

**NUCLEAR DATA, INC.**  
**Post Office Box 451**  
**Palatine, Illinois 60067**

**PRINCIPLES OF PROGRAMMING**  
**THE ND812 COMPUTER**

**Copyright 1971 by Nuclear Data, Inc.**  
**Printed in U.S.A.**

## TABLE OF CONTENTS

| <u>SECTION</u> | <u>TITLE</u>   | <u>PAGE</u> |
|----------------|--|-------------|
| I              | INTRODUCTION . . . . .                                       | 1-1         |
|                | 1.1 Minicomputers and the Systems Concept. . . . .           | 1-1         |
|                | 1.2 Purpose of this Manual . . . . .                         | 1-2         |
| II             | COMPUTER NUMBER SYSTEMS . . . . .                            | 2-1         |
|                | 2.1 General . . . . .  | 2-1         |
|                | 2.2 Introduction . . . . .                                   | 2-1         |
|                | 2.3 The Binary System . . . . .                              | 2-2         |
|                | 2.4 The Octal System . . . . .                               | 2-5         |
|                | 2.5 Intra-System Conversions . . . . .                       | 2-7         |
| III            | COMPUTER ORGANIZATION . . . . .                              | 3-1         |
|                | 3.1 General . . . . .  | 3-1         |
|                | 3.2 ND812 Architecture . . . . .                             | 3-2         |
|                | 3.3 Computer Word Formats. . . . .                           | 3-4         |
|                | 3.4 Addressing . . . . .                                     | 3-9         |
| IV             | INSTRUCTION REPERTOIRE. . . . .                              | 4-1         |
|                | 4.1 General . . . . .  | 4-1         |
|                | 4.2 Memory Reference Instructions. . . . .                   | 4-1         |
|                | 4.3 Logical Operations . . . . .                             | 4-4         |
|                | 4.4 Arithmetic Operations on Accumulator Registers . . . . . | 4-5         |
|                | 4.5 Shift/Rotate Instructions . . . . .                      | 4-9         |
|                | 4.6 Load and Exchange Operations . . . . .                   | 4-10        |
|                | 4.7 Control Instructions . . . . .                           | 4-12        |
|                | 4.8 Literal Instructions . . . . .                           | 4-20        |
|                | 4.9 Input/Output . . . . .                                   | 4-21        |

| <u>SECTION</u> | <u>TITLE</u>   | <u>PAGE</u> |
|----------------|--|-------------|
| V              | PROGRAMMING FUNDAMENTALS . . . . .                               | 5-1         |
|                | 5.1 General . . . . .  | 5-1         |
|                | 5.2 Programming Procedures . . . . .                             | 5-1         |
|                | 5.3 Flow Charting. . . . .                                       | 5-2         |
|                | 5.4 Programming Concepts . . . . .                               | 5-3         |
|                | 5.5 Program Preparation . . . . .                                | 5-8         |
| VI             | COMPUTER LANGUAGES . . . . .                                     | 6-1         |
|                | 6.1 BASC-12 Assembly Language . . . . .                          | 6-1         |
|                | 6.2 NUTRAN Language. . . . .                                     | 6-2         |
| VII            | DESCRIPTION OF THE ND812 PROCESSOR AND PERIPHERALS . . . . .     | 7-1         |
|                | 7.1 General . . . . .  | 7-1         |
|                | 7.2 The ND812 Computer . . . . .                                 | 7-1         |
|                | 7.3 The ASR33 Teletypewriter. . . . .                            | 7-12        |
|                | 7.4 Peripheral Equipment . . . . .                               | 7-13        |
| VIII           | THE ND PROGRAM LISTING . . . . .                                 | 8-1         |
|                | 8.1 General . . . . .  | 8-1         |
|                | 8.2 Utility Programs. . . . .                                    | 8-1         |
|                | 8.3 System Software . . . . .                                    | 8-4         |
|                | 8.4 Diagnostic Programs . . . . .                                | 8-6         |
|                | 8.5 Physics Analyzer Programs . . . . .                          | 8-9         |
| APPENDICES     |  |             |
| A              | ND812 INSTRUCTION SET IN ALPHABETIC ORDER BY MNEMONIC . . . . .  | A-1         |
| B              | ND812 INSTRUCTION SET IN NUMERICAL ORDER BY OCTAL CODE . . . . . | B-1         |
| C              | FLOWCHARTING SYMBOLS . . . . .                                   | C-1         |
| D              | POWERS OF TWO. . . . .   | D-1         |
| E              | OCTAL-TO-DECIMAL CONVERSION TABLE . . . . .                      | E-1         |
| F              | FRACTIONAL CONVERSION TABLE. . . . .                             | F-1         |
| G              | ND CODE CONVERSION CHART . . . . .                               | G-1         |

## LIST OF ILLUSTRATIONS

| <u>FIGURE</u> | <u>TITLE</u>  | <u>PAGE</u> |
|---------------|---|-------------|
| 3-1           | Computer Elements . . . . .                           | 3-1         |
| 3-2           | Data Word Format . . . . .                            | 3-4         |
| 3-3           | Single-Word Format . . . . .                          | 3-5         |
| 3-4           | Two-Word Format . . . . .                             | 3-6         |
| 3-5           | Literal Format. . . . .                               | 3-7         |
| 3-6           | Group 1 Format . . . . .                              | 3-8         |
| 3-7           | Group 2 Format . . . . .                              | 3-8         |
| 3-8           | Status Word Bit Assignments (J-Register) . . . . .    | 3-9         |
| 4-1           | Typical Cassette Read Flowchart. . . . .              | 4-29        |
| 4-2           | Typical Cassette Write Data Flow Chart. . . . .       | 4-30        |
| 4-3           | Typical Cassette Write Filemark Flow Chart . . . . .  | 4-31        |
| 4-4           | Typical Cassette Filemark Search Flow Chart . . . . . | 4-32        |
| 5-1           | A Straight-Line Flow Chart . . . . .                  | 5-3         |
| 5-2           | A branched Flow Chart. . . . .                        | 5-4         |
| 5-3           | Typical Looping Situation. . . . .                    | 5-5         |
| 5-4           | Typical Address Modification Situation . . . . .      | 5-6         |

| <u>FIGURE</u> | <u>TITLE</u>                         | <u>PAGE</u> |
|---------------|--------------------------------------|-------------|
| 5-5           | Example Problem Flow Chart . . . . . | 5-21        |
| 7-1           | ND812 Front Panel . . . . .          | 7-2         |

## LIST OF TABLES

| <u>TABLE</u> | <u>TITLE</u>   | <u>PAGE</u> |
|--------------|--|-------------|
| 2-1          | Octal Multiplication Table . . . . .                       | 2-7         |
| 4-1          | Tape Cassette Control Flags . . . . .                      | 4-25        |
| 5-1          | Example Problem, Coded . . . . .                           | 5-11        |
| 5-2          | Teletype Printout of Example Problem. . . . .              | 5-17        |
| 5-3          | Listing of Example Program Produced by Assembler . . . . . | 5-23        |
| 7-1          | ND812 Central Processor Controls and Indicators. . . . .   | 7-1         |

## SECTION I INTRODUCTION

### 1.1 MINICOMPUTERS AND THE SYSTEMS CONCEPT

As recently as the early 1960's computerization was a prerogative of only very well capitalized people. The equipment was massive and complex; therefore, only highly-trained personnel could hope to extract the benefits such apparatus offered. Strict environmental control was often a necessity as well.

However, the advent of the minicomputer voided this situation. Barely larger than an attache case, the minicomputer can do almost anything that its big brothers do. It can perform the same type of logical and arithmetic operations and use the same data storage and input/output devices that the large computers do such as card readers and punches, magnetic tape and disk systems, cathode ray tube displays, plotters, and line printers. Advances in electronic circuitry and a reduction of environmental restrictions have allowed the "mini" to find applications in loci other than hermetically-sealed, antiseptically clean rooms.

The minicomputer also offers more computing power per dollar invested. For less than \$10,000 a central processor and 4,096 words (4K) of core memory can be purchased - hardware that approximates the computing power of the larger computers available today.

The applications of the minicomputer run the gamut of the computer business. It is used alone to solve scientific and engineering problems. It gives both small and large businesses the ability to automate payrolls, billings and inventory-control operations. It is used to control the operations of process industries and manufacturing plants. It replaces hard-wired logic in switching systems. It performs as a data concentrator for data-communications systems. It operates test lines in manufacturing and reads, records, and reduces data for engineers in development laboratories. It maintains the medical and financial records of patients in hospitals. The list is endless.

Because they are so much smaller and less expensive than big computers, most minicomputers find applications where large computers are never seen - built into research and general-purpose laboratory instruments, connected to industrial process and manufacturing equipment, in field research labs, and even in classrooms.

Minicomputers are so inexpensive that they are often used as special-purpose computers. Rather than trying to put together a laboratory system that interfaces one large computer with many instruments, an industrial or research laboratory may dedicate one minicomputer to each important instrument. In this case, a program is developed, the interface is designed, and the computer never does anything but the specific dedicated function. It becomes a permanent part of an instrument system. The implications of all this are exciting to contemplate, but one must first learn to program, and teaching that is the purpose of this **Manual**.

## 1.2 PURPOSE OF THIS MANUAL

This Manual is oriented toward the programming novice; its intent is to provide the ND812 user-errant with the technical foundation he will need to fully exploit the capabilities of his machine.

Section II is a discussion of computer number systems and their impact for the programming student.

Section III is a discussion of basic computer architecture, the configuration of computer "words", and the techniques which the programmer uses in communicating with his machine.

Section IV delineates in useful detail the ND812 instruction repertoire, which is nothing more than the range of operations the computer can perform upon receipt of the appropriate **command(s)**.

Section V is a discussion of the mechanics of the programming task.

Section VI contains descriptions of the programming languages commonly used with the ND812 computer.

Section VII consists of general descriptions of the ND812 computer itself and the sundry hardware devices available for use with it which offer the user so much flexibility in constructing task-dedicated systems.

Section VIII describes the programs presently available for the ND812.

The reader should also take note of the time-saving appendices to the volume.

Nuclear Data offers another companion volume ("NUTRAN") which augments the concepts offered in this book. It is obtainable from:

The Technical Documentation Department  
Nuclear Data, Inc.  
Golf and Meacham Roads  
Schaumburg, Illinois 60172



## SECTION II COMPUTER NUMBERING SYSTEMS

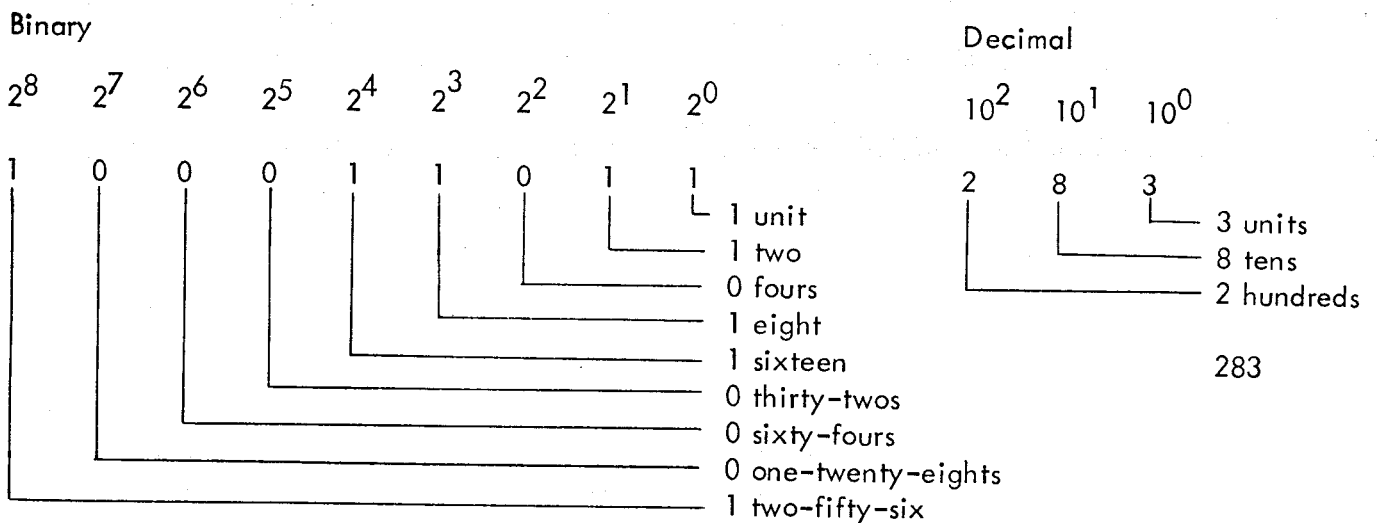
### 2.1 GENERAL

Numbering systems are generally identified by their respective radices (bases). The radix of any numbering system is the number of digit symbols which comprise that system. The decimal system is so named because it uses ten digit symbols (the numerals 0 through 9). This means that each system is based upon a radix, or root number, and that each position within a number represents a specific power of the radix of the system being used.

Programming principles derive from the extrapolation of mnemonics and number systems. It is therefore vital that the potential programmer master these concepts before he proceeds.

### 2.2 INTRODUCTION

In the decimal system, a number is represented by a sum of positional terms, each of which represents the product of a power of ten and some integer from 0 to 9. The number 283 may be expressed as the sum of each positional integer and the product of that positional power of ten:  $(2 \times 10^2) + (8 \times 10^1) + (3 \times 10^0)$ . Note,  $10^0$  equals one. In binary representation of numbers, the positions do not have the meaning of units, tens, hundreds, thousands, etc.; instead, these positions signify units, twos, fours, eights, sixteens, etc. The sum of these binary positions yields the same decimal sum.



## 2.3 THE BINARY SYSTEM

The ND812 computer uses a number system based on a radix of two (binary) and so uses two digits, 0 and 1. Binary is the common internal system for digital computation because of its relative simplicity. The electronic components that make up a digital computer are inherently binary. A relay is either opened or closed; magnetic materials (tape or cores) are magnetized in one direction or another; a transistor is either fully conducting or not conducting; an electrical pulse may be transmitted at a given time or not transmitted.

### 2.3.1 COUNTING IN BINARY NUMBERS

Binary counting starts in the same manner as in the decimal system with 0 for zero and 1 for one. However, because all possible symbols are then used, another position must be used to designate a decimal two. Therefore, at two in the binary system the same move is made that is made when ten is reached in the decimal system. That is, a one is placed in the position to the left and a zero is retained in the original position. In the binary system, any even number will contain a zero in the least significant position; an odd number will have a one in this position. Thus, the binary symbol 11 is equivalent to a 3 in the decimal system. Counting is continued with a carry into the position to the left each time the radix is exhausted.

| <u>Binary</u> | <u>Decimal</u> |
|---------------|----------------|
| 0             | 0              |
| 1             | 1              |
| 10            | 2              |
| 11            | 3              |
| 100           | 4              |
| 101           | 5              |
| 110           | 6              |
| 111           | 7              |
| 1000          | 8              |
| 1001          | 9              |
| 1010          | 10             |

Convention dictates that whenever two or more number systems are under discussion, the expressions are subscripted with their respective radices (bases). For instance, the decimal expression 530 would be written  $530_{10}$ , etc.

### 2.3.2 BINARY ADDITION

Three rules apply in binary addition:

a.  $0 + 0 = 0$

b.  $0 + 1 = 1 + 0 = 1$

c.  $1 + 1 = 0$ , with a carry of one to the position to the left, i.e.,  $= 10$

EXAMPLE

|         | 16s | 8s | 4s | 2s | 1s | Decimal |
|---------|-----|----|----|----|----|---------|
| Carries | 1   | 1  | 1  |    |    |         |
| Augend  | 0   | 1  | 1  | 1  | 0  | = 14    |
| Addend  | 0   | 1  | 0  | 1  | 1  | = +11   |
| Sum     | 1   | 1  | 0  | 0  | 1  | = 25    |

2.3.3 BINARY SUBTRACTION

Four rules apply in the binary subtraction operation:

1.  $0 - 0 = 0$

2.  $1 - 1 = 0$

3.  $1 - 0 = 1$

4.  $0 - 1 = 1$ , with one borrowed from the left

EXAMPLE

|            | 16s | 8s | 4s | 2s | 1s | Decimal |
|------------|-----|----|----|----|----|---------|
| Borrows    | -1  | -1 |    |    |    |         |
| Minuend    | 1   | 1  | 0  | 1  | 0  | = 26    |
| Subtrahend | 0   | 1  | 1  | 1  | 0  | = 14    |
| Difference | 0   | 1  | 1  | 0  | 0  | = 12    |

Rule 1 applies in the 1's column. Rule 2 applies to the 2's column. Rule 4 applies to the 4's column. Rules 2 and 4 apply to the 8's column. Rule 2 applies to the 16's column. The difference contains ones in the 8's and 4's columns. The decimal sum of this binary presentation is equal to 12, which is the correct difference of 26 and 14.

2.3.4 BINARY MULTIPLICATION

Three rules apply for binary multiplication:

1)  $0 \times 0 = 0$

$$2) 0 \times 1 = 1 \times 0 = 0$$

$$3) 1 \times 1 = 1$$

No carries are considered in multiplying. Each digit of the multiplier is examined; when a one is found, the multiplicand is added to the result. When a zero is found in the multiplier, zeros are added to the result. The multiplicand must be shifted left one digit for each multiplier digit.

#### EXAMPLE

$$\text{Multiplicands: } 01101 = 8 + 4 + 0 + 1 = 13$$

$$\text{Multipliers: } \begin{array}{r} \times 0101 \\ \hline \end{array} = 0 + 4 + 0 + 1 = \times 5$$

$$\begin{array}{r} 1101 \\ 0000 \\ 1101 \\ 0000 \\ \hline \end{array}$$

$$\text{Products: } 1000001 = 64 + 0 + 0 + 0 + 0 + 0 + 1 = 65$$

### 2.3.5 BINARY DIVISION

By applying the concepts of binary addition, subtraction, and multiplication, division may be accomplished. The dividend is inspected for the first group of digits from which the divisor may be subtracted once. A one is placed in the quotient over the last digit of the dividend group. This is continued with zeros appearing in the quotient where a subtraction is not possible after the next dividend digit is brought down to form the least significant digit of the new dividend.

#### EXAMPLE

$$1100/100$$

$$12/4$$

$$\begin{array}{r} \text{(Divisor) } 100 \overline{) 1100} \text{ (Dividend)} \\ \underline{100} \\ 100 \\ \underline{100} \\ 0 \end{array} \quad \begin{array}{l} 11 = 3 \text{ (Quotient)} \end{array}$$

The binary symbol 100 is greater than the binary symbol 1 or 11; therefore, binary 100 cannot be subtracted from binary 11. Binary 100 is subtracted from binary 110. The new dividend, binary 100, is formed by bringing down the next digit of the original dividend. The binary quotient is 11.

## 2.4 THE OCTAL SYSTEM

The octal system of assigning numerical values to binary forms is useful as a shorthand method of writing pure binary numbers. The octal system deals with groups of three binary positions; each group is considered a single digit. This means that, in any octal digit, there is a possibility of eight different binary positions; each group is considered a single digit (that is, 000, 001, 010, 011, 100, 101, 110, and 111). The octal equivalents of these representations are: 0, 1, 2, 3, 4, 5, 6, and 7 respectively. Given a series of binary digits, the first three on the far right are represented by the decimal notation  $1, 2, 3 \dots 7 \times 8^0$  and the next three digits toward the left are represented decimally by  $1, 2, 3 \dots 7 \times 8^1$ . It can be seen that each group of three binary bits represents some number (from 0-7) multiplied by a positional power of base eight. Also, the sum of these octal equivalent groups, i.e.,  $(1, 2, 3 \dots 7 \times 8^7) + \dots + (1, 2, 3 \dots 7 \times 8^1) + (1, 2, 3 \dots 7 \times 8^0)$  yields the decimal equivalent.

### EXAMPLE

|                | Binary Groups   | Octal Equivalents                                  | Decimal Equivalents                                      |
|----------------|---|--|--|
|                | $\underbrace{001}$ $\underbrace{\quad}$ $\underbrace{101}$<br>1      4      5 | $(1 \times 8^2) + (4 \times 8^1) + (5 \times 8^0)$ | $5$<br>$32$<br>$+64$<br><hr style="width: 100%;"/> $101$ |
| Octal Notation |   |  |  |

This binary number can be converted without using octal notation; however, the process requires the addition of seven quantities, rather than the three in octal notation.

### 2.4.1 OCTAL ADDITION

Addition for octal numbers should be no problem if the following basic rules for addition in any number system are kept in mind:

- a. If the sum of any column is equal to or greater than the base of the system being used, the base must be subtracted from the sum to obtain the final result of the column.
- b. If the sum of any column is equal to or greater than the base, there will be a carry to the next column which is equal to the number of times the base was subtracted.
- c. If the result of any column is less than the base, the base is not subtracted and no carry will be generated. Examples:

$$\begin{array}{r}
 5 \\
 + 4 \\
 \hline
 9 \\
 - 8 \\
 \hline
 11_8
 \end{array}
 = 5_{10} \\
 = 4_{10} \\
 = 9_{10}$$

$$\begin{array}{r}
 4 \ 5_8 \\
 5 \ 2_8 \\
 \hline
 1 \ 9 \ 7_8 \\
 - 8 \\
 \hline
 1 \ 1 \ 7_8
 \end{array}
 = 37_{10} \\
 = 42_{10} \\
 = 79_{10}$$

### 2.4.2 OCTAL SUBTRACTION

Subtraction is performed directly in the octal number system.

$$\begin{array}{r} 4567 \\ - 4321 \\ \hline 0246_8 \end{array}$$

$$\begin{array}{r} 4213 \\ - 3564 \\ \hline 0427_8 \end{array}$$

Whenever a borrow is needed in octal subtraction, an 8 is borrowed as in the second example above. In the first column, an 8 is borrowed and added to the 3 already in the first column and the 4 is subtracted from the resultant 11. In the second column, an 8 is borrowed and added to the 0 which is already in the column (after the previous borrow) and the 6 is subtracted from the resultant 8. In the third column 8 is borrowed and added to the 1 which is already in the column (after the previous borrow) and the 5 is subtracted from the resultant 9, and in the last column  $3 - 3 = 0$ .

### 2.4.3 OCTAL MULTIPLICATION

Multiplication of octal numbers is performed like multiplication of decimal numbers as long as the result is less than  $10_8$ . Obviously this could be a problem if it weren't for the fact that an octal multiplication table can be established which makes the job of multiplication of octal numbers quite simple. On the next page is an octal multiplication table that is quite useful.

Using the octal multiplication table 2-1, the following problems may be solved.

$$\begin{array}{r} 262_8 \\ \times 21_8 \\ \hline 262 \\ 544 \\ \hline 5722_8 \end{array}$$

#### NOTE

The left most digit (from table 2-1) is carried and added to the next number to the left as follows.

$$\begin{array}{r} 4567_8 \\ \times 1234_8 \\ \hline 22734 \\ 16145 \\ 11356 \\ 4567 \\ \hline 06131204_8 \end{array}$$

Carry  
Result

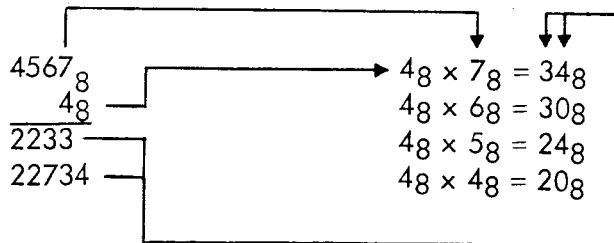


Table 2-1. OCTAL MULTIPLICATION TABLE

|   |   |   |    |    |    |    |    |    |
|---|---|---|----|----|----|----|----|----|
|   | 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  |
| 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 1 | 0 | 1 | 2  | 3  | 4  | 5  | 6  | 7  |
| 2 | 0 | 2 | 4  | 6  | 10 | 12 | 14 | 16 |
| 3 | 0 | 3 | 6  | 11 | 14 | 17 | 22 | 25 |
| 4 | 0 | 4 | 10 | 14 | 20 | 24 | 30 | 34 |
| 5 | 0 | 5 | 12 | 17 | 24 | 31 | 36 | 43 |
| 6 | 0 | 6 | 14 | 22 | 30 | 36 | 44 | 52 |
| 7 | 0 | 7 | 16 | 25 | 34 | 43 | 52 | 61 |

#### 2.4.4 OCTAL DIVISION

Octal division uses the same principles as decimal division. All multiplication and subtraction must, however, be done in octal (per the octal multiplication table 2-1). The following problems illustrate octal division.

$$\begin{array}{r} \text{Octal} \\ 66_8 \\ \hline 3_8 \end{array}$$

$$\begin{array}{r} \text{Decimal} \\ 54_{10} \\ \hline 3_{10} = 18_{10} \end{array}$$

$$\begin{array}{r} \text{Octal} \\ 2355_8 \\ \hline 15_8 \end{array}$$

$$\begin{array}{r} \text{Decimal} \\ 1261_{10} \\ \hline 13_{10} = 97_{10} \end{array}$$

$$\begin{array}{r} 22 \\ 3 \overline{)66} \\ \underline{6} \\ 06 \\ \underline{6} \\ 0 \end{array}$$

$$22_8 = 18_{10}$$

$$\begin{array}{r} 141 \\ 15 \overline{)2355} \\ \underline{15} \\ 65 \\ \underline{64} \\ 15 \\ \underline{15} \\ 0 \end{array}$$

$$141_8 = 97_{10}$$

## 2.5 INTRA-SYSTEM CONVERSIONS

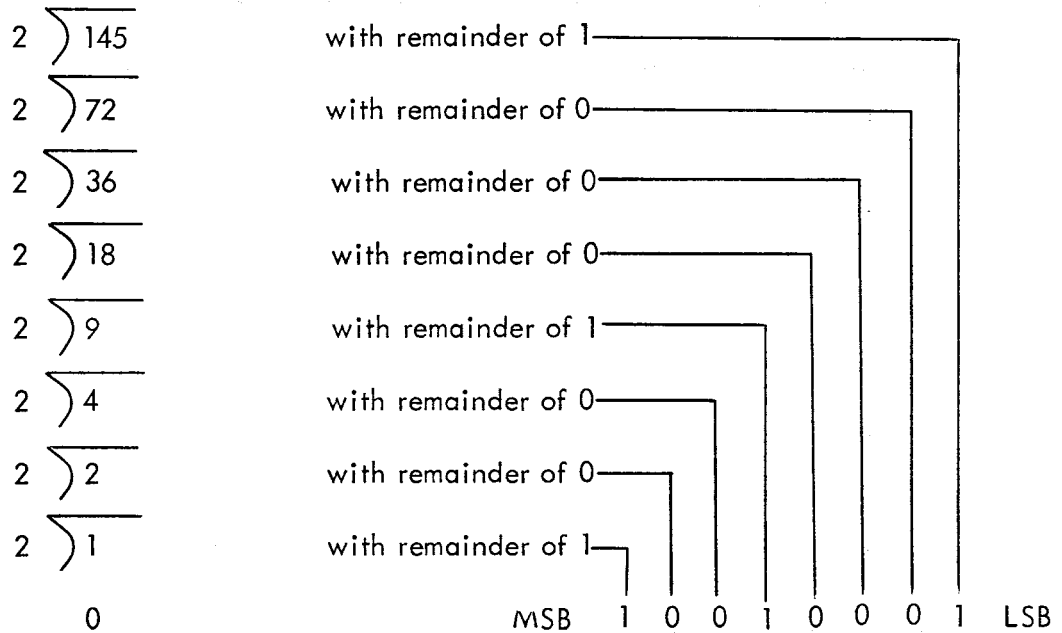
### 2.5.1 DECIMAL TO BINARY CONVERSION

A decimal number can be converted to its binary equivalent by dividing the number by two. If there is a remainder after the first division is performed, a binary bit of one will appear

in the least significant binary position. The appearance or lack of a remainder after each division determines the binary state of each position as illustrated below. Binary and octal conversion tables are provided as appendices to this manual for quick reference.

EXAMPLE

Convert Decimal 145 to Binary



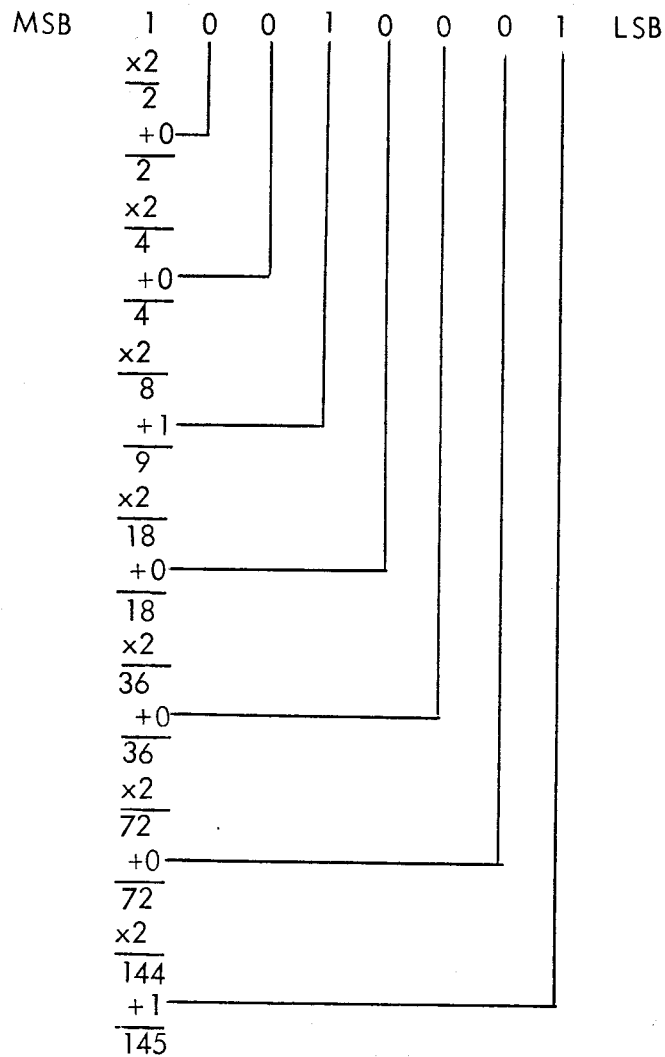
2.5.2 BINARY TO DECIMAL CONVERSION

Binary is converted to decimal by (starting with the most significant binary digit) multiplying each digit by two (the radix of the system) and adding the binary value of the next digit to the right as shown on the next page.

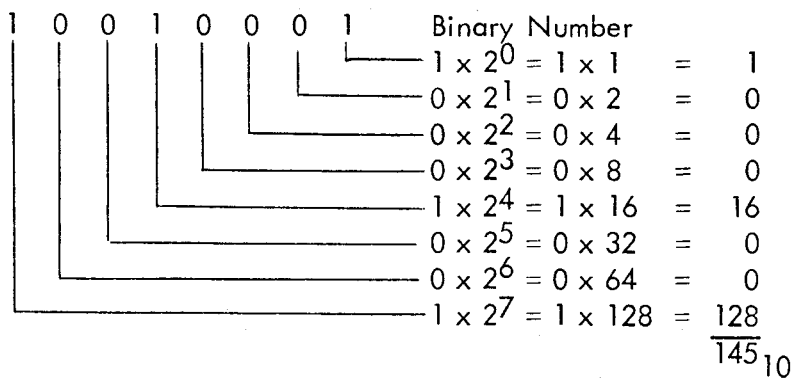


### EXAMPLE

Convert 10010001 to Decimal



Binary can also be converted to its decimal equivalent by (starting with the right most binary digit) multiplying each binary digit by its positional power of base two and adding the decimal values together as illustrated below.



Note that where a binary 1 appears, the positional power of base two is used directly and where a binary 0 appears, the resultant is 0.

### 2.5.3 DECIMAL TO OCTAL CONVERSION

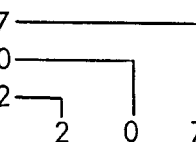
A decimal number can be converted to an octal equivalent by dividing the decimal number by eight and developing the octal number from the remainder as illustrated below.

#### EXAMPLE

Convert Decimal 135 to Octal

$$\begin{array}{r} 8 \overline{)135} \\ 8 \overline{)16} \\ 8 \overline{)2} \end{array}$$

with remainder of 7  
with remainder of 0  
with remainder of 2



### 2.5.4 OCTAL TO DECIMAL CONVERSION

Octal representation can be converted to its decimal equivalent by (starting with the right most octal digit) multiplying each octal digit by its positional power of base eight (the radix of the system) and adding the decimal values together as illustrated below.

#### EXAMPLE

Convert  $1427_8$  to Decimal

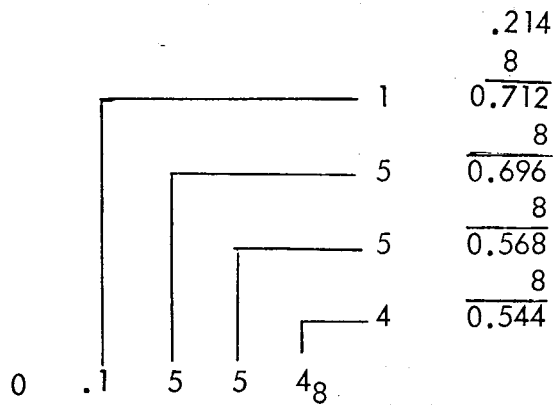
|   |   |   |   |                              |   |                          |
|---|---|---|---|------------------------------|---|--------------------------|
| 1 | 4 | 2 | 7 |                              |   |                          |
|   |   |   |   | 7 × 8 <sup>0</sup> = 7 × 1   | = | 7                        |
|   |   |   |   | 2 × 8 <sup>1</sup> = 2 × 8   | = | 16                       |
|   |   |   |   | 4 × 8 <sup>2</sup> = 4 × 64  | = | 256                      |
|   |   |   |   | 1 × 8 <sup>3</sup> = 1 × 512 | = | 512                      |
|   |   |   |   |                              |   | <u>791</u> <sub>10</sub> |

### 2.5.5 FRACTIONAL CONVERSION

Fractional conversions are performed in essentially the same manner as the respective integer conversions. A fractional decimal number can be converted to octal by multiplying the decimal number by eight. The fractional octal number is developed from the numbers to the left of the decimal point and must be preceded by a decimal point itself. It should be noted that conversion from decimal to octal or binary results in an approximation that may be carried to any number of places.

EXAMPLE

Convert Decimal 0.214 to Octal



EXAMPLE

Convert Octal 0.432 to Base 10 Equivalent

$$0.432 = (4 \times 8^{-1}) + (3 \times 8^{-2}) + (2 \times 8^{-3})$$

$$= (4/8 + 3/64 + 2/512)$$

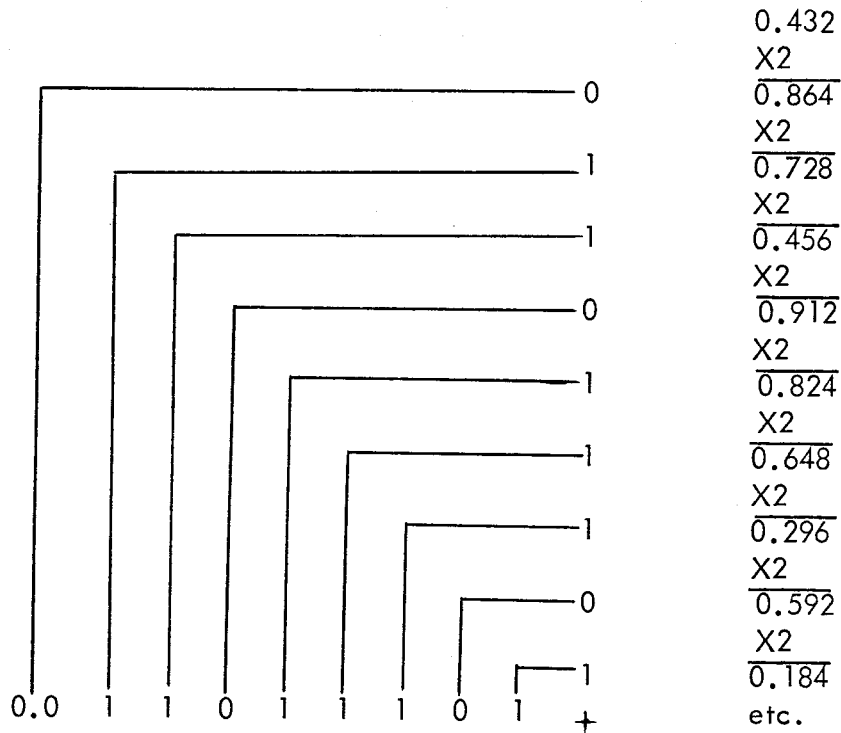
$$= 282/512$$

$$= 0.5507 \text{ (or rounded)}$$

$$= 0.551$$

EXAMPLE

Convert Decimal 0.432 to Binary



2.5.6 IMPROPER FRACTION CONVERSION

Improper fractions are converted from one system to another by converting the digit to the left of the decimal and the fraction separately. The result is then combined to form the conversion presentation.

2.5.7 OCTAL TO BINARY AND BINARY TO OCTAL CONVERSION

An octal number can be converted to binary form by considering each digit as a binary group of three. Also, a binary number can be converted to octal by considering each binary group of three as a digit.

EXAMPLE

|        |                    |                    |                    |                          |
|--------|--------------------|--------------------|--------------------|--------------------------|
| Octal  | 2                  | 7                  | 0                  |                          |
|        | $\underbrace{010}$ | $\underbrace{111}$ | $\underbrace{000}$ | = 010111000 <sub>2</sub> |
| Binary | $\underbrace{010}$ | $\underbrace{111}$ | $\underbrace{000}$ |                          |
|        | 2                  | 7                  | 0                  | = 270 <sub>8</sub>       |

## SECTION III COMPUTER ORGANIZATION

### 3.1 GENERAL

A machine, if it is to be called a computer, must be able to perform a certain type of logical operations. The element of the computer that performs this task is called the arithmetic/logical unit. If the arithmetic/logical unit is to perform its required task, it must be told what to do. The computer element performing this task is called the control unit.

Because mathematical operations are performed by the arithmetic unit, it may be necessary to store a partial answer while the unit is computing another part of the problem. This stored partial answer can then be used to solve other parts of the problem. The element meeting this requirement is called the memory or storage unit. The prime purpose of a digital computer is to provide a service; if it is to do this, there must be a means of both communicating needs to the computer and of obtaining the results. The element serving these functions is the input/output unit. Figure 3-1 shows the relationship of these units.

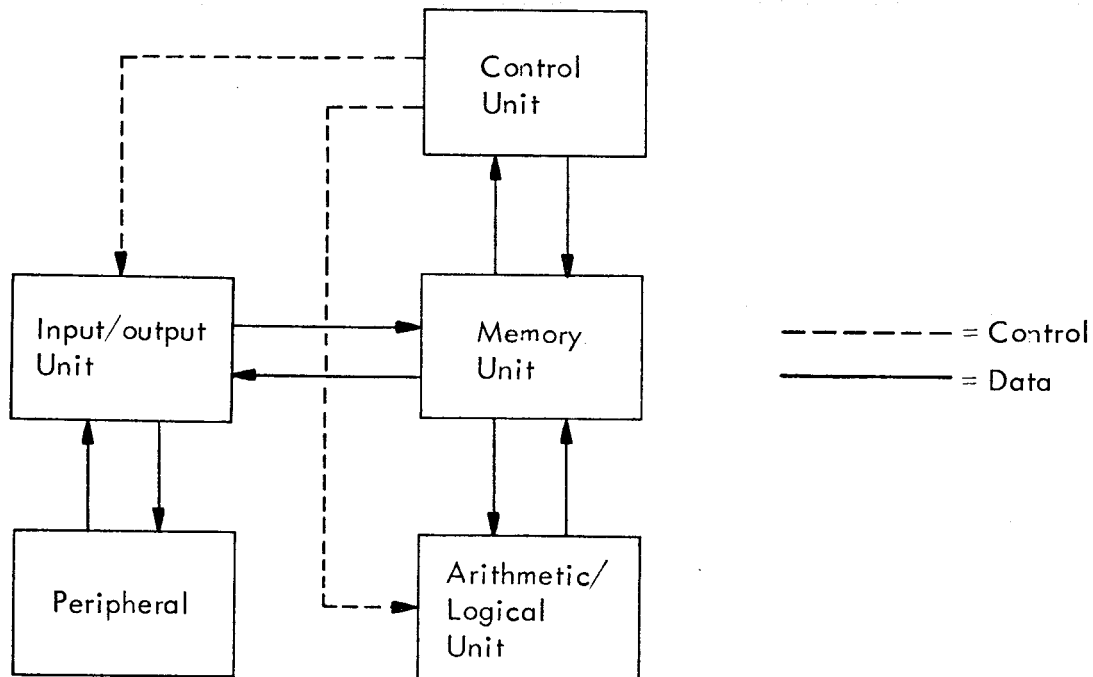


Figure 3-1. Computer Elements

## 3.2 ND812 ARCHITECTURE

The ND812 is a high-speed, general purpose, digital computer which operates on 12-bit binary numbers. It is a single-address, synchronous, sequential, parallel machine using two's complement arithmetic. It is composed of the four basic computer elements: control, arithmetic/logical, memory, input/output units.

### 3.2.1 CONTROL UNIT

The control unit is the coordinator or director of all operations within the computer. Its actions include directing the reading of information from memory, controlling the inputs and outputs of the computer, directing the operations within the arithmetic unit, and transferring information back into memory. It is consequently necessary for the control unit to determine each operation to be performed, the location of the data involved in the operation, and where to place the results. The control unit knows what it is to do by interpreting a set of instructions. This set of instructions is called a program and is stored in the memory unit.

The two basic functions of the control unit are (1), to obtain instruction words from memory, and (2), to execute these instructions. The control function performs these actions in two cycles: it fetches and executes. The fetch cycle is performed under the direct influence of the stored program so that the instructions are read in a fashion determined by program logic.

Each instruction read from memory is fed to the instruction register (IR), which holds the instruction word throughout the execution cycle. The instruction word contains two sections: the first indicates the function or operation code and the second is the operand (although it most frequently contains the address of data involved). The operand portion of the instruction word may be the number to be used in a calculation or it may be the address in memory of the number to be used. The second part of the instruction word (namely, the address section) generally represents the memory address of the data to be operated on. It should be understood that the instruction word does not necessarily contain the address of the operand; that is, it may be the address of an address of the operand.

Another portion of the control unit is the program counter (PC) which is used to record the memory location of the instruction to be executed. The PC always contains the address of the next instruction to be executed. Normally, instructions are executed in sequence; therefore, the PC is incremented by one to obtain the address of the next instruction. When an instruction causes transfer to another portion of the program, the PC is set to the appropriate address.

### 3.2.2 ARITHMETIC/LOGICAL UNIT

The arithmetic/logical unit performs all the actual work of the computation and calculation of a program in operation. Data which the arithmetic unit uses in performing computation are obtained from memory via the control unit. Arithmetic operations to be performed are also

determined in the control unit. The results of the arithmetic operations may then be stored back into memory. The basic arithmetic operations performed by the ND812 are addition, subtraction, multiplication, and division.

The ND812 arithmetic/logical unit has four 12-bit accumulator registers; two are capable of restricted operations. They are called accumulators because they accumulate partial sums during operation. All arithmetic operations are performed in the accumulators. The four accumulator registers are the J, K, R, and S registers. Registers J and K are commonly referenced as the main accumulators, because they are capable of direct storage and loading from the memory and are used in transmitting (under program control) 12 or 24-bit input/outputs. These contents may be added to, subtracted from, or exchanged. As a rule, all arithmetic results will appear in these two registers. There is only one exception: multiplication, in which the result appears in the R and S accumulators.

Registers R and S are commonly referenced as the sub-accumulators. They cannot be directly loaded from or stored into memory. They can, however, be exchanged, loaded, added to, and subtracted from the contents of the J and K accumulators. No result will appear in either of these two registers except for the aforementioned multiplication results.

### 3.2.3 MEMORY UNIT

The memory unit of the computer (also called magnetic core storage) contains information for the control and arithmetic units. The stored information for the control unit is in the form of instructions which are used to direct the processing of data in a predetermined and organized fashion. The information for the arithmetic unit is called data.

The ND812 memory unit is composed of ferrite cores which record binary information via the polarities of their magnetization. The memory unit is configured so that it can store 8K (8192) 12-bit words of binary information.

Each core storage location has a unique address. This method of storage is referred to as random-access storage. This means that any specific location in memory can be addressed as readily as any other and in the same amount of time. The ND812 basic memory unit is equipped with an 8,192 word (12-bit/word), 2 microsecond magnetic core memory. The memory unit is also available in 4K, 12K and 16K word memory configurations. There are two major registers in a memory unit configuration: the memory address register (MAR) and the memory data register (MR).

The MAR, a 12-bit register, contains only the address of the memory location currently being accessed. The MAR specifies (during both the fetch and execute cycles) which location is currently being used -- first for the instruction itself and then the execute phase (if there is one) for the operand. The register is not directly accessible to the programmer. It can be displayed and loaded, when desired, from the front panel.

The MR, a 12-bit register, is the data transfer path between the other registers of the ND812 and memory. It holds data read from the memory and any to be entered into memory

and is used in restoring data to a register. The MR is not directly accessible to the programmer, but it can be loaded and displayed, when desired, on the maintenance panel.

### 3.2.4 INPUT UNIT

Input devices are used to supply data needed by the computer and the instruction which tells the computer what to do with the data. Typical input devices are: teletype, magnetic tape, paper tape, punched cards, and disc units.

### 3.2.5 OUTPUT UNIT

Output devices record the results of the computer operations. Results may be recorded in a permanent form (such as printout on a teleprinter) or may be images on CRT devices. Many of the media used for input (paper tape, punched cards, magnetic tape, disc, etc.) can also be used for output.

## 3.3 COMPUTER WORD FORMATS

### 3.3.1 STORAGE DATA WORD FORMAT

The ND812 is oriented toward 12-bit binary words. The octal numbering system is employed to represent the binary word because it is more compact than binary. It can represent the state of each group of three bits in a word with a number representable by the arabic numerals 0 to 7. Consequently, the value representable in any single word will range from 0000 octal to 7777 octal, or from zero decimal to 4095 decimal.

A 12-bit word may represent decimal numerical values from zero to 4095 (4096 values). Therefore, a 12-bit word has the capacity to address the same number of words, which is precisely the number of words in a standard ND812 memory stack. Thus, a value contained in a single 12-bit word can address any location within a stack. The basic data word format is shown in Figure 3-2.

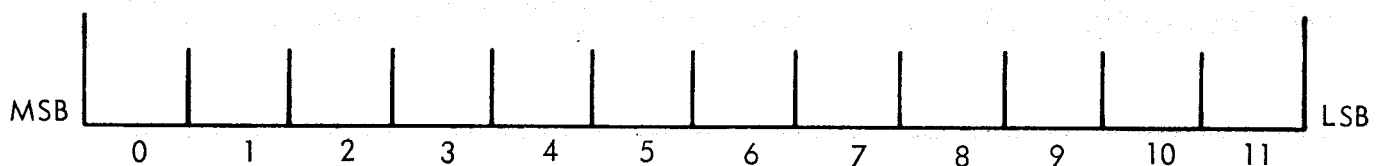


Figure 3-2. Data Word Format



Numbering of bit positions in each word is conventional; that is, the left-most bit is numbered zero, and the right-most bit, 11. Therefore, the most significant bit is bit 0 and the least significant bit is 11.

Two's complement arithmetic is employed in the addition and subtraction operations of the ND812. Bit 0 may be used to test the polarity of the number. If bit 0 equals 0, the number is positive. If bit 0 equals 1, the number is negative.

### 3.3.2 INSTRUCTION WORD FORMAT

There are three major instruction formats: single-word, two-word, and operate instructions. Although operate instructions are single word commands, their format is quite different from all others.

An important sub-class is the literal command. These are the only commands whose address portion is actually the operand employed in the instruction. Three are single word instructions. One is a modified single-word instruction.

Input/Output commands are also included in single and two-word formats.

**3.3.2.1 SINGLE WORD FORMAT.** Single-word memory reference instructions occupy only one 12-bit word. Because there are 4K words in a memory field, the bits in a single word are not sufficient to specify an operation code and a full address. In fact, only six bits are designated to specify the address of the operand in single-word memory reference instructions; those remaining are the command and variance bits.

The six address bits can specify a displacement which is added to the program counter to obtain the effective address. Because the value contained is equivalent to the range 0 to 63 (decimal), that is the range of addresses which can be accessed. However, one of the operation code bits (bit 5) can specify whether this range is forward or backward, so that a single-word memory reference command can access plus or minus 63 locations from its location in the program. The single-word format is shown in Figure 3-3.

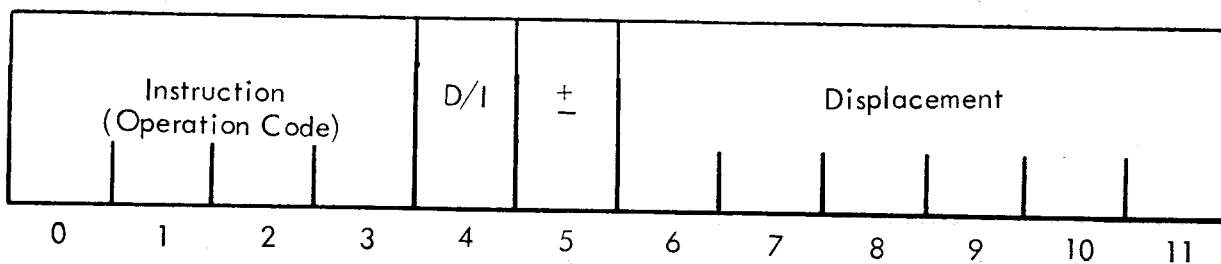


Figure 3-3. Single-Word Format

Bit 4, when set to 1, permits indirect addressing. When the displacement is used as an indirect address, the contents of the location which is plus or minus 63 locations from the instruction location is used as a pointer to the actual operand. Normally, only a single level of indirect address is possible.

There are many single-word instructions which do not reference memory. These are the instructions of the operate classes (Class I and Class II) and the input/output instructions.

Operate class instructions are instructions which do not reference memory. They do not, as a result, need bits 4 and 5 to specify forward/backward and direct/indirect addressing. Instead, these bits become part of the operation specification in the eight bits following the instruction code. These meanings vary, depending upon which group is being specified.

Single-word instructions for input/output are characterized by the presence of the octal operation code of 7400. The remaining eight bits specify device selection and which peripheral control pulses are desired.

**3.3.2.2 TWO-WORD FORMAT.** Two-word memory reference instructions have the operation code in the first word and the absolute 12-bit address in the second. The two must be contiguous and in the same field. The format of the first word of a two-word format is shown in Figure 3-4.

The specification of the operation to be performed is in the first word, but, to indicate that this is the two word format, bits 0, 1, and 2 of the first word are always set to zero. The remaining nine bits specify the command to be executed.

The numerical values of the instruction codes for memory reference instructions are the same in two-word format as in single-word format, except that the contents of the instruction code field have been effectively shifted right three bits or one octal digit. Otherwise, their value is exactly that of the corresponding single-word commands. The two-word format is shown in Figure 3-4.

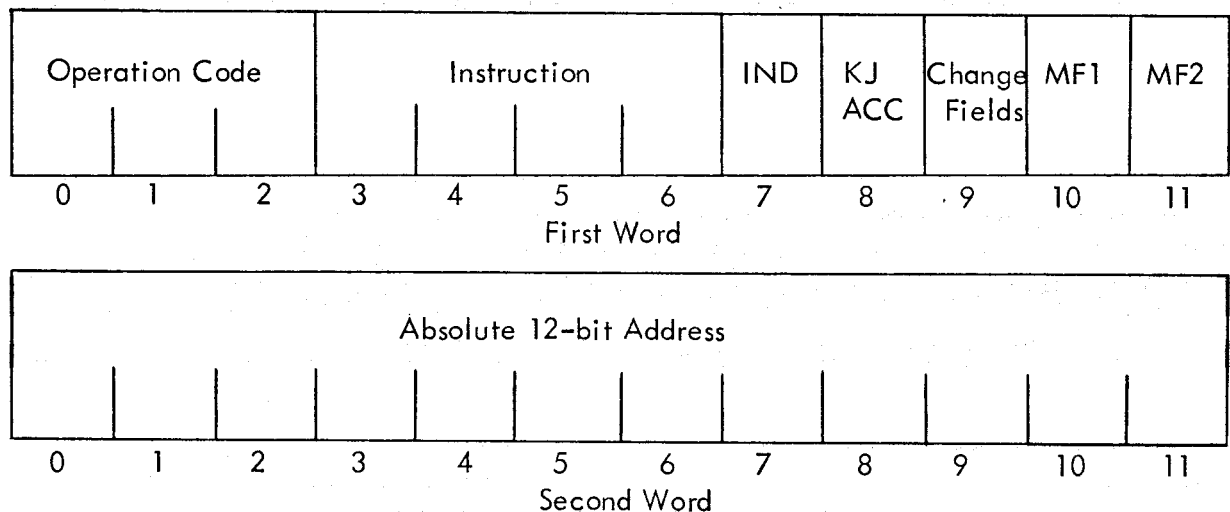


Figure 3-4. Two-Word Format

This format offers considerably more control than single word format. It provides the ability to address operands in fields other than the one in which the instruction resides. This ability generates from the fact that bits 10 and 11 can refer to any one of four fields, 00 to 11 (binary). However, by setting bit 9 to a zero, this effect could be cancelled. So, by controlling bit 9, it is possible to cause the absolute address in the second word to reference the current field or another field. If the instruction with bit 9 set to one is a jump command, or a jump-to-subroutine command, the change is permanent until it is changed by another jump or indirect jump. For all non-jump commands, the field selection change is only for the execute portion of that instruction.

The two-word format also allows determination of which of the main accumulators is employed in the operation specified (if applicable). If bit 8 is one, the ND812 will employ the upper accumulator (the K register) if it is zero, the ND812 will employ the lower accumulator (the J register). Bit 7 allows the selection of an indirect address. If bit 7 is a one, indirect addressing is specified.

The two-word format input/output instruction is characterized by its content of the value 0740 octal as a basic; the second word is used to develop device addresses and control. This permits 12-bit control words.

**3.3.2.3 LITERAL FORMAT.** It is unusual for a 12-bit word computer to incorporate literal instructions; the ND812 does. These instructions permit the programmer to save both time and storage space, because the literal instructions enable the storage of counter initialization constants, increment and decrement constants, and logical AND masks in the instruction which uses the value. This saves space otherwise needed to store the constants separately and the time to access these constants.

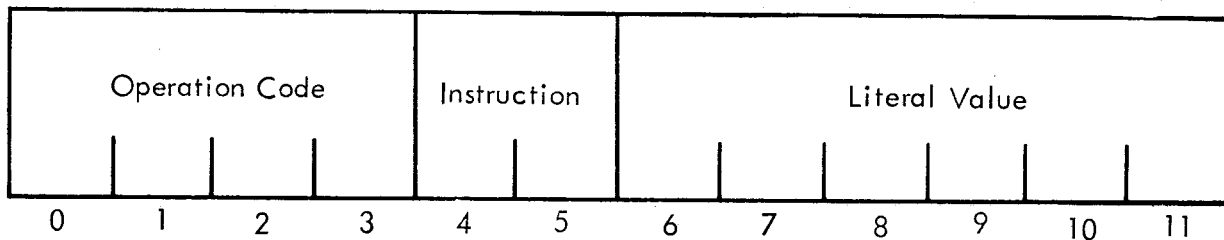


Figure 3-5. Literal Format

The values in bits 4 and 5 specify which operation is performed, while the value used is in bits 6 to 11. One special case uses the literal value to obtain a value located in memory as a 12-bit literal value.

3.3.2.4 GROUP 1 INSTRUCTION FORMAT. Instructions of the Group 1 class are characterized by the bit pattern 0010 in bit positions 0-3. They are generally concerned with performing arithmetic, logical, exchange and shifting functions in operations on the internal accumulator registers. This group also contains the hardware multiply and divide instructions. Most Group 1 instructions have the format shown in Figure 3-6.

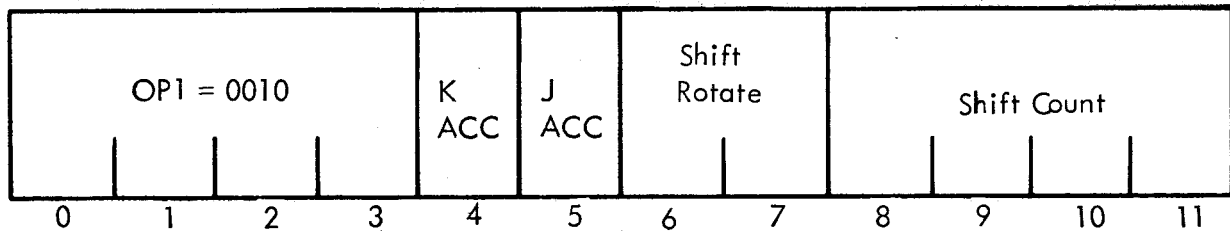


Figure 3-6. Group 1 Format

Bit 4 is set if the K register is to be affected; bit 5 is set if the J register is to be affected; and bits 4 and 5 are set if both the J and K registers are to be affected.

3.3.2.5 GROUP 2 INSTRUCTION FORMAT. The instructions of Group 2 are primarily concerned with testing for internal conditions of the main accumulators (registers J and K). Several variants of the Group 2 instructions can also test, set, clear, and complement the overflow and flag bits; others can complement, increment, and negate the contents of the J and K registers.

The instructions of this group are microprogrammable, i.e., they can be OR'ed together to produce both results. The bit pattern constituting the instruction may be combined to produce different effects. The format for Group 2 instructions is shown in Figure 3-7. Note that various bit positions have different assignments; some address and others control.

If it is desired, for instance, to determine a condition in the J register, a "1" in bit 5 would address it. The same would apply to bit 4, which addresses the K register. Bits 9, 10, and 11 all control the selection of conditions to be tested.

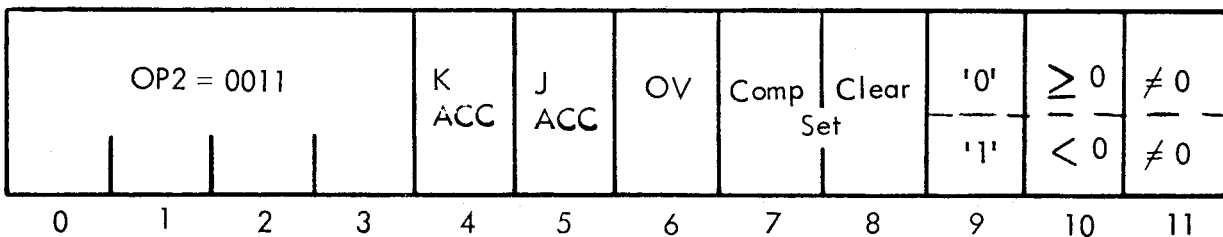


Figure 3-7. Group 2 Format

When a condition is tested via the group 2 instructions, the ND812 takes one of two possible actions:

- 1) If the condition tested is TRUE, the contents of the program counter are incremented again by one, so that the word immediately following is skipped over;
- 2) If the condition tested is FALSE, the contents of the program counter are not incremented and the instruction in the following location will be executed.

**3.3.2.6 STATUS WORD FORMAT.** The status register does not actually exist as a true register. It is the contents of several groups of indicators, all commonly accessed by storing them in the J register, when desired. Since each bit of each indicator is stored at a particular bit position of the J Register, it is customary to refer to this bit order as the bit assignments of the status register.

The bit assignments for the status word are shown in Figure 3-8. A single instruction will result in the storing of the indicated bits into the J register. The two, 2-bit fields, labeled JPS and INT are the storage for the current memory field contents whenever a Jump to Subroutine or Interrupt are encountered. These two bits are actually the values which are restored into the current execution memory field bits when the INT or JPS registers recognize that a return condition exists.

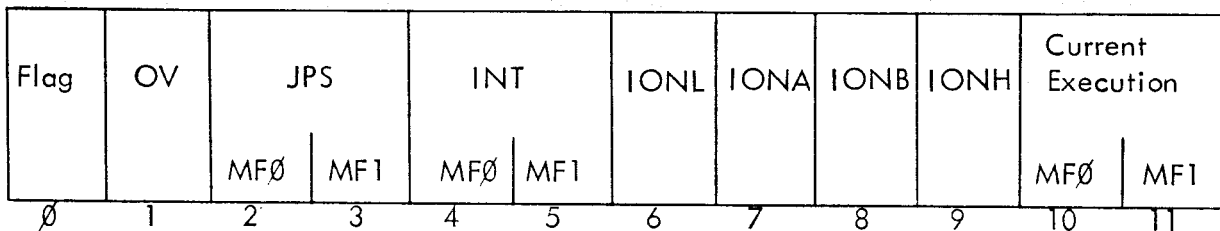


Figure 3-8. Status Word Bit Assignments (J-Register)

Two bits, the flag and overflow, appearing in J0 and J1, may be transferred to their respective registers by the execution of a RFOV instruction. However, most of the status bits must be restored by executing the instructions which create the conditions stored. For example, executing an IONH instruction to set bit 9 of the status register.

### 3.4 ADDRESSING

The memory storage locations which contain the instructions and data of a program are identified to the machine by their particular memory addresses. Every word in memory is directly addressable with a unique address.

An instruction is stored in a field of one or two words, depending on the type of instruction and the mode of address.

#### 3.4.1 DIRECT ADDRESSING

The two-word format is used to obtain direct addressing of all of memory. Because 12 bits can reference only 4096 ( $2^{12}$ ) locations, the last two bits of the first word specify which of the four possible memory stacks the address is in; through this combination, the ND812 has direct addressing of all memory.

#### 3.4.2 RELATIVE ADDRESSING

A relative address is always relative to the program counter; the single-word format is used to obtain the relative addressing of memory. Because the value contained is equivalent to the range 0 to 63 (decimal), that is the range of addresses which can be accessed. However a seventh bit (bit 5 of the word) can specify whether this range is forward or backward, so that a single-word memory reference command can access plus or minus  $63_{10}$  locations from its location in the program.

#### 3.4.3 INDIRECT ADDRESSING

The indirect addressing uses the relative addressing form or the single-word format. The only difference between the address forms is that bit 4 of the word is set to specify indirect addressing. Relative addressing is defined by the value being added or subtracted from the program counter to determine the location of the corresponding word. Indirect addressing is one step further (i.e., the effective address of the word is contained in the relative address). The relative address word specifies an address of the address; thus, the indirect addressing capability. Although single or two-word indirect addressing is possible, there is only a single level of indirect addressing possible in either case.

#### 3.4.4 AUTO-INDEX ADDRESSING

Two words in each memory field of the ND812 may be used as auto-index locations. These locations have the property that if they are addressed directly, their behavior is normal; that is, they act as the operand location and their contents are used normally. However, if they are indirectly addressed by a single-word instruction, they first increment their contents by one and store the resulting value as their contents (which points to the operand). The ND812 uses the modified contents of the auto-index location to access the operand desired.

Single-word format instructions may address these two locations relatively, indirectly, and directly; however, the operand (with one exception) must be in the field in which that instruction resides.

When a single-word format instruction directly accesses either of the two locations, it specifies as much with a special value in the displacement field, ("00" octal). The forward/backward bit specifies which of the two locations is used; the direct or indirect bit specifies whether the contents of the auto-index location are the operand or point to the operand.

Two-word format memory reference instructions may also use the auto-index location (both as an operand and as the pointer to the operand in an indirect address). When used indirectly (in the two word format), the auto-index locations do not automatically increment.

## SECTION IV INSTRUCTION REPERTOIRE

### 4.1 GENERAL

This chapter describes the instruction set for the ND812. The instructions are described in functional order. The ND812 repertoire includes nine types of instructions: memory reference, logical, arithmetic, shift/rotate, load and exchange, control, literal, input/output, and miscellaneous.

Within each group the instructions are described in detail. Listed in each entry, from left to right, are the assembler mnemonic, the octal code, a verbal description, and the affected registers. Below the instruction mnemonic is a description of the effective operation of that instruction and any restrictions or suggestions.

### 4.2 MEMORY REFERENCE INSTRUCTIONS

All instructions which can reference memory for the word to be used in the execution of an instruction are called memory reference instructions. They include all loads, stores, compares, most addition and subtraction, and the increment and decrement contents of memory instructions. Jump and jump-to-subroutine are also classed as memory reference instructions.

There are two classes of memory reference instructions: two-word and single-word. Chapter 3 contains complete descriptions of their formats. In the following listing, if an instruction can have either the single or two-word format, the first line of the description is the single-word format and the second is the two-word.

|      |      |                     |   |
|------|------|---------------------|---|
| ANDF | 20XX | AND with J, Forward | J |
|------|------|---------------------|---|

Quantity (12-bit) located within 63 locations of this instruction is ANDed with contents of J. Result replaces previous contents of J. Memory is unaltered. Only forward, relative addressing is permitted. Indirect bit (bit 4) must be used for part of operation code; therefore, execution time is fixed. Literal in this case is really displaced 12-bit value rather than indirect address pointer.

|       |      |                     |   |
|-------|------|---------------------|---|
| LDJ   | 5000 | Single-Word, Load J | J |
| TWLDJ | 0500 | Two-Word, Load J    |   |



Loads J with contents of effective memory address. Original contents of J are lost. Memory is unchanged. All address modes are permitted.

|       |      |                      |        |
|-------|------|----------------------|--------|
| STJ   | 5400 | Single-Word, Store J | Memory |
| TWSTJ | 0540 | Two-Word, Store J    |        |

Stores contents of J in contents of effective memory address. Original contents of memory are lost. Contents of J are unchanged. All address modes are permitted.

|       |      |                  |   |
|-------|------|------------------|---|
| TWLDK | 0510 | Two-Word, Load K | K |
|-------|------|------------------|---|

Loads K with contents of effective memory address. Original contents of K are lost. Memory is unchanged. Address modes are direct and indirect.

|       |      |                   |        |
|-------|------|-------------------|--------|
| TWSTK | 0550 | Two-Word, Store K | Memory |
|-------|------|-------------------|--------|

Stores contents of K in contents of effective memory address. Original contents of memory are lost. Contents of K are unchanged. Address modes are direct and indirect.

|       |      |                    |       |
|-------|------|--------------------|-------|
| ADJ   | 4400 | Single-Word, Add J | J, OV |
| TWADJ | 0440 | Two-Word, Add J    |       |

Adds contents of effective address to contents of J. The sum appears in J. Overflow will complement overflow bit. All address modes are permitted.

|       |      |                              |       |
|-------|------|------------------------------|-------|
| SBJ   | 4000 | Single-Word, Subtract from J | J, OV |
| TWSBJ | 0400 | Two-Word, Subtract from J    |       |

Subtracts contents of effective address from contents of J. The difference appears in J. Overflow will complement overflow bit. All address modes are permitted.

|       |      |                 |       |
|-------|------|-----------------|-------|
| TWADK | 0450 | Two-Word, Add K | K, OV |
|-------|------|-----------------|-------|

Adds contents of effective address to contents of K. The sum appears in K. Overflow will complement overflow bit. Address modes are direct and indirect.

|       |      |                           |       |
|-------|------|---------------------------|-------|
| TWSBK | 0410 | Two-Word, Subtract from K | K, OV |
|-------|------|---------------------------|-------|

Subtracts contents of effective address from contents of K. The difference appears in K. Overflow will complement overflow bit. Address modes are direct and indirect.

|       |      |   |            |
|-------|------|---|------------|
| ISZ   | 3400 | Single-Word, Increment<br>Memory and Skip if Zero | Memory, PC |
| TWISZ | 0340 | Two-Word, Increment<br>Memory and Skip if Zero    |            |

Increments contents of effective address by one. If result equals zero, next location is skipped. Overflow is not possible. All address modes are permitted.

|       |      |   |            |
|-------|------|---|------------|
| DSZ   | 3000 | Single-Word, Decrement<br>Memory and Skip if Zero | Memory, PC |
| TWDSZ | 0300 | Two-Word, Decrement<br>Memory and Skip if Zero    |            |

Decrements contents of effective address by one. If result equals zero, next location is skipped. Overflow is not possible. All address modes are permitted.

|       |      |  |    |
|-------|------|--|----|
| SMJ   | 2400 | Single-Word, Skip<br>if Memory Not Equal J | PC |
| TWSMJ | 0240 | Two-Word, Skip<br>if Memory Not Equal J    |    |

Compares contents of effective address with contents of J. If not equal, next location is skipped. If equal, next location is accessed. Contents of memory and contents of J are not altered. All address modes are permitted.

|       |      |   |    |
|-------|------|---|----|
| TWSMK | 0250 | Two-Word, Skip<br>if Memory Not Equal K | PC |
|-------|------|---|----|

Compares contents of effective address with contents of K. If not equal, next location is skipped. If equal, next location is accessed. Contents of memory and contents of K are not altered. Address modes are direct and indirect.

|     |      |                                    |    |
|-----|------|------------------------------------|----|
| JMP | 6000 | Single-Word,<br>Unconditional Jump | PC |
|-----|------|------------------------------------|----|

Relative and indirect addressing are permitted. Relative addressing results in algebraic sum of displacement and current contents of program register replacing current contents of Program Counter. Indirect addressing results in indirectly addressed value obtained from pointer replacing 12-bit contents of program counter.

|       |      |                                 |    |
|-------|------|---------------------------------|----|
| TWJMP | 0600 | Two-Word,<br>Unconditional Jump | PC |
|-------|------|---------------------------------|----|

Replaces contents of program counter with contents of address portion or contents of indirectly addressed location. Contents of program counter are lost. If field bits are set and selection bit 9 is set, jump can be to another field.

|     |      |                                |            |
|-----|------|--------------------------------|------------|
| JPS | 6400 | Single-Word<br>Jump Subroutine | Memory, PC |
|-----|------|--------------------------------|------------|

Relative and indirect addressing are permitted. Relative addressing results in algebraic sum of displacement and current contents of program counter replacing current contents of memory address register. Current contents of program counter are then written into

memory at address loaded into the memory address register. The memory address register is then incremented and placed into program counter (replacing its original contents). If indirect addressing is employed, operand address obtained from location obtained by algebraically adding program counter and displacement is placed in memory address register. Contents of program counter are written into memory at that location. Contents of memory address register are incremented. Result replaces current contents of program counter.

|       |      |                             |            |
|-------|------|-----------------------------|------------|
| TWJPS | 0640 | Two-Word Jump<br>Subroutine | Memory, PC |
|-------|------|-----------------------------|------------|

Direct and indirect addressing are permitted. For direct or indirect addressing, contents of second word (or word pointed to by indirect pointer) are loaded into memory address register. Contents of program counter are written into memory at that location. Contents of memory address register are incremented. Result replaces contents of program counter.

|     |      |                     |
|-----|------|---------------------|
| XCT | 7000 | Execute Instruction |
|-----|------|---------------------|

Execute enables performance of all instructions except JMP and TWJMP without changing current contents of program counter. Is especially useful for programs which are varied in function each time they are run. No two-word form exists. Relative and indirect addressing are permitted.

Instruction located at effective address may be any legal ND812 instruction (single-word or two-word). This includes execute instruction itself. Obvious error to avoid is executing execute instruction which referenced first execute (creating endless loop). If jump is executed by XCT command, contents of program counter are changed and original contents are lost.

XCT instruction with indirect address may also execute instruction with indirect address. This is only way to get more than single level of indirect addressing.

### 4.3 LOGICAL OPERATIONS

The ND812 can perform the logical AND function by using the contents of accumulator registers J and K as a mask. The result appears in the register shown in each instruction, but it is always one of the two arithmetic registers. All require one memory cycle. The logical operation instructions use the Group I instruction format described in Section III.

|       |      |                            |   |
|-------|------|----------------------------|---|
| AND J | 1100 | Logical AND J, K<br>into J | J |
|-------|------|----------------------------|---|

Using contents of K as mask, logical AND function is performed on contents of the J & K accumulators. Results replace previous contents of J. K is not altered. AND states that resultant bit is zero unless corresponding bits in both accumulators are "ones".

|       |      |                            |   |
|-------|------|----------------------------|---|
| AND K | 1200 | Logical AND J,<br>K into K | K |
|-------|------|----------------------------|---|

Using contents of J as mask, logical AND function is performed on contents of the J & K accumulators. Results replace previous contents of K. J is not altered. AND states that resultant bit is zero unless corresponding bits in both accumulators are ones.

|        |      |                               |      |
|--------|------|-------------------------------|------|
| AND JK | 1300 | Logical AND J,<br>K into J, K | J, K |
|--------|------|-------------------------------|------|

Using K as mask, logical AND function is performed on contents of the J & K accumulators. Results replace contents of the J & K accumulators. AND states that each resultant bit is zero unless corresponding bits in both accumulators are "ones".

#### 4.4 ARITHMETIC OPERATIONS ON ACCUMULATOR REGISTERS

This instruction group includes addition, subtraction, multiplication, and division operations. All require one memory cycle to perform, (except for the multiply and divide), and all four accumulator registers may participate. Overflow will complement the overflow bit. These instructions all use the Group 1 instruction format described in Section III.

|       |      |            |       |
|-------|------|------------|-------|
| AJK J | 1120 | J + K to J | J, OV |
|-------|------|------------|-------|

Adds contents of J to contents of K. Replaces contents of J with sum. Overflow is possible. K is unaltered.

|        |      |               |       |
|--------|------|---------------|-------|
| NAJK J | 1130 | -(J + K) to J | J, OV |
|--------|------|---------------|-------|

Adds contents of J to contents of K. Negates sum. Replaces contents of J with negated sum. Overflow is possible, K is unaltered.

|       |      |            |       |
|-------|------|------------|-------|
| SJK J | 1121 | J - K to J | J, OV |
|-------|------|------------|-------|

Subtracts contents of K from contents of J. Replaces contents of J with difference. Overflow is possible. K is unaltered.

|        |      |               |       |
|--------|------|---------------|-------|
| NSJK J | 1131 | -(J - K) to J | J, OV |
|--------|------|---------------|-------|

Subtracts contents of K from contents of J. Negates difference. Replaces contents of J with negated difference. Effect is to subtract J from K and place difference in J. Overflow is possible. K is unaltered.

|       |      |            |       |
|-------|------|------------|-------|
| ADR J | 1122 | R + J to J | J, OV |
|-------|------|------------|-------|

Adds contents of R to contents of J. Replaces contents of J with sum. Overflow is possible. R is unaltered.

NADR J      1132       $-(R + J)$  to J      J, OV

Adds contents of R to contents of J. Negates sum. Replaces contents of J with negated sum. Overflow is possible. R is unaltered.

ADS J      1124       $S + J$  to J      J, OV

Adds contents of S to contents of J. Replaces contents of J with sum. Overflow is possible. S is unaltered.

NADS J      1134       $-(S + J)$  to J      J, OV

Adds contents of S to contents of J. Negates sum. Replaces contents of J with negated sum. Overflow is possible. S is unaltered.

SBR J      1123       $R - J$  to J      J, OV

Subtracts contents of J from contents of R. Replaces contents of J with difference. Overflow is possible. R is unaltered.

NSBR J      1133       $-(R - J)$  to J      J, OV

Subtracts contents of J from contents of R. Negates difference. Replaces contents of J with negated difference. Effect is to subtract R from S and place difference in J. Overflow is possible. R is unaltered.

SBS J      1125       $S - J$  to J      J, OV

Subtracts contents of J from contents of S. Replaces contents of J with difference. Overflow is possible. S is unaltered.

NSBS J      1135       $-(S - J)$  to J      J, OV

Subtracts contents of J from contents of S. Negates difference. Replaces contents of J with negated difference. Effect is to subtract S from J and place difference in J. Overflow is possible. S is unaltered.

AJK K      1220       $J + K$  to K      K, OV

Adds contents of J to contents of K. Replaces contents of K with sum. Overflow is possible. J is unaltered.

NAJK K      1230       $-(J + K)$  to K      K, OV

Adds contents of J to contents of K. Negates sum. Replaces contents of K with negated sum. Overflow is possible. J is unaltered.

SJK K            1221             $J - K$  to K            K, OV

Subtracts contents of K from contents of J. Replaces contents of K with difference. Overflow is possible. J is unaltered.

NSJK K           1231             $-(J - K)$  to K

Subtracts contents of K from contents of J. Negates difference. Replaces contents of K with negated difference. Effect is to subtract J from K and place difference in K. Overflow is possible. J is unaltered.

ADR K            1222             $R + K$  to K            K, OV

Adds contents of R to contents of K. Replaces contents of K with sum. Overflow is possible. R is unaltered.

NADR K           1232             $-(R + K)$  to K            K, OV

Adds contents of R to contents of K. Negates sum. Replaces contents of K with negated sum. Overflow is possible. R is unaltered.

ADS K            1224             $S + K$  to K            K, OV

Adds contents of S to contents of K. Replaces contents of K with sum. Overflow is possible. S is unaltered.

NADS K           1234             $-(S + K)$  to K            K, OV

Adds contents of S to contents of K. Negates sum. Replaces contents of K with negated sum. Overflow is possible. S is unaltered.

SBR K            1223             $R - K$  to K            K, OV

Subtracts contents of K from contents of R. Replaces contents of K with difference. Overflow is possible. R is unaltered.

NSBR K           1233             $-(R - K)$  to K            K, OV

Subtracts contents of K from contents of R. Negates difference. Replaces contents of K with negated difference. Effect is to subtract R from K and place difference in K. Overflow is possible. R is unaltered.

SBS K            1225             $S - K$  to K            K, OV

Subtracts contents of K from contents of S. Replaces contents of K with difference. Overflow is possible. S is unaltered.

NSBS K            1235             $-(S - K)$  to K            K, OV

Subtracts contents of K from contents of S. Negates difference. Replaces contents of K with negated difference. Effect is to subtract S from K and place difference in K. Overflow is possible. S is unaltered.

AJK JK            1320             $J + K$  to J, K            J, K, OV

Adds contents of J to contents of K. Replaces contents of both J and K with sum. Is not a 24-bit sum in two registers. Overflow is possible. Both J and K are altered.

NAJK JK            1330             $-(J + K)$  to J, K            J, K, OV

Adds contents of J to contents of K. Negates sum. Replaces contents of both J and K with negated sum. Is not a 24-bit sum in two registers. Overflow is possible. Both J and K are altered.

SJK JK            1321             $J - K$  to J, K

Subtracts contents of K from contents of J. Replaces contents of both J and K with difference. Overflow is possible. Both J and K are altered.

NSJK JK            1331             $-(J - K)$  to J, K

Subtracts contents of K from contents of J. Negates difference. Replaces contents of both J and K with negated difference. Effect is to subtract J from K and place difference in both J and K. Overflow is possible. Both J and K are altered.

MPY                1000            Multiply J by K            J, K, R, S, OV

Logically multiplies contents of J by contents of K. Multiplication requires all four accumulators. Multiplier is loaded into J register and multiplicand into K register. Product appears in sub-accumulators. Most significant half goes into S and least significant half into R. R and S are cleared prior to starting of product accumulation. They do not require instructions to clear prior to multiplication. Multiplier and multiplicand are assumed to be positive integers. J, K, R and S are altered by multiply. Overflow in Multiplication is not possible, but the previous contents of the OV indicator may be destroyed.

Fixed multiplication time makes it possible to accurately estimate execution time of process control and real-time programs as they are written. Because ND812 has single-word and two-word instructions which can load J and K register before commencement of multiplication, time-critical situations and noncritical situations can be solved.

DIV                    1001                    Divide K, J by R                    J, K, R, OV, S

Logically divides contents of J and K by contents of R. Divide is also a register-to-register operation. Previous to execution of DIV, divisor must be loaded into R and dividend in K and J. Most significant half of 24-bit dividend resides in K; Least significant half of dividend is in J. At completion of DIV, quotient appears in J and remainder in K. S may be altered on divide. Overflow clears at start of DIV operation.

If contents of R is less than contents of K and J at beginning of DIV, divide error is indicated by setting of overflow bit to one and termination of any activity on remainder of DIV time. None of the factors in J, K or R registers is altered in event of divide overflow. Condition of overflow register should be tested at completion of DIV instruction. Divide by zero also sets overflow register to one and terminates DIV activity.

While divide is infrequent in operation of most programs, it is of value on programs doing non-integer factor scaling, ratioing of variables, etc. Because these are usually real-time operations, the fact that divide is fast and fixed in execution time is of great benefit on pre-analysis and program writing for fixed reaction times.

#### 4.5            SHIFT/ROTATE INSTRUCTIONS

A single instruction can shift or rotate the contents of J or K or both up to 15 bit positions. The four-bit value in bits 8 to 11 of the instruction specify the number of positions to be shifted. The actual time to shift or rotate each bit in the J and K registers is 0.125  $\mu$ s per bit. In a full cycle, eight shift periods are available. This allows up to 8 bit shifts or rotates in a single memory cycle. More than 8 bit positions shifted or rotated will take longer, but only the amount of time consumed for the number of bits shifted in excess of eight. The shift/rotate instructions use the Group 1 instruction format described in Section III.

SFTZ J                    1140                    Shift J Left N                    J

Shifts contents of J left by N bits. N ranges from 0 to 15 (as specified by bits 8 to 11 of instruction). Each bit position shifted requires 0.125  $\mu$ s. 8 or fewer bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Bits shifted out of Bit 0 of J are lost. Zeroes are shifted into Bit 11. Overflow bit is unaffected.

SFTZ K                    1240                    Shift K Left N                    K

Shifts contents of K left by N bits. N ranges from 0 to 15 (as specified by bits 8 to 11 of instruction). Each bit position shifted requires 0.125  $\mu$ s. 8 or fewer bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Bits shifted out of Bit 0 of K are lost. Zeroes are shifted into Bit 11. Overflow bit is unaffected.



SFTZ JK      1340      Shift J to K Left N      J, K

Shifts contents of both J and K left by N bits. N ranges from 0 to 15 (as specified by bits 8 to 11 of instruction). Bit 0 of J is shifted into bit 11 of K and bits shifted out of bit 0 of K are lost. Zeroes are shifted into bit 11 of J. Each bit position shifted requires 0.125  $\mu$ s. 8 or fewer bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Overflow bit is unaffected.

ROTD J      1160      Rotate J Left N      J

Rotates contents of J left N bits. N ranges from 0 to 15 (as specified by bits 8 to 11 of instruction). Bit 0 of J is shifted into bit 11 of J. No bits are lost. Each bit position rotated requires 0.125  $\mu$ s. 8 or fewer bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Overflow bit is unaffected.

ROTD K      1260      Rotate K Left N

Rotates contents of K left N bits. N ranges from 0 to 15 (as specified by bits 8 to 11 of instruction). Bit 0 of K is shifted into Bit 11 of K. No bits are lost. Each bit position rotated requires 0.125  $\mu$ s. 8 or fewer bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Overflow bit is unaffected.

ROTD JK      1360      Rotate J, K Left N      J, K

Rotates contents of both J and K left N bits. N ranges from 0 to 15 bits (as specified by bits 8 to 11 of instruction). Bit 0 of K is shifted into Bit 11 of J. Bit 0 of J goes into bit 11 of K. No bits are lost. Each bit position rotated requires 0.125  $\mu$ s. 8 or less bits can be shifted in 1 memory cycle. More than 8 automatically obtain required delay to complete. Overflow bit is unaffected.

Because J and K are each 12-bits long, effective right shift of either can be performed in single ROTD J or K. Example: to effectively right shift J three positions, execute ROTD J, 9 places.

#### 4.6      LOAD AND EXCHANGE OPERATIONS

This group of instructions enables the exchange of information between the accumulators and the switch and status registers. No other method is provided for loading and storing the contents of the two sub-accumulators, R and S, for they lack a direct route to memory.

The Status Register enables storage of internal status conditions in the event of a power failure condition. It stores the present conditions of the overflow register, flag register, enabled interrupts, current memory field, and the INTERRUPT and JPS memory fields. The load and exchange operations enable, among other things, the ability to store and reload status conditions. These instructions use the Group I instruction format described in Section III. All instructions in this grouping require 1 cycle for execution.

LJSW            1010            Load J From Switch Register    J

Replaces contents of J with contents of switch register, as determined by positions of front panel SWITCH REGISTER switches.

LRF J            1101            Load R From J                    R

Replaces contents of R with contents of J. J is unaltered.

LJFR            1102            Load J From R                    J

Replaces contents of J with contents of R. R is unaltered.

EXJR            1103            Exchange J and R                 R, J

Exchanges contents of J and contents of R. Information is exchanged without alteration.

LSFK            1201            Load S from K                    R

Replaces contents of S with contents of K. K is unaltered.

LKFS            1202            Load K from S                    K

Replaces contents of K with contents of S. S is unaltered.

EXKS            1203            Exchange K and S                 S, K

Exchanges contents of K and contents of S. Information is exchanged without alteration.

LKFJ            1204            Load K from J                    K

Replace contents of K with contents of J. J is unaltered.

EXJK            1374            Exchange J and K                 J, K

Exchanges contents of J and contents of K. Information is exchanged without alteration.

LRSFJK        1301            Load R, S from J, K              R, S

Replaces contents of R with contents of J. Replaces contents of S with contents of K. Both J and K are unaltered.

LJKFRS        1302            Load J, K from R, S              J, K

Replaces contents of J with contents of R. Replaces contents of K with contents of S. Both R and S are unaltered.

EXJRKS          1303          Exchange J, K and R, S          J, K, R, S

Exchanges contents of J with contents of R. Exchanges contents of K with contents of S. Information is exchanged without alteration.

LJST            1011          Load Status Register in J          J

Replaces contents of J with contents of status register. All bit positions are represented in J. If contents of J are stored in memory after loading of J, information may subsequently be used to return ND812 to its original status.

RFOV           1002          Read Flag, OV from J          J

Contents of J (bits 0 and 1) are ORed into flag and overflow bits. Flag and overflow bits should be clear prior to this instruction. No other status register bits are affected by this instruction.

## 4.7 CONTROL INSTRUCTIONS

### 4.7.1 CONDITIONAL SKIPS

This instruction group tests the respective registers for certain conditions. If the condition is true, the next instruction is skipped; otherwise the next instruction is executed. All instructions in this grouping require 1 cycle for execution. These instructions use the Group 2 instruction format described in Section III.

SIZ J           1505          Skip if J equals zero          PC

Tests contents of J for all-zero. If true, skips next word; otherwise, next word is executed.

SIZ K           1605          Skip if K equals zero          PC

Tests contents of K for all-zero. If true, skips next word; otherwise, next word is executed.

SIZ JK          1705          Skip if both J and K  
equal zero          PC

Tests contents of both J and K for all-zero condition. If both J and K equal zero, next word is skipped; otherwise, next word is executed.

SNZ J          1501          Skip if J not  
equal zero          PC

Tests contents of J for presence of at least single one. If true, skips next word; otherwise, next word is executed.

|       |      |                                |    |
|-------|------|--------------------------------|----|
| SNZ K | 1601 | Skip if K not<br>equal to zero | PC |
|-------|------|--------------------------------|----|

Tests contents of K for presence of at least single one. If true, skips next word; otherwise, next word is executed.

|        |      |                                  |    |
|--------|------|----------------------------------|----|
| SNZ JK | 1701 | Skip if J or K<br>not equal zero | PC |
|--------|------|----------------------------------|----|

Tests contents of both J and K for presence of at least single one. If true, skips next word; otherwise, next word is executed.

|       |      |                    |    |
|-------|------|--------------------|----|
| SIP J | 1502 | Skip if J positive | PC |
|-------|------|--------------------|----|

If J bit zero equals zero, value contained is positive. All-zero also tests as positive. If true, skips next word; otherwise, next word is executed.

|       |      |                    |    |
|-------|------|--------------------|----|
| SIP K | 1602 | Skip if K positive | PC |
|-------|------|--------------------|----|

If K bit zero equals zero, value contained is positive. All-zero also tests as positive. If true, skips next word; otherwise, next word is executed.

|        |      |                                  |    |
|--------|------|----------------------------------|----|
| SIP JK | 1702 | Skip if both J and K<br>positive | PC |
|--------|------|----------------------------------|----|

If bit zero of both J and K is zero, value contained in both is positive. All-zero also tests as positive. If true, skips next word; otherwise, next word is executed.

|       |      |                    |    |
|-------|------|--------------------|----|
| SIN J | 1506 | Skip if J negative | PC |
|-------|------|--------------------|----|

If bit zero of J is one, value contained is negative. If true, skips next word; otherwise, next word is executed.

|       |      |                    |    |
|-------|------|--------------------|----|
| SIN K | 1606 | Skip if K negative | PC |
|-------|------|--------------------|----|

If bit zero of K is one, value contained is negative. If true, skips next word; otherwise, next word is executed.

|        |      |                                  |    |
|--------|------|----------------------------------|----|
| SIN JK | 1706 | Skip if both J and K<br>negative | PC |
|--------|------|----------------------------------|----|

If Bit of J and K are both one, value contained in both is negative. If true, skips next word; otherwise, next word is executed.

#### 4.7.2 CLEAR, COMPLEMENT, INCREMENT AND SET

The instructions in this group can clear, complement, increment, and set the registers. All instructions in this grouping require 1 cycle for execution and use the Group 2 instruction format described in Section III.

|  |      |                         |      |
|--|------|-------------------------|------|
| CLR J  | 1510 | Clear J                 | J    |
| Unconditionally sets all bits of J to zero.                  |      |                         |      |
| CLR K  | 1610 | Clear K                 | K    |
| Unconditionally sets all bits of K to zero.                  |      |                         |      |
| CLR JK   | 1710 | Clear both J and K      | J, K |
| Unconditionally sets all bits of both J and K to zero.       |      |                         |      |
| CMP J  | 1520 | Complement J            | J    |
| Changes all 1 bits to 0 and all 0 bits to 1 in J.            |      |                         |      |
| CMP K  | 1620 | Complement K            | K    |
| Changes all 1 bits to 0 and all 0 bits to 1 in K.            |      |                         |      |
| CMP JK   | 1720 | Complement both J and K | J, K |
| Changes all 1 bits to 0 and all 0 bits to 1 in both J and K. |      |                         |      |
| SET J  | 1530 | Set J to -1             | J    |
| Sets J to all one's.   |      |                         |      |
| SET K  | 1630 | Set K to -1             | K    |
| Sets K to all one's.   |      |                         |      |
| SET JK   | 1730 | Set both J and K to -1  | J, K |
| Sets both J and K to all one's.                              |      |                         |      |

### 4.7.3 OVERFLOW BIT INSTRUCTIONS

The overflow bit is a part of the arithmetic unit employed to indicate whether an overflow condition existed on the last operation. It can also be program-controlled as part of a program's logic. Every arithmetic operation, whether memory reference or operate, can complement the overflow bit. It should be tested immediately after an arithmetic operation which might generate an overflow condition of interest. These instructions use the Group 2 instruction format described in Section III.

SIZ O            1445            Skip if Overflow Zero            PC

If overflow bit is zero, skip next word; otherwise, execute next word.

SNZ O            1441            Skip if Overflow One            PC

If overflow bit is one, skip next word; otherwise, execute next word.

CLR O            1450            Clear Overflow            0V

Unconditionally sets overflow bit to zero.

CMP O            1460            Complement Overflow            0V

If overflow bit is zero, set to one; if one, set to zero.

SET O            1470            Set Overflow            0V

Unconditionally sets overflow bit to one.

### 4.7.4 FLAG BIT INSTRUCTIONS

The flag bit can be set, cleared, complemented, and tested by the program. It can therefore be used to indicate the presence of some condition, remember a program branching condition, or indicate the state of some external condition. These instructions use the Group 2 instruction format described in Section III.

SIZ                1405            Skip if Flag Zero            PC

If flag bit is zero, skip next word; otherwise, execute next word.

SNZ                1401            Skip if Flag One            PC

If flag bit is one, skip next word; otherwise, execute next word.

|     |      |            |   |
|-----|------|------------|---|
| CLR | 1410 | Clear Flag | F |
|-----|------|------------|---|

Unconditionally sets flag bit to zero.

|     |      |                 |   |
|-----|------|-----------------|---|
| CMP | 1420 | Complement Flag | F |
|-----|------|-----------------|---|

If flag bit is zero, set to one; if one, set to zero.

|     |      |          |   |
|-----|------|----------|---|
| SET | 1430 | Set Flag | F |
|-----|------|----------|---|

Unconditionally sets flag bit to one.

#### 4.7.5 INCREMENT AND NEGATE

Although it is possible to microprogram (e.g., "OR") the several kinds of instructions described in paragraphs 4.7.1 through 4.7.4, not all such microprogrammed instructions are either meaningful or executable. Specifically, the functions of increment and negate (two's complement) are mutually exclusive with any skip instruction, because the three bits 9 to 11 which specify skipping conditions, must not contain any pattern other than "100" (octal 4); otherwise, incrementation and negation may not be performed. These instructions use the Group 2 instruction format described in Section III. All instructions in this grouping are performed in one memory cycle. There are no execute cycles.

|       |      |             |   |
|-------|------|-------------|---|
| INC J | 1504 | Increment J | J |
|-------|------|-------------|---|

Adds one to contents of J. Replaces contents of J with sum. If previous contents of J were  $7777_8$ , overflow is complemented.

|       |      |             |   |
|-------|------|-------------|---|
| INC K | 1604 | Increment K | K |
|-------|------|-------------|---|

Adds one to contents of K. Replaces contents of K with sum. If previous contents of K were  $7777_8$ , overflow is complemented.

|        |      |                        |      |
|--------|------|------------------------|------|
| INC JK | 1704 | Increment both J and K | J, K |
|--------|------|------------------------|------|

Adds one to contents of both J and K. Replaces each with its own incremented contents. If contents of either J or K were  $7777_8$  prior to execution of this instruction, overflow is complemented.

|       |      |          |   |
|-------|------|----------|---|
| NEG J | 1524 | Negate J | J |
|-------|------|----------|---|

Complements and increments contents of J, leaving result in J. Effect is to generate two's complement of value. Overflow bit is unaltered unless contents of J were zero.

NEG K            1624            Negate K            K

Complements and increments contents of K, leaving result in K. Effect is to generate two's complement of value. Overflow bit is unaltered unless contents of K were zero.

NEG JK           1724            Negate both J and K           J, K

Complements and increments contents of both J and K, leaving result of each in itself. Effect is to generate two's complements of their separate values. Overflow bit is unaltered unless contents of J or K were zero.

#### 4.7.6 INTERRUPT INSTRUCTIONS

The interrupt system is controlled by instructions in the instruction Group 2 set. These instructions control the enabling and disabling of the three interrupt-enable lines to the peripheral devices. For ease of use, the system is treated as if there were four possible interrupt enable conditions:

1. The highest level ("H");
2. The high and middle level ("A")
3. The high and middle level ("B")
4. All levels.

The very highest priority devices (such as ADC's) are not connected to the ND812 via any of the interrupt-enable levels. Instead, such devices are directly connected to the interrupt request line so that if the interrupt system itself is enabled by at least an IONH instruction, those devices can always request an interrupt at any time.

The instructions in this set of Group 2 instructions can merely enable or disable levels; they do not themselves generate an interrupt request, nor do they initiate the interrupt response routines, except insofar as the devices they enable have the ability to "trap" the ND812 to such a routine.

It must be emphasized that a wire named "level H" does not exist. The term "level H" is for convenience of reference, and means simply that one is referencing the interrupt flip-flop itself. It can be understood that when any device is directly connected to the interrupt request line, it is actually requesting that the interrupt state flip-flop be set. If a device is connected via one of the interrupt enable levels, the level must be low if the device is to generate the interrupt request. Otherwise, there is no real difference between "level H" and the other levels. It should be remembered that it is the use which causes the difference.

When the ND812 recognizes the interrupt request, it effectively disables the interrupt request line. This prevents an interrupt response from being interrupted itself until safe.



This is assured by the program re-enabling only those interrupt enables which are desired upon completing the interrupt request, thus re-enabling the device which causes the particular interrupt request.

IONH            1004            Enable Level H

Enables interrupt system and all devices directly connected via the interrupt request line. Any device not furnishing trap address traps to MFØ, location 0001. Until this instruction is executed, no devices of high priority can generate an interrupt request (sets bit 9 in Status register).

IONA            1006            Enable H and Level A

Enables interrupt system and interrupt enable level B. Any devices on level B can then initiate interrupt requests. Devices not furnishing trap address will trap to MFØ, location 0001. No devices on level B can initiate interrupt requests unless this instruction or IONN instruction is executed (sets bits 8 and 9 in Status register).

IONB            1005            Enable H and Level B

Enables interrupt system and interrupt enable level A. Any devices on level A can then initiate interrupt requests. Devices not furnishing trap address will trap to MFØ, location 0001. No devices on level A can initiate interrupt requests unless this instruction or IONN instruction is executed (sets bits 7 and 9 in Status register).

IONN            1007            Enable All Levels

Enables interrupt system and all interrupt enable levels. Any devices then present can generate an interrupt request. If some devices present do not have ability to generate trap address, ND812 responds by trapping to MFØ, location 0001. This instruction enables all devices present to initiate interrupt requests (sets bits 6,7,8, and 9 in Status register).

IOFF            1003            Disable All Interrupts

Disables all lines so that they do not respond to interrupt request (clears bits 6,7,8 and 9 in Status register).

It should be noted once again that this instruction is given if all devices are to be prevented from interrupting. However, when ND812 responds to interrupt request, it effectively executes an IOFF instruction, because it disables all interrupt enable levels; (but does not clear the Status word bits); they must be re-enabled as required by executing proper enable instructions.

#### 4.7.7    POWERFAIL LOGIC INSTRUCTIONS

Powerfail instructions are concerned solely with use of the internal powerfail detector logic. This logic monitors internal voltage levels from the power supply and determines whether they are at a certain predetermined level. When it detects a deviation which is considered

dangerous, it will raise its "Power Low" Flag. If the logic has been enabled to interrupt, and if the interrupt system itself is allowed, this action will also generate an interrupt request to the ND812. The logic will raise its flag even if the powerfail interrupt is not enabled and can still be program-tested.

There is approximately 1 ms of secure power remaining after a powerfail is detected (hence, many memory cycles before complete failure); however, because external peripheral conditions may take longer to execute, this condition should be the first tested in the polling routine.

Powerfail may be allowed (turned ON) or disallowed (turned OFF). It must be emphasized that if powerfail is ON, but the interrupt system has not been enabled by execution of at least the IONH instruction, the interrupt request generated by the powerfail logic will not be recognized. It is essential for proper operation of powerfail logic that both it and the interrupt control logic be enabled.

PION            1500            Powerfail On

Enables powerfail interrupt. Until this instruction is executed, powerfail does not operate. After this instruction, interrupt system must be enabled to permit powerfail to generate interrupt request (even though it is enabled).

PIOF            1600            Powerfail Off

Disables powerfail interrupt. After this instruction is executed, powerfail logic will not attempt to generate interrupt request. No interrupt request can be generated even if interrupt system is enabled and power fail logic raises "Power Low" flag.

SKPL            1440            Skip on Power Low            PC

If powerfail logic detects that internal voltages have deviated from optimum value, powerfail logic raises "Power Low" flag. Regardless of condition of interrupt system, or whether powerfail interrupt has been enabled, this instruction skips next word if powerfail flag is up. Otherwise, next word is executed.

It is usual for SKPL instruction to be given in the interrupt polling routine, following location 0002 on memory field zero. It is also usually the first instruction in the routine. Following example illustrates polling routine and how powerfail and other interrupts can be efficiently handled.

## SAMPLE POLLING ROUTINE

|           |            |                        |
|-----------|------------|------------------------|
| 0000 0000 | 0          |                        |
| 0001 0000 | INTPR,     | 0 /PR AFTER INTERRUPT  |
| 0002 1440 | SKPL       | /POWER LOW:            |
| 0003 6002 | SKIP       | /                      |
| 0004 6076 | JMP PDOWN  | /YES - STORE REGISTERS |
| 0005 7414 | TOS        | /NO - TELETYPE FLAG?   |
| 0006 6002 | SKIP       | /                      |
| 0007 6050 | JMP TTYOUT | /YES                   |
| 0010 7404 | TIS        | /NO - READER FLAG?     |
| 0011 6002 | SKIP       | /                      |
| .. ..     | JMP READ   | /YES                   |
| .. ..     |            | /NO                    |
| 0030 1007 | IONN       | /INTERRUPT LOW ON      |
| 0031 6330 | JMP@ INTPR | /RETURN TO PROGRAM     |

### 4.8 LITERAL INSTRUCTIONS

Literal instructions save both time and memory space because they permit frequently-used constants and counter-initializations to be stored in the same instruction space as the command which employs the data. The ND812 is equipped with three literals which permit the use of six-bit literal quantities. Literal quantities may be ANDed with, added to, or subtracted from J. These instructions use the Group 2 instruction format described in Chapter 3.

ANDL            21XX            AND Literal            J

Six bits of literal are ANDed with bits 6 to 11 of J register. J register bits 0 to 5 are cleared to zero because literal value is considered to have six zero bits for its bits 0 to 5. Results are in J (replacing its former contents). Memory is unaltered.

ADDL            22XX            Add Literal            J, O

Six bits of literal field of instruction are added to contents of J; sum replaces its previous contents. Literal is considered to be six-bit, positive value with bits 0 to 5 being zero. Overflow is possible. Memory is unaltered.

SUBL            23XX            Subtract Literal            J, O

Six bits of literal are subtracted from J register. J Register bits 0-5 are cleared to zeroes because literal value is considered to have six zero bits for its bits 0-5. Difference replaces previous contents of J. Memory is unaltered and overflow is possible.

#### 4.9 INPUT/OUTPUT

This section treats the standard input/output instruction set with which the ND812 is equipped. Some of these instructions are two-word I/O and others are single-word I/O. The two-word I/O commands are designated TWIO and are followed by the command description. This always implies that the octal value of the first word of that command is 0740. The two-word I/O cycle time is 5  $\mu$ s.

Certain instructions are included which do not appear to be input/output commands (primarily those which handle the INT and JPS registers). They, however, use the single-word format, and are based upon the octal code for single-word I/O commands, 7400. The single-word I/O cycle time is 3  $\mu$ s.

##### 4.9.1 INT AND JPS REGISTER INSTRUCTIONS

There are three instructions in this group. Their function is to enable the preservation and reloading of the contents of the JPS and INT registers. They store the address at which the program counter was deposited upon generation of either a JPS instruction or an interrupt condition. It is essential that their contents be preserved if a powerfail condition should arise, because there would be no way of recovering the status of the system if their contents were lost.

LDREG            7720            Load JPS from J, INT from K

Restores contents of JPS and INT registers previously loaded from memory (following their deposition there in powerfail condition). Loads JPS and INT registers at any time.

LDJK            7721            Load JPS to J, INT to K

Loads contents of JPS register into J and contents of INT register into K. Is usually in powerfail routine to allow saving of JPS and INT register contents.

RJIB            7722            Set JPS and INT status

The two memory field bits for JPS and two memory field bits for INT are OR'ed from their locations in J (J2 and J3 for JPS, J4 and J5 for INT) to the status register.

## 4.9.2 TELETYPE SYSTEM

The ND812 is usually equipped with a teletype interface, to which an ASR33 teletype is normally connected. This device contains a keyboard, printer, paper tape reader, and a paper tape punch; all operate at 10 characters/second. The ASR33 cannot punch independently of printing. It is a serial device; that is, the 11 bits constituting each character are received by the interface and sent out least significant bit first. Eight of the bits are information bits; the other three are timing or synchronizing bits.

When a character has been loaded into the input register of the teletype interface, the interface raises its flag. Similarly, when a character has been shifted out of the print buffer of the interface and print/punched, a separate flag is raised. Both flags will generate an interrupt at the lowest level if it is enabled by an IONN instruction. If the level is not enabled, or after the interrupt, the status of the flags may be tested toward determining what is to be done.

Teletype input and teletype output each has four commands which may be executed to transfer data into and out of the ND812. It is unnecessary that the ND812 operate in the interrupt mode, but if a steady stream of characters is incoming, the programmer must be certain that if he does not desire to operate in the interrupt environment, there is a status and data transfer instruction executed at least every 80 ms; otherwise, there is a possibility that data will be lost. The teletype always operates in the step modes.

TIS            7404            Skip if Keyboard Ready

Skips if character is ready from keyboard/reader. If tape were in reader and reader switch is set to START, or if key were struck, keyboard flag ready would be raised as soon as that character were shifted into the interface buffer. If ND812 is operating in interrupt mode, interrupt would be generated. If not, this instruction could be loop-executed until it skips. In either event, skip occurs when flag is raised.

TIR            7402            Load Keyboard Into J

Loads contents of keyboard buffer into J. Clears keyboard ready flag. Does not cause another character to be read from reader or keyboard.

TIF            7401            Keyboard-Reader Fetch

Reads another character into keyboard buffer. Does not transfer information into J. Clears keyboard ready flag until loading complete. Then sets flag again.

TRF            7403            Keyboard Read-Fetch

Combines functions of TIR and TIF. Transfers keyboard buffer contents into J. Reads more tape from keyboard into keyboard buffer. Clears keyboard ready flag until loading complete.

TOS            7414            Skip if Printer-Punch Ready

Skips if teletype ready to accept character.

TOC            7411            Clear Flag

Clears print-punch ready flag. Is used if it is desired to lower flag without printing another character. Clears output interrupt conditions.

TCP            7413            Clear Flag, Print-Punch  
TOP            7412            Print-Punch

Clears flag. Loads new character into print-punch buffer. New character is printed. When print is complete, interface raises flag again. If lowest level interrupt is enabled by an IONN instruction, a trap to location 1 (field 0) will occur. These two commands are approximately equivalent (resulting in same effect).

#### 4.9.3    HIGH SPEED PAPER TAPE

The ND812 High Speed Paper Tape System is an option consisting of either a 125 or 300 character/second optical reader or a 110 character/second punch. Both input and output through the J register.

Operation of the high speed paper tape is very similar to that of the low speed teletype. The high speed paper tape and high speed paper tape punch have four commands to control them. Both devices may be operated in either the interrupt or programmed modes. The instruction timing is as given for all instructions.

HIS            7424            Skip HS Reader Ready

If character has been read since high speed reader flag was cleared, causes processor to skip next instruction. Raising of HS reader flag generates (if lowest level interrupt is enabled) interrupt request.

HIR            7422            Clear Flag; Read HS buffer

Clears HS reader flag, transfers contents of HS reader buffer to least significant 8-bits of J (bits 4 to 11). Does not cause HS reader to read another character.

HIF            7421            HS Reader Fetch

Causes HS reader to move and read another character from paper tape and clears ready flag. Does not clear buffer.

HRF            7423            HS Reader Read-Fetch

Combines actions of HIR and HIF. Causes transfer of character into J. Clears flag. Causes HS reader to read another character.

#### NOTE

There is a similarity between the commands for the HS reader and the standard teletype read-keyboard commands; therefore, the same programming techniques work.

HOS            7434            Skip if HS Punch Ready

Initiates punch sequence after initial HOL command is given. Skips when HS punch buffer has completed punching of last character. Raising of flag generates interrupt from lowest interrupt enable if enabled.

HOL            7432            Clear Flag; Load Buffer

Clears HS punch flag and loads HS punch buffer from bits 4-11 or J register. Does not alter J.

HOP            7431            Punch on HS Punch

Initiates HS punch cycle. Does not clear flag. J is not altered.

HLP            7433            Load and Punch HS Punch

Combines functions of HOL and HOP commands to cause clearing of flag, reloading of punch buffer and punching of character.

#### 4.9.4    MAGNETIC CASSETTE TAPE SYSTEM

The magnetic cassette tape system records and recovers digital data from the ND812 processor on and from tape cassettes. Data is transferred via the lower 8 bits of the J register at a rate of 500 characters/second. Facilities are included for installing from one to three cassette tape drives, all of which can read and write filemarks, and move forward at high speed or rewind. All of these operations are under program control.

If multiple cassette tape units are installed, each may be individually selected and commanded to perform functions. Only one may be reading or spacing forward at a time, but all may be writing or rewinding.

The cassette tape system is supplied with a full set of operating instructions (including the ability to operate in the interrupt mode). In the interrupt mode, the cassette tape system requests interrupts when enabled with the level output from the Level A interrupt enable

line. The cassette tape system generates trap addresses to the ND812, thus permitting efficient program utilization of the interrupt system.

There are instructions to test for end-of-tape, beginning-of-tape, and filemark. Read and write ready may also be tested, so operation of the magnetic cassette tape system in the polled mode is possible. Testing of the non-read error may also be accomplished.

4.9.4.1 CONTROL FLAGS. Nine hardware flags are generated by the Tape Cassette System which control the programming sequence. For example, a CSWR (skip if write ready) instruction will not cause a skip unless the Ready Flag is set to "1". Table 4-1 lists and describes the states of all nine control flags.

Table 4-1. Tape Cassette Control Flags

| Flag Name            | Signal Name | Flag States  |
|----------------------|-------------|--|
| Tape Error Flag      | ERFG        | ON (set to "1"):<br>Transport selected, a CRDT instruction issued, and an error detected in both track A and B.<br>OFF (set to "0"):<br>Reset by a CCLF instruction or read re-initialized.                  |
| Read Flag            | RDFG        | ON (set to "1"):<br>Selected transport in read mode and a character has been read into the read buffer.<br>OFF (set to "0"):<br>Reset by a CRDT or CCLF instruction, or the detection of an interrecord gap. |
| Write Flag           | WTFG        | ON (set to "1"):<br>Selected transport in write mode and the write buffer is empty.<br>OFF (set to "0"):<br>Reset by a CWRT instruction.   |
| Write Interrupt Flag | WTIFG       | ON (set to "1"):<br>When Write Flag makes transition from OFF to ON.<br>OFF (set to "0"):<br>Reset by a CCLF or CWRT instruction.  |



Table 4-1. Tape Cassette Control Flags (Cont'd.)

| Flag Name  | Signal Name | Flag States   |
|--|-------------|---|
| Ready Flag   | RDY         | <p>ON (set to "1"):<br/>Transport selected, cassette mounted, and transport motion stopped.</p> <p>OFF (set to "0"):<br/>Transport not selected, or cassette not mounted, or transport in motion.</p> |
| <p>NOTE</p> <p>This flag is a test flag and cannot be reset.</p> |             |   |
| Ready Interrupt Flag   | RDYFG       | <p>ON (set to "1"):<br/>When Ready Flag makes a transition from OFF to ON.</p> <p>OFF (set to "0"):<br/>Reset by a CCLF instruction or processor start clear.</p>                                     |
| BOT Flag   | BOT         | <p>ON (set to "1"):<br/>When Ready Flag is on and transport rewound to beginning of tape.</p> <p>OFF (set to "0"):<br/>When Ready Flag is off and transport not at beginning of tape.</p>             |
| <p>NOTE</p> <p>This flag is a test flag and cannot be reset.</p> |             |   |
| EOT Flag   | EOT         | <p>ON (set to "1"):<br/>When Ready Flag is on and transport wound to end of tape.</p> <p>OFF (set to "0"):<br/>Reset by a CSLCT or CHSR instruction, or processor start clear.</p>                    |

Table 4-1. Tape Cassette Control Flags (Cont'd.)

| Flag Name     | Signal Name | Flag States   |
|---------------|-------------|---|
| Filemark Flag | FMFG        | ON (set to "1"):<br>When transport is selected and a filemark is read during forward or reverse tape motion.<br>OFF (set to "0"):<br>Reset by a CCLF instruction, when BOT is on, or processor start clear. |

4.9.4.2 PROGRAMMING GUIDELINE. Three trap locations in Memory Field 00 can be used by the ND812 Central Processor if the user desires to program the Tape Cassette System on an interrupt basis. To use the trap locations, the processor's level A interrupt circuitry (IONB) must be enabled and one of the following conditions must exist.

- a. Read Flag set to "1". Causes the processor to trap to location 0041<sub>g</sub>.
- b. Write Interrupt Flag set to "1". Causes the processor to trap to location 0051<sub>g</sub>.
- c. Filemark Flag or Ready Interrupt Flag set to "1". Causes the processor to trap to location 0061<sub>g</sub>.

NOTE

If the low level interrupt circuitry is accidentally enabled and programming provisions were not made for interrupt servicing, the program will begin executing at one of the three trap locations.

Records written on tape can vary in length from 1 character (8 bits) to 120,000 characters (limited by tape length). Termination of a record will be accomplished when the processor fails to respond to a Write Flag within 400  $\mu$ s. If the processor does not load additional data within this time, the Write Flag will be reset and the write process will terminate.

When writing data or filemarks, do not issue another transport select until the currently selected transport becomes ready (Ready Flag set to "1").

A read operation will be terminated on an interrecord gap (IRG).

Start clear is generated when the ND812 Central Processor is turned on. Start clear rewinds

all cassette transports to BOT, and clears all control flags.

When a cassette transport is running, a cassette select I/O instruction (760X) will be ignored by the transport. To ascertain that a cassette is properly selected, the following routine is suggested.

```
CRDY, 0
      760X
      TWIO
      CSTR
      JMP.-3
      JMP @ CRDY
```

4.9.4.3 TYPICAL PROGRAM SEQUENCE. Four flow charts are included which depict typical cassette routine programming sequence. These figures are intended as a guide not as a standard convention. Figure 4-1 provides a typical ready flow chart; figure 4-2 provides a typical cassette write data flow chart; figure 4-3 provides a typical write file-mark flow chart; and figure 4-4 provides a typical cassette filemark search flow chart.

4.4.4.4 SPECIFIC INSTRUCTION SET. The Tape Cassette System is a software controlled device that responds to a selected number of I/O instructions. The following discussion lists and describes these I/O instructions in four groups; Transport Select Instructions, Transport Status Instructions, Transport Write Instructions, and Transport Read Instructions. Transport Select Instructions are single word I/O instructions; all others are two word instructions. Refer to Table 4-1 for a description of control flags.

4.9.4.4.1 Transport Select Instructions. Magnetic cassette tape units do not obey commands unless they are first selected by a cassette tape unit select command. These commands are all single-word instructions. Whenever a cassette tape unit is to be selected, it must be stopped. Selection can, in fact, be accomplished only when all units have completed any prior commands and stopped. Once a unit has been selected and commanded to perform a function, its selection or function may not be changed until it has again come to a stop.

CSLCT1            7601            Place Cassette 1 On-Line

Selects cassette tape unit one and de-selects all other cassette tape units previously selected. No commands are accepted by cassette tape unit one until it is selected.

CSLCT2            7602            Place Cassette 2 On-Line

Selects cassette tape unit two and de-selects all other cassette tape units previously selected. No commands are accepted by cassette tape unit two until it is selected.

CSLCT3            7604            Place Cassette 3 On-Line

Selects cassette tape unit three and de-selects all other cassette tape units previously selected. No commands are accepted by cassette tape unit three until it is selected.

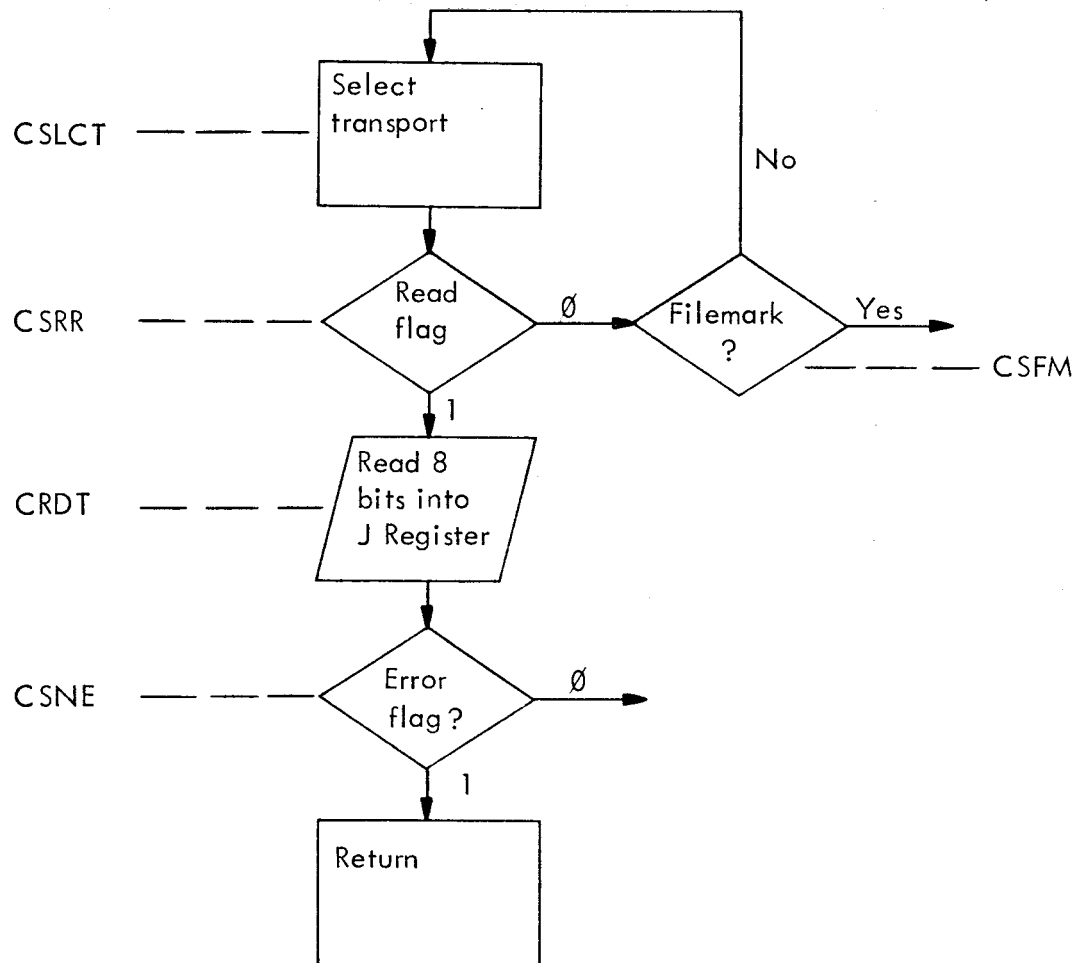


Figure 4-1. Typical Cassette Read Flow Chart

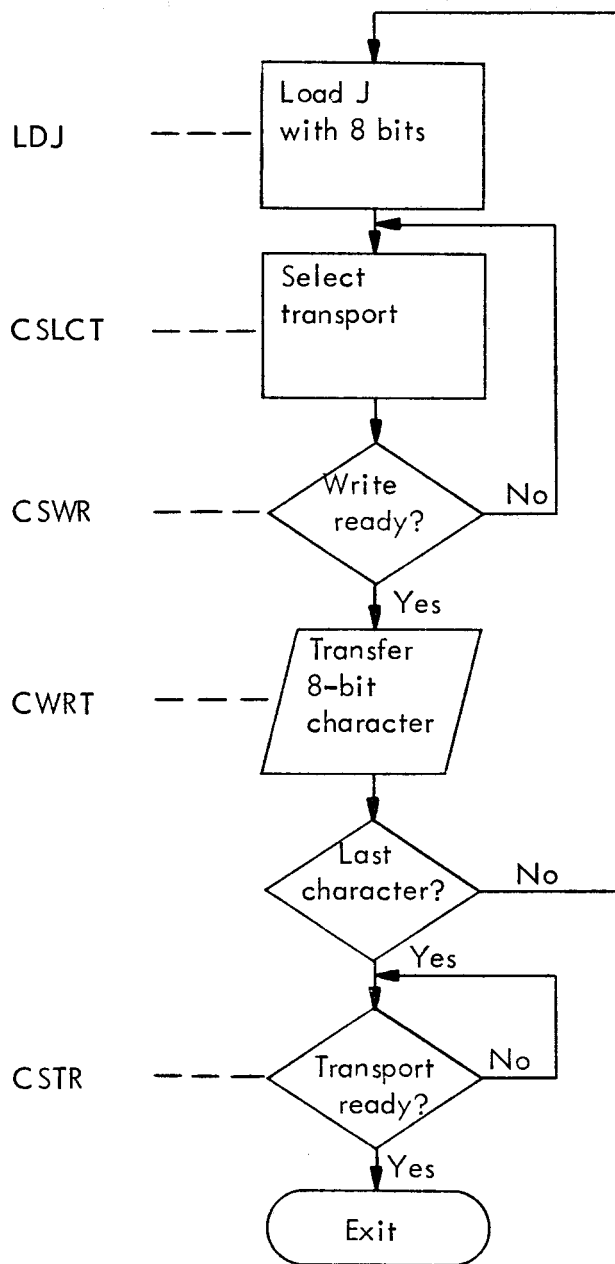


Figure 4-2. Typical Cassette Write Data Flow Chart

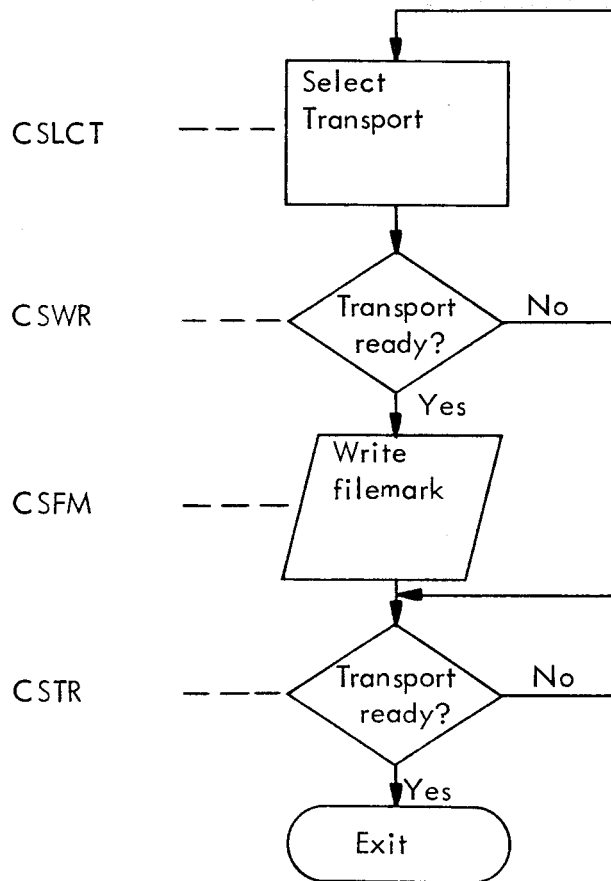


Figure 4-3. Typical Cassette Write Filemark Flow Chart

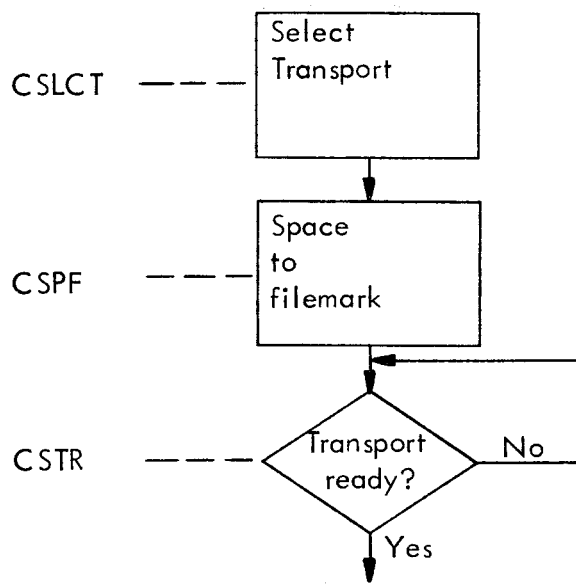


Figure 4-4. Typical Cassette Filemark Search Flow Chart

4.9.4.4.2 Transport Status Instructions. After a cassette tape unit has been selected, its ready status should be checked to determine whether a cassette has been mounted or all previously selected operations have been completed. A selected cassette tape unit can test ready only if it has a tape mounted and has completed a previous operation, or is at BOT.

CSTR                    0740-0124    Selected TWIO Skip if Transport is Ready (Ready Flag set to "1")

If selected unit is finished with any command, and if cassette tape is mounted, next command is skipped. If cassette tape unit did not have tape mounted, command does not skip. If cassette tape unit was moving its tape on read or write, or had moved forward to EOT, skip would not result. If cassette tape unit is high-speed-reversed, resulting BOT condition enables skip function.

CSFM                    0740-0104    Skip on Filemark (Filemark Flag set to "1")

If a filemark has been sensed by system hardware, interrupt request is generated if lowest interrupt enable were allowed. Cassette tape system then traps to location octal 61, MFC. If interrupt system were off, interrupt would not occur but detection of filemark would cause this instruction to skip. Filemark may be detected by system at any time. Skip executes properly if it is given before tape unit passes end of record gap.

CSET                    0740-0110    Skip if Transport at End of Tape (EOT Flag set to "1")

Skips if On-Line magnetic cassette tape unit is in EOT condition. EOT condition is logically derived from sensing tape-end signal while in forward motion state. If cassette tape is dismounted after EOT is reached, interface recognizes EOT signal as BOT signal when tape is remounted. It is therefore impossible for unit on which it was mounted to obey high speed backward to BOT (rewind) command.

CSNE                    0740-0122    Skip if NO-ERROR (Tape Error Flag set to "0")

If no track switching (as result of read error) has occurred since last time this instruction was given, skip is executed. This instruction must be given within 1 ms following successful skip on read ready flag. No-skip does not necessarily indicate that erroneous character was transferred, because system hardware selects data from other tape track automatically.

CSBT                    0740-0130    Skip if Transport at Beginning of Tape (BOT Flag set to "1")

Skips in BOT condition. Is arrived at by sensing end of high-speed reverse command or mounting cassette tape. Whenever unit is initially turned on, it attempts to rewind (HS reverse) to BOT. If cassette tape is mounted, it will do so.

The flags should be cleared to initialize the unit after a powerfail or turn-on. However, it is not desirable that this command, if given at any other time, clear the write flag, because this could prevent termination of an otherwise normal write operation.



CCLF                    0740-0141    Clear All Cassette Control Flags

Resets the Read Flag, Write Interrupt Flag, Ready Interrupt Flag, Filemark Flag, and Tape Error Flag set to "0".

4.9.4.4.3 Transport Write Instructions. A write operation may be executed if a cassette tape unit is in the ready state. Write status is attained by executing a skip if write ready. This instruction actually causes the cassette tape unit to begin moving and enter the write condition. If no subsequent write transfer instruction is given within 400  $\mu$ s, the system assumes that the last character has been transferred, writes the initial interrecord gap, resets the write flag, and stops. This is the normal method for terminating a write operation. Record length written is limited only by the length of the tape (approximately 120,000 characters).

CWFM                    0740-0151    Write Filemark

Writes filemark code on cassette on-line unit. Unit must be ready, not executing any other command, and not be at either BOT or EOT.

CSWR                    0740-0152    Skip if Write Ready (Write Flag set to "1")

When initially given, causes motion of cassette tape; also places unit in write condition. In interrupt mode, raising of flag causes interrupt if lowest level interrupt enable is armed. This command does not clear write flag. Skipping occurs only when unit has transferred previous character onto tape and is ready to write next character, or when initial interrecord gap writing has been completed.

CWRT                    0740-0154    Write Transfers 8 Bits to Buffer

Must be given within 400  $\mu$ s of raising of write flag. Transfers data from bits 4-11 of J into buffer of magnetic cassette tape unit controller. Also clears flag and is only command which can clear write flag.

4.9.4.4.4 Transport Read Instructions. A read operation may be executed, if a cassette tape unit is in the ready state. Read status is achieved by executing the instruction to skip if read ready. The actual transfer of data from the cassette tape unit is accomplished by executing a transfer eight bits to J. If the read status is attained by executing a skip if read ready, but no subsequent transfer data to J is given, the selected cassette tape unit will continue moving the tape forward; continue raising its flag to request data transfer or no data; or will transfer. If the Level A interrupt enable is disarmed, no further interrupts will be recognized. When the cassette tape unit finds the interrecord gap, however, it comes to a halt and is ready to read the next block of data.

CSRR                    0740-0142    TWIO Skip if Read Ready (Read Flag set to "1")

If selected cassette tape unit is stopped in interrecord gap or is at BOT, execution of this

command starts tape moving in forward direction; when first character of data is read from tape, read ready flag is raised and skip occurs.

If Level A interrupt line is enabled during the time the read ready flag is raised, interrupt occurs. If interrupt enable were not allowed, skip if read ready causes skip of next instruction. In either event, execution of skip if read ready instruction results in clearing of read ready flag.

CRDT                    0740-0144    TWIO Read Transfer 8-bits of J

If read ready flag is raised (signifying that cassette tape unit has loaded 8-bits into cassette tape unit buffer), execution of this command causes 8 bits to be loaded into J register (bit positions 4 to 11). Buffer is then cleared, and the read Ready Flag is set to "0". Tape Error Flag is set to "1" if errors were detected.

CHSF                    0740-0101    High Speed Forward To EOT

Causes the transport to run forward (left to right) at high speed if the Ready Flag is set to "1". Sets the Filemark Flag to "1" on detection of a filemark. Forward motion is terminated when EOT Flag is set to "1" or when CSPF is issued.

CSPF                    0740-0102    Space Forward to Filemark

Causes the transport to run forward at normal speed until the Filemark Flag is set to "1". Can be used with CHSF or CHSR for a high speed filemark search or during a read operation to position the tape at the next filemark. After accepting command, unit stops in interrecord gap following filemark.

CHSR                    0740-0121    High-Speed Reverse to BOT

Causes the transport to run in reverse at high speed if the Ready Flag is set to "1". Sets the Filemark Flag to "1" on detection of a filemark. Reverse motion is terminated when BOT flag is set to "1" or if a CSPF is issued. It is good practice to give more than one on-line unit this command (provided that, of course, more than one unit is off BOT position). BOT is logically derived from unit being turned on initially, or EOT condition following a high speed reverse command. High-speed continues at approximately twice normal motion rate until BOT is sensed or space forward to filemark command is given.

4.9.4.5 MISCELLANEOUS INSTRUCTIONS. Two instructions perform important functions which do not really fit the other functional classifications. They are STOP and unconditional skip. Because it is often important to bring the ND812 to an order by program control, the STOP command is important; the unconditional skip permits skips of two-word and one-word commands.

STOP                    0000            Stop Execution

Causes ND812 to suspend operation. Depression of continue switch causes ND812 to resume

operation (if desired). Contents of low-order six bits is inconsequential. These can be employed to contain numerical value identifying which STOP has been executed in a program containing several STOPS.

SKIP                    1442                    Unconditional Skip

This instruction is a skip unconditionally command which skips the instruction following it.

IDLE                    1400                    One cycle delay

Delays execution of next instruction for one memory cycle.

TWIO                    0740                    Two Word I/O

First word of a two-word input/output instruction.

|    |      |         |
|----|------|---------|
| F0 | XXX4 | Field 0 |
| F1 | XXX5 | Field 1 |
| F2 | XXX6 | Field 2 |
| F3 | XXX7 | Field 3 |

Specifies memory field in which two-word memory reference instructions will be executed.

## SECTION V PROGRAMMING FUNDAMENTALS

### 5.1 GENERAL

Understanding the instruction set is the first step in learning to program the ND812 computer system. The next is learning the use of the instruction set to obtain correct results efficiently. This is best done by studying the following programming procedures and techniques.

### 5.2 PROGRAMMING PROCEDURES

To successfully solve a problem with a computer, the programmer proceeds through the phases of writing a program. These can be broken down into six basic steps.

#### 5.2.1 DEFINITION

The definition of the problem is not always obvious. A great amount of time and energy can be wasted if the problem is defined inadequately; therefore, the programmer must form a clear and comprehensive statement of the problem.

#### 5.2.2 ANALYSIS

Determining the method to be followed is the second important step. There are, conceivably, many methods of solving the problem, but one must be selected. After a method is selected, other analysis consists of laying out the problem in a form susceptible to arithmetical and/or logical computation, determining what logical decisions must be made, and in what format the data must be.

#### 5.2.3 FLOW DIAGRAM

The programmer must design and analyze the solution by identifying the necessary steps to solve the problem and arranging them in a logical order. Flowcharting is a graphical means of representing the logical steps of the solution by the use of special symbols which denote the various operations and the sequence in which they occur. The flowcharting technique provides an overview of the logical solution flow.

#### 5.2.4 CODING

Having designed the problem solution, the programmer begins to code the solution in the programming language. This step is commonly called programming but is actually coding and is only one part of the programming process. Coding is the process of converting the operations listed in the flowchart into the language the computer will use (either instruction language or compiler statements). When the program has been coded and the program instructions have been stored in the computer memory, the problem can be solved.

#### 5.2.5 DEBUGGING

The program checkout step requires the programmer to retrace the flow of the instruction methodically to find any program errors that may exist.

If needed instructions are omitted or coding is performed incorrectly, the results will be in error. These flaws ("bugs) must be found and corrected. Debugging is the process of locating these errors in the program and correcting them. Various techniques are available for this purpose. A program may be written to include some aids or a separate debugging program may be run to test the operation of a malfunctioning program.

#### 5.2.6 DOCUMENTATION

Merely writing a program which runs properly is not sufficient. Changes may be necessary or it may be desirable to use the program or subroutines from it within another program. To accomplish any of these tasks readily, it is necessary to include documentation which includes a description of the program, flowcharts, and data format layouts of inputs and outputs.

### 5.3 FLOW CHARTING

When a complex problem is to be solved by a computer, the program involves many steps; writing it often becomes tedious and confusing. A written method of solving a problem is extremely difficult to follow; coding of computer instructions from such a document would be equally difficult.

The flowcharting technique serves a number of very important functions. It is a map of how the programmer intends to solve a problem. The chart illustrates the logical steps required, the decisions to be reached, and the paths to be followed as a result of the decisions. If it is properly annotated, it calls the programmer's attention to memory allocation, input/output requirements, data accuracy considerations, and register usage. The flow diagram is of vital importance in making such program changes as may be required and debugging a malfunctioning program.

Flowcharts may be constructed at various levels of complexity. A high-level flow chart is a very general overview, while a low-level flow chart may reach a correspondence between symbols and instructions. Painstaking flowcharting has its own reward in the encoding and debugging stages; the returns increase in direct proportion to the complexity of the program.

The flowchart is basically a collection of boxes and lines. The boxes indicate what is to be done and the lines indicate the sequence. The boxes are of various shapes which represent actions performed in the program. Appendix B is a guide to the flowchart symbols and procedures used.

The following flowcharts are examples of two types of flowcharting. The first is straight-line programming, and the second is decision-making and branching. The examples illustrate methods of attacking the problem via a computer program as well as flowcharting techniques. In Figure 5-1, two numbers are added together and the result is stored in a third location ( $X + Y \rightarrow Z$ ).

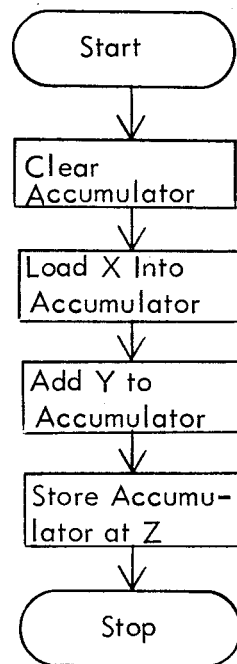


Figure 5-1. A Straight-Line Flowchart

Figure 5-2 illustrates how the largest and smallest of three unequal numbers (A, B, C) are determined. The program must branch upon determining which numbers are larger or smaller.

#### 5.4 PROGRAMMING CONCEPTS

There are many concepts and techniques involved in programming which constitute the basis of writing and developing a good program. Full understanding of when and where these concepts can or should be used comes only from experience gained in programming. Some of these basic concepts are discussed in the following paragraphs.

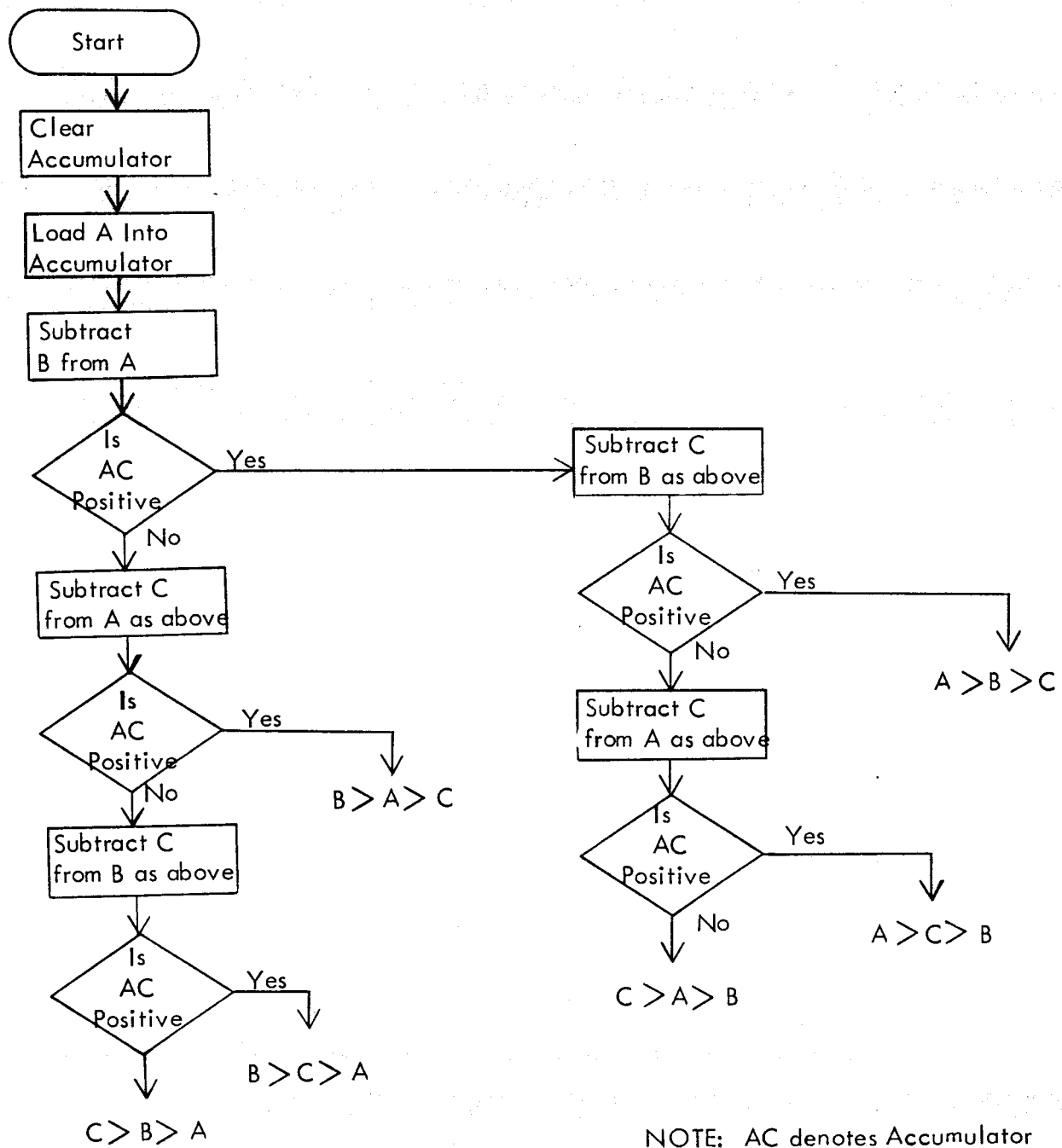


Figure 5-2. A Branched Flowchart

### 5.4.1 LOOPING

A loop is a group of instructions designed to perform an iterative function. Therefore, the loop must initiate, compute, modify, and terminate. Looping of a program is one of the most powerful tools the programmer has. It enables him to perform similar operations many times using the same instructions; thus memory locations are saved because he is not required to store identical instructions several times. Looping also renders a program more flexible, because it is relatively easy to change the number of loops required for various conditions by resetting a counter. It should be remembered that looping is little more than a jump to an earlier part of the program; however, the jump is usually predicated upon changing program conditions. Figure 5-3 shows a typical looping situation.

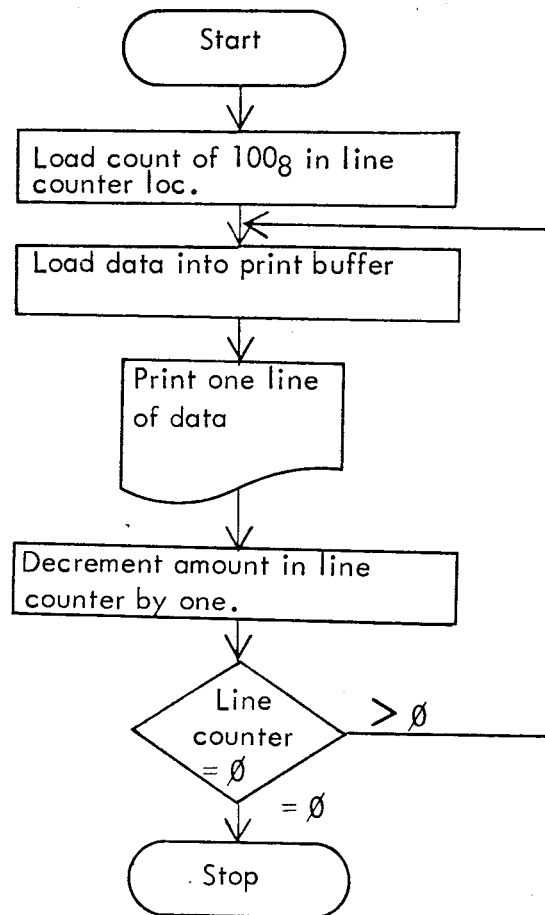


Figure 5-3. Typical Looping Situation



### 5.4.2 ADDRESS MODIFICATION

Address modification is a very powerful tool of the programmer. Address modification refers to the inclusion of instructions in a program to modify the operand portion of a memory reference instruction. It is a particularly useful technique in working with large blocks of stored data. However, because addresses are modified as the program runs, the program cannot be rerun without being reloaded. Moreover, in debugging, the addresses will not be as shown in the assembler listing. A programmer should include extra instructions in the program to reset these values before they are encountered. This procedure is often referred to as "housekeeping". Figure 5-4 shows a typical address modification situation.

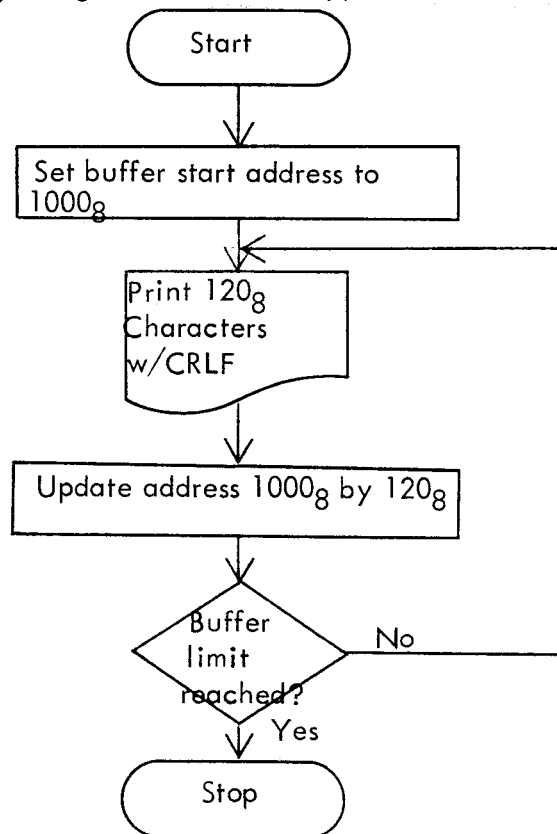


Figure 5-4. Typical Address Modification Situation

### 5.4.3 INDEXING

This term refers to a number of operations. The operation can be counting the number of times an operation is performed; the process of modifying the operand portion of an instruction word prior to its execution (without modifying it as contained in memory); or tagging data in a file in memory.

Certain addressable registers are built into the computer to facilitate indexing. These index registers and their associated circuitry are able to increment and/or decrement themselves as the result of a comparison of their contents and some other value.

The index registers are counters which are generally used to change the numerical value of the address portion of a computer instruction to obtain an effective address. This action is accomplished by modifying the word address register with the absolute value of a number stored in the index register. The index process does not alter the instruction word nor the number contained in the index register; thus, the use of the word as many times as necessary in its indexed or non-indexed form is made possible.

#### 5.4.4 AUTO-INDEXING

The method of indexing used in the ND812 is called auto-indexing. Two words are used as auto-index registers in each memory stack of the ND812.

These locations have the property that if they are addressed directly, their behavior is normal; that is, they simply act as the operand location and their contents are used normally. However, if they are indirectly addressed by a single-word instruction, they first increment their contents by one, after which the resultant value is restored as their contents. Finally, the ND812 uses the modified contents of the auto-index location to access the operand **desired**.

Single-word format instructions may address these two locations relatively, indirectly and directly, but the operand must always be in the memory stack in which that instruction **resides**.

When a single-word format instruction directly accesses either of the two locations, it specifies this with a special value in the displacement field ("00g"). The forward/backward bit specifies which of the two locations is to be used, and the direct or indirect bit specifies whether the contents of the auto-index location are the operand or the address of the operand. Two-word format memory reference instructions use the auto-index locations as either an operand or the address to the operand in an indirect address. When used indirectly, the auto-index locations do not automatically increment.

#### 5.4.5 SUBROUTINES

Subroutines are important means of developing conciseness in a program. Obviously, as a program grows larger, certain functions are repeated. If the instructions required to perform these functions are grouped, they may be referenced by relatively few instructions in the main program -- thus obviating the necessity of writing the instructions in the main program each time the function is being performed. A subroutine may contain other subroutines and also be a part of a larger subroutine.

Included in the instruction repertoire is the instruction, "Jump-to-Subroutine". This instruction makes linking to a subroutine from the main program possible. The "Jump-to-Subroutine" instruction automatically stores the address of the instruction after the "Jump-to-Subroutine" in the location to which the program is instructed to jump; thereby, a return is enabled. The programmer need only terminate the subroutine with an indirect jump to

the first location of the subroutine (JMPC) to return to the next instruction following the "Jump-to Subroutine" in the main program.

#### 5.4.6 INPUT/OUTPUT PROGRAMMING

Input/output programming is the process of communicating with the computer. It involves not only the transfer of data, but commands which control the operation of the peripheral equipment. The computer I/O section is independent of the rest of the computer once it is initiated; this permits I/O operations and computations to occur simultaneously. For instance, it is often desirable to alert the main program when a block buffer is complete; the I/O instruction can perform such a function, which is generally classified as an I/O interrupt. That is, when the data block has been transferred in or out, the I/O section generates an interrupt command to notify the processor that the I/O transfer is complete. This does not specify that it was a good transfer of data; that is for the programmer to **determine**.

### 5.5 PROGRAM PREPARATION

Now that programming procedures, flowcharting, and various programming concepts have been defined, an example problem can be presented which will demonstrate the mechanics involved in solving a problem using the ND812 computer. The example problem illustrates programming concepts such as branching, looping and input/output subroutines. The example problem also illustrates the mechanics involved in generating and modifying a source program via Teletype keyboard using the Text Editor, and then assembling the source program into an object (binary) program via the Assembler. The object program is then loaded into the ND812 Computer and executed to solve the example problem.

#### 5.5.1 DEFINITION OF EXAMPLE PROBLEM

The example problem is as follows. Input two unequal numbers defined as "A" and "B", compare the two numbers and determine which is larger, and output a literal statement "A > B", or "B > A" as applicable.

#### 5.5.2 ANALYSIS AND FLOWCHARTING OF EXAMPLE PROBLEM

Since the example problem is intended to demonstrate the mechanics involved in solving a problem using the ND812 Computer, the program will be as brief as possible. The following ground rules apply to programming the example problem.

- a. The inputs will be limited to two numbers defined as "A" and "B" for brevity. The inputs will be unequal numbers in order to eliminate a check for A equal to B. The inputs will be printed (echoed) at the Teletype for verification. The inputs will be converted from ASCII Code to a constant and stored in memory.
- b. The inputs will be compared to determine which is larger. The result will point to one of two addresses for a literal statement "A > B" or "B > A".

- c. The literal statement "A > B" or "B > A" will be printed at the Teletype.
- d. The input and output controls will be programmed as subroutines since they are used more than once.

Now that the example problem has been defined and analyzed, a flowchart can be constructed. Figure 5-5 illustrates the flowcharting of the example problem.

5.5.2.1 FLOWCHART DESCRIPTION OF EXAMPLE PROBLEM. (Refer to Figure 5-5.) The program is given a starting address. The Teletype flag is cleared, and a jump to the input subroutine is initiated to fetch an input for "A". The input is fetched, echoed at the Teletype, and converted from ASCII to a constant. A jump is initiated which allows return to the main program. The constant for input "A" is stored in memory. A jump to the input subroutine is initiated to fetch an input for "B". The input is fetched, echoed at the Teletype, and converted from ASCII to a constant. A jump is initiated which allows return to the main program. The constant for input "B" is stored in memory. The value for "A" is loaded into the accumulator. The memory location containing value "B" is subtracted from the accumulator. The resultant is tested for a positive value. If resultant is positive, the accumulator is loaded with the address of the literal statement "A > B". If resultant is not positive, the accumulator is loaded with the address of the literal statement "B > A".

A jump to output subroutine is initiated. The address of "A > B" literal or "B > A" literal is stored at memory location which is used in the output routine.

#### NOTE

The output subroutine consists of a loop which outputs ASCII characters one at a time. A constant defined as loop counter is stored in memory, and is set equal to the number of loops required to output a given set of stored ASCII characters. During each loop, the address (which points to the address of next ASCII character) is incremented by one and the loop counter is decremented by one. When the loop counter is zero, indicating all ASCII characters have been printed at the Teletype, a jump is made back to the main program.

After the address of the first ASCII character has been stored in memory, the loop counter constant is loaded into the accumulator from memory. The loop counter is then stored in a difference memory location to allow a decrement of one during each output loop. An instruction is executed which loads the first ASCII character into the accumulator. The first character is printed on the Teletype, and the Teletype flag is cleared when done. The literal address is incremented by one, the loop counter is decremented by one and the loop counter is tested for a zero. If the loop counter is not zero, the next ASCII character is loaded into the accumulator. The second character is then printed on the Teletype, and the Teletype flag is cleared when done. The literal address is incremented by one, and the

loop counter is again tested for a zero. If the loop counter is not zero, the cycle repeats until the loop counter is zero indicating that all ASCII characters have been printed at the Teletype. A jump is then executed for return to the main program. Upon return to the main program, a stop is initiated. When the computer front panel CONT switch is depressed, a jump is executed to return to the starting address. The jump instruction eliminates re-loading the program into the computer.

### 5.5.3 CODING EXAMPLE PROBLEM IN ASSEMBLY LANGUAGE

Now that solution of the example problem is defined, and flowcharted (Figure 5-5), the problem is ready to be coded. This step is commonly referred to as programming, but is actually coding and is only one phase of the programming process. The problem is coded in Assembly Language utilizing the Assembler mnemonics presented in Section IV. Refer to Figure 5-5, and Section IV while coding the example problem. The coding for the example problem is given in Table 5-1.

The example problem (Table 5-1) is coded in the source program format acceptable by the BASC-12 Assembler. The statement format has four fields; a LABEL field, an INSTRUCTION field, an OPERAND field, and a COMMENT field. Refer to the applicable BASC-12 General Assembler Software Instruction Manual, Section II, for a detailed definition of terms, symbols, and terminators (such as a comma, slash, or asterisk) used in coding the example problem.

**5.5.3.1 LOCATION ASSIGNMENT.** The programmer assigns an absolute location to the first instruction which serves as the starting address. The Assembler then assigns successive locations in order when the program is assembled. In programming the ND812 Computer, the initial location is preceded by an asterisk (\*). When the program is assembled via the Assembler, the Assembler maintains a "current location counter" by which it assigns successive locations to instructions. The asterisk causes the current location counter to be initially set to the value followed by the asterisk. The starting address is usually 0200 denoted as \*200 in the coded program (Table 5-1).

**5.5.3.2 SYMBOLIC ADDRESSES.** When coding the program initially, the programmer does not know which locations he will use to store constants or instructions. Therefore, when coding a Memory Reference Instruction, the programmer assigns symbolic address tags which were predefined or will be defined later (a symbolic name followed by a comma is a symbolic address). The Assembler maintains a symbol table in which it records the octal values of all symbolic addresses. Refer to Table 5-1 and note the symbolic address name tags following each Memory Reference Instruction.

Table 5-1. Example Problem, Coded

| LABEL  | INSTR | OPERAND | COMMENTS                  |
|--|-------|---------|---------------------------|
| /Input and store values for A & B            |       |         |                           |
|  | *200  |         |                           |
| Start,                                       | TIF   |         | /Clear TTY Flag           |
|  | JPS   | Input   | /Get Value for A          |
|  | STJ   | A       |                           |
|  | JPS   | Input   | /Get Value for B          |
|  | STJ   | B       |                           |
| /  |       |         |                           |
| /Determine which of the two values is larger |       |         |                           |
|  | LDJ   | A       |                           |
|  | SBJ   | B       | /Subtract B from A        |
|  | SIP   | J       | /Test for A positive      |
|  | JMP   | BRAN    | /No! B > A                |
|  | LDJ   | ABCST   | /Yes! A > B               |
|  | SKIP  |         | /Skip Next Instruction    |
| BRAN,  | LDJ   | BACST   |                           |
| /  |       |         |                           |
| /Set up and output expression                |       |         |                           |
| /  |       |         |                           |
|  | JPS   | OUT     |                           |
|  | STOP  |         |                           |
|  | JMP   | START   |                           |
| /  |       |         |                           |
| /Working or data storage area                |       |         |                           |
| /  |       |         |                           |
| A,   | Ø     |         | /Constant A               |
| B,   | Ø     |         | /Constant B               |
| ABCST,                                       | AB    |         | /Address of A > B Literal |
| BACST,                                       | BA    |         | /Address of B > A Literal |
| C26Ø,  | 26Ø   |         | /ASCII Zone Constant      |
| /  |       |         |                           |
| /Input routine + ASCII zone strip            |       |         |                           |
| /  |       |         |                           |
| Input,                                       | Ø     |         | /Entry Point              |
|  | TIS   |         |                           |
|  | JMP   | .-1     |                           |
|  | TRF   |         |                           |
|  | TCP   |         | /Echo Input at Teletype   |
|  | TOS   |         |                           |
|  | JMP   | .-1     |                           |
|  | SBJ   | C26Ø    |                           |
|  | JMP@  | INPUT   |                           |
| /  |       |         |                           |

Table 5-1. Example Problem, Coded (Cont'd.)

| LABEL                                     | INSTR | OPERAND | COMMENTS                          |
|---|-------|---------|-----------------------------------|
| /Output Routine - Output ASCII Expression |       |         |                                   |
| /   |       |         |                                   |
| Out,                                      | Ø     |         | /Entry Point                      |
|   | STJ   | LOOP+1  |                                   |
|   | LDJ   | C5      | /Set Number of Character Constant |
|   | STJ   | CTR     |                                   |
| /   |       |         |                                   |
| /Output Data Loop                         |       |         |                                   |
| /   |       |         |                                   |
| Loop,                                     | TWLDJ |         |                                   |
|   | Ø     |         |                                   |
|   | TCP   |         |                                   |
|   | TOS   |         |                                   |
|   | JMP   | .-1     |                                   |
|   | ISZ   | LOOP+1  |                                   |
|   | DSZ   | CTR     | /Test For All Characters Out      |
|   | JMP   | LOOP    | /No                               |
|   | JMP@  | OUT     | /Return                           |
| C5,                                       | 5     |         |                                   |
| CTR,                                      | Ø     |         |                                   |
| /   |       |         |                                   |
| /Output Messages                          |       |         |                                   |
| /   |       |         |                                   |
| AB,                                       | 215   |         |                                   |
|   | 212   |         |                                   |
|   | 3Ø1   | /A      |                                   |
|   | 276   | />      |                                   |
|   | 3Ø2   | /B      |                                   |
| BA,                                       | 215   |         |                                   |
|   | 212   |         |                                   |
|   | 3Ø2   | /B      |                                   |
|   | 276   | />      |                                   |
|   | 3Ø1   | /A      |                                   |
| \$  |       |         | /End Character                    |

- NOTES:
1. The dollar sign is the terminal character for the assembler.
  2. The comma after a symbol (e.g., START,) indicates to the assembler that the symbol is a symbolic address.

5.5.3.3 DESCRIPTION OF CODING FOR EXAMPLE PROBLEM. The coding for the example problem (Table 5-1) is divided into eight groups for ease of understanding. The groups are headed by a comment line preceded by a slash. The comment lines have no significance in solution of the problem by the computer, and are provided only as an aid in understanding the coding. Comment lines are always preceded by a slash. The eight groups are as follows (refer to Table 5-1).

1. /Input and Store Values for A & B
2. /Determine Which of Two Values is Larger
3. /Set Up and Output Expression
4. /Working or Data Storage Area
5. /Input Routine + ASCII Zone Strip
6. /Output Routine - Output ASCII Expression
7. /Output Data Loop (Part of output routine)
8. /Output Messages

The following discussion of the coding will be presented under the above headings individually.

5.5.3.3.1 Input And Store Values For A & B. The starting address for the example problem is 0200 signified by \*200, which sets the program counter to 0200. The START, in the label field of the second line of coding provides a tag for return to the beginning of the program. Next, the Teletype flag is cleared to allow an input for "A" to be entered into the Teletype buffer via keyboard. The JPS instruction initiates a jump to the input subroutine +ASCII Zone strip defined by INPUT tag. The input for "A" is fetched from the Teletype buffer and stored in the accumulator (J register), echoed at the Teletype printer, and the number 260 (stored in memory) is subtracted from the J register. Thus, the ASCII character input from the Teletype is now converted to a decimal constant which resides in the J register. Next, an unconditional jump (JMP@) is initiated which allows return back to the main program via INPUT tag (which contains return address).

The J register which contains the decimal constant for input "A" is stored in memory. The second JPS instruction initiates a jump to the input subroutine +ASCII Zone strip. The input for "B" is fetched from the Teletype, echoed, and converted to a decimal constant which resides in the J register. An unconditional jump is again initiated which allows return to the main program via INPUT tag. Next, the J register which contains the decimal constant for input "B", is stored in memory.

5.5.3.3.2 Determine Which Of Two Values Is Larger. The decimal constant for "A" is



loaded into the J register from memory. The memory location containing decimal constant for "B" is subtracted from the J register. The J register is tested for a positive value, if positive, the next instruction is skipped. The J register is then loaded with "ABCST", which is the address of the ASCII Code for a carriage return in the literal output statement. The skip instruction allows an unconditional skip of the next instruction. The next instruction is a JPS instruction which initiates a jump to the output subroutine (paragraph 5.5.3.3.6)

If the J register is not positive, indicating  $B > A$ , an unconditional jump is initiated to "BRAN", a symbolic address tag for an LDJ instruction. The LDJ instruction causes the J register to be loaded with "BACST", which is the address of the ASCII Code for a carriage return in the literal output statement. The next instruction is a JPS instruction which initiates a jump to the output subroutine (paragraph 5.5.3.3.6).

5.5.3.3.3 Set Up And Output Expression. This area consists of a JPS, STOP, and JMP instruction. The JPS instruction allows a jump to the output subroutine, and OUT provides a tag to the saved address for return to the STOP instruction. The STOP instruction stops the computer signifying the end of this computation. The JMP instruction allows return to START (via symbolic address tag "START") when the computer front panel CONT switch is depressed. This instruction eliminates reloading the program for execution of successive inputs for "A" and "B".

5.5.3.3.4 Working Or Data Storage Area. This area provides storage for symbolic address tags A, B, ABCST, BACST, and C260. Locations "A" and "B" are initially loaded with the value zero, and provide storage for the decimal constants for "A" and "B" during execution. ABCST is the address of AB which contains the ASCII Code 215. BACST is the address of BA which also contains the ASCII Code 215. C260 contains the value of ASCII Zone constant 260 which is subtracted from the Inputs to obtain the decimal constants for "A" and "B".

5.5.3.3.5 Input Routine +ASCII Zone Strip. Input (initially set to zero) is the saved address for return to the main program after completion of the input subroutine. The TIR instruction checks Teletype flag, and if the flag is not cleared, the JMP .-1 causes loop back to the TIS instruction until an input is entered at the Teletype keyboard. When an input for "A" or "B" is entered at the Teletype keyboard, the TRF instruction causes the flag to be cleared, the input is loaded into the J register, and the flag is set to one when done. The TCP instruction causes the input for "A" or "B" to be echoed at the Teletype printer. The TOS and JMP .-1 instructions check to see if the input has been printed and causes a skip to the next instruction when done. The SBJ instruction causes the stored ASCII constant 260 to be subtracted from the contents of the J register via symbolic address tag C260. The J register now contains the decimal constant for the "A" or "B" input. The JMP@ instruction allows a jump back to the main program area via symbolic address tag INPUT.

5.5.3.3.6 Output Subroutine - Output ASCII Expression. Out (initially set to zero) is the saved address for return to the main program after completion of the output subroutine. At entry of the subroutine, the J register contains the address of the first character for the "A > B" or "B > A" literal (which is a carriage return). The STJ instruction causes this address

to be stored one location past loop via loop+1 symbolic address tag. The LDJ instruction causes the J register to be loaded with the address of the loop counter constant. The STJ instruction causes the contents of Memory at C5 to be stored at memory location defined as CTR.

#### NOTE

The output data loop is set up to produce a carriage return and line feed at the Teletype and output "A > B" or "B > A" literally, one character at a time. The number of output characters is five, thus five loops are required to output all characters. Therefore, memory location C5 contains a decimal constant of five. The CTR location allows this count to be decremented during each loop thus saving the constant loop value contained at location C5 for successive executions.

5.5.3.3.7 Output Data Loop. The output data loop begins with a TWLDJ instruction which is a two-word instruction. The J register now contains the first ASCII literal character (ASCII value 215). The TCP instruction causes the first output character (carriage return) to be sent to the Teletype printer. The TOS and JMP.-1 instructions check to see if the Teletype input has been printed, and causes a skip to the next instruction when done. The ISZ instruction causes the memory location "LOOP+1" to be incremented by one via symbolic address tag "LOOP+1". Loop+1 now contains the address of next ASCII character (212). The memory location tagged CTR is decremented by one, and checked for a zero. If location CTR is not zero, the JMP instruction causes an unconditional jump to loop. The J register is loaded with the next ASCII character (212). The above loop repeats itself until all characters are printed out at the Teletype (CTR = 0). When CTR is equal to zero, the next instruction (JMP) is skipped, and the JMP@ instruction causes a return to the main program via saved address at label OUT.

5.5.3.3.8 Output Messages. The output messages contain the literal statements to be printed at the Teletype during the output data loop. The "AB" is the label for the first character of the A > B literal message, and the "BA" is the label for the first character of the B > A message. ASCII values 215 and 212 cause a carriage return and line feed at the Teletype, respectively. During the output data loop, the A > B or B > A literal characters are printed one at a time. For example, if the inputs for "A" and "B" were "3" and "6" respectively, the output would be as follows.

```
36  
B>A63  
A>B
```

#### 5.5.4 TEXT EDITOR

The Symbolic Text Editor (a program itself) is used to create and modify symbolic program (source) tapes via the Teletype keyboard on line. This eliminates the tedious task of generating source program tapes off-line.

With the Symbolic Text Editor loaded into the ND812 Computer, the programmer uses the Teletype keyboard as a typewriter. As the program is entered on the keyboard (as coded), it is immediately stored in a buffer storage area of the ND812 Computer where it can be checked, corrected, and modified. When the programmer is ready to generate the source program tape, the proper command causes the Symbolic Text Editor to produce a source tape suitable for assembling into an object (binary) tape which will, in turn, run on the ND812 Computer.

The Symbolic Text Editor operates in either Command Mode or Text Mode to distinguish between editing commands, and actual text which is entered into the buffer. All commands are single letter or single letter with arguments. Commands are executed by typing the RETURN key at the Teletype keyboard. Refer to the ND812 Symbolic Text Editor Software Instruction Manual, IM41-0002 for detailed description and use of the Editor.

5.5.4.1 PRODUCING EXAMPLE PROGRAM USING THE SYMBOLIC TEXT EDITOR. Now that the example problem has been coded (Table 5-1), the programmer may generate a symbolic source tape via the Symbolic Text Editor. Refer to ND812 Symbolic Text Editor Software Instruction Manual, IM41-0002 for loading and use of the Editor. Appendix A of the Editor Manual provides complete loading and initialization procedures for the ND812 Text Editor. Sections I through V describe the Symbolic Text Editor and its use in generating source program tapes.

Upon command, the Symbolic Text Editor will print the contents of the text buffer at the Teletype. A printout of the example is given in Table 5-2.

#### 5.5.5 BASC-12 GENERAL ASSEMBLER

The BASC-12 Assembler is a 2-pass Assembler (with optional 3rd pass) which is loaded into the ND812 Computer via Teletype or Tape Cassette. The BASC-12 Assembler, hereinafter referred to as the Assembler, translates symbolic mnemonics (source programs in the form of paper tape or cassette) into binary machine instructions (object program). The object program is then directly executable by the ND812 Computer.

There are three Assemblers, as follows.

1. BASC-12 General Assembler, 41-0001, designed to run in a 4K ND812 Computer.
2. BASC-12 General Assembler (8K) for Line Printer Printout, 41-0028.
3. BASC-12 General Assembler (8K) for Teletype Printout, 41-0084.

Refer to the BASC-12 General Assembler Software Instruction Manual, IM41-0001 for detailed description and use of the Assembler.

Table 5-2. Teletype Printout of Example Problem

```

L
/LABEL INSTR OPERAND COMMENTS
/
/INPUT AND STORE VALUES FOR A & B
*200
START, TIF /CLEAR TTY FLAG
      JPS INPUT /GET VALUE FOR A
      STJ A
      JPS INPUT /GET VALUE FOR B
      STJ B
/
/DETERMINE WHICH OF THE TWO VALUES IS LARGER
      LDJ A
      SBJ B /SUBTRACT B FROM A
      SIP J /TEST FOR A POSITIVE
      JMP BRAN /NO! B > A
      LDJ ABCST /YES! A > B
      SKIP /SKIP NEXT INSTRUCTION
BRAN, LDJ BACST
/
/SET UP AND OUTPUT EXPRESSION
      JPS OUT
      STOP
      JMP START
/
/WORKING OR DATA STORAGE AREA
/
A, 0 /CONSTANT A
B, 0 /CONSTANT B
ABCST, AB /ADDRESS OF A > B LITERAL
BACST, BA /ADDRESS OF B > A LITERAL
C260, 260 /ASCII ZONE CONSTANT
/
/INPUT ROUTINE + ASCII ZONE STRIP
/
INPUT, 0 /ENTRY POINT
      TIS
      JMP .-1
      TRF
      TCP /ECHO INPUT AT TELETYPE
      TOS
      JMP .-1
      SBJ C260
      JMP@ INPUT

```

Table 5-2. Teletype Printout of Example Problem (Cont'd.)

```

/
/OUTPUT ROUTINE - OUTPUT ASCII EXPRESSION
/
OUT,      0          /ENTRY POINT
          STJ      LOOP+1
          LDJ      C5      /SET NUMBER OF CHARACTER CONSTANT
          STJ      CTR
/
/OUTPUT DATA LOOP
/
LOOP,     TWLDJ
          0
          TCP
          TOS
          JMP      .-1
          ISZ      LOOP+1
          DSZ      CTR      /TEST FOR ALL CHARACTERS OUT
          JMP      LOOP      /NO
          JMP@     OUT      /RETURN

C5,       5
CTR,      0
/
/OUTPUT MESSAGES
/
AB,       215
          212
          301      /A
          276      />
          302      /B
BA,       215
          212
          302      /B
          276      />
          301      /A

$          /END CHARACTER

```

#### 5.5.5.1 ASSEMBLING THE EXAMPLE PROGRAM USING THE BASC-12 ASSEMBLER.

Once the source tape for the example program has been produced, the programmer may generate a binary (object) tape via the Assembler. There are various options available to the programmer in assembly of the source program. Refer to the BASC-12 General Assembler Software Instruction Manual, IM41-0001 for specific instructions on loading and using the Assembler. Appendix E of the Assembler Manual provides complete procedures for loading and initialization of the Assembler. Sections I through V describe the Assembler, the options available, and the use of the Assembler in generating binary tapes and listings.

If Assembly Language mistakes exist in the coding, the Assembler will detect these errors and provide an error message on printout (pass 3) of the assembler. The following is an example of an error indication on the pass 3 printout.

```

/OUTPUT MESSAGES
/
  0250  0301  AB,      301      /A
  0251  0276                276      />
  0252  0302                302      /B
  0253  0302  BA,      302      /B
IS ^          AT 0254
  0254  0000                276      >
  0255  0301                301      /A
```

Note the "IS ^ AT 0254" which indicates that an error exists at location 0254. The > character should have been preceded by a slash (i.e., / > ).

Even though a source program assembles successfully, the Computer will not execute the program if logic errors exist. In this case, the program would require debugging, editing, and re-assembly.

Table 5-3 provides a listing of the example program as produced by pass 3 of the Assembler.

#### 5.5.6 LOAD AND EXECUTE THE EXAMPLE PROGRAM.

The binary object paper tape produced by the Assembler may be loaded directly into the ND812 Computer via Teletype and executed.

Load and execute the binary tape as follows.

- a. Set Computer front panel power switch to POWER ON position, and Teletype LINE/OFF/LOCAL switch to LINE.
- b. Depress ND812 Computer STOP switch.

- c. Place the binary tape into the Teletype Reader with the leader (level 8 punched) over read head.
- d. Set Teletype START/FREE/STOP switch to START position.
- e. Simultaneously depress ND812 LOAD AR and NEXT WORD switches. The Teletype Reader will step through the paper tape leader and read the program into the ND812 Computer Memory. Upon completion, the Reader automatically stops. After Reader stops, set ND812 Computer SELECTED REGISTER switch in J position and verify that J register is zero (all lamps off). If J register is not zero, repeat steps a through e.
- f. Set Teletype START/FREE/STOP switch to FREE position.
- g. Set ND812 SWITCH REGISTER switches to  $\emptyset 2 \emptyset \emptyset$ , and depress LOAD AR and START switches.

The example program is now in the computer and running waiting for an input for "A". Type a number at the Teletype keyboard and the number will be immediately echoed at the Teletype. Now type another number greater or less for input "B" and the number will be immediately echoed at the Teletype. Next, a carriage return and line feed will occur and a literal statement "A > B" or "B > A" will be printed at the Teletype. The Computer will then stop. Depressing the CONT key restarts the computer for successive execution of the program.

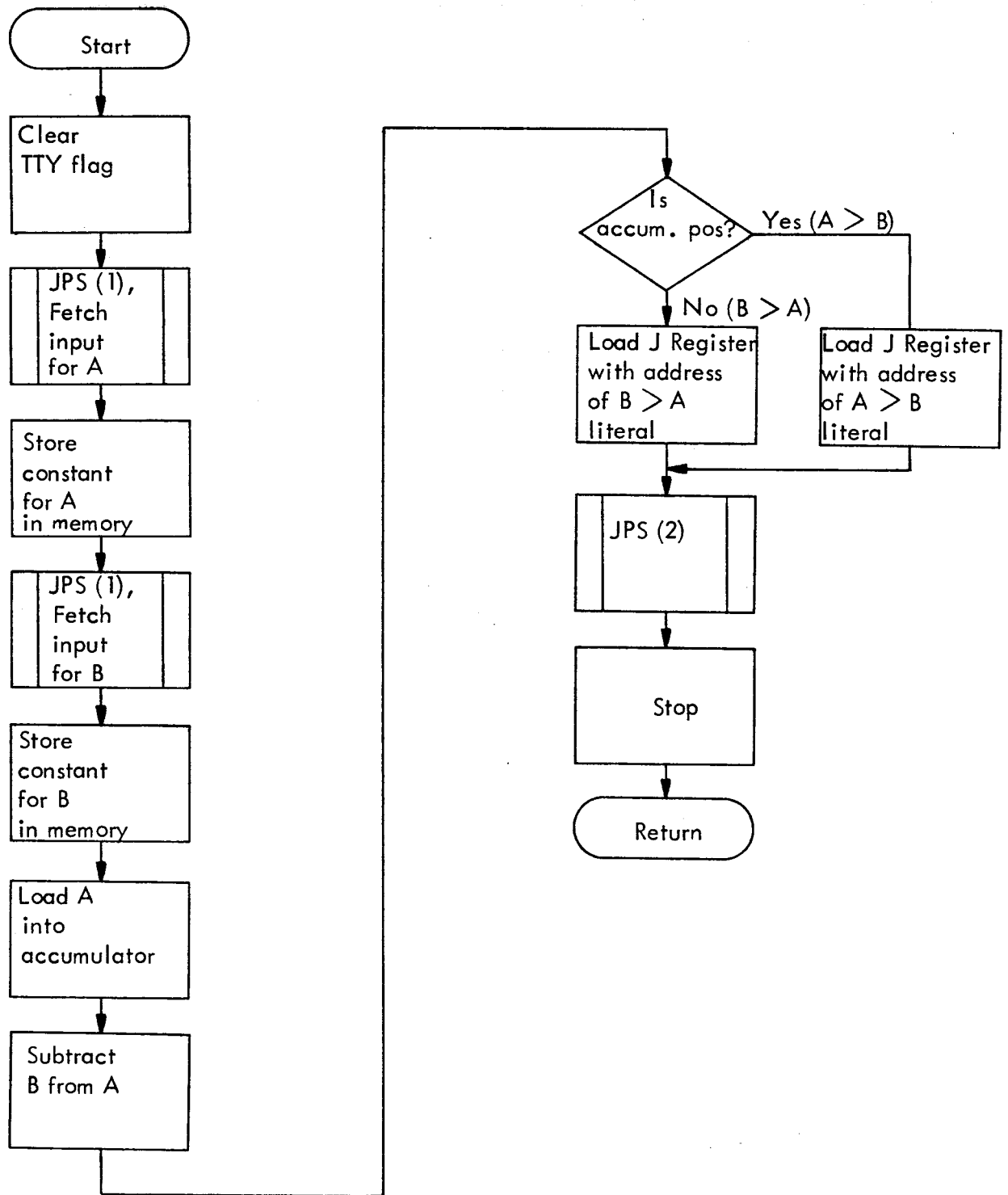


Figure 5-5. Example Program Flowchart  
(Sheet 1 of 2)



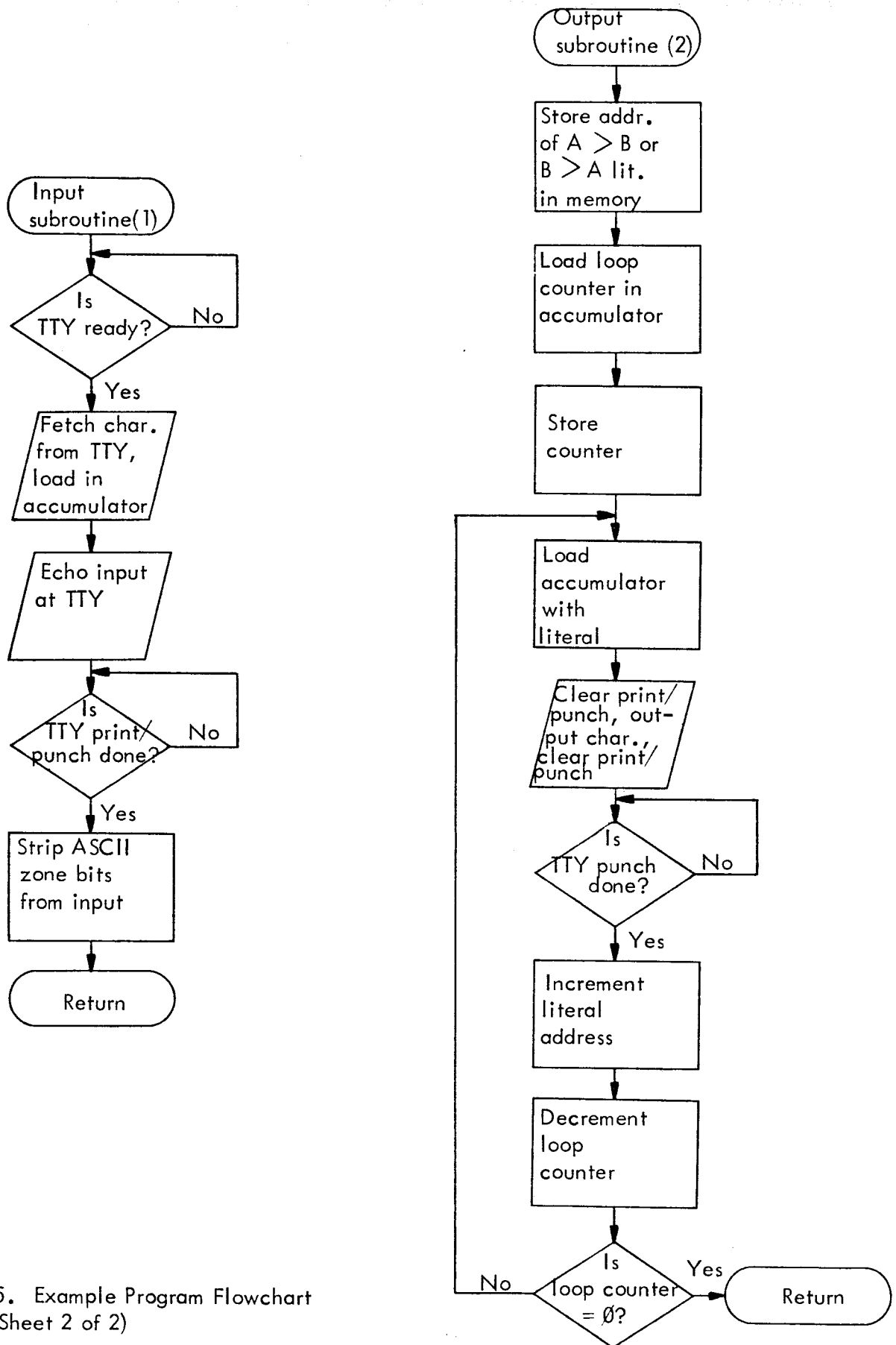


Figure 5-5. Example Program Flowchart  
(Sheet 2 of 2)

Table 5-3. Listing of Example Program Produced by Assembler

```

/INPUT AND STORE VALUES FOR A & B
      *200
0200 7401  START,  TIF          /CLEAR TTY FLAG
0201 6423          JPS      INPUT /GET VALUE FOR A
0202 5415          STJ      A
0203 6421          JPS      INPUT /GET VALUE FOR B
0204 5414          STJ      B

/DETERMINE WHICH OF THE TWO VALUES IS LARGER
0205 5012          LDJ      A
0206 4012          SBJ      B      /SUBTRACT B FROM A
0207 1502          SIP      J      /TEST FOR A POSITIVE
0210 6003          JMP      BRAN  /NO! B > A
0211 5010          LDJ      ABCST /YES! A > B
0212 1442          SKIP     /SKIP NEXT INSTRUCTION
0213 5007  BRAN,  LDJ      BACST

/SET UP AND OUTPUT EXPRESSION
0214 6421          JPS      OUT
0215 0000          STOP
0216 6116          JMP      START

/WORKING OR DATA STORAGE AREA
0217 0000  A,      0          /CONSTANT A
0220 0000  B,      0          /CONSTANT B
0221 0254  ABCST,  AB        /ADDRESS OF A > B LITERAL
0222 0261  BACST,  BA        /ADDRESS OF B > A LITERAL
0223 0260  C260,   260       /ASCII ZONE CONSTANT

/INPUT ROUTINE + ASCII ZONE STRIP
0224 0000  INPUT,  0          /ENTRY POINT
0225 7404          TIS
0226 6101          JMP      .-1
0227 7403          TRF
0230 7413          TCP          /ECHO INPUT AT TELETYPE
0231 7414          TOS
0232 6101          JMP      .-1
0233 4110          SBJ      C260
0234 6310          JMP@    INPUT

/OUTPUT ROUTINE - OUTPUT ASCII EXPRESSION
0235 0000  OUT,    0          /ENTRY POINT
0236 5404          STJ      LOOP+1
0237 5013          LDJ      C5   /SET NUMBER OF CHARACTER CONSTANT

```

Table 5-3. Listing of Example Program Produced by Assembler (Cont'd.)

```

0240 5413          STJ    CTR
/OUTPUT DATA LOOP
0241 0500  LOOP,   TWLDJ
0242 0000          0
0243 7413          TCP
0244 7414          TOS
0245 6101          JMP    .-1
0246 3504          ISZ   LOOP+1
0247 3004          DSZ   CTR    /TEST FOR ALL CHARACTERS OUT
0250 6107          JMP   LOOP   /NO
0251 6314          JMP#  OUT    /RETURN

0252 0005  C5,     5
0253 0000  CTR,    0
/OUTPUT MESSAGES
0254 0215  AB,     215
0255 0212          212
0256 0301          301    /A
0257 0276          276    />
0260 0302          302    /B
0261 0215  BA,     215
0262 0212          212
0263 0302          302    /B
0264 0276          276    />
0265 0301          301    /A

SE 1200
A      ■ 0217
AB     ■ 0254
ABCST  ■ 0221
B      ■ 0220
BA     ■ 0261
BACST  ■ 0222
BRAN   ■ 0213
C260   ■ 0223
C5     ■ 0252
CTR    ■ 0253
INPUT  ■ 0224
LOOP   ■ 0241
OUT    ■ 0235
START  ■ 0200
ER 0000

```

## SECTION VI COMPUTER LANGUAGES

### 6.1 BASC-12 ASSEMBLY LANGUAGE

The BASC-12 Assembly Language provides the programmer with symbolic mnemonics which can be interpreted by the BASC-12 Assembler. It is composed of simple, brief expressions which provide translation from symbolic coding to machine language object coding for the ND812. The BASC-12 Assembler is a two-pass assembler (with an optional third pass) which translates the mnemonics of the source language into machine instructions executable by the ND812 hardware. Pass one generates a symbol table, pass two produces a binary (object) output tape, and pass three provides a listing of the program.

The assembly language includes a wide variety of operations which allow the fabrication of desired fields based on information generated at assembly time. The instruction operation codes are assigned mnemonics which describe the hardware function of each instruction. Assembler directive commands provide the programmer with the ability to generate data words and values based on specific conditions at assembly time. The program counter provides a means of controlling address generation during assembly of a source code program.

#### 6.1.1 SYMBOLIC CODING FORMAT

In writing instructions using the assembly language, the programmer is primarily concerned with three fields: a label field, an operation field, and an operand field. It is possible to relate the symbolic coding to its associated flowchart (if desired) by appending comments to each instruction line or program segments. All of the fields are free-form to provide the greatest convenience possible for the programmer. Consequently, the programmer is not hampered by the necessity to consider fixed-form boundaries in the design of his symbolic coding.

#### 6.1.2 MNEMONIC INSTRUCTION DIRECTIVES

The assembly program recognizes a set of mnemonic instructions representing the machine code instructions listed in Appendix B.

The symbolic assembler directives control the assembly processor just as operation codes

control the central processor. These directives are represented by mnemonics which are written in the operation field of a symbolic line of code; the flexibility of these directives is the key to the power of the assembler. The directives are used to equate expressions, adjust the program counter values, and afford the programmer special control over the generation of object coding. These directives and their respective functions are as follows.

- a. BLOCK, which repeats an instruction n times.
- b. PAUSE, which stops the program to allow some job to be performed and continues when the operator requests it.
- c. FIXTAB, allows labels to be added to permanent symbol table which would normally be erased after pass one.
- d. ERASE, which deletes all entries in the label table except standard system directive labels.
- e. RETURN, which generates the necessary instruction at the end of a subroutine to allow the program to return to the main program.
- f. ENABLE, which defines a special directive for a programmer and allows him to code his own directive.

## 6.2 NUTRAN LANGUAGE

NUTRAN is a conversational, FORTRAN-like language intended for general computational use in scientific applications. Simple commands, a conversational mode, and thorough input checking make the language easy to use without previous programming experience. The NUTRAN programming concept thereby provides the user with an ultimately flexible, expandable, and extremely "usable" data acquisition and analysis center which users can tailor to subjective needs.

The uses of NUTRAN are varied. Nuclear Data initially designed NUTRAN for scientific uses, and in particular, for stating mathematical and scientific problems in a language more closely associated with experimental requirements than with direct control of the ND812 Computer. NUTRAN, however, has also proven itself in many commercial and industrial applications. As specific user needs develop, any of the valid NUTRAN commands described in NUTRAN manual may be implemented to further extend the practicality of NUTRAN.

The outstanding characteristic of NUTRAN is the continuing dialog between user and computer. NUTRAN statements are entered by the user at a remote device. When the program is executed, the statements are then automatically translated during execution, the interpreter responds by directing an error printout on Teletype. Also, if desired, as the program is being executed, literal messages and results of computations may be printed on Teletype. The features of NUTRAN conversational language are as follows.

1. The user has immediate and sustained access to the computer.
2. The user may selectively construct, execute, and edit statements or complete routines, change values of variables, and request information from the computer.
3. The user has diagnostic facilities to debug his NUTRAN program.
4. The user need not be concerned about integer and floating point data type formats.

## SECTION VII PROCESSOR AND PERIPHERALS

### 7.1 GENERAL

A typical ND812 processing system is comprised of an ND812 computer, an ASR33 Teletypewriter set, and an assortment of peripheral devices tailored to needs of the user. This chapter addresses itself to general descriptions of the individual equipments or "building blocks" which constitute Nuclear Data systems.

### 7.2 THE ND812 COMPUTER

The ND812 is a general-purpose computer designed for scientific applications. The basic ND812 is a 12-bit, 4K computer, with optional 8K, 12K or 16K memories. The ND812 is extremely versatile in that all core locations (up to 16K) are directly addressable by a two-word instruction. A total of 256 single-word or 4095 two-word input/output (I/O) commands is possible. Other outstanding features are the 12 or 24-bit programmed I/O transfer, a four-level programmable priority interrupt, four microprogrammable pulses per I/O instruction, direct memory access, four arithmetic registers, hardware multiply and divide, and fully-integrated control logic circuitry.

#### 7.2.1 ND812 COMPUTER FRONT PANEL

Figure 7-1 illustrates physical location of the ND812 Central Processor front panel controls and indicators. Table 7-1 lists and describes ND812 Central Processor front panel controls and indicators. The first column lists nomenclature, second column lists the control description and the third column describes the function.

Table 7-1. ND812 Central Processor Controls and Indicators

| Control/Indicator                     | Description               | Function  |
|---------------------------------------|---------------------------|---|
| POWER OFF/POWER ON/CONTROL OFF switch | Three position key switch | Placing this key switch in POWER OFF position disables all primary power for the processor. In POWER ON position, power is applied to all circuits and manual program control is possible. In |

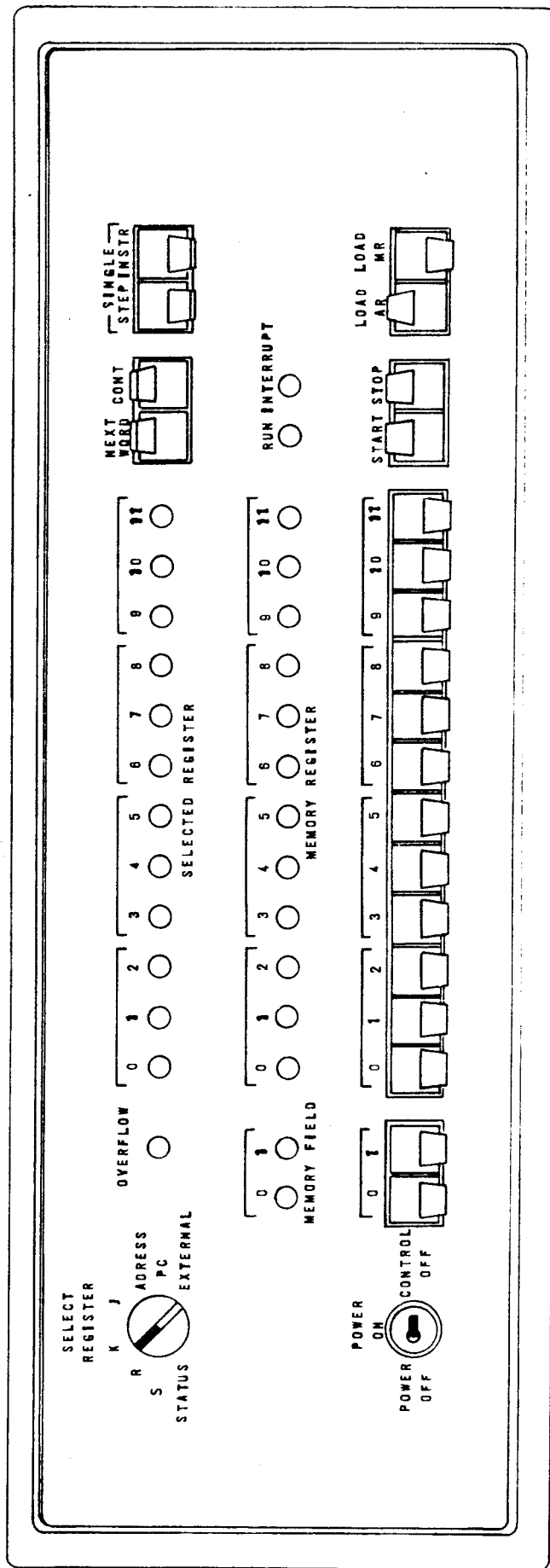


Figure 7-1. ND812 Central Processor Controls and Indicators



Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator | Description | Function |
|-------------------|-------------|----------|
|-------------------|-------------|----------|

SELECT REGISTER switch

Eight position rotary switch

CONTROL OFF position, power is maintained but all ND812 front panel switches are disabled.

SELECT REGISTER switch is an eight position rotary switch that allows the contents of major registers to be displayed when the processor is stopped. Content of the chosen register is displayed by SELECTED REGISTER indicator lamps.

SELECTED REGISTER indicators

12 selectable indicator lamps

Displays contents of the register selected by SELECT REGISTER switch. Listed below are individual SELECT REGISTER switch selections and their significance.

a. STATUS position

This position monitors an extremely valuable 12-bit word that indicates the following conditions.

SELECTED REGISTER Indicator Lamp

Designation

- |   |   |
|---|---|
| 0 | When this lamp is on, the flag is non-zero. This lamp is extinguished when the flag is zero.            |
| 1 | When this lamp is on, the overflow is non-zero. This lamp is extinguished when the overflow is zero.    |
| 2 | These lamps indicate the Memory Field in which the last executed JPS Instruction is located that caused |
| 3 |   |

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator | Description | Function |
|-------------------|-------------|----------|
|-------------------|-------------|----------|

the program to branch to another Memory Field.

| <u>Memory Field</u> | <u>Lamp 2</u> | <u>Lamp 3</u> |
|---------------------|---------------|---------------|
| 0                   | off           | off           |
| 1                   | off           | on            |
| 2                   | on            | off           |
| 3                   | on            | on            |

- 4
  - 5
- These lamps indicate the Memory Field in which execution was taking place at the time the last interrupt occurred.

| <u>Memory Field</u> | <u>Lamp 4</u> | <u>Lamp 5</u> |
|---------------------|---------------|---------------|
| 0                   | off           | off           |
| 1                   | off           | on            |
| 2                   | on            | off           |
| 3                   | on            | on            |

- 6
  - 7
  - 8
- When this lamp is on, the highest level priority interrupt circuitry is enabled.
- When this lamp is on, the highest level and the B level priority interrupt circuits are enabled.
- When this lamp is on, the highest level and the A level priority interrupt circuits are enabled.

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator | Description | Function |
|-------------------|-------------|----------|
|-------------------|-------------|----------|

9 When this lamp is on, the lowest level priority interrupt circuitry is enabled as well as the A, B, and highest level interrupt circuitry.

10 These lamps indicate the Memory Field in which the program is currently being executed (actual extension of the Program Counter and Address Register).  
11

| Memory Field | Lamp 10 | Lamp 11 |
|--------------|---------|---------|
| 0            | off     | off     |
| 1            | off     | on      |
| 2            | on      | off     |
| 3            | on      | on      |

b. S Position

Displays the 12-bit contents of the S Register via the SELECTED REGISTER indicator lamps.

c. R Position

Displays the 12-bit contents of the R Register via SELECTED REGISTER indicator lamps.

d. K Position

Displays the 12-bit contents of the K Register via SELECTED REGISTER indicator lamps.

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator  | Description                     | Function  |
|--------------------|---------------------------------|---|
|                    |                                 | <p>e. J Position</p> <p>Displays the 12-bit contents of the J Register via SELECTED REGISTER indicator lamps.</p> <p>f. ADDRESS Position</p> <p>Displays the 12-bit contents of the Address Register via SELECTED REGISTER indicator lamps.</p> <p>g. PC Position</p> <p>Displays the 12-bit contents of the Program Counter via SELECTED REGISTER indicator lamps.</p> <p>h. EXTERNAL Position</p> <p>Used for service only.</p> |
| OVERFLOW indicator | Indicator Lamp                  | An overflow condition created by either a J or K Register arithmetic operation causes the overflow bit to be complemented. OVERFLOW indicator lamp will light when the overflow bit is non-zero.  |
| NEXT WORD switch   | Momentary contact rocker switch | Momentarily depressing the spring loaded NEXT WORD switch, sets the contents of the Program Counter into the Address Register, increments the Program Counter, and updates the MEMORY REGISTER indicator lamps to reflect the contents of memory at the address now contained in the Address Register.  |
| CONT switch        | Momentary contact rocker switch | Momentarily depressing the spring loaded CONT switch initiates program execution at the address specified by the Program  |

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator             | Description                     | Function   |
|-------------------------------|---------------------------------|--|
|                               |                                 | <p>Counter. Start clear is not generated.</p> <p>This switch is disabled when the processor is in the run mode.</p>  |
| SINGLE STEP/<br>INSTR switch  | Two position<br>rocker switches | <p>With the SINGLE STEP switch in the Up position, the run mode is terminated and the timing circuits are disabled at the completion of one cycle (step) of the current instruction. Depressing CONT switch advances the program one additional cycle of the current instruction.</p> <p>Interrupt circuitry is disabled when a Single Step operation is performed.</p> <p>With the SINGLE INSTR switch in the Up position, execution is stopped at the end of each complete instruction. Depressing CONT switch executes the next logical instruction.</p> <p>DMA circuitry is disabled when a Single Instruction operation is performed.</p> |
| INTERRUPT<br>indicator        | Indicator<br>lamp               | When the INTERRUPT indicator lamp is lit, one or more of the priority interrupt levels are enabled.  |
| RUN indicator                 | Indicator<br>lamp               | When the RUN indicator lamp is lit, program execution is in process.   |
| MEMORY REGISTER<br>indicators | Indicator<br>lamp (12)          | MEMORY REGISTER indicator lamps indicate the 12-bit contents of memory at the location specified by the Address Register. The 12-bit word is displayed in binary format with bit 0 representing the most significant bit.  |

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator        | Description                     | Function  |
|--------------------------|---------------------------------|---|
| LOAD MR switch           | Momentary contact rocker switch | <p>Momentarily lifting the LOAD MR switch transfers the Program Counter into the Address Register, initiates a memory cycle that loads the Switch Register contents into the address specified by the updated Address Register, and increments the Program Counter. Memory Register indicator lamps will then display the deposit, and the Address Register indicator lamps will display the deposit address.</p> <p>This switch is disabled when the processor is in the run mode.</p> |
| LOAD AR switch           | Momentary contact rocker switch | <p>Momentarily depressing the LOAD AR switch loads the contents of the Switch Register into the Program Counter and Address Register, and updates the Memory Register to reflect the contents of memory at the address contained in the Address Register. MEMORY FIELD switches are loaded into the Memory Field bits as an extension of the Program Counter.</p>   |
| STOP switch              | Momentary contact rocker switch | <p>Momentarily depressing the STOP switch terminates program execution at completion of the current instruction. Program Counter contains the address of the next instruction after program termination.</p>  |
| START switch             | Momentary contact rocker switch | <p>Momentarily depressing the START switch initiates program execution at the memory location specified by the Program Counter and generates a start clear. This switch is disabled when the processor is in the run mode.</p>  |
| SWITCH REGISTER switches | Two position rocker switches    | <p>Manual loading of a 12-bit word is accomplished by these switches. Words are arranged in binary format with bit 0 repre-</p>   |

Table 7-1. ND812 Central Processor Controls and Indicators (Cont'd.)

| Control/Indicator | Description | Function |
|-------------------|-------------|----------|
|-------------------|-------------|----------|

MEMORY FIELD switches and indicators

Two position rocker switches and indicator lamps

senting the most significant bit. Switches in the Up position correspond to binary 1's, Down to 0's. Contents of the SWITCH REGISTER is loaded into the Program Counter and Address Register by depressing the LOAD AR switch, or into memory by lifting the LOAD MR switch. In addition, the SWITCH REGISTER can be read by the processor during program execution with a LJSW Instruction.

MEMORY FIELD switches determine the specific Memory Field into which data is read from, or loaded into, or execution initiated. Functionally, these switches are an extension of the Program Counter and Address Register and only affect the Hardware Loader and the LOAD AR switch. Memory Fields are numbered in binary increments from 0 to 3 and each field represents  $4096_{10}$  or  $10000_8$  memory locations (0000-7777<sub>8</sub>).

| Memory Field | Switch 0 | Switch 1 |
|--------------|----------|----------|
| 0            | off      | off      |
| 1            | off      | on       |
| 2            | on       | off      |
| 3            | on       | on       |

MEMORY FIELD indicator lamps indicate the Memory Field in which a program is currently being executed. Lamps are numbered in a form identical to the Memory Field switches.

### 7.2.2 REAR PANEL

External features of the ND812 Processor rear panel consist of printed circuit and coaxial-type electrical connections:

| <u>Panel Device</u>                               | <u>Function</u>  |
|---|--|
| Input/Output Printed Circuit Board Connectors (2) | Provided connection for the I/O signals of I/O devices. Refer to the FUNCTIONAL ANALYSIS section of the ND812 Computer Maintenance Manual for individual signal terminations.                |
| Teletype Integrated Circuit (IC) Connector        | Provides connection for the input/output signals of the Teletype 33ASR. Refer to the FUNCTIONAL ANALYSIS Section of the ND812 Computer Maintenance Manual for individual signal termination. |
| AC Line Receptacles (2)                           | Provide connection for supplying primary power to the teletype.  |

### 7.2.3 ND812 TECHNICAL SPECIFICATIONS

| <u>Feature</u> | <u>Function</u>   |
|----------------|---|
| Memory         | Magnetic core, 4096 words, 12 bits, 2 $\mu$ s cycle time. Memory options: minimum 4K, field expandable to 16K in 4K increments.   |
| Addressing     | Relative, indirect, and direct. Hardware multiple field control.  |
| Arithmetic     | Parallel, binary, fixed point, 2's complement. Hardware multiply and divide are standard features.  |
| Instructions   | Single and two-word instructions which include sixteen memory reference instructions, three literals, and more than fifty arithmetic and register control instructions. |
| Input/Output   | Interrupt: programmable 3-level priority interrupt. Trap to any odd numbered core location in first 4K of memory.   |



Feature

Function

|                    |  |
|--------------------|--|
|                    | <p>Programmed I/O transfer; capability per single I/O instruction:</p> <ul style="list-style-type: none"><li>Transmit 12 or 24 bits.</li><li>Receive 12 or 24 bits.</li><li>Transmit 12 and receive 12 bits.</li><li>Receive 12 and transmit 12 bits.</li></ul> <p>I/O instruction: Includes four microrprogrammable pulses for multi-function operation with a single instruction.</p> <p>Single-word instructions: 256 possible I/O commands at 3 <math>\mu</math>s per instruction.</p> <p>Two-word instructions: 4096 possible I/O commands at 5 <math>\mu</math>s per instruction.</p> <p>Control, data, and sense lines: total of 75 available on a single connector.</p> <p>Direct Memory Access (DMA): 6 megabits per second; read, load, increment or decrement on DMA on a single cycle.</p> |
| Accumulator        | Dual accumulators with individual subaccumulators.   |
| Control Panel      | <p>Constant display of memory register with switch-selected display of six other registers and two busses.</p> <p>Front panel removable key lock. Power off, on, panel lock.</p>   |
| Timing             | 16 MHz crystal-controlled clock assures absolute timing and drift-free operation.  |
| Size               | 19-in w x 7-in h x 22-in d.  |
| Weight             | 60 lb.   |
| Power Requirements | 400 W @ 115/230 Vac, 50/60 Hz.   |

## 7.3 THE ASR33 TELETYPEWRITER

ND has selected Model 33ASRs (automatic send-receive) as the basic input/output (I/O) terminals for its computer systems because it has proved to be the most versatile, reliable, and economical device available for rapid data communications.

### 7.3.1 CAPABILITIES

The Model 33ASR can transmit information manually (through its keyboard) or automatically (by sensing the perforations in paper tape). It can receive data from its own keyboard or tape reader or from distant sets (such as page copy with or without an accompanying perforated tape).

The equipment operates on an 8-level code compatible with the permutation code approved by the American Standards Association for Information Interchange (ASCII). This means that the Model 33 can communicate with computers and other business machines to provide a fast, efficient system for the collection, processing, and distribution of data. Teletypes can also use the eighth level of the code to generate "even" parity for error detection.

The paper tape punch and reader of the Model 33ASR offers a number of data communication uses; for instance, it can combine tape data from a number of sources into one error-free master tape. The tape reader then can automatically transmit this data to other teletypes or computers at maximum speed.

Use of paper tape offers many advantages. It is easy to handle, accommodates data of any length, and is still the least expensive and most reliable continuous recording medium available.

### 7.3.2 TECHNICAL SPECIFICATIONS

| <u>Feature</u> | <u>Function</u>  |
|----------------|--|
| Speed          | Char/sec      6.0      6.0      7.5      10                    |
|                | Wds/min        60.0     66.0     75       100                  |
|                | Bauds          66.0     74.0     82.5     110                  |
| Code           | 8-level, 11 unit basis (ASCII)                                 |
| Tape           | 8-level, 1-in wide oiled paper                                 |
| Printer        | Friction feed platen for 8 1/2-in single or multiple-ply paper |
|                | Horizontal spacing 10 cpi (12 characters optional)             |
|                | Vertical spacing single or double row (3 or 6 lpi)             |

| <u>Feature</u>       | <u>Function</u>  |          |         |     |     |
|----------------------|--|----------|---------|-----|-----|
| Keyboard             | 4-row, 8-level. Similar to typewriter.   |          |         |     |     |
| Temperature          | Operating: 40°-110°F ambient; humidity: 95% max.   |          |         |     |     |
| Size                 | 22-in w x 37 1/2-in h x 18 1/2-in d.   |          |         |     |     |
| Weight               | 56 lb.   |          |         |     |     |
| Power Requirements   | 115 V AC $\pm$ 10%, 60 Hz $\pm$ 0.45 Hz, single-phase synchronous motor; 50 Hz motor also available.<br><br>Approx. input current:<br><br><table border="0" style="margin-left: 100px;"> <tr> <td>starting</td> <td>running</td> </tr> <tr> <td>8 A</td> <td>2 A</td> </tr> </table><br>Approx. wattage:<br>RO - 95 W<br>KSR - 95 W<br>ASR - 110 W | starting | running | 8 A | 2 A |
| starting             | running  |          |         |     |     |
| 8 A                  | 2 A  |          |         |     |     |
| Maintenance Interval | Once every six months or after 500 operating hours, whichever occurs first.  |          |         |     |     |

## 7.4 PERIPHERAL EQUIPMENT

Selection of peripheral equipment and software are fundamental aspects of computer system design; these considerations, quite literally, are what distinguish a mere processor from an application-tailored, cost-efficient computer system. Some of the options and peripheral equipment presently available for ND812 systems are outlined below.

### 7.4.1 ND812 MEMORY EXTENSIONS

The Memory Extension option expands the storage capacity of the ND812 computer to 16,384, 12-bit words. Two types of memory extension are available: 4096, 12-bit words or 8192, 12-bit words. Expansion of the 4K, ND812 computer to 8K is accomplished by exchanging the 4K memory stack for an 8K memory stack and adding one memory field control (MFC) and memory inhibit sense (MIS) printed circuit board. Expansion of the 8K ND812 computer to 12K or 16K is accomplished by addition of the 4096, 12-bit or 8192, 12-bit memory extension units. The 8K ND812 computer, equipped with a 4K memory extension unit, can be expanded to 16K by exchanging the 4K memory stack of the extension unit for an 8K memory stack and adding one MIS printed circuit board.

Extended address selected control for directly addressing up to 16,384 words is provided by the MFC printed circuit board. Addition of this board activates the indicators and switches associated with the extended addressing capability. These switches function in the same manner as the switch register to load information into the memory register when the load address key is depressed.

#### 7.4.2 TAPE CASSETTE SYSTEM

The Nuclear Data Tape Cassette System (TCS) is a high-performance, serial-by-bit, digital tape cassette drive designed specifically to provide a precision data storage/retrieval capability for the ND812 computer. Other systems applications of this unit include: data acquisition, keyboard control, analytical instrumentation, medical instrumentation, or any area in which high density storage and high speed read/write capability are required.

The tape cassette is available with one, two or three tape cassette drives; hence the computer or data system offers the advantages of multiple magnetic tape files in a single integrated unit. Data are written on two redundant tracks to provide single-bit error correction on a character-by-character basis. Each tape unit employs a spindle rather than capstan drive. This decreases tape wear and allows easier bi-directional operation and faster access to stored information. Adaptation of the tape cassette to most data systems is accomplished by use of single input/output (I/O) circuitry. All I/O logic levels are DTL/TTL compatible.

The tape cassette operates under program control of the ND812 or applicable data systems. Each cassette is independently controlled (providing up to three separate files).

Data are written in records of any length. The records may be written or read alternately among the cassettes in any program sequence. Standard ND812 program controls are: write data, write a filemark, read data, high speed forward, space forward, space forward to a selected file, and high speed reverse.

#### 7.4.3 MAGNETIC TAPE

The magnetic tape provides an IBM-compatible magnetic tape I/O facility for the ND812 Computer; it is capable of operating at a synchronous read/write speed of 45 in/s. The system consists of a synchronous read/write, 7 or 9-track magnetic tape transport, a 7 or 9-Track magnetic tape formatter and an interface to the ND812 computer.

The tape transport employs a single capstan velocity drive system and a constant tension mechanism to hold the tape in contact with the capstan at all times. The controlled-tension tape path offers increased tape life and maximum tape protection. Start/stop characteristics and tape speed are determined solely by the servo driver single capstan and are held constant regardless of normal environment, line voltage, or frequency variations. Positive control of start/stop cycles results in restriction-free programming. The unit uses a single magnetic head which is electronically switched from write or read operations. Because the read/write head is the only surface in sliding contact with the oxide side of the tape, dropout errors are virtually eliminated.

An operator control panel is supplied with the unit for local operation and indication. Indicators show the status of the systems under both local and remote command conditions.

Local operator controls include; on/off, load, on-line/off line, forward, reverse, and rewind.

The data format is NRZI, IBM-compatible including the precise requirements for System/360, 9-track, 800 BPI, operation. All IBM-required tape marks, gaps, parities, and cyclic redundancy checks are performed internally.

#### 7.4.4 CARTRIDGE DISC MEMORY

The cartridge disc memory is a medium-speed, random-access, bulk storage device. The standard system operates through the ND812 data break facility to provide one million, unformatted, 12-bit words of storage.

Two basic assemblies comprise the disc memory system: a cartridge disc drive unit and a controller interface to the ND812 computer. The drive unit contains a removable cartridge which houses one disc; a spindle drive and control circuit; and a single read/write head positioner. Start/stop characteristics of the head positioner and spindle drive speed are electronically controlled and held constant regardless of line voltage or frequency variations by a regulated power supply. A single controller unit can be used to service up to four disc drive units.

The disc drive unit, controller, and associated power supplies are contained in standard 19-in rack mount chassis.

#### 7.4.5 FIXED HEAD DISC MEMORY

The fixed-head disc memory is a high-speed, random-access, bulk storage device. The disc memory operates through the ND812 data break facility to provide 262,000 12-bit words of storage. Optional disc memories with storage capacities from 32,000 to 500,000 words are available.

Two basic assemblies comprise the disc memory: a fixed-head disc storage unit and a controller-interface for the ND812 computer. The storage unit contains a nickel cobalt-plated disc, driven by an integrally mounted, direct-drive motor. Data are recorded on a single disc surface by a separate read/write head for each track. The integral drive system and electronic track switching combine to maximize system data throughput.

The disc memory, controller, and associated power supplies are contained in a standard 19-in rack mount chassis.

#### 7.4.6 LINE PRINTER AND CONTROL

The line printer and control provides a high speed output facility capable of printing alphanumeric data at speeds of up to 1110 lines per minute. The line printer is an impact

type which uses a revolving 64-character drum and one hammer per column. The drum speed of 1760 rpm allows printing speeds of 356 lines per minute for a full 80 columns and listings as fast as 1110 lines per minute for 20 columns.

Paper feed is controlled by a pair of pen-fed tractors for 1/2-in hold center, edge-punched, fan-fold paper. The tractors are adjustable to accommodate paper widths from 4 to 9-7/8 inches. The printer uses single-ply or multiple-ply carbon fan-fold paper and prints up to six copies.

#### 7.4.7 PAPER TAPE PERFORATOR, READER AND CONTROL

The Paper Tape Perforator, Reader and Control provide either a medium or high-speed program and data input and/or output facility for the ND812 Computer. Each system consists of a paper tape punch and reader interface to the ND812 Computer and a compatible reader, punch, or reader-punch combination.

Both paper tape readers are unidirectional, use servo stepping motors, and contain photoelectric tape sensors. One reader is equipped with two tape-handling reel assemblies, each of which consists of a six-inch reel; a constant torque drive; a tension arm; and an "on/off" switch operated when the tension arm is in its raised position. Each reeler operates independently and allows high-speed spooling when the tape is not passed through the read head. Loop tape operation is possible by placing both tension arms in the raised position. The other reader is designed for strip and loop reading and is equipped with supply and take-up bins for fan-fold paper tape. Both readers can be mounted in standard 19-in racks for simple, full-view tape loading.

The paper tape perforator is unidirectional, uses a synchronous sprocket drive, contains a removable chad disposal bin, and is equipped with a paper tape supply reel. Included with the perforator is a 19-in rack mount unit which contains a power supply and the punch drive circuits. The perforator is equipped with an automatic punch turn-on circuit. This circuit places the punch motor under control of the ND812 Computer so that it is enabled only during punch operations. The punch turn-on circuit can also be enabled by a front-panel pushbutton for generating blank tape.

#### 7.4.8 COMPUTER INPUT/OUTPUT WRITER

The computer input/output writer provides a hard copy output and keyboard input facility with input/output speeds of 15 characters per second. Both keyboard entry and typeout use IBM-correspondence code to provide all alphabetic, numeric and special characters. Input facilities for carriage return, space, tabulation, backspace, and upper case are provided by the keyboard. Output facilities for carriage return, space, tabulation, upper case and lower case are provided by the ND812 computer.

#### 7.4.9 POWER RESTART OPTION

A power restart option is available which traps to octal location 40 whenever a power failure or low power problem is encountered. Data contained in all registers are saved and a routine is written which restores these registers and re-initiates the program.

#### 7.4.10 REAL TIME CLOCK OPTION

This is a program-controlled, 100-kHz, clock-interrupt which can be preset to 20  $\mu$ s minimum to 10 s maximum. Two presettable digits can be loaded into the J register while the clock is running, allowing the program to determine the remaining time before the next clock event. This option is of value for any timed or gated event, e.g., acquisition time or variable pulse generator applications. The clock-interrupts trap to octal location 1.

## SECTION VIII THE ND PROGRAM LISTING

### 8.1 GENERAL

The Nuclear Data Program Listing iterates all software available for the ND812 processor. Programs are arranged by category (utility, system, or diagnostics) and control number (e.g., 41-0001); beside each entry is a brief description of the given program's capability.

The dynamics of computer technology are such that new programs and program applications are generated at a rate which requires a continuing update of the Program Listing. Consequently, ND publishes addenda for the benefit of ND812 users and other interested parties which are periodically compiled into new master listings. Copies are available from:

Technical Documentation Department  
Nuclear Data, Incorporated  
Golf and Meacham Roads  
Schaumburg, Illinois 60172

Following is the ND812 Program Listing; for the aforementioned reason, however, it should not be construed to be comprehensive.

### 8.2 UTILITY PROGRAMS

| <u>Control No.</u> | <u>Title</u>  | <u>Description</u>  |
|--------------------|---------------|---|
| 41-0005            | Binary Loader | Loads binary formatted program records into the computer via high or low speed paper tape or magnetic tape cassette.  |
| 41-0006            | Binary Writer | Writes binary formatted records in arbitrary block sizes from the memory field in which it is located via low or high speed paper tape or magnetic tape cassette. |
| 41-0007            | Chess Game    | A demonstration game which permits the user to play chess with the ND812. The   |



| <u>Control No.</u> | <u>Title</u>               | <u>Description</u>  |
|--------------------|----------------------------|---|
|                    |                            | program maintains the chess board and will not allow an illegal move. Chess moves are entered via the Teletype.   |
| 41-0008            | Binary Copier              | Duplicates and verifies binary formatted paper tapes.   |
| 41-0009            | Master Tape Duplicator     | Permits duplication of any paper tape. The program allows for creation of a master tape, duplication of the master tape and verification of the duplicate or master.  |
| 41-0010            | Binary Loader/Verifier     | Compares the original binary formatted paper tape with the contents of the computer memory. Differences are listed on the teletype as they are encountered. The program also allows reloading of the original tape during comparison. |
| 41-0017            | Integer Interpreter        | Provides double precision addition, subtraction, multiplication, division, and I/O routines for the BASC-12 coded programs.   |
| 41-0018            | Numbers Game               | A demonstration game designed to indicate the sort of user-processor interaction that is typical of ND812 software systems.   |
| 41-0022            | Short Form Binary Loader   | Loads binary formatted paper tapes into the computer via the low speed reader only.   |
| 41-0023            | Short Form Binary Writer   | Writes binary formatted program records in arbitrary block sizes from the memory field of the computer in which it is located via the high speed paper tape punch only.   |
| 41-0024            | Short Form Octal Debug Aid | Permits interrogation and modification of the computer memory using the teletype keyboard. The program  |

| <u>Control No.</u> | <u>Title</u>                                       | <u>Description</u>   |
|--------------------|--|--|
|                    |  | aids in debugging and modification of programs created with the BASC-12 General Assembler (41-0001).   |
| 41-0030            | Binary Paper Tape to Magnetic Tape Cassette Copier | Duplicates binary formatted paper tapes on magnetic tape cassettes.  |
| 41-0031            | Multiple Field Binary Writer                       | Writes binary formatted program records in arbitrary block sizes from any of the computer memory fields via low or high speed paper tape or magnetic tape cassette.  |
| 41-0033            | Multiple Field Octal Debug                         | Permits interrogation and modification of the contents of any address in any memory field via the teletype keyboard. The program aids in debugging and modification of multiple field programs created with the BASC-12 General Assembler (41-0001). |
| 41-0035            | Disk System Supervisor                             | Defines the read/write commands for the cartridge disk. Included are set and print disk read/write address, load program from the teletype at current disk write address and load program into computer memory from current disk read address.       |
| 41-0041            | Multiple Field Floating Point Interpreter          | Provides multiple field arithmetic floating point and input/output (I/O) routines for the BASC-12 coded programs.  |
| 41-0042            | Extended Functions I                               | An overlay program for the Multiple Field Floating Point Interpreter (41-0041) which provides exponent log, square, and square root functions.   |
| 41-0043            | Extended Functions II                              | An overlay program for the Multiple Field Floating Point Interpreter (41-0041) which provides sine, cosine, and arc tangent functions.   |

| <u>Control No.</u> | <u>Title</u>                                    | <u>Description</u>   |
|--------------------|---|--|
| 41-0044            | Floating Point Operate Instructions             | An overlay program for the Multiple Field Floating Point Interpreter (41-0041) which provides floating point and operate (FNEG, FCLR, FSIM, FSIP and FSIZ) instructions.   |
| 41-0050            | Cassette Verifier                               | Compares the original binary formatted magnetic tape cassette with the content of the computer memory. Differences are listed on the teletype as they occur.   |
| 41-0052            | Basic Disk Autoloader                           | Writes itself into disk sector one (auto-load sector). When autoload is selected, the Disk System Supervisor (41-0035) is loaded into memory and activated.  |
| 41-0053            | Basic Disk Handler Dump                         | Writes the Disk System Supervisor (41-0035) from memory to the appropriate disk sectors for the Basic Disk Autoloader (41-0052).   |
| 41-0054            | Octal Memory Dump                               | Dumps the entire contents of any memory field at the teletype or line printer with address identification every eighth address. The program aids in debugging when an image of the entire memory is to be studied in detail to localize a problem. |
| 41-0080            | Disk System Supervisor - Hi Density             | Basically the same as the Disk System Supervisor (41-0035) except that it uses a high density cartridge disk.  |
| 41-0085            | PEC Magnetic Tape Copier                        | Reads or writes 8K core images from or to PEC 7 or 9 track magnetic tape with each 8K block identified by a user specified tagword.  |
| 41-0089            | Multi-Field Binary Loader For High Speed Reader | Loads binary formatted programs into any ND812 memory field via high-speed paper tape reader.  |
| 41-0091            | Binary Handler                                  | Transfers binary formatted program records from high or low speed tape reader or magnetic tape cassette to high or low speed tape punch or another magnetic tape cassette.   |

| <u>Control No.</u> | <u>Title</u>             | <u>Description</u>  |
|--------------------|--------------------------|---|
| 41-0116            | Trace Diagnostic Program | A single-field relocatable diagnostic that prints out ND812 status information for each line of code. Print-out may be via high-speed line printer or Teletype. |

### 8.3 SYSTEM SOFTWARE

| <u>Control No.</u> | <u>Title</u>                        | <u>Description</u>   |
|--------------------|-------------------------------------|--|
| 41-0001            | BASC-12 General Assembler           | Translates source programs written in BASC-12 assembly language into binary formatted object programs. Statements are translated on a one-for-one basis, allowing complete control over the statements actually executed by the computer during run time. Input is via the Teletype or magnetic tape cassette. Output is via the Teletype. |
| 41-0002            | Symbolic Text Editor                | Manipulates strings of BASC-12 coded source programs or other text material using keyboard entry commands. Insertions, deletions, and additions to the text are accomplished without retyping the entire text each time modification is necessary. Output is via the Teletype or magnetic tape cassette.                                   |
| 41-0026            | BASC-12 Line Printer Assembler      | Basically the same as the BASC-12 General Assembler (41-0001) except that it uses a line printer as an output device in place of the Teletype.   |
| 41-0028            | BASC-12 Line Printer Assembler (8K) | Basically the same as the BASC-12 Line Printer Assembler (41-0026) except that it allows use of a larger number of user symbols and permits output via magnetic tape cassette. Requires an 8K computer memory.   |

| <u>Control No.</u> | <u>Title</u>                      | <u>Description</u>  |
|--------------------|-----------------------------------|---|
| 41-0036            | Disk Editor                       | Basically the same as the symbolic Text Editor (41-0002) except that it allows a larger amount of text material and uses a cartridge disk as an output device in place of the Teletype. Requires an 8K computer memory.   |
| 41-0037            | BASC-12 Disk Assembler            | Basically the same as the BASC-12 General Assembler (41-0001) except it allows a larger number of user symbols and uses a cartridge disk as an output device in place of the Teletype. Requires 8K computer memory.   |
| 41-0059            | NUTRAN Conversational Compiler    | NUTRAN is an on-line conversational compiler which permits interpretive execution of programs written in FORTRAN syntax using the Teletype as the principal input/output device. The program is intended to provide the scientific user with a means of writing mathematically oriented programs with a minimum of programming knowledge. Requires an 8K computer memory. |
| 41-0081            | Basic Disk Assembler - Hi Density | Basically the same as the Basic Disk Assembler (41-0037) except that it uses a high density cartridge disk. Requires an 8K computer memory.   |
| 41-0084            | BASC-12 General Assembler (8K)    | Basically the same as the BASC-12 General Assembler (41-0001) except that it allows a larger number of user symbols and permits output via a magnetic tape cassette. Requires an 8K computer memory.  |

#### 8.4 DIAGNOSTIC PROGRAMS

| <u>Control No.</u> | <u>Title</u> | <u>Description</u>   |
|--------------------|--------------|--|
| 41-8001            | OPR-MRI Test | Serves as a go, no-go check for both classes of operate instructions |

| <u>Control No.</u> | <u>Title</u>               | <u>Description</u>   |
|--------------------|----------------------------|--|
|                    |                            | and all forms of single-word memory reference instruction using forward, reverse and indirect references.  |
| 41-8002            | XCT-TWI Test               | Serves as a go, no-go check of the execute instructions, all forms of two-word memory reference instructions, and combinations of single and two-word memory reference instructions with the execute instructions.               |
| 41-8004            | Memory Address Test        | Tests the addressing circuitry of the computer memory to verify that each word has a unique address. This is accomplished by setting the contents of a word equal to the address and checking the contents forward and backward. |
| 41-8005            | High/Low Speed Reader Test | Tests the high or low speed reader using a tape loop.  |
| 41-8006            | Low Speed Punch Test       | Tests the punched paper tape output of the Teletype for missing or extra levels.   |
| 41-8007            | High Speed Punch Test      | Tests the accuracy and registration of the high speed punch with the high speed reader.  |
| 41-8008            | High Speed Reader Test     | Tests the high-speed reader for accuracy and stopping ability with random length character blocks.   |
| 41-8009            | Cassette Diagnostic Test   | Tests input/output and control functions of the Single, Dual or Triple Magnetic Tape Cassette System using keyboard entry routines. Detection of errors is indicated by messages printed at the Teletype.                        |

| <u>Control No.</u> | <u>Title</u>   | <u>Description</u>   |
|--------------------|--|--|
| 41-8013            | Random ISZ-DSZ Test  | Tests the ISZ and DSZ memory reference instructions using random or fixed addresses.   |
| 41-8014            | Random ADJ-SBJ Test  | Tests the ADJ and SBJ memory reference instructions using random or fixed addresses.   |
| 41-8015            | Random LDJ-STJ Test  | Tests the LDJ and STJ memory reference instructions using random or fixed addresses.   |
| 41-8016            | Random JMP-JPS Test  | Tests the JMP and JPS memory reference instructions using random or fixed addresses.   |
| 41-8018            | Creepy Crawler   | Tests the storage capability of the computer memory by sequentially addressing each memory location.   |
| 41-8019            | Hardware Multiply/Divide Test                                  | Tests the hardware multiply and divide functions (ND812 serial numbers 0 -235).  |
| 41-8026            | Multiple Field Random TWJPS-TWJPS@ and Interrupt Test - 8K/16K | Tests random two-word jumps, indirect or direct, and the four level interrupt in any memory field of the 8K or 16K computer.   |
| 41-8028            | Multiple Field Random TWJPS-TWJPS@ and Interrupt Test - 12K    | Tests random two word jump, indirect or direct, and the four level interrupt in any memory field of the 12K computer.  |
| 41-8030            | PEC Diagnostic Test  | Tests the input/output and control function of the PEC 7 or 9-Track Magnetic Tape System using keyboard entry routines. The program also permits exchanging blocks of data between the computer memory and magnetic tape and provides a means of altering data in a specific area of memory. |
| 41-8041            | Worst Case Memory Pattern Test                                 | Tests the computer memory core stacks using worst case patterns.   |

| <u>Control No.</u> | <u>Title</u>                         | <u>Description</u>   |
|--------------------|--------------------------------------|--|
| 41-8042            | Literal Exerciser                    | Tests the literal, combined operate group 2, rotate and interrupt instructions using a program loop.   |
| 41-8043            | Diablo Disk Diagnostic               | Exercises the Diablo Disk Interface and the Diablo Disk Drive using a worst case serial bit pattern. The test parameters inputted via the Teletype include: drive selection, data field, starting sector, last sector, errors printed per sector, test disk, and last data word. |
| 41-8045            | Hardware Multiply/Divide Test        | Tests the hardware multiply and divide functions (ND812 serial numbers 236 and up).  |
| 41-8054            | Teletype Speed Test                  | Measures Teletype speed by averaging the time between print/punch flags for ten characters, eliminating the need for oscilloscope adjustment of interface print/punch circuitry.   |
| 41-8055            | Semiconductor Memory Test            | Completely tests memory and associated peripheral logic by three basic tests: 1) field addressing test to verify that a field requested can be addressed; 2) immediate load and read test to check for bit errors, and; 3) worst case pattern test.                              |
| 41-8057            | Semiconductor Memory Addressing Test | Fully exercises all memory addressing logic by three tests: 1) data test; 2) pattern test, and; 3) write/read test.  |



**APPENDIX A**  
**ND812 INSTRUCTION SET IN**  
**ALPHABETIC ORDER BY MNEMONIC**

| <u>Mnemonic</u> | <u>Octal Code</u> | <u>Operation</u>  | <u>Time (<math>\mu</math>S)</u> |
|-----------------|-------------------|---|---------------------------------|
| ADDL            | 22xx              | Add last 6 bits of instruction (xx) to J  | 2                               |
| ADJ             | 4400              | Add memory to J   | 4                               |
| ADR J           | 1122              | R + J to J  | 2                               |
| ADR K           | 1222              | R + K to K  | 2                               |
| ADS J           | 1124              | S + J to J  | 2                               |
| ADS K           | 1224              | S + K to K  | 2                               |
| AJK J           | 1120              | J + K to J  | 2                               |
| AJK K           | 1220              | J + K to K  | 2                               |
| AJK JK          | 1320              | J + K to J, K   | 2                               |
| ANDF            | 20xx              | Logical AND J with memory (forward only - no indirect)  | 4                               |
| AND J           | 1100              | Logical AND J, K to J   | 2                               |
| AND K           | 1200              | Logical AND J, K to K   | 2                               |
| AND JK          | 1300              | Logical AND J, K to J, K  | 2                               |
| AND L           | 21xx              | Logical AND last 6 bits of instruction (xx) with J <sub>6</sub> to J <sub>11</sub> , set J <sub>0</sub> to J <sub>5</sub> = 0 | 2                               |
| CCLF            | 0141              | Clear all cassette flags (TWIO)   | 5                               |
| CHSF            | 0101              | High Speed forward to EOT (TWIO)  | 5                               |
| CHSR            | 0121              | High speed reverse to BOT (TWIO)  | 5                               |
| CLR             | 1410              | Clear flag bit  | 2                               |
| CLR J           | 1510              | Clear J   | 2                               |
| CLR K           | 1610              | Clear K   | 2                               |
| CLR O           | 1450              | Clear overflow bit  | 2                               |
| CLR JK          | 1710              | Clear J and K   | 2                               |
| CMP             | 1420              | Complement flag bit   | 2                               |
| CMP J           | 1520              | Complement J  | 2                               |
| CMP K           | 1620              | Complement K  | 2                               |
| CMP O           | 1460              | Complement overflow bit   | 2                               |
| CMP JK          | 1720              | Complement J and K  | 2                               |
| CRDT            | 0144              | Transfer cassette buffer to J (TWIO)  | 5                               |
| CSBT            | 0130              | Skip if cassette at BOT (TWIO)  | 5                               |
| CSET            | 0110              | Skip if cassette at EOT (TWIO)  | 5                               |

| <u>Mnemonic</u> | <u>Octal Code</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u> |
|-----------------|-------------------|--|---------------------------------|
| CSFM            | 0104              | Skip on cassette filemark (TWIO)   | 5                               |
| CSLCT1          | 7601              | Set cassette 1 on-line   | 3                               |
| CSLCT2          | 7602              | Set cassette 2 on-line   | 3                               |
| CSLCT3          | 7604              | Set cassette 3 on-line   | 3                               |
| CSNE            | 0122              | Skip if no cassette errors (TWIO)  | 5                               |
| CSPF            | 0102              | Space cassette forward to filemark (TWIO)  | 5                               |
| CSRR            | 0142              | Skip if cassette read flag = 1 (TWIO)  | 5                               |
| CSTR            | 0124              | Skip if on-line cassette ready (TWIO)  | 5                               |
| CSWR            | 0152              | Skip if cassette write flag = 1 (TWIO)   | 5                               |
| CWFM            | 0151              | Write filemark on cassette (TWIO)  | 5                               |
| CWRT            | 0154              | Transfer J to cassette buffer (TWIO)   | 5                               |
| DIV             | 1001              | J, K/R to J; remainder in K  | 11                              |
| DSZ             | 3000              | Decrement memory; skip if = 0  | 4                               |
| EXJK            | 1374              | Exchange J and K   | 2.5                             |
| EXJR            | 1103              | Exchange J and R   | 2                               |
| EXJRKS          | 1303              | Exchange J and R; K and S  | 2                               |
| EXKS            | 1203              | Exchange K and S   | 2                               |
| HIF             | 7421              | Clear HS reader flag, read next character in HS reader buffer and set HS reader flag = 1 when done | 3                               |
| HIR             | 7422              | Clear HS reader flag and load J from HS reader buffer  | 3                               |
| HIS             | 7424              | Skip if HS reader flag = 1   | 3                               |
| HLP             | 7433              | HOL and HOP combined   | 3                               |
| HOL             | 7432              | Clear HS punch flag and load HS punch buffer from J  | 3                               |
| HOP             | 7431              | Clear HS punch flag and punch HS punch buffer  | 3                               |
| HOS             | 7434              | Skip if HS punch flag = 1  | 3                               |
| HRF             | 7423              | HIR and HIF combined   | 3                               |
| IDLE            | 1400              | One cycle delay  | 2                               |
| INC J           | 1504              | Increment J  | 2                               |
| INC K           | 1604              | Increment K  | 2                               |
| INC JK          | 1704              | Increment J and K  | 2                               |
| IOFF            | 1003              | Disable all interrupts   | 2                               |
| IONA            | 1006              | Enable class A and highest priority interrupts   | 2                               |
| IONB            | 1005              | Enable class B and highest priority interrupts   | 2                               |
| IONH            | 1004              | Enable highest priority interrupt  | 2                               |
| IONN            | 1007              | Enable all interrupts  | 2                               |
| ISZ             | 3400              | Increment memory; skip if = 0  | 4                               |
| JMP             | 6000              | Jump unconditionally   | 2                               |
| JPS             | 6400              | Jump to subroutine   | 4                               |
| LDJ             | 5000              | Load memory into J   | 4                               |
| LDJK            | 7721              | Load J from JPS; K from INT  | 3                               |
| LDREG           | 7720              | Load JPS from J; INT from K  | 3                               |

| <u>Mnemonic</u> | <u>Octal Code</u> | <u>Operation</u>                                    | <u>Time (<math>\mu</math>S)</u>      |
|-----------------|-------------------|---|--------------------------------------|
| LJFR            | 1102              | Load J from R                                       | 2                                    |
| LJKFRS          | 1302              | Load J from R; K from S                             | 2                                    |
| LJST            | 1011              | Load J from status bus                              | 2                                    |
| LJSW            | 1010              | Load J from Switch Register                         | 2                                    |
| LKFJ            | 1204              | Load K from J                                       | 2                                    |
| LKFS            | 1202              | Load K from S                                       | 2                                    |
| LRFJ            | 1101              | Load R from J                                       | 2                                    |
| LRSFJK          | 1301              | Load R from J; S from K                             | 2                                    |
| LSFK            | 1201              | Load S from K                                       | 2                                    |
| MPY             | 1000              | J x K to R, S                                       | 10.75                                |
| NADR J          | 1132              | -(R + J) to J                                       | 2                                    |
| NADR K          | 1232              | -(R + K) to K                                       | 2                                    |
| NADS J          | 1134              | -(S + J) to J                                       | 2                                    |
| NADS K          | 1234              | -(S + K) to K                                       | 2                                    |
| NAJK J          | 1130              | -(J + K) to J                                       | 2                                    |
| NAJK K          | 1230              | -(J + K) to K                                       | 2                                    |
| NAJK JK         | 1330              | -(J + K) to J, K                                    | 2                                    |
| NEG J           | 1524              | Negate J (complement and increment J)               | 2                                    |
| NEG K           | 1624              | Negate K (complement and increment K)               | 2                                    |
| NEG JK          | 1724              | Negate J and K (complement and increment J and K)   | 2                                    |
| NSBR J          | 1133              | -(R - J) to J                                       | 2                                    |
| NSBR K          | 1233              | -(R - K) to K                                       | 2                                    |
| NSBS J          | 1135              | -(S - J) to J                                       | 2                                    |
| NSBS K          | 1235              | -(S - K) to K                                       | 2                                    |
| NSJK J          | 1131              | -(J - K) to J                                       | 2                                    |
| NSJK K          | 1231              | -(J - K) to K                                       | 2                                    |
| NSJK JK         | 1331              | -(J - K) to J, K                                    | 2                                    |
| PIOF            | 1600              | Disable power interrupt                             | 2                                    |
| PION            | 1500              | Enable power interrupt                              | 2                                    |
| RFOV            | 1002              | Restore flag and overflow bits                      | 2                                    |
| RJIB            | 7722              | Restore JPS and INT field bits                      | 3                                    |
| ROTD J          | 1160              | Rotate data left in J (0 to 15 binary positions)    | $n \leq 8=2$<br>$n > 8=2+0.125(n-8)$ |
| ROTD K          | 1260              | Rotate data left in K (0 to 15 binary positions)    | $n \leq 8=2$<br>$n > 8=2+0.125(n-8)$ |
| ROTD JK         | 1360              | Rotate data left in J, K (0 to 15 binary positions) | $n \leq 8=2$<br>$n > 8=2+0.125(n-8)$ |
| SBJ             | 4000              | Subtract memory from J                              | 4                                    |
| SBR J           | 1123              | R - J to J  | 2                                    |
| SBR K           | 1223              | R - K to K  | 2                                    |
| SBS J           | 1125              | S - J to J  | 2                                    |
| SBS K           | 1225              | S - K to K  | 2                                    |
| SET             | 1430              | Set flag bit = 1 (clear & complement flag bit)      | 2                                    |
| SET J           | 1530              | Set J = 7777g (clear and complement J)              | 2                                    |

| <u>Mnemonic</u> | <u>Octal Code</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u>            |
|-----------------|-------------------|--|--|
| SET K           | 1630              | Set $K = 7777_8$ (clear and complement K)  | 2  |
| SET O           | 1470              | Set overflow bit = 1 (clear and complement overflow bit)   | 2  |
| SET JK          | 1730              | Set J and $K = 7777_8$ (clear and complement J and K)  | 2  |
| SFTZ J          | 1140              | Shift zeroes left into J (0 to 15 binary positions)  | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| SFTZ K          | 1240              | Shift zeroes left into K (0 to 15 binary positions)  | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| SFTZ JK         | 1340              | Shift zeroes left into J, K (0 to 15 binary positions)   | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| SIN J           | 1506              | Skip if $J < 0$  | 2  |
| SIN K           | 1606              | Skip if $K < 0$  | 2  |
| SIN JK          | 1706              | Skip if J and $K < 0$  | 2  |
| SIP J           | 1502              | Skip if $J > 0$  | 2  |
| SIP K           | 1602              | Skip if $K > 0$  | 2  |
| SIP JK          | 1702              | Skip if J and $K = 0$  | 2  |
| SIZ             | 1405              | Skip if flag bit = 0   | 2  |
| SIZ J           | 1505              | Skip if $J = 0$  | 2  |
| SIZ K           | 1605              | Skip if $K = 0$  | 2  |
| SIZ O           | 1445              | Skip if overflow bit = 0   | 2  |
| SIZ JK          | 1705              | Skip if J and $K = 0$  | 2  |
| SJK J           | 1121              | J - K to J   | 2  |
| SJK K           | 1221              | J - K to K   | 2  |
| SJK JK          | 1331              | J - K to J, K  | 2  |
| SKIP            | 1442              | Skip unconditionally   | 2  |
| SKPL            | 1440              | Skip on power low  | 2  |
| SMJ             | 2400              | Skip if $J \neq$ memory  | 4  |
| SNZ             | 1401              | Skip if flag bit $\neq 0$  | 2  |
| SNZ J           | 1501              | Skip if $J \neq 0$   | 2  |
| SNZ K           | 1601              | Skip if $K \neq 0$   | 2  |
| SNZ O           | 1441              | Skip if overflow bit $\neq 0$  | 2  |
| SNZ JK          | 1701              | Skip if J and $K \neq 0$   | 2  |
| STJ             | 5400              | Store J in memory  | 4  |
| STOP            | 0000              | Stop execution of program  | 2  |
| SUBL            | 23xx              | Subtract last 6 bits of instruction (xx) from J  | 2  |
| TCP             | 7413              | TOP and TOC combined   | 3  |
| TIF             | 7401              | Clear keyboard/reader flag, read next character into keyboard/reader buffer and set keyboard/reader flag = 1 when done | 3  |
| TIR             | 7402              | Clear keyboard/reader flag and load J from keyboard/reader buffer  | 3  |
| TIS             | 7404              | Skip if keyboard/reader flag = 1   | 3  |

| <u>Mnemonic</u> | <u>Octal Code</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u> |
|-----------------|-------------------|--|---------------------------------|
| TOC             | 7411              | Clear printer/punch flag   | 3                               |
| TOP             | 7412              | Clear printer/punch flag, load printer/punch buffer from J and print/punch | 3                               |
| TOS             | 7414              | Skip if printer/punch flag = 1   | 3                               |
| TRF             | 7403              | TIR and TIF combined   | 3                               |
| TWADJ           | 0440              | Add memory to J  | 6                               |
| TWADK           | 0450              | Add memory to K  | 6                               |
| TWDSZ           | 0300              | Decrement memory; skip if = 0  | 6                               |
| TWIO            | 0740              | Two-word I/O   | 5                               |
| TWISZ           | 0340              | Increment memory; skip if = 0  | 6                               |
| TWJMP           | 0600              | Jump unconditionally   | 4                               |
| TWJPS           | 0640              | Jump to subroutine   | 6                               |
| TWLDJ           | 0500              | Load memory from J   | 6                               |
| TWLDK           | 0510              | Load memory from K   | 6                               |
| TWSBJ           | 0400              | Subtract memory from J   | 6                               |
| TWSBK           | 0410              | Subtract memory from K   | 6                               |
| TWSMJ           | 0240              | Skip if J $\neq$ memory  | 6                               |
| TWSMK           | 0250              | Skip if K $\neq$ memory  | 6                               |
| TWSTJ           | 0540              | Store J in memory  | 6                               |
| TWSTK           | 0550              | Store K in memory  | 6                               |
| XCT             | 7000              | Execute instruction n  | $2 + t_n$                       |

**APPENDIX B**  
**ND812 INSTRUCTION SET IN**  
**NUMERICAL ORDER BY OCTAL CODE**

| <u>Octal Code</u> | <u>Mnemonic</u> | <u>Operation</u>                          | <u>Time (<math>\mu</math>S)</u> |
|-------------------|-----------------|---|---------------------------------|
| 0000              | STOP            | Stop execution of program                 | 2                               |
| 0101              | CHSF            | High speed forward to cassette EOT (TWIO) | 5                               |
| 0102              | CSPF            | Space forward to cassette filemark (TWIO) | 5                               |
| 0104              | CSFM            | Write filemark on cassette (TWIO)         | 5                               |
| 0110              | CSET            | Skip if cassette at EOT (TWIO)            | 5                               |
| 0121              | CHSR            | High speed forward to cassette BOT (TWIO) | 5                               |
| 0122              | CSNE            | Skip if no cassette errors (TWIO)         | 5                               |
| 0124              | CSTR            | Skip if on-line cassette ready (TWIO)     | 5                               |
| 0130              | CSBT            | Skip if cassette at BOT (TWIO)            | 5                               |
| 0141              | CCLF            | Clear all cassette flags (TWIO)           | 5                               |
| 0142              | CSRR            | Skip if cassette read flag = 1 (TWIO)     | 5                               |
| 0144              | CRDT            | Transfer cassette buffer to J (TWIO)      | 5                               |
| 0151              | CWFM            | Write filemark on cassette (TWIO)         | 5                               |
| 0152              | CSWR            | Skip if cassette write flag = 1 (TWIO)    | 5                               |
| 0154              | CWRT            | Transfer J to cassette buffer (TWIO)      | 5                               |
| 0240              | TWSMJ           | Skip if J $\neq$ memory                   | 6                               |
| 0250              | TWSMK           | Skip if K $\neq$ memory                   | 6                               |
| 0300              | TWDSZ           | Decrement memory; skip if = 0             | 6                               |
| 0340              | TWISZ           | Increment memory; skip if = 0             | 6                               |
| 0400              | TWSBJ           | Subtract memory from J                    | 6                               |
| 0410              | TWSBK           | Subtract memory from K                    | 6                               |
| 0440              | TWADJ           | Add memory to J                           | 6                               |
| 0450              | TWADK           | Add memory to K                           | 6                               |
| 0500              | TWLDJ           | Load memory into J                        | 6                               |
| 0510              | TWLDK           | Load memory into K                        | 6                               |
| 0540              | TWSTJ           | Store J in memory                         | 6                               |
| 0550              | TWSTK           | Store K in memory                         | 6                               |
| 0600              | TWJMP           | Jump unconditionally                      | 4                               |
| 0640              | TWJPS           | Jump to subroutine                        | 6                               |
| 0740              | TWIO            | Two word I/O                              | 5                               |
| 1000              | MPY             | J x K to R, S                             | 10.75                           |

| <u>Octal Code</u> | <u>Mnemonic</u> | <u>Operation</u>                                    | <u>Time (<math>\mu</math>S)</u>      |
|-------------------|-----------------|---|--------------------------------------|
| 1001              | DIV             | J, K/R to J; remainder in K                         | 11                                   |
| 1002              | RFOV            | Restore flag and overflow bits                      | 2                                    |
| 1003              | IOFF            | Disable all interrupts                              | 2                                    |
| 1004              | IONH            | Enable highest priority interrupt                   | 2                                    |
| 1005              | IONB            | Enable class B and highest priority interrupts      | 2                                    |
| 1006              | IONA            | Enable class A and highest priority interrupts      | 2                                    |
| 1007              | IONN            | Enable all interrupts                               | 2                                    |
| 1010              | LJSW            | Load J from Switch Register                         | 2                                    |
| 1011              | LJST            | Load J from Status Bus                              | 2                                    |
| 1100              | AND J           | Logical AND J, K to J                               | 2                                    |
| 1101              | LRFJ            | Load R from J                                       | 2                                    |
| 1102              | LJFR            | Load J from R                                       | 2                                    |
| 1103              | EXJR            | Exchange J and R                                    | 2                                    |
| 1120              | AJK J           | J + K to J  | 2                                    |
| 1121              | SJK J           | J - K to J  | 2                                    |
| 1122              | ADR J           | R + J to J  | 2                                    |
| 1123              | SBR J           | R - J to J  | 2                                    |
| 1124              | ADS J           | S + J to J  | 2                                    |
| 1125              | SBS J           | S - J to J  | 2                                    |
| 1130              | NAJK J          | -(J + K) to J                                       | 2                                    |
| 1131              | NSJK J          | -(J - K) to J                                       | 2                                    |
| 1132              | NADR J          | -(R + J) to J                                       | 2                                    |
| 1133              | NSBR J          | -(R - J) to J                                       | 2                                    |
| 1134              | NADS J          | -(S + J) to J                                       | 2                                    |
| 1135              | NSBS J          | -(S - J) to J                                       | 2                                    |
| 1140              | SFTZ J          | Shift zeroes left into J (0 to 15 binary positions) | $n \leq 8=2$<br>$n > 8=2+0.125(n-8)$ |
| 1160              | ROTD J          | Rotate data left in J (0 to 15 binary positions)    | $n \leq 8=2$<br>$n > 8=2+0.125(n-8)$ |
| 1200              | AND K           | Logical AND J, K to K                               | 2                                    |
| 1201              | LSFK            | Load S from K                                       | 2                                    |
| 1202              | LKFS            | Load K from S                                       | 2                                    |
| 1203              | EXKS            | Exchange K and S                                    | 2                                    |
| 1204              | LKFJ            | Load K from J                                       | 2                                    |
| 1220              | AJK K           | J + K to K  | 2                                    |
| 1221              | SJK K           | J - K to K  | 2                                    |
| 1222              | ADR K           | R + K to K  | 2                                    |
| 1223              | SBR K           | R - K to K  | 2                                    |
| 1224              | ADS K           | S + K to K  | 2                                    |
| 1225              | SBS K           | S - K to K  | 2                                    |
| 1230              | NAJK K          | -(J + K) to K                                       | 2                                    |
| 1231              | NSJK K          | -(J - K) to K                                       | 2                                    |
| 1232              | NADR K          | -(R + K) to K                                       | 2                                    |
| 1233              | NSBR K          | -(R - K) to K                                       | 2                                    |
| 1234              | NADS K          | -(S + K) to K                                       | 2                                    |
| 1235              | NSBS K          | -(S - K) to K                                       | 2                                    |

| <u>Octal Code</u> | <u>Mnemonic</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u>            |
|-------------------|-----------------|--|--|
| 1240              | SFTZ K          | Shift zeroes left into K (0 to 15 binary positions)      | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| 1260              | ROTD K          | Rotate data left in K (0 to 15 binary positions)         | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| 1300              | AND JK          | Logical AND J, K to J, K                                 | 2  |
| 1301              | LRSFJK          | Load R from J; S from K                                  | 2  |
| 1302              | LJKFRS          | Load J from R; K from S                                  | 2  |
| 1303              | EXJRKS          | Exchange J and R; K and S                                | 2  |
| 1320              | AJK JK          | J + K to J, K  | 2  |
| 1321              | SJK JK          | J - K to J, K  | 2  |
| 1330              | NAJK JK         | -(J + K) to J, K   | 2  |
| 1331              | NSJK JK         | -(J - K) to J, K   | 2  |
| 1340              | SFTZ JK         | Shift zeroes left into J, K (0 to 15 binary positions)   | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| 1360              | ROTD JK         | Rotate data left in J, K (0 to 15 binary positions)      | $n \leq 8 = 2$<br>$n > 8 = 2 + 0.125(n-8)$ |
| 1374              | EXJK            | Exchange J and K   | 2,5  |
| 1400              | IDLE            | One cycle delay  | 2  |
| 1401              | SNZ             | Skip if flag bit $\neq 0$                                | 2  |
| 1405              | SIZ             | Skip if flag bit = 0                                     | 2  |
| 1410              | CLR             | Clear flag bit   | 2  |
| 1420              | CMP             | Complement flag bit                                      | 2  |
| 1430              | SET             | Set flag bit = 1 (clear and complement flag bit)         | 2  |
| 1440              | SKPL            | Skip on power low  | 2  |
| 1441              | SNZ O           | Skip if overflow, bit $\neq 0$                           | 2  |
| 1442              | SKIP            | Skip unconditionally                                     | 2  |
| 1445              | SIZ O           | Skip if overflow bit = 0                                 | 2  |
| 1450              | CLR O           | Clear overflow bit                                       | 2  |
| 1460              | CMP O           | Complement overflow bit                                  | 2  |
| 1470              | SET O           | Set overflow bit = 1 (clear and complement overflow bit) | 2  |
| 1500              | PION            | Enable power interrupt                                   | 2  |
| 1501              | SNZ J           | Skip if J $\neq 0$                                       | 2  |
| 1502              | SIP J           | Skip if J > 0  | 2  |
| 1504              | INC J           | Increment J  | 2  |
| 1505              | SIZ J           | Skip if J = 0  | 2  |
| 1506              | SIN J           | Skip if J < 0  | 2  |
| 1510              | CLR J           | Clear J  | 2  |
| 1520              | CMP J           | Complement J   | 2  |
| 1524              | NEG J           | Negate J (complement and increment J)                    | 2  |
| 1530              | SET J           | Set J = $7777_8$ (clear and complement J)                | 2  |
| 1600              | PIOF            | Disable power interrupt                                  | 2  |
| 1601              | SNZ K           | Skip if K $\neq 0$                                       | 2  |
| 1602              | SIP K           | Skip if K > 0  | 2  |



| <u>Octal Code</u> | <u>Mnemonic</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u> |
|-------------------|-----------------|--|---------------------------------|
| 1604              | INC K           | Increment K  | 2                               |
| 1605              | SIZ K           | Skip if $K = 0$  | 2                               |
| 1606              | SIN K           | Skip if $K < 0$  | 2                               |
| 1610              | CLR K           | Clear K  | 2                               |
| 1620              | CMP K           | Complement K   | 2                               |
| 1624              | NEG K           | Negate K (complement and increment K)  | 2                               |
| 1630              | SET K           | Set $K = 7777_8$ (clear and complement K)  | 2                               |
| 1701              | SNZ JK          | Skip if $J$ and $K \neq 0$   | 2                               |
| 1702              | SIP JK          | Skip if $J$ and $K > 0$  | 2                               |
| 1704              | INC JK          | Increment J and K  | 2                               |
| 1705              | SIZ JK          | Skip if $J$ and $K = 0$  | 2                               |
| 1706              | SIN JK          | Skip if $J$ and $K < 0$  | 2                               |
| 1710              | CLR JK          | Clear J and K  | 2                               |
| 1720              | CMP JK          | Complement J and K   | 2                               |
| 1724              | NEG JK          | Negate J and K (complement and increment J and K)  | 2                               |
| 1730              | SET JK          | Set J and $K = 7777_8$ (clear and complement J and K)  | 2                               |
| 20xx              | ANDF            | Logical AND J with memory (forward only; no indirect)  | 4                               |
| 21xx              | ANDL            | Logical AND last 6 bits of instruction (xx) with $J_6$ to $J_{11}$ ; set $J_0$ to $J_5 = 0$                            | 2                               |
| 21xx              | ADDL            | Add last 6 bits of instruction (xx) to J   | 2                               |
| 23xx              | SUBL            | Subtract last 6 bits of instruction (xx) from J  | 2                               |
| 2400              | SMJ             | Skip if $J \neq$ memory  | 4                               |
| 3000              | DSZ             | Decrement memory; skip if = 0  | 4                               |
| 3400              | ISZ             | Increment memory; skip if = 0  | 4                               |
| 4000              | SBJ             | Subtract memory from J   | 4                               |
| 4400              | ADJ             | Add memory to J  | 4                               |
| 5000              | LDJ             | Load memory from J   | 4                               |
| 5400              | STJ             | Store J in memory  | 4                               |
| 6000              | JMP             | Jump unconditionally   | 2                               |
| 6400              | JPS             | Jump to subroutine   | 4                               |
| 7000              | XCT             | Execute instruction n  | $2 + t_n$                       |
| 7401              | TIF             | Clear keyboard/reader flag, read next character into keyboard/reader buffer and set keyboard/reader flag = 1 when done | 3                               |
| 7402              | TIR             | Clear keyboard/ready flag and load J from keyboard/reader buffer   | 3                               |
| 7403              | TRF             | TIR and TIF combined   | 3                               |
| 7404              | TIS             | Skip if keyboard/reader flag = 1   | 3                               |
| 7411              | TOC             | Clear printer/punch flag   | 3                               |

| <u>Octal Code</u> | <u>Mnemonic</u> | <u>Operation</u>   | <u>Time (<math>\mu</math>S)</u> |
|-------------------|-----------------|--|---------------------------------|
| 7412              | TOP             | Clear printer/punch flag, load printer/punch buffer from J and print/punch                           | 3                               |
| 7413              | TCP             | TOP and TOC combined   | 3                               |
| 7414              | TOS             | Skip if printer/punch flag = 1   | 3                               |
| 7421              | HIF             | Clear HS reader flag, read next character into HS reader buffer and set HS reader flag = 1 when done | 3                               |
| 7422              | HIR             | Clear HS reader flag and load J from HS reader buffer  | 3                               |
| 7423              | HRF             | HIR and HIF combined   | 3                               |
| 7424              | HIS             | Skip if HS reader flag = 1   | 3                               |
| 7431              | HOP             | Clear HS punch flag and punch HS punch buffer  | 3                               |
| 7432              | HOL             | Clear HS punch flag and load HS punch buffer from J  | 3                               |
| 7433              | HLP             | HOL and HOP combined   | 3                               |
| 7434              | HOS             | Skip if HS punch flag = 1  | 3                               |
| 7601              | CSLCT1          | Set cassette 1 on-line   | 3                               |
| 7602              | CSLCT2          | Set cassette 2 on-line   | 3                               |
| 7604              | CSLCT3          | Set cassette 3 on-line   | 3                               |
| 7720              | LDREG           | Load JPS from J; INT from K  | 3                               |
| 7721              | LDJK            | Load J from JPS; K from INT  | 3                               |
| 7722              | RJIB            | Restore JPS and INT field bits   | 3                               |

## APPENDIX C FLOW CHARTING SYMBOLS

The American Standards Institute has adopted the following symbols for flow diagram use.

### A. Input/Output



This symbol represents the basic functions of entering data into the computer or outputting the data. This is a high level symbol, because individual devices have unique symbols.

### B. Punched Tape



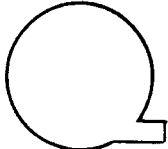
This symbol represents an I/O function which uses devices. It can represent the reading in of data from punched tape through reader or the dumping data by punching tape.

### C. On-line Storage



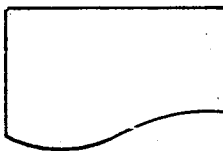
The on-line storage symbol indicates the use of a mass storage unit such as disk file or drum. The symbol may indicate the storage and/or retrieval of data. The data is directly accessible to the computer.

### D. Magnetic Tape



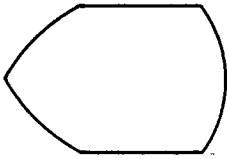
This symbol indicates the use of magnetic tape as the I/O medium.

### E. Document



The document symbol denotes the use of a line or page printer as an output device.

F. Display Output



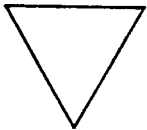
This symbol represents the video display of computer data.

G. Punched Card



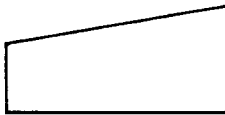
This symbol is used whenever the input and/or output data will be on a punched card.

H. Off Line Storage



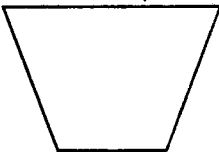
The use of this symbol refers to data storage which is not directly accessible by the computer.

I. Manual Input



The manual input symbol represents the use of a keyboard device, such as teletype, to enter data into the computer.

J. Manual Operation



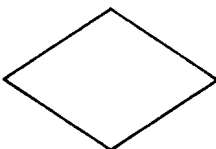
This symbol denotes data handling not involving the computer, or throwing a switch on the computer, etc.

K. Processing



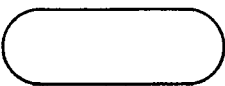
The processing symbol is used for several functions. It may, at the lowest level, represent one instruction; at a higher level, it represents all instructions necessary to perform a given task.

L. Decision



The decision symbol marks the branch point in a program. Therefore, there are two or more possible exits from the symbol.

M. Terminal



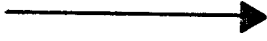
The terminal marks the beginning of and all possible terminations to the program.

N. Communication Link



This symbol indicates the transferral of data between various locations. Phone lines and radio networks are common examples.

O. Flow Direction



P. Connector



The various symbols are connected by lines; convention dictates that flow will normally be from top to bottom and from left to right.

The connector symbol is used to identify common points in the flow paths when connecting lines either cannot be drawn or would be confusing.

## APPENDIX D POWERS OF TWO

| 2 <sup>n</sup>  | n  | 2 <sup>-n</sup>                                       |
|-----------------|----|---|
| 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             |
| 2 147 583 648   | 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 |

# APPENDIX E OCTAL-TO-DECIMAL CONVERSION TABLE

|  | 0 1 2 3 4 5 6 7 |      |      |      |      |      |      |      | 0 1 2 3 4 5 6 7 |      |      |      |      |      |      |      |      |
|--|-----------------|------|------|------|------|------|------|------|-----------------|------|------|------|------|------|------|------|------|
| 0000   0000<br>to   to<br>0777   0511<br>(Octal)   (Decimal)   | 0000            | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0400            | 0256 | 0257 | 0258 | 0259 | 0260 | 0261 | 0262 | 0263 |
|  | 0010            | 0008 | 0009 | 0010 | 0011 | 0012 | 0013 | 0014 | 0410            | 0264 | 0265 | 0266 | 0267 | 0268 | 0269 | 0270 | 0271 |
|  | 0020            | 0016 | 0017 | 0018 | 0019 | 0020 | 0021 | 0022 | 0420            | 0272 | 0273 | 0274 | 0275 | 0276 | 0277 | 0278 | 0279 |
|  | 0030            | 0024 | 0025 | 0026 | 0027 | 0028 | 0029 | 0030 | 0430            | 0280 | 0281 | 0282 | 0283 | 0284 | 0285 | 0286 | 0287 |
|  | 0040            | 0032 | 0033 | 0034 | 0035 | 0036 | 0037 | 0038 | 0440            | 0288 | 0289 | 0290 | 0291 | 0292 | 0293 | 0294 | 0295 |
|  | 0050            | 0040 | 0041 | 0042 | 0043 | 0044 | 0045 | 0046 | 0450            | 0296 | 0297 | 0298 | 0299 | 0300 | 0301 | 0302 | 0303 |
|  | 0060            | 0048 | 0049 | 0050 | 0051 | 0052 | 0053 | 0054 | 0460            | 0304 | 0305 | 0306 | 0307 | 0308 | 0309 | 0310 | 0311 |
|  | 0070            | 0056 | 0057 | 0058 | 0059 | 0060 | 0061 | 0062 | 0470            | 0312 | 0313 | 0314 | 0315 | 0316 | 0317 | 0318 | 0319 |
| Octal Decimal<br>10000 - 4096<br>20000 - 8192<br>30000 - 12288<br>40000 - 16384<br>50000 - 20480<br>60000 - 24576<br>70000 - 28672 | 0100            | 0064 | 0065 | 0066 | 0067 | 0068 | 0069 | 0070 | 0500            | 0320 | 0321 | 0322 | 0323 | 0324 | 0325 | 0326 | 0327 |
|  | 0110            | 0072 | 0073 | 0074 | 0075 | 0076 | 0077 | 0078 | 0510            | 0328 | 0329 | 0330 | 0331 | 0332 | 0333 | 0334 | 0335 |
|  | 0120            | 0080 | 0081 | 0082 | 0083 | 0084 | 0085 | 0086 | 0520            | 0336 | 0337 | 0338 | 0339 | 0340 | 0341 | 0342 | 0343 |
|  | 0130            | 0088 | 0089 | 0090 | 0091 | 0092 | 0093 | 0094 | 0530            | 0344 | 0345 | 0346 | 0347 | 0348 | 0349 | 0350 | 0351 |
|  | 0140            | 0096 | 0097 | 0098 | 0099 | 0100 | 0101 | 0102 | 0540            | 0352 | 0353 | 0354 | 0355 | 0356 | 0357 | 0358 | 0359 |
|  | 0150            | 0104 | 0105 | 0106 | 0107 | 0108 | 0109 | 0110 | 0550            | 0360 | 0361 | 0362 | 0363 | 0364 | 0365 | 0366 | 0367 |
|  | 0160            | 0112 | 0113 | 0114 | 0115 | 0116 | 0117 | 0118 | 0560            | 0368 | 0369 | 0370 | 0371 | 0372 | 0373 | 0374 | 0375 |
|  | 0170            | 0120 | 0121 | 0122 | 0123 | 0124 | 0125 | 0126 | 0570            | 0376 | 0377 | 0378 | 0379 | 0380 | 0381 | 0382 | 0383 |
|  | 0200            | 0128 | 0129 | 0130 | 0131 | 0132 | 0133 | 0134 | 0600            | 0384 | 0385 | 0386 | 0387 | 0388 | 0389 | 0390 | 0391 |
|  | 0210            | 0136 | 0137 | 0138 | 0139 | 0140 | 0141 | 0142 | 0610            | 0392 | 0393 | 0394 | 0395 | 0396 | 0397 | 0398 | 0399 |
|  | 0220            | 0144 | 0145 | 0146 | 0147 | 0148 | 0149 | 0150 | 0620            | 0400 | 0401 | 0402 | 0403 | 0404 | 0405 | 0406 | 0407 |
|  | 0230            | 0152 | 0153 | 0154 | 0155 | 0156 | 0157 | 0158 | 0630            | 0408 | 0409 | 0410 | 0411 | 0412 | 0413 | 0414 | 0415 |
|  | 0240            | 0160 | 0161 | 0162 | 0163 | 0164 | 0165 | 0166 | 0640            | 0416 | 0417 | 0418 | 0419 | 0420 | 0421 | 0422 | 0423 |
|  | 0250            | 0168 | 0169 | 0170 | 0171 | 0172 | 0173 | 0174 | 0650            | 0424 | 0425 | 0426 | 0427 | 0428 | 0429 | 0430 | 0431 |
|  | 0260            | 0176 | 0177 | 0178 | 0179 | 0180 | 0181 | 0182 | 0660            | 0432 | 0433 | 0434 | 0435 | 0436 | 0437 | 0438 | 0439 |
|  | 0270            | 0184 | 0185 | 0186 | 0187 | 0188 | 0189 | 0190 | 0670            | 0440 | 0441 | 0442 | 0443 | 0444 | 0445 | 0446 | 0447 |
|  | 0300            | 0192 | 0193 | 0194 | 0195 | 0196 | 0197 | 0198 | 0700            | 0448 | 0449 | 0450 | 0451 | 0452 | 0453 | 0454 | 0455 |
|  | 0310            | 0200 | 0201 | 0202 | 0203 | 0204 | 0205 | 0206 | 0710            | 0456 | 0457 | 0458 | 0459 | 0460 | 0461 | 0462 | 0463 |
|  | 0320            | 0208 | 0209 | 0210 | 0211 | 0212 | 0213 | 0214 | 0720            | 0464 | 0465 | 0466 | 0467 | 0468 | 0469 | 0470 | 0471 |
|  | 0330            | 0216 | 0217 | 0218 | 0219 | 0220 | 0221 | 0222 | 0730            | 0472 | 0473 | 0474 | 0475 | 0476 | 0477 | 0478 | 0479 |
|  | 0340            | 0224 | 0225 | 0226 | 0227 | 0228 | 0229 | 0230 | 0740            | 0480 | 0481 | 0482 | 0483 | 0484 | 0485 | 0486 | 0487 |
|  | 0350            | 0232 | 0233 | 0234 | 0235 | 0236 | 0237 | 0238 | 0750            | 0488 | 0489 | 0490 | 0491 | 0492 | 0493 | 0494 | 0495 |
|  | 0360            | 0240 | 0241 | 0242 | 0243 | 0244 | 0245 | 0246 | 0760            | 0496 | 0497 | 0498 | 0499 | 0500 | 0501 | 0502 | 0503 |
|  | 0370            | 0248 | 0249 | 0250 | 0251 | 0252 | 0253 | 0254 | 0770            | 0504 | 0505 | 0506 | 0507 | 0508 | 0509 | 0510 | 0511 |
|  | 1000            | 0512 | 0513 | 0514 | 0515 | 0516 | 0517 | 0518 | 1400            | 0768 | 0769 | 0770 | 0771 | 0772 | 0773 | 0774 | 0775 |
| 1000   0512<br>to   to<br>1777   1023<br>(Octal)   (Decimal)   | 1010            | 0520 | 0521 | 0522 | 0523 | 0524 | 0525 | 0526 | 1410            | 0776 | 0777 | 0778 | 0779 | 0780 | 0781 | 0782 | 0783 |
|  | 1020            | 0528 | 0529 | 0530 | 0531 | 0532 | 0533 | 0534 | 1420            | 0784 | 0785 | 0786 | 0787 | 0788 | 0789 | 0790 | 0791 |
|  | 1030            | 0536 | 0537 | 0538 | 0539 | 0540 | 0541 | 0542 | 1430            | 0792 | 0793 | 0794 | 0795 | 0796 | 0797 | 0798 | 0799 |
|  | 1040            | 0544 | 0545 | 0546 | 0547 | 0548 | 0549 | 0550 | 1440            | 0800 | 0801 | 0802 | 0803 | 0804 | 0805 | 0806 | 0807 |
|  | 1050            | 0552 | 0553 | 0554 | 0555 | 0556 | 0557 | 0558 | 1450            | 0808 | 0809 | 0810 | 0811 | 0812 | 0813 | 0814 | 0815 |
|  | 1060            | 0560 | 0561 | 0562 | 0563 | 0564 | 0565 | 0566 | 1460            | 0816 | 0817 | 0818 | 0819 | 0820 | 0821 | 0822 | 0823 |
|  | 1070            | 0568 | 0569 | 0570 | 0571 | 0572 | 0573 | 0574 | 1470            | 0824 | 0825 | 0826 | 0827 | 0828 | 0829 | 0830 | 0831 |
|  | 1100            | 0576 | 0577 | 0578 | 0579 | 0580 | 0581 | 0582 | 1500            | 0832 | 0833 | 0834 | 0835 | 0836 | 0837 | 0838 | 0839 |
|  | 1110            | 0584 | 0585 | 0586 | 0587 | 0588 | 0589 | 0590 | 1510            | 0840 | 0841 | 0842 | 0843 | 0844 | 0845 | 0846 | 0847 |
|  | 1120            | 0592 | 0593 | 0594 | 0595 | 0596 | 0597 | 0598 | 1520            | 0848 | 0849 | 0850 | 0851 | 0852 | 0853 | 0854 | 0855 |
|  | 1130            | 0600 | 0601 | 0602 | 0603 | 0604 | 0605 | 0606 | 1530            | 0856 | 0857 | 0858 | 0859 | 0860 | 0861 | 0862 | 0863 |
|  | 1140            | 0608 | 0609 | 0610 | 0611 | 0612 | 0613 | 0614 | 1540            | 0864 | 0865 | 0866 | 0867 | 0868 | 0869 | 0870 | 0871 |
|  | 1150            | 0616 | 0617 | 0618 | 0619 | 0620 | 0621 | 0622 | 1550            | 0872 | 0873 | 0874 | 0875 | 0876 | 0877 | 0878 | 0879 |
|  | 1160            | 0624 | 0625 | 0626 | 0627 | 0628 | 0629 | 0630 | 1560            | 0880 | 0881 | 0882 | 0883 | 0884 | 0885 | 0886 | 0887 |
|  | 1170            | 0632 | 0633 | 0634 | 0635 | 0636 | 0637 | 0638 | 1570            | 0888 | 0889 | 0890 | 0891 | 0892 | 0893 | 0894 | 0895 |
|  | 1200            | 0640 | 0641 | 0642 | 0643 | 0644 | 0645 | 0646 | 1600            | 0896 | 0897 | 0898 | 0899 | 0900 | 0901 | 0902 | 0903 |
|  | 1210            | 0648 | 0649 | 0650 | 0651 | 0652 | 0653 | 0654 | 1610            | 0904 | 0905 | 0906 | 0907 | 0908 | 0909 | 0910 | 0911 |
|  | 1220            | 0656 | 0657 | 0658 | 0659 | 0660 | 0661 | 0662 | 1620            | 0912 | 0913 | 0914 | 0915 | 0916 | 0917 | 0918 | 0919 |
|  | 1230            | 0664 | 0665 | 0666 | 0667 | 0668 | 0669 | 0670 | 1630            | 0920 | 0921 | 0922 | 0923 | 0924 | 0925 | 0926 | 0927 |
|  | 1240            | 0672 | 0673 | 0674 | 0675 | 0676 | 0677 | 0678 | 1640            | 0928 | 0929 | 0930 | 0931 | 0932 | 0933 | 0934 | 0935 |
|  | 1250            | 0680 | 0681 | 0682 | 0683 | 0684 | 0685 | 0686 | 1650            | 0936 | 0937 | 0938 | 0939 | 0940 | 0941 | 0942 | 0943 |
|  | 1260            | 0688 | 0689 | 0690 | 0691 | 0692 | 0693 | 0694 | 1660            | 0944 | 0945 | 0946 | 0947 | 0948 | 0949 | 0950 | 0951 |
|  | 1270            | 0696 | 0697 | 0698 | 0699 | 0700 | 0701 | 0702 | 1670            | 0952 | 0953 | 0954 | 0955 | 0956 | 0957 | 0958 | 0959 |
|  | 1300            | 0704 | 0705 | 0706 | 0707 | 0708 | 0709 | 0710 | 1700            | 0960 | 0961 | 0962 | 0963 | 0964 | 0965 | 0966 | 0967 |
|  | 1310            | 0712 | 0713 | 0714 | 0715 | 0716 | 0717 | 0718 | 1710            | 0968 | 0969 | 0970 | 0971 | 0972 | 0973 | 0974 | 0975 |
|  | 1320            | 0720 | 0721 | 0722 | 0723 | 0724 | 0725 | 0726 | 1720            | 0976 | 0977 | 0978 | 0979 | 0980 | 0981 | 0982 | 0983 |
|  | 1330            | 0728 | 0729 | 0730 | 0731 | 0732 | 0733 | 0734 | 1730            | 0984 | 0985 | 0986 | 0987 | 0988 | 0989 | 0990 | 0991 |
|  | 1340            | 0736 | 0737 | 0738 | 0739 | 0740 | 0741 | 0742 | 1740            | 0992 | 0993 | 0994 | 0995 | 0996 | 0997 | 0998 | 0999 |
|  | 1350            | 0744 | 0745 | 0746 | 0747 | 0748 | 0749 | 0750 | 1750            | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 |
|  | 1360            | 0752 | 0753 | 0754 | 0755 | 0756 | 0757 | 0758 | 1760            | 1008 | 1009 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 |
|  | 1370            | 0760 | 0761 | 0762 | 0763 | 0764 | 0765 | 0766 | 1770            | 1016 | 1017 | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 |

|      | 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)



|                               |                                 | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    |      |      | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    |
|-------------------------------|---------------------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 4000<br>to<br>4777<br>(Octal) | 2048<br>to<br>2559<br>(Decimal) | 4000 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 | 4400 | 2304 | 2305 | 2306 | 2307 | 2308 | 2309 | 2310 | 2311 |
|                               |                                 | 4010 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 | 4410 | 2312 | 2313 | 2314 | 2315 | 2316 | 2317 | 2318 | 2319 |
|                               |                                 | 4020 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 | 4420 | 2320 | 2321 | 2322 | 2323 | 2324 | 2325 | 2326 | 2327 |
|                               |                                 | 4030 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 | 4430 | 2328 | 2329 | 2330 | 2331 | 2332 | 2333 | 2334 | 2335 |
|                               |                                 | 4040 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 | 4440 | 2336 | 2337 | 2338 | 2339 | 2340 | 2341 | 2342 | 2343 |
|                               |                                 | 4050 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 | 4450 | 2344 | 2345 | 2346 | 2347 | 2348 | 2349 | 2350 | 2351 |
|                               |                                 | 4060 | 2096 | 2097 | 2098 | 2099 | 2100 | 2101 | 2102 | 2103 | 4460 | 2352 | 2353 | 2354 | 2355 | 2356 | 2357 | 2358 | 2359 |
|                               |                                 | 4070 | 2104 | 2105 | 2106 | 2107 | 2108 | 2109 | 2110 | 2111 | 4470 | 2360 | 2361 | 2362 | 2363 | 2364 | 2365 | 2366 | 2367 |
|                               |                                 | 4100 | 2112 | 2113 | 2114 | 2115 | 2116 | 2117 | 2118 | 2119 | 4500 | 2368 | 2369 | 2370 | 2371 | 2372 | 2373 | 2374 | 2375 |
|                               |                                 | 4110 | 2120 | 2121 | 2122 | 2123 | 2124 | 2125 | 2126 | 2127 | 4510 | 2376 | 2377 | 2378 | 2379 | 2380 | 2381 | 2382 | 2383 |
| 4120                          | 2128                            | 2129 | 2130 | 2131 | 2132 | 2133 | 2134 | 2135 | 4520 | 2384 | 2385 | 2386 | 2387 | 2388 | 2389 | 2390 | 2391 |      |      |
| 4130                          | 2136                            | 2137 | 2138 | 2139 | 2140 | 2141 | 2142 | 2143 | 4530 | 2392 | 2393 | 2394 | 2395 | 2396 | 2397 | 2398 | 2399 |      |      |
| 4140                          | 2144                            | 2145 | 2146 | 2147 | 2148 | 2149 | 2150 | 2151 | 4540 | 2400 | 2401 | 2402 | 2403 | 2404 | 2405 | 2406 | 2407 |      |      |
| 4150                          | 2152                            | 2153 | 2154 | 2155 | 2156 | 2157 | 2158 | 2159 | 4550 | 2408 | 2409 | 2410 | 2411 | 2412 | 2413 | 2414 | 2415 |      |      |
| 4160                          | 2160                            | 2161 | 2162 | 2163 | 2164 | 2165 | 2166 | 2167 | 4560 | 2416 | 2417 | 2418 | 2419 | 2420 | 2421 | 2422 | 2423 |      |      |
| 4170                          | 2168                            | 2169 | 2170 | 2171 | 2172 | 2173 | 2174 | 2175 | 4570 | 2424 | 2425 | 2426 | 2427 | 2428 | 2429 | 2430 | 2431 |      |      |
| 4200                          | 2176                            | 2177 | 2178 | 2179 | 2180 | 2181 | 2182 | 2183 | 4600 | 2432 | 2433 | 2434 | 2435 | 2436 | 2437 | 2438 | 2439 |      |      |
| 4210                          | 2184                            | 2185 | 2186 | 2187 | 2188 | 2189 | 2190 | 2191 | 4610 | 2440 | 2441 | 2442 | 2443 | 2444 | 2445 | 2446 | 2447 |      |      |
| 4220                          | 2192                            | 2193 | 2194 | 2195 | 2196 | 2197 | 2198 | 2199 | 4620 | 2448 | 2449 | 2450 | 2451 | 2452 | 2453 | 2454 | 2455 |      |      |
| 4230                          | 2200                            | 2201 | 2202 | 2203 | 2204 | 2205 | 2206 | 2207 | 4630 | 2456 | 2457 | 2458 | 2459 | 2460 | 2461 | 2462 | 2463 |      |      |
| 4240                          | 2208                            | 2209 | 2210 | 2211 | 2212 | 2213 | 2214 | 2215 | 4640 | 2464 | 2465 | 2466 | 2467 | 2468 | 2469 | 2470 | 2471 |      |      |
| 4250                          | 2216                            | 2217 | 2218 | 2219 | 2220 | 2221 | 2222 | 2223 | 4650 | 2472 | 2473 | 2474 | 2475 | 2476 | 2477 | 2478 | 2479 |      |      |
| 4260                          | 2224                            | 2225 | 2226 | 2227 | 2228 | 2229 | 2230 | 2231 | 4660 | 2480 | 2481 | 2482 | 2483 | 2484 | 2485 | 2486 | 2487 |      |      |
| 4270                          | 2232                            | 2233 | 2234 | 2235 | 2236 | 2237 | 2238 | 2239 | 4670 | 2488 | 2489 | 2490 | 2491 | 2492 | 2493 | 2494 | 2495 |      |      |
| 4300                          | 2240                            | 2241 | 2242 | 2243 | 2244 | 2245 | 2246 | 2247 | 4700 | 2496 | 2497 | 2498 | 2499 | 2500 | 2501 | 2502 | 2503 |      |      |
| 4310                          | 2248                            | 2249 | 2250 | 2251 | 2252 | 2253 | 2254 | 2255 | 4710 | 2504 | 2505 | 2506 | 2507 | 2508 | 2509 | 2510 | 2511 |      |      |
| 4320                          | 2256                            | 2257 | 2258 | 2259 | 2260 | 2261 | 2262 | 2263 | 4720 | 2512 | 2513 | 2514 | 2515 | 2516 | 2517 | 2518 | 2519 |      |      |
| 4330                          | 2264                            | 2265 | 2266 | 2267 | 2268 | 2269 | 2270 | 2271 | 4730 | 2520 | 2521 | 2522 | 2523 | 2524 | 2525 | 2526 | 2527 |      |      |
| 4340                          | 2272                            | 2273 | 2274 | 2275 | 2276 | 2277 | 2278 | 2279 | 4740 | 2528 | 2529 | 2530 | 2531 | 2532 | 2533 | 2534 | 2535 |      |      |
| 4350                          | 2280                            | 2281 | 2282 | 2283 | 2284 | 2285 | 2286 | 2287 | 4750 | 2536 | 2537 | 2538 | 2539 | 2540 | 2541 | 2542 | 2543 |      |      |
| 4360                          | 2288                            | 2289 | 2290 | 2291 | 2292 | 2293 | 2294 | 2295 | 4760 | 2544 | 2545 | 2546 | 2547 | 2548 | 2549 | 2550 | 2551 |      |      |
| 4370                          | 2296                            | 2297 | 2298 | 2299 | 2300 | 2301 | 2302 | 2303 | 4770 | 2552 | 2553 | 2554 | 2555 | 2556 | 2557 | 2558 | 2559 |      |      |
|                               |                                 | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    |      |      | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    |
| 5000<br>to<br>5777<br>(Octal) | 2560<br>to<br>3071<br>(Decimal) | 5000 | 2560 | 2561 | 2562 | 2563 | 2564 | 2565 | 2566 | 2567 | 5400 | 2816 | 2817 | 2818 | 2819 | 2820 | 2821 | 2822 | 2823 |
|                               |                                 | 5010 | 2568 | 2569 | 2570 | 2571 | 2572 | 2573 | 2574 | 2575 | 5410 | 2824 | 2825 | 2826 | 2827 | 2828 | 2829 | 2830 | 2831 |
|                               |                                 | 5020 | 2576 | 2577 | 2578 | 2579 | 2580 | 2581 | 2582 | 2583 | 5420 | 2832 | 2833 | 2834 | 2835 | 2836 | 2837 | 2838 | 2839 |
|                               |                                 | 5030 | 2584 | 2585 | 2586 | 2587 | 2588 | 2589 | 2590 | 2591 | 5430 | 2840 | 2841 | 2842 | 2843 | 2844 | 2845 | 2846 | 2847 |
|                               |                                 | 5040 | 2592 | 2593 | 2594 | 2595 | 2596 | 2597 | 2598 | 2599 | 5440 | 2848 | 2849 | 2850 | 2851 | 2852 | 2853 | 2854 | 2855 |
|                               |                                 | 5050 | 2600 | 2601 | 2602 | 2603 | 2604 | 2605 | 2606 | 2607 | 5450 | 2856 | 2857 | 2858 | 2859 | 2860 | 2861 | 2862 | 2863 |
|                               |                                 | 5060 | 2608 | 2609 | 2610 | 2611 | 2612 | 2613 | 2614 | 2615 | 5460 | 2864 | 2865 | 2866 | 2867 | 2868 | 2869 | 2870 | 2871 |
|                               |                                 | 5070 | 2616 | 2617 | 2618 | 2619 | 2620 | 2621 | 2622 | 2623 | 5470 | 2872 | 2873 | 2874 | 2875 | 2876 | 2877 | 2878 | 2879 |
|                               |                                 | 5100 | 2624 | 2625 | 2626 | 2627 | 2628 | 2629 | 2630 | 2631 | 5500 | 2880 | 2881 | 2882 | 2883 | 2884 | 2885 | 2886 | 2887 |
|                               |                                 | 5110 | 2632 | 2633 | 2634 | 2635 | 2636 | 2637 | 2638 | 2639 | 5510 | 2888 | 2889 | 2890 | 2891 | 2892 | 2893 | 2894 | 2895 |
| 5120                          | 2640                            | 2641 | 2642 | 2643 | 2644 | 2645 | 2646 | 2647 | 5520 | 2896 | 2897 | 2898 | 2899 | 2900 | 2901 | 2902 | 2903 |      |      |
| 5130                          | 2648                            | 2649 | 2650 | 2651 | 2652 | 2653 | 2654 | 2655 | 5530 | 2904 | 2905 | 2906 | 2907 | 2908 | 2909 | 2910 | 2911 |      |      |
| 5140                          | 2656                            | 2657 | 2658 | 2659 | 2660 | 2661 | 2662 | 2663 | 5540 | 2912 | 2913 | 2914 | 2915 | 2916 | 2917 | 2918 | 2919 |      |      |
| 5150                          | 2664                            | 2665 | 2666 | 2667 | 2668 | 2669 | 2670 | 2671 | 5550 | 2920 | 2921 | 2922 | 2923 | 2924 | 2925 | 2926 | 2927 |      |      |
| 5160                          | 2672                            | 2673 | 2674 | 2675 | 2676 | 2677 | 2678 | 2679 | 5560 | 2928 | 2929 | 2930 | 2931 | 2932 | 2933 | 2934 | 2935 |      |      |
| 5170                          | 2680                            | 2681 | 2682 | 2683 | 2684 | 2685 | 2686 | 2687 | 5570 | 2936 | 2937 | 2938 | 2939 | 2940 | 2941 | 2942 | 2943 |      |      |
| 5200                          | 2688                            | 2689 | 2690 | 2691 | 2692 | 2693 | 2694 | 2695 | 5600 | 2944 | 2945 | 2946 | 2947 | 2948 | 2949 | 2950 | 2951 |      |      |
| 5210                          | 2696                            | 2697 | 2698 | 2699 | 2700 | 2701 | 2702 | 2703 | 5610 | 2952 | 2953 | 2954 | 2955 | 2956 | 2957 | 2958 | 2959 |      |      |
| 5220                          | 2704                            | 2705 | 2706 | 2707 | 2708 | 2709 | 2710 | 2711 | 5620 | 2960 | 2961 | 2962 | 2963 | 2964 | 2965 | 2966 | 2967 |      |      |
| 5230                          | 2712                            | 2713 | 2714 | 2715 | 2716 | 2717 | 2718 | 2719 | 5630 | 2968 | 2969 | 2970 | 2971 | 2972 | 2973 | 2974 | 2975 |      |      |
| 5240                          | 2720                            | 2721 | 2722 | 2723 | 2724 | 2725 | 2726 | 2727 | 5640 | 2976 | 2977 | 2978 | 2979 | 2980 | 2981 | 2982 | 2983 |      |      |
| 5250                          | 2728                            | 2729 | 2730 | 2731 | 2732 | 2733 | 2734 | 2735 | 5650 | 2984 | 2985 | 2986 | 2987 | 2988 | 2989 | 2990 | 2991 |      |      |
| 5260                          | 2736                            | 2737 | 2738 | 2739 | 2740 | 2741 | 2742 | 2743 | 5660 | 2992 | 2993 | 2994 | 2995 | 2996 | 2997 | 2998 | 2999 |      |      |
| 5270                          | 2744                            | 2745 | 2746 | 2747 | 2748 | 2749 | 2750 | 2751 | 5670 | 3000 | 3001 | 3002 | 3003 | 3004 | 3005 | 3006 | 3007 |      |      |
| 5300                          | 2752                            | 2753 | 2754 | 2755 | 2756 | 2757 | 2758 | 2759 | 5700 | 3008 | 3009 | 3010 | 3011 | 3012 | 3013 | 3014 | 3015 |      |      |
| 5310                          | 2760                            | 2761 | 2762 | 2763 | 2764 | 2765 | 2766 | 2767 | 5710 | 3016 | 3017 | 3018 | 3019 | 3020 | 3021 | 3022 | 3023 |      |      |
| 5320                          | 2768                            | 2769 | 2770 | 2771 | 2772 | 2773 | 2774 | 2775 | 5720 | 3024 | 3025 | 3026 | 3027 | 3028 | 3029 | 3030 | 3031 |      |      |
| 5330                          | 2776                            | 2777 | 2778 | 2779 | 2780 | 2781 | 2782 | 2783 | 5730 | 3032 | 3033 | 3034 | 3035 | 3036 | 3037 | 3038 | 3039 |      |      |
| 5340                          | 2784                            | 2785 | 2786 | 2787 | 2788 | 2789 | 2790 | 2791 | 5740 | 3040 | 3041 | 3042 | 3043 | 3044 | 3045 | 3046 | 3047 |      |      |
| 5350                          | 2792                            | 2793 | 2794 | 2795 | 2796 | 2797 | 2798 | 2799 | 5750 | 3048 | 3049 | 3050 | 3051 | 3052 | 3053 | 3054 | 3055 |      |      |
| 5360                          | 2800                            | 2801 | 2802 | 2803 | 2804 | 2805 | 2806 | 2807 | 5760 | 3056 | 3057 | 3058 | 3059 | 3060 | 3061 | 3062 | 3063 |      |      |
| 5370                          | 2808                            | 2809 | 2810 | 2811 | 2812 | 2813 | 2814 | 2815 | 5770 | 3064 | 3065 | 3066 | 3067 | 3068 | 3069 | 3070 | 3071 |      |      |

|      | 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 F FRACTIONAL CONVERSION TABLE

| OCTAL | DECIMAL | OCTAL | DECIMAL | OCTAL | DECIMAL | OCTAL | DECIMAL |
|-------|---------|-------|---------|-------|---------|-------|---------|
| .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 | OCTAL   | DECIMAL | OCTAL   | DECIMAL | OCTAL   | DECIMAL |
|---------|---------|---------|---------|---------|---------|---------|---------|
| .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 | OCTAL   | DECIMAL | OCTAL   | DECIMAL | OCTAL   | DECIMAL |
|---------|---------|---------|---------|---------|---------|---------|---------|
| .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 | .001268 | .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 G ND CODE CONVERSION CHART

Use of a "packed" ASCII character set permits the programmer to increase the effective core capacity of the ND812. The code conversion character set follows.

|   | <u>Packed</u> | <u>ASCII</u> |           | <u>Packed</u> | <u>ASCII</u> |
|---|---------------|--------------|-----------|---------------|--------------|
| A | 41            | 301          | 0         | 20            | 260          |
| B | 42            | 302          | 1         | 21            | 261          |
| C | 43            | 303          | 2         | 22            | 262          |
| D | 44            | 304          | 3         | 23            | 263          |
| E | 45            | 305          | 4         | 24            | 264          |
| F | 46            | 306          | 5         | 25            | 265          |
| G | 47            | 307          | 6         | 26            | 266          |
| H | 50            | 310          | 7         | 27            | 267          |
| I | 51            | 311          | 8         | 30            | 270          |
| J | 52            | 312          | 9         | 31            | 271          |
| K | 53            | 313          | \$        | -             | 244          |
| L | 54            | 314          | *         | 12            | 252          |
| M | 55            | 315          | +         | 13            | 253          |
| N | 56            | 316          | '         | 14            | 254          |
| O | 57            | 317          | -         | 15            | 255          |
| P | 60            | 320          | .         | 16            | 256          |
| Q | 61            | 321          | /         | 17            | 257          |
| R | 62            | 322          | ;         | 33            | 273          |
| S | 63            | 323          | =         | 35            | 275          |
| T | 64            | 324          | Space     | 00            | 240          |
| U | 65            | 325          | Tab       | 74            | 211          |
| V | 66            | 326          | Form Feed | 75            | 214          |
| W | 67            | 327          | CR/LF     | 77            | 212-215      |
| X | 70            | 330          |           |               |              |
| Y | 71            | 331          |           |               |              |
| Z | 72            | 332          |           |               |              |

**NUCLEAR DATA INC.**

Nuclear Data Inc.  
P. O. Box 451  
100 West Golf Road  
Palatine, Illinois 60067  
Tel: (312) 529-4600

Nuclear Data Inc.  
103 Pincushion Road  
Framingham, Massachusetts 01701  
Tel: (617) 899-4927

Nuclear Data Inc.  
P. O. Box 2192  
14278 Wicks Boulevard  
San Leandro, California 94577  
Tel: (415) 483-9200

Nuclear Data Inc.  
2335 Brannen Road, S.E.  
Atlanta, Georgia 30316  
Tel: (404) 241-3220

Nuclear Data, GmbH  
Mainzerlandstrasse 29  
6 Frankfurt/M, Germany  
Tel: 23 11 44

Nuclear Data Inc. (U.K.)  
Rose Industrial Estate  
Cores End Road  
Bourne End, Bucks., England  
Tel: 22733

Nuclear Data (Ireland) Ltd.  
Kinsale Road, Ballycurreen  
P. O. Box #23  
Cork, Ireland  
Tel: 22137

Nuclear Data (Scandinavia)  
Division of Selektionik A/S  
Hammervej 3  
2970 Hørsholm, Denmark  
Tel: (01) 86 30 00