 2943 |
| 5600 | 2944 | 2945 | 2946 | 2947 | 2948 | 2949 | 2950 | 2951 |
| 5610 | 2952 | 2953 | 2954 | 2955 | 2956 | 2957 | 2958 | 2959 |
| 5620 | 2960 | 2961 | 2962 | 2963 | 2964 | 2965 | 2966 | 2967 |
| 5630 | 2968 | 2969 | 2970 | 2971 | 2972 | 2973 | 2974 | 2975 |
| 5640 | 2976 | 2977 | 2978 | 2979 | 2980 | 2981 | 2982 | 2983 |
| 5650 | 2984 | 2985 | 2986 | 2987 | 2988 | 2989 | 2990 | 2991 |
| 5660 | 2992 | 2993 | 2994 | 2995 | 2996 | 2997 | 2998 | 2999 |
| 5670 | 3000 | 3001 | 3002 | 3003 | 3004 | 3005 | 3006 | 3007 |
| 5700 | 3008 | 3009 | 3010 | 3011 | 3012 | 3013 | 3014 | 3015 |
| 5710 | 3016 | 3017 | 3018 | 3019 | 3020 | 3021 | 3022 | 3023 |
| 5720 | 3024 | 3025 | 3026 | 3027 | 3028 | 3029 | 3030 | 3031 |
| 5730 | 3032 | 3033 | 3034 | 3035 | 3036 | 3037 | 3038 | 3039 |
| 5740 | 3040 | 3041 | 3042 | 3043 | 3044 | 3045 | 3046 | 3047 |
| 5750 | 3048 | 3049 | 3050 | 3051 | 3052 | 3053 | 3054 | 3055 |
| 5760 | 3056 | 3057 | 3058 | 3059 | 3060 | 3061 | 3062 | 3063 |
| 5770 | 3064 | 3065 | 3066 | 3067 | 3068 | 3069 | 3070 | 3071 |

Octal-Decimal Integer Conversion Table

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 6000 | 3072 | 3073 | 3074 | 3075 | 3076 | 3077 | 3078 | 3079 |
| 6010 | 3080 | 3081 | 3082 | 3083 | 3084 | 3085 | 3086 | 3087 |
| 6020 | 3088 | 3089 | 3090 | 3091 | 3092 | 3093 | 3094 | 3095 |
| 6030 | 3096 | 3097 | 3098 | 3099 | 3100 | 3101 | 3102 | 3103 |
| 6040 | 3104 | 3105 | 3106 | 3107 | 3108 | 3109 | 3110 | 3111 |
| 6050 | 3112 | 3113 | 3114 | 3115 | 3116 | 3117 | 3118 | 3119 |
| 6060 | 3120 | 3121 | 3122 | 3123 | 3124 | 3125 | 3126 | 3127 |
| 6070 | 3128 | 3129 | 3130 | 3131 | 3132 | 3133 | 3134 | 3135 |
| 6100 | 3136 | 3137 | 3138 | 3139 | 3140 | 3141 | 3142 | 3143 |
| 6110 | 3144 | 3145 | 3146 | 3147 | 3148 | 3149 | 3150 | 3151 |
| 6120 | 3152 | 3153 | 3154 | 3155 | 3156 | 3157 | 3158 | 3159 |
| 6130 | 3160 | 3161 | 3162 | 3163 | 3164 | 3165 | 3166 | 3167 |
| 6140 | 3168 | 3169 | 3170 | 3171 | 3172 | 3173 | 3174 | 3175 |
| 6150 | 3176 | 3177 | 3178 | 3179 | 3180 | 3181 | 3182 | 3183 |
| 6160 | 3184 | 3185 | 3186 | 3187 | 3188 | 3189 | 3190 | 3191 |
| 6170 | 3192 | 3193 | 3194 | 3195 | 3196 | 3197 | 3198 | 3199 |
| 6200 | 3200 | 3201 | 3202 | 3203 | 3204 | 3205 | 3206 | 3207 |
| 6210 | 3208 | 3209 | 3210 | 3211 | 3212 | 3213 | 3214 | 3215 |
| 6220 | 3216 | 3217 | 3218 | 3219 | 3220 | 3221 | 3222 | 3223 |
| 6230 | 3224 | 3225 | 3226 | 3227 | 3228 | 3229 | 3230 | 3231 |
| 6240 | 3232 | 3233 | 3234 | 3235 | 3236 | 3237 | 3238 | 3239 |
| 6250 | 3240 | 3241 | 3242 | 3243 | 3244 | 3245 | 3246 | 3247 |
| 6260 | 3248 | 3249 | 3250 | 3251 | 3252 | 3253 | 3254 | 3255 |
| 6270 | 3256 | 3257 | 3258 | 3259 | 3260 | 3261 | 3262 | 3263 |
| 6300 | 3264 | 3265 | 3266 | 3267 | 3268 | 3269 | 3270 | 3271 |
| 6310 | 3272 | 3273 | 3274 | 3275 | 3276 | 3277 | 3278 | 3279 |
| 6320 | 3280 | 3281 | 3282 | 3283 | 3284 | 3285 | 3286 | 3287 |
| 6330 | 3288 | 3289 | 3290 | 3291 | 3292 | 3293 | 3294 | 3295 |
| 6340 | 3296 | 3297 | 3298 | 3299 | 3300 | 3301 | 3302 | 3303 |
| 6350 | 3304 | 3305 | 3306 | 3307 | 3308 | 3309 | 3310 | 3311 |
| 6360 | 3312 | 3313 | 3314 | 3315 | 3316 | 3317 | 3318 | 3319 |
| 6370 | 3320 | 3321 | 3322 | 3323 | 3324 | 3325 | 3326 | 3327 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 6400 | 3328 | 3329 | 3330 | 3331 | 3332 | 3333 | 3334 | 3335 |
| 6410 | 3336 | 3337 | 3338 | 3339 | 3340 | 3341 | 3342 | 3343 |
| 6420 | 3344 | 3345 | 3346 | 3347 | 3348 | 3349 | 3350 | 3351 |
| 6430 | 3352 | 3353 | 3354 | 3355 | 3356 | 3357 | 3358 | 3359 |
| 6440 | 3360 | 3361 | 3362 | 3363 | 3364 | 3365 | 3366 | 3367 |
| 6450 | 3368 | 3369 | 3370 | 3371 | 3372 | 3373 | 3374 | 3375 |
| 6460 | 3376 | 3377 | 3378 | 3379 | 3380 | 3381 | 3382 | 3383 |
| 6470 | 3384 | 3385 | 3386 | 3387 | 3388 | 3389 | 3390 | 3391 |
| 6500 | 3392 | 3393 | 3394 | 3395 | 3396 | 3397 | 3398 | 3399 |
| 6510 | 3400 | 3401 | 3402 | 3403 | 3404 | 3405 | 3406 | 3407 |
| 6520 | 3408 | 3409 | 3410 | 3411 | 3412 | 3413 | 3414 | 3415 |
| 6530 | 3416 | 3417 | 3418 | 3419 | 3420 | 3421 | 3422 | 3423 |
| 6540 | 3424 | 3425 | 3426 | 3427 | 3428 | 3429 | 3430 | 3431 |
| 6550 | 3432 | 3433 | 3434 | 3435 | 3436 | 3437 | 3438 | 3439 |
| 6560 | 3440 | 3441 | 3442 | 3443 | 3444 | 3445 | 3446 | 3447 |
| 6570 | 3448 | 3449 | 3450 | 3451 | 3452 | 3453 | 3454 | 3455 |
| 6600 | 3456 | 3457 | 3458 | 3459 | 3460 | 3461 | 3462 | 3463 |
| 6610 | 3464 | 3465 | 3466 | 3467 | 3468 | 3469 | 3470 | 3471 |
| 6620 | 3472 | 3473 | 3474 | 3475 | 3476 | 3477 | 3478 | 3479 |
| 6630 | 3480 | 3481 | 3482 | 3483 | 3484 | 3485 | 3486 | 3487 |
| 6640 | 3488 | 3489 | 3490 | 3491 | 3492 | 3493 | 3494 | 3495 |
| 6650 | 3496 | 3497 | 3498 | 3499 | 3500 | 3501 | 3502 | 3503 |
| 6660 | 3504 | 3505 | 3506 | 3507 | 3508 | 3509 | 3510 | 3511 |
| 6670 | 3512 | 3513 | 3514 | 3515 | 3516 | 3517 | 3518 | 3519 |
| 6700 | 3520 | 3521 | 3522 | 3523 | 3524 | 3525 | 3526 | 3527 |
| 6710 | 3528 | 3529 | 3530 | 3531 | 3532 | 3533 | 3534 | 3535 |
| 6720 | 3536 | 3537 | 3538 | 3539 | 3540 | 3541 | 3542 | 3543 |
| 6730 | 3544 | 3545 | 3546 | 3547 | 3548 | 3549 | 3550 | 3551 |
| 6740 | 3552 | 3553 | 3554 | 3555 | 3556 | 3557 | 3558 | 3559 |
| 6750 | 3560 | 3561 | 3562 | 3563 | 3564 | 3565 | 3566 | 3567 |
| 6760 | 3568 | 3569 | 3570 | 3571 | 3572 | 3573 | 3574 | 3575 |
| 6770 | 3576 | 3577 | 3578 | 3579 | 3580 | 3581 | 3582 | 3583 |

6000 to 6777 (Octal) | 3072 to 3583 (Decimal)

Octal to Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 7000 | 3584 | 3585 | 3586 | 3587 | 3588 | 3589 | 3590 | 3591 |
| 7010 | 3592 | 3593 | 3594 | 3595 | 3596 | 3597 | 3598 | 3599 |
| 7020 | 3600 | 3601 | 3602 | 3603 | 3604 | 3605 | 3606 | 3607 |
| 7030 | 3608 | 3609 | 3610 | 3611 | 3612 | 3613 | 3614 | 3615 |
| 7040 | 3616 | 3617 | 3618 | 3619 | 3620 | 3621 | 3622 | 3623 |
| 7050 | 3624 | 3625 | 3626 | 3627 | 3628 | 3629 | 3630 | 3631 |
| 7060 | 3632 | 3633 | 3634 | 3635 | 3636 | 3637 | 3638 | 3639 |
| 7070 | 3640 | 3641 | 3642 | 3643 | 3644 | 3645 | 3646 | 3647 |
| 7100 | 3648 | 3649 | 3650 | 3651 | 3652 | 3653 | 3654 | 3655 |
| 7110 | 3656 | 3657 | 3658 | 3659 | 3660 | 3661 | 3662 | 3663 |
| 7120 | 3664 | 3665 | 3666 | 3667 | 3668 | 3669 | 3670 | 3671 |
| 7130 | 3672 | 3673 | 3674 | 3675 | 3676 | 3677 | 3678 | 3679 |
| 7140 | 3680 | 3681 | 3682 | 3683 | 3684 | 3685 | 3686 | 3687 |
| 7150 | 3688 | 3689 | 3690 | 3691 | 3692 | 3693 | 3694 | 3695 |
| 7160 | 3696 | 3697 | 3698 | 3699 | 3700 | 3701 | 3702 | 3703 |
| 7170 | 3704 | 3705 | 3706 | 3707 | 3708 | 3709 | 3710 | 3711 |
| 7200 | 3712 | 3713 | 3714 | 3715 | 3716 | 3717 | 3718 | 3719 |
| 7210 | 3720 | 3721 | 3722 | 3723 | 3724 | 3725 | 3726 | 3727 |
| 7220 | 3728 | 3729 | 3730 | 3731 | 3732 | 3733 | 3734 | 3735 |
| 7230 | 3736 | 3737 | 3738 | 3739 | 3740 | 3741 | 3742 | 3743 |
| 7240 | 3744 | 3745 | 3746 | 3747 | 3748 | 3749 | 3750 | 3751 |
| 7250 | 3752 | 3753 | 3754 | 3755 | 3756 | 3757 | 3758 | 3759 |
| 7260 | 3760 | 3761 | 3762 | 3763 | 3764 | 3765 | 3766 | 3767 |
| 7270 | 3768 | 3769 | 3770 | 3771 | 3772 | 3773 | 3774 | 3775 |
| 7300 | 3776 | 3777 | 3778 | 3779 | 3780 | 3781 | 3782 | 3783 |
| 7310 | 3784 | 3785 | 3786 | 3787 | 3788 | 3789 | 3790 | 3791 |
| 7320 | 3792 | 3793 | 3794 | 3795 | 3796 | 3797 | 3798 | 3799 |
| 7330 | 3800 | 3801 | 3802 | 3803 | 3804 | 3805 | 3806 | 3807 |
| 7340 | 3808 | 3809 | 3810 | 3811 | 3812 | 3813 | 3814 | 3815 |
| 7350 | 3816 | 3817 | 3818 | 3819 | 3820 | 3821 | 3822 | 3823 |
| 7360 | 3824 | 3825 | 3826 | 3827 | 3828 | 3829 | 3830 | 3831 |
| 7370 | 3832 | 3833 | 3834 | 3835 | 3836 | 3837 | 3838 | 3839 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------|------|------|------|------|------|------|------|------|
| 7400 | 3840 | 3841 | 3842 | 3843 | 3844 | 3845 | 3846 | 3847 |
| 7410 | 3848 | 3849 | 3850 | 3851 | 3852 | 3853 | 3854 | 3855 |
| 7420 | 3856 | 3857 | 3858 | 3859 | 3860 | 3861 | 3862 | 3863 |
| 7430 | 3864 | 3865 | 3866 | 3867 | 3868 | 3869 | 3870 | 3871 |
| 7440 | 3872 | 3873 | 3874 | 3875 | 3876 | 3877 | 3878 | 3879 |
| 7450 | 3880 | 3881 | 3882 | 3883 | 3884 | 3885 | 3886 | 3887 |
| 7460 | 3888 | 3889 | 3890 | 3891 | 3892 | 3893 | 3894 | 3895 |
| 7470 | 3896 | 3897 | 3898 | 3899 | 3900 | 3901 | 3902 | 3903 |
| 7500 | 3904 | 3905 | 3906 | 3907 | 3908 | 3909 | 3910 | 3911 |
| 7510 | 3912 | 3913 | 3914 | 3915 | 3916 | 3917 | 3918 | 3919 |
| 7520 | 3920 | 3921 | 3922 | 3923 | 3924 | 3925 | 3926 | 3927 |
| 7530 | 3928 | 3929 | 3930 | 3931 | 3932 | 3933 | 3934 | 3935 |
| 7540 | 3936 | 3937 | 3938 | 3939 | 3940 | 3941 | 3942 | 3943 |
| 7550 | 3944 | 3945 | 3946 | 3947 | 3948 | 3949 | 3950 | 3951 |
| 7560 | 3952 | 3953 | 3954 | 3955 | 3956 | 3957 | 3958 | 3959 |
| 7570 | 3960 | 3961 | 3962 | 3963 | 3964 | 3965 | 3966 | 3967 |
| 7600 | 3968 | 3969 | 3970 | 3971 | 3972 | 3973 | 3974 | 3975 |
| 7610 | 3976 | 3977 | 3978 | 3979 | 3980 | 3981 | 3982 | 3983 |
| 7620 | 3984 | 3985 | 3986 | 3987 | 3988 | 3989 | 3990 | 3991 |
| 7630 | 3992 | 3993 | 3994 | 3995 | 3996 | 3997 | 3998 | 3999 |
| 7640 | 4000 | 4001 | 4002 | 4003 | 4004 | 4005 | 4006 | 4007 |
| 7650 | 4008 | 4009 | 4010 | 4011 | 4012 | 4013 | 4014 | 4015 |
| 7660 | 4016 | 4017 | 4018 | 4019 | 4020 | 4021 | 4022 | 4023 |
| 7670 | 4024 | 4025 | 4026 | 4027 | 4028 | 4029 | 4030 | 4031 |
| 7700 | 4032 | 4033 | 4034 | 4035 | 4036 | 4037 | 4038 | 4039 |
| 7710 | 4040 | 4041 | 4042 | 4043 | 4044 | 4045 | 4046 | 4047 |
| 7720 | 4048 | 4049 | 4050 | 4051 | 4052 | 4053 | 4054 | 4055 |
| 7730 | 4056 | 4057 | 4058 | 4059 | 4060 | 4061 | 4062 | 4063 |
| 7740 | 4064 | 4065 | 4066 | 4067 | 4068 | 4069 | 4070 | 4071 |
| 7750 | 4072 | 4073 | 4074 | 4075 | 4076 | 4077 | 4078 | 4079 |
| 7760 | 4080 | 4081 | 4082 | 4083 | 4084 | 4085 | 4086 | 4087 |
| 7770 | 4088 | 4089 | 4090 | 4091 | 4092 | 4093 | 4094 | 4095 |

7000 to 7777 (Octal) | 3584 to 4095 (Decimal)

APPENDIX D

Octal-Decimal Fraction Conversion Table

| OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. |
|-------|---------|-------|---------|-------|---------|-------|---------|
| .000 | .000000 | .100 | .125000 | .200 | .250000 | .300 | .375000 |
| .001 | .001953 | .101 | .126953 | .201 | .251953 | .301 | .376953 |
| .002 | .003906 | .102 | .128906 | .202 | .253906 | .302 | .378906 |
| .003 | .005859 | .103 | .130859 | .203 | .255859 | .303 | .380859 |
| .004 | .007812 | .104 | .132812 | .204 | .257812 | .304 | .382812 |
| .005 | .009765 | .105 | .134765 | .205 | .259765 | .305 | .384765 |
| .006 | .011718 | .106 | .136718 | .206 | .261718 | .306 | .386718 |
| .007 | .013671 | .107 | .138671 | .207 | .263671 | .307 | .388671 |
| .010 | .015625 | .110 | .140625 | .210 | .265625 | .310 | .390625 |
| .011 | .017578 | .111 | .142578 | .211 | .267578 | .311 | .392578 |
| .012 | .019531 | .112 | .144531 | .212 | .269531 | .312 | .394531 |
| .013 | .021484 | .113 | .146484 | .213 | .271484 | .313 | .396484 |
| .014 | .023437 | .114 | .148437 | .214 | .273437 | .314 | .398437 |
| .015 | .025390 | .115 | .150390 | .215 | .275390 | .315 | .400390 |
| .016 | .027343 | .116 | .152343 | .216 | .277343 | .316 | .402343 |
| .017 | .029296 | .117 | .154296 | .217 | .279296 | .317 | .404296 |
| .020 | .031250 | .120 | .156250 | .220 | .281250 | .320 | .406250 |
| .021 | .033203 | .121 | .158203 | .221 | .283203 | .321 | .408203 |
| .022 | .035156 | .122 | .160156 | .222 | .285156 | .322 | .410156 |
| .023 | .037109 | .123 | .162109 | .223 | .287109 | .323 | .412109 |
| .024 | .039062 | .124 | .164062 | .224 | .289062 | .324 | .414062 |
| .025 | .041015 | .125 | .166015 | .225 | .291015 | .325 | .416015 |
| .026 | .042968 | .126 | .167968 | .226 | .292968 | .326 | .417968 |
| .027 | .044921 | .127 | .169921 | .227 | .294921 | .327 | .419921 |
| .030 | .046875 | .130 | .171875 | .230 | .296875 | .330 | .421875 |
| .031 | .048828 | .131 | .173828 | .231 | .298828 | .331 | .423828 |
| .032 | .050781 | .132 | .175781 | .232 | .300781 | .332 | .425781 |
| .033 | .052734 | .133 | .177734 | .233 | .302734 | .333 | .427734 |
| .034 | .054687 | .134 | .179687 | .234 | .304687 | .334 | .429687 |
| .035 | .056640 | .135 | .181640 | .235 | .306640 | .335 | .431640 |
| .036 | .058593 | .136 | .183593 | .236 | .308593 | .336 | .433593 |
| .037 | .060546 | .137 | .185546 | .237 | .310546 | .337 | .435546 |
| .040 | .062500 | .140 | .187500 | .240 | .312500 | .340 | .437500 |
| .041 | .064453 | .141 | .189453 | .241 | .314453 | .341 | .439453 |
| .042 | .066406 | .142 | .191406 | .242 | .316406 | .342 | .441406 |
| .043 | .068359 | .143 | .193359 | .243 | .318359 | .343 | .443359 |
| .044 | .070312 | .144 | .195312 | .244 | .320312 | .344 | .445312 |
| .045 | .072265 | .145 | .197265 | .245 | .322265 | .345 | .447265 |
| .046 | .074218 | .146 | .199218 | .246 | .324218 | .346 | .449218 |
| .047 | .076171 | .147 | .201171 | .247 | .326171 | .347 | .451171 |
| .050 | .078125 | .150 | .203125 | .250 | .328125 | .350 | .453125 |
| .051 | .080078 | .151 | .205078 | .251 | .330078 | .351 | .455078 |
| .052 | .082031 | .152 | .207031 | .252 | .332031 | .352 | .457031 |
| .053 | .083984 | .153 | .208984 | .253 | .333984 | .353 | .458984 |
| .054 | .085937 | .154 | .210937 | .254 | .335937 | .354 | .460937 |
| .055 | .087890 | .155 | .212890 | .255 | .337890 | .355 | .462890 |
| .056 | .089843 | .156 | .214843 | .256 | .339843 | .356 | .464843 |
| .057 | .091796 | .157 | .216796 | .257 | .341796 | .357 | .466796 |
| .060 | .093750 | .160 | .218750 | .260 | .343750 | .360 | .468750 |
| .061 | .095703 | .161 | .220703 | .261 | .345703 | .361 | .470703 |
| .062 | .097656 | .162 | .222656 | .262 | .347656 | .362 | .472656 |
| .063 | .099609 | .163 | .224609 | .263 | .349609 | .363 | .474609 |
| .064 | .101562 | .164 | .226562 | .264 | .351562 | .364 | .476562 |
| .065 | .103515 | .165 | .228515 | .265 | .353515 | .365 | .478515 |
| .066 | .105468 | .166 | .230468 | .266 | .355468 | .366 | .480468 |
| .067 | .107421 | .167 | .232421 | .267 | .357421 | .367 | .482421 |
| .070 | .109375 | .170 | .234375 | .270 | .359375 | .370 | .484375 |
| .071 | .111328 | .171 | .236328 | .271 | .361328 | .371 | .486328 |
| .072 | .113281 | .172 | .238281 | .272 | .363281 | .372 | .488281 |
| .073 | .115234 | .173 | .240234 | .273 | .365234 | .373 | .490234 |
| .074 | .117187 | .174 | .242187 | .274 | .367187 | .374 | .492187 |
| .075 | .119140 | .175 | .244140 | .275 | .369140 | .375 | .494140 |
| .076 | .121093 | .176 | .246093 | .276 | .371093 | .376 | .496093 |
| .077 | .123046 | .177 | .248046 | .277 | .373046 | .377 | .498046 |

Octal-Decimal Fraction Conversion Table

| OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. |
|---------|---------|---------|---------|---------|---------|---------|---------|
| .000000 | .000000 | .000100 | .000244 | .000200 | .000488 | .000300 | .000732 |
| .000001 | .000003 | .000101 | .000247 | .000201 | .000492 | .000301 | .000736 |
| .000002 | .000007 | .000102 | .000251 | .000202 | .000495 | .000302 | .000740 |
| .000003 | .000011 | .000103 | .000255 | .000203 | .000499 | .000303 | .000743 |
| .000004 | .000015 | .000104 | .000259 | .000204 | .000503 | .000304 | .000747 |
| .000005 | .000019 | .000105 | .000263 | .000205 | .000507 | .000305 | .000751 |
| .000006 | .000022 | .000106 | .000267 | .000206 | .000511 | .000306 | .000755 |
| .000007 | .000026 | .000107 | .000270 | .000207 | .000514 | .000307 | .000759 |
| .000010 | .000030 | .000110 | .000274 | .000210 | .000518 | .000310 | .000762 |
| .000011 | .000034 | .000111 | .000278 | .000211 | .000522 | .000311 | .000766 |
| .000012 | .000038 | .000112 | .000282 | .000212 | .000526 | .000312 | .000770 |
| .000013 | .000041 | .000113 | .000286 | .000213 | .000530 | .000313 | .000774 |
| .000014 | .000045 | .000114 | .000289 | .000214 | .000534 | .000314 | .000778 |
| .000015 | .000049 | .000115 | .000293 | .000215 | .000537 | .000315 | .000782 |
| .000016 | .000053 | .000116 | .000297 | .000216 | .000541 | .000316 | .000785 |
| .000017 | .000057 | .000117 | .000301 | .000217 | .000545 | .000317 | .000789 |
| .000020 | .000061 | .000120 | .000305 | .000220 | .000549 | .000320 | .000793 |
| .000021 | .000064 | .000121 | .000308 | .000221 | .000553 | .000321 | .000797 |
| .000022 | .000068 | .000122 | .000312 | .000222 | .000556 | .000322 | .000801 |
| .000023 | .000072 | .000123 | .000316 | .000223 | .000560 | .000323 | .000805 |
| .000024 | .000076 | .000124 | .000320 | .000224 | .000564 | .000324 | .000808 |
| .000025 | .000080 | .000125 | .000324 | .000225 | .000568 | .000325 | .000812 |
| .000026 | .000083 | .000126 | .000328 | .000226 | .000572 | .000326 | .000816 |
| .000027 | .000087 | .000127 | .000331 | .000227 | .000576 | .000327 | .000820 |
| .000030 | .000091 | .000130 | .000335 | .000230 | .000579 | .000330 | .000823 |
| .000031 | .000095 | .000131 | .000339 | .000231 | .000583 | .000331 | .000827 |
| .000032 | .000099 | .000132 | .000343 | .000232 | .000587 | .000332 | .000831 |
| .000033 | .000102 | .000133 | .000347 | .000233 | .000591 | .000333 | .000835 |
| .000034 | .000106 | .000134 | .000350 | .000234 | .000595 | .000334 | .000839 |
| .000035 | .000110 | .000135 | .000354 | .000235 | .000598 | .000335 | .000843 |
| .000036 | .000114 | .000136 | .000358 | .000236 | .000602 | .000336 | .000846 |
| .000037 | .000118 | .000137 | .000362 | .000237 | .000606 | .000337 | .000850 |
| .000040 | .000122 | .000140 | .000366 | .000240 | .000610 | .000340 | .000854 |
| .000041 | .000125 | .000141 | .000370 | .000241 | .000614 | .000341 | .000858 |
| .000042 | .000129 | .000142 | .000373 | .000242 | .000617 | .000342 | .000862 |
| .000043 | .000133 | .000143 | .000377 | .000243 | .000621 | .000343 | .000865 |
| .000044 | .000137 | .000144 | .000381 | .000244 | .000625 | .000344 | .000869 |
| .000045 | .000141 | .000145 | .000385 | .000245 | .000629 | .000345 | .000873 |
| .000046 | .000144 | .000146 | .000389 | .000246 | .000633 | .000346 | .000877 |
| .000047 | .000148 | .000147 | .000392 | .000247 | .000637 | .000347 | .000881 |
| .000050 | .000152 | .000150 | .000396 | .000250 | .000640 | .000350 | .000885 |
| .000051 | .000156 | .000151 | .000400 | .000251 | .000644 | .000351 | .000888 |
| .000052 | .000160 | .000152 | .000404 | .000252 | .000648 | .000352 | .000892 |
| .000053 | .000164 | .000153 | .000408 | .000253 | .000652 | .000353 | .000896 |
| .000054 | .000167 | .000154 | .000411 | .000254 | .000656 | .000354 | .000900 |
| .000055 | .000171 | .000155 | .000415 | .000255 | .000659 | .000355 | .000904 |
| .000056 | .000175 | .000156 | .000419 | .000256 | .000663 | .000356 | .000907 |
| .000057 | .000179 | .000157 | .000423 | .000257 | .000667 | .000357 | .000911 |
| .000060 | .000183 | .000160 | .000427 | .000260 | .000671 | .000360 | .000915 |
| .000061 | .000186 | .000161 | .000431 | .000261 | .000675 | .000361 | .000919 |
| .000062 | .000190 | .000162 | .000434 | .000262 | .000679 | .000362 | .000923 |
| .000063 | .000194 | .000163 | .000438 | .000263 | .000682 | .000363 | .000926 |
| .000064 | .000198 | .000164 | .000442 | .000264 | .000686 | .000364 | .000930 |
| .000065 | .000202 | .000165 | .000446 | .000265 | .000690 | .000365 | .000934 |
| .000066 | .000205 | .000166 | .000450 | .000266 | .000694 | .000366 | .000938 |
| .000067 | .000209 | .000167 | .000453 | .000267 | .000698 | .000367 | .000942 |
| .000070 | .000213 | .000170 | .000457 | .000270 | .000701 | .000370 | .000946 |
| .000071 | .000217 | .000171 | .000461 | .000271 | .000705 | .000371 | .000949 |
| .000072 | .000221 | .000172 | .000465 | .000272 | .000709 | .000372 | .000953 |
| .000073 | .000225 | .000173 | .000469 | .000273 | .000713 | .000373 | .000957 |
| .000074 | .000228 | .000174 | .000473 | .000274 | .000717 | .000374 | .000961 |
| .000075 | .000232 | .000175 | .000476 | .000275 | .000720 | .000375 | .000965 |
| .000076 | .000236 | .000176 | .000480 | .000276 | .000724 | .000376 | .000968 |
| .000077 | .000240 | .000177 | .000484 | .000277 | .000728 | .000377 | .000972 |

Octal-Decimal Fraction Conversion Table

| OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. | OCTAL | DEC. |
|---------|---------|---------|---------|---------|---------|---------|---------|
| .000400 | .000976 | .000500 | .001220 | .000600 | .001464 | .000700 | .001708 |
| .000401 | .000980 | .000501 | .001224 | .000601 | .001468 | .000701 | .001712 |
| .000402 | .000984 | .000502 | .001228 | .000602 | .001472 | .000702 | .001716 |
| .000403 | .000988 | .000503 | .001232 | .000603 | .001476 | .000703 | .001720 |
| .000404 | .000991 | .000504 | .001235 | .000604 | .001480 | .000704 | .001724 |
| .000405 | .000995 | .000505 | .001239 | .000605 | .001483 | .000705 | .001728 |
| .000406 | .000999 | .000506 | .001243 | .000606 | .001487 | .000706 | .001731 |
| .000407 | .001003 | .000507 | .001247 | .000607 | .001491 | .000707 | .001735 |
| .000410 | .001007 | .000510 | .001251 | .000610 | .001495 | .000710 | .001739 |
| .000411 | .001010 | .000511 | .001255 | .000611 | .001499 | .000711 | .001743 |
| .000412 | .001014 | .000512 | .001258 | .000612 | .001502 | .000712 | .001747 |
| .000413 | .001018 | .000513 | .001262 | .000613 | .001506 | .000713 | .001750 |
| .000414 | .001022 | .000514 | .001266 | .000614 | .001510 | .000714 | .001754 |
| .000415 | .001026 | .000515 | .001270 | .000615 | .001514 | .000715 | .001758 |
| .000416 | .001029 | .000516 | .001274 | .000616 | .001518 | .000716 | .001762 |
| .000417 | .001033 | .000517 | .001277 | .000617 | .001522 | .000717 | .001766 |
| .000420 | .001037 | .000520 | .001281 | .000620 | .001525 | .000720 | .001770 |
| .000421 | .001041 | .000521 | .001285 | .000621 | .001529 | .000721 | .001773 |
| .000422 | .001045 | .000522 | .001289 | .000622 | .001533 | .000722 | .001777 |
| .000423 | .001049 | .000523 | .001293 | .000623 | .001537 | .000723 | .001781 |
| .000424 | .001052 | .000524 | .001296 | .000624 | .001541 | .000724 | .001785 |
| .000425 | .001056 | .000525 | .001300 | .000625 | .001544 | .000725 | .001789 |
| .000426 | .001060 | .000526 | .001304 | .000626 | .001548 | .000726 | .001792 |
| .000427 | .001064 | .000527 | .001308 | .000627 | .001552 | .000727 | .001796 |
| .000430 | .001068 | .000530 | .001312 | .000630 | .001556 | .000730 | .001800 |
| .000431 | .001071 | .000531 | .001316 | .000631 | .001560 | .000731 | .001804 |
| .000432 | .001075 | .000532 | .001319 | .000632 | .001564 | .000732 | .001808 |
| .000433 | .001079 | .000533 | .001323 | .000633 | .001567 | .000733 | .001811 |
| .000434 | .001083 | .000534 | .001327 | .000634 | .001571 | .000734 | .001815 |
| .000435 | .001087 | .000535 | .001331 | .000635 | .001575 | .000735 | .001819 |
| .000436 | .001091 | .000536 | .001335 | .000636 | .001579 | .000736 | .001823 |
| .000437 | .001094 | .000537 | .001338 | .000637 | .001583 | .000737 | .001827 |
| .000440 | .001098 | .000540 | .001342 | .000640 | .001586 | .000740 | .001831 |
| .000441 | .001102 | .000541 | .001346 | .000641 | .001590 | .000741 | .001834 |
| .000442 | .001106 | .000542 | .001350 | .000642 | .001594 | .000742 | .001838 |
| .000443 | .001110 | .000543 | .001354 | .000643 | .001598 | .000743 | .001842 |
| .000444 | .001113 | .000544 | .001358 | .000644 | .001602 | .000744 | .001846 |
| .000445 | .001117 | .000545 | .001361 | .000645 | .001605 | .000745 | .001850 |
| .000446 | .001121 | .000546 | .001365 | .000646 | .001609 | .000746 | .001853 |
| .000447 | .001125 | .000547 | .001369 | .000647 | .001613 | .000747 | .001857 |
| .000450 | .001129 | .000550 | .001373 | .000650 | .001617 | .000750 | .001861 |
| .000451 | .001132 | .000551 | .001377 | .000651 | .001621 | .000751 | .001865 |
| .000452 | .001136 | .000552 | .001380 | .000652 | .001625 | .000752 | .001869 |
| .000453 | .001140 | .000553 | .001384 | .000653 | .001628 | .000753 | .001873 |
| .000454 | .001144 | .000554 | .001388 | .000654 | .001632 | .000754 | .001876 |
| .000455 | .001148 | .000555 | .001392 | .000655 | .001636 | .000755 | .001880 |
| .000456 | .001152 | .000556 | .001396 | .000656 | .001640 | .000756 | .001884 |
| .000457 | .001155 | .000557 | .001399 | .000657 | .001644 | .000757 | .001888 |
| .000460 | .001159 | .000560 | .001403 | .000660 | .001647 | .000760 | .001892 |
| .000461 | .001163 | .000561 | .001407 | .000661 | .001651 | .000761 | .001895 |
| .000462 | .001167 | .000562 | .001411 | .000662 | .001655 | .000762 | .001899 |
| .000463 | .001171 | .000563 | .001415 | .000663 | .001659 | .000763 | .001903 |
| .000464 | .001174 | .000564 | .001419 | .000664 | .001663 | .000764 | .001907 |
| .000465 | .001178 | .000565 | .001422 | .000665 | .001667 | .000765 | .001911 |
| .000466 | .001182 | .000566 | .001426 | .000666 | .001670 | .000766 | .001914 |
| .000467 | .001186 | .000567 | .001430 | .000667 | .001674 | .000767 | .001918 |
| .000470 | .001190 | .000570 | .001434 | .000670 | .001678 | .000770 | .001922 |
| .000471 | .001194 | .000571 | .001438 | .000671 | .001682 | .000771 | .001926 |
| .000472 | .001197 | .000572 | .001441 | .000672 | .001686 | .000772 | .001930 |
| .000473 | .001201 | .000573 | .001445 | .000673 | .001689 | .000773 | .001934 |
| .000474 | .001205 | .000574 | .001449 | .000674 | .001693 | .000774 | .001937 |
| .000475 | .001209 | .000575 | .001453 | .000675 | .001697 | .000775 | .001941 |
| .000476 | .001213 | .000576 | .001457 | .000676 | .001701 | .000776 | .001945 |
| .000477 | .001216 | .000577 | .001461 | .000677 | .001705 | .000777 | .001949 |

APPENDIX E

Command List

| Operation | Mnemonic Code | Numeric Code | Description |
|--------------------|---------------|-----------------------|-----------------------------|
| Arithmetic | ADD | 14 | Add |
| | SUB | 15 | Subtract |
| | DPA | 16 | Double Precision Add |
| | DPS | 17 | Double Precision Subtract |
| | SQR | 30 | Square Root |
| | DIV | 31 | Divide |
| | DVR | 31 | Divide Remainder |
| | MUP | 32 | Multiply |
| | CLA | 45 | Clear A |
| | CLB | 43 | Clear B |
| | CLC | 44 | Clear C |
| | GTB | 41 | Gray to Binary |
| | CAM | 56 | Compare A and M |
| | Transfer | TAN | 35 |
| TBN | | 36 | Transfer if B Negative |
| TCN | | 34 | Transfer if C Negative |
| TRU | | 37 | Transfer Unconditionally |
| TOF | | 75 | Transfer on Overflow |
| TES | | 77 | Transfer on External Signal |
| Loading & Storing | LDA | 05 | Load A |
| | LDB | 06 | Load B |
| | LDC | 04 | Load C |
| | LDP | 07 | Load Double Precision |
| | IAC | 01 | Interchange A & C |
| | IBC | 02 | Interchange B & C |
| | ROT | 03 | Rotate |
| | IAM | 25 | Interchange A & M |
| | STA | 11 | Store A |
| | STB | 12 | Store B |
| | STC | 10 | Store C |
| | STD | 13 | Store Double Precision |
| | MCL | 71 | Move Command Line Block |
| MLX | 26 | Move Line X to Line 7 | |
| Logical & Shifting | EBP | 40 | Extend Bit Pattern |
| | AMC | 42 | AND M & C |
| | MAC | 00 | Merge A into C |
| | AOC | 46 | AND OR Combined |
| | EXF | 47 | Extract Field |
| | NAD | 20 | Normalize and Decrement |
| | LSD | 21 | Left Shift and Decrement |
| | RSI | 22 | Right Shift and Increment |
| | SAI | 23 | Scale Right and Increment |
| | SBR | 33 | Shift B Right |
| Control | NOP | 24 | No Operation |
| | HLT | 00 | Halt |
| Input-Output | DIU | 50 | Disconnect Input Unit |
| | RTK | 51 | Read Typewriter Keyboard |
| | RPT | 52 | Read Paper Tape |
| | RFU | 53 | Read Fast Unit |
| | LAI | 55 | Load A From Input Buffer |
| | CIB | 57 | Clear Input Buffer |
| | WOC | 6X | Write Output Character |
| | PTU | 70 | Pulse to Specified Unit |
| | BSO | 72 | Block Serial Output |
| | BSI | 73 | Block Serial Input |