

# Burroughs



# PUBLICATION CHANGE NOTICE

PCN No.: 1045481-001 Date: March 29, 1971

Publication Title: Burroughs L/TC Basic Assembler Reference Manual

Other Affected Publications: None

Supersedes: N/A

## Description

This PCN provides both replacement and additional pages for the Burroughs L/TC Basic Assembler Manual. This material includes corrections to existing information plus a reorganization of certain sections designed to clarify and improve the manual. Black bars were not used to indicate revisions in the manual since the PCN is to a large extent a rearrangement of the sections and appendices of the original manual.

| <u>Replace</u>                 | <u>Add</u>                | <u>Delete</u> |
|--------------------------------|---------------------------|---------------|
| Table of Contents              | Pages J-5, J-6<br>and J-7 | Section Six   |
| Introduction                   | (Appendix J)              | Appendix L    |
| Section Two                    |                           |               |
| Section Three                  |                           |               |
| Appendices B, C, E,<br>G and K |                           |               |

Retain this PCN as a record of changes made to this basic publication.

The above replacement pages covering PCN 1045481-001

**COPYRIGHT © 1971 BURROUGHS CORPORATION**



# TABLE OF CONTENTS

| SECTION | TITLE                                     | PAGE |
|---------|---|------|
|         | INTRODUCTION                              | xi   |
| 1       | ASSEMBLER CODING FORM . . . . .           | 1-1  |
|         | Program Identification . . . . .          | 1-1  |
|         | Page Number and Heading . . . . .         | 1-1  |
|         | Sequence . . . . .                        | 1-1  |
|         | Label . . . . .                           | 1-2  |
|         | Operation Code . . . . .                  | 1-2  |
|         | Field Length . . . . .                    | 1-3  |
|         | A Parameter — Label . . . . .             | 1-3  |
|         | A Parameter ± Increment . . . . .         | 1-3  |
|         | B Parameter . . . . .                     | 1-4  |
|         | C Parameter . . . . .                     | 1-4  |
|         | Constant Data (Numeric) . . . . .         | 1-4  |
|         | Alphanumeric Data or Print Mask . . . . . | 1-4  |
|         | Remarks . . . . .                         | 1-4  |

## TABLE OF CONTENTS (continued)

| <b>Ref. No.</b> | <b>Subject</b>                               |
|-----------------|--|
| <b>2.00</b>     | <b>INTRODUCTION</b>                          |
| <b>2.01</b>     | <b>ASSEMBLER PSEUDO INSTRUCTIONS</b>         |
| <b>2.02</b>     | <b>KEYBOARD INSTRUCTIONS</b>                 |
| 2.02.01         | Enable Numeric Keyboard Instructions         |
| 2.02.02         | Operation Control and Program Keys           |
| 2.02.03         | Typewriter Keyboard Instructions             |
| <b>2.03</b>     | <b>PRINT INSTRUCTIONS</b>                    |
| 2.03.01         | Modes for Printing                           |
| 2.03.02         | Load Position Register Instruction           |
| 2.03.03         | Print Alphanumeric from Memory Instruction   |
| 2.03.04         | Load Print-Numeric Base Register Instruction |
| 2.03.05         | Mask Word                                    |
| 2.03.06         | Numeric Printing Instructions                |
| 2.03.07         | Single Character Print Instructions          |
| 2.03.08         | Ribbon Shift Instruction                     |
| <b>2.04</b>     | <b>FORMS CONTROL INSTRUCTION</b>             |
| 2.04.01         | Forms Handler – Open and Close Instruction   |
| 2.04.02         | Platen Control Register Instructions         |
| 2.04.03         | Line Advance Instructions                    |
| <b>2.05</b>     | <b>ARITHMETIC INSTRUCTIONS</b>               |
| 2.05.01         | Addition Instructions                        |
| 2.05.02         | Add Constant to Accumulator Instruction      |
| 2.05.03         | Clear Instructions                           |
| 2.05.04         | Insert Constant in Accumulator Instruction   |
| 2.05.05         | Multiplication and Division Instructions     |
| 2.05.06         | Subtract Instructions                        |
| <b>2.06</b>     | <b>DATA MOVEMENT INSTRUCTIONS</b>            |
| 2.06.01         | Transfer Instructions                        |
| 2.06.02         | Shift Accumulator Instructions               |
| <b>2.07</b>     | <b>FLAG INSTRUCTIONS</b>                     |
| <b>2.08</b>     | <b>INDEX REGISTER INSTRUCTIONS</b>           |
| <b>2.09</b>     | <b>BRANCH AND DECISION INSTRUCTIONS</b>      |
| 2.09.01         | Branch Unconditional Instruction             |
| 2.09.02         | Subroutine Jump and Return Instructions      |
| 2.09.03         | Compare Alphanumeric Instruction             |
| 2.09.04         | Accumulator Skip and Execute Instructions    |
| 2.09.05         | Flag Execute and Skip Instructions           |
| 2.09.06         | Skip and Execute Instructions for TC 700     |

## TABLE OF CONTENTS (continued)

| Ref. No.    | Subject   |
|-------------|---|
| <b>2.10</b> | <b>MISCELLANEOUS INSTRUCTIONS</b>                             |
| <b>2.11</b> | <b>CHECK DIGIT INSTRUCTIONS</b>                               |
| <b>2.12</b> | <b>DATA COMMUNICATIONS INSTRUCTIONS</b>                       |
| 2.12.01     | General Description   |
| 2.12.02     | Establishing Receive/Transmit Record Areas                    |
| 2.12.03     | Transferring Data From One Memory Address to Another          |
| 2.12.04     | Unpacking Messages Received                                   |
| 2.12.05     | Preparing Messages for Transmission                           |
| 2.12.06     | Field Identifier Codes and Variable Length Fields             |
| 2.12.07     | "D" Flag Group  |
| 2.12.08     | Send and Receive Address Instructions                         |
| 2.12.09     | Transmission Numbers  |
| <b>2.13</b> | <b>POINT-TO-POINT PROGRAMING PROCEDURES</b>                   |
| 2.13.01     | Basic Point-to-Point Line Discipline                          |
| 2.13.02     | Control Registers   |
| 2.13.03     | Indicator Register Flags                                      |
| <b>2.14</b> | <b>CENTRAL TC CONTROLLER PROGRAMING PROCEDURES</b>            |
| 2.14.01     | Line Discipline Format Registers                              |
| 2.14.02     | Data Comm Processor Operations                                |
| 2.14.03     | Main Memory Processor   |
| <b>2.15</b> | <b>INPUT WITH PUNCHED PAPER TAPE/EDGE PUNCHED CARD READER</b> |
| 2.15.01     | Paper Tape Reader Instructions                                |
| 2.15.02     | Paper Tape/Edge Punched Card Input Instructions               |
| <b>2.16</b> | <b>OUTPUT WITH PAPER TAPE/EDGE PUNCHED CARD PERFORATOR</b>    |
| 2.16.01     | Paper Tape/Edge Punched Card Output Instructions              |
| 2.16.02     | Reader and Punch Flags  |
| <b>2.17</b> | <b>80-COLUMN PUNCHED CARD INPUT INSTRUCTIONS</b>              |
| 2.17.01     | 80-Column Card Input Instructions                             |
| 2.17.02     | Input Indicator Lights and Flags                              |
| 2.17.03     | Program Keys  |
| <b>2.18</b> | <b>80-COLUMN PUNCHED CARD OUTPUT INSTRUCTIONS</b>             |
| 2.18.01     | Punching Alphanumeric Data                                    |
| 2.18.02     | Punching Numeric Data from the Accumulator                    |
| 2.18.03     | Card Column Synchronization With the Punch Count Register     |
| 2.18.04     | Output Indicator Lights and Flags                             |

## TABLE OF CONTENTS (continued)

| <b>Ref. No.</b> | <b>Subject</b>  |
|-----------------|---|
| <b>2.19</b>     | <b>MAGNETIC UNIT RECORD INSTRUCTIONS</b>                  |
| 2.19.01         | Magnetic Unit Record Formats                              |
| 2.19.02         | Magnetic Unit Record Pseudo Instructions                  |
| 2.19.03         | Magnetic Unit Record Flags                                |
| 2.19.04         | Write Instructions  |
| 2.19.05         | Read Instruction  |
| 2.19.06         | Print Alpha From Magnetic Record Area Instruction         |
| 2.19.07         | Arithmetic Instructions                                   |
| 2.19.08         | Transfer Instructions                                     |
| 2.19.09         | Unit Record Alignment Instructions                        |
| 2.19.10         | Record Alignment Errors and Flag Indicators               |
| <b>2.20</b>     | <b>MESSAGE UNPACKING ROUTINE</b>                          |
| 2.20.01         | General Description                                       |
| 2.20.02         | Position Table  |
| 2.20.03         | Data Element Codes  |
| 2.20.04         | Storage Area  |
| 2.20.05         | Error Conditions  |
| 2.20.06         | Delimiter   |
| 2.20.07         | Programing Requirements                                   |
| <b>2.21</b>     | <b>TRANSACTION CODE TRANSLATOR</b>                        |
| 2.21.01         | General Description                                       |
| 2.21.02         | Translation Table Format                                  |
| 2.21.03         | Automatic Codes   |
| 2.21.04         | Code Modification   |
| 2.21.05         | Error Conditions  |
| 2.21.06         | Machine Code for Transaction Code Translation Instruction |
| 2.21.07         | Word 576  |
| 2.21.08         | User Program Requirements                                 |
| 2.21.09         | Programing Example  |

## TABLE OF CONTENTS (continued)

| SECTION | TITLE   | PAGE |
|---------|---|------|
| 3       | SYMBOLIC PROGRAMING PROCEDURES . . . . .                                | 3-1  |
|         | Program Definition . . . . .  | 3-1  |
|         | Program Writing . . . . .   | 3-1  |
|         | Program Debugging . . . . .   | 3-3  |
|         | Data Comm Debugging . . . . .   | 3-3  |
| 4       | PROGRAMING EXAMPLE. . . . .   | 4-1  |
|         | Problem . . . . .   | 4-1  |
|         | Solution . . . . .  | 4-1  |
|         | Solution Index . . . . .  | 4-1  |
|         | General Systems Flowchart . . . . .                                     | 4-2  |
|         | Program Definition Worksheets . . . . .                                 | 4-3  |
|         | Program Definition Charts . . . . .                                     | 4-4  |
|         | Sample Coding Forms . . . . .   | 4-7  |
|         | Assembler III Listing . . . . .   | 4-33 |
|         | Sample Output. . . . .  | 4-69 |
|         | Cross Reference Table . . . . .   | 4-70 |
| 5       | ASSEMBLERS. . . . .   | 5-1  |
|         | Functional Description of Basic Assemblers . . . . .                    | 5-1  |
|         | Assembler I L/TC Paper Tape Version . . . . .                           | 5-1  |
|         | Equipment Required . . . . .  | 5-1  |
|         | Phase I. . . . .  | 5-1  |
|         | Phase I – Input . . . . .   | 5-1  |
|         | Phase I – Operating Instructions . . . . .                              | 5-2  |
|         | Phase I – Condensed Operating Instructions and Reference List . . . . . | 5-7  |
|         | Phase I – Diagnostic Facilities . . . . .                               | 5-9  |
|         | Phase I – Output . . . . .  | 5-11 |
|         | Phase I – Print-out . . . . .   | 5-11 |
|         | Phase I – Output Tape . . . . .   | 5-11 |
|         | Phase II . . . . .  | 5-13 |
|         | Phase II – Input . . . . .  | 5-13 |
|         | Phase II – Operating Instructions . . . . .                             | 5-13 |
|         | Phase II – Condensed Operating Instructions. . . . .                    | 5-13 |
|         | Phase II – Error Detection . . . . .                                    | 5-14 |
|         | Phase II – Output . . . . .   | 5-16 |
|         | Phase II – Print-out . . . . .  | 5-16 |
|         | Phase II – Output Tape . . . . .  | 5-16 |
|         | Assembler II L/TC 80-column Card I/O . . . . .                          | 5-17 |
|         | Environment . . . . .   | 5-17 |
|         | Input . . . . .   | 5-17 |
|         | Control Cards . . . . .   | 5-17 |
|         | Operating Instructions. . . . .   | 5-18 |
|         | Readying the System . . . . .   | 5-18 |
|         | Pass I . . . . .  | 5-18 |
|         | Pass I Errors. . . . .  | 5-18 |

**TABLE OF CONTENTS (continued)**

| <b>SECTION</b> | <b>TITLE</b>  | <b>PAGE</b> |
|----------------|---|-------------|
| 5 (cont'd)     | Pass II . . . . .   | 5-20        |
|                | Pass II – Errors . . . . .  | 5-20        |
|                | Assembler III B 3500 Version. . . . .                                   | 5-22        |
|                | Environment . . . . .   | 5-22        |
|                | MCP Control Cards . . . . .   | 5-23        |
|                | Option Control Cards . . . . .  | 5-23        |
|                | Operating Instructions. . . . .   | 5-25        |
|                | Error Detection . . . . .   | 5-28        |
|                | Output . . . . .  | 5-32        |
|                | L/TC Assembler IV B 5500 Version . . . . .                              | 5-35        |
|                | Environment . . . . .   | 5-35        |
|                | MCP Control Cards . . . . .   | 5-35        |
|                | Operating Instructions. . . . .   | 5-37        |
|                | Operation. . . . .  | 5-38        |
|                | Error Detection . . . . .   | 5-38        |
|                | Output . . . . .  | 5-38        |
|                | L/TC Assembler V B 300 Version . . . . .                                | 5-39        |
|                | Environment . . . . .   | 5-39        |
|                | Input . . . . .   | 5-39        |
|                | Output . . . . .  | 5-39        |
|                | Control Cards . . . . .   | 5-39        |
|                | Operating Instructions. . . . .   | 5-41        |
|                | Programed Halts . . . . .   | 5-44        |
|                | Error Detection . . . . .   | 5-45        |
|                | Assembler VI Series L 40 Track Version . . . . .                        | 5-50        |
|                | Equipment Required . . . . .  | 5-50        |
|                | Phase I. . . . .  | 5-50        |
|                | Phase I – Input . . . . .   | 5-50        |
|                | Phase I – Operating Instructions . . . . .                              | 5-50        |
|                | Phase I – Condensed Operating Instructions and Reference List . . . . . | 5-57        |
|                | Phase I – Diagnostic Facilities . . . . .                               | 5-59        |
|                | Phase I – Output . . . . .  | 5-60        |
|                | Phase I – Print-out . . . . .   | 5-60        |
|                | Phase I – Output Tape . . . . .   | 5-62        |
|                | Phase II . . . . .  | 5-62        |
|                | Phase II – Input . . . . .  | 5-62        |
|                | Phase II – Operating Instructions . . . . .                             | 5-62        |
|                | Phase II – Condensed Operating Instructions. . . . .                    | 5-63        |
|                | Phase II – Error Detection . . . . .                                    | 5-63        |
|                | Phase II – Output Tape . . . . .  | 5-65        |



**TABLE OF CONTENTS (continued)**

| <b>SECTION</b>     | <b>TITLE</b>   | <b>PAGE</b> |
|--------------------|--|-------------|
| APPENDIX A         | Glossary . . . . .                                     | A-1         |
| APPENDIX B         | GP 300 Instructions to Machine Language . . . . .      | B-1         |
| APPENDIX C         | Assembler Pseudo Instructions . . . . .                | C-1         |
| APPENDIX D         | Series L/TC Character Sets . . . . .                   | D-1         |
| APPENDIX E         | Table of Mask Codes . . . . .                          | E-1         |
| APPENDIX F         | Error Messages for B 3500 Assembly . . . . .           | F-1         |
|                    | Error Messages for B 5500 Assembly . . . . .           | F-2         |
|                    | Error Messages for B 300 Assembly . . . . .            | F-2         |
| APPENDIX G         | Instructions for Key punching Symbolic Cards . . . . . | G-1         |
|                    | Symbolic Card Format . . . . .                         | G-1         |
|                    | A 142/A 150 Key punching Instructions . . . . .        | G-2         |
|                    | 024/026/029 Key punching Instructions . . . . .        | G-3         |
| APPENDIX H         | Character Sets . . . . .                               | H-1         |
|                    | USASCII . . . . .                                      | H-2         |
|                    | BCL . . . . .  | H-2         |
|                    | EBCDIC . . . . .                                       | H-2         |
| APPENDIX I         | Table of Input Code Assignments . . . . .              | I-1         |
|                    | Input Functions for 6, 7, 8 Channel Tape . . . . .     | I-1         |
|                    | Field Identifier Codes . . . . .                       | I-3         |
|                    | Table of Output Code Assignments . . . . .             | I-4         |
| APPENDIX J         | GP 300 Timings . . . . .                               | J-1         |
| APPENDIX K         | Series L/TC Object Code . . . . .                      | K-1         |
| ALPHABETICAL INDEX | . . . . .  | One         |

# INTRODUCTION

This manual will provide the information necessary for the L/TC user to write and assemble symbolic programs using the GP 300 Basic Language. In Section 1 the coding form is analyzed by column. It is suggested that the reader remove the coding form sample on page xiv and locate each specific area on the form as he reads the text. In Section 2 each of the GP 300 series firmware instructions is presented. Individual instructions are discussed in a narrative section followed by an example which illustrates the capabilities of the instruction. The instructions (Op Codes) are presented alphabetically by a category which relates to machine function.

Section 3 defines the rules and techniques used in symbolic program writing and debugging. To the non-experienced user it is suggested that he read pages 3-1–3-2 of Section 3 before attempting the other materials contained in this manual.

A typical billing problem is discussed in Section 4. The analysis begins with the program definition and carries through to the sample output on an invoice. Section 5 is a functional description of the Basic Assemblers. Operating instructions are included.

Users are provided a means of quickly referencing selected areas of the manual by coded boxes placed in the upper corner of key pages. The information contained within the box is indicative of the material on that page. In Section 2 the symbolic OP code is placed in these boxes along with a symbol to indicate the type of firmware set to which the instruction applies. These are: CD—check digit add-on firmware sets, CRD-80—column card firmware sets, DC—data communications firmware sets, and PT—paper tape firmware sets.

Boxes which do not contain a firmware code apply to the basic instructions which are generally common to all firmware sets.

The information provided in this manual applies to the 32-track styles and the 40-track styles of the Series L/TC.

# SECTION 2

## GP 300 INSTRUCTIONS

### 2.00 – INTRODUCTION

General Purpose Language (GP 300) is a programming language, consisting of machine instructions to control system operation, and is used for Series L/TC. For ease of programming the Series L/TC, the programmer can write his programs in symbolic language and can convert them to machine language through the use of an assembler program. By using an assembler program, the programmer is not burdened with keeping track of the memory location used, or the actual machine language for the symbolic instructions being used.

The GP 300 instruction list is implemented in the system by various Firmware Sets; the number of different instructions implemented is dependent on the particular Firmware Set used in the system. Firmware is defined as a control program, and is stored in a designated area of the systems memory. The firmware performs some of the logic and control functions, programmatically, that are usually performed by hardware electronic circuits in larger computer systems.

Firmware consists of "MICRO-programs" which implement each instruction of GP 300. A MICRO-program consists of a "string" of MICRO instructions, each performing a step to accomplish the function of the GP 300 instruction (referred to as MACRO instructions). Thus, in the execution of an applicational program, the firmware identifies each MACRO instruction used by the programmer, and selects the proper "MICRO string" to perform the function of the instruction.

#### 2.00.01 MEMORY ORGANIZATION

Memory in the L/TC consists of 1,280 words of 64 bits each, and is organized into 5 blocks of 8 tracks each, or a total of 40 tracks. Each track containing 32 words. Main Memory is subdivided into two sections: The Control area and the Normal area.

The Control area contains the firmware which determine the system control functions and which implement the GP 300 instruction list. The Normal area is used to store the user's programs which are written with the MACRO instructions. The MACRO instructions are used by the programmer to exercise all of the capabilities of the L/TC such as arithmetic, logical comparisons, printing, input/output (paper tape or 80-column cards), and data transmission. The Normal area is also used for storing constant data, messages, and for accumulating totals. The amount of Normal area available to the user is dependent upon the firmware in the Control area (some firmware requires more memory than others).

#### 2.00.02 MEMORY WORD ORGANIZATION

Each word of memory contains 16 digits (64 bits) and may be used to store one of the following:

1. NUMERIC WORD – Contains only numeric values plus sign. Each digit within the number occupies a single digit within the word. Digit position 15 is reserved for flag settings.

|       |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| FLAGS | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

2. ALPHA WORD – Contains only alphanumeric values, left justified. Each alpha character requires two digit positions within the word. Eight is the maximum number of alpha characters that can be contained within a word.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

3. PROGRAM WORD – Contains 4 MACRO instructions. Each instruction requires 4 digit positions (termed a syllable) within the word.

|            |            |            |            |
|------------|------------|------------|------------|
| Syllable 3 | Syllable 2 | Syllable 1 | Syllable 0 |
|------------|------------|------------|------------|

4. PRINT FORMAT WORD – Contains only print format codes. Each code value occupies a single digit position within the word. Digit position 15 is reserved for flag settings.

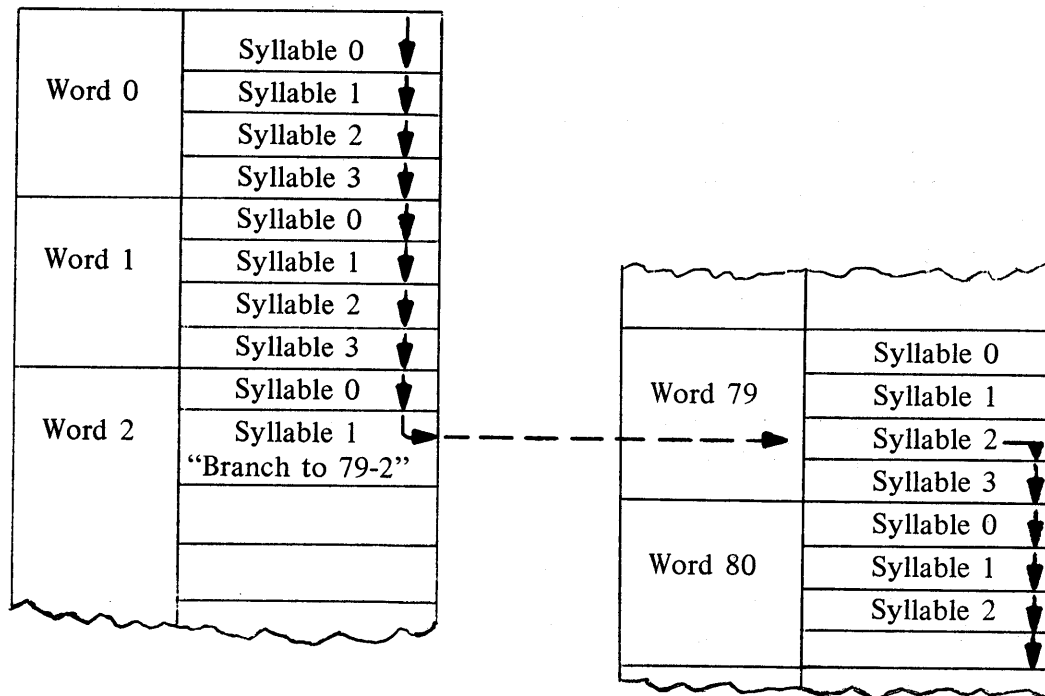
|       |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| FLAGS | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

The words are addressed by a word number. The word number is an integer which lies between 0 and the highest available word to the user. The word number is sometimes referred to as memory address or memory location. If a word contains program instructions, it is divided into four syllables, each syllable containing one instruction. The syllables are numbered 0, 1, 2, 3 as shown above within the word.

### 2.00.03 PROGRAM EXECUTION

When the system is activated and the program mode is entered by depression of the START key, execution of the program instructions begins in word 0, syllable 0. Execution continues sequentially by incrementing the syllable value by 1 (certain instructions can modify this procedure, e.g., a branch instruction). When the syllable value attains 3, the next increment will cause the word number to be increased by 1 and the syllable counter to be set back to 0. The current word number and syllable value are contained in the Program Counter.

The following example shows only word numbers and syllable values within those words.\*The arrows show how the values in the program counter are changed.



Sequential Program Execution and the effect of using the branch instruction

After the "START" key is depressed and program execution begins, the program counter always starts at word 0, syllable 0, it continues to be incremented until the execution of the instruction in word 2, syllable 1 (Branch instruction). After execution of this instruction causes the program counter to change value from word 2, syllable 1 to word 79, syllable 2, the program counter continues to increment until another path is selected.

#### 2.00.04 ACCUMULATOR

Set aside from the Normal area of memory, is one word called the Accumulator. It, like other numeric words, contains 15 digits and a flag position. It is not addressed by a word number, but rather, access to it is a function of certain instructions. It is a working memory location for the movement of data from one area to another. It receives all numeric data entered through the keyboard including the keys that set the Accumulator flags [RE(-), C, M]; it must contain any numeric data to be printed; it can sum up several amounts and store the result in another word; it receives the product or quotient of computations; it must be used to accumulate one word of data into another; and it can be used to move alphanumeric information from one word to another.

When the Accumulator contains 0, the minus flag is reset (i.e., the Accumulator is positive).

Certain instructions will destroy the prior contents of the Accumulator (i.e., clear the Accumulator before the instruction is executed). This frees the programmer from clearing the Accumulator through instruction before moving data.

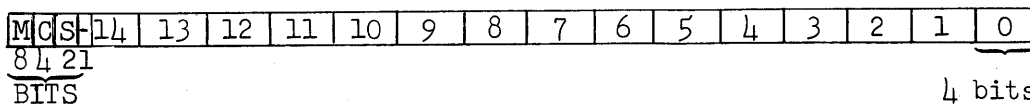
#### 2.00.05 FLAGS

Instructions are provided to "test" whether or not certain conditions exist during the execution of the program, so that alternate paths of program may be selected, depending on the state of the condition being tested. In GP 300 the user has 28 "Flags" divided into 7 groups, each of which can be tested.

There are flags for testing the condition of the Accumulator, flags to test the condition of tape or card readers and tape or card punches, flags for the OCK Keys which the operator will use, flags for forms limits, index registers used to control loops, plus general purpose flags which the user can assign for his own particular needs.

Each flag consists of 1 "bit." When the bit is "ON," the flag is "Set"; when the bit is "off," the flag is "Reset." The program can interrogate a flag to test whether or not it is set or reset, and select a path of program accordingly.

A graphic explanation below of the Accumulator which has 4 flags will show how each flag is assigned one bit.

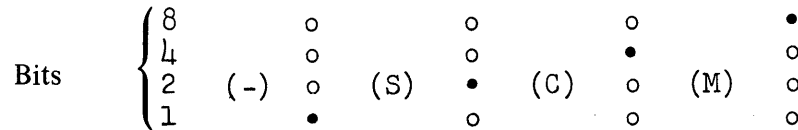


Accumulator  
Flags

(M) Per Thousand  
(C) Per Hundred

(S) Special  
(-) Minus or Negative

If we were to examine the bit configuration for the flags, they would be represented as follows:



|      |
|------|
| ADVL |
| ALF  |

**2.01 -- ASSEMBLER PSEUDO INSTRUCTIONS**

Pseudo instructions control the manner of assembly and determine the interpretation of data fed to the assembler. They generally do not directly produce machine language instructions, except in some cases where they fill in syllables to increment the program counter to the next word.

The following instructions are valid for this Basic Assembler Language.

**2.01.01 ADVANCE LINE INSTRUCTION**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| ADVL           | 1-4      |

The ADVL pseudo instruction will advance the assembler output form the number of lines specified in the A parameter. No machine language instruction is assembled.

**2.01.02 ALPHA CONSTANT INSTRUCTION**

|                |
|----------------|
| <u>OP CODE</u> |
| ALF            |

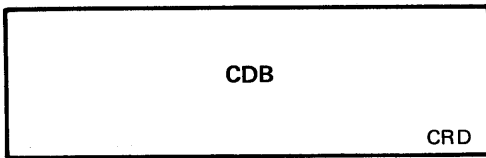
The ALF pseudo instruction permits alphanumeric data, up to 24 characters, to be stored in memory as constant data during program loading. Any character on the keyboard, including space, is a valid character. (Except for Assembler I, a CC in columns 27 and 28 will allow a second line of 24 characters to be entered.)

If the syllable counter is not 0 at the beginning of the ALF, "STOP" instructions are inserted until the counter is 0. The alphanumeric constant is then assembled starting in the next full word.

The alpha data is identified by placing a label in the label field, unless reference will be made by + or - incrementing from another entry. For assemblers other than Assembler I, the total number of characters in the ALF constant must appear in the FIELD LENGTH

Example:

|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    | FIELD<br>LEN-<br>GTH |    | PARAMETER |    |    |    |  |  |  |  |  |  |  |  |
|-------|----|----|----|----|----|----------|----|----|----|----|----|-------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----------------------|----|-----------|----|----|----|--|--|--|--|--|--|--|--|
| LABEL |    |    |    |    |    | OP. CODE |    |    |    |    |    | A     |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    | LABEL |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28    | 29 | 30 | 31 | 32 | 33 | 34                | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42                   | 43 | 44        | 45 | 46 | 47 |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |                   |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |  |  |  |  |  |  |  |



LABEL

OP CODE

A

PA

NAME

NAME

ALF

JOHN DOE

When the PA instruction is executed, the alphanumeric characters JOHN DOE would be printed (including the space).

**2.01.03 RESERVE CARD BUFFER INSTRUCTION**

OP CODE

CDB

The CDB pseudo instruction inserts the instruction "BRU to word 11, syllable 0" in word 0, syllable 0. This causes the assembler to reserve words 1-10 as the card read-in buffer area. If the assembly word counter is not at word 0, syllable 0, an error message will print. (When using Assembler I, the assembly will halt; with Assembler III or IV it will not halt, but 10 words will not be reserved.)

Accordingly, the CDB instruction must be the first instruction in the program except for pseudo instructions which do not affect memory allocation such as "Note."

When the card input data is no longer needed, the 10-word read-in area may be referenced as working memory by other parts of a program. This is accomplished by providing the CDB instruction with a label.

Example:

| LABEL  |    | OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|--------|----|----------|----|----------------------|-----------|----|----|----|----|-------------------|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        |    |          |    |                      | A         |    |    |    |    | B                 |        |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16     | 17 | 18       | 19 | 20                   | 21        | 22 | 23 | 24 | 25 | 26                | 27     | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| CARDIN |    | CDB      |    |                      |           |    |    |    |    |                   |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | RCD      |    |                      |           |    |    |    |    |                   |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | TRM      |    |                      |           |    |    |    |    |                   | CARDIN |    |    |    |    | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|--------------|----------------|----------|----------|--|
| CARDIN       | CDB            |          |          | Reserve Card Buffer,   |
|              | RCD            |          |          | Read 1 card.   |
|              | TRM            | CARDIN+2 |          | Use 3rd word of card read buffer as a working memory location. |



The card input area can be reserved by using the "REG" pseudo instruction. In this circumstance the programmer must include his own provision to by-pass the 10-word buffer area.

Example:

| LABEL |    | OP. CODE |    |    |    |      | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|------|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |      |                      | A         |    |    |    |    | B                 |    |    |    |    |    |    |    |    |    | C  |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |      |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22   | 23                   | 24        | 25 | 26 | 27 | 28 | 29                | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    |          |    |    |    | LPNR |                      |           |    |    |    |    | PMASK             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | LPKR |                      |           |    |    |    |    | PKEYS             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | LLLR |                      |           |    |    |    |    | SI                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | BRU  |                      |           |    |    |    |    | BEGIN             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | REG  |                      |           |    |    |    |    | 10                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| BEGIN |    |          |    |    |    | RCD  |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                        |
|--------------|----------------|----------|----------|---------------------------------------|
|              | LPNR           | PMASK    |          | Assembles in word 0                   |
|              | LPKR           | PKEYS    |          | Assembles in word 0                   |
|              | LLLR           | 51       |          | Assembles in word 0                   |
|              | BRU            | BEGIN    |          | Assembles in word 0                   |
|              | REG            | 10       |          | Assembles in words 1-10               |
| BEGIN        | RCD            |          |          | Assembles in word 11, syl-<br>lable 0 |

**2.01.04 CARD FORMAT INSTRUCTION**

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| CDF            | 1-80     | 1-80     |

The CDF pseudo instruction is used to define each field for 80-column card input. The A parameter denotes the beginning card column of the field. The B parameter indicates the number of card columns in the field. The values entered are assembled into one syllable as part of the card format table.

The field formats defined in the table may pertain to one or several types of input cards, and may be in any sequence in relation to the card.

CODE

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>            |
|--------------|----------------|----------|----------|---------------------------|
|              | LCFR           | FIELDS   |          | Load Card Format Register |
|              |                | WORD     |          |                           |
| FIELDS       | CDF            | 1        | 1        | 1 - type of card          |
|              | CDF            | 2        | 7        | 2 - Acct. No.             |
|              | CDF            | 9        | 6        | 3 - Product Codes         |
|              | CDF            | 15       | 36       | 4 - Product Description   |
|              | CDF            | 51       | 6        | 5 - Gross Weight          |
|              | CDF            | 57       | 8        | 6 - Price No. 1           |
|              | CDF            | 65       | 8        | 7 - Price No. 2           |
|              | CDF            | 73       | 8        | 8 - Cost                  |
|              | CDF            | 9        | 24       | 9 - Name                  |
|              | CDF            | 33       | 24       | 10 - Address              |
|              | CDF            | 57       | 24       | 11 - City-State           |

2.01.05 CODE INSTRUCTION

| <u>OP CODE</u> | <u>A</u>             |
|----------------|----------------------|
| CODE           | 4 hexadecimal digits |

The CODE pseudo instruction permits the insertion of 4 hexadecimal digits into the next available syllable of a word of memory. The value designated by the 4 digits in the A parameter is assembled into the word syllable. Other instructions may precede or follow its use in the same word of memory, or it may be used successively to insert a full word or several words.

Example:

|       |    |    |    |    |    |             |    |    |    |    |    |                      |             |    |    |    |    |    |                   |    |    |    |    |    |    | PARAMETER |    |    |    |    |  |  |  |  |  |  |  |
|-------|----|----|----|----|----|-------------|----|----|----|----|----|----------------------|-------------|----|----|----|----|----|-------------------|----|----|----|----|----|----|-----------|----|----|----|----|--|--|--|--|--|--|--|
| LABEL |    |    |    |    |    | OP. CODE    |    |    |    |    |    | FIELD<br>LEN-<br>GTH | A           |    |    |    |    |    | B                 |    |    | C  |    |    |    |           |    |    |    |    |  |  |  |  |  |  |  |
|       |    |    |    |    |    |             |    |    |    |    |    |                      | LABEL       |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |           |    |    |    |    |  |  |  |  |  |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22          | 23 | 24 | 25 | 26 | 27 | 28                   | 29          | 30 | 31 | 32 | 33 | 34 | 35                | 36 | 37 | 38 | 39 | 40 | 41 | 42        | 43 | 44 | 45 | 46 |  |  |  |  |  |  |  |
|       |    |    |    |    |    |             |    |    |    |    |    |                      |             |    |    |    |    |    |                   |    |    |    |    |    |    |           |    |    |    |    |  |  |  |  |  |  |  |
|       |    |    |    |    |    | <b>CODE</b> |    |    |    |    |    |                      | <b>C925</b> |    |    |    |    |    |                   |    |    |    |    |    |    |           |    |    |    |    |  |  |  |  |  |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>           |
|----------------|----------|--------------------------|
| CODE           | C925     | Print word 293 as alpha. |

|      |
|------|
| DEF  |
| DEFT |

C925 is the machine language code for PA Word 293 and would be assembled into the next available syllable. It may sometimes be convenient to use the CODE instruction in this manner to have access to memory locations or program routines which have been loaded with another program.

**2.01.06 DEFINE INSTRUCTIONS**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| DEF            | 0-767    |          |
| DEFT           | 0-15     | 0-15     |

The DEF pseudo instruction is used to assign a numeric value to a label. This applies to labels which name something other than a memory location.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    | B                 | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29                | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|              |                |          |
|--------------|----------------|----------|
| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|              | POS            | SHIPTO   |
|              | }              | }        |
| SHIPTO       | DEF            | 35       |

The print ball positions at position 35.

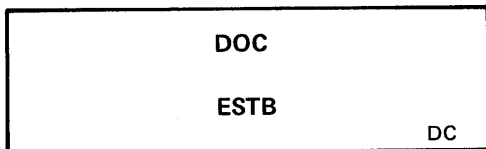
The function of the DEFT pseudo instruction is the same as that of the DEF instruction. The DEFT instruction is used with instructions which require both an A and a B parameter. Values between 0 and 15 are permitted in each parameter.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    | B                 | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29                | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|              |                |          |          |
|--------------|----------------|----------|----------|
| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|              | N.K            | ORDER    |          |
|              | }              |          |          |
| ORDER        | DEFT           | 6        | 0        |



|              |                |          |          |
|--------------|----------------|----------|----------|
| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|              | NK             | ORDER    |          |
| ORDER        | DEFT           | 6        | 0        |

The DEF or DEFT instruction must be used in conjunction with a label (in columns 16-21) to denote the item being defined.

**2.01.07 DOCUMENTATION INSTRUCTION (USED ONLY FOR ASSEMBLY ON B 2500/3500/5500.)**

OP CODE

DOC

The DOC pseudo instruction permits more extensive narrative to be included in programs and in the subroutine library. Remarks of up to 49 characters are entered (beginning in card column 29) which print on the assembly documentation from the B 3500, but which do not punch into the program tape (or card deck).

**2.01.08 ESTABLISH BUFFER INSTRUCTION**

OP CODE

ESTB

The ESTB pseudo instruction is used for reserving main memory buffer areas in connection with the data communications message handling instruction. This is required when it is desired to move a message from the Data Communications Message Received Buffer into main memory before unpacking the message, or to build a message in main memory and then transfer it (completely formatted) to the Data Communications transmit buffer.

The ESTB instruction reserves a 32 word area (256 characters) or 1 track in user memory. It selects the highest track of user memory that is available, reserving 32 words starting with the first word of that track.

For example, if 384 words of user memory (0 to 383) are designated in the program assembly, the first use of ESTB would reserve words 352 through 383; the second use of ESTB would reserve words 320-351. ESTB has no parameters, but it must be labeled.

Example:

|         |    |    |    |    |    |          |    |    |    |    |    |               |    |       |    |    |    |    |                |    |    |    |    |    |    | PARAMETER |    |    |    |  |  |  |  |  |  |
|---------|----|----|----|----|----|----------|----|----|----|----|----|---------------|----|-------|----|----|----|----|----------------|----|----|----|----|----|----|-----------|----|----|----|--|--|--|--|--|--|
| LABEL   |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD LEN-GTH |    | A     |    |    |    |    | B              |    |    | C  |    |    |    |           |    |    |    |  |  |  |  |  |  |
|         |    |    |    |    |    |          |    |    |    |    |    |               |    | LABEL |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |           |    |    |    |  |  |  |  |  |  |
| 16      | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28            | 29 | 30    | 31 | 32 | 33 | 34 | 35             | 36 | 37 | 38 | 39 | 40 | 41 | 42        | 43 | 44 | 45 |  |  |  |  |  |  |
| RECEIVE |    |    |    |    |    | ESTB     |    |    |    |    |    |               |    |       |    |    |    |    |                |    |    |    |    |    |    |           |    |    |    |  |  |  |  |  |  |
| SEND    |    |    |    |    |    | ESTB     |    |    |    |    |    |               |    |       |    |    |    |    |                |    |    |    |    |    |    |           |    |    |    |  |  |  |  |  |  |
|         |    |    |    |    |    |          |    |    |    |    |    |               |    |       |    |    |    |    |                |    |    |    |    |    |    |           |    |    |    |  |  |  |  |  |  |

|      |
|------|
| END  |
| EQU  |
| MASK |

| <u>LABEL</u> | <u>OP CODE</u> |
|--------------|----------------|
| RECEIV       | ESTB           |
| SEND         | ESTB           |

In the above example, RECEIV would be assembled with a word number of 352 and SEND would be assembled with a word number of 320.

#### 2.01.09 END INSTRUCTION

| <u>OP CODE</u> |
|----------------|
| END            |

The END pseudo instruction terminates the assembly program and must be used as the last line of code in the program.

#### 2.01.10 EQUATE INSTRUCTION

| <u>OP CODE</u> |
|----------------|
| EQU            |

The EQU pseudo instruction will permit one label to be given the identical value of another label. The label coded in columns 16-21 will be equated to the label in columns 29-34. The label contained in the parameter field (column 29-34) must have been previously used or defined.

#### 2.01.11 MASK INSTRUCTION

| <u>OP CODE</u> |
|----------------|
| MASK           |

The MASK pseudo instruction is used to enter the table of mask words. An entry of up to 24 print format characters is accepted.

If the syllable counter is not 0 at the beginning of the Mask instruction, "Stop" instructions are inserted until the counter reaches 0. The Mask Characters are then assembled in the next full word.

The appearance of any character other than those listed in the Mask Character Table (see Appendix E) results in an error condition.

The mask table must be identified by placing its label in the label field (columns 16-21) on the line of the first mask word entry. For Assemblers other than the Assembler I, the number of mask characters must appear in the field length.

|      |
|------|
| NOTE |
| NUM  |

Example: See subject 2.03.05.

**2.01.12 NOTE INSTRUCTION**

OP CODE

NOTE

The NOTE pseudo instruction will permit the entry of up to 25 characters in the REMARKS field (columns 53-77). No machine language instruction is assembled. No parameter field entry is required. If one is given, it will be ignored.

Example:

| OP. CODE |    |    |    |    |    |    |    |    |    |    |    |    |    | FIELD LEN-GTH |    | PARAMETER |    |    |    |    |    |    |    |    |    |    | REMARKS              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|----|-----------|----|----|----|----|----|----|----|----|----|----|----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |               |    | A         |    |    |    |    |    | B  |    |    | C  |    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36            | 37 | 38        | 39 | 40 | 41 | 42 | 43 | 53 | 54 | 55 | 56 | 57 | 58                   | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 |
| NOTE     |    |    |    |    |    |    |    |    |    |    |    |    |    |               |    |           |    |    |    |    |    |    |    |    |    |    | BEGIN TOTALS ROUTINE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

OP CODE

NOTE

REMARKS

Begin total routine.

**2.01.13 NUMBER INSTRUCTION**

OP CODE

NUM

The NUM pseudo instruction permits a word of numeric data to be stored as constant data in memory during program loading.

A numeric constant of from 0 to 15 digits (Assembler I will allow only 14 digits) consisting of the digits 0-9 is accepted. In addition, the “-,” “C” and “M” codes preceding the digit positions of the constant are accepted, and set their respective flags in the flag positions of the word.

If the syllable counter is not 0, “Stop” instructions are inserted until the counter is 0. The numeric constant is then assembled in the next full word, right justified.

The number must be identified by placing its name label in the label field (columns 16-21) of the coding form, unless reference will be made to it by +/- incrementing from another entry.

Example:

| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |     |    |    |    | OP. CODE |    |    |                 |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |  |  |  |  |   |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|-----|----|----|----|----------|----|----|-----------------|----|----|----------------------|-----------|--|--|--|--|--|--|--|--|--|--|--|--|--|---|
|       |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |     |    |    |    |          |    |    |                 |    |    |                      | A         |  |  |  |  |  |  |  |  |  |  |  |  |  | B |
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |     |    |    |    |          |    |    |                 |    |    |                      |           |  |  |  |  |  |  |  |  |  |  |  |  |  |   |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30                | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38  | 39 | 40 | 41 | 42       | 43 | 44 | 45              | 46 | 47 |                      |           |  |  |  |  |  |  |  |  |  |  |  |  |  |   |
| PI    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    | MUL |    |    |    |          |    |    | PI              |    |    |                      |           |  |  |  |  |  |  |  |  |  |  |  |  |  |   |
| PI    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    | NUM |    |    |    |          |    |    | 314159265358979 |    |    |                      |           |  |  |  |  |  |  |  |  |  |  |  |  |  |   |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>        | <u>B</u> | <u>REMARKS</u>   |
|--------------|----------------|-----------------|----------|------------------|
|              | MUL            | PI              |          | Multiply by PI   |
| PI           | NUM            | 314159265358979 |          | PI to 14 places. |

**2.01.14 ORIGIN INSTRUCTION**

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| ORG            | 0-767    |

The ORG pseudo instruction will assemble the next instruction in syllable 0 of the word specified in the parameter field. If the specified word has already been assigned by the assembler, an error message will be printed and entry assignment will start at the same sequence.

No machine language instruction is assembled.

**2.01.15 PAGE INSTRUCTION**      OP CODE  
PAGE

The PAGE pseudo instruction will cause the assembler output to be spaced to the top of a new form.

**2.01.16 REGION INSTRUCTION**

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| REG            | 1-255    |

The REG pseudo instruction sets aside the number of words of memory specified by the A parameter. The actual memory address is assigned by the assembler. If the syllable counter is not 0, "Stop" instructions are inserted until the counter equals zero.

WORD

The word counter is advanced by the amount in the A parameter field. If the word counter exceeds the highest order word available, an error message is printed and entry assignment will start at the same sequence number.

No machine language instruction is assembled. The region must be identified by placing its name label in the label field (columns 16-21) of the coding form. This region is not cleared.

Example:

|       |    |    |    |    |    |          |    |    |    |    |    |                      |       |    |    |                   |    |    |    |    | PARAMETER |    |    |    |    |    |    |    |    |    |  |  |
|-------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|-------|----|----|-------------------|----|----|----|----|-----------|----|----|----|----|----|----|----|----|----|--|--|
| LABEL |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | A     |    |    |                   |    |    | B  |    |           | C  |    |    |    |    |    |    |    |    |  |  |
|       |    |    |    |    |    |          |    |    |    |    |    |                      | LABEL |    |    | + OR -<br>INC/REL |    |    |    |    |           |    |    |    |    |    |    |    |    |    |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28                   | 29    | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37        | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |  |  |
|       |    |    |    |    |    | L        | K  | B  | R  |    |    |                      | A     | R  | E  | A                 |    |    |    |    |           |    |    |    |    |    |    |    |    |    |  |  |
|       |    |    |    |    |    | T        | K  | M  |    |    |    |                      | 2     | 5  |    |                   |    |    |    |    |           |    |    |    |    |    |    |    |    |    |  |  |
|       |    |    |    |    |    | ?        |    |    |    |    |    |                      |       |    |    |                   |    |    |    |    |           |    |    |    |    |    |    |    |    |    |  |  |
| A     | R  | E  | A  |    |    | R        | E  | G  |    |    |    |                      | 4     |    |    |                   |    |    |    |    |           |    |    |    |    |    |    |    |    |    |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u> |
|--------------|----------------|----------|----------|----------------|
|              | LKBR           | AREA     |          | Load keyboard  |
|              | TKM            | 25       |          | Type 25        |
| AREA         | REG            | 4        |          | Save 4 words   |

**2.01.17 WORD INSTRUCTION**

OP CODE

WORD

The WORD pseudo instruction causes the assembler to assign the next instruction at the beginning syllable of the next word.

If the syllable counter is not 0, it will be incremented and "Stop" instruction inserted into each syllable until the counter reaches 0.

This instruction should immediately precede the entry of a Program Key Table.



WORD

Example:

|       |    |    |    |    |    |          |    |    |    |    |                      |       |    |    |    |    |    |                   |  |    |    |    |    |    |    | PARAMETER |    |    |  |  |  |  |  |  |  |  |  |
|-------|----|----|----|----|----|----------|----|----|----|----|----------------------|-------|----|----|----|----|----|-------------------|--|----|----|----|----|----|----|-----------|----|----|--|--|--|--|--|--|--|--|--|
| LABEL |    |    |    |    |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | A     |    |    |    |    |    | B                 |  |    |    | C  |    |    |    |           |    |    |  |  |  |  |  |  |  |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 |                      | 29    | 30 | 31 | 32 | 33 | 34 | + OR -<br>INC/REL |  | 39 | 40 | 41 | 42 | 43 | 44 | 45        | 46 | 47 |  |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    | LPKR     |    |    |    |    |                      | PKEYS |    |    |    |    |    |                   |  |    |    |    |    |    |    |           |    |    |  |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    | S        |    |    |    |    |                      |       |    |    |    |    |    |                   |  |    |    |    |    |    |    |           |    |    |  |  |  |  |  |  |  |  |  |
|       |    |    |    |    |    | WORD     |    |    |    |    |                      |       |    |    |    |    |    |                   |  |    |    |    |    |    |    |           |    |    |  |  |  |  |  |  |  |  |  |
| PKEYS |    |    |    |    |    | BRU      |    |    |    |    |                      | START |    |    |    |    |    |                   |  |    |    |    |    |    |    |           |    |    |  |  |  |  |  |  |  |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u> |
|--------------|----------------|----------|----------|----------------|
|              | LPKR           | PKEYS    |          |                |
|              | WORD           |          |          |                |
| PKEYS        | BRU            | START    |          |                |

|     |       |
|-----|-------|
| NK  | NKCM  |
| NKR | NKRCM |

## 2.02 – KEYBOARD INSTRUCTIONS

### 2.02.01 ENABLE NUMERIC KEYBOARD INSTRUCTIONS

|   | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---|----------------|----------|----------|
| NUMERIC KEYBOARD  | NK             | 0-15     | 0-15     |
| NUMERIC KEYBOARD, PERMIT REVERSE ENTRY                  | NKR            | 0-15     | 0-15     |
| NUMERIC KEYBOARD, PERMIT C AND M KEYS                   | NKCM           | 0-15     | 0-15     |
| NUMERIC KEYBOARD, PERMIT REVERSE ENTRY,<br>C AND M KEYS | NKRCM          | 0-15     | 0-15     |

The four numeric keyboard instructions provide for the entry of a maximum of 15 digits of numeric information into the Accumulator digit positions 0-14. The Accumulator digit position 15 contains 4 flags designated “minus” (-), “special” (S), “per hundred” (C) and “per thousand” (M). These four flags are always reset at the start of any numeric keyboard or numeric entry instruction. (RE) identifies the data entered into the Accumulator as negative by setting the minus flag. The C, M Keys set the appropriate flag when depressed.

The “-,” “C,” “M” flags will be set if the particular keyboard instruction enables the use of their related keys (RE, C, M respectively) and the operator depresses these keys during the instruction. The special flag “S” cannot be set by the depression of any keyboard key. Control of this flag is accomplished by other means (see flag set/reset instructions).

The settings of the four flags transfer with the data from the Accumulator to memory and from memory back to the Accumulator and thus can be retained for future use in the program.

The A field of the instruction specifies the maximum number of digits permitted to the left of the decimal point. The parameter values range from 0-15.

The B field specifies the maximum number of digits permitted to the right of the decimal point. The parameter values range from 0-15. The sum of the A and B parameter cannot exceed 15.

When entering data, if either the A or B limits are exceeded, the Keyboard Error Indicator is turned on and the alarm bell sounds, halting the program. When the Keyboard Error Indicator is lit, all keys are disabled from performing their functions except the reset or ready push button. The entire entry must be re-indexed following the use of the reset key.

Other conditions which will cause the Keyboard Error Indicator to turn on:

1. The RE, C, M Keys are depressed during a numeric keyboard instruction that does not permit their use.
2. A typewriter key is depressed (other than 0-9, open/close key, line advance key or typewriter OCK's) during a numeric keyboard instruction.
3. A non-enabled program key has been depressed.
4. A numeric keyboard instruction is initiated when the capacity of the keyboard buffer has been exceeded and when the valid codes in the buffer do not terminate the instruction.

|     |       |
|-----|-------|
| NK  | NKCM  |
| NKR | NKRCM |

Under control of the A field the programed number of digits enter the Accumulator. Although the B field specifies how many digits can be entered to the right of the decimal point, it also determines the digit position where the whole number enters the Accumulator. The entry of each whole number causes the previously indexed digits to shift left one digit position permitting the newly indexed digit to enter the vacated digit position. A zero key depression counts as a digit even if used as the most significant digit entry. Double and triple zero keys act in the same manner counting two or three digits respectively.

Under control of the B field (following recognition of the decimal point key), the first digit is entered to the right of the phantom decimal point and the second digit in the second position with the remaining digits entered accordingly. A zero counts as a digit even if entered as the last digit after the decimal point key. It is not necessary to depress the Decimal Point Key if there are no decimal entries, even though the B field permits decimals. When the B field is zero, the error light will not become activated if the decimal point key is depressed without ensuing digit keys.

Example:

Suppose the Accumulator digit positions 0-14 contain 0. Examine the instruction.

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
| NK |    |    |    |    |    | 6  |    |    |    |    |    | 2  |    |    |    |    |    |    |    |    |    |

The operator wishes to index the number 5432.10.

The most significant digit “5” is indexed first and enters the Accumulator at digit position 2. The next digit “4” is indexed and enters the Accumulator at digit position 2 and shifts the 5 to digit position 3. This process continues until we have 000000000543200 in the Accumulator.

The decimal key is now used, and the digit 1 enters the first position to the right of the phantom decimal point. The next digit indexed enters in the next Accumulator digit position to the right of the previous entry. We now terminate the instruction with an appropriate OCK (i.e., according to program instructions).

The Accumulator now contains:

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                            |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|----------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Accumulator Digit Position |
|    | 0  | 0  | 0  | 0  | 0  | 0 | 0 | 0 | 0 | 5 | 4 | 3 | 2 | 1 | 0 | Content of Accumulator     |

Flag Position

|     |       |
|-----|-------|
| NK  | NKCM  |
| NKR | NKRCM |

Example 1: Illustrates the use of the NK instruction.

| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------------------|-----------|--|--|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|---|
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | OP. CODE             | A         |  |  |  |  |  |  |  |  |  |  | B                 |  |  |  |  | C |
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | OP. CODE             | LABEL     |  |  |  |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |  | C |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43                   |           |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | NK |    |    |    |    | 6                    |           |  |  |  |  |  |  |  |  |  |  | 5                 |  |  |  |  |   |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| NK             | 6        | 5        | Will allow for 11 characters to be entered into the Accumulator. No printing occurs. 6 to the left of digit position 5 and 5 to the right of it. |

Example 2: Illustrates the use of the NK R instruction.

| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|----|----|----|----|----------------------|-----------|--|--|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|---|
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | OP. CODE             | A         |  |  |  |  |  |  |  |  |  |  | B                 |  |  |  |  | C |
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | OP. CODE             | LABEL     |  |  |  |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |  | C |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38   | 39 | 40 | 41 | 42 | 43                   |           |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | NK R |    |    |    |    | 6                    |           |  |  |  |  |  |  |  |  |  |  | 5                 |  |  |  |  |   |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|----------------|----------|----------|---|
| NKR            | 6        | 5        | Will permit use of negative numbers (set minus flag). |

Example 3: Illustrates the use of the NKCM instruction.

| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|----|----|----|----|----------------------|-----------|--|--|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|---|
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | OP. CODE             | A         |  |  |  |  |  |  |  |  |  |  | B                 |  |  |  |  | C |
| LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |      |    |    |    |    | OP. CODE             | LABEL     |  |  |  |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |  | C |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38   | 39 | 40 | 41 | 42 | 43                   |           |  |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |
|       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | NKCM |    |    |    |    | 9                    |           |  |  |  |  |  |  |  |  |  |  | 6                 |  |  |  |  |   |

If the operator indexes 123456789, then the decimal point and 654321, the Accumulator will then contain in digit positions 0-14

123456789654321

If in addition the operator depresses the C or M key, the C or M flag will be set. Both keys can be used during the same instruction. Both flags will be set.

|     |      |
|-----|------|
| PKA | PKC  |
| PKB | LPKR |

**2.02.02 OPERATION CONTROL AND PROGRAM KEYS**

Depression of any of the Operation Control Keys (OCK's, on either the numeric or typewriter keyboard) terminates the numeric or typewriter keyboard entry, sets the corresponding OCK flag, resets the other OCK flags, and causes the next instruction in the program to be executed. All program keys are turned off.

|                            | <u>OP CODE</u> | <u>A</u> |
|----------------------------|----------------|----------|
| ENABLE PROGRAM KEY GROUP A | PKA            | 12345678 |
| ENABLE PROGRAM KEY GROUP B | PKB            | 12345678 |
| ENABLE PROGRAM KEY GROUP C | PKC            | 12345678 |

The function of a Program Key is to select and execute one instruction programed and stored in an area of memory called a Program Key Table. It also will terminate a keyboard instruction instead of an OCK, in which case all OCK flags are reset.

Program Key Group A refers to Program Keys A1-A8. Program Key Group B refers to Program Keys B1-B8. Program Key Group C refers to Program Keys C1-C8. The allowable Program Key Groups are dependent upon the machine style. The A parameter can include any number of the program keys 1-8 for a specific group (A, B or C).

All PK's that are desired must be specified by the PK command for that group, as a later command calling for that group will void the effect of an earlier command for the same group.

When in the ready mode PK: A1, A2, A3 (Start, Load, Utility respectively) have specially assigned functions and are always enabled. In the ready mode the specially assigned firmware functions take precedence over any functions programed for these keys.

After an enable program key instruction the program will not stop automatically to allow the operator time to exercise a decision. This must be done by the programmer with an instruction such as TK or NK.

|                                | <u>OP CODE</u> | <u>A</u> |
|--------------------------------|----------------|----------|
| LOAD PROGRAM KEY BASE REGISTER | LPKR           | LABEL    |

The instruction Load Program Key Base Register is used to reference the first word of a Program Key Table. (4 syllables per word). The A parameter is a label addressing the first word of the table.

The table must begin in syllable 0 of a word. Each PK has one instruction in the table. The Op-Codes for a 24 PK machine would be arranged as follows:

LKBR

|              |   |             |      |              |   |             |    |
|--------------|---|-------------|------|--------------|---|-------------|----|
| BASE WORD    | 0 | OP CODE for | PKA1 | BASE WORD +3 | 0 | OP CODE for | B5 |
|              | 1 | OP CODE for | A2   |              | 1 | OP CODE for | B6 |
|              | 3 | OP CODE for | A3   |              | 2 | OP CODE for | B7 |
|              | 4 | OP CODE for | A4   |              | 3 | OP CODE for | B8 |
| BASE WORD +1 | 0 | OP CODE for | A5   | BASE WORD +4 | 0 | OP CODE for | C1 |
|              | 1 | OP CODE for | A6   |              | 1 | OP CODE for | C2 |
|              | 2 | OP CODE for | A7   |              | 2 | OP CODE for | C3 |
|              | 3 | OP CODE for | A8   |              | 3 | OP CODE for | C4 |
| BASE WORD +2 | 0 | OP CODE for | B1   | BASE WORD +5 | 0 | OP CODE for | C5 |
|              | 1 | OP CODE for | B2   |              | 1 | OP CODE for | C6 |
|              | 2 | OP CODE for | B3   |              | 2 | OP CODE for | C7 |
|              | 3 | OP CODE for | B4   |              | 3 | OP CODE for | C8 |

There may be more than one PK table in memory at a time. The LPKR instruction must be used prior to changing the functions of the PK's in order to locate the base address of the new table.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |
| LABEL |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
|       |    |          |    |    |    | P  | K                    | A         |    |    |    |    | 1  | 2  | 3  |                   |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | N  | K                    |           |    |    |    |    | 0  |    |    |                   |    |    |    |    |    |    | 0  |    |    |    |    |

This example illustrates the use of an NK instruction to halt the program and allow the operator to select a PK key.

**2.02.03 TYPEWRITER KEYBOARD INSTRUCTIONS**

|                             |                |          |
|-----------------------------|----------------|----------|
|                             | <u>OP CODE</u> | <u>A</u> |
| LOAD KEYBOARD BASE REGISTER | LKBR           | LABEL    |

The LKBR instruction specifies the starting memory location into which information will be transferred for all succeeding TKM and EAM instructions. That is, until another LKBR instruction is executed. The A parameter addresses the starting word location in which the alpha characters will be stored.

The keyboard base register contains the location that is loaded into it until a subsequent LKBR instruction loads a new location into it.

This instruction is somewhat modified in firmware sets containing data communications capability. See Subject 2.12.03.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |  |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|--|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |  |
| LABEL |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |  |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |  |
|       |    |          |    |    |    | L  | K                    | B         | R  |    |    |    | T  | Y  | P  | E                 |    |    |    |    |    |    |    |    |    |    |    |  |
|       |    |          |    |    |    | T  | K                    | M         |    |    |    |    | 2  | 5  |    |                   |    |    |    |    |    |    |    |    |    |    |    |  |

The instructions above will allow 25 alpha characters to be stored sequentially beginning in the memory location addressed by the label TYPE.

|      | OP CODE | A                        |
|------|---------|--------------------------|
| TYPE | TK      | 0-150 15½" forms handler |
|      | TK      | 0-255 26" forms handler  |

The type instruction provides for typing and printing as a maximum the number of alphanumeric characters as specified in the A field. The A parameter ranges from 0 to 150 for 15½ inch forms handlers, while 26 inch forms handler styles provide for a 0 to 255 range. This instruction is terminated by depression of an OCK or an enabled PK.

Printing of the first character will begin at the position of the print head. If printing in a specified area is required, the print head must be repositioned to the beginning left-hand position of the print area before the typewriter instruction is reached in the program.

If typing of more than the number of characters specified in the A field is attempted, the Error Indicator is lit, and further typing is prevented. The error condition can be corrected by depression of the Reset Key. If the Reset Key is depressed during a TYPE instruction without an error condition, the instruction will be re-initiated and the print head will return to the beginning typing position.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |  |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|--|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |  |
| LABEL |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |  |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |  |
|       |    |          |    |    |    | T  | K                    |           |    |    |    |    | 9  |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |  |

The above coding will allow the computer to act as a typewriter for 9 alpha characters.

TKM

OP CODE

A

TYPE INTO MEMORY PRINT

TKM

0-150 15½" forms handler

TKM

0-255 26" forms handler

The Type into Memory instruction differs from the Type instruction in that in addition to printing alphanumeric information, the characters are also stored in memory. The space character is considered a print character and stores a code in memory. The codes for Backspace, Open/Close, Line Advance, OCK's and Program Keys are not stored in memory.

Example:

|       |    |    |    |    |    |          |    |    |    |    |    |                      |                   |    |    |    |    |    |    |    |    |    |    |    |    | PARAMETER |    |  |  |  |  |  |  |  |
|-------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|--|--|--|--|--|--|--|
| LABEL |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | A                 |    |    |    |    |    | B  |    |    | C  |    |    |    |           |    |  |  |  |  |  |  |  |
| LABEL |    |    |    |    |    |          |    |    |    |    |    |                      | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |           |    |  |  |  |  |  |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28                   | 29                | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42        | 43 |  |  |  |  |  |  |  |
|       |    |    |    |    |    | T        | K  | M  |    |    |    |                      |                   | 3  | 1  |    |    |    |    |    |    |    |    |    |    |           |    |  |  |  |  |  |  |  |

A maximum of 31 alpha characters can be typed and a maximum of 32 alpha characters (31 alpha characters plus end of alpha code 0, 0) will be entered into memory. See LKBR instruction Subject 2.02.03

This instruction is somewhat modified in firmware sets containing data communications capability. See Subject 2.12.03

The code, for each key depressed before instruction termination, is stored in memory with the first character stored in the most significant character location of the word specified by the keyboard base register. A single word can store 8 characters.

ALPHA WORD – (8 characters)

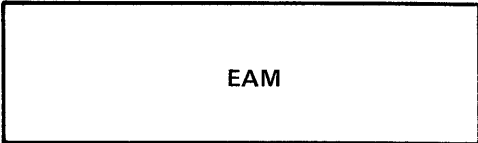
|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|

The depression of the backspace key effectively removes the last typing key code from memory. Backspacing will not occur past the first typing position.

On a TKM instruction each word is cleared before any characters are entered. The unused portion of the word remains clear. If no typing is done and the TKM instruction is terminated by an OCK, the word is clear. If exactly 8 characters were entered and then an OCK was used, the next sequential word in memory would be cleared. If a TKM is used again, without another LKBR, the data will enter memory at the first position of the last LKBR.

Note this is modified when used with Data Comm firmware. See SCP, Subject 2.12.03.





ENTER ALPHA INTO MEMORY

| <u>OP CODE</u> | <u>A</u>                 |
|----------------|--------------------------|
| EAM            | 0-150 15½" forms handler |
| EAM            | 0-255 26" forms handler  |

This instruction is identical to the TKM instruction except that printing does not occur. The print head does not escape.

|     |
|-----|
| POS |
| PA  |

### 2.03 – PRINT INSTRUCTIONS

#### 2.03.01 MODES FOR PRINTING

Instructions are provided to print in three modes:

1. Alphanumeric printing of data either from keyboard entry or from memory. When printing in this mode, the field is left justified.
2. Printing of numeric data from Accumulator. In this mode printing is right justified.
3. Printing of a single character with the actual character specified by the instruction. A single character prints in the position indicated.

#### 2.03.02 LOAD POSITION REGISTER INSTRUCTION

|                |                          |
|----------------|--------------------------|
| <u>OP CODE</u> | <u>A</u>                 |
| POS            | 0-150 15½” forms handler |
| POS            | 0-255 26” forms handler  |

The Position Register is loaded with the value of the A field. The A field ranges from 1 to 150 for 15½ inch forms handlers and 1-255 for 26 inch forms handlers. The position loaded in the position register corresponds with the actual position at which the printer will print. The print ball does not move until the program reaches an instruction which specifies that a character is to be printed, or until a keyboard instruction is reached. The print head escapes in 1/10 inch increments.

| SEQUENCE |    | LABEL |    | OP. CODE |    | FIELD LENGTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |
|----------|----|-------|----|----------|----|--------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|
|          |    |       |    |          |    |              | A         |    |    |    |    | B  |    |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |
| 11       | 12 | 13    | 14 | 15       | 16 | 17           | 18        | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |  |  |
| 0        | 1  |       |    |          |    |              |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |
| 0        | 2  |       |    |          |    |              |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |
| 0        | 3  |       |    |          |    |              |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |

The above instruction will position at position 101 or 10 inches from position 1.

#### 2.03.03 PRINT ALPHANUMERIC FROM MEMORY INSTRUCTION

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| PA             | LABEL    |

LPNR

The Print Alphanumeric instruction prints alphanumeric information from memory beginning with the first character in the memory location specified by the "A" field. Printing continues until an end of alpha code (0,0) is encountered, regardless of the number of words used.

For the PA instruction, the ribbon will be in the normal (generally black) position, although it can be changed to the reverse position by other instructions.

Example:

Suppose the alpha characters MESSAGE (and an end alpha code) are stored in memory location SAVE and we desire to print the contents of this memory location.

Initially, we position the print head. The second step is to provide for the actual printing. These two steps are programmed.

| LABEL |    |    |    |    |    |          |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    | FIELD LENGTH |    | PARAMETER |    |    |    |    |  |   |  |  |   |  |  |
|-------|----|----|----|----|----|----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|--------------|----|-----------|----|----|----|----|--|---|--|--|---|--|--|
|       |    |    |    |    |    |          |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |              |    | A         |    |    |    |    |  | B |  |  | C |  |  |
| LABEL |    |    |    |    |    | OP. CODE |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |              |    |           |    |    |    |    |  |   |  |  |   |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30                | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42           | 43 | 44        | 45 | 46 | 47 | 48 |  |   |  |  |   |  |  |
|       |    |    |    |    |    | P.ØS     |    |    |    |    |    |    |    | 95                |    |    |    |    |    |    |    |    |    |    |    |              |    |           |    |    |    |    |  |   |  |  |   |  |  |
|       |    |    |    |    |    | PA       |    |    |    |    |    |    |    | SAVE              |    |    |    |    |    |    |    |    |    |    |    |              |    |           |    |    |    |    |  |   |  |  |   |  |  |

The printed message would appear at print position 95, left justified and read MESSAGE.

**2.03.04 LOAD PRINT-NUMERIC BASE REGISTER INSTRUCTION**

|                |          |
|----------------|----------|
| <b>OP CODE</b> | <b>A</b> |
| LPNR           | LABEL    |

The Print Numeric Base Register is loaded with the value of the base address for the print mask table. All succeeding print instructions reference this table until another LPNR instruction is executed. The "A" parameter designates the base address of the print mask table.

Mask words are grouped into a table in memory. A Print Numeric Base Register contains the base address or starting word of the table. The location of a mask word is specified by using the relative addresses 0 thru 15.

MASK

Example:

|       |    |    |    |    |    |             |    |    |    |    |    |    |                  |       |    |    |    |    |    |                   |    |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |   |  |  |   |  |  |
|-------|----|----|----|----|----|-------------|----|----|----|----|----|----|------------------|-------|----|----|----|----|----|-------------------|----|----|----|----|----|----------------------|-----------|--|--|--|--|--|---|--|--|---|--|--|
|       |    |    |    |    |    |             |    |    |    |    |    |    |                  |       |    |    |    |    |    |                   |    |    |    |    |    |                      | A         |  |  |  |  |  | B |  |  | C |  |  |
| LABEL |    |    |    |    |    | OP. CODE    |    |    |    |    |    | 27 | 28               | LABEL |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22          | 23 | 24 | 25 | 26 | 29 |    |                  | 30    | 31 | 32 | 33 | 34 | 35 | 36                | 37 | 38 | 39 | 40 | 41 | 42                   | 43        |  |  |  |  |  |   |  |  |   |  |  |
|       |    |    |    |    |    | LPNR        |    |    |    |    |    |    | FORMAT           |       |    |    |    |    |    |                   |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |
|       |    |    |    |    |    | Z           |    |    |    |    |    |    |                  |       |    |    |    |    |    |                   |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |
|       |    |    |    |    |    | FORMAT MASK |    |    |    |    |    |    | DD.D             |       |    |    |    |    |    |                   |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |
|       |    |    |    |    |    | MASK        |    |    |    |    |    |    | ZZZ.DD           |       |    |    |    |    |    |                   |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |
|       |    |    |    |    |    | MASK        |    |    |    |    |    |    | ZZZ, ZZZ, ZZZ.DD |       |    |    |    |    |    |                   |    |    |    |    |    |                      |           |  |  |  |  |  |   |  |  |   |  |  |

The Print-Numeric Base Register is loaded with the word number of the label (FORMAT). Relative address 0 would access the mask word in location FORMAT + 0 or DD.D. Mask number 1 would be ZZZ.DD, (FORMAT + 1), etc.

A maximum of 16 different masks can be referenced relative to the base address value in the Print Numeric Base Register. If more than 16 masks are required, the register must be reloaded with a new value before referencing the masks in the second table (by use of LPNR instruction), and then reloaded with the original value before reusing any of the first set of 16 masks. If fewer than 16 masks are required, those words of memory never referenced as mask numbers may be used for any other purpose.

**2.03.05 MASK WORD (PRINT FORMAT)**

The mask enables printing in varied formats. The mask word consists of control codes and control flags. The control codes are entered into the mask word in digit positions 0-14. They control the printing (or non-printing) and punctuation of each corresponding Accumulator digit. Mask flags are entered into digit position 15 of the mask word, and are used to modify the effects of the control codes.

**TABLE OF MASK CONTROL CODES**

| <u>NAME</u>             | <u>CODE</u> | <u>PRINTING RESULT</u>   |
|-------------------------|-------------|--|
| Digit                   | D           | Accumulator Digit prints unconditionally.  |
| Decimal Point and Digit | .D          | Decimal Point and Accumulator Digit print unconditionally.                             |
| Digit and Decimal Point | D:          | Accumulator Digit and Decimal Point print unconditionally.                             |
| Digit and Comma         | D,          | Accumulator Digit and Comma print unconditionally.                                     |
| Leading Zero Suppress   | Z           | Accumulator Digit prints if non-zero, or if a previous digit to the left was non-zero. |

TABLE OF MASK CONTROL CODES (Continued)

| <u>NAME</u>                              | <u>CODE</u> | <u>PRINTING RESULT</u>  |
|--|-------------|---|
| Leading Zero Suppress and Decimal Point  | Z:          | Accumulator Digit and Decimal Point print if digit is non-zero or if previous digit to the left was non-zero.   |
| Leading Zero Suppress and Comma          | Z,          | Accumulator Digit and Comma print if digit is non-zero or if previous digit to the left was non-zero.   |
| Units of Cents                           | C           | Accumulator Digit prints if significant or if there is a significant digit to the right. Ignore if digit is zero and if significance is not established by either a preceding digit or a digit to the right.                  |
| Tens of Cents                            | .C          | Decimal Point and Digit print if significant or if there is a significant digit to the right.<br><br>Ignore if digit is zero and if significance is not established by either a preceding digit or a digit to the right.      |
| Terminal Zero Suppress                   | X           | Accumulator Digit prints if non-zero, or if any digit to the right in this terminal zero suppression field is non-zero.   |
| Decimal Point and Terminal Zero Suppress | .X          | Decimal Point and Digit print if digit or any succeeding digits in this terminal zero suppression field are non-zero.<br><br>Ignore if the digit and all digits to the right in the terminal zero suppression field are zero. |
| Ignore Digit                             | I           | Digit is ignored, printer does not escape.  |
| Ignore Digit End                         | E           | Digit is ignored, the print instruction is <u>terminated</u> , printer does not escape.   |
| Single Digit Zero Suppress               | S           | Digit prints if non-zero. Escape if zero. Digits to the right and left have no effect.  |

# MASK

## TABLE OF MASK FLAGS

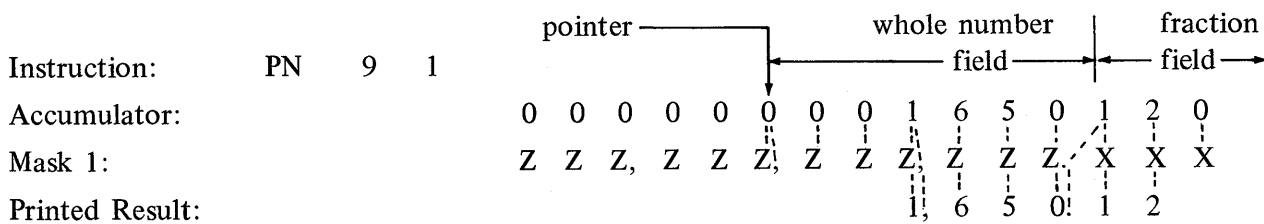
| <u>NAME</u>             | <u>CODE</u> | <u>PRINTING RESULTS</u>   |
|-------------------------|-------------|---|
| Safeguard               | F           | When the Safeguard flag is set, the safeguard symbol (\$) is printed to the left of the most significant digit printed. |
| Suppress Punction       | +           | Print positions where commas or decimal points would normally be inserted are replaced by spaces.                       |
| Punch Leading Zeros     | P           | No effect on printing, causes preceding zeros to punch even though they may not print, starting at the pointer.         |
| Print Condensed Numeric | -           | Monetary punctuation prints without causing printer escapement. Requires PIP hardware.                                  |

### MASK WORD EXAMPLES:

The examples below illustrate the filtering and control that a mask word and its control codes exert over the printing of each accumulator digit.

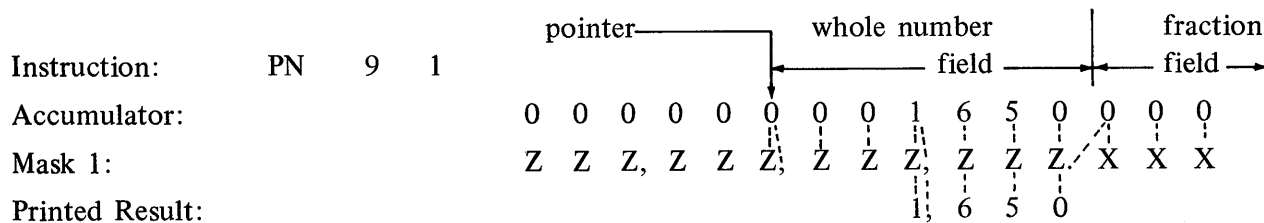
Sample: Printing decimal fractions allowing for a 7-digit whole number and 3 decimal places:

Example 1:



Mask 1 provides 1 field for whole numbers and 1 for decimal fractions: The "Z" and "Z," mask codes establish a "leading zero suppression field" from digit position 3 through the pointer in position 9, and the proper comma punctuation for whole numbers; thus, digit positions 7, 8, & 9 are suppressed because they are not significant. The "X" and ".X" mask codes establish a "terminal zero suppression field" from digit position 0 thru 2 and provide the decimal point, thus digit position zero is suppressed because it is non-significant.

Example 2:



|       |
|-------|
| PN    |
| PNS — |
| PNS + |

Using the same mask word as in example 1, this illustrates the printing effect when there is no significant fraction value. The printed result being only a whole number. Also, as in example 1, digit positions 7, 8 & 9 are suppressed for lack of significance. In both examples, digit positions 10 through 14 are ignored due to the pointer having been specified at position 9.

As we will see due to the PN instruction, the mask need not fill the entire mask word.

**2.03.06 NUMERIC PRINTING INSTRUCTIONS**

Numeric values to be printed must be contained in the Accumulator and can have a maximum of 15 digits. It is not possible to print numeric data directly from memory.

|               | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---------------|----------------|----------|----------|
| PRINT NUMERIC | PN             | 0-14     | 0-15     |

The Print Numeric instruction prints the contents of the Accumulator with the ribbon in the normal (generally black) position regardless of sign. (Unless previously shifted by the RR instruction.)

The “A” field contains the Accumulator digit position number for the most significant digit to be printed. This is independent of the print mask. All positions higher than the digit position specified are ignored and lost from printing. Since the Accumulator digit positions start with 0, to print out a maximum of 5 digits the “A” parameter should contain a 4.

The “B” field of this instruction identifies the print mask to be used during printing. There is a maximum of 16 print masks per LPNR instruction so the B field contains a value from 0-15. The value referenced in the B field is a function of the mask table. (See LPNR instruction).

|   | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---|----------------|----------|----------|
| PRINT NUMERIC, SHIFT RIBBON<br>IF MINUS | PNS—           | 0-14     | 0-15     |
| PRINT NUMERIC, SHIFT RIBBON<br>IF PLUS  | PNS+           | 0-14     | 0-15     |

The PNS— and PNS+ instruction are similar to the PN instruction, the difference being:

1. The PNS— instruction shifts the ribbon if the sign of the Accumulator is negative. The PNS— instruction also allows for Print in Place Capability. The ability to print in place is actuated by the insertion of a Dash (—) in digit position 15 of the mask word. This will print the comma (,) and period (.) without letting the printer actually escape the 1/10 inch normally permitted.

|       |
|-------|
| PN    |
| PNS - |
| PNS + |

2. The PNS+ instruction shifts the ribbon if the sign of the Accumulator is positive.

Example 1:

| CODE | SEQUENCE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  | OP. CODE |  |  |  |  | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|----------|--|--|--|--|----------------------|-----------|--|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|---|--|--|--|--|
|      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      | A         |  |  |  |  |  |  |  |  |  | B                 |  |  |  |  | C |  |  |  |  |
|      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      | LABEL     |  |  |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |  |   |  |  |  |  |
|      | 11       | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32    | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
| I    |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>             |
|--------------|----------------|----------|----------|----------------------------|
|              | NK             | 5        | 3        | Enable Numeric keys        |
|              | PN             | 8        | 0        | Print Accumulator contents |

The contents of the Accumulator are printed beginning with digit position 8 and with the format dictated by print mask 0.

Example 2:

| CODE | SEQUENCE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | LABEL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  | OP. CODE |  |  |  |  | FIELD<br>LEN-<br>GTH | PARAMETER |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|----------|--|--|--|--|----------------------|-----------|--|--|--|--|--|--|--|--|--|-------------------|--|--|--|--|---|--|--|--|--|
|      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      | A         |  |  |  |  |  |  |  |  |  | B                 |  |  |  |  | C |  |  |  |  |
|      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      | LABEL     |  |  |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |  |   |  |  |  |  |
|      | 11       | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32    | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 4  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |
|      |          |    | 0  | 5  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |          |  |  |  |  |                      |           |  |  |  |  |  |  |  |  |  |                   |  |  |  |  |   |  |  |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>          |
|--------------|----------------|----------|----------|-------------------------|
|              | NKR            | 5        | 3        | Enable Reverse Entry    |
|              | PNS-           | 8        | 0        | Print Shift if negative |

Printing will occur as in the above example, but the ribbon will Shift if the Accumulator "minus" flag is set.



|     |      |
|-----|------|
| PC  | PC - |
| PCP | PC + |

**2.03.07 SINGLE CHARACTER PRINT INSTRUCTIONS**

|                 |                |                         |
|-----------------|----------------|-------------------------|
|                 | <u>OP CODE</u> | <u>A</u>                |
| PRINT CHARACTER | PC             | Character to be printed |

This instruction unconditionally prints the character specified in the "A" field. If the "A" field is blank, the instruction causes a single printer space operation. The PC instruction prints with the ribbon in the normal position (unless previously shifted. See RR instruction).

|                                 |                |                         |
|---------------------------------|----------------|-------------------------|
|                                 | <u>OP CODE</u> | <u>A</u>                |
| PRINT CHARACTER PREVIOUS RIBBON | PCP            | Character to be printed |

The PCP instruction will print a character with the same ribbon position that was used on the last print operation.

|   |                |                         |
|---|----------------|-------------------------|
|   | <u>OP CODE</u> | <u>A</u>                |
| PRINT CHARACTER IF ACCUMULATOR MINUS, PREVIOUS RIBBON | PC-            | Character to be printed |
| PRINT CHARACTER IF ACCUMULATOR PLUS, PREVIOUS RIBBON  | PC+            | Character to be printed |

Printing of these instructions is dependent upon the Accumulator sign flag (+ or -). The character specified in the "A" field is printed according to the following conditions:

1. PC- Print if Accumulator negative (i.e., sign flag set); do not print if plus.
2. PC+ Print if Accumulator positive (i.e., sign flag reset); do not print if negative.

Example:

| SEQUENCE |    | LABEL |    | OP. CODE |    | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|----------|----|-------|----|----------|----|---------------|-----------|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
|          |    |       |    |          |    |               | A         |    |    |    |    |    | B              |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|          |    |       |    |          |    |               | LABEL     |    |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
| 11       | 12 | 13    | 14 | 15       | 16 | 17            | 18        | 19 | 20 | 21 | 22 | 23 | 24             | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |
|          |    | 0     | 1  |          |    |               |           |    |    |    | N  | K  | R              |    |    |    |    | 8  |    |    |    |    |    |    |    |    |    | 3  |    |    |    |    |    |    |    |    |  |
|          |    | 0     | 2  |          |    |               |           |    |    |    | P  | N  | S              | -  |    |    |    |    | 1  | 0  |    |    |    |    |    |    |    | 0  |    |    |    |    |    |    |    |    |  |
|          |    | 0     | 3  |          |    |               |           |    |    |    | P  | C  | +              |    |    |    |    |    | +  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|          |    | 0     | 4  |          |    |               |           |    |    |    | P  | C  | -              |    |    |    |    |    | -  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |

RR

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>       |
|--------------|----------------|----------|----------|----------------------|
|              | NKR            | 8        | 3        | Allow negative entry |
|              | PNS-           | 10       | 0        | Print amount         |
|              | PC+            | +        |          | Print if positive    |
|              | PC-            | -        |          | Print if negative    |

If the Accumulator contains a positive quantity, a "+" character will be printed. A negative content would produce a "-" character.

**2.03.08 RIBBON SHIFT INSTRUCTION**

Printing of data normally is with the ribbon color black, except for certain print instructions that cause minus amounts to print in red. However, a ribbon shift instruction is also provided to change the normal color of printing.

RED RIBBON

OP CODE

RR

The RR instruction is used to change the ribbon color of only the next printing instruction. The ribbon color will be opposite to the color normally expected from the data and type of the next print instruction.

Example 1:

| CODE | SEQUENCE       | LABEL             | OP. CODE       | FIELD<br>LEN-<br>GTH | PARAMETER         |             |             |    |             |  |  |                   |  |  |   |  |  |
|------|----------------|-------------------|----------------|----------------------|-------------------|-------------|-------------|----|-------------|--|--|-------------------|--|--|---|--|--|
|      |                |                   |                |                      | A                 |             |             |    |             |  |  | B                 |  |  | C |  |  |
|      |                |                   |                |                      | LABEL             |             |             |    |             |  |  | + OR -<br>INC/REL |  |  |   |  |  |
| I    | 11 12 13 14 15 | 16 17 18 19 20 21 | 22 23 24 25 26 | 27 28                | 29 30 31 32 33 34 | 35 36 37 38 | 39 40 41 42 | 43 | 44 45 46 47 |  |  |                   |  |  |   |  |  |
|      | 0, 1           |                   | <b>NKR</b>     |                      | <b>8</b>          |             | <b>3</b>    |    |             |  |  |                   |  |  |   |  |  |
|      | 0, 2           |                   | <b>RR</b>      |                      |                   |             |             |    |             |  |  |                   |  |  |   |  |  |
|      | 0, 3           |                   | <b>PN</b>      |                      | <b>5</b>          |             | <b>3</b>    |    |             |  |  |                   |  |  |   |  |  |
|      | 0, 4           |                   | <b>PN</b>      |                      | <b>5</b>          |             | <b>3</b>    |    |             |  |  |                   |  |  |   |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>       |
|--------------|----------------|----------|----------|----------------------|
|              | NKR            | 8        | 3        | Allow negative entry |
|              | RR             |          |          | Reverse Ribbon       |
|              | PN             | 5        | 3        |                      |
|              | PN             | 5        | 3        |                      |

The Accumulator contents would print according to the PN 5 3 instruction but the ribbon would change to the opposite color. The second PN 5 3 would not be affected by the RR instruction.

RR

Example 2:

| CODE | SEQUENCE |    |    |    |    | LABEL |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |  |
|------|----------|----|----|----|----|-------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|-----------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|--|
|      |          |    |    |    |    |       |    |    |    |    |    |          |    |    |    |    |    |                      | A         |    |    |    |    |    | B                 |    |    | C  |    |    |    |    |    |    |    |    |    |  |
|      |          |    |    |    |    |       |    |    |    |    |    |          |    |    |    |    |    |                      | LABEL     |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |  |
|      | 11       | 12 | 13 | 14 | 15 | 16    | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 | 28                   | 29        | 30 | 31 | 32 | 33 | 34 | 35                | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |
|      |          | 0  | 1  |    |    |       |    |    |    |    |    | N        | K  | R  |    |    | 8  |                      |           |    |    |    |    |    |                   |    |    |    | 3  |    |    |    |    |    |    |    |    |  |
|      |          | 0  | 2  |    |    |       |    |    |    |    |    | R        | R  |    |    |    |    |                      |           |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |  |
|      |          | 0  | 3  |    |    |       |    |    |    |    |    | P        | N  | S  | -  |    | 10 |                      |           |    |    |    |    |    |                   |    |    |    | 0  |    |    |    |    |    |    |    |    |  |
|      |          | 0  | 4  |    |    |       |    |    |    |    |    | R        | R  |    |    |    |    |                      |           |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |  |
|      |          | 0  | 5  |    |    |       |    |    |    |    |    | P        | N  | S  | +  |    | 10 |                      |           |    |    |    |    |    |                   |    |    |    | 0  |    |    |    |    |    |    |    |    |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|--------------|----------------|----------|----------|------------------|
|              | NKR            | 8        | 3        | Allow negative   |
|              | RR             |          |          | Reverse Ribbon   |
|              | PNS-           | 10       | 0        | Shift Ribbon "-" |
|              | RR             |          |          | Reverse Ribbon   |
|              | PNS+           | 10       | 0        | Shift Ribbon "+" |

The effects of the PNS- and PNS+ instruction are reversed.

|    |
|----|
| OC |
| CC |

**2.04 – FORMS CONTROL INSTRUCTIONS**

**2.04.01 FORMS HANDLER – OPEN AND CLOSE INSTRUCTION**

|                    | <u>OP CODE</u> | <u>A</u>                 |
|--------------------|----------------|--------------------------|
| Open Forms Handler | OC             | 0-255 rear feed handler  |
|                    | OC             | BLANK front feed handler |

The OC instruction is used to open the forms handler mechanism in order to permit the insertion or removal of a completed unit document. The A parameter is blank for front feed styles. For rear feed styles of the L/TC the A parameter of the OC instruction specifies the number of lines the left forms mechanism will advance when the handler mechanism is next closed.

This closing may be from any of the following sources:

1. The execution of a PN or PA instruction of any type.
2. The entering of alpha information at a TK instruction. If a TK instruction were terminated by an OCK without the entering of alpha data, the handler mechanism would not close.
3. A CC instruction.
4. Manual depression of the open/close key on the keyboard.

When programing for automatic alignment of rear-fed unit documents, the number that must be placed in the OC parameter must be 3 greater than the line number of the first actual line of print.

To align a unit document to line number 14

| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>   |
|----------------|----------|------------------|
| OC             | 17       | Will align to 14 |

Although the form aligns to line 14, the Count Register contains 17. Thus, it may be desirable to reload the Count Register with 14 before any further vertical spacing is performed.

|                     | <u>OP CODE</u> |
|---------------------|----------------|
| Close Forms Handler | CC             |

The CC instruction closes the forms handler. This instruction generally is not required since execution of any print instruction or depression of a typing key during a type instruction will automatically close the forms handler

If the handler is open as the result of executing an OC instruction, when the CC instruction is executed, the Left Forms mechanism will advance the number of times specified by the OC instruction.

|      |      |      |      |      |
|------|------|------|------|------|
| LLCR | LRCR | LLLR | LRLR |      |
| AL   | AR   | ALR  | ALTO | ARTO |

#### 2.04.02 PLATEN CONTROL REGISTER INSTRUCTIONS

|                                  | <u>OP CODE</u> | <u>A</u> |
|----------------------------------|----------------|----------|
| LOAD LEFT PLATEN COUNT REGISTER  | LLCR           | 0-255    |
| LOAD LEFT PLATEN LIMIT REGISTER  | LLLR           | 0-255    |
| LOAD RIGHT PLATEN COUNT REGISTER | LRCR           | 0-255    |
| LOAD RIGHT PLATEN LIMIT REGISTER | LRLR           | 0-255    |

The programmer is provided with four platen control registers to control vertical spacing. These are the Left and Right Forms Count Registers, and the Left and Right Limit Registers. In addition, there is a Forms Limit Flag.

A forms count register is associated with each platen advance mechanism. This register is automatically incremented by 1 each time the respective (left or right) platen is advanced a line either programmatically or by use of the Line Advance Key.

A forms limit register is also associated with each platen advance mechanism. This register contains a limit to which the forms count register can be compared.

The LLLR and LRLR preset the forms limit registers to a specified line. The count register will be set to 1 (not 0) on the next line advance after the respective limit and count registers are equal.

On the line advance following when the count register equals the corresponding limit register, the forms limit flag is set. The limit flag becomes reset on the next line advance.

LLLR = 50

LLCR = 50

On the next line advance the left count register equals 1 and the Forms Limit Flag will be set. The next line advance (2nd after LLCR = LLLR) resets the flag.

The execution of a LLCR or LRCR will reload the appropriate count register. The count register is not incremented when the platen is advanced by the platen twirlers.

The LLLR and LRLR instructions load the Left and Right Platen Limit Registers respectively with the contents of the "A" field.

#### 2.04.03 LINE ADVANCE INSTRUCTIONS

|                         | <u>OP CODE</u> | <u>A</u> |
|-------------------------|----------------|----------|
| ADVANCE LEFT PLATEN     | AL             | 0-255    |
| ADVANCE RIGHT PLATEN    | AR             | 0-255    |
| ADVANCE BOTH PLATENS    | ALR            | 0-255    |
| ADVANCE LEFT PLATEN TO  | ALTO           | 1-255    |
| ADVANCE RIGHT PLATEN TO | ARTO           | 1-255    |

|      |      |      |      |      |
|------|------|------|------|------|
| LLCR | LLLR | LRCR | LRLR |      |
| AL   | AR   | ALR  | ALTO | ARTO |

The AL, AR, and ALR instructions advance the form the number of lines specified by the "A" parameter. These provide a single line advance with a maximum advance of 255 lines. The vertical spaces occur in the 1/6 inch increments. The respective count register is incremented by 1 for each single line advance.

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| AL             | 1        |

The form will advance 1 line. The Count Register will be incremented by 1.

The ALTO and ARTO instructions advance a form until the associated count register is equal to the value of the "A" field. If the Count Register equals the line number specified in the ALTO or ARTO instruction prior to its execution, no advance occurs. Specifying "0" or an integer larger than the contents of the Limit Register in the "A" parameter of the ALTO/ARTO instruction is a programming error. This will result in a continuous search for a line number that does not exist.

- To determine the number of lines which will be advanced, subtract the Count Register from the value of the "A" parameter in the ALTO or ARTO instruction. If positive, this will be the number of lines advanced. If negative, assume this number is positive, then subtract from the value of the Limit Register to ascertain the number of lines advanced.

|    |                |          |                          |
|----|----------------|----------|--------------------------|
| a. | <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>           |
|    | LLLR           | 255      | Load Left Limit Register |
|    | LLCR           | 20       | Load Left Count Register |
|    | ALTO           | 3        | Advance to line 3        |

Value of ALTO parameter – Value of Count Register

$$3 \quad - \quad 20 \quad = \quad -17$$

Since negative assume positive (i.e.,  $-17 = 17$ )

Value of Limit Register – 17 =

$$255 \quad - \quad 17 \quad = \quad 238$$

There will be an advance of 238 lines.

|    |                |          |                          |
|----|----------------|----------|--------------------------|
| b. | <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>           |
|    | LLLR           | 255      | Load Left Limit Register |
|    | LLCR           | 20       | Load Left Count Register |
|    | ALTO           | 25       | Advance to Line 25       |

Value of ALTO parameter – Value of Count Register =

$$25 \quad - \quad 20 \quad = \quad 5$$

Since resultant is positive, there will be 5 line advance.

|      |     |      |      |      |
|------|-----|------|------|------|
| LLCR | LLL | LRCR | LRLR |      |
| AL   | AR  | ALR  | ALTO | ARTO |

|    |                |          |
|----|----------------|----------|
| 2. | <u>OP CODE</u> | <u>A</u> |
|    | LLL            | 30       |
|    | ALTO           | 5        |

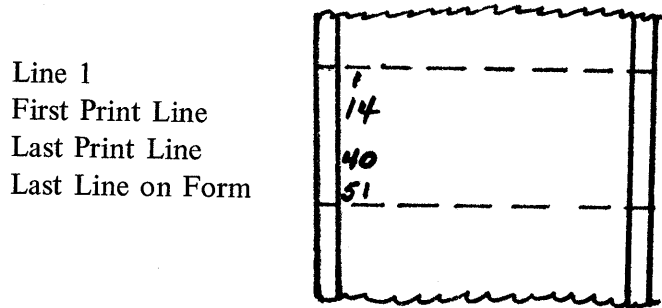
Assume contents of Left Count Register = 20, when ALTO command is executed. This is an example of the type of programming employed when using pin fed continuous forms with the requirement that the program automatically advance from the last line on one form to the first line of a new form.

The form advances 10 lines, then the LLLR = LLCR, on the next line advance the Count Register is set to 1. Advancing continues for 4 more lines to line 5 of the new form. In this case, the last line on the form would be line 30.

Another method of continuous forms programming utilizes the forms limit flag.

Example:

Suppose we have the following form:



The following programming will advance the form automatically when the forms limit flag is set.

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|----------------|----------|----------|----------|
| LLL            | 40       |          |          |
| LLCR           | 0        |          |          |
| AL             | 1        |          |          |
| EX             | T        | L        | 1        |
| ALTO           | 17       |          |          |
| LLL            | 6        |          |          |

The following illustrates the use of the Limit Register to enable the program to know when 40 lines have been filled on the invoice. The total length of the invoice is 8½ inches (8.5 x 6 = 51 lines). The first print line is 14 as measured from the top of the form.

|      |       |      |      |      |
|------|-------|------|------|------|
| LLCR | LLLRR | LRCR | LRLR |      |
| AL   | AR    | ALR  | ALTO | ARTO |

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| LLLRR          | 40       |
| LLCR           | 37       |
| OC             | 17       |
| TK             | 10       |

When the forms handler is closed, the form will advance 17 lines. The first three lines increment the Count Register to 40, the next advance will set the Count Register to 1. After an advance of the remaining 13 lines, the Count Register will be at 14. This is the actual first print line, and the number wanted in the Count Register.



|     |
|-----|
| ADA |
| ADM |

2.05 – ARITHMETIC INSTRUCTIONS

2.05.01 ADDITION INSTRUCTION

|                    |                |          |
|--------------------|----------------|----------|
|                    | <u>OP CODE</u> | <u>A</u> |
| ADD TO ACCUMULATOR | ADA            | LABEL    |
| ADD TO MEMORY      | ADM            | LABEL    |

The ADA instruction provides for adding the contents of a memory location, specified by the A field to the contents of the Accumulator. The resultant sum is placed in the Accumulator leaving the memory location undisturbed.

The ADM instruction provides for adding the contents of the Accumulator to the contents of the memory location specified in the A field. The resultant sum is placed in memory location A leaving the Accumulator undisturbed.

The overflow flag is set if an overflow occurs and reset if there is no overflow.

The ADA and ADM commands cannot be used to move alpha data, even if the receiving location is clear.

Example 1:

| LABEL |    | OP. CODE |    |    |       | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|-------|----|----------|----|----|-------|----------------------|-----------|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|
|       |    |          |    |    |       |                      | A         |    |    |                   |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    | LABEL |                      |           |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
| 16    | 17 | 18       | 19 | 20 | 21    | 22                   | 23        | 24 | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |

|                |          |          |  |
|----------------|----------|----------|--|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
| ADA            | AREA     |          | Add to Accumulator the contents of Area, content of Area is unchanged. |

Example 2:

| LABEL |    | OP. CODE |    |    |       | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|-------|----|----------|----|----|-------|----------------------|-----------|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|
|       |    |          |    |    |       |                      | A         |    |    |                   |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    | LABEL |                      |           |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
| 16    | 17 | 18       | 19 | 20 | 21    | 22                   | 23        | 24 | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |       |                      |           |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |

|                |          |          |  |
|----------------|----------|----------|--|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
| ADM            | AREA     |          | Add to memory location Area contents of Accumulator leaving Accumulator unchanged. |

|     |
|-----|
| ADK |
| CLM |
| CLA |

**2.05.02 ADD CONSTANT TO ACCUMULATOR INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| ADK            | 0-14     | 0-9      |

The ADK instruction provides algebraic addition of the digit contained in the B field to the digit in the Accumulator position specified by the A field, with carries propagated in succeeding high order digits.

The Special (S), per thousand (M) and per hundred (C) flags are unconditionally reset.

The sign flag is reset (+) if the result is positive or set (-) if negative.

The overflow flag is set if an overflow occurs and reset if there is no overflow.

Example:

| LABEL |    | OP. CODE |    |    |                | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|-------|----|----------|----|----|----------------|---------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
|       |    |          |    |    |                |               | A         |    |    |    |    | B  |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
| LABEL |    |          |    |    | + OR - INC/REL |               |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
| 16    | 17 | 18       | 19 | 20 | 21             | 22            | 23        | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |
|       |    |          |    |    |                | A             | D         | K  |    |    |    | 6  |    |    |    |    |    |    |    |    |    |    | 3  |    |    |    |    |    |    |    |    |  |

|                |          |          |   |
|----------------|----------|----------|---|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                                |
| ADK            | 6        | 3        | Add 3 to digit position 6 in the Accumulator. |

**2.05.03 CLEAR INSTRUCTIONS**

|                                       |                |          |          |
|---------------------------------------|----------------|----------|----------|
| CLEAR MEMORY WORD                     | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|                                       | CLM            | LABEL    |          |
| CLEAR ACCUMULATOR AND INSERT CONSTANT | CLA            | 0-15     | 0-15     |

The CLM instruction will clear the 16 digits of the memory location specified in the A field.

The CLA instruction sets all 16 digits of the Accumulator to zero, thus resetting the four Accumulator flags (M, C, special, and sign); it places the digit specified by the B field in the digit position of the Accumulator specified by the A field.

It is important to notice that the B parameter although expressed as 0-15 on the coding form, is placed in the Accumulator as a hexadecimal digit (0-F) rather than two decimal digits.

Arithmetic operations can only use the values from 0-9 in any digit position. Any values over 9 will not arithmetically combine.

INK

Example 1:

| LABEL |    | OP. CODE |    |    |       |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------|----|----------------------|-----------|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |       |    |                      | A         |    |                   |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | LABEL |    |                      |           |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21    | 22 | 23                   | 24        | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    |          |    |    | CLM   |    |                      |           |    | AREA              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| OP CODE | A    | B | REMARKS   |
|---------|------|---|---|
| CLM     | AREA |   | The Memory location called Area will contain all zeros. |

Example 2:

| LABEL |    | OP. CODE |    |    |       |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------|----|----------------------|-----------|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |       |    |                      | A         |    |                   |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | LABEL |    |                      |           |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21    | 22 | 23                   | 24        | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    |          |    |    | CLA   |    |                      |           |    | 0                 |    |    |    |    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| OP CODE | A | B | REMARKS   |
|---------|---|---|---|
| CLA     | 0 | 0 | The Accumulator contains zeros in positions 0-15. |

**2.05.04 INSERT CONSTANT IN ACCUMULATOR INSTRUCTION**

|                                | OP CODE | A    | B    |
|--------------------------------|---------|------|------|
| INSERT CONSTANT IN ACCUMULATOR | INK     | 0-15 | 0-15 |

The INK instruction inserts the digit specified by the B field in the digit position of the Accumulator specified by the A field. The remaining digit positions are unaffected.

Similar to the CLA instruction the B parameter field in this instruction also permits entry of a value from 0-15. Again this is a hexadecimal value rather than a decimal value.

Arithmetic operations can only use the values 0-9 in any digit position. Any values over 9 (i.e., A-F) will not arithmetically combine.

Example:

| LABEL |    | OP. CODE |    |    |       |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------|----|----------------------|-----------|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |       |    |                      | A         |    |                   |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | LABEL |    |                      |           |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21    | 22 | 23                   | 24        | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    |          |    |    | INK   |    |                      |           |    | 0                 |    |    |    |    | 3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |



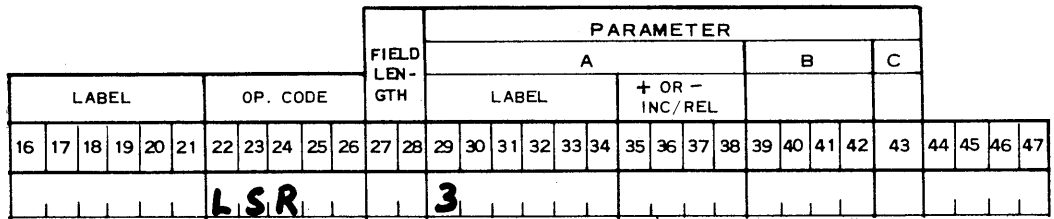
| OP CODE | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|---------|----------|----------|---|
| INK     | 0        | 3        | The digit 3 will be placed in Accumulator digit position 0 replacing the previous contents of Accumulator digit position 0. |

**2.05.05 MULTIPLICATION AND DIVISION INSTRUCTIONS**

**LOAD SHIFT REGISTER**

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| LSR            | 0-15     |

The LSR instruction provides for loading the multiply-divide shift register with the contents of the A field. The shift register must be loaded prior to the execution of a Multiply or Divide instruction. The shift register will contain the value loaded until a subsequent load shift register command is executed. For multiplication, the shift register designates the number of places the product is shifted right after multiplication. The shifted off digits are lost, the remaining digits set in the Accumulator as the product. Division will be carried out to the number of places specified in the shift register. These operations take into account the shift register even though it is not loaded immediately preceding each MUL or DIV instruction. The contents of the shift register must be changed only when the shift requirements are changed.



| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>             |
|----------------|----------|----------------------------|
| LSR            | 3        | Load shift register with 3 |

**Computing the Value of the Shift Register**

FOR MULTIPLICATION—To compute the value which must be loaded in the shift register, subtract the desired number of decimal places in the final result from the sum of decimal places in the multiplier and multiplicand.

|                                   |   |                                     |   |                             |   |                            |
|-----------------------------------|---|-------------------------------------|---|-----------------------------|---|----------------------------|
| Number of places<br>in multiplier | + | Number of places<br>in multiplicand | - | Desired Number<br>of places | = | Value of<br>Shift Register |
| 100.00                            |   | .25                                 |   |                             |   |                            |
| 2                                 | + | 2                                   | - | 1                           | = | 3                          |

Accumulator contains 250 in digit positions 0-2, when printed with one decimal this becomes 25.0.

|     |
|-----|
| MUL |
|-----|

FOR DIVISION—The value to be loaded into the Shift Register can be determined with a knowledge of the assumed decimal places needed in the quotient as well as the divisor and dividend.

|                                      |   |                                    |   |                                     |   |                            |
|--------------------------------------|---|------------------------------------|---|-------------------------------------|---|----------------------------|
| Assumed decimal<br>places in divisor | + | Assumed decimal<br>places quotient | = | Assumed decimal<br>places dividends | = | Value of<br>Shift Register |
| .25                                  |   | 100.00                             |   | 25.0                                |   |                            |
| 2                                    | + | 2                                  | = | 1                                   | = | 3                          |

|          |                |          |
|----------|----------------|----------|
|          | <u>OP CODE</u> | <u>A</u> |
| MULTIPLY | MUL            | LABEL    |

The multiply instruction provides for multiplying the contents of the Accumulator by the contents of the memory location specified in the A parameter. The product is shifted right the number of places specified in the multiply – divide shift register, causing the shifted off digits to be lost. The next 15 low order digits are placed in the Accumulator as the product.

If the Accumulator and the memory location in the A parameter have identical signs, the sign of the product is positive [Accumulator sign flag is reset (+)]. With unlike signs, the product is assigned a negative sign [Accumulator sign flag is set (-)].

Both the Accumulator and the memory location can contain a maximum of 15 digits each. If the product contains more than 15 digits after shifting occurs, the excess number of digits are lost and the overflow flag is set. The flag is reset otherwise. (In the event of an overflow there is not an indication light).

If the possibility of an overflow condition exists, the program should provide for interrogating the flag to determine if a corrective routine should be employed.

The number of significant digits in the multiplier (memory location in the A field) determines the length of time for the execution of the multiplication instruction. The number of digits in the multiplicand (Accumulator) has no effect on the timing.

Example:

| LABEL             |  | OP. CODE       |  | FIELD<br>LEN-<br>GTH | PARAMETER    |  |                   |             |    |             |
|-------------------|--|----------------|--|----------------------|--------------|--|-------------------|-------------|----|-------------|
|                   |  |                |  |                      | A            |  | B                 | C           |    |             |
| 16 17 18 19 20 21 |  | 22 23 24 25 26 |  | 27 28                | LABEL        |  | + OR -<br>INC/REL | 39 40 41 42 | 43 | 44 45 46 47 |
|                   |  | <b>MUL</b>     |  |                      | <b>PRICE</b> |  |                   |             |    |             |

|                |          |                               |
|----------------|----------|-------------------------------|
| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>                |
| MUL            | PRICE    | Multiply Accumulator by PRICE |

|      |
|------|
| MULR |
| DIV  |

**MULTIPLY AND ROUND**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| MULR           | LABEL    |

The MULR instruction is the same as the MUL instruction except that a 5 is added to the last digit which was shifted off in the product. The product contained in the Accumulator is increased by 1 (decreased if -) if the last digit shifted off was greater than or equal to 5. If the shift register value is zero, there will be no rounding.

**DIVIDE**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| DIV            | LABEL    |

The DIV instruction divides the contents of the Accumulator by the contents of the memory location specified in the A field. The quotient is placed in the Accumulator. After division has been carried out, the number of decimal places specified in the shift register, any remainder is placed in working memory (in the control area). (See REM instruction.)

Example:

| LABEL |    | OP. CODE |    |    |    |                | FIELD LENGTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----------------|--------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |                |              | A         |    |    |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| LABEL |    |          |    |    |    | + OR - INC/REL |              |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22             | 23           | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |
| DIU   |    |          |    |    |    | TOTAL          |              |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|                |          |                             |
|----------------|----------|-----------------------------|
| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u>              |
| DIV            | TOTAL    | Divide Accumulator by TOTAL |

Both the Dividend and the Divisor may contain up to 15 digits. If the signs of the operands are alike, the sign of the quotient is positive (accumulator sign flag is reset +): if the signs are unlike, the sign of the quotient is negative (accumulator sign flag is set -). The remainder is always positive.

Example 1:

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| LSR            | 5        |
| DIV            | 200      |

|                                |   |         |   |
|--------------------------------|---|---------|---|
| Accumulator (dividend)         | = | 100     |   |
| Memory location 200 (divisor)  | = | 3       |   |
| Multiply-Divide Shift Register | = | 5       |   |
| Accumulator (quotient)         | = | 3333333 | = { printed with decimal = 33.33333<br>printing of decimal provided by<br>print mask. |
| Remainder                      | = | 1       |   |

# DIV

The division process treats the contents of the Accumulator and the specified memory location as whole numbers, even though they may have “assumed” decimal points; for example:  $6 \wedge 25 \div 5 \wedge 00$  produces a quotient of 1 and a remainder of 125 if the shift register has a zero value:

|                               |   |     |   |
|-------------------------------|---|-----|---|
| Accumulator (dividend)        | = | 625 |   |
| Memory location 200 (divisor) | = | 500 |   |
| Shift register                | = | 0   |   |
| Accumulator (quotient)        | = | 1   | } |
| Remainder                     | = | 125 |   |

= could be printed as “1” or “1.”. Since it is in first digit position, any other decimal places shown in printing would require shifting it left such as to permit “1.0000”

Thus, since division halts once the dividend can no longer be divided, the shift register must contain a value equal to the number of decimal places desired beyond what the “whole numbers” themselves would provide. In the above example, by giving the shift register a value of 4, the quotient reflects the “assumed” decimal values:

|                               |   |       |                                 |
|-------------------------------|---|-------|---------------------------------|
| Accumulator (dividend)        | = | 625   |                                 |
| Memory location 200 (divisor) | = | 500   |                                 |
| Shift register                | = | 4     |                                 |
| Accumulator (quotient)        | = | 12500 | (printed with decimal = 1.2500) |
| Remainder                     | = | 0     |                                 |

The value to be loaded into the shift register can be determined in the following manner with a knowledge of the “assumed” decimal places needed in the quotient as well as the dividend and divisor:

$$\left\{ \begin{array}{l} \text{assumed decimal} \\ \text{places in} \\ \text{DIVISOR} \end{array} \right\} \text{ PLUS } \left\{ \begin{array}{l} \text{assumed decimal} \\ \text{places in} \\ \text{QUOTIENT} \end{array} \right\} \text{ LESS } \left\{ \begin{array}{l} \text{assumed decimal} \\ \text{places in} \\ \text{DIVIDEND} \end{array} \right\} = \text{Value of SHIFT REGISTER}$$

Ex:  $5 \wedge 00$                        $1 \wedge 2500$                        $6 \wedge 25$

2                                      +                                      4                                      2                                      =                                      4

If the quotient after final shift exceeds 15 digits, the overflow flag is set, otherwise the flag is reset. The size of the quotient can be estimated and a prediction of possible overflow made if the following rule is used:

“Add the MAXIMUM size DIVIDEND to the Value of the SHIFT REGISTER plus 1, subtract the MINIMUM size DIVISOR and that equals the MAXIMUM size Quotient possible.”

The rule is in terms of the number of significant digits expected in each operand including intervening and terminal zeros, and without regard to “assumed” decimal places.

|     |
|-----|
| SUA |
| SUK |

Example 2:

| Maximum size<br>DIVIDEND | + | 1 | + | Value of<br>SHIFT REG. | - | Minimum size<br>DIVISOR | = | Maximum size<br>QUOTIENT |
|--------------------------|---|---|---|------------------------|---|-------------------------|---|--------------------------|
| Ex: (9999)               |   |   |   | (2)                    |   | (1)                     |   | (999900)                 |
| 4                        | + |   | + | 2                      | - | 1                       | = | 6                        |
| Ex: (9999)               |   |   |   | (3)                    |   | (100)                   |   | (99990)                  |
| 4                        | + |   | + | 3                      | - | 3                       | = | 5                        |

When an overflow occurs, the division is halted and the result in the Accumulator is meaningless (reflects some stage of partial quotient development).

**2.05.06 SUBTRACT INSTRUCTIONS**

|                                    | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|------------------------------------|----------------|----------|----------|
| SUBTRACT FROM ACCUMULATOR          | SUA            | LABEL    |          |
| SUBTRACT CONSTANT FROM ACCUMULATOR | SUK            | 0-14     | 0-9      |

The SUA instruction provides for subtracting the contents of the memory location specified by the A field from the contents of the Accumulator. The difference is placed in the Accumulator leaving memory location A undisturbed.

The SUK instruction provides algebraic subtraction of the digit contained in the B field from the digit in the Accumulator position stated in the A field with carries propagated in succeeding high order digits. (The special (S), per thousand (M), and per hundred (C) flags are unconditionally reset.) The overflow flag is set if an overflow occurs and reset if there is no overflow.

Example 1:

| LABEL |    | OP. CODE |    |    |                   |      | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------------------|------|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |                   |      |                      | A         |    |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | + OR -<br>INC/REL |      |                      |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21                | 22   | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| SUA   |    |          |    |    |                   | AREA |                      |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| SUA            | AREA     |          | Subtract the contents of the memory location called Area from the Accumulator. |

Example 2:

| LABEL |    | OP. CODE |    |    |                   |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------------------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |                   |    |                      | A         |    |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | + OR -<br>INC/REL |    |                      |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21                | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| SUK   |    |          |    |    |                   | 0  |                      |           |    |    |    |    |    |    |    | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |



SUM

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|----------------|----------|----------|---|
| SUK            | 0        | 2        | Algebraic subtraction of the integer 2 from the 0 digit position in the Accumulator |

SUBTRACT FROM MEMORY

| <u>OP CODE</u> | <u>A</u> | <u>REMARKS</u> |
|----------------|----------|----------------|
| SUM            | LABEL    |                |

The SUM instruction provides for subtracting the contents of the Accumulator from the contents of the memory location specified in the A parameter.

The difference is placed in the given memory location, leaving the Accumulator unchanged.

Example:

| LABEL             |  | OP. CODE       |  |  |  |  | FIELD<br>LEN-<br>GTH | PARAMETER         |  |  |  |  |  |  |  |                   |  |  |  |             |  |  |  |                |  |  |  |
|-------------------|--|----------------|--|--|--|--|----------------------|-------------------|--|--|--|--|--|--|--|-------------------|--|--|--|-------------|--|--|--|----------------|--|--|--|
|                   |  |                |  |  |  |  |                      | A                 |  |  |  |  |  |  |  | B                 |  |  |  | C           |  |  |  |                |  |  |  |
| 16 17 18 19 20 21 |  | 22 23 24 25 26 |  |  |  |  | 27 28                | LABEL             |  |  |  |  |  |  |  | + OR -<br>INC/REL |  |  |  |             |  |  |  |                |  |  |  |
| 16 17 18 19 20 21 |  | 22 23 24 25 26 |  |  |  |  | 27 28                | 29 30 31 32 33 34 |  |  |  |  |  |  |  | 35 36 37 38       |  |  |  | 39 40 41 42 |  |  |  | 43 44 45 46 47 |  |  |  |
|                   |  | <b>SUM</b>     |  |  |  |  |                      | <b>AREA</b>       |  |  |  |  |  |  |  |                   |  |  |  |             |  |  |  |                |  |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| SUM            | AREA     |          | Subtract the contents of the Accumulator from the memory location called Area. |

|     |
|-----|
| TRA |
| TRM |

**2.06 – DATA MOVEMENT INSTRUCTIONS**

**2.06.01 TRANSFER INSTRUCTIONS**

|                             | <u>OP CODE</u> | <u>A</u> |
|-----------------------------|----------------|----------|
| TRANSFER TO THE ACCUMULATOR | TRA            | LABEL    |
| TRANSFER TO MEMORY          | TRM            | LABEL    |

The TRA instruction provides for transferring the contents of the memory location specified in the A field to the Accumulator, keeping the contents of the memory location unchanged.

The TRM instruction provides for transferring the contents of the Accumulator to the memory location specified by the A field. There is no change in the contents of the Accumulator.

Example 1:

| LABEL |    | OP. CODE   |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER   |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|------------|----|----|----|----|----------------------|-------------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |            |    |    |    |    |                      | A           |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
| LABEL |    |            |    |    |    |    |                      | LABEL       |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18         | 19 | 20 | 21 | 22 | 23                   | 24          | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    | <b>TRA</b> |    |    |    |    |                      | <b>AREA</b> |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| TRA            | AREA     |          | Transfer the contents of memory location Area to Accumulator. Memory location unchanged. |

Example 2:

| LABEL |    | OP. CODE   |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER   |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|------------|----|----|----|----|----------------------|-------------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |            |    |    |    |    |                      | A           |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |
| LABEL |    |            |    |    |    |    |                      | LABEL       |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18         | 19 | 20 | 21 | 22 | 23                   | 24          | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|       |    | <b>TRM</b> |    |    |    |    |                      | <b>AREA</b> |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| TRM            | AREA     |          | Transfer the contents of Accumulator to memory location addressed by label area. |

|      |
|------|
| REM  |
| SLRO |

OP CODE

TRANSFER REMAINDER TO ACCUMULATOR

REM

The REM instruction transfers the remainder of a division operation to the Accumulator from the control area. The transfer will reset all Accumulator flags.

Example:

| LABEL      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|-----------|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|
| A          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | B                 |    |    | C  |    |    |          |    |    |    |    |    |                      |           |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LABEL      |    |    |    |    |    |    |    |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |          |    |    |    |    |    |                      |           |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16         | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31                | 32 | 33 | 34 | 35 | 36 | 37       | 38 | 39 | 40 | 41 | 42 | 43                   | 44        | 45 | 46 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| <b>REM</b> |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |          |    |    |    |    |    |                      |           |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |

**2.06.02 SHIFT ACCUMULATOR INSTRUCTIONS**

SHIFT OFF

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| SLRO           | 0-14     | 0-14     |

The SLRO instruction first causes the 15 digits of the Accumulator to be shifted left the number of positions specified by the A field. Any non-zero digits shifted off causes the overflow flag to be set. If the digits shifted off are zero, the flag is reset.

The 15 Accumulator digit positions are then shifted right the number of positions specified by the B field. Any non-zero digit shifted off does not set the overflow flag. Rounding is not performed. The shifted off digits are lost.

Example:

The Accumulator contains

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                            |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|----------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ACCUMULATOR DIGIT POSITION |
|    | 1  | 2  | 3  | 4  | 5  | 6 | 7 | 8 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | VALUE                      |

└ Flag Position

Examine the results when we execute the following instruction:

| LABEL       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|-----------|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|
| A           |    |    |    |    |    |    |    |    |    |    |    |    |    |    | B                 |    |    | C  |    |    |          |    |    |    |    |    |                      |           |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |
| LABEL       |    |    |    |    |    |    |    |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |          |    |    |    |    |    |                      |           |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |
| 16          | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31                | 32 | 33 | 34 | 35 | 36 | 37       | 38 | 39 | 40 | 41 | 42 | 43                   | 44        | 45 | 46 | 47 |  |  |  |  |  |  |  |  |  |  |  |  |
| <b>SLRO</b> |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |          |    |    |    |    |    |                      |           |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |
| <b>5</b>    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | <b>6</b>          |    |    |    |    |    |          |    |    |    |    |    |                      |           |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| SLRO           | 5        | 6        |

|       |
|-------|
| SLROS |
|-------|

After the 5 in the A parameter is executed the Accumulator contains

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                            |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|----------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ACCUMULATOR DIGIT POSITION |
|    | 6  | 7  | 8  | 9  | 8  | 7 | 6 | 5 | 4 | 3 | 0 | 0 | 0 | 0 |   | VALUE                      |

└ Flag Position

The overflow flag is set.

Then the contents are shifted right

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                            |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|----------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ACCUMULATOR DIGIT POSITION |
|    | 0  | 0  | 0  | 0  | 0  | 0 | 6 | 7 | 8 | 9 | 8 | 7 | 6 | 5 | 4 | VALUE                      |

└ Flag Position

SHIFT OFF WITH SIGN

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| SLROS          | 0-15     | 0-15     |

The SLROS instruction is the same as the SLRO instruction except that the sign position is also shifted.

This instruction may be used to shift alpha information.

|     |
|-----|
| CHG |
| LOD |

**2.07 – FLAG INSTRUCTIONS**

**2.07.01 CHANGE FLAGS INSTRUCTION**

| <u>OP CODE</u> | <u>A</u>       | <u>B</u>           |
|----------------|----------------|--------------------|
| CHG            | A K X<br>Y R P | 1 2 3 4<br>– S C M |

The CHG instruction reverses the condition (set or reset) of selected flags of any one flag group. A set flag is reset, a reset flag is set.

The flag group is designated in the A field and represented as:

| <u>DESIGNATION</u> | <u>FLAG GROUP</u>                              |
|--------------------|--|
| A                  | Accumulator Flags (–, S, C, M)                 |
| K                  | Operation Control Key Flags (1, 2, 3, 4)       |
| X                  | General Purpose Flags (1, 2, 3, 4)             |
| Y                  | General Purpose Flags (1, 2, 3, 4)             |
| R                  | Reader (Paper Tape or Card) Flags (1, 2, 3, 4) |
| P                  | Punch (Paper Tape or Card) Flags (1, 2, 3, 4)  |

The flags to be changed are represented as symbols or numbers in the B field. Any or all of the four flags of a flag group may be changed; all other flags in the group not changed are left unaltered.

Example:

| LABEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | FIELD<br>LEN-<br>GTH |    | PARAMETER |    |    |    |  |                   |  |   |  |  |  |   |  |  |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------------------|----|-----------|----|----|----|--|-------------------|--|---|--|--|--|---|--|--|
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                      |    | A         |    |    |    |  |                   |  | B |  |  |  | C |  |  |
| OP. CODE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | LABEL                |    |           |    |    |    |  | + OR -<br>INC/REL |  |   |  |  |  |   |  |  |
| 16       | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42                   | 43 | 44        | 45 | 46 | 47 |  |                   |  |   |  |  |  |   |  |  |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |                   |  |   |  |  |  |   |  |  |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |                   |  |   |  |  |  |   |  |  |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |                   |  |   |  |  |  |   |  |  |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                      |    |           |    |    |    |  |                   |  |   |  |  |  |   |  |  |

**2.07.02 LOAD FLAGS INSTRUCTION**

| <u>OP CODE</u> | <u>A</u>       | <u>B</u>           |
|----------------|----------------|--------------------|
| LOD            | A K X<br>Y R P | 1 2 3 4<br>– S C M |

The LOD instruction provides for setting selected flags of any one flag group. The A field designates the flag group to be set (refer to CHG instruction). The flags to be set are designated by numbers or symbols in the B field. Any or all of the four flags in a group may be set. All other flags in the group not set, are reset.

|     |
|-----|
| RST |
| SET |

Example:

| LABEL |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    | OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |  |  |  |     |   |  |  |  |   |  |  |
|-------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|----|----|----------|----|----------------------|-----------|----|----|--|--|--|-----|---|--|--|--|---|--|--|
|       |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    |          |    |                      | A         |    |    |  |  |  |     | B |  |  |  | C |  |  |
| LABEL |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    |          |    |                      |           |    |    |  |  |  |     |   |  |  |  |   |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23                | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38  | 39 | 40 | 41 | 42       | 43 | 44                   | 45        | 46 | 47 |  |  |  |     |   |  |  |  |   |  |  |
|       |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    | LØD |    |    |    |          |    | X                    |           |    |    |  |  |  | 2,3 |   |  |  |  |   |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|----------------|----------|----------|---|
| LOD            | X        | 2,3      | General purpose (group X) flags 2,3 are set, the other X flags are reset. |

**2.07.03 RESET FLAGS INSTRUCTION**

| <u>OP CODE</u> | <u>A</u>       | <u>B</u>           |
|----------------|----------------|--------------------|
| RST            | A K X<br>Y R P | 1 2 3 4<br>- S C M |

An RST instruction resets selected flags of any one flag group. The flag group is designated in the A field. (See CHG instructions for flag group designation.) The flags to be reset are specified by numbers or symbols in the B field. Any or all of the four flags may be reset. All other flags not reset are left unaltered.

Example:

| LABEL |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    | OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |  |  |  |   |   |  |  |  |   |  |  |
|-------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|----|----|----------|----|----------------------|-----------|----|----|--|--|--|---|---|--|--|--|---|--|--|
|       |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    |          |    |                      | A         |    |    |  |  |  |   | B |  |  |  | C |  |  |
| LABEL |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |    |          |    |                      |           |    |    |  |  |  |   |   |  |  |  |   |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22 | 23                | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38  | 39 | 40 | 41 | 42       | 43 | 44                   | 45        | 46 | 47 |  |  |  |   |   |  |  |  |   |  |  |
|       |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    | RST |    |    |    |          |    | A                    |           |    |    |  |  |  | - |   |  |  |  |   |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
|----------------|----------|----------|--|
| RST            | A        | -        | The "minus" flag of the Accumulator flag group is reset. <u>ALL</u> others are left unaltered. |

**2.07.04 SET FLAGS INSTRUCTIONS**

| <u>OP CODE</u> | <u>A</u>       | <u>B</u>           |
|----------------|----------------|--------------------|
| SET            | A K X<br>Y R P | 1 2 3 4<br>- S C M |

The SET instructions sets selected flags of any one flag group. The flag group is designated in the A field. (Ref. to CHG instruction for flag group and designation.) The flags to be set are designated by number or symbols in the B field. Any or all of the four flags of a group may be set. All other flags in the group not set, are left unaltered.

SET

Example:

| LABEL |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  | OP. CODE |  |  |    |  |  | FIELD<br>LEN-<br>GTH | PARAMETER |  |    |                   |  |  |  |   |    |  |  |   |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |
|-------|--|--|--|--|--|----|--|--|--|--|--|----|--|--|--|--|--|----|--|--|----------|--|--|----|--|--|----------------------|-----------|--|----|-------------------|--|--|--|---|----|--|--|---|--|--|----|--|--|--|--|--|----|--|--|--|--|--|----|--|--|--|--|--|----|--|--|--|--|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|----|--|
|       |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |          |  |  |    |  |  |                      | A         |  |    |                   |  |  |  | B |    |  |  | C |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |
| 16    |  |  |  |  |  | 17 |  |  |  |  |  | 18 |  |  |  |  |  | 19 |  |  |          |  |  | 20 |  |  |                      |           |  | 21 |                   |  |  |  |   | 22 |  |  |   |  |  | 23 |  |  |  |  |  | 24 |  |  |  |  |  | 25 |  |  |  |  |  | 26 |  |  |  |  |  | 27 |  | 28 |  | 29 |  | 30 |  | 31 |  | 32 |  | 33 |  | 34 |  | 35 |  | 36 |  | 37 |  | 38 |  | 39 |  | 40 |  | 41 |  | 42 |  | 43 |  | 44 |  | 45 |  | 46 |  | 47 |  |
| L     |  |  |  |  |  | A  |  |  |  |  |  | B  |  |  |  |  |  | C  |  |  |          |  |  | L  |  |  |                      |           |  |    | + OR -<br>INC/REL |  |  |  |   |    |  |  |   |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |
| SET   |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  | K        |  |  |    |  |  |                      | 3         |  |    |                   |  |  |  |   |    |  |  |   |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |  |  |  |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |    |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                                    |
|----------------|----------|----------|---|
| SET            | K        | 3        | The OCK flag 3 is set, other flags are unaltered. |

|      |
|------|
| LIR  |
| ADIR |
| DIR  |

**2.08 -- INDEX REGISTER INSTRUCTIONS**

**2.08.01 LOAD INDEX REGISTER INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| LIR            | 1-4      | 0-255    |

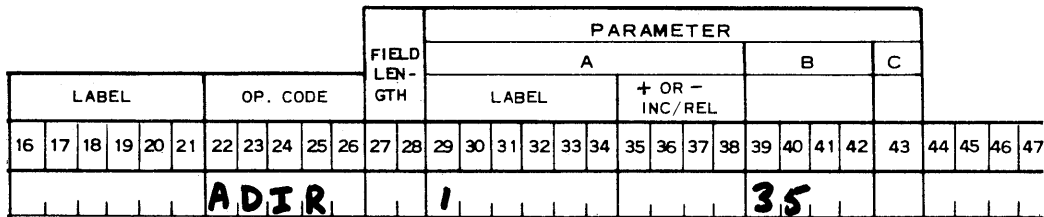
The LIR instruction loads the value contained in the B field into the index register indicated in the A parameter (1, 2, 3 or 4). The B parameter can be any positive value from 0 to 255. The prior contents of the index register are destroyed.

**2.08.02 ADD TO INDEX REGISTER INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| ADIR           | 1-4      | 0-255    |

The number contained in the B field is added to the contents of the index register (1, 2, 3 or 4) indicated by the A parameter. The B field contents and the index register contents are always positive. If the sum of the index register contents and the B field number equal 256, the register is reset to 0. If the sum is greater than 256, only the overflow is retained in the index register. In both cases, the overflow causes the Index Register Flag to be set. If the sum is less than 256, the flag is reset.

Example: Index Register 1 contains 225.



|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| ADIR           | 1        | 35       |

After execution of the above command, the contents of Index Register 1 is equal to 4 (225 + 35 - 256 = 4). The Index Register Flag is set.

**2.08.03 DECREMENT INDEX REGISTER INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| DIR            | 1-4      | 0-255    |

DECREMENT INDEX REGISTER

The DIR instruction decreases by 1, the contents of the index register designated by the A field. If the



|      |
|------|
| IIR  |
| TAIR |

index register contains 0, a decrement causes the value 255 to be entered into the register. The B field designates a value which is compared to the contents of the index register.

If the contents of the index register, designated by the A field, is equal to the value of the B field before decrementing is effected, the Index Register Flag is set after execution. If an unequal condition exists, the flag is reset after execution. Thus, if the flag is set during one decrementing, it will be reset during the next. For that reason, it becomes necessary to test this flag after each decrementing.

The value of the B field does not halt decrementing or turn the register back to 0, once decrementing has reached that limit.

**2.08.04 INCREMENT INDEX REGISTER INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| IIR            | 1-4      | 0-255    |

The IIR instruction increases by 1, the contents of the index register denoted by the A field. If the index register contains 255, incrementing causes the register to become 0. The B field designates a value which is compared to the contents of the index register.

The Index Register Flag is set and reset as in the DIR instruction.

Example: Use of Index Registers to terminate a loop (see SK instruction).

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|--------------|----------------|----------|----------|----------|
|              | LIR            | 2        | 0        |          |
| BEGIN        | MOD            | 2        |          |          |
|              | TRA            | TABLE    |          |          |
|              | IIR            | 2        | 9        |          |
|              | SK             | T        | I        | 1        |
|              | BRU            | BEGIN    |          |          |

**2.08.05 TRANSFER ACCUMULATOR CONTENTS TO INDEX REGISTER**

|  |          |
|--|----------|
| <u>OP CODE</u>                         | <u>A</u> |
| TRANSFER ACCUMULATOR TO INDEX REGISTER | TAIR 1-4 |

The TAIR instruction transfers the contents of the Accumulator to the register indicated by the A field. The prior contents of that index register are destroyed. The value of the Accumulator is treated as an absolute number, regardless of any "assumed" decimal places during entry in the Accumulator, and regardless of the setting of the Sign Flag.

MOD

Since an index register has a capacity of 255, an Accumulator value greater than 255 that is transferred to an index register will be accepted as that amount that exceeds the nearest multiple of 256 (maximum of 1024).

Example:

If the Accumulator contains 258, then 2 is transferred ( $258 - 256 = 2$ ).

If the Accumulator contains 525, then 13 is transferred ( $525 - (2 \times 256) = 13$ ).

**2.08.06 MODIFY BY INDEX REGISTER INSTRUCTION**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| MOD            | 1-4      |

The MOD instruction provides for adding the value in the index register designated by the A field to the parameter (or parameters) of the next instruction in program sequence following the MOD instruction. The instruction following MOD is then executed in accordance with the combined parameter values.

The MOD instruction does not change the instruction stored in memory. Modification occurs during the execution of the instruction, as the parameter is extracted from the instruction and placed in a special register. The MOD instruction affects the execution of only the one instruction immediately following.

Example: 1

| SEQUENCE |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |    |                |    |    |    |    |    |    | FIELD LEN-GTH |    | PARAMETER |    |    |    |    |    |    |    |    |  |  |   |  |  |
|----------|----|----|----|----|----|----------|----|----|----|----|----|-------|----|----|----|----|----|----|----------------|----|----|----|----|----|----|---------------|----|-----------|----|----|----|----|----|----|----|----|--|--|---|--|--|
|          |    |    |    |    |    |          |    |    |    |    |    |       |    |    |    |    |    |    |                |    |    |    |    |    |    |               |    | A         |    |    |    |    |    |    | B  |    |  |  | C |  |  |
| LABEL    |    |    |    |    |    | OP. CODE |    |    |    |    |    | LABEL |    |    |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |               |    |           |    |    |    |    |    |    |    |    |  |  |   |  |  |
| 11       | 12 | 13 | 14 | 15 | 16 | 17       | 18 | 19 | 20 | 21 | 22 | 23    | 24 | 25 | 26 | 27 | 28 | 29 | 30             | 31 | 32 | 33 | 34 | 35 | 36 | 37            | 38 | 39        | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |   |  |  |
| 0 1      |    |    |    |    |    | MOD      |    |    |    |    |    | 1     |    |    |    |    |    |    |                |    |    |    |    |    |    |               |    |           |    |    |    |    |    |    |    |    |  |  |   |  |  |
| 0 2      |    |    |    |    |    | POS      |    |    |    |    |    | 7     |    |    |    |    |    |    |                |    |    |    |    |    |    |               |    |           |    |    |    |    |    |    |    |    |  |  |   |  |  |

Assume Index Register Number 1 contains 50

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| MOD            | 1        |
| POS            | 7        |

The index register value of 50 combined with the value of the A parameter for the POS instruction causes the printer to position to 57 (7 + 50).

Although the MOD instruction is most generally used to modify those instructions which address word locations in memory, it may also be used to modify the parameters of most other instructions. The

|     |
|-----|
| MOD |
|-----|

contents of the index register are added to the parameter field to modulo 256. Modulo 256 means that if the index register (maximum capacity of 256) when added to the parameter field (also a maximum capacity of 256 in machine language), exceeds 256, a “carry” of 1 is generated and the excess value starts back to 0.

Example: 2

An index register with a value of 150, when added to an AL 200, generates a “carry” of 1 and a remaining parameter of 94 ( $350 - 256 = 94$ ). The carry is propagated to machine language operation code. Because of this, caution must be used in modifying most instructions since a “carry” may improperly modify the Op Code.

Different types of instructions will have the A parameter, or the B parameter, or both the A and B parameters modified. Some instructions cannot be modified.

The contents of the index register specified by the MOD instruction are added to the A parameter. If the combined value exceeds the range shown for each instruction parameter, either a “carry” will generate a new instruction, or the instruction will otherwise be improperly modified:

MOD

TABLE

Instructions in which only the A parameter is modifiable.

| <u>OP CODE</u> | <u>A</u> | <u>OP CODE</u> | <u>A</u> | <u>OP CODE</u> | <u>A</u> |
|----------------|----------|----------------|----------|----------------|----------|
| ADA            | LABEL*   | LRLR           | 0-255    | SUA            | LABEL*   |
| ADM            | LABEL*   | LSR            | 0-15     | SUM            | LABEL*   |
| AL             | 0-255    |                |          | TAIR           | 1-4      |
| ALR            | 0-255    | MUL            | LABEL*   | TK             | 0-150    |
| ALTO           | 0-255    | MULR           | LABEL*   | TKM            | 0-150    |
| AR             | 0-255    | OC             | 0-255    | TRA            | LABEL*   |
| ARTO           | 0-255    | PA             | LABEL*   | TRAB           | 0-15     |
| BRU            | LABEL*   | PAB            | 0-150    | TRB            | 1-15     |
| CLM            | LABEL*   | PBA            | 1-16     | TRBA           | 0-16     |
| CPA            | LABEL*   | POS            | 1-150    | TRCA           | 1-16     |
| DIV            | LABEL*   | RCP            | 1-255    | TRCM           | 1-16     |
| DUP            | 1-80     | REAM           | 0-150    | TRF            | 0-255    |
| EAM            | 0-150    | RTK            | 0-150    | TRM            | LABEL*   |
| IRCP           | 0-255    | RTKM           | 0-150    | TSB            | 1-15     |
| LCD            | 0-255    | RXEAM          | 0-150    | XA             | LABEL*   |
| LCFR           | LABEL*   | RXTK           | 0-150    | XB             | 0-255    |
| LKBR           | LABEL*   | RXTKM          | 0-150    | XBA            | 1-16     |
| LLCR           | 0-255    | SCP            | 1-255    | XEAM           | 0-150    |
| LLR            | 0-255    | SKP            | 1-80     | XMOD           |          |
| LPKR           | LABEL*   | SRJ            | LABEL*   | XPA            | LABEL*   |
| LPNR           | LABEL*   | SRR            | 1-4      | XPBA           | 1-16     |
| LRBR           | LABEL*   |                |          | XTK            | 0-150    |
| LRCR           | 0-255    |                |          | XTKM           | 0-150    |

\*The memory address referenced by the LABEL will be incremented by the value of the index register.

In the following instructions, only the B parameter field is modified; other parameter fields are unmodified. The contents of the index register is added to the B parameter of the instruction. If the combined value exceeds 255, either a "carry" will create a different instruction, or the instruction will otherwise be improperly modified.

TABLE

Instructions in which only the B parameter can be modified.

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| ADIR           | 1-4      | 0-255    |
| DIR            | 1-4      | 0-255    |
| IIR            | 1-4      | 0-255    |
| LIR            | 1-4      | 0-255    |

A. ONE PARAMETER CAN SPECIFY ONE OR MORE ITEMS. For some instructions the A and B parameters represent a binary pattern to the machine. The PKA, PKB instructions as well as the LOD, SET, RST and CHG flag instructions are programmed by listing the digits 1-8 (in the case of the PK instructions) and 1-4 (in the case of the flag instructions) in the A, B or A and B parameters for the desired pattern.

The EX, EXE, SK and SKE instructions are programmed by listing the digits 1-4 in the B parameter to designate the particular flag pattern desired.

To modify this binary pattern, it is necessary to find the decimal equivalent of the pattern desired and add it to the Index Register used in the MOD instruction. The value table below may be used to determine the number necessary to obtain the desired pattern.

TABLE

| Value Table                    |                    |                   |           |
|--------------------------------|--------------------|-------------------|-----------|
| No. in A, B or<br>A & B Fields | Decimal Equivalent |                   |           |
|                                | PKA<br>PKB         | Flag Instructions |           |
|                                | A & B field        | B field only      | A field   |
| 1                              | 1                  | 2                 | Punch = 0 |
| 2                              | 2                  | 4                 | Read = 16 |
| 3                              | 4                  | 8                 | X = 64    |
| 4                              | 8                  | 1                 | Y = 80    |
| 5                              | 16                 |                   | T = 128   |
| 6                              | 32                 |                   | K = 144   |
| 7                              | 64                 |                   | A = 192   |
| 8                              | 128                |                   |           |

For PK's, add together all of the equivalent values for the PK's specified in the A field, to determine the total value which must be loaded in the index register.

For Flag instructions (Set/Reset and Skip/Execute), add together the equivalent values for the flags specified in the B parameter. If the flag group is also to be modified, add its value to the total value for the individual flags, and the resulting sum is the value to be loaded in the index register.

To modify these instructions it is essential to originate them with 0 in the parameter fields and the desired pattern in the index register.

If these instructions are originated with some significant value in the parameter fields, an attempt to modify the parameters can propagate a carry which will be added to the Op Code, changing it to another Op Code.

MOD

TABLE

Instructions in which A and B parameters can be modified.

ONE PARAMETER CAN SPECIFY ONE OR MORE ITEMS.

| <u>OP CODE</u> | <u>A</u>         | <u>B</u> | <u>C</u> |
|----------------|------------------|----------|----------|
| PKA            | 12345678         |          |          |
| PKB            | 12345678         |          |          |
| LOD            | A K X<br>Y R P   | 1 2 3 4  |          |
| SET            | A K X<br>Y R P   | 1 2 3 4  |          |
| RST            | A K X<br>Y R P   | 1 2 3 4  |          |
| CHG            | A K X<br>Y R P   | 1 2 3 4  |          |
| EX             | A T K X<br>Y R P | 1 2 3 4  | 1-4      |
| EXE            | A T K X          |          | 1-4      |
| SK             | A T K X<br>Y R P | 1 2 3 4  | 1-4      |
| SKE            | A T K X<br>Y R P | 1 2 3 4  | 1-4      |

B. EACH PARAMETER CAN SPECIFY ONLY ONE ITEM. In these instructions, either or both, the A or B parameter can be modified. The C parameter, if one exists, is not modified. The A and B parameters combined cannot exceed 256. The sixteen possibilities in the B parameter requires a value from 0 to 15 in the index register for modification. The sixteen possibilities in the A parameter field require a value expressed in multiples of 16 (reflecting the digit position value of the A parameter in the instruction format).

The following table illustrates the proper values to be loaded in the index register to achieve the desired values for the A and B parameters.

|     |
|-----|
| MOD |
|-----|

TABLE FOR VALUES

| Number desired<br>in A field | “m”<br>Value to be<br>contained in<br>Index Reg. | Number desired<br>in B field | “n”<br>Value to be<br>contained in<br>Index Reg. |
|------------------------------|--|------------------------------|--|
| 0                            | 0  | 0                            | 0  |
| 1                            | 16   | 1                            | 1  |
| 2                            | 32   | 2                            | 2  |
| 3                            | 48   | 3                            | 3  |
| 4                            | 64   | 4                            | 4  |
| 5                            | 80   | 5                            | 5  |
| 6                            | 96   | 6                            | 6  |
| 7                            | 112  | 7                            | 7  |
| 8                            | 128  | 8                            | 8  |
| 9                            | 144  | 9                            | 9  |
| 10                           | 160  | 10                           | 10   |
| 11                           | 176  | 11                           | 11   |
| 12                           | 192  | 12                           | 12   |
| 13                           | 208  | 13                           | 13   |
| 14                           | 224  | 14                           | 14   |
| 15                           | 240  | 15                           | 15   |

“m” + “n” = total value to be contained in register.

Example: Modify NK 0 0 to provide 8 whole numbers and 3 decimal fractions:

Parameters required:

A = 8                    =

B = 3                    =

Index Register value required:

128

3

---

131 (total value)

Thus:    LIR    1    131

          MOD    1

          NK    0    0

The index register value of 131 modifies the NK instruction to permit 8 whole numbers and 3 fractions.

Any time that the modification of the B parameter results in a carry (exceeds 15), the carry will add to the A parameter changing its specification. A carry resulting from modification of the A parameter (exceeds 255) will add to the Op Code causing an improper modification.

MOD

EACH PARAMETER CAN SPECIFY ONLY ONE ITEM

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|----------|----------------|----------|----------|
| ADK            | 0-14     | 0-9      |          | PN             | 0-14     | 0-15     |
| CLA            | 0-15     | 0-9      |          | PNS+           | 0-14     | 0-15     |
| EXL            | 0-15     | 0-15     | 1-4      | PNS-           | 0-14     | 0-15     |
| INK            | 0-14     | 0-9      |          | TRCB           | 0-15     | 0-15     |
| NK             | 0-15     | 0-15     |          | XC             | 0-15     | 0-15     |
| NKCM           | 0-15     | 0-15     |          | XN             | 0-14     | 0-15     |
| NKR            | 0-15     | 0-15     |          | XPN            | 0-14     | 0-15     |
| NKRCM          | 0-15     | 0-15     |          | XPNS+          | 0-14     | 0-15     |
| SKL            | 0-15     | 0-15     | 1-4      | XPNS-          | 0-14     | 0-15     |
| SLRO           | 0-14     | 0-14     |          |                |          |          |
| SLROS          | 0-15     | 0-15     |          |                |          |          |
| SUK            | 0-14     | 0-9      |          |                |          |          |
| RNK            | 0-15     | 0-15     |          |                |          |          |

The following instructions cannot be modified:

TABLE

Instructions which are not modifiable.

| <u>OP CODE</u> | <u>A</u> | <u>OP CODE</u> | <u>A</u> | <u>OP CODE</u> | <u>A</u> |
|----------------|----------|----------------|----------|----------------|----------|
| ALARM          |          | LSN            |          | RPR            |          |
| ALTP           |          | LTN            |          | RR             |          |
| CC             |          | LXC            | 1        | RRA            |          |
| EXZ            | 1-4      | NOP            |          | RSA            |          |
| LPF            |          | RCD            |          | RSN            |          |
| LPR            |          | REL            |          | RTH            |          |
| LRA            |          | REM            |          | RTN            |          |
| LSA            |          | RPF            |          | SKZ            | 1-4      |
|                |          |                |          | STOP           |          |

The character in the A parameter of a PC instruction may be modified to obtain a different character. The MOD instruction will add the contents of the index register to the internal code of the character in the A parameter of the PC instruction.



MOD

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    | B                 | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29                | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| OP CODE | A |
|---------|---|
| MOD     | 1 |
| PC      | A |

If PC A (A = index value of 65) is to be modified to print M (M = index value of 77), a value of 12 (77-65 = 12) is loaded into the index register #1. Index values are contained in Appendix D. The above remarks also apply to PC+, PC- and PCP.

A MOD instruction may be used to modify another modify instruction with the same or different index register. The total amount of modification equals the sum of the MOD instructions, and should not exceed 255. When the total exceeds 255, only the difference between the total and 255 remains in the index register.

BRU

**2.09 – BRANCH AND DECISION INSTRUCTIONS**

**2.09.01 BRANCH UNCONDITIONAL INSTRUCTION**

|                |          |                |
|----------------|----------|----------------|
| <u>OP CODE</u> | <u>A</u> | <u>+/- REL</u> |
| BRU            | LABEL    | +<br>- N       |

The BRU instruction provides the ability to branch unconditionally to a different segment of the program. This instruction does not automatically provide for return to the branched from segment of the program.

The A parameter contains the label which identifies the memory address to where the program will branch. The A parameter can be incremented by an integer (N, positive or negative) located in the +/-REL field. A ± increment without a label will branch the program to either an instruction further ahead (+) or one behind(-) the current (BRU) instruction.

Example:

| LABEL  |    | OP. CODE |    |    |                   |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|--------|----|----------|----|----|-------------------|----|----------------------|-----------|----|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        |    |          |    |    |                   |    |                      | A         |    |        |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| LABEL  |    |          |    |    | + OR -<br>INC/REL |    |                      |           |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16     | 17 | 18       | 19 | 20 | 21                | 22 | 23                   | 24        | 25 | 26     | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
|        |    |          |    |    | SET               |    |                      |           |    | X      |    |    |    |    |    | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    | AL                |    |                      |           |    | 2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    | BRU               |    |                      |           |    | SHIPTO |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    | ?                 |    |                      |           |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| SHIPTO |    |          |    |    | POS               |    |                      |           |    | NMAD-P |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    | TK                |    |                      |           |    | 31     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    | AL                |    |                      |           |    | 1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

When the BRU instruction is executed program execution continues with the Op Code contained in the memory location referenced by the label. In this case the label is SHIPTO and the Op Code is POS.

|     |
|-----|
| SRJ |
| SRR |

**2.09.02 SUBROUTINE JUMP AND RETURN INSTRUCTIONS**

|                   | <u>OP CODE</u> | <u>A</u> | <u>+/- INC</u> |
|-------------------|----------------|----------|----------------|
| SUBROUTINE JUMP   | SRJ            | LABEL    | <u>+</u> N     |
| SUBROUTINE RETURN | SRR            | 1-4      |                |

The SRJ and SRR instruction facilitate branching to, and returning from a subroutine. The A parameter of the SRJ instruction contains the label of the memory location to where the jump will occur.

The SRJ and SRR instructions utilize the Subroutine Return Stack which appears thusly:

| LOCATION | ADDRESS         |
|----------|-----------------|
| 1        | MEMORY LOCATION |
| 2        | MEMORY LOCATION |
| 3        | MEMORY LOCATION |
| 4        | MEMORY LOCATION |

This example illustrates the use of these instructions and explains the A parameter of the SRR instruction.

| <u>WORD/SYLLABLE</u> | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>       |
|----------------------|--------------|----------------|----------|----------|----------------------|
| 25                   | 0            | NKR            | 10       | 4        | Allow Numeric Entry. |
|                      | 1            | AL             | 1        |          | Advance 1 line.      |
|                      | 2            | POS            | 63       |          | Position to print.   |
|                      | 3            | SRJ            | PRNC     |          | SRJ to print.        |
| 48                   | 0            | PRNC           | PNS-     | 14       | 0                    |
|                      | 1            |                | PC-      | -        |                      |
|                      | 2            |                | PC+      | +        |                      |
|                      | 3            |                | SRJ      | TKMAD    |                      |
| 50                   | 0            | TKMAD          | POS      | 95       | Positions for type.  |
|                      | 1            |                | TK       | 31       | Type 31 characters.  |
|                      | 2            |                | SRR      | 1        | Subroutine return.   |
|                      | 3            |                |          |          |                      |

When the SRJ instruction in word 25 syllable 3 is executed, the program counter is increased by 1 syllable. The new program counter content, word 26 syllable 0 is stored in Subroutine Return Stack location 1. The value of the A parameter in the SRJ instruction is inserted in the program, execution now begins at word 48, syllable 0. The Subroutine Return Stack would appear:

|     |
|-----|
| SRJ |
| SRR |

| LOCATION | ADDRESS   |
|----------|-----------|
| 1        | 26 0      |
| 2        | UNKNOWN-1 |
| 3        | UNKNOWN-2 |
| 4        | UNKNOWN-3 |

When the SRJ instruction in word 48, syllable 3 is reached, the contents of the Return Stack are shifted down 1 location. The memory address in location 4 is lost. Execution continues in word 50 syllable 0. The stack now contains:

| LOCATION | ADDRESS   |
|----------|-----------|
| 1        | 49 0      |
| 2        | 26 0      |
| 3        | UNKNOWN-1 |
| 4        | UNKNOWN-2 |

If the process is repeated 5 times, the original address entered (word 25 syllable 3) is lost from program control. Each additional repetition loses another memory address. It is recommended to limit the nesting of subroutines to 4.

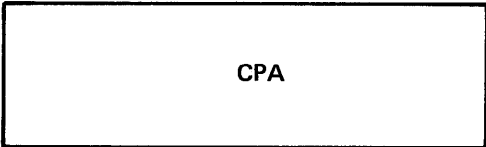
The execution of the SRR instruction in word 50 syllable 2 will cause the program counter to be loaded with a value from the Subroutine Return Stack. The value loaded is a function of the A parameter for the SRR instruction.

If the A value is 1, the memory address in location 1 is inserted in the program counter. A value of 2 would select location 2. A value of 3 would select location 3. A value of 4 would select the fourth location.

Since in our example we have a value of 1, word 49, syllable 0 is inserted into the program counter. Program execution begins with that value. The Return Stack would appear:

| LOCATION | ADDRESS   |
|----------|-----------|
| 1        | 26 0      |
| 2        | UNKNOWN-1 |
| 3        | UNKNOWN-2 |
| 4        | UNKNOWN-4 |

If the A value had been 2, word 26, syllable 0 would have been inserted in the program counter. All addresses with location numbers less than the selected location are lost. The remaining values are pushed to the top of the stack.



In this case the Subroutine Return Stack would appear:

| LOCATION | ADDRESS   |
|----------|-----------|
| 1        | UNKNOWN-1 |
| 2        | UNKNOWN-2 |
| 3        | UNKNOWN-4 |
| 4        | UNKNOWN-5 |

Program execution begins at word 26, syllable 0.

**2.09.03 COMPARE ALPHANUMERIC INSTRUCTION**

|                                      |                |          |
|--------------------------------------|----------------|----------|
|                                      | <u>OP CODE</u> | <u>A</u> |
| <b>SKIP AND EXECUTE INSTRUCTIONS</b> | CPA            | LABEL    |

The CPA instruction compares the contents of the memory word, referenced by the label contained in the "A" field, to the contents of the Accumulator. The outcome:

1. Execute the next instruction if contents are equal.
2. Execute second if memory word content is less than Accumulator content. Skip the first in sequence and begin execution.
3. If memory location content is greater than the Accumulator content, skip the first two in sequence and execute the third.

Refer to Appendix for collating sequence of character set.

Example:

| LABEL |    | OP. CODE |    |    |       |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|-------|----|----------------------|-----------|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |       |    |                      | A         |    |                   |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | LABEL |    |                      |           |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21    | 22 | 23                   | 24        | 25 | 26                | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| MAX   |    |          |    |    | NK    |    |                      |           |    | 4                 |    |    |    |    |    |    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | CPA   |    |                      |           |    | TEST              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | NOP   |    |                      |           |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | BRU   |    |                      |           |    |                   |    |    |    |    | +  |    | 2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | BRU   |    |                      |           |    |                   |    |    |    |    | +  |    | 3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | ALARM |    |                      |           |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    | BRU   |    |                      |           |    | MAX               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

This routine will allow the operator to index a value less than the value contained in the memory location TEST.

EXZ  
SKZ

**2.09.04 ACCUMULATOR SKIP AND EXECUTE INSTRUCTIONS**

|                             |                |          |
|-----------------------------|----------------|----------|
|                             | <b>OP CODE</b> | <b>A</b> |
| EXECUTE IF ACCUMULATOR ZERO | EXZ            | 1-4      |

If the content of the Accumulator is zero, the EXZ instruction will cause the number of instructions in the "A" field to be executed. If it is not zero, the next "A" instructions will be skipped.

|                          |                |          |
|--------------------------|----------------|----------|
|                          | <b>OP CODE</b> | <b>A</b> |
| SKIP IF ACCUMULATOR ZERO | SKZ            | 1-4      |

The SKZ instruction will cause the next 1-4 instructions (as specified in the "A" field) to be skipped when the Accumulator content is zero. Otherwise, the next instruction is executed.

Example: 1 Routine to enforce a non-zero keyboard listing

| LABEL  |    | OP. CODE |    | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|--------|----|----------|----|---------------|-----------|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        |    |          |    |               | A         |    |    |    |    |    | B              |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |               | LABEL     |    |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16     | 17 | 18       | 19 | 20            | 21        | 22 | 23 | 24 | 25 | 26 | 27             | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| NUMRIC |    | NK       |    |               | 5         |    |    |    |    |    |                |    |    | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | EXZ      |    |               | 1         |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | BRU      |    |               | NUMRIC    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | PNS-     |    |               | 5         |    |    |    |    |    |                |    |    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                             |
|--------------|----------------|----------|----------|--|
| NUMRIC       | NK             | 5        | 1        | Enable numeric keyboard.                   |
|              | EXZ            | 1        |          | Execute 1 instruction if Accumulator zero. |
|              | BRU            | NUMERIC  |          | Branch to numeric keyboard.                |
|              | PNS-           | 5        | 0        | Print shift ribbon (-).                    |

If an OCK is depressed without a numeric keyboard entry, the Accumulator contains zero. In the above example, whenever the Accumulator contains zero the BRU instruction is executed and the program branches to the NK command. This occurs until a numeric keyboard listing is made and the Accumulator is not zero; the BRU instruction is then skipped.

|     |
|-----|
| EXL |
| SKL |
| EX  |

Example 2: Do not print if the Accumulator is zero.

| LABEL   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER   |    |    |    |    |  |  |  |   |  |  |  |   |  |  |  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----------------------|---|----|----|----|----|--|--|--|---|--|--|--|---|--|--|--|
|   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |                      | A   |    |    |    |    |  |  |  | B |  |  |  | C |  |  |  |
| L A B C D E F G H I J K L M N O P Q R S T U V W X Y Z |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |                      | L A B C D E F G H I J K L M N O P Q R S T U V W X Y Z |    |    |    |    |  |  |  |   |  |  |  |   |  |  |  |
| 16  | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37       | 38 | 39 | 40 | 41 | 42                   | 43  | 44 | 45 | 46 | 47 |  |  |  |   |  |  |  |   |  |  |  |
|   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | TRA      |    |    |    |    |                      | AREA  |    |    |    |    |  |  |  |   |  |  |  |   |  |  |  |
|   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | SKZ      |    |    |    |    |                      | 1   |    |    |    |    |  |  |  |   |  |  |  |   |  |  |  |
|   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | SRJ      |    |    |    |    |                      | PRINT   |    |    |    |    |  |  |  |   |  |  |  |   |  |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>              |
|----------------|----------|----------|-----------------------------|
| TRA            | AREA     |          | Transfer to Accumulator.    |
| SKZ            | 1        |          | Skip 1 instruction if zero. |
| SRJ            | PRINT    |          | Branch to print routine.    |

|                                     | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|-------------------------------------|----------------|----------|----------|----------|
| EXECUTE IF DIGIT LESS THAN CONSTANT | EXL            | 0-15     | 0-15     | 1-4      |

The EXL instruction causes the next instruction to be executed if the digit in the Accumulator digit position specified in the "A" field is less than the constant contained in the "B" field, otherwise the next "C" are skipped. The Accumulator is undisturbed.

|                                  | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|----------------------------------|----------------|----------|----------|----------|
| SKIP IF DIGIT LESS THAN CONSTANT | SKL            | 0-15     | 0-15     | 1-4      |

The SKL instruction causes the next 1-4 instructions (as specified by the "C" field) to be skipped if the digit in the Accumulator digit position specified in the "A" parameter is less than the constant contained in the "B" field. Otherwise, the next instruction is executed. The Accumulator is undisturbed.

**2.09.05 FLAG EXECUTE AND SKIP INSTRUCTIONS**

|                     | <u>OP CODE</u> | <u>A</u>                           | <u>B</u>                               | <u>C</u> |
|---------------------|----------------|------------------------------------|--|----------|
| EXECUTE IF ANY FLAG | EX             | A T K<br>X Y R P<br>L B D S<br>V W | O L I U<br>1 2 3 4<br>- S C M<br>W R F | 1-4      |

EX

The EX instruction causes the next instruction in sequence to be executed if any of the flags specified in the "B" field (of the flag group designated in "A" field) are set. Otherwise, the next "C" instructions are skipped. (See SKE instruction for flags and flag groups.)

Example 1: Use of OCK to choose alternate branch of program

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
| LABEL |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31                | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|       |    |          |    |    |    | L  | K                    | B         | R  |    |    |    | T  | Y  | P                 | E  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | T  | K                    | M         |    |    |    |    | 2  | 5  |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | E  | X                    |           |    |    |    |    | K  |    |                   |    |    |    |    |    |    |    | 1  | 2  |    |    | 1  |    |    |    |
|       |    |          |    |    |    | B  | R                    | U         |    |    |    |    | S  | T  | A                 | R  | T  |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | P  | A                    |           |    |    |    |    | T  | Y  | P                 | E  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>        |
|----------------|----------|----------|----------|-----------------------|
| LKBR           | TYPE     |          |          | Load Base Register.   |
| TKM            | 25       |          |          | Type into memory.     |
| EX             | K        | 1 2      | 1        | Execute 1 if OCK 1, 2 |
| BRU            | START    |          |          | Branch                |
| PA             | TYPE     |          |          |                       |

In the above example the program will branch if OCK 1 or 2 was used. OCK 3 or 4 would cause a print.

Example 2: Load the Shift Register with 2 if the C key is used and with 3 if the M key is used

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
| LABEL |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31                | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|       |    |          |    |    |    | L  | S                    | R         |    |    |    |    | 0  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | E  | X                    |           |    |    |    |    | A  |    |                   |    |    |    |    |    |    |    | C  |    |    |    | 1  |    |    |    |
|       |    |          |    |    |    | L  | S                    | R         |    |    |    |    | 2  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       |    |          |    |    |    | E  | X                    |           |    |    |    |    | A  |    |                   |    |    |    |    |    |    |    | M  |    |    |    | 1  |    |    |    |
|       |    |          |    |    |    | L  | S                    | R         |    |    |    |    | 3  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |



|     |
|-----|
| EXE |
| SK  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>              |
|----------------|----------|----------|----------|-----------------------------|
| LSR            | 0        |          |          |                             |
| EX             | A        | C        | 1        | Test if "C" key used.       |
| LSR            | 2        |          |          | Load shift register with 2. |
| EX             | A        | M        | 1        | Test if "M" key used.       |
| LSR            | 3        |          |          | Load shift register with 3. |

| <u>EXECUTE IF EVERY FLAGS</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|-------------------------------|----------------|----------|----------|----------|
|                               | EXE            | A T K    | O L I U  | 1-4      |
|                               |                | X Y R P  | 1 2 3 4  |          |
|                               |                | L B D S  | - S C M  |          |
|                               |                | V W      | W R F    |          |

The EXE instruction causes the next instruction to be executed if all the flags specified in the "B" field (of flag group designated by the "A" field) are set. Otherwise, the next "C" instructions are skipped.

| LABEL |    | OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|-------|----|----------|----|----------------------|-----------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
|       |    |          |    |                      | A         |    |    |    |    |    | B                 |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|       |    |          |    |                      | LABEL     |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
| 16    | 17 | 18       | 19 | 20                   | 21        | 22 | 23 | 24 | 25 | 26 | 27                | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |  |
|       |    |          |    |                      |           | N  | K  | C  | M  |    |                   |    | 5  |    |    |    |    |    |    |    |    |    |    | 2  |    |    |    |    |    |    |  |
|       |    |          |    |                      |           | E  | X  | E  |    |    |                   |    | A  |    |    |    |    |    |    |    |    |    |    | C  | M  |    | 2  |    |    |    |  |
|       |    |          |    |                      |           | A  | L  | A  | R  | M  |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|       |    |          |    |                      |           | B  | R  | U  |    |    |                   |    |    |    |    |    |    |    |    | -  |    |    | 3  |    |    |    |    |    |    |    |  |

If the operator indexes both C and M keys, the alarm will sound.

| <u>SKIP IF ANY FLAGS</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|--------------------------|----------------|----------|----------|----------|
|                          | SK             | A T K    | O L I U  | 1-4      |
|                          |                | X Y R P  | 1 2 3 4  |          |
|                          |                | L B D S  | - S C M  |          |
|                          |                | V W      | W R F    |          |

The SK instruction causes the next "C" instructions (1-4) to be skipped if any of the flags specified in the "B" field, (flag group specified in "A" field) are set. Otherwise, the next instruction is executed.

Example: To terminate a loop

|        |    |    |    |    |    |          |    |    |    |    |    |                      |        |    |    |    |    |    |    |                   |  |   |    |    |    | PARAMETER |    |    |    |    |    |  |  |  |  |  |  |
|--------|----|----|----|----|----|----------|----|----|----|----|----|----------------------|--------|----|----|----|----|----|----|-------------------|--|---|----|----|----|-----------|----|----|----|----|----|--|--|--|--|--|--|
| LABEL  |    |    |    |    |    | OP. CODE |    |    |    |    |    | FIELD<br>LEN-<br>GTH | A      |    |    |    |    |    | B  |                   |  | C |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
| 16     | 17 | 18 | 19 | 20 | 21 | 22       | 23 | 24 | 25 | 26 | 27 |                      | 28     | 29 | 30 | 31 | 32 | 33 | 34 | + OR -<br>INC/REL |  |   | 39 | 40 | 41 | 42        | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |  |
| NUMRIC |    |    |    |    |    | LIR      |    |    |    |    |    |                      | 1      |    |    |    |    |    | 0  |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | NK       |    |    |    |    |    |                      | 2      |    |    |    |    |    | 3  |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | POS      |    |    |    |    |    |                      | 1 2 1  |    |    |    |    |    |    |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | AL       |    |    |    |    |    |                      | 3      |    |    |    |    |    |    |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | PN       |    |    |    |    |    |                      | 4      |    |    |    |    |    | 0  |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | IIR      |    |    |    |    |    |                      | 1      |    |    |    |    |    | 4  |                   |  |   |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | SK       |    |    |    |    |    |                      | T      |    |    |    |    |    | I  |                   |  | 1 |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |
|        |    |    |    |    |    | BRU      |    |    |    |    |    |                      | NUMRIC |    |    |    |    |    | +  |                   |  | 1 |    |    |    |           |    |    |    |    |    |  |  |  |  |  |  |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>               |
|--------------|----------------|----------|----------|----------|------------------------------|
| NUMRIC       | LIR            | 1        | 0        |          | Load Index Register.         |
|              | NK             | 2        | 3        |          | Enable numeric keyboard.     |
|              | POS            | 1        | 2        |          | Position printer.            |
|              | AL             | 3        |          |          | Advance 3 lines.             |
|              | PN             | 4        | 0        |          |                              |
|              | IIR            | 1        | 4        |          | Increment Index Register.    |
|              | SK             | T        | 1        | 1        | Skip 1 instruction if T set. |
|              | BRU            | NUMRIC   | +1       |          | Branch to NUMRIC plus 1.     |

|                     | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|---------------------|----------------|----------|----------|----------|
| SKIP IF EVERY FLAGS | SKE            | A T K    | O L I U  | 1-4      |
|                     |                | X Y R P  | 1 2 3 4  |          |
|                     |                | L B D S  | - S C M  |          |
|                     |                | V W      | W R F    |          |

The SKE instruction will cause the next "C" instructions to be skipped if all the flags specified in the "B" field (of the flag group specified) are set. Otherwise, the next instruction is executed.

The flags and flag groups are designed thusly:

- |   |   |  |
|---|---|--|
| <p>1. ACCUMULATOR FLAGS</p> <p>A — [ — Sign<br/>S Special<br/>C Per Hundred<br/>M Per Thousand</p>                            | <p>5. L FLAGS (SHIFT REG)</p> <p>L — [ 1<br/>2<br/>3<br/>4</p>  | <p>9. TEST FLAGS</p> <p>T — [ ∅ Overflow<br/>L Forms Limit<br/>I Index Register<br/>U Unassigned</p>               |
| <p>2. KEYBOARD BUFFER FLAGS</p> <p>B — [ 2 KB Buffer Filled<br/>3 KB Buffer Empty</p>   | <p>6. PUNCH FLAGS</p> <p>P — [ 1 Media Not Present<br/>2 Echo Check<br/>3 Tape Supply<br/>4 Punch Off</p>             | <p>10. TELLER LOCK FLAGS</p> <p>V — [ 1 Teller 1<br/>2 Teller 2<br/>3 Supervisor<br/>4 Not Used</p>                |
| <p>3. DATA COMM FLAGS</p> <p>D — [ 1 Received TR# Not<br/>Equal Expected TR #<br/>2 Message Received<br/>3 Transmit Ready</p> | <p>7. READER FLAGS</p> <p>R — [ 1 Reader Condition<br/>2 Message Received<br/>3 Transmit Ready<br/>4 Invalid Code</p> | <p>11. PASSBOOK FLAGS</p> <p>W — [ 1 Passbook Fold<br/>2 Last Print Line<br/>3 Not Used<br/>4 First Print Line</p> |
| <p>4. OCK FLAGS</p> <p>K — [ 1 OCK - 1<br/>2 OCK - 2<br/>3 OCK - 3<br/>4 OCK - 4</p>  | <p>8. STRIPE LEDGER FLAG</p> <p>S — [ ∅ Not Used<br/>W Write Error<br/>R Read Error<br/>F Filled Sheet</p>            | <p>12. GENERAL PURPOSE FLAGS</p> <p>X — [ 1<br/>2<br/>3<br/>4</p>  |

### 2.09.06 SKIP AND EXECUTE INSTRUCTIONS FOR THE TC 700

The lock flags and passbook signal flags may be interrogated using the SKIP and EXECUTE instructions (see Subject 2.09.05). They cannot be referenced with the SET, RESET, LOAD or CHANGE macro instructions.

#### Lock Flags (V flag group)

Three flags are provided which test the status of the Teller 1 lock, Teller 2 lock and Supervisor lock. These are:

- Flag V1 for the Teller 1 flag
- Flag V2 for the Teller 2 flag
- Flag V3 for the Supervisor Override Flag
- Flag V4 is not used

When the Teller 1 key is inserted in its lock and turned, the Teller 1 flag will be set. When the key is removed from its lock, the Teller 1 flag will be reset. The same applies to the Teller 2 key and the Supervisor key.

| <u>INSTRUCTION</u>    | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|-----------------------|----------------|----------|----------|----------|
| Skip if any flags     | SK             | V        | 123      | 1-4      |
| Skip if every flag    | SKE            | V        | 123      | 1-4      |
| Execute if any flags  | EX             | V        | 123      | 1-4      |
| Execute if every flag | EXE            | V        | 123      | 1-4      |

Passbook Signal Flags (W flag group)

Three flags test the sensors in the passbook alignment area. These are:

Flag W4 for 1st Print Line

Flag W1 for Passbook Fold

Flag W2 for Last Line

Flag W3 not used

When the Passbook is inserted to the fixed rear limit, the 1st Print Line Flag will be set. It will be reset at all other times. When the Passbook is so situated in the alignment area that the current print line will fall within the passbook fold area, the Passbook Fold Flag will be set. It will be reset when this condition does not exist.

When the Passbook is so aligned that the current print line is below the last printing line of the Passbook, the last Print Line Flag will be set. It will be reset when the passbook is aligned to any of the actual printing lines of the book.

A separate Passbook Present Flag does not exist. This condition can be determined by testing for the NOT SET condition of the Last Line Flag. This result occurs because if a passbook is present in the alignment mechanism and is aligned to any of the possible posting lines of the passbook, the Last Line Flag will be reset. The flag will be set if the passbook is aligned to the line below the last print line or if there is no passbook in the mechanism at all.

| <u>INSTRUCTION</u>    | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|-----------------------|----------------|----------|----------|----------|
| Skip if any flags     | SK             | W        | 124      | 1-4      |
| Skip if every flag    | SKE            | W        | 124      | 1-4      |
| Execute if any flag   | EX             | W        | 124      | 1-4      |
| Execute if every flag | EXE            | W        | 124      | 1-4      |

Machine language code for V and W flag groups.

Reference the appropriate SKIP or EXECUTE instruction in Appendix B.

Use the weights:

Parameter upper position:

|         |       |
|---------|-------|
| V flags | use E |
| W flags | use F |

Parameter lower position:

| <u>FLAG</u> | <u>WEIGHT</u> |
|-------------|---------------|
| W1 or V1    | 2             |
| W2 or V2    | 4             |
| W3 or V3    | 8             |
| W4 or V4    | 1             |

|       |
|-------|
| ALARM |
| NOP   |

**2.10 – MISCELLANEOUS INSTRUCTIONS**

**2.10.01 ALARM INSTRUCTION**

OP CODE

ALARM

The ALARM instruction will sound the Error Alarm once. The system does not go into the error state.

Example: Notify operator an error has been made. See the EXE instruction.

**2.10.02 NO OPERATION INSTRUCTION**

OP CODE

NOP

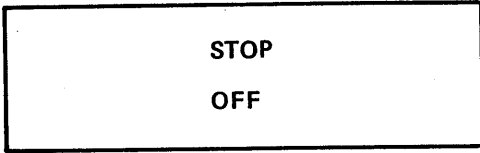
The NOP instruction performs no operation, but 10 milliseconds are expended when this instruction is used. Program execution continues, sequentially, uninterrupted. The NOP instruction is particularly useful in building the PK table and in conjunction with the CPA instruction.

Example: Use only PKA 4, 6 and 8.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|--------------|----------------|----------|
| PKEYS        | NOP            |          |
|              | NOP            |          |
|              | NOP            |          |
|              | BRU            | TOTAL    |
|              | NOP            |          |
|              | BRU            | SUBTTL   |
|              | NOP            |          |
|              | BRU            | START    |

Example: If the contents of memory word TOTAL are equal to or less than the contents of the Accumulator, branch to START. If the contents are greater, go to error.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|--------------|----------------|----------|
|              | CPA            | TOTAL    |
|              | NOP            |          |
|              | BRU            | START    |
|              | BRU            | ERROR    |



### 2.10.03 STOP PROGRAM INSTRUCTION

#### OP CODE

STOP

The STOP instruction halts the execution of a program and returns the computer to the Ready Mode.

### 2.10.04 POWER OFF

#### OP CODE

OFF

The OFF instruction provides the ability for the TC to turn itself off by causing the power to the entire system to be turned off. This instruction permits the data center to notify a TC to shut down, by sending a reserved character or other unique data (selected by user) to it. Upon testing and recognizing this character, the TC would branch to the instruction OFF as a part of the user program.

**2.11 – CHECK DIGIT INSTRUCTIONS**

Macro instructions to compute and verify check digits are available for use on the L/TC by incorporating a CDC-CDV Add-On Firmware Set with the Basic Main Memory Firmware Set being utilized. CDC-CDV Add-On Firmware Sets occupy the highest track of user memory provided by the main memory firmware set.

**2.11.01 CHECK DIGIT COMPUTE INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| CDC            | 1-15     | 0-9      |

The CDC instruction, when used in conjunction with a check digit table, will generate a check digit for a number located in the Accumulator. The check digit will be generated for the number which begins in the Accumulator digit position indicated by the A parameter and ending in Accumulator digit position 1. The generated check digit will be inserted in Accumulator digit position 0, remaining Accumulator digit positions are not disturbed.

The B parameter specifies the constant remainder that is to be used when computing the check digit.

Example 1:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |
|       |    |          |    |    |    | C  | D                    | C         |    |    |    |    | 6  |    |    |    |    |    |    |    |    |    |    | 1  |    |    |    |    |    |    |    |  |

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| CDC            | 6        | 1        |

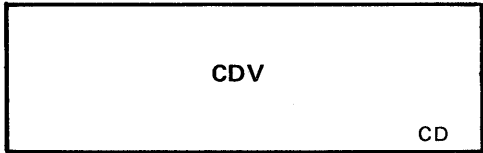
If the Accumulator contains:

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                        |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Accumulator Digit Pos. |
|    | 6  | 9  | 8  | 4  | 2  | 9 | 6 | 3 | 8 | 4 | 2 | 9 | 6 | 3 | 0 | Value                  |

└─ Flag Position

the check digit will be calculated for the number beginning in Accumulator digit position 6 and ending in Accumulator digit position 1; in this case 842963.

The remainder factor used will be 1.



Example 2:

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>        | <u>B</u> | <u>C</u> | <u>REMARKS</u>                  |
|--------------|----------------|-----------------|----------|----------|---------------------------------|
| INITIL<br>}  | LPNR<br>}      | TABLE<br>}      |          |          | LOAD CD & P MASK TABLE<br>}     |
|              | TRA            | BAL             |          |          | RD NEW BALANCE                  |
|              | SLRO           | 1               | 0        |          | POSITION FOR CD                 |
|              | EX             | A               | -        | 1        | TEST IF MINUS BALANCE           |
|              | CDC            | 8               | 3        |          | COMPUTE CD ON MINUS USING REM 3 |
|              | SK             | A               | -        | 1        | SKIP IF MINUS BALANCE           |
|              | CDC            | 8               | 2        |          | COMPUTE CD ON PLUS USING REM 2  |
|              | PNS-           | 8               | 2        |          | PRINT NEW BALANCE               |
|              | PNS-           | 0               | 3        |          | PRINT CHECK DIGIT               |
|              | NOTE           |                 |          |          | ALTERNATE COL DOUBLE ADD DOUBLE |
|              | NOTE           |                 |          |          | MOD 10 CD TABLE & P MASKS       |
| TABLE        | NUM            | 166009753186420 |          |          | 1ST WORD CD TABLE               |
|              | NUM            | 066009876543210 |          |          | LAST WORD CD TABLE              |
|              | MASK           | ZZZ,ZZZ,DDE     |          |          | P MASK BALANCE                  |
|              | MASK           | +D              |          |          | P MASK CHECK DIGIT              |

**2.11.02 CHECK DIGIT VERIFY INSTRUCTION**

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| CDV            | 1-15     | 0-9      |

The CDV instruction will verify the check digit of a number located in the Accumulator. The number begins in the Accumulator digit position specified by the A parameter and ends in Accumulator digit position 1. Any significant digits located to the left of the Accumulator digit position specified by the A parameter are ignored by the CDV instruction.

The check digit must be located in Accumulator digit position 0.

The B parameter specifies the constant remainder that is used in computing the check digit. If the check digit is not equal to the computed check digit, the Accumulator S Flag is set and a Keyboard Error Condition occurs at the next keyboard instruction. The programmer should provide the required instructions to check the S Flag after verification.

The checking method is determined by the table designated in the A parameter of the last executed LPNR instruction.



CDV

CD

Example 1:

| LABEL      |    | OP. CODE |    |    |                |    | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----------|----|----|----------------|----|---------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|            |    |          |    |    |                |    |               | A         |    |    |    |    | B  | C  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| L.A.B.E.L. |    |          |    |    | + OR - INC/REL |    |               |           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16         | 17 | 18       | 19 | 20 | 21             | 22 | 23            | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| CDV        |    |          |    |    | 8              |    |               |           |    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

|                |          |          |
|----------------|----------|----------|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| CDV            | 8        | 0        |

If the Accumulator contains:

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                        |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Accumulator Digit Pos. |
|    | 0  | 0  | 0  | 0  | 0  | 0 | 2 | 3 | 5 | 6 | 8 | 9 | 2 | 4 | 5 | Value                  |

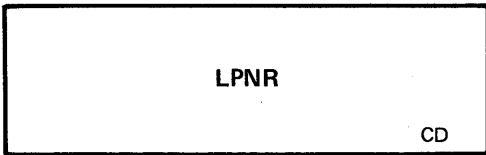
└─ Flag Position

the number to be verified begins in Accumulator digit position 8 and ends in Accumulator digit position 1, in this case 23568924.

The remainder factor is 0. The check digit is 5.

Example 2: The CDV Instruction in conjunction with a Modulus 11 weighted system could be utilized in the user program in the following manner.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>        | <u>B</u> | <u>C</u> | <u>REMARKS</u>                 |
|--------------|----------------|-----------------|----------|----------|--------------------------------|
| INITIL       | LPNR           | TABLE           |          |          | LOAD CD & P MASK TABLE         |
| ACCTNO       | NKCM           | 7               | 0        |          | INDEX ACCT NO. & CHECK DIGIT   |
|              | NOTE           |                 |          |          | USE "C" FOR C.D. VALUE OF "A." |
|              | EX             | A               | C        | 2        | TEST FOR "A"                   |
|              | SLRO           | 1               | 0        |          | POSITION NUMBER                |
|              | INK            | 0               | A        |          | INSERT CHK DIGIT "A"           |
|              | CDV            | 6               | 0        |          | VERIFY                         |
|              | EX             | A               | S        | 1        | EX IF NOT VERIFIED             |
|              | BRU            | ACCTNO          |          |          | BR TO REINDEX                  |
|              | PN             | 6               | 1        |          | PRINT ACCT NO.                 |
|              | NOTE           |                 |          |          | 1, 3, 7 MODULUS 11 CHK DIGIT   |
|              | NOTE           |                 |          |          | TABLE AND PRINT MASKS          |
| TABLE        | NUM            | 355003692581470 |          |          | WT. 7 VALUES 1ST WORD CD TABLE |
|              | MASK           | +DDDDDD,D       |          |          | ACCT. NO. PRINT MASK           |
|              | MASK           | ZZZ,ZZZ.DD      |          |          | AMOUNT PRINT MASK              |
|              | NUM            | 455007418529630 |          |          | WT. 3 VALUES 2ND WORD CD TABLE |
|              | NUM            | 055009876543210 |          |          | WT. 1 VALUES 3RD WORD CD TABLE |



**2.11.03 LOAD CHECK DIGIT AND PRINT NUMERIC TABLE INSTRUCTION**



The LPNR instruction is used to locate the check digit and print mask tables when check digit firmware is used. The first entry of the table must be a check digit entry. The table can vary in size from 1 to 256 words. The reader should reference CHECK DIGIT TABLE CONSTRUCTION.

**2.11.04 CHECK DIGIT TABLE CONSTRUCTION**

The table(s) that are utilized by the CDC-CDV instruction determine the checking method to be used. The table(s) can be located anywhere within user memory and are referenced by the A Parameter of the LPNR instruction. The table can vary in size from 1 word to 256 words and the individual entries within the table do not have to be stored in consecutive order. However, the first entry in the table must be labeled so that it can be referenced by the LPNR instruction.

Each entry (word) in the table is divided into three sections. These divisions are as follows:

1. Location of the next table entry to be referenced (digit positions 15 & 14).
2. Modulus used (digit positions 13 & 12).
3. Digit values (digit positions 0-9).

The CDC & CDV instructions start with the table entry specified by LPNR. The location of the next table entry to be referenced by the CDC or CDV instruction is determined by the Hexadecimal value of digit positions 15 & 14 of the table entry. This location is relative to the base word of the table (the beginning word of the table which is referenced by the A parameter of the LPNR instruction).

Example:

| HEXADECIMAL VALUE<br>IN 15 | & | 14 | RELATIVE LOCATION OF<br>NEXT TABLE ENTRY |
|----------------------------|---|----|--|
| 0                          |   | 1  | Base Word + 1                            |
| 0                          |   | 2  | Base Word + 2                            |
| 1                          |   | 1  | Base Word + 17                           |
| 0                          |   | 0  | Base Word + 0                            |

Digit positions 13 & 12 specify the modulus to be used in the verification scheme. The values in both digit positions within the word must be identical and the value in positions 13 & 12 in each table entry must be identical. The table assumes a base modulus of 16.

Therefore, to determine the entry for positions 13 & 12 the decimal values of the modulus desired must be subtracted from the base modulus of 16. For example, if a modulus 10 scheme is to be used a 6 would be entered in digit positions 13 & 12 of every table entry (16-10 = 6).

Each digit position of an integer (to be checked/computed) has 10 possible values (0 to 9). Each table entry word represents certain digit positions in the integer.

Example: A table with 3 entries (words) is used to check/compute a check digit for a 6-digit integer.

- The 1st table entry is used for digit positions 1 and 4
- The 2nd table entry is used for digit positions 2 and 5
- The 3rd table entry is used for digit positions 3 and 6

The Digit Values section of each table entry contains the weighted or assigned values for the digit positions that the table entry represents. The weighted or assigned values are located within the digit values section (Digit Positions 0-9) in order according to the possible value that it represents. For example, the weighted or assigned values for the possible digit position value of 7 on the integer is stored in digit position 7 of the table entry.

A simple alternate column Double-Add-Double Check Digit scheme would require a two-word table with the following values in digit positions 0-9 (Digit Values Section) of the table entries.

| Integer Digit Value and Table Entry Digit Position | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--|---|---|---|---|---|---|---|---|---|---|
| 1st Table Entry Values                             | 9 | 7 | 5 | 3 | 1 | 8 | 6 | 4 | 2 | 0 |
| 2nd Table Entry Values                             | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Example 1:

Alternate Column, Double-Add-Double

Modulus 10

Remainder 0

|                                    |   |    |    |    |    |    |   |       |      |
|------------------------------------|---|----|----|----|----|----|---|-------|------|
| Integer (Acct No.)                 | 4 | 3  | 2  | 2  | 5  | 7  |   |       |      |
| Assigned Values From Table         | 4 | +6 | +2 | +4 | +5 | +5 | = | 26    |      |
| Remainder                          |   |    |    |    |    |    |   | 0     |      |
| Total Sum of Assigned Values       |   |    |    |    |    |    |   | 26+0  | = 26 |
| Next High Multiple Of Modulus (10) |   |    |    |    |    |    |   | 30    |      |
| Check Digit                        |   |    |    |    |    |    |   | 30-26 | = 4  |

The values assigned in computing the check digit for the above integer (Acct No.) are as follows: The assigned values for the digits located in positions 1, 3 & 5 of the integer are taken from the 1st table entry. The assigned values for the digits located in positions 2, 4 & 6 of the integer are taken from the 2nd table entry.

COMPLETE TABLE

|               | POSITIONS     |    |     |    |    |              |   |   |   |   |   |   |   |   |   |   |
|---------------|---------------|----|-----|----|----|--------------|---|---|---|---|---|---|---|---|---|---|
|               | 15            | 14 | 13  | 12 | 11 | 10           | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|               | Next Word LOC |    | Mod |    |    | Digit Values |   |   |   |   |   |   |   |   |   |   |
| TABLE ENTRY 1 | 0             | 1  | 6   | 6  |    | 9            | 7 | 5 | 3 | 1 | 8 | 6 | 4 | 2 | 0 |   |
| TABLE ENTRY 2 | 0             | 0  | 6   | 6  |    | 9            | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |   |

Example 2: 1, 3, 7 MODULUS 11 METHOD

In this method the assigned value for each digit is obtained by assigning weights of 7, 3, 1, 7, 3, 1, ... continuously; starting with the least significant digit of the number. A three-word table is required.

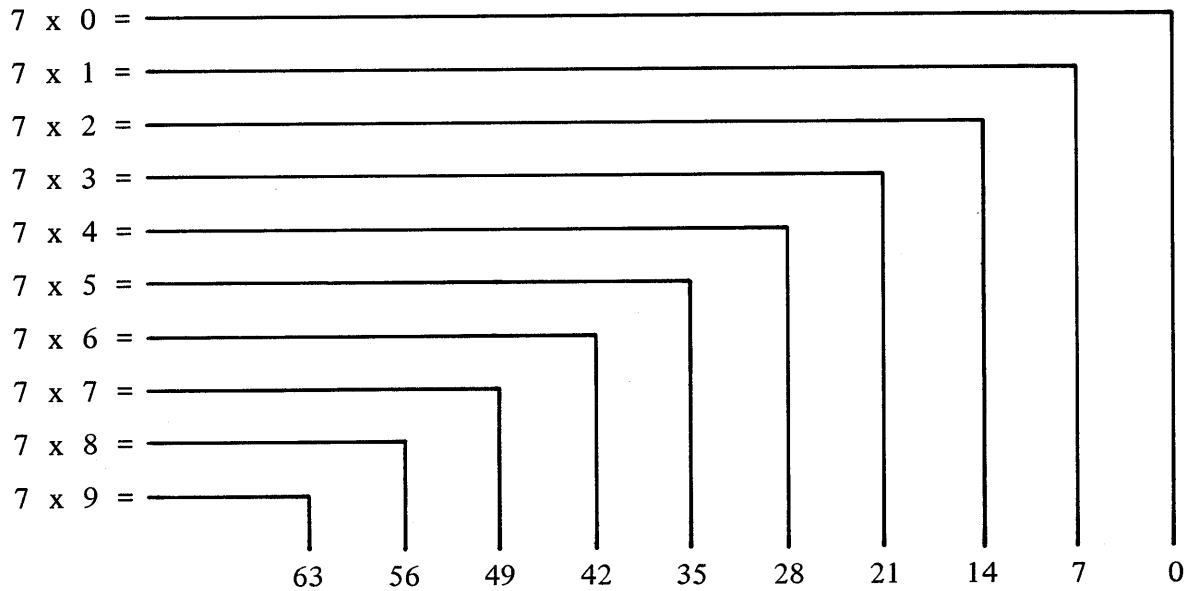
|                                 |   |   |   |   |   |    |   |    |   |    |   |   |    |
|---------------------------------|---|---|---|---|---|----|---|----|---|----|---|---|----|
| Integer                         | 4 | 3 | 2 | 2 | 7 | 7  |   |    |   |    |   |   |    |
| Assigned Value From Table       | 4 | + | 9 | + | 3 | +  | 2 | +  | A | +  | 5 | = | 33 |
| Remainder                       |   |   |   |   |   | 1  |   |    |   |    |   |   |    |
| Total Sum of Assigned Values    |   |   |   |   |   | 33 | + | 1  | = | 34 |   |   |    |
| Next Higher Multiple of Modulus |   |   |   |   |   | 44 |   |    |   |    |   |   |    |
| Check Digit                     |   |   |   |   |   | 44 | - | 34 | = | A  |   |   |    |

TABLE

|  | POSITIONS     |    |     |    |    |              |   |   |   |   |   |   |   |   |   |   |
|--|---------------|----|-----|----|----|--------------|---|---|---|---|---|---|---|---|---|---|
|  | 15            | 14 | 13  | 12 | 11 | 10           | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | Next Word LOC |    | Mod |    |    | Digit Values |   |   |   |   |   |   |   |   |   |   |
|  | 0             | 1  | 5   | 5  |    | 8            | 1 | 5 | 9 | 2 | 6 | A | 3 | 7 | 0 |   |
|  | 0             | 2  | 5   | 5  |    | 5            | 2 | A | 7 | 4 | 1 | 9 | 6 | 3 | 0 |   |
|  | 0             | 0  | 5   | 5  |    | 9            | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |   |

The table for the example of the 1, 3, 7 Modulus 11 Method was derived in the following manner.

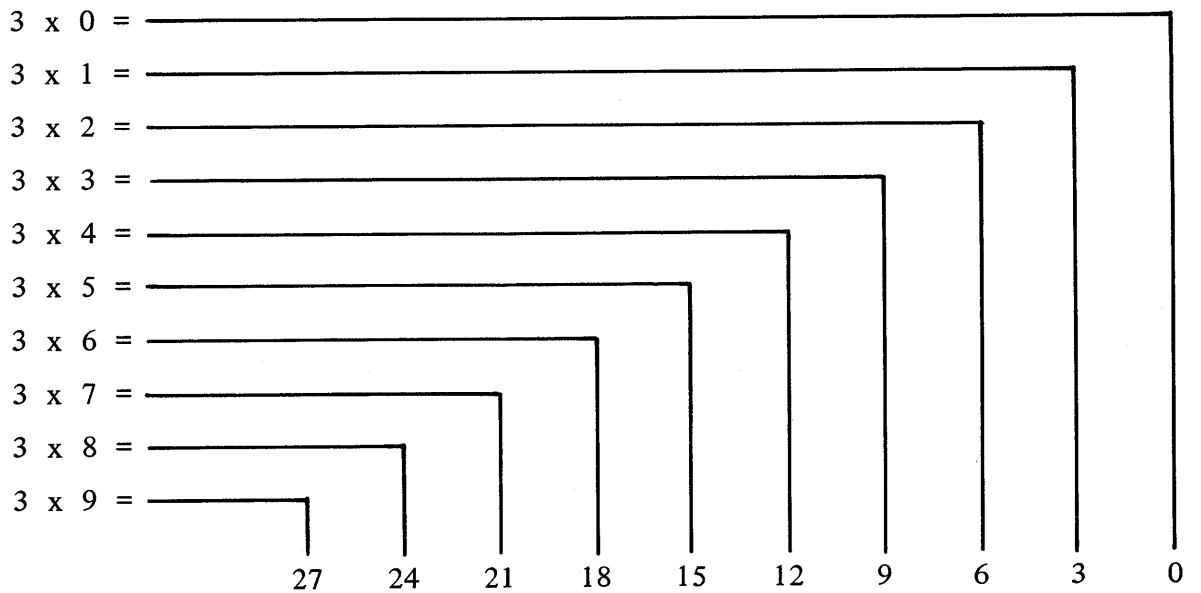
1st Table Entry (Weighted 7).



Minus Next Lowest

|                     |            |            |            |            |            |            |            |            |           |           |
|---------------------|------------|------------|------------|------------|------------|------------|------------|------------|-----------|-----------|
| Multiple of Modulus | <u>-55</u> | <u>-55</u> | <u>-44</u> | <u>-33</u> | <u>-33</u> | <u>-22</u> | <u>-11</u> | <u>-11</u> | <u>-0</u> | <u>-0</u> |
| 1st Table Entry =   | 8          | 1          | 5          | 9          | 2          | 6          | A          | 3          | 7         | 0         |

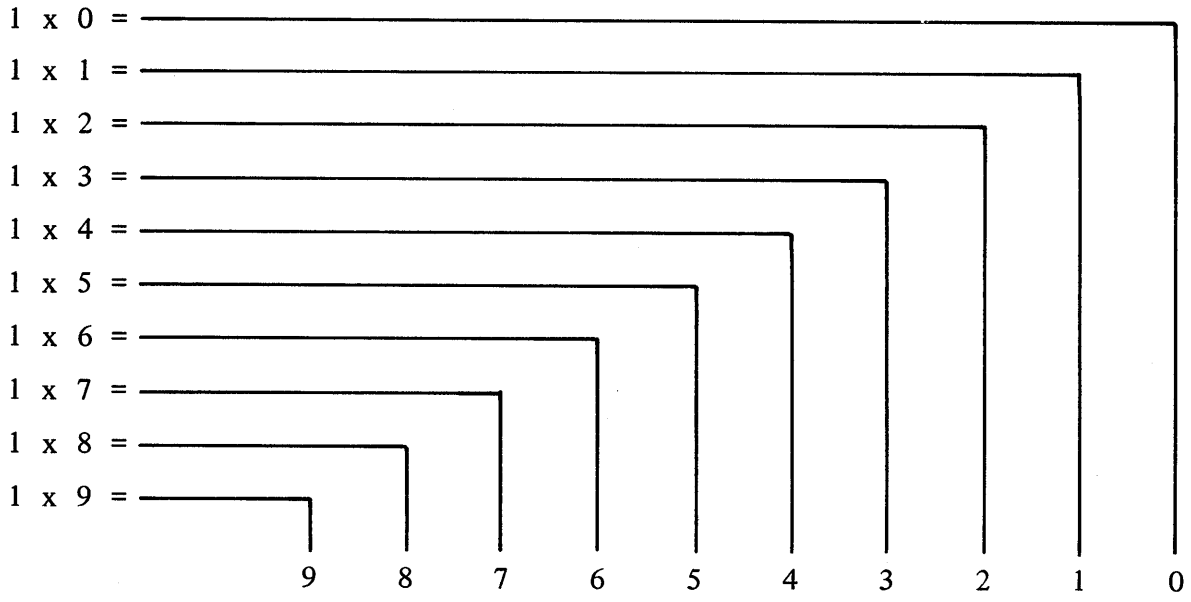
2nd Table Entry (Weighted 3).



Minus Next Lowest

|                     |            |            |            |            |            |            |           |           |           |           |
|---------------------|------------|------------|------------|------------|------------|------------|-----------|-----------|-----------|-----------|
| Multiple of Modulus | <u>-22</u> | <u>-22</u> | <u>-11</u> | <u>-11</u> | <u>-11</u> | <u>-11</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> |
| 2nd Table Entry =   | 5          | 2          | A          | 7          | 4          | 1          | 9         | 6         | 3         | 0         |

3rd Table Entry (Weighted 1).



|                     |           |           |           |           |           |           |           |           |           |           |
|---------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Minus Next Lowest   |           |           |           |           |           |           |           |           |           |           |
| Multiple of Modulus | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> | <u>-0</u> |
| 3rd Table Entry =   | 9         | 8         | 7         | 6         | 5         | 4         | 3         | 2         | 1         | 0         |

A check digit can be accurately computed and verified on fixed length alphanumeric fields that do not exceed 7 characters in length. The check digit would make the 8th character.

Example 3:

The following example illustrates how a check digit could be computed on a 5-character fixed length alpha field (check digit is entered as the 6th character) using a 1, 3, 7 Modulus 10 Method.

| <u>SEQ</u> | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>                   |
|------------|--------------|----------------|----------|----------|----------|----------------------------------|
| 1          | INITAL       | LPNR           | TABLE    |          |          | LOAD CHECK DIGIT TABLE           |
| 2          | CMPCD        | POS            | 10       |          |          | POSITION PRINTER                 |
| 3          |              | LKBR           | PARTNO   |          |          | SET KB BASE REGISTER             |
| 4          |              | TKM            | 5        |          |          | ENTER PART NUMBER                |
| 5          |              | TRA            | PARTNO   |          |          | READ ALPHA TO ACCUMULATOR        |
| 6          |              | SLROS          | 0        | 4        |          | RIGHT JUSTIFY ALPHA NUMBER       |
| 7          |              | INK            | 1        | 3        |          | INSERT 3 COL 1                   |
| 8          |              | NOTE           |          |          |          | THE 3 IS INSERTED SO THAT THE CD |
| 9          |              | NOTE           |          |          |          | NUMBER CAN BE ENTERED THROUGH    |
| 10         |              | NOTE           |          |          |          | THE ALPHA KEYBOARD AS A COL 3    |
| 10.1       |              | NOTE           |          |          |          | USASCII NUMERAL.                 |
| 11         |              | ADK            | 0        | 0        |          | **DECIMAL CORRECT ALPHA          |

| <u>SEQ</u> | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>       | <u>B</u> | <u>C</u> | <u>REMARKS</u>             |
|------------|--------------|----------------|----------------|----------|----------|----------------------------|
| 12         |              | CDC            | 12             | 3        |          | COMPUTE CD USING REM 3     |
| 13         |              | PN             | 0              | 3        |          | PRINT CHECK DIGIT          |
| 14         | VERCD        | AL             | 1              |          |          | ALIGN FORM                 |
| 15         |              | POS            | 10             |          |          | POSITION PRINTER           |
| 16         |              | LKBR           | PARTNO         |          |          | SET BASE REGISTER POINTER  |
| 17         |              | TKM            | 6              |          |          | ENTER PART NUMBER & CD     |
| 18         |              | TRA            | PARTNO         |          |          | RD ALPHA TO ACCUMULATOR    |
| 19         |              | SLROS          | 0              | 4        |          | POSITION CD TO POS 0       |
| 20         |              | ADK            | 0              | 0        |          | DECIMAL CORRECT            |
| 21         |              | CDV            | 12             | 3        |          | VERIFY USING REM 3         |
| 22         |              | EX             | A              | S        | 2        | TEXT IF VERIFIED           |
| 23         |              | ALARM          |                |          |          | SIGNAL OPERATOR IF ERROR   |
| 24         |              | BRU            | VERCD          |          |          | BR TO RE ENTER             |
| 25         |              | PA             | OK             |          |          | PRINT VERIFIED MSG         |
| 26         |              | AL             | 1              |          |          | ALIGN FORM                 |
| 27         |              | BRU            | CMPCD          |          |          | BR FOR NEXT                |
| 28         | TABLE        | NUM            | 16600369258147 |          |          | FIRST ENTRY 7 WT CD TABLE  |
| 29         |              | NUM            | 26600741852963 |          |          | SECOND ENTRY 3 WT CD TABLE |
| 30         |              | NUM            | 06600987654321 |          |          | LAST ENTRY 1 WT CD TABLE   |
| 31         |              | MASK           | +,D            |          |          | PRINT CD ON CDC            |
| 32         | OK           | ALF            | OK             |          |          | VERIFIED MSG               |
| 33         |              | END            |                |          |          |                            |

**\*\*NOTE:** The eight bit alpha characters stored in the accumulator must be decimal corrected to eliminate hexadecimal values greater than 9 (A-F).

If a Modulus 11 method is used, the following additional instructions would be required in the VERCD Routine.

| <u>SEQ</u> | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>          |
|------------|--------------|----------------|----------|----------|----------|-------------------------|
| 20.1       |              | SKL            | 1        | 4        | 2        | SK IF CHECK DIGIT NOT A |
| 20.2       |              | INK            | 1        | 3        |          | RESET DIGIT 1 TO COL. 3 |
| 20.3       |              | INK            | 0        | A        |          | INSERT A IN COLUMN 0    |

These instructions are used to test and compensate for a check digit value of A, which is entered as an "A" (4,1 on the USASCII Chart). The 4,1 must be tested and compensated for or the alphanumeric number will not verify. The A must be corrected to the Col. 3 USASCII numeral that was derived during the compute phase (3A).

## 2.12 – DATA COMMUNICATIONS INSTRUCTIONS

### 2.12.01 GENERAL DESCRIPTION

The Data Communications Procedures and Configurations of the various TC's are covered in detail in the Series L/TC Equipment Reference Manual. The Equipment Reference Manual also discusses the basic characteristics of the Data Communications Processor and the way in which its associated firmware controls the interaction of the TC with the communications network and devices on that network.

Two tracks of the Data Communications Processor are permanently assigned as communications buffers, one for receiving messages from the network and one for sending messages to the network. Each buffer has a capacity of 255 characters of data plus the ETX character.

The Data Communications Processor firmware validates all incoming messages, removes the header information and stores the data (text) with the ETX in the receive buffer. Conversely, the Data Communications Processor firmware attaches the Header, ETX and BCC information to any outgoing message, the programmer being required to place only data (text, up to 255 characters) into the Transmit Buffer.

Messages to be transmitted are placed into the Data Communications Transmit buffer by the user program and the Transmit Ready Flag (R3 or D3) is set – See Subject 2.12.07. The Data Communications Processor will then handle the transmission of the message leaving the Main Memory Processor free to continue with the user program.

After the successful transmission of a message the Transmit Ready Flag (R3 and D3) will be reset. The user program should always examine the R3 flag (or the D3 flag which is the Data Communications Processor equivalent of R3) prior to placing another message into the Transmit Buffer to determine if the previous message has been transmitted.

In a data communications environment, the most efficient operation is achieved by using only the “D” flags.

The Data Communications Processor indicates to the user program that it has successfully received a message by setting the Data Communications Processor Receive Ready Flag (D2) and the Main Memory Processor Receive Ready Flag (R2) – Refer to Subject 2.12.07. The user program will interrogate one of these flags to determine when a message has been received.

After removing the data from the Receive buffer, the user program will reset the R2 or D2 flag to indicate to the Data Communications Processor that the buffer is free to receive another message.

The Data Communications Instructions covered in this section fall into three main groups, all are used in combination with the normal Main Memory instructions.

1. Send Instructions

These instructions provide for preparing messages to be transmitted from the TC.

2. Receive Instructions

These instructions provide for unpacking and processing messages that have been received by the TC





3. Control Instructions

These instructions provide for accessing and loading the various Terminal Addresses, Transmission Numbers, and other registers of the TC.

All of these instructions are executed as part of the user program. Their combined effect is to provide the most efficient handling of data communications with the TC.

**2.12.02 ESTABLISHING RECEIVE/TRANSMIT RECORD AREAS**

|                               | <u>LABEL</u> | <u>OP CODE</u> |
|-------------------------------|--------------|----------------|
| ESTABLISH RECEIVE RECORD AREA | RECEIV       | ESTB           |
| ESTABLISH SEND RECORD AREA    | SEND         | ESTB           |

It is usually desirable to use a receive record area to unpack messages while freeing the data comm receive buffer to accept more data. These receive record areas have a counterpart in the send record area, used to prepare a message for transmission while another message is in the transmit buffer awaiting a poll from the central processor.

These record areas are always thirty-two words (1 track) in length and are assigned space in memory by the assembler according to two things:

1. Memory size – as specified by the option “MEMORY NNN”
2. and by the use of the pseudo instruction ESTB.

The first use of the ESTB pseudo instruction will cause the assembler to assign the record area to the highest thirty-two words of memory available that fall on a track boundary (as indicated by the memory size option card) in user memory. The second use of the ESTB instruction will cause the record area to be established in the next 32 words of user memory available. For example, if user memory is 384 words, (0-383), the first record area will be in words 352-383. The second use of ESTB will establish the record area in words 320-351.

The ESTB pseudo instruction has no parameter, but it must always be labeled.

So far, we have only established receive and transmit record areas. The use of them will be discussed later.

**NOTE: If the last user word is specified in assembly rather than the total number of user words of user memory (example: 383 rather than 384), the assembler will select the next lower track available (example: words 320 to 352). This would cause the last 32 words to be inaccessible to the assembler for other use.**

An alternate, but less frequently used method of reserving main memory buffer areas is to specify a word value as in the following examples which assume 384 words of memory.

|      |    |
|------|----|
| LRBR |    |
| RCP  |    |
| IRCP |    |
|      | DC |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|--------------|----------------|----------|
|              | ORG            | 352      |
| RECEIV       | REG            | 32       |

In this example, Receive would be assembled with a starting word of 352. The word number must be the first word of a track. Track 0 is not a valid entry.

Any number of transmit or receive record areas may be used. The number is determined by system requirements and memory availability.

### 2.12.03 TRANSFERRING DATA FROM ONE MEMORY ADDRESS TO ANOTHER MEMORY ADDRESS

The unpacking of messages received and the constructing of messages to be transmitted usually involves moving data FROM one memory location TO another. The transfer can be from a record area to the transmit buffer, from the receive buffer to a memory location, or from one memory address to another memory address. The following instructions deal with this data movement.

|                              | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>       |
|------------------------------|--------------|----------------|----------------|
| LOAD RECEIVE BUFFER REGISTER |              | LRBR           | BLANK OR LABEL |

The LRBR instruction designates the starting memory address from which data will be transferred until the next LRBR is encountered, or the Character Pointer Register is otherwise altered. It is the origin address. The A parameter is the label of a memory address, often a record area which has already been established. The A parameter may be blank, however, in which case the data will be transferred directly from the Receive Buffer. Each time the LRBR instruction is executed, the character pointer for that record area or buffer is set to 1. This means the first character transferred will be the high order character of the first word in the designated memory location.

|                               | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|-------------------------------|--------------|----------------|----------|
| SET RECEIVE CHARACTER POINTER |              | RCP            | 1-255    |

Each use of the LRBR instruction sets the associated character pointer to one. For each character transferred or printed from the track, this character pointer is incremented serially. The RCP instruction sets the pointer to the character position specified by the "A" parameter relative to the last LRBR word location.

This instruction permits transfer of data starting with the character position designated by the "A" parameter.

|                                     | <u>OP CODE</u> | <u>A</u> |
|-------------------------------------|----------------|----------|
| INCREMENT RECEIVE CHARACTER POINTER | IRCP           | 1-255    |

The IRCP instruction increments the receive character pointer by the number of character positions designated in the A field, or until the next field identifier code is encountered. The pointer is incremented for the field identifier code also. This instruction permits by-passing a data field in a

|      |      |
|------|------|
| LKBR | SCP  |
| TRB  | TRBA |
|      | DC   |

message containing variable length fields. If the RCP is incremented past 255, the Overflow Test Flag will be set, otherwise it will be reset.

|                             | <u>OP CODE</u> | <u>A</u>       |
|-----------------------------|----------------|----------------|
| LOAD KEYBOARD BASE REGISTER | LKBR           | BLANK OR LABEL |

The LKBR instruction designates the starting memory address to which data will be transferred, until the next LKBR is encountered, or the Character Pointer Register is otherwise altered. It is the destination address. (The A parameter is the label of a memory address, often a record area.) The A parameter may be blank however, in which case the data will be transferred directly to the transmit buffer. Each time the LKBR instruction is executed, the Send Character Pointer for that memory address, record area or buffer is set to 1. This means the first character transferred will be placed in the first character position of the designated memory location.

|                            | <u>OP CODE</u> | <u>A</u> |
|----------------------------|----------------|----------|
| SET SEND CHARACTER POINTER | SCP            | 1-255    |

Each use of the LKBR instruction sets the associated character pointer to one. For each character transferred, the character pointer is incremented serially. The SCP instruction sets the character position specified-by the "A" parameter relative to the last LKBR word location.

This instruction permits transfer of data starting with the character position designated by the "A" parameter.

#### 2.12.04 UNPACKING MESSAGES RECEIVED

Normally, when transferring the contents of a word in the Accumulator, the whole word is transferred. Likewise, when printing the alpha contents of a word, the entire contents (up to an end alpha code) are printed. The data comm instructions used to unpack messages pay no attention to word boundaries in the receive buffer or receive record area. In Data Communication programing, it is possible to transfer any number of digits up to 16 to the Accumulator and it is possible to move alpha characters from one location to another regardless of the number of word boundaries crossed.

|  | <u>OP CODE</u> | <u>A</u> |
|--|----------------|----------|
| TRANSFER RECEIVE BUFFER TO RECORD AREA | TRB            | LABEL    |

The TRB instruction transfers the contents of the Data Communications Receive Buffer to the Normal Memory Receive Record area (32 words on one track) specified by the "A" parameter. The Receive Record area must have been established using the ESTB instruction previously described in this section. This instruction permits the use of one or several Receive Record areas in Normal memory.

|                                    | <u>OP CODE</u> | <u>A</u> |
|------------------------------------|----------------|----------|
| TRANSFER TO ACCUMULATOR AS NUMERIC | TRBA           | 1-16     |

The TRBA instruction, transfers the number of characters specified in the "A" field from the Receive Buffer, or working record area, to the Accumulator as Numeric digits. The buffer or Receive Record

TRBA

DC

area is the one specified by the last LRBR instruction, and the beginning character is determined by the current position of the RCP. The TRBA instruction is terminated by the transfer of the number of characters specified or by a field identifier code, whichever comes first. The field identifier code sets a specified flag pattern (see Subject 2.12.06). The RCP is incremented for each character transferred and for the field identifier code (which is not transferred into the Accumulator). The Overflow flag will be set if the RCP is incremented past 255 and the instruction will be terminated; otherwise, the Overflow flag is reset.

Although alpha numerals occupy 2 digit positions (8 bits) for the character in either the Receive Buffer or Receive Record area, the TRBA instruction places them in the Accumulator as numeric digits (4 bits). Thus, up to 16 buffer characters can be transferred to the Accumulator as 16 digits (any data required for computational purposes must be limited to 15 digits).

Valid codes accepted by TRBA are any codes from column 3 of the USASCII table. These include the numerals 0 to 9 and : ; < = > ? In addition, the minus (-) and plus (+) codes and any field identifier codes from columns 0 and 1 are valid. When used in a numeric field, the minus or plus code may be any character in the field. After first use in a given numeric field, subsequent plus or minus codes are invalid. The minus code will set the sign flag in the accumulator; the plus code will reset the sign flag. The minus or plus code will not be counted as one of the characters transferred as specified by the parameter field, however, the RCP will be incremented for this character. The field identifier codes are not transferred to the Accumulator but do terminate the TRBA instruction. The characters : ; < = > and ? are transferred to the accumulator as hexadecimal digits (undigits) with binary values of 10, 11, 12, 13, 14 and 15 respectively (values are designated by A, B, C, D, E, and F).

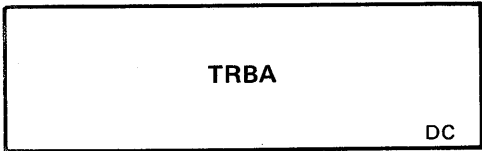
Other characters will be considered as invalid, will cause the "S" flag of the Accumulator to be set, will count as a code transferred, but the instruction will not be terminated.

Remember that if it is desired to read a terminating FI Code the TRBA parameter must be one more than maximum numeric field likely to be transferred in order to ensure that the FI Code is transferred and sets the flag patterns.

EXAMPLES

| Instruction | Buffer contents      | Result in accumulator  |
|-------------|----------------------|--|
| TRBA 4      | - 1234 ABC           | 1000000000001234   |
| TRBA 5      | - 1234 <u>FI</u> ABC | 1000000000001234   |
| TRBA 5      | 1234 - <u>FI</u> ABC | 1000000000001234   |
| TRBA 4      | 1234 - ABC           | 0000000000001234<br>(Sign is lost)                                       |
| TRBA 5      | 1234 + ABC           | 20000000000012341<br>(S flag is set by transfer of<br>A an invalid code) |

It is important to remember that the TRBA instruction, while designed to transfer one character at a time into the Accumulator, must "scoop up" two digit positions from the memory location indicated by the current LRBR and RCP instruction in order to determine the digit being transferred. Look at the USASCII chart (Appendix H). Every code in the table is represented by a row and column and must occupy 8 bits. The "numbers" in the table are located in column three. Since there are 16 rows in the



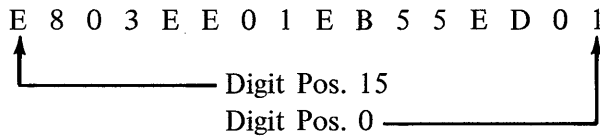
table, column 3 has 16 entries: 0-9 and the hexadecimal digits A through E. This information is useful when, for instance, an "A" is desired in the Accumulator as a result of a TRBA instruction. The central processor would send to the TC an USASCII equivalent of a colon (:). In USASCII code, it is "3,A." When the TRBA instruction encounters the 8 bit representation of a colon (3,A), the upper four bits are pared off and the lower four bits are placed in the Accumulator.

Used this way, the TRBA instruction is an instrumental tool for loading programs in the TC using codes sent from a central processor.

Example:

|                                    | <u>OP CODE</u> | <u>A</u> |
|------------------------------------|----------------|----------|
| TRANSFER TO ACCUMULATOR AS NUMERIC | TRBA           | 16       |

Result in Accumulator:



In this instance, the "E's," "B's," and "D's" in the Accumulator resulted from a 3,E, and a 3,B, and a 3,D in memory which are valid codes for the TRBA instruction. The "E" in the Accumulator is, in reality, a hexadecimal 14, the "B" a hexadecimal 11, and the "D" a hexadecimal 13.

**NOTE:** Let's say the contents of the Accumulator were moved to a memory location, e.g., word 30. Word 30 would then look like this:

- syllable 0 : ED01
- syllable 1 : EB55
- syllable 2 : EE01
- syllable 3 : E803

These are the machine codes for these mnemonics:

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>          |
|--------------|----------------|----------|----------|----------|-------------------------|
|              | AL             | 1        |          |          | Advance left 1          |
|              | POS            | 86       |          |          | Position to 86          |
|              | AR             | 1        |          |          | Advance right           |
|              | OC             | 3        |          |          | Open handler, advance 3 |



TRANSFER ALPHA

OP CODE

TRF

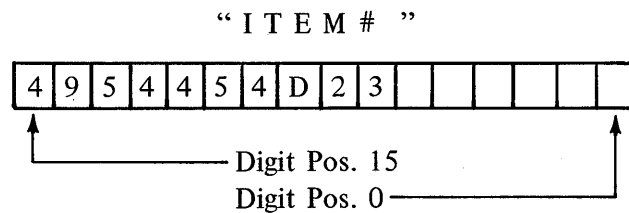
A

0-255

The TRF instruction transfers alphanumeric (8 bit) characters from the memory location specified by the last LRBR instruction beginning at the current RCP position to the memory location specified by the last LKBR instruction beginning at the current SCP position. The number of characters to be transferred is specified by the A parameter of the TRF instruction; the instruction is terminated by the transferring of the exact number of characters specified or by encountering a field identifier code. When the instruction is terminated, no matter how it is terminated, (by reaching the number of characters specified or by encountering a field identifier code) an end of alpha code will be inserted in the next character position of the memory address indicated by the LKBR. The SCP is not incremented for that code, however.

The following example attempts to show how several product codes, which have come from a central processor, can be stored in TC user memory:

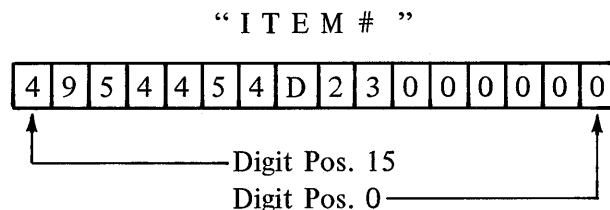
First word of Receive Buffer:



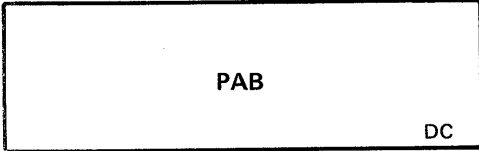
Example:

|                              | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|------------------------------|--------------|----------------|----------|
| LOAD RECEIVE BUFFER REGISTER |              | LRBR           | RECEIV   |
| LOAD KEYBOARD BASE REGISTER  |              | LKBR           | STORE    |
| TRANSFER ALPHA               |              | TRF            | 5        |
| RESERVE REGION               | STORE        | REG            | 1        |

This is what "STORE" would look like after the transfer:



The RCP and SCP are incremented for each character transferred; the RCP will also be incremented for a field identifier code if one is present. The overflow flag will be set if either pointer is incremented past 255, or if ETX is encountered.



PRINT ALPHA RECEIVE BUFFER

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| PAB            | 0-150    |

The PAB instruction usually is used with a receive buffer or record area but will print from any memory location designated by the last LRBR instruction beginning with the current RCP position. The printing will continue until the exact number of characters have been printed, or until a field identifier code is encountered. For each character printed, the RCP will be incremented by 1. If the RCP is incremented past 255, the overflow flag will be set. If printing is attempted beyond 150 on a 15½ inch platen, the system will return to ready mode.

Example:

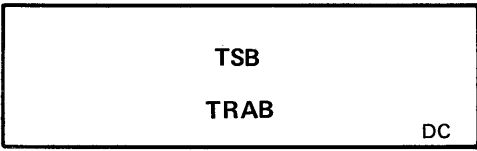
|                               | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|-------------------------------|--------------|----------------|----------|
| ESTABLISH RECEIVE RECORD AREA | RECEIV       | ESTB           |          |
| LOAD RECEIVE BUFFER REGISTER  |              | LRBR           | RECEIV   |
| PRINT ALPHA                   |              | PAB            | 15       |

**NOTE:** It is also possible to print from memory using the PA instruction. The distinction is the flexibility of the PAB instruction since it allows the programmer to designate a starting character position within a word (done by setting RCP) and to designate the exact number of characters to be printed. The PA instruction simply prints from the first character position of the word specified by its A parameter until it encounters the end alpha code.

**2.12.05 PREPARING MESSAGES FOR TRANSMISSION**

Remember from the discussion of unpacking messages received that instructions which transferred characters and printed characters were not limited by word boundaries. The transfer is guided by a character pointer (RCP). Likewise, in preparing a message for transmission, those instructions dependent on a character pointer (SCP) and an LKBR instruction are not limited by word boundaries.

If any of these instructions are used to transfer data to the transmit buffer while the transmit ready flag is set, execution of the instruction is delayed. The transmit ready flag is always interrogated before information is moved into the transmit buffer.



A message may be prepared for transmission in a user memory send record area and then be transferred to the transmit buffer. This transfer will move the entire 32 words of a send record area to the transmit buffer. The send record area is determined by the A parameter of the TSB instruction. The A parameter is the label of a record area established by one of the routines using ESTB. The End of Text Character will be automatically inserted after the last character of the message.

|                                |                |          |          |
|--------------------------------|----------------|----------|----------|
|                                | <u>OP CODE</u> | <u>A</u> |          |
| TRANSFER SEND RECORD AREA      | TSB            | LABEL    |          |
|                                | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| TRANSFER ACCUMULATOR TO "LKBR" | TRAB           | 0-15     | 0 or 1   |

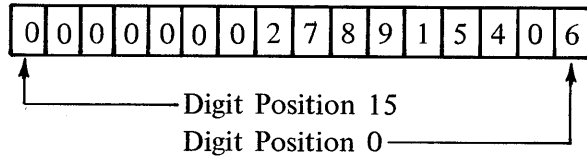
The TRAB instruction will transfer up to 15 numeric digits (4 bits) from the Accumulator into the memory location designated by the last LKBR instruction, placing the digits into memory as 8 bit alpha characters beginning with the current position of the SCP.

The digit position of the Accumulator from which digits are to be transferred is designated by the A parameter. The B parameter must be either a zero or one: A "1" meaning leading zeros will be transferred and a "0" meaning leading zeros will not be transferred.

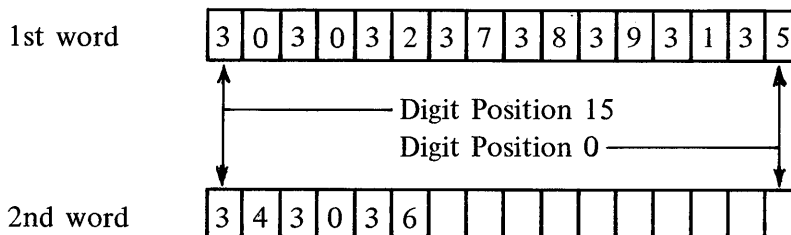
Example 1:

|                             |              |                |          |          |
|-----------------------------|--------------|----------------|----------|----------|
|                             | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| LOAD KEYBOARD BASE REGISTER | LKBR         | SEND           |          |          |
| TRANSFER ACCUMULATOR        | TRAB         | 10             | 1        |          |

If the Accumulator looks like this prior to execution of TRAB:



then the digit 0 located in position 10 would be transferred to the current position of the SCP as the character 0 (represented in hexadecimal as 30). The digit 0 in position 9 of the Accumulator would be transferred as the character 0 (represented in hexadecimal as 30); digit 2 would transfer as character 2 (hexadecimal 32); etc. The first and second words of the memory location designated by the last LKBR would look like this after the execution:





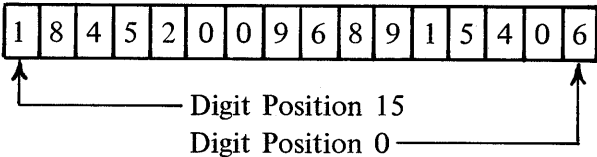


The transfer could also have been directly to the Data Communications Transmit Buffer.

Example 2:

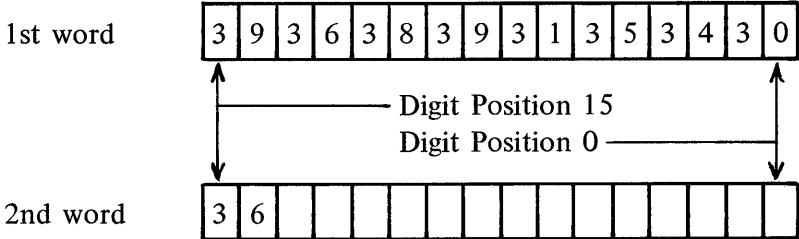
|                                     | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-------------------------------------|----------------|----------|----------|
| LOAD KEYBOARD BASE REGISTER         | LKBR           |          |          |
| TRANSFER ACCUMULATOR TO LAST "LKBR" | TRAB           | 10       | 0        |

If the Accumulator looked like this prior to execution:



then the first digit transferred would be the digit 9 in position 8 of the Accumulator, since the B parameter indicates zero suppression. It would be transferred to the current position of the SCP as the character 9 (hexadecimal 39). The digit 6 in position 7 would transfer as character 6 (hexadecimal 36), etc.

The first and second words of the memory location designated by the last LKBR would look like this:



Those digits occupying positions in the Accumulator higher than the digit position specified by the A parameter were ignored.

Example 3: Transferring signed numbers.

|                                     | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|-------------------------------------|----------------|----------|----------|----------|
| LOAD KEYBOARD BASE REGISTER         | LKBR           | WORK     |          |          |
|                                     | EX             | A        | -        | 1        |
| TRANSFER CHARACTER                  | TRCB           | 2        | 13       |          |
| TRANSFER ACCUMULATOR TO LAST "LKBR" | TRAB           | 9        | 0        |          |

|      |    |
|------|----|
| TRAB | DC |
| TRF  |    |
| TRCB |    |

If the Accumulator appears like this prior to execution:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 5 | 5 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

then after execution, the memory location specified by the last LKBR would appear as follows:

|   |   |   |   |   |   |   |   |   |   |   |   |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|--|--|--|--|
| 2 | D | 3 | 1 | 3 | 2 | 3 | 5 | 3 | 5 | 3 | 0 |  |  |  |  |
| - |   | 1 |   | 2 |   | 5 |   | 5 |   | 0 |   |  |  |  |  |

It is necessary to test for the presence of the minus flag in the Accumulator and to insert the actual minus character (hexadecimal 2D) into memory, since a minus flag would be converted to the character 1 (hexadecimal 31) by the TRAB instruction.

To insert a plus sign into memory, the following code could be used:

|                                | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|--------------------------------|----------------|----------|----------|----------|
| LOAD KEYBOARD BASE REGISTER    | LKBR           | WORK     |          |          |
|                                | }              | }        |          |          |
|                                | SK             | A        | -        | 1        |
| TRANSFER CHARACTER             | TRCB           | 2        | 11       |          |
| TRANSFER ACCUMULATOR TO MEMORY | TRAB           | 9        | 0        |          |

|                | <u>OP CODE</u> | <u>A</u> |
|----------------|----------------|----------|
| TRANSFER ALPHA | TRF            | 0-255    |

Refer to previous discussion on this instruction under Subject 2.12.04 "Unpacking Messages Received".

|                              | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|------------------------------|----------------|----------|----------|
| TRANSFER CHARACTER TO BUFFER | TRCB           | 0-7      | 0-15     |

The TRCB instruction transfers the USASCII code designated by the decimal value in the "A" and "B" parameters into the memory address specified by the last LKBR instruction, with the first character being transferred to the position indicated by the current position of the SCP. For each character transferred, the SCP is incremented by one.

To use this instruction, it is necessary to know the USASCII row and column designation of the character to be transferred. The A parameter indicates the column number from the USASCII table, and the B parameter is the row number.

For example, if an asterisk (\*), USASCII column 2, row 10, is to be placed in the buffer, then the instruction to accomplish this is:

|                         |
|-------------------------|
| TRCB<br>TKM<br>FI CODES |
|-------------------------|

|                              | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|------------------------------|----------------|----------|----------|
| TRANSFER CHARACTER TO BUFFER | TRCB           | 2        | 10       |

|                | <u>OP CODE</u> | <u>A</u> |
|----------------|----------------|----------|
| TYPE TO MEMORY | TKM            | 0-150    |

The TKM instruction allows the operator to enter data directly into the memory address specified by the last LKBR beginning with the current position of the SCP. The SCP will be incremented for each character entered and an end of alpha code will be placed in memory after the last character ended. However, the SCP is not incremented for this character.

The use of the backspace key will cause the SCP to be decremented for each depression. However, the SCP cannot be decremented beyond the position held when the TKM instruction was encountered.

|                         | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|-------------------------|--------------|----------------|----------|
| LOAD KEYBOARD REGISTER  |              | LKBR           | AREA     |
| TYPE INTO MEMORY        |              | TKM            | 16       |
| ESTABLISH 4 WORD REGION | AREA         | REG            | 4        |

The instruction may have been used to enter data into the transmit record area:

|                             | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|-----------------------------|--------------|----------------|----------|
| LOAD KEYBOARD BASE REGISTER |              | LKBR           | SEND     |
| TYPE INTO LAST "LKBR"       |              | TKM            | 25       |
| ESTABLISH SEND RECORD AREA  | SEND         | ESTB           |          |

## 2.12.06 FIELD IDENTIFIER CODES AND VARIABLE LENGTH FIELDS

### EXAMPLE:

A customer's name, street address, city and state are being transmitted to the TC to be printed on 3 different lines of an invoice. The message is in the Receive Buffer and the programmer wishes to use the PAB instruction to print the name on the ship-to portion of the invoice. If the name is "Acme Printing," the A parameter of the PAB instruction should be 13 characters. Names may be of variable length, and a convention in GP 300 allows for varying length fields. This convention is called a "field identifier code." Whenever a field identifier code is encountered by any of the following data comm instructions, execution is terminated and the next instruction will begin. These instructions are:

**FI CODES**

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>                      |
|--------------|----------------|----------|----------|----------|-------------------------------------|
|              | TRBA           | 0-16     |          |          | Transfer as numeric                 |
|              | TRF            | 0-255    |          |          | Transfer alpha                      |
|              | PAB            | 0-150    |          |          | Print from buffer                   |
|              | IRCP           | 0-255    |          |          | Increment receive character pointer |

Valid field identifier codes are in columns 0 and 1 of the USASCII Chart. The two charts below show the codes, their 4 bit hexadecimal value and their accompanying flag patterns.

The codes from column 0 present problems if the "Y" flags are used in the TC user program. After reading a column 0 field identifier code, all four Y flags are either set or reset, and the appearance of these Y flags could seriously upset the logic of the TC program if the Y flags are interrogated and acted upon without knowledge of these additional flag settings. This same problem could arise when reading column 1 codes and when interrogating the K flags. Therefore, the use of these field identifier codes must be given careful consideration and their use must be coordinated with the central processor.

| <b>NO FLAGS SET</b> | <b>Y FLAGS SET*</b>   | <b>K FLAGS SET*</b>  | <b>TEST FLAGS SET</b> |
|---------------------|---|--|-----------------------|
|                     | 3 2 1 4   | 3 2 1 4  | U I L O               |
| NUL                 | SOH 0 0 0 1<br>STX 0 0 1 0  | DC1 0 0 0 1<br>DC2 0 0 1 0<br>DC3 0 0 1 1<br>DC4 0 1 0 0   | ETX 0 0 0 1           |
|                     | ENQ 0 1 0 1<br>ACK 0 1 1 0<br>BEL 0 1 1 1<br>BS 1 0 0 0<br>HT 1 0 0 1<br>LF 1 0 1 0<br>VT 1 0 1 1<br>FF 1 1 0 0<br>CR 1 1 0 1<br>SO 1 1 1 0<br>SI 1 1 1 1 | NAK 0 1 0 1<br>SYN 0 1 1 0<br>ETB 0 1 1 1<br>CAN 1 0 0 0<br>EM 1 0 0 1<br>SUB 1 0 1 0<br>ESC 1 0 1 1<br>FS 1 1 0 0<br>GS 1 1 0 1<br>RS 1 1 1 0<br>US 1 1 1 1 |                       |

\*Y and K flags designated are set if "1" and reset if "0"

It is generally agreed that many of the above USASCII codes should never appear in a text. EOT is specifically filtered out by the Data Communications Processor. NUL does serve as a field identifier but, as indicated in the chart above, it terminates the instruction but does not set any flags; neither does it reset any previous flags. It merely terminates the instruction. ETX has special significance in that when ETX is detected during a transfer instruction, the Overflow flag will be set and the instruction terminated.

|          |
|----------|
| FI CODES |
|----------|

The following examples show the proper use of field identifier codes.

Example 1:

An invoice ship-to region has been defined as consisting of from 2 to 4 lines of not more than 25 characters per line. In addition, the last line of the ship-to address will determine if the sold-to address is "SAME" or if it requires a separate address.

**PROBLEM:** The TC programmer must program for variable length fields and for a variable number of fields. He must also decide whether to print "SAME" in the sold-to address area or to begin printing a new sold-to address.

**DECISION:** After each field or line of ship-to address a field identifier code will be inserted by the central processor. For example, "DC1," after each line except for the last line of the ship-to address which will be "DC2" if the sold-to address is "SAME" or a "DC4" if sold-to address is another distinct address. A "CAN" code will terminate the last line of the sold-to address.

On the following page are some programing suggestions that will accomplish the necessary invoice addressing routine. (Assume the necessary steps have been taken to establish a receive record area, to establish alpha constants, etc.)

This routine is very flexible. Each line printed can be of any length up to 25 characters. If the field (line) is less than 25 characters\*, the field identifier will terminate the instruction and set a K flag pattern. Also, there may be any number of lines to an address since either K1 or K2 will mark the end of the last line of the address.

**\*Notice the A parameter of the PAB instruction is 26. The problem definition permits only 25 characters per line. In the event, however, the field is exactly 25 characters long, the extra character in the A parameter will allow the PAB instruction to pick up the field identifier code. Otherwise, the character pointer will be pointing at the 26th character at the time of execution of the next PAB instruction since it is not incremented when reading an F.I. This PAB instruction would read the field identifier and terminate, instead of reading the next field.**

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>#</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>               |
|--------------|----------------|----------|----------|----------|----------|------------------------------|
| PRTLIN       | LRBR           | RECEIV   |          |          |          | Load Receive Buffer Register |
|              | AL             | 1        |          |          |          | Advance left 1 line          |
|              | POS            | 5        |          |          |          | Position to print            |
|              | PAB            | 26       |          |          |          | Print on address line        |
|              | EX             | K        |          | 4        | 1        | K4 – means more lines        |
|              | BRU            | PRTLIN   | +1       |          |          | Print another line           |
|              | EX             | K        |          | 1        | 3        | K1 – ship-to = sold-to       |
|              | ALTO           | 15       |          |          |          | Advance to sold-to area      |
|              | PA             | SAME     |          |          |          | Print “SAME”                 |
|              | BRU            | RIBBON   |          |          |          | Exit the routine             |
|              | EX             | K        |          | 2        | 2        | K2 – means sold-to address   |
|              | ALTO           | 15       |          |          |          | Advance to sold-to area      |
|              | BRU            | PRTLIN   | +1       |          |          | Base to print new address    |
| RIBBON       | ALTO           | 22       |          |          |          | Ribbon Routine               |

Example 2:

This example shows how field identifier codes may be helpful while constructing messages for transmission to the central processor.

Assume we are in a file maintenance routine and wish to send the name and number of a customer to the central processor. Every name has a corresponding number.

**PROBLEM:** The TC programmer must allow for several such combinations of names and numbers and also must distinguish between the names and numbers.

**DECISION:** Every name will be followed by the field identifier "DC2." Every customer number will be identified by a trailing "DC4" if there are more names and numbers to follow or a "CAN" if the current customer number is the last one. After indexing a name, the operator terminates with OCK 1. After indexing a number, the operator terminates with OCK 2 if there are more names and numbers and OCK 3 or OCK 4 if there are no more.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | #  | <u>B</u> | <u>C</u> | <u>REMARKS</u>       |
|--------------|----------------|----------|----|----------|----------|----------------------|
| LODBUF       | LKBR           | XMIT     |    |          |          | Load transmit buffer |
|              | AL             | 2        |    |          |          | Advance to type      |
|              | POS            | 5        |    |          |          | Position to print    |
|              | TKM            | 25       |    |          |          | Index name/number    |
|              | EX             | K        |    | 1        | 2        | K1 – means name      |
|              | TRCB           | 1        |    | 1        |          | 1,2 = DC2 = OCK 1    |
|              | BRU            | LODBUF   | #1 |          |          | Index again          |
|              | EX             | K        |    | 2        | 2        | K2 – means number    |
|              | TRCB           | 1        |    | 2        |          | 1,4 = DC4 = OCK 2    |
|              | BRU            | LODBUF   | #1 |          |          | Index again          |
|              | EX             | K        |    | 3,4      | 3        | K3,4 – last number   |
|              | TRCB           | 1        |    | 8        |          | 1,8 = CAN = OCK 3    |
|              | SET            | D        |    | 3        |          | Set transmit flag    |
|              | BRU            | AWAY     |    |          |          | Exit routine         |

The function of the "D" flag group is to provide a method for interrogating and changing the status of the DCP Transmit and Receive Buffers. The "R" flag group may also be utilized in the same manner as the "D" flag group. However it is recommended that the "D" flag group be used due to timing and syllable placement considerations involved in using the "R" flags.

|                |
|----------------|
| <b>D FLAGS</b> |
| <b>RSA</b>     |
| DC             |

**2.12.07 "D" FLAG GROUP**

All versions of the Series L/TC Assemblers which have the capability of assembling a data communications program have been revised to allow any flag in the D Flag Group to be set (SET) and reset (RST). Previously the D flags could be interrogated but the status could not be altered. When it was necessary for the application program to notify the DCP of a change in the status of the Transmit and Receive buffers, it had to be done via the R2 (Ready to Receive new data) or R3 (message ready for transmission) flags.

It is suggested that only the D flag group be used when it is required to set, reset or interrogate the status of the DCP. The previous method of setting or resetting the R flags and interrogating the D flags, although confusing, will also work.

**IMPORTANT:** The CHG or LOD instructions can not be used to change or load the R or D flag groups when the TC is functioning with any Data Communications Main Memory Firmware Set. The CHG or LOD Instructions may be used to change or load the R flags only when using any non Data Communication firmware set.

The following flags are available in the Data Communication Flag Group:

- D1 – Trouble Flag
- D2 – Message Received Flag
- D3 – Transmit Ready Flag
- D4 – Micro Flag. Not available to the macro programmer.

**2.12.08 SEND AND RECEIVE ADDRESS INSTRUCTIONS**

GP 300 has a group of instructions, which allow the programmer to assume some firmware responsibilities. An example is the transmission number that is part of the header portion of a message. This number is usually calculated by firmware and can be an important programming consideration. There are two instructions in GP 300 that allow the programmer to transfer the transmission number to the Accumulator and also to assign any 1, 2, or 3 digit number as the transmission number. The transmission number must initially be set by the programmer to effectively check for lost messages.

**OP CODE**

RETRIEVE SEND ADDRESS

RSA

This instruction transfers the two-character send machine address from the send address register in the Data-Communications Processor into the four (4) most significant digit positions of the Accumulator. The balance of the Accumulator will be zero.

Example: If Send Machine Address is: 1A, Accumulator will be as follows:

|    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |                            |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|----------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ACCUMULATOR DIGIT POSITION |
| 3  | 1  | 4  | 1  | 0  | 0  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | VALUE                      |



|     |
|-----|
| LSA |
| RRA |
| LRA |

These two characters may be any characters from columns 2 through 6 of the USASCII set (except circumflex and underline). With a range of 78 different characters in each of the two positions, the total machine address range potential would be 6,084 different combinations.

OP CODE

LOAD SEND ADDRESS LSA

This instruction transfers the four most significant digits of the Accumulator into the Send Machine Address Register in the Data Communications Processor. Only the 4 most significant digits of the Accumulator may contain significant digits (i.e., 2 characters). The balance of the Accumulator must contain zeros.

Example for loading Send Address:

|      |      |      |                           |
|------|------|------|---------------------------|
|      | LKBR | WORK | DESIGNATE MEMORY AREA     |
|      | TKM  | 2    | ENTER 2 CHARACTER ADDRESS |
|      | TRA  | WORK | TRANSFER TO ACCUMULATOR   |
|      | LSA  |      | LOAD SEND ADDRESS         |
| WORK | REG  | 1    | RESERVE MEMORY AREA       |

OP CODE

RETRIEVE RECEIVE ADDRESS RRA

This instruction functions in exactly the same fashion as RSA, except it will transfer the machine address from the Receive Address Register in the Data Communications Processor into the four (4) most significant digit positions of the Accumulator. The balance of the Accumulator will contain zeros. Generally the Receive and Send Machine Addresses are alike, however, a condition can exist where they could be different.

Normally, in addition to the Receive and Send addresses, the TC has a permanent machine address, located in word 1064. This address is loaded into the Send and Receive register every time power is turned on or when the program halt button is used. The Ready Button has no effect on Send or Receive addresses.

The Permanent machine address can be changed by unprotecting block 4, track 4, and using Memory Modify.

OP CODE

LOAD RECEIVE ADDRESS LRA

This instruction transfers the contents of the accumulator into the Receive Machine Address Register in the Data Communications Processor. Only the four (4) most significant digit positions of the Accumulator may contain significant characters. The balance of the Accumulator must contain zeros.

Refer to example for loading send machine address.

|     |
|-----|
| RSN |
| LSN |
| DC  |

**2.12.09 TRANSMISSION NUMBERS**

The TC may maintain a transmission number that accompanies every message it sends to a central processor. It may be a one, two or three digit number, or no transmission number. A separate transmission number is maintained for normal transmission, group select and broadcast select.

If the transmission number is one digit only, it will return to zero every ten transmissions. If it is a two-digit number, it will return to zero after each one hundred transmissions, and for a three-digit number after every thousand transmissions.

The Send Transmission number is included in the header of all data transmissions from the terminal and is automatically incremented by 1 when transmission has succeeded so that the next message will carry the next transmission number in sequence.

The Expected Receive Transmission number is maintained by the data communications processor, and automatically compared with the actual transmission number on all data messages received from the data center.

If a message is received successfully from the data center, the expected transmission number is incremented in anticipation of the next message transmission number.

If the transmission number from the data center does not agree with the expected transmission number in the TC, the transmission failure flag (D1) is set. This flag can be interrogated by the user program for necessary recovery procedures. The D1 flag will be reset by the next transmission received, unless the number still does not agree.

OP CODE

RETRIEVE SEND TRANSMISSION NUMBER

RSN

This instruction transfers the 1, 2, or 3 digit USASCII Send Transmission Number from its register into the 2, 4, or 6 most significant digit positions of the Accumulator. The balance of the Accumulator will contain zeros. The user program will process the send transmission number depending on requirements.

OP CODE

LOAD SEND TRANSMISSION NUMBER

LSN

Execution of this instruction will cause transfer of the Accumulator to the Send Transmission Number Register. Only the 2, 4, or 6 high order digit positions may contain significant digits. The rest must contain zeros. (The number of positions in the Accumulator that may contain significant digits is determined by the length of the Send Transmission Number – 1, 2, or 3 digits.)

**NOTE: The Transmission Number must be in the high order positions of a word. IT IS IMPERATIVE THAT THE SEND TRANSMISSION NUMBER BE SET UP AS USASCII NUMERALS. IF THE NUMBER IS SET UP IN THE NUMERIC MODE (4 BIT DIGITS), COLUMN 0 USASCII CODES WILL BE INSERTED IN THE HEADER PORTION OF THE MESSAGE WHICH WILL EVENTUALLY CAUSE A DATA LOSS WHEN THE TC ATTEMPTS TO TRANSMIT THE MESSAGE AFTER INCREMENTING THE TRANSMISSION NUMBER.**

|     |     |
|-----|-----|
| RTN | LGN |
| LTN | RBN |
| RGN |     |
|     | DC  |

As a result of the addition of fast select, group select, and broadcast select three sets of expected transmission numbers are maintained by 2-1044-006-00 in those processors that use TR numbers. The following instructions are provided to load and retrieve the three sets of transmission numbers. Standard Select and Fast Select use the same Expected Transmission number.

OP CODE

RTN

RETRIEVE EXPECTED TRANSMISSION NUMBER

The RTN instruction transfers the 1, 2 or 3 USASCII numeric Character “Expected Transmission Number” from its appropriate Register into the 2, 4, or 6 most significant digit positions of the Accumulator. The balance of the Accumulator will contain zeros. This instruction retrieves the Expected Transmission Number (word 1190) for Select and Fast Select messages.

OP CODE

LTN

LOAD EXPECTED TRANSMISSION NUMBER REGISTER

The LTN instruction transfers the contents of the Accumulator into the Expected Transmission Number Register for messages received. Only the 2, 4, and 6 most significant digit positions of the Accumulator may have significant characters. The expected Transmission number may be up to 3 USASCII numeric characters in length. The balance of the Accumulator must contain zeros. This instruction loads the expected Select and Fast Select Transmission number.

OP CODE

RGN

RETRIEVE EXPECTED GROUP TRANSMISSION NUMBER

The RGN instruction transfers the Expected Group Transmission Number from the Expected Group Transmission Number Register (word 1192) to the Accumulator. The Expected Group Transmission number may be up to 3 USASCII numerals in length and will occupy the most significant positions in the Accumulator. The remaining positions are ignored.

OP CODE

LGN

LOAD EXPECTED GROUP TRANSMISSION NUMBER

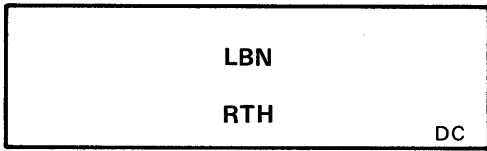
The LGN instruction transfers the contents of the Accumulator into the Expected Group Transmission Number Register. The Expected Group Transmission number may be up to 3 USASCII numeric characters (left justified) in length and although the entire Accumulator is transferred, the remaining locations are ignored.

OP CODE

RBN

RETRIEVE EXPECTED BROADCAST TRANSMISSION NUMBER

The RBN instruction transfers the Expected Broadcast transmission number from its register (word 1193) to the Accumulator. The Expected Broadcast Transmission number may be either 0, 1, 2 or 3 USASCII numeric characters in length and is contained in the most significant digit positions of the word in the Data Communications Processor and, after the transfer, in the most significant digit positions of the Accumulator. Any remaining digit positions are ignored.



**LOAD EXPECTED BROADCAST TRANSMISSION NUMBER**      OP CODE  
LBN

LBN transfers the contents of the Accumulator into the Expected Broadcast Transmission Number Register in the Data Communications Processor. The Expected Broadcast Transmission Number may be 0, 1, 2 or 3 USASCII numeric characters located in the most significant positions of the Accumulator. The remaining positions, although transferred, are ignored.

**RETRIEVE TRANSMISSION HEADER**      OP CODE  
RTH

The Retrieve Header Transmission Number (RTH) instruction transfers the Transmission Header Register (word 1184) into the Accumulator. This register is loaded with the 8 characters following the start of header (SOH) character of any message received whether by select, fast select, group select, or broadcast select. Among these 8 characters will be the transmission number of the message received if the DCP uses TR numbers. The numbers will be in their 8-bit USASCII representation. The format of this register for each of the four cases (0, 1, 2, or 3 transmission numbers) is shown below. When necessary to determine the communications procedure used by the data center, a character in the text of the message can be used to indicate how the message was transmitted. Below is the format of the transmission header register for 0, 1, 2, and 3 transmission number systems.

**3 Transmission Numbers**

|                    |     |     |     |     |     |     |           |   |
|--------------------|-----|-----|-----|-----|-----|-----|-----------|---|
| Character Position | 7   | 6   | 5   | 4   | 3   | 2   | 1         | 0 |
|                    | AD1 | AD2 | TR# | TR# | TR# | STX | TEXT DATA |   |

**2 Transmission Numbers**

|                    |     |     |     |     |     |           |   |   |
|--------------------|-----|-----|-----|-----|-----|-----------|---|---|
| Character Position | 7   | 6   | 5   | 4   | 3   | 2         | 1 | 0 |
|                    | AD1 | AD2 | TR# | TR# | STX | TEXT DATA |   |   |

**1 Transmission Number**

|                    |     |     |     |     |           |   |   |   |
|--------------------|-----|-----|-----|-----|-----------|---|---|---|
| Character Position | 7   | 6   | 5   | 4   | 3         | 2 | 1 | 0 |
|                    | AD1 | AD2 | TR# | STX | TEXT DATA |   |   |   |

**0 Transmission Number**

|                    |     |     |     |           |   |   |   |   |
|--------------------|-----|-----|-----|-----------|---|---|---|---|
| Character Position | 7   | 6   | 5   | 4         | 3 | 2 | 1 | 0 |
|                    | AD1 | AD2 | STX | TEXT DATA |   |   |   |   |

**2.12.10 SPECIAL PURPOSE REGISTERS**

The TC may be connected to a central processor two ways. When operating in two-wire direct connect (TDI) or over leased duplex (four-wire) lines, a four-wire mode must be specified. When operating over a switched line or through half-duplex (two-wire) leased line, two-wire mode must be specified.

|                   |
|-------------------|
| RTF<br>LTF<br>RPR |
|-------------------|

The Data Communications Processor contains a special register to enable two or four wire transmission mode. One bit in this register is used to determine which mode is active.

RETRIEVE TWO/FOUR WIRE REGISTER

OP CODE  
RTF

Execution of this instruction will transfer the contents of the two wire/four wire register into the Accumulator. When the Accumulator M Flag is on, the mode is two wire; when it is off the mode is four wire. Like other "Retrieve" instructions, the user program must then interrogate the Accumulator flag and perform according to program requirements.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>              |
|--------------|----------------|----------|----------|----------|-----------------------------|
|              | LTF            |          |          |          | Load Two/Four Wire Register |

Execution of this instruction will transfer the contents of the Accumulator into the Two or Four wire register. The mode will then be 2 or 4 wire depending upon the status of the Accumulator M flag at the time of execution.

RETRIEVE POINTER REGISTER

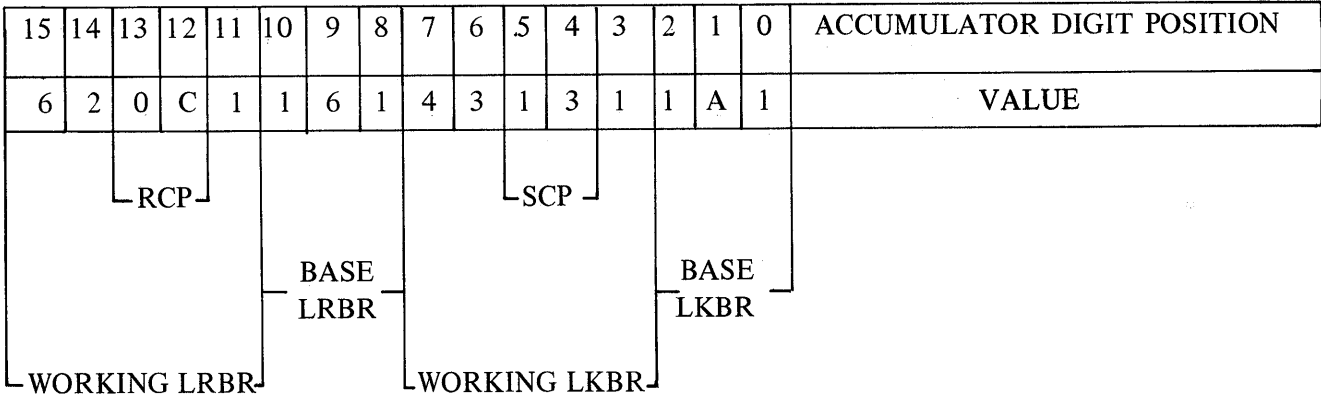
OP CODE  
RPR

This instruction will transfer the contents of the Character Pointer Register into the Accumulator. All digits in the Accumulator will be hexadecimal and the format of the Accumulator will be as follows:

|     |
|-----|
| RPR |
|-----|

Example 1:

|     |     |         |         |
|-----|-----|---------|---------|
|     | B B |         | B B     |
| W W | L L | W W W W | L L W W |
| O O | O O | O O O O | O O O O |
| R R | C C | R R R R | C C R R |
| D D | K K | D D D D | K K D D |



|                           |       |     |       |    |  |
|---------------------------|-------|-----|-------|----|--|
| BASE LRBR                 | BLOCK | 1   |       |    |  |
|                           | WORD  | 97  | (352) |    |  |
| WORKING LRBR              | BLOCK | 1   |       |    |  |
|                           | WORD  | 98  | (353) |    |  |
| RECEIVE CHARACTER POINTER |       |     |       | 12 |  |
| BASE LKBR                 | BLOCK | 1   |       |    |  |
|                           | WORD  | 161 | (416) |    |  |
| WORKING LKBR              | BLOCK | 1   |       |    |  |
|                           | WORD  | 67  | (322) |    |  |
| SEND CHARACTER POINTER    |       |     |       | 19 |  |

When controlling the loading of a buffer, it is necessary to be able to check the buffer capacity at the start of each line of the message.

We must establish that we can put another full message line in the buffer.

To do this, we use a technique which examines the SCP

|            |              |  |  |         |
|------------|--------------|--|--|---------|
| Example 2: | Buffer size  |  |  | 255 CH+ |
|            | Maximum Line |  |  | 60 CH-  |
|            | SCP Limit    |  |  | 195 CH  |

SCP is hexadecimal. Therefore,

$$195 = C3 \text{ in digit positions 5-4}$$

|     |          |    |   |  |                            |
|-----|----------|----|---|--|----------------------------|
| RPR |          |    |   |  | RETRIEVE POINTER REGISTER  |
| SKL | 5        | 12 | 3 |  | TEST FOR UPPER DIGIT < 12  |
| EXL | 5        | 13 | 1 |  | TEST FOR UPPER DIGIT = 12  |
| SKL | 4        | 4  | 1 |  | TEST FOR LOWER DIGIT < 4   |
| BRU | TRANSMIT |    |   |  | IF NO. OF CHARACTERS > 195 |
| BRU | CONTINUE |    |   |  | IF NO. OF CHARACTERS < 195 |

|            |              |  |  |         |
|------------|--------------|--|--|---------|
| Example 3: | Buffer Size  |  |  | 255 CH+ |
|            | Maximum Line |  |  | 63 CH-  |
|            | SCP Limit    |  |  | 192 CH  |

Therefore, SCP = C0

|     |          |    |   |  |                            |
|-----|----------|----|---|--|----------------------------|
| RPR |          |    |   |  | RETRIEVE POINTER REGISTER  |
| EXL | 5        | 12 | 1 |  | TEXT FOR UPPER DIGIT < 12  |
| BRU | CONTINUE |    |   |  | IF NO. OF CHARACTERS < 192 |
| BRU | TRANSMIT |    |   |  | IF NO. OF CHARACTERS > 192 |

|                       |                |
|-----------------------|----------------|
|                       | <u>OP CODE</u> |
| LOAD POINTER REGISTER | LPR            |

Execution of this instruction will transfer the contents of the Accumulator into the Character Pointer Register.

## **SUBJECT 2.13 – POINT-TO-POINT PROGRAMING PROCEDURES**

### **2.13.01 BASIC POINT-TO-POINT LINE DISCIPLINE**

Point-to-Point Firmware provides the TC with a contention type line control procedure which allows Series TC Computers to communicate on an equal basis with another Data Communications Unit (CPU or another TC). The basic Point-to-Point line discipline does not provide a terminal addressing scheme nor a transmission number sequence. Since an address scheme is not provided, only two units can be listening to the line at any given time. When operating in this mode, the TC can communicate with a CPU or another TC. When this Line Discipline is implemented, either unit on the line can initiate transmissions without previously being interrogated (TC does not have to be polled).

In a Point-to-Point environment, the TC must normally contend for control of the line before it can transmit a message. After a successful transmission is completed, control of the line is given to the receiving unit. The receiving unit may then, if transmit ready, transmit a message without having to contend for control. If the receiving unit is not transmit ready, the sequence is terminated.

A TC can operate in a switched line, leased line or a direct connect communication network when utilizing Point-to-Point Firmware.

Point-to-Point Data Comm Processor (DCP) firmware is compatible with all standard main memory Data Comm Firmware sets.

### **2.13.02 CONTROL REGISTERS**

Five Control Registers are provided which allow user program control of the functions listed below.

Time Out Limit  
Demand Disconnect  
Idle Line Disconnect  
Line Mode (2 wire or 4 wire)  
NAK/NO Response Limit

The Control Registers can be controlled by user program or they can be set manually when the TC is installed. To avoid the possibility of human error it is recommended that the Control Registers be set by the user program.

a. Accessing of Control Registers:

The Control Registers are stored in the DCP memory (word 1188) and are accessed via the RTF and LTF macro instructions. The RTF instruction retrieves the control register word and stores it in the accumulator for manipulation. The LTF instruction transfers the contents of the accumulator to the control register word in the DCP memory.

b. Control Word Format:

The five control registers are arranged within the control word in the following manner:



LINE MODE – Digit Position 15.  
 NAK/NO RESPONSE LIMIT – Digit positions 7 and 6.  
 TIMEOUT LIMIT – Digit positions 5 and 4.  
 DEMAND DISCONNECT – Digit position 2.  
 IDLE LINE DISCONNECT – Digit positions 1 and 0.

The digit positions within the control word which are not used must be set to zero.

c. Time Out Limit Register:

The length of time which the TC will wait for a response after transmitting is determined by the value in the Time Out Limit Register.

The Time Out value for the two units must be different to avoid “locking up” the line when both units are contending for control of the line at the same time. The optimum difference between timeout values is 500 milliseconds and this difference should be maintained if possible.

The time out value in the TC can vary from 0 to 2550 milliseconds. To determine the minimum time out value double the turn around time of the data set being used and add 100 milliseconds.

EXAMPLE:

If communication is over switched lines, using 202C data sets, the Time Out Limit should be determined and set as follows:

Minimum Time Out = Time X  
 Maximum Time Out = Time Y  
 Turnaround time of 202C data set = 200 milliseconds.

Time X = (200ms) (2) + 100 ms = 500 ms  
 Time Y = Time X + 500 ms = 1000 ms

The value which is inserted into the Time Out Limit Register to achieve the desired time out limit is the hexadecimal representation of 1/10 of the desired time out limit.

EXAMPLE:

Using the time out limit for time X computed in the example above, the values which would be inserted into the Time Out Limit Register would be as follows:

Value for TIME X =  $1/10 \times 500 = 50$ .  
 50 expressed hexadecimally = 32.

A hexadecimal value of 32 would be inserted into the Time Out Limit Register to achieve a Time Out Limit of 500 milliseconds.

When the DCP transmits or Receives a DLE-EOT message the trouble flag (D1) and an Indicator Register flag are set for user program interrogation.

d. NAK/NO Response Limit:

The number of times that the TC will attempt to transmit an ENQ or TEXT before taking alternate actions is determined by the value in the NAK/NO Response Limit Register.

When the TC receives a NAK or a time out occurs, the TC will increment the NAK/NO Response Counter and check the new count against the limit register. If the limit has not been reached, the TC will return to the Transmit sequence and attempt to transmit the message again. Upon reaching the NAK/NO Response Limit the Data Comm Processor (DCP) sets the Trouble Flag (D1) and an Indicator Flag for user program interrogation. The Data Comm Processor will then delay retransmission of its message for two seconds. During this transmission delay the DCP is sensitive to the line and can receive a message. This delay is required to permit the unit with the longer time out limit to gain control of the line if it has been sending NAK's due to having its receive buffer loaded and it has a message to send.

The value which is inserted into the NAK/NO Response Limit Register is the hexadecimal representation of the desired decimal value. For example, if it is desired to set the NAK/NO Response Limit to 10, a hexadecimal value of A would be inserted into the register.

e. Demand Disconnect:

A Demand Disconnect sequence is provided to allow a TC to programmatically disconnect the line. When a TC demands a disconnect a DLE-EOT message is automatically transmitted. The trouble Flag (D1) and an Indicator Register Flag is set when a TC transmits or receives a DLE-EOT message.

The disconnect sequence is under the control of the user program and is initiated by setting the Demand Disconnect Register to a value of 1. The Demand Disconnect Register must be set to 0 at all other times. The disconnect sequence can be used by a TC running in the unattended mode to notify the other unit that it has completed transmission and is going to turn itself off.

Additional capabilities of the Demand Disconnect feature will be published later.

f. Idle Line Disconnect:

When the line has been inactive for the length of time specified in the Idle Line Disconnect Register, an Idle Line Timeout is declared and a DLE-EOT message is transmitted by the DCP. The length of time specified in this register can vary from approximately 1 minute to 42 minutes or it can be set to never declare an Idle Line Timeout. However, the minimum time allowed by the DCP is 60 seconds regardless of the time specified in the Register.

The number which is inserted into the Idle Line Disconnect Register to achieve the desired length of time for an Idle Line Timeout is in hexadecimal format and has a weighted value of 10 seconds. The correct value can be determined by dividing the number of seconds desired for an Idle Line Timeout by a factor of 10 and then converting the resulting quotient to its corresponding hexadecimal value.

**EXAMPLE:**

The value to insert in the Idle Line Disconnect Register to declare an Idle Line Timeout after 5 minutes may be computed in the following manner.

1. 5 minutes X 60 = 300 seconds
2. 300 divided by 10 = 30
3. 30 expressed hexadecimally = 1E
4. 1E would be inserted into the register.

If it is desired to never declare an Idle Line Timeout, a value of 00 must be inserted into the Idle Line Disconnect Register.

Additional capabilities of the Idle Line Disconnect Register will be published at a later date.

**g. Line Mode:**

The Line mode register is used by the DCP to determine the line configuration in which it is operating. This register must be set properly to provide the correct timing for the type of line being used.

The values for the two modes of operation are: If 2 wire mode is used, insert a value of 8 in the register; if a 4 wire mode is used, insert a 0 in the register.

**EXAMPLE:**

The control registers could be set programmatically using the parameters listed below:

**PARAMETERS:**

- LINE MODE – 2 wire
- NAK/NO Response Limit – 6
- TIMEOUT LIMIT – 500 Milliseconds
- IDLE LINE DISCONNECT – 5 minutes.

| <u>LABEL</u> | <u>INST</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>              |
|--------------|-------------|----------|----------|----------|-----------------------------|
| CTLREG       | CLA         | 0        | 0        |          | CLEAR ACCUMULATOR           |
|              | SET         | A        | M        |          | SET 2 WIRE MODE             |
|              | INK         | 6        | 6        |          | SET NAK/NO TO SIX           |
|              | INK         | 5        | 3        |          | SET UPPER TIMEOUT           |
|              | INK         | 4        | 2        |          | SET LOWER TIMEOUT = 500 ms  |
|              | INK         | 1        | 1        |          | SET UPPER IDLE LINE         |
|              | INK         | 0        | E        |          | SET LOWER IDLE LINE = 5 min |
|              | LTF         |          |          |          | LOAD CONTROL REGISTERS      |

**2.13.03 INDICATOR REGISTER FLAGS**

Eight flags are provided in the Indicator Register to allow the user program to interrogate the cause of exception conditions which can occur in the Data Comm Processor. The Indicator Register is located in the DCP (word 1197) and is accessed via the RPF and LPF macro instructions.

PT TO PT

RETRIEVE PROBLEM FLAGS

OP CODE

RPF

The RPF instruction transfers the contents of the Indicator Register from the DCP to the Accumulator where the flags can be tested using the Accumulator flag group (A flags).

LOAD PROBLEM FLAGS

OP CODE

LPF

The LPF instruction transfers the contents of the Accumulator to the Indicator Register in the DCP.

The following flags are provided in the Indicator Register. The Flags in Group 1 are located in digit position 15 of the register (word 1197) and the flags in group 2 are located in digit position 14 of the Indicator Register.

Group 1:

| <u>"A" FLAG</u> | <u>Exception Item</u>          |
|-----------------|--------------------------------|
| M               | Received DLE-EOT message       |
| C               | Transmitted DLE-EOT message    |
| S               | Break                          |
| —               | NAK/NO Response limit reached. |

Group 2:

| <u>"A" FLAG</u> | <u>Exception Item</u>       |
|-----------------|-----------------------------|
| M               | Received Buffer overload    |
| C               | Transmitted Buffer overload |
| S               | Parity Error Received       |
| —               | Invalid Character Received  |

**NOTE: Flag Group 2 must be shifted into digit position 15 of the accumulator before testing.**

The Flags in Group 2 are provided mainly as a debugging aid to help in qualifying a data communications network and/or application programs and normally would not be used in a live operating environment.

a. Trouble Flag:

When a condition occurs which causes an Indicator Flag to be set, the Trouble Flag (D1) is also set. The Trouble Flag can only be tested by the user program using skip and execute instructions: It cannot be set or reset. The Trouble Flag is reset by firmware when it finds that the Indicator Flags have been reset by the user program.

b. Program Requirements:

The following steps are recommended in handling exception conditions in order to get a valid test of the Indicator Register Flags and to avoid the possibility of losing an Indicator Flag setting:

1. Test Trouble Flag (D-1): If set, go to Step 2; if reset, continue mainline program.
2. READ Indicator Flag Register to the Accumulator.
3. Test if any Indicator Flags are set.
  1. If set – Go to Step 4.
  2. If reset – Return to mainline program (see note below).
4. Process all Flags set (more than one can be set).
5. Reset Indicator Register.

**NOTE: It is possible under some circumstances for the User Program to retest D-1 before Firmware can reset D-1.**

|          |
|----------|
| PT TO PT |
|----------|

EXAMPLE:

The Indicator Flags could be tested for in the following manner:

| <u>LABEL</u> | <u>INST</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>              |
|--------------|-------------|----------|----------|----------|-----------------------------|
|              | EX          | D        | 1        | 1        | TEST FOR DATA COMM ERROR    |
|              | SRJ         | TSTERR   |          |          | GO TEST ERROR               |
|              | }           | }        |          |          | }                           |
| TSTERR       | RPF         |          |          |          | RETRIEVE INDICATOR FLAGS    |
|              | SK          | A        | -SCM     | 3        | TEST NEW TROUBLE GRP 1 FLAG |
|              | EXZ         | 1        |          |          | TEST NEW TROUBLE GRP 2 FLAG |
|              | SRR         | 1        |          |          | RETURN – NOT NEW TROUBLE    |
|              | BRU         | GP2FLG   |          |          | TROUBLE IN GROUP 2          |
|              | EX          | A        | –        | 1        | TEST NAK/NO LIMIT           |
|              | SRJ         | NAK      |          |          | GO PROCESS ERROR            |
|              | EX          | A        | S        | 1        | TEST BREAK                  |
|              | SRJ         | BREAK    |          |          | GO PROCESS ERROR            |
|              | EX          | A        | C        | 1        | TEST TRANS DLE-EOT          |
|              | SRJ         | TRMEOT   |          |          | GO PROCESS – DISCONNECT     |
|              | EX          | A        | M        | 1        | TEST RECEIVE DLE-EOT        |
|              | SRJ         | RECEOT   |          |          | GO PROCESS – DISCONNECT     |
| GP2FLG       | EXZ         | 1        |          |          | TEST TROUBLE THIS GROUP     |
|              | BRU         | RESET    |          |          | GO RESET INDICATOR FLAGS    |
|              | SLROS       | 1        | 0        |          | POSITION GP2 FLAGS          |
|              | EX          | A        | –        | 1        | TEST STRANGE CHAR           |
|              | SRJ         | STRANG   |          |          | GO PROCESS                  |
|              | EX          | A        | S        | 1        | TEST PARITY ERROR RECV      |
|              | SRJ         | PARITY   |          |          | GO PROCESS                  |
|              | EX          | A        | C        | 1        | TEST TRANS OVERLOAD         |
|              | SRJ         | TROVER   |          |          | GO PROCESS                  |
|              | EX          | A        | M        | 1        | TEST RECV OVERLOAD          |
|              | SRJ         | RCVOVR   |          |          | GO PROCESS                  |
| RESET        | CLA         | 0        | 0        |          | CLEAR FLAGS                 |
|              | LPF         |          |          |          | RESET INDICATOR REGISTER    |
|              | SRR         | 1        |          |          | RETURN TO MAINLINE          |

## **2.14 – CENTRAL TC CONTROLLER PROGRAMING PROCEDURES**

Central TC Controller (CTCC) is a Data Communications Processor (DCP) firmware set which allows a TC to assume the Data Communication I/O functions of a central processing unit in a polling and selecting environment. A TC which utilizes the Central Controller DCP Firmware can control from 1 to 16 remote TC's in an on-line applicational environment.

The Central TC Controller operates in a standard Polling and Selecting line control environment. In addition to standard selection of remote units the following types of special select formats are provided:

Fast Select, Group Select, and Broadcast Select.

The polling or selecting of the various terminals in a network is controlled by a series of 16 control words which are stored in the memory of the Data Communications Processor. These control words can be easily accessed and manipulated as required by macro programing techniques thus giving the user program positive operational control of the network.

In addition to controlling the polling and selecting operations, the line discipline of the Central TC (the term used to describe the TC loaded with the CTCC firmware) can also be controlled. This is possible because the line discipline of a Central TC is not buried in the program codes of the Data Comm Processor. Instead, it is specified and controlled by a collection of Line Procedure Format Registers. A degree of flexibility of line discipline is thus achieved because a change of line discipline does not require a change in the firmware.

The controller will function via a switched, leased or direct connect line configuration.

The following sections discuss in detail: the line disciplines of a Central TC 500 as controlled by the Format Registers; the Data Comm Processor operations of polling and selecting as controlled by the Control Registers; and the Main Memory firmware requirements.

### **2.14.01 LINE DISCIPLINE FORMAT REGISTERS**

Several disciplines are made possible through the use of the Central TC Controller firmware. The line procedures that can be implemented by this new Data Comm firmware are: poll, select, fast select, group select, and broadcast select.

Each line procedure uses two Format Registers; each register consists of one word or eight (8) characters. The most significant character position is called the Data Character Counter (DCC) and is used to specify the number of significant characters contained in the Format Register (this is indicated in digit position 14) along with other information (digit position 15). The seven (7) remaining character positions accommodate the necessary format character which must be right justified. Dummy characters are used as substitute for the address (AD1, AD2, and group address) and the transmission number (TR1, TR2, and TR3). The actual terminal address and transmission number will be fitted in by the Controller firmware during the actual transmission.

CTCC

The dummy characters used in each of the Format Registers are further defined:

|                                 | <u>Character</u> | <u>Dummy Hexadecimal Value</u> |
|---------------------------------|------------------|--------------------------------|
|                                 | AD1              | 80                             |
|                                 | AD2              | 81                             |
|                                 | AD3 (Group)      | 82                             |
| These values must be            | TR1              | 88                             |
| used in a three (3)             | TR2              | 89                             |
| TR # system.                    | TR3              | 8A                             |
| Must be used in a two (2)       | TR1              | 89                             |
| TR # system.                    | TR2              | 8A                             |
| Value in a one (1) TR # system. | TR1              | 8A                             |

All of the actual characters to be transmitted from each of the Format Registers have their normal USASCII format with their parity bits equal to zero (0). Their correct parity bits are generated by hardware as each character goes out on the line.

The succeeding sections specify the formats of the individual pairs of Format Registers used with the various line disciplines supported by the Central TC Controller.

a. Poll Format Registers

These two registers are the Poll Message Register and the Expected Header Register.

The Poll Message Register is located in word 1155 and consists of the actual (and dummy) characters, right justified and in their proper sequence, that are used to poll the slave terminal(s). The Data Character Counter (DCC) in character position eight (8) of the Poll Register contains a value from zero (0) to six (6) depending on the number of characters in the poll message. A poll message one (1) character in length would have a DCC value of zero (0). A poll message seven (7) characters in length would have a DCC value of six (6).

EXAMPLE: Poll Message Register containing the standard TC polling characters.

|                    |     |    |    |     |     |     |     |     |
|--------------------|-----|----|----|-----|-----|-----|-----|-----|
| Character Position | 8   | 7  | 6  | 5   | 4   | 3   | 2   | 1   |
| Word 1155          | 04  | 00 | 00 | 04  | 80  | 81  | 70  | 05  |
|                    | DCC |    |    | EOT | AD1 | AD2 | POL | ENQ |

The Expected Header Register is located in word 1154 and consists of the actual (and dummy) characters, right justified and in the sequence desired are in the header portion of the remote terminals message. A comparison is made using only the first and last character of the actual received header against the first and last character of the expected header. The DCC in character position eight (8) of the Expected Header Register again contains a value from zero (0) to six (6) depending on the number of characters loaded into the register. The BCC is computed, starting with the second significant character in the Expected Header Register.



EXAMPLE: Expected Header Register containing the standard TC header for a three-digit transmission number system.

|                    |     |     |     |     |     |     |     |     |
|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Character Position | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   |
| Word 1154          | 06  | 01  | 80  | 81  | 88  | 89  | 8A  | 02  |
|                    | DCC | SOH | AD1 | AD2 | TR1 | TR2 | TR3 | STX |

b. Select Format Registers

There are two select registers; the Select Message Register and the Header Format Register.

The Select Message Register is located in word 1157 and contains the characters (both actual and dummy) that are used to select the slave terminal(s).

EXAMPLE: Select Message Register with standard TC select characters.

|                    |     |    |    |     |     |     |     |     |
|--------------------|-----|----|----|-----|-----|-----|-----|-----|
| Character Position | 8   | 7  | 6  | 5   | 4   | 3   | 2   | 1   |
| Word 1157          | 04  | 00 | 00 | 04  | 80  | 81  | 71  | 05  |
|                    | DCC |    |    | EOT | AD1 | AD2 | SEL | ENQ |

The Header Format Register is located in word 1156. It contains the characters (actual and dummy) that are in the header portion of the Central TC's message. Depending on the number of characters in the header, character position eight (8) of the Header Register contains one of the following hexadecimal values for the DCC.

| <u>No. of Characters in Header</u> | <u>DCC Value</u> |
|------------------------------------|------------------|
| 1                                  | 08               |
| 2                                  | 09               |
| 3                                  | 0A               |
| 4                                  | 0B               |
| 5                                  | 04               |
| 6                                  | 05               |
| 7                                  | 06               |

The BCC is computed, starting with the second significant character in the Header Register.

EXAMPLE: Header Format Register containing the standard TC header for a no transmission number system.

|                    |     |    |    |    |     |     |     |     |
|--------------------|-----|----|----|----|-----|-----|-----|-----|
| Character Position | 8   | 7  | 6  | 5  | 4   | 3   | 2   | 1   |
| Word 1156          | 0B  | 00 | 00 | 00 | 01  | 80  | 81  | 02  |
|                    | DCC |    |    |    | SOH | AD1 | AD2 | STX |

c. Fast Select (FSL) Format Registers

The characters used in implementing the fast select line discipline are defined as those characters that precede the actual message text. They are further defined as consisting of a

CTCC

first half (all characters up to and including the SOH) and a second half (all characters following the SOH up to and including the STX). Each half of the fast select discipline has a separate format register.

The first half is located in word 1159. Character position eight (8) of word 1159 contains both the Data Character Counter (in digit position 14) and special information (digit position 15) peculiar to halved line discipline formats. Depending on the number of characters in the first half register digit position 14 contains one of the following hexadecimal values for the DCC.

| <u>No. of Characters in First Half</u> | <u>DCC Value</u> |
|--|------------------|
| 1                                      | 8                |
| 2                                      | 9                |
| 3                                      | A                |
| 4                                      | B                |
| 5                                      | 4                |
| 6                                      | 5                |
| 7                                      | 6                |

Digit position 15 contains one of three possible hexadecimal values. A hex 4 indicates there is no second half. In this case, the actual message text is transmitted immediately after the first half. A hex 8 indicates the characters in the first half register are not to be transmitted; proceed to inspect the second half. Hex 0 implies normal (first and second half) fast select.

The BCC computation does not include any of the characters in the first half register.

EXAMPLE: Fast Select Format Register (first half) indicating no second half.

| Character | Position | 8  | 7  | 6  | 5   | 4   | 3   | 2   | 1   |
|-----------|----------|----|----|----|-----|-----|-----|-----|-----|
| Word      | 1159     | 44 | 00 | 00 | 04  | 80  | 81  | 73  | 01  |
|           |          | D  |    |    | EOT | AD1 | AD2 | FSL | SOH |
|           |          | C  |    |    |     |     |     |     |     |
|           |          | C  |    |    |     |     |     |     |     |

The second half of the fast select format is located in word 1158. Again character position eight (8) contains both the DCC (digit position 14) and special information (digit position 15). The possible hexadecimal values for the DCC are the same as those outlined for the first half register. Digit position 15 of the second half register contains one of four possible values.

- Hexadecimal 0 — Implies normal mode.
- Hexadecimal 2 — Indicates the first half register contains four (4) characters or less.
- Hexadecimal 4 — Indicates there is no first half.
- Hexadecimal 8 — The characters in this register are not to be transmitted; proceed to actual message text.

The BCC is computed, starting with the first significant character in the second half register.

EXAMPLE: Fast Select Format Register (second half) indicating first half contained 4 characters or less.

|           |          |    |    |     |     |     |     |     |     |
|-----------|----------|----|----|-----|-----|-----|-----|-----|-----|
| Character | Position | 8  | 7  | 6   | 5   | 4   | 3   | 2   | 1   |
| Word      | 1158     | 25 | 00 | 80  | 81  | 88  | 89  | 8A  | 02  |
|           |          | D  |    | AD1 | AD2 | TR1 | TR2 | TR3 | STX |
|           |          | C  |    |     |     |     |     |     |     |
|           |          | C  |    |     |     |     |     |     |     |

d. Group Select (GSL) Format Registers

The GSL Format Registers also specify a first half and a second half. The first half is located in word 1161, the second half in word 1160. Their structures are identical to those of the first and second halves respectively, of the Fast Select Format Registers.

Broadcast (BSL)

The Broadcast Format Registers again specify a first half (located in word 1163) and a second half (located in word 1162). Their structures are also identical to those of the first and second halves, respectively, of the Fast Select Format Registers.

e. Summary

When the Central TC Controller firmware is first loaded into the machine, all format registers become initialized to their corresponding standard (3 transmission numbers) TC line disciplines. These disciplines can be changed to meet most non-Burroughs standards by altering the contents of the appropriate Format Register(s).

However, in spite of this scheme to seek flexibility, certain basic structures of line disciplines have to be adhered to. Refer to charts 1, 2 and 3 at the end of this subject for illustrations of the basic structures for polls, selects, fast selects, group selects, and broadcast selects.

**2.14.02 DATA COMM PROCESSOR OPERATIONS**

The operation of the Data Comm Processor of a Central TC is dictated by the contents of sixteen (16) Control Registers. Since each terminal connected to a Central TC requires the use of only one (1) Control Register, the CTCC firmware can handle up to sixteen (16) terminals at any one time.

These registers occupy memory words 1184-1199 in the DCP memory. Each register is one (1) word in length and contains:

1. The address (AD1, AD2, and group address) of its associated terminal. This information is contained in character positions 8, 7 and 6 respectively.
2. The beginning transmission number of the outgoing message to this terminal in a three (3) transmission number system, character positions 5, 4 and 3 of the Control Register are used for the TR numbers. In a two (2) TR number system, character positions 4 and 3 are used. A one (1) TR number system uses character position 3. In a zero TR number system, character positions 5, 4 and 3 must be cleared.

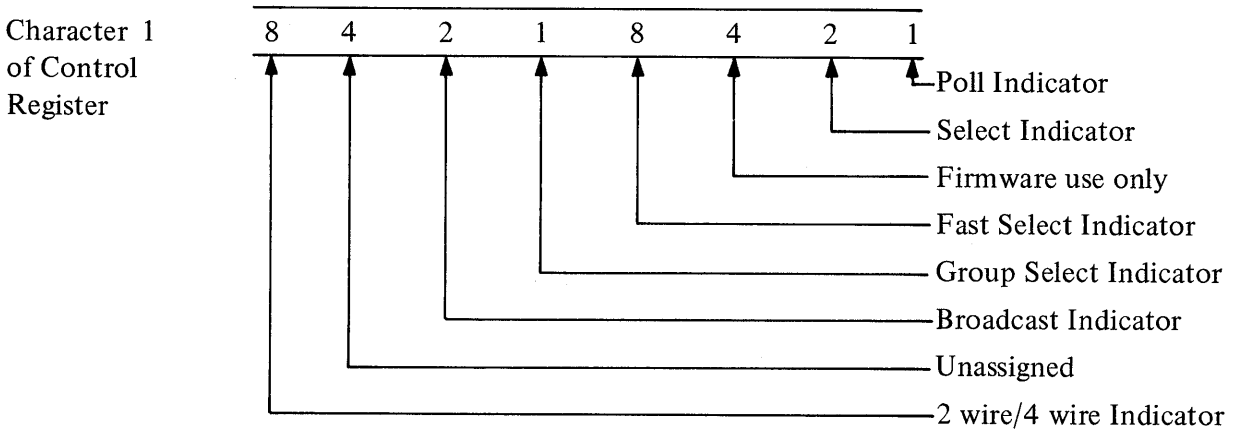
CTCC

3. Operation Indicators to service this terminal. These are located in character position 1.

The Control Registers are placed in memory in the form of a list. The Data Comm Processor will process this list of sixteen (16) Control Registers one at a time, in sequence, beginning at the top. It will perform the function(s) indicated by the Operation Indicator(s) contained within the Control Register. Thus, the terminals will be serviced in the sequence in which their corresponding Control Register is placed. When the 16th Control Register is processed, operation will return to the top of the list. Should less than sixteen terminals be connected to a Central TC, and AD1 hexadecimal value of 00 in the first un-used Control Register causes the Data Comm Processor to return to the register at the top of the list. Any column 0 code from the USASCII chart (except 00) or any column 1 code used in place of AD1, causes the current register to be skipped. Operation then proceeds to the next Control Register in the list.

a. Operation Indication

As mentioned, Operation Indicators occupy the least significant character position of a Control Register. Their individual bit allocations are shown:



If any of the above operations result in no response, strange response, or inability to transmit a message, due to some condition at the remote terminal, the Data Comm Processor times out and goes into an idle state. A special flag (D1) is set and the exact cause of the time out is contained in a special Time Out Register. This register is available to the macroprogram. (See section on Data Comm Processor Time Out.)

1. Poll Indicator

The Poll Indicator is normally reset. To poll a terminal, the Main Processor sets the Poll Indicator of a Control Register, as specified by the macroprogram. The input buffer of the Data Comm Processor should be empty and D2 should be reset. The Processor will not poll any terminal unless D2 is reset. After a successful poll, the Poll Indicator will be reset, and the Message Received Flag (D2) set. A special register (called the Header Register) containing the received message header, right justified, is available. This allows the macroprogram to retrieve the address of the terminal from which the message came, and the transmission number of the message received. By numbering the terminals sequentially, and organizing the Control Register list in the same manner, the address in the Header Register serves as a pointer to its corresponding Control Register. Upon completion of a poll procedure, the Data Comm Processor will time out and assume the idle state. The other bits in

the Operation Indicators will be interrogated only when the macro programmer releases the processor from its idle state. This is accomplished through the use of the RESUME command.

2. Select Indicator

This indicator is normally reset. To transmit a message to a terminal, the macroprogram must transfer the message to the output buffer, set the Transmit Ready Flag, and set the Select Indicator in the appropriate Control Register. The Data Comm Processor then selects this terminal when its Control Register is processed. The Select Indicator is reset by firmware after a successful Select.

3. Fast Select Indicator

This indicator is normally reset. To transmit a message to a terminal via Fast Select, the macroprogram must set up the output message, set the Transmit Ready Flag, and set the Fast Select Indicator in the appropriate Control Register. The Data Comm Processor then Fast Selects this terminal when its Control Register is processed. The FSL Indicator is reset by the CTCC after a successful Fast Select.

4. Group Select and Broadcast Indicators

Both of these indicators perform their respective functions in the identical manner of the Fast Select Indicator.

5. 2 Wire/4 Wire Indicator

This indicator must be set by the macroprogram for a 2 wire system. It must be reset (0) for a 4 wire system.

The following example illustrates the initial format of the Control Register in a three (3) terminal, 4 wire, network using a two-digit TR number.

|           |          |  |       |     |     |     |     |     |     |                |
|-----------|----------|--|-------|-----|-----|-----|-----|-----|-----|----------------|
| Character | Position |  | 8     | 7   | 6   | 5   | 4   | 3   | 2   | 1              |
| Word      | 1184     |  | /31   | /41 | /31 | /00 | /30 | /30 | /00 | /01/           |
|           |          |  | AD1   | AD2 | GSL |     | TR1 | TR2 |     | POL            |
|           |          |  |       |     |     |     |     |     |     |                |
| Character | Position |  | 8     | 7   | 6   | 5   | 4   | 3   | 2   | 1              |
| Word      | 1185     |  | /31   | /42 | /31 | /00 | /30 | /30 | /00 | /01/           |
|           |          |  | AD1   | AD2 | GSL |     | TR1 | TR2 |     | POL            |
|           |          |  |       |     |     |     |     |     |     |                |
| Character | Position |  | 8     | 7   | 6   | 5   | 4   | 3   | 2   | 1              |
| Word      | 1186     |  | /31   | /43 | /31 | /00 | /30 | /30 | /00 | /03/           |
|           |          |  | AD1   | AD2 | GSL |     | TR1 | TR2 |     | POL and<br>SEL |
|           |          |  |       |     |     |     |     |     |     |                |
| Character | Position |  | 8     | 7   | 6   | 5   | 4   | 3   | 2   | 1              |
| Word      | 1187     |  | /00   | /00 | /00 | /00 | /00 | /00 | /00 | /00/           |
|           |          |  | * AD1 | AD2 | GSL |     |     |     |     |                |

\* The AD1 hexadecimal value of 00 causes the Data Comm Processor to return to word 1184.

CTCC

b. Data Comm Flags

Three Data Comm flags are defined to serve as communications between the Data Comm Processor and the Main Memory Firmware:

- D1 – This flag is set by the Data Comm Processor whenever it goes into an idle state. An idle state occurs when either the Data Comm Processor times out or the macroprogram issues an Idle Request. (See section on macro instructions under MAIN MEMORY). D1 is reset when the macroprogram re-activates the Data Comm Processor to bring it out of the idle state.
- D2 – Message Received Flag
- D3 – Transmit Ready Flag. It is set by the macroprogram to indicate that the output buffer contains a message ready for transmission. However, this message will be transmitted to a terminal only if the Select Indicator in the proper Control Register is also set.

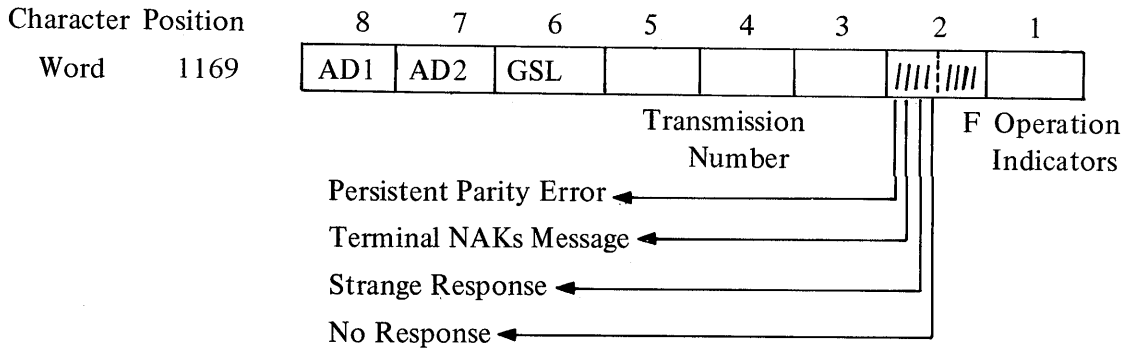
All three flags are available for interrogation through the regular Skip/Execute instructions.

c. Data Processor Time Out

The following situations cause the Data Comm Processor to time out:

1. The Central TC receives no response from a terminal to any of the following: poll, select, fast select, group select, or broadcast.
2. The Central TC receives a strange response from a terminal to any of the following: poll, select, fast select, group select, or broadcast.
3. Terminal NAKs a select, fast select, group select or a broadcast.
4. Persistent parity error occurs between the Central TC and the terminal.

As previously discussed, while the Data Comm Processor is in the time out condition, D1 is set and a register is available for interrogation. This register, called the Time Out Register, is located in word 1169 and is a replica of the Control Register that is involved at the time, supplemented by information stored in character position 2 as shown:



- F = 0000 POL
- F = 0001 SEL
- F = 0010 FSL
- F = 0011 GSL
- F = 0100 BSL

When the Data Comm Processor times out and goes into an idle state due to one of the above conditions, the macroprogram must retrieve the Time Out Register and, after examining its contents, clear it. This must be done prior to re-initiating the Data Comm Processor to its normal operation. Re-initiating the Processor also resets the D1 flag.

The Time Out Register serves no purpose if the idle state of the Data Processor is initiated by the macroprogram as there is no indicator in the register to reflect such a condition.

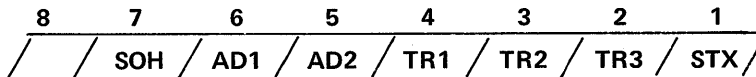
d. Header Register

If a remote TC makes an affirmative response to a Poll Enquiry (data) the Header portion of the remote TC's message is stored in the Header Register (word 1166) for use by the macroprogrammer.

If a sequential numeric addressing scheme is utilized, the macroprogrammer can examine the contents of the Header Register to determine which remote is responding to the POLL and reload the appropriate control register. The data in the Header Register is right justified and contains all of the header information up to and including the STX character.

EXAMPLE:

Character Position



**2.14.03 MAIN MEMORY PROCESSOR**

Main Memory can not access Data Comm firmware unless the latter is in an idle state. However, the Main Processor can cause an idle condition by issuing an idle request. A special macroinstruction, IDLE REQUEST, is implemented to perform this function.

The operation of the IDLE REQUEST instruction involves the setting of the D1 flag. The Data Comm Processor interrogates D1 at certain convenient points during its regular operation. Should the flag be set, any Data Comm procedure previously initiated is allowed to terminate before the Processor goes into the idle state. When the Processor is re-initiated, D1 is reset.

A situation can arise where the Data Comm Processor encounters one of the previously discussed conditions that cause a time out after the Main Processor initiates an idle request and before the Data Comm Processor actually goes into idle. Since the Main Processor was first in initiating the setting of D1, the Time Out Register is left unchanged if no error conditions are encountered. If the Time Out Register has been cleared everytime it was interrogated, the fact that the register is zero (0) is adequate indication to the macroprogram that the idle state is due to the request and not to any error condition. However, since the idle request does leave the register unchanged, it must be cleared everytime so that it will always reflect to the macroprogram the correct cause of the time out (i.e., error condition or idle request).

|      |
|------|
| CTCC |
|------|

The Central TC Controller firmware will operate with any of the standard GP 300, Data Comm main memory firmware sets that are supplemented by the CTCC main memory add-on tape. This add-on implements special macroinstructions described below:

a. Resume

| <u>OP CODE</u> | <u>LABEL</u> |
|----------------|--------------|
| CODE           | 1000         |

This command re-initiates the Data Comm Processor's normal operation. It should only be given when the Processor is in an idle state. If the DCP processor is not in an idle state, the machine will hang on the instruction.

b. Idle Request

| <u>OP CODE</u> | <u>LABEL</u> |
|----------------|--------------|
| CODE           | 1100         |

This command allows the macroprogram to interrupt the normal operation of the Data Comm Processor and cause an idle state.

c. Retrieve Header Register

| <u>OP CODE</u> | <u>LABEL</u> |
|----------------|--------------|
| CODE           | 3C8E         |

The actual received header is placed into the Accumulator.

d. Retrieve Time Out Register to Accumulator

| <u>OP CODE</u> | <u>LABEL</u> |
|----------------|--------------|
| CODE           | 3C91         |

e. Load Time Out Register from Accumulator

| <u>OP CODE</u> | <u>LABEL</u> |
|----------------|--------------|
| CODE           | 3491         |

f. Retrieve Control Register

| <u>OP CODE</u> | <u>LABEL</u> | <u>A</u> |
|----------------|--------------|----------|
| CODE           | 3CA          | O-F      |

The A-field specifies which of the sixteen (16) Control Registers is to be placed into the Accumulator.

g. Load Control Register

| <u>OP CODE</u> | <u>LABEL</u> | <u>A</u> |
|----------------|--------------|----------|
| CODE           | 34A          | O-F      |



The A-field specifies which of the sixteen (16) Control Registers to load the contents of the Accumulator into.

Example of controlling one terminal in a normal TC Polling and Selecting environment.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>                  |
|--------------|----------------|----------|----------|----------|---------------------------------|
|              | LIR            | 2        | 0        |          |                                 |
|              | CLM            | ADDR     |          |          |                                 |
|              | LKBR           | ADDR     |          |          |                                 |
|              | TKM            | 2        |          |          | ENTER REMOTE'S ADDR             |
|              | TRA            | ADDR     |          |          | LOAD INTO ACCUM                 |
|              | INK            | 0        | 1        |          | LOAD POL OP-INDICATOR           |
|              | CODE           | 1100     |          |          | IDLE REQUEST                    |
|              | CODE           | 34A0     |          |          | LOAD 0 CONTROL REGISTER         |
|              | CLA            | 0        | 0        |          |                                 |
|              | CODE           | 34A1     |          |          | LOAD 1 CONTROL REGISTER         |
|              | CODE           | 1000     |          |          | RESUME                          |
| LISTEN       | EX             | D        | 1        | 1        |                                 |
|              | SRJ            | ERROR    |          |          |                                 |
|              | SK             | B        | 3        | 1        |                                 |
|              | BRU            | SELECT   |          |          |                                 |
|              | EX             | D        | 2        | 1        |                                 |
|              | BRU            | MSGE     |          |          |                                 |
|              | BRU            | LISTEN   |          |          |                                 |
| ERROR        | IIR            | 2        | 10       |          | KEEP ERROR COUNT                |
|              | CODE           | 3C91     |          |          | RETRIEVE ERROR REG              |
|              | SKL            | 3        | 8        | 4        | TEST FOR PARITY                 |
|              | AL             | 1        |          |          |                                 |
|              | POS            | 50       |          |          |                                 |
|              | PA             | PARMSG   |          |          |                                 |
|              | LIR            | 2        | 0        |          | RESET COUNTER                   |
|              | SK             | T        | 1        | 4        |                                 |
|              | CLA            | 0        | 0        |          | CLEAR TIME OUT REG              |
|              | CODE           | 3491     |          |          | RELOAD TIME-OUT REG             |
|              | CODE           | 1000     |          |          | RESUME                          |
|              | SRR            | 1        |          |          |                                 |
|              | LIR            | 2        | 0        |          | RESET COUNTER                   |
|              | AL             | 1        |          |          |                                 |
|              | POS            | 50       |          |          |                                 |
|              | EXL            | 3        | 2        | 2        | CHECK FOR NO RESPONSE INDICATOR |
|              | PA             | NOMSG    |          |          |                                 |
|              | BRU            | RTNE     |          |          |                                 |
|              | EXL            | 3        | 3        | 2        | CHECK FOR A STRANGE RESPONSE    |
|              | PA             | STRMSG   |          |          |                                 |
|              | BRU            | RTNE     |          |          |                                 |
|              | EXL            | 3        | 4        | 2        | CHECK FOR INVALID               |

|      |
|------|
| CTCC |
|------|

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>          |
|--------------|----------------|----------|----------|----------|-------------------------|
|              | PA             | INDIC    |          |          | DIGIT IN INDICATOR      |
|              | BRU            | RTNE     |          |          |                         |
|              | EXL            | 3        | 5        | 2        |                         |
|              | PA             | NAKMSG   |          |          | CHECK FOR NAK           |
|              | BRU            | RTNE     |          |          | LIMIT INDICATOR         |
|              | SKL            | 3        | 5        | 1        | CHECK FOR INVALID       |
|              | PA             | INDIC    |          |          | DIGIT IN INDICATOR      |
| RTNE         | CLA            | 0        | 0        |          | CLEAR REGISTER          |
|              | CODE           | 3491     |          |          | LOAD TIME OUT REG       |
|              | CODE           | 1000     |          |          | RESUME                  |
|              | SRR            | 1        |          |          |                         |
| SELECT       | AL             | 2        |          |          |                         |
|              | POS            | 10       |          |          |                         |
|              | LKBR           | SEND     |          |          |                         |
|              | TKM            | 150      |          |          |                         |
|              | EX             | D        | 3        | 3        |                         |
|              | EX             | D        | 1        | 1        |                         |
|              | SRJ            | ERROR    |          |          |                         |
|              | BRU            | -3       |          |          |                         |
|              | CODE           | 1100     |          |          | IDLE REQUEST            |
|              | CODE           | 3CA0     |          |          |                         |
|              | INK            | 0        | 3        |          | SET POL-SEL INDICATOR   |
|              | CODE           | 34A0     |          |          | LOAD CONTROL REG 0      |
|              | TSB            | SEND     |          |          |                         |
|              | CODE           | 1000     |          |          | RESUME                  |
|              | SET            | R        | 3        |          |                         |
|              | BRU            | LISTEN   |          |          |                         |
| MSGE         | CODE           | 1100     |          |          | IDLE REQUEST            |
|              | TRB            | RECEIV   |          |          | TRANSFER TO RECORD AREA |
|              | CODE           | 3CA0     |          |          | RETRIEVE CONTROL REG    |
|              | INK            | 0        | 1        |          | SET POL INDICATOR       |
|              | CODE           | 34A0     |          |          | LOAD CONTROL REGISTER   |
|              | CODE           | 1000     |          |          | RESUME                  |
|              | RST            | R        | 2        |          |                         |
|              | LRBR           | RECEIV   |          |          | PRINT                   |
|              | AL             | 2        |          |          | MESSAGE                 |
|              | POS            | 10       |          |          | ROUTINE                 |
|              | PR             |          |          |          |                         |
|              | PAB            | 130      |          |          |                         |
|              | SK             | T        | 0        | 2        |                         |
|              | AL             | 1        |          |          |                         |
|              | BRU            | -5       |          |          |                         |
|              | BRU            | LISTEN   |          |          |                         |

CTCC

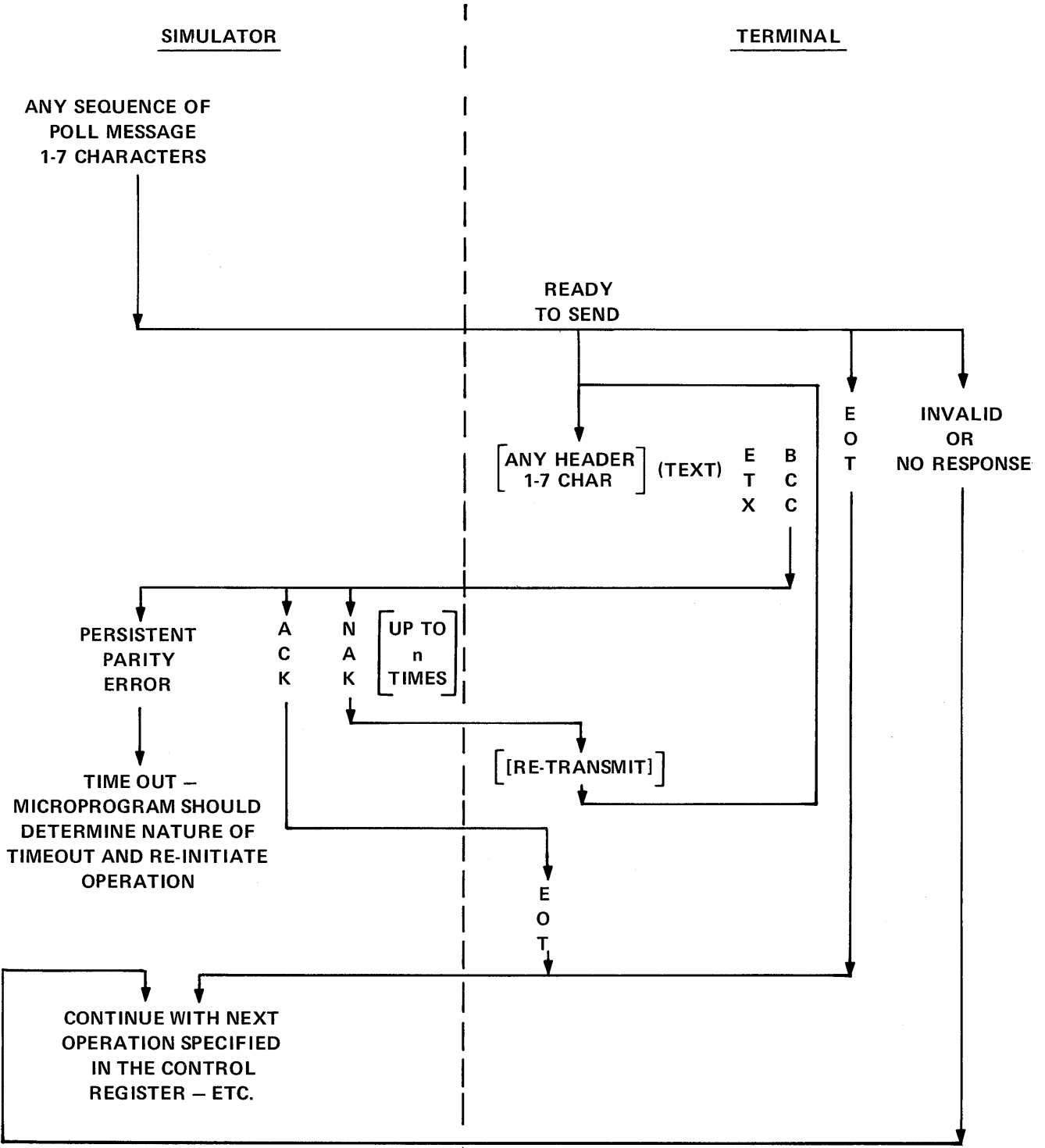


Chart 1. Poll

CTCC

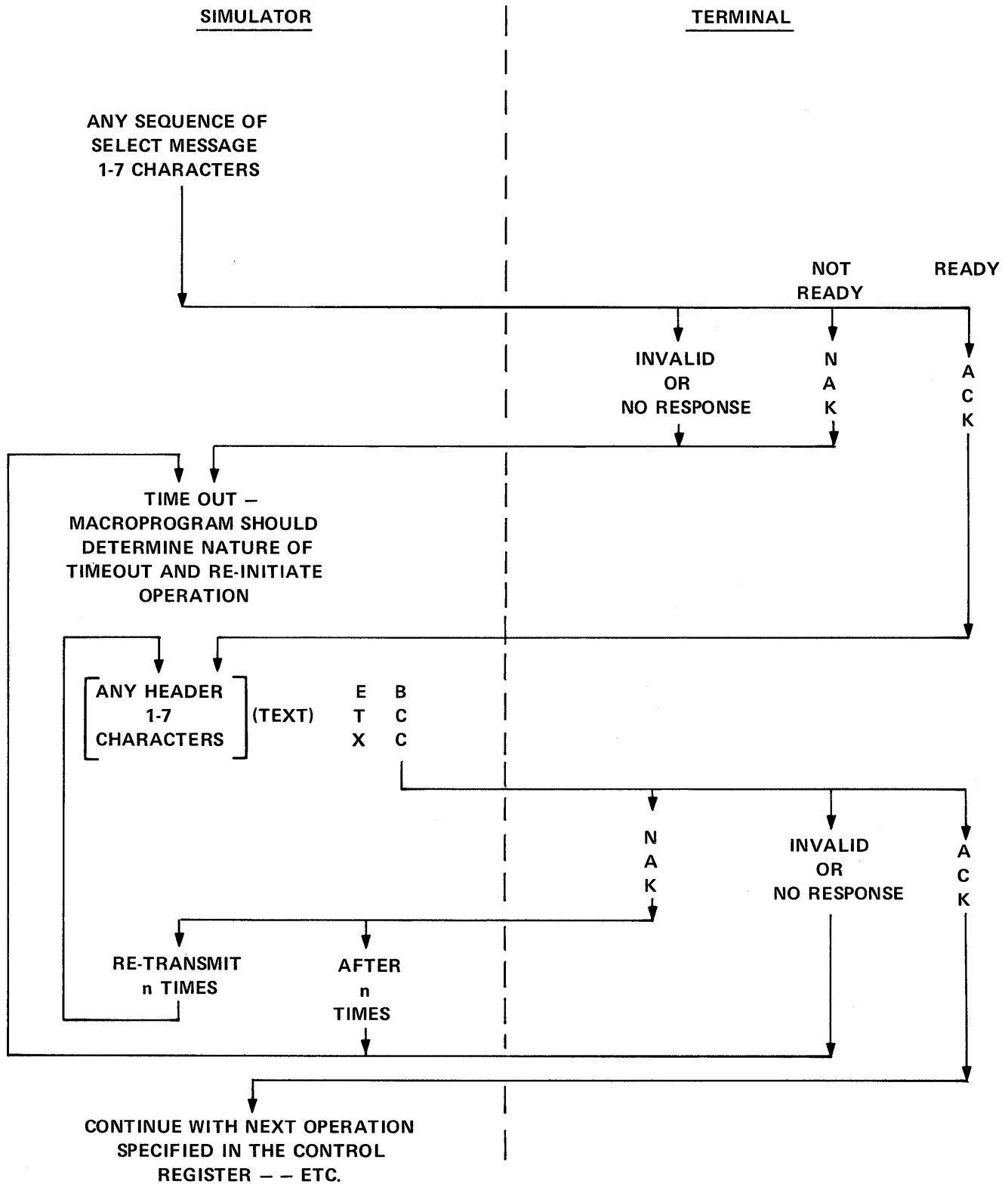


Chart 2. Select

CTCC

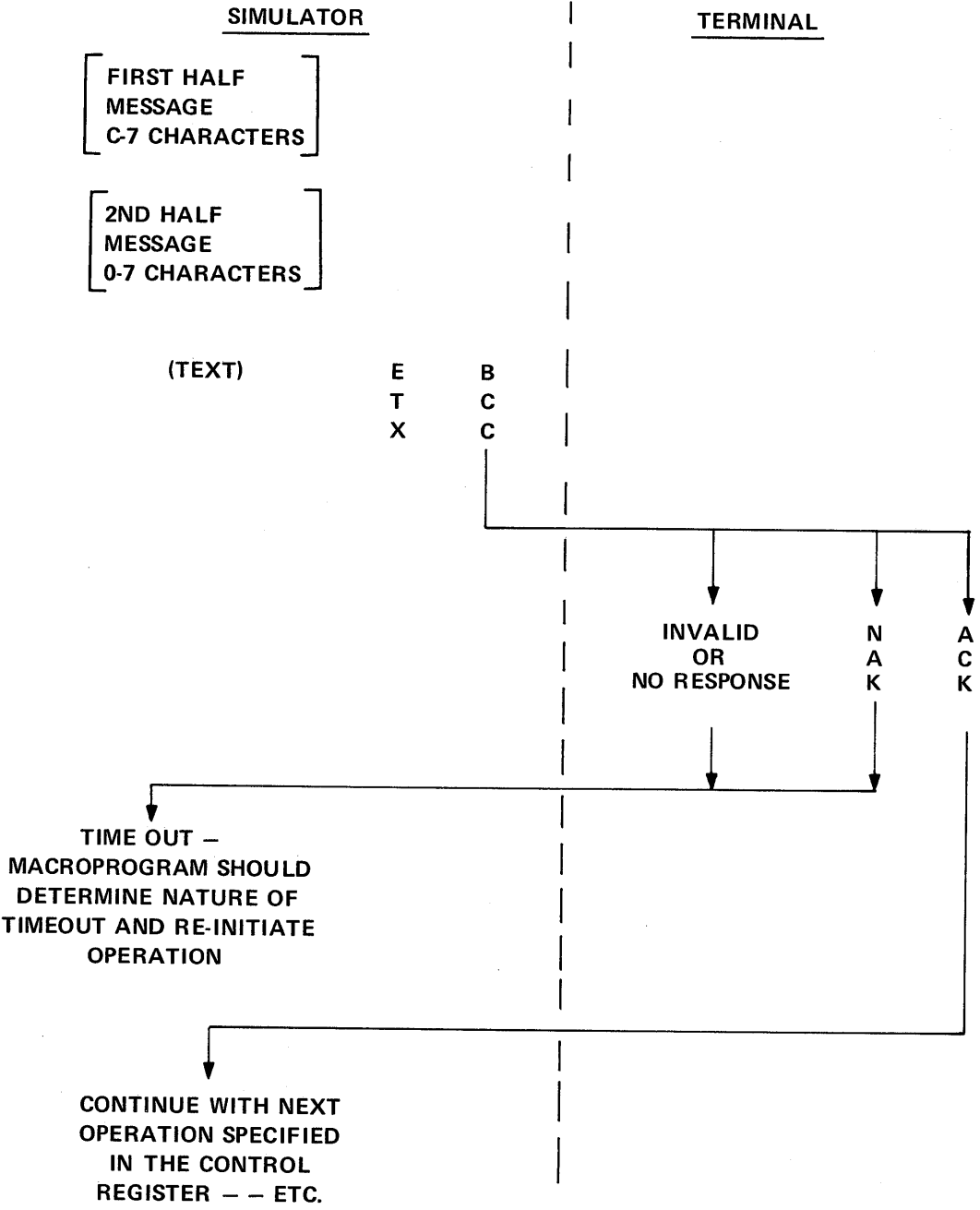


Chart 3. Fast Select, Group Select, and Broadcast Select

## 2.15 – INPUT WITH PUNCHED PAPER TAPE/EDGE PUNCHED CARD READER

Instructions are provided to read punched paper tape or edge punched cards, using a Burroughs Style A 581 Paper Tape/Edge Card Reader as the input adjunct. All subsequent reference to “paper tape” applies both to punched paper tape and to edge punched cards, unless indicated otherwise.

Tape reading is serial, one character at a time, at a speed up to 40 characters per second (when no printing accompanies it). When reading paper tape and printing, the reading speed is up to 20 characters per second; when reading and punching only (no printing), reading speed is up to 40 cps.

The Series L/TC internal character code is USASCII; however, any 5, 6, 7, 8 channel paper tape code can be read and interpreted by utilizing a Table of Input Code Assignments for conversion of the paper tape code into the internal USASCII code. The functional codes in a code set may be used as field identifier codes to terminate tape reading and set flag patterns, or may be ignored (refer to the Table of Input Assignments in Appendix I). The scheme of character parity checking for a particular code set is also a function of the Table of Code Assignments. Firmware for 5 channel code is different than that for 6, 7, or 8 channel “table look-up” firmware or for USASCII No Table firmware.

### 2.15.01 PAPER TAPE READER INSTRUCTIONS

The Paper Tape Reader instructions are designed to function both as “read” instructions and as “keyboard” instructions.

When all tape reading conditions exist, i.e., the reader is on, the photo-electric light is on, and media is present, reading of the paper tape will occur according to the specifications of the instruction.

If any of the above conditions do not exist, then the reader is not operable (a “reader condition” has occurred). The read instruction now reverts to its keyboard counterpart\*\*, and the keyboard buffer is cleared so that the operator may manually index that data required by the altered read instruction. Note that any data resident in the keyboard buffer is lost when the read instruction fails to execute. It follows that the read instruction must be reached before a manual entry is made in its place, because if the operator anticipates this condition and indexes data before the program halts, the data will be lost.

The mnemonic representations of the read instructions are the same as selected keyboard instructions with the addition of a prefix letter “R.”

Instructions that involve punching paper tape along with reading of paper tape will inhibit the punch part of the instruction if the tape perforator is turned off. In addition, the Punch Off Indicator light is turned on and Punch Off Flag is set (refer to Subject 2.16.02).

\*\* EXCEPTION: RNK reverts to a NKRCM (see Subject 2.02.01).

|                     |    |
|---------------------|----|
| RTK<br>RTKM<br>REAM | PT |
|---------------------|----|

**2.15.02 PAPER TAPE/EDGE PUNCHED CARD INPUT INSTRUCTIONS**

|                      | <u>OP CODE</u> | <u>A</u>                 |
|----------------------|----------------|--------------------------|
| READ ALPHA AND PRINT | RTK            | 0-150 15½” forms handler |
|                      | RTK            | 0-255 26” forms handler  |

The RTK instruction reads from tape (i.e., paper tape or edge punched card) and prints the number of alphanumeric characters specified by the “A” field. The instruction will be terminated upon reading a field identifier code or after reading the number of alphanumeric characters as denoted by the “A” parameter.

The flag patterns to be set by the field identifier codes are determined by the Table of Input Code Assignments (see Appendix I).

When a “reader condition” exists, the RTK instruction reverts to a TK instruction and the keyboard buffer is CLEARED in anticipation of manual input.

|                                  | <u>OP CODE</u> | <u>A</u>                 |
|----------------------------------|----------------|--------------------------|
| READ ALPHA INTO MEMORY AND PRINT | RTKM           | 0-150 15½” forms handler |
|                                  | RTKM           | 0-255 26” forms handler  |

The RTKM instruction reads from tape into memory and prints the number of alphanumeric characters specified by the “A” field. The RTKM should be preceded by an LKBR instruction to indicate the starting word location in memory for character storage. (See Subject 2.02.03.)

The LKBR is incremented to the next higher word after each eight characters have been read. The instruction will be terminated upon reading a field identifier code or completion of reading the number of alphanumeric characters specified in the “A” field. The flag patterns to be set by the field identifier codes are determined by the table of input code assignments. (See Appendix I).

If a reader condition exists, the RTKM instruction will revert to a TKM instruction. (See RTK instruction).

|                                   | <u>OP CODE</u> | <u>A</u>                 |
|-----------------------------------|----------------|--------------------------|
| READ ALPHA INTO MEMORY, NON-PRINT | REAM           | 0-150 15½” forms handler |
|                                   | REAM           | 0-255 26” forms handler  |

The REAM instruction reads from tape into memory the number of alphanumeric characters specified in the “A” parameter; no printing occurs. The REAM instruction should be preceded by an LKBR instruction to denote the starting word location in memory for character storage. The LKBR is incremented to the next higher order word after each set of eight characters has been read. The instruction will be terminated upon reading a field identifier code or completion of reading the number of alphanumeric characters specified in the “A” field. The flag patterns to be set by the field identifier codes are determined by the Table of Input Code Assignments.

|       |      |
|-------|------|
| RXEAM | RXTK |
| RXTKM | RNK  |
|       | PT   |

If a reader condition exists, the REAM instruction reverts to an EAM instruction. (See RTK instruction).

|   | <u>OP CODE</u> | <u>A</u>  |
|---|----------------|---|
| READ ALPHA INTO MEMORY AND PUNCH, NON-PRINT | RXEAM          | 0-150 15½" forms handler<br>0-255 26" forms handler |

The RXEAM instruction is the same as the REAM instruction, except that punching will also occur.

The RXEAM instruction can revert to an XEAM instruction if the tape reader is not operable, to an REAM instruction if the tape perforator is turned off, or to an EAM instruction if neither the reader nor the perforator is operable.

|                             | <u>OP CODE</u> | <u>A</u>                 |
|-----------------------------|----------------|--------------------------|
| READ ALPHA, PRINT AND PUNCH | RXTK           | 0-150 15½" forms handler |
|                             | RXTK           | 0-255 26" forms handler  |

The RXTK instruction reads from tape, and simultaneously prints and punches the number of characters specified in the A parameter. The instruction is terminated after reading the specified number of characters or upon reading a field identifier code.

The flag patterns to be set by the field identifier codes are determined by the Table of Input Assignments. (See table in Appendix I).

The RXTK instruction can revert to an XTK instruction if the tape reader is not operable. If the paper tape punch is off, the RXTK will revert to a RTK instruction; or to a TK instruction if both a reader and perforator condition exist. (See RTK instruction).

|   | <u>OP CODE</u> | <u>A</u>                 |
|---|----------------|--------------------------|
| READ ALPHA INTO MEMORY, PRINT AND PUNCH | RXTKM          | 0-150 15½" forms handler |
|   | RXTKM          | 0-255 26" forms handler  |

The RXTKM instruction is the same as the RTKM instruction, except that tape punching occurs simultaneously.

The RXTKM instruction can revert to an XTKM instruction if the tape reader is not operable. If a perforator condition exists, the RXTKM will revert to a RTKM instruction; or to a TKM instruction if both a reader and perforator condition exist.

|                               | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-------------------------------|----------------|----------|----------|
| READ NUMERIC INTO ACCUMULATOR | RNK            | 0-15     | 0-15     |

The RNK instruction reads from the tape into the Accumulator the total number of characters specified by the sum (maximum of 15) of the A and B parameters. The instruction is terminated after the total number of characters specified have been read (fixed field) or upon reading a field identifier code (variable fields). The paper tape characters enter the Accumulator as digits, from low to high order digit positions. NOTE: No printing occurs.



A number may be read into the Accumulator as either a fixed field or a variable field.

With a fixed field, the tape must contain as many codes as the total number of digits required by the instruction. This may require that preceding zeros be included in the tape in order to obtain the fixed field size. Because the codes enter the low order position, reading a decimal number into the Accumulator requires that the maximum number of decimal places to the right of the decimal point be filled with digits or zeros. Note that the separation of the fields into whole and decimal digits is provided to permit keyboard flexibility when a reader condition occurs (see use of NK, Subject 2.02.01).

Example 1: Read 12.25 into the Accumulator, allow for 3 decimal places, fixed field of 9.

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| RNK            | 6        | 3        |

Tape must contain: 000012250 (no field I.D. code)

Manual entry must be: 12250 (left to right)

Manual entry format: 1, 2, decimal, 2, 5, and 0

Variable fields eliminate the “preceding zeros” requirement of fixed fields. Instead, a “field identifier code” immediately follows the number in the tape causing termination of the RNK. With variable fields, the A parameter must be 1 greater than the maximum digits allowed for that quantity so that the field identifier code may be read.

Example 2: Read 12.25 into the Accumulator, allow for 3 decimal places with maximum of 9 digits.

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| RNK            | 6+FS = 7 | 3        |

Tape contains 12250 FS (FS denotes field I.D. code)

Example 3: Read 4000 into the Accumulator. Maximum of 4 digits.

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| RNK            | 5        | 0        |

Tape contains 4000FS

| <u>RELEASE MEDIA CLAMP</u> | <u>REL</u> |
|----------------------------|------------|
| RELEASE MEDIA CLAMP        | REL        |

The REL instruction will cause the media clamp for paper tape or edge punched cards to open, thus halting any further reading until the operator places new material in the reader.

This instruction is useful when using edge punched cards, to release the card after necessary information has been read, and to prevent any additional information on the card from enabling the read instruction for the next entry.

**2.16 – OUTPUT WITH PAPER TAPE/EDGE PUNCHED CARD PERFORATOR**

The instructions described in this section provide the means to output data into punched paper tape and/or edge punched cards by using a Style A 562 Paper Tape/Edge Punched Card Perforator as the output adjunct. All subsequent reference to “paper tape” applies both to punched paper tape and to edge punched cards, unless indicated otherwise.

Tape punching is serial at a speed up to 40 characters per second when no printing accompanies it. When printing accompanies punching paper tape, the punching speed is up to 20 characters per second.

The Series L/TC internal character code is USASCII and output to paper tape will normally be in this code. However, any 5, 6, 7, or 8 channel paper tape code can be punched by utilizing a Table of Output Code Assignments for conversion of the internal code into a different paper tape code (refer to Appendix I). The firmware for 5 channel code is different than that for 6, 7, or 8 channel “table look-up” firmware or for USASCII No Table firmware.

The Paper Tape Punch Instructions provide the ability to print and punch data from the Accumulator, print and punch alphanumeric data from memory, and to type or type into memory while punching. In addition, a register is provided which counts the number of codes punched. This enables the use of continuous edge punched cards by making it possible to determine when one continuous card has been filled or when to fill any unused portion of a continuous card with feed codes before aligning the next continuous card to the first sprocket hole.

The Paper Tape Punch Instructions are designed to function in three ways:

1. When proper tape punching conditions exist, punching will occur according to the specifications of the instruction.
2. If the perforator is not connected or is turned off, the punch portion of the instruction is inhibited and the instruction is executed in accordance with its counterpart keyboard or print instruction. Thus, although the program may provide for punching, the perforator may be turned off or discontinued without affecting the operation of the rest of the system.
3. If the perforator is turned on but does not have media loaded, execution of the punch instruction is held up until the condition is corrected.

The mnemonic representations of the punch instructions are the same as selected keyboard and print instructions with the addition of a prefix letter “X.”

**2.16.01 PAPER TAPE/EDGE PUNCHED CARD OUTPUT INSTRUCTIONS**

| TYPE, PUNCH | <u>OP CODE</u> | <u>A</u>                 |
|-------------|----------------|--------------------------|
|             | XTK            | 0-150 15½” forms handler |
|             | XTK            | 0-255 26” forms handler  |

The XTK instruction allows typing, printing and punching up to the number of characters specified in the A field. The instruction functions like a TK instruction except that punching occurs with it. The termination of this instruction with an OCK or PK does not cause a code to punch.

|      |     |    |
|------|-----|----|
| XTKM | XPA |    |
| XEAM | XA  | PT |

If the perforator is turned off or disconnected, the XTK instruction will operate only as a TK instruction.

|                                   | <u>OP CODE</u> | <u>A</u>                 |
|-----------------------------------|----------------|--------------------------|
| TYPE INTO MEMORY, PUNCH AND PRINT | XTKM           | 0-150 15½" forms handler |
|                                   | XTKM           | 0-255 26" forms handler  |

The XTKM instruction allows typing into memory, printing and punching up to the maximum number of characters specified in the A field. This instruction should be used in conjunction with the LKBR instruction to denote the entry position in memory for the characters typed. (See Subject 2.02.03.)

The XTKM instruction functions like a TKM instruction except that punching also occurs. The termination of this instruction with an OCK or PK places an End Alpha code in memory but does not cause a code punch.

If the perforator is turned off, or disconnected, the XTKM instruction functions as a TKM instruction.

|                             | <u>OP CODE</u> | <u>A</u>                 |
|-----------------------------|----------------|--------------------------|
| ENTER INTO MEMORY AND PUNCH | XEAM           | 0-150 15½" forms handler |
|                             | XEAM           | 0-255 26" forms handler  |

The XEAM instruction functions exactly like the XTKM instruction except that printing does not occur. If the perforator is turned off, or disconnected, XEAM will operate only as an EAM instruction.

|                       | <u>OP CODE</u> | <u>A</u> |
|-----------------------|----------------|----------|
| PRINT ALPHA AND PUNCH | XPA            | LABEL    |

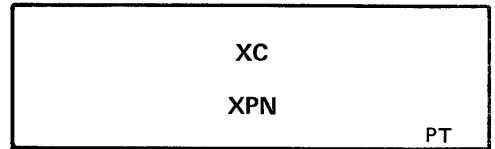
The XPA instruction prints and punches the alphanumeric data stored in the memory location designated by the A field. The instruction is terminated upon reaching an End of Alpha code in the data; the End of Alpha code is not punched. This instruction operates like a PA instruction in every respect except that punching occurs.

With the perforator turned off or disconnected, the XPA will operate as a PA instruction.

|                                    | <u>OP CODE</u> | <u>A</u> |
|------------------------------------|----------------|----------|
| PUNCH ALPHA FROM MEMORY, NON-PRINT | XA             | LABEL    |

The XA instruction functions exactly as an XPA instruction except that printing does not occur.

If the perforator is turned off or disconnected, the XA functions as a No Operation (NOP) instruction. When using Data Comm P. T. I/O firmware the XA will terminate on any Col. 0 USASCII Code. Codes from either column will punch.



|            | OP CODE | <u>A</u> | <u>B</u> |
|------------|---------|----------|----------|
| PUNCH CODE | XC      | 0-15     | 0-15     |

The XC instruction punches into tape the bit pattern specified by the parameter fields. The A parameter indicates the decimal value of the high order 4 bits (b<sub>8</sub>, b<sub>7</sub>, b<sub>6</sub>, b<sub>5</sub>, having decimal values of 8, 4, 2, 1 respectively); the B parameter represents the decimal value of the low order 4 bits (b<sub>4</sub>, b<sub>3</sub>, b<sub>2</sub>, b<sub>1</sub>, having decimal values of 8, 4, 2, 1 respectively) in the bit configuration of the desired code. The parity bit must be included in the appropriate bit position when applicable if a table look-up Firmware set is being utilized. If the standard USASCII I/O firmware set is used, the parity bit will be automatically inserted when applicable.

In the case of USASCII code the column number of the desired code in the table represents the A field (parity bit must be added when applicable); the row number of the desired code represents the B field.

Printing does not occur with this instruction. If the perforator is turned off or disconnected, the XC will function as a "No Operation" (NOP) instruction.

Example: Punch the USASCII code "RS"

|                                  | b <sub>8</sub>    | b <sub>7</sub> | b <sub>6</sub> | b <sub>5</sub> | b <sub>4</sub> | b <sub>3</sub> | b <sub>2</sub> | b <sub>1</sub> |
|----------------------------------|-------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Bit pattern ("X" = hole in tape) | 0                 | 0              | 0              | X              | X              | X              | X              | 0              |
| Decimal value                    | 8                 | 4              | 2              | 1              | 8              | 4              | 2              | 1              |
| Parameter value                  | A = (0+0+0+1) = 1 |                |                |                |                |                |                |                |
|                                  | B = (8+4+2) = 14  |                |                |                |                |                |                |                |

This corresponds to the USASCII table location of RS in column 1, row 14.

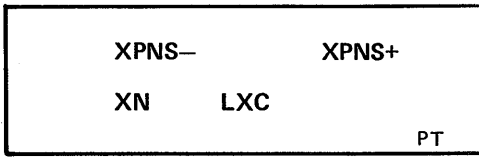
|                         | OP CODE | <u>A</u> | <u>B</u> |
|-------------------------|---------|----------|----------|
| PRINT AND PUNCH NUMERIC | XPN     | 0-14     | 0-15     |

The XPN instruction prints and punches the contents of the Accumulator, beginning with the high order digit position specified in the A parameter and with the print mask designated by the B parameter. The print mask is relative to the mask table established by the last LPNR instruction. (See Subject 2.03.04.)

There will be no affect on the Accumulator flags position or any other data in Accumulator positions to the left of the digit position specified by the A parameter.

This instruction functions like the PN instruction except that punching occurs.

If the perforator is turned off, or disconnected, the XPN instruction will operate only as a PN instruction.



|  | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--|----------------|----------|----------|
| PRINT AND PUNCH NUMERIC, SHIFT RIBBON IF MINUS | XPNS-          | 0-14     | 0-15     |
| PRINT AND PUNCH NUMERIC, SHIFT RIBBON IF PLUS  | XPNS+          | 0-14     | 0-15     |

The XPNS- instruction is the same as the XPN instruction except that the ribbon color is changed if the Accumulator Sign Flag is set (minus).

The XPNS+ instruction is the same as the XPN instruction except that the ribbon color is changed (opposite to the normal operating color of black, is red) if the Accumulator Sign Flag is reset (plus).

If the perforator is turned off or disconnected, the XPNS- and XPNS+ function as PNS- and PNS+ instructions respectively.

|                          | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--------------------------|----------------|----------|----------|
| PUNCH NUMERIC, NON-PRINT | XN             | 0-14     | 0-15     |

The XN instruction is the same as the XPN instruction except that printing does not occur. A mask word is used with this instruction since it controls the punching. (See Subject 2.03.05.) The mask word selected may be the same as is used with other Print Numeric Instructions since it would not affect the non-print function of this instruction.

If the perforator is turned off or disconnected, the XN will operate as a "No Operation" (NOP) instruction.

|                           | <u>OP CODE</u> | <u>A</u> |
|---------------------------|----------------|----------|
| LOAD PUNCH COUNT REGISTER | LXC            | 0-255    |

The Punch Count Register is provided to count the number of holes punched. This enables the use of continuous edge punched cards by making it possible to determine when one edge punched card has been filled or to fill any unused portion of a continuous card with feed codes before aligning the next continuous card to the first sprocket hole.

The LXC instruction will load the number contained in the A field, into the punch count register. The instruction is normally used at the start of each new continuous edge punched card to reset the count. The punch count register is incremented by one for each code punched from any punching instruction. If the register is equal to 255, incrementing causes the register to become 0.

|      |    |
|------|----|
| XMOD | XB |
|------|----|

OP CODE

MODIFY BY PUNCH COUNT REGISTER

XMOD

The XMOD instruction will modify the parameter field of the next instruction by the contents of the punch count register. This modification occurs as in the MOD instruction. The XMOD cannot be changed by the Index Register instructions. (i.e., IIR, ADIR, etc.)

OP CODE

A

PUNCH FEED CODES

XB

0-255

The XB instruction causes feed (sprocket) holes to be punched. The number of codes punched will be the difference between the number in the A field and 255.

If the perforator is turned off, XB will operate as a “No Operation” (NOP) instruction.

When edge punched cards are the media present, punching of sprocket holes is inhibited. Therefore, the card is just advanced without sprocket hole punching.

**2.16.02 READER AND PUNCH FLAGS**

Two reader flags are provided to enable program control over the tape reader.

Reader flag R1 is set when a reader condition exists. A reader condition exists if any of these contingencies arise:

1. The Paper Tape Reader is not turned on.
2. Media (paper tape or an edge punched card) must be positioned in the reader.
3. The media clamp must be closed.
4. The photo-electric device must be illuminated.

When the reader condition exists, along with the R1 flag being set, the keyboard buffer is cleared, and the instruction is held up from execution pending operator action. The action depends on two conditions:

1. The reader is intended to be used: Turn on the reader and then depress the Read Key. This reinitiates the read instruction and causes the media to be read. The R1 flag is reset.
2. The reader is not intended to be used: The operator may make an entry through the keyboard. (At this point, remember, the reader instruction has reverted to its keyboard instruction). The Reset Key will reinitiate the tape read instruction, but it must be indexed prior to the use of an OCK or PK.

Once the operator has taken either course of action, the indicator light is turned off and reader flag R1 is reset.

**NOTE: The keyboard buffer is cleared every time a reader instruction reverts to its keyboard counterpart. If the operator has anticipated this and indexed data prior to the halt in the program when the reader instruction becomes a keyboard instruction, then that data will be lost. The operator would have to index the data again.**

Reader flags R2, R3 are reserved for Data Communication operations.

Reader Flag R4 is set when an invalid tape code is read. Reading is not halted on the invalid tape code. The next read instruction will reset the R4 flag.

The Reader flag settings can be manipulated by use of the Flag instructions.

Four Punch Flags are provided to alert the operator of the perforator condition.

The Punch Flag P1 is set if media is not present in the perforator and the program attempts to execute a punch instruction. The instruction is halted. Correction of the situation will cause the system to resume execution of the punch instruction.

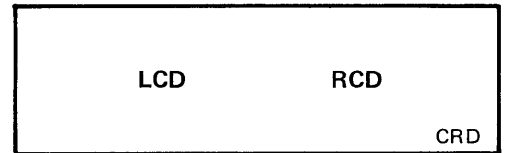
The Punch flag P2 is set if incorrect punching has occurred during a punch instruction. The echo check indicator light is lit. The punching is not terminated; the flag remains set.

The program should provide for checking flag P2 at least after each line of punching. When the flag is set, a Skip or Execute instruction would enable performing the necessary instruction to sound the alarm, punch a tape error code, or to take other corrective action.

The Punch flag P3 is set if reel tape is being used and the supply is nearly exhausted (approximately 20 feet remaining). The Tape Supply indicator is lit. Placing a new roll of tape in the supply reel will turn off the indicator and reset the flag on the next punch instruction. This condition does not halt program execution nor inhibit punching.

The Punch Flag P4 is set if the paper tape perforator is "OFF." The instruction will be executed, but the punching will be inhibited. Switching the perforator to the "ON" condition causes the P4 flag to be reset on the next instruction. However, the data to be punched on the first "punch" instruction would be missing from the output tape. Therefore, it is recommended that a punch instruction be used during the program initialization routine with subsequent testing of the Punch Flags (especially the P4 flag) since the perforator condition is only apparent once a punch instruction is initiated. All punch flags may be examined by use of the flag instructions.





## 2.17 – 80-COLUMN PUNCHED CARD INPUT INSTRUCTIONS

With the A 595 Card Reader and the A 149 Card Punch used as peripherals to either the Series L or TC, 80-column punched cards can be used as input and 80-column punched cards can be punched as output. The programing instructions required to use these two peripherals as part of a program will be explained in two sections. The first section will deal with card input instructions, the second will explain card output instructions.

### 2.17.01 80-COLUMN CARD INPUT INSTRUCTIONS

|                       | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-----------------------|----------------|----------|----------|
| LOAD MEMORY FROM CARD | LCD            | 0-255    |          |

The LCD instruction causes the reading of object program cards and stores the new object program instructions into memory locations specified in the program cards. The A parameter specifies the number of cards to be read. This instruction utilizes and requires that the Card Reader Memory Load Routine be present in the Utility Track.

LCD allows programmatic control of program overlays. After reading the designated number of program cards, the program execution continues on to the next instruction in accordance with the program counter. Thus, caution must be exercised to ensure that a program does not overlay the same memory area occupied by the LCD instruction. The program cards must be of the same format as required for regular program loading with the Card Reader. (Refer to Appendix K for card format required to load object program by card.)

After execution of this instruction, a “Hash Total” of the program data read in, is in the Accumulator.

If the specified number of program cards are not read, the instruction is held up, the Reader Condition light is turned on and the R1 flag is set.

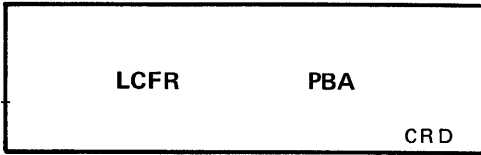
Placing the remaining cards to read in the Card Reader and depressing the Restart switch on the Card Reader, or depressing the Ready push button to return the machine to Ready mode, are the only two alternatives available to complete the LCD instruction.

|           | <u>OP CODE</u> |
|-----------|----------------|
| READ CARD | RCD            |

The RCD instruction reads a single 80-column punched card into words 1 through 10 of memory. All 80 columns are read and placed in memory including blank card columns.

During the execution of each RCD instruction, the contents of the Accumulator are destroyed and the Accumulator is not cleared. Any number in the Accumulator prior to a RCD instruction which is to be used later in the program, should be transferred to a memory location to save it, else it will be destroyed in the Accumulator.

If a card is not present in the Card Reader, when a RCD instruction is to be executed, the Reader Condition indicator light is turned on, flag R1 is set, and the instruction is held up.



Placing a card in the Card Reader and depressing the Restart switch on the Card Reader will enable the instruction to be completed and allow the program to continue to the next instruction. The other alternative would be to depress the Ready push button, to return the machine to Ready mode.

**LOAD CARD FORMAT REGISTER**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| LCFR           | LABEL    |

The LCFR instruction loads into the Card Format Register the word number associated with the Label name. A Card Format Table may contain up to 16 different card field formats. If more than 16 are required, another table location (i.e., another LCFR instruction with a different label) must be established before any formats can be referenced in the second table. Only one table can be referenced at one time, and that table referenced is dependent upon the last LCFR instruction.

The label in the A parameter must reference the beginning of a word. The Pseudo Instruction "WORD" should be used preceding the label of the first CDF pseudo instruction, so that it starts at the beginning of a word. (Refer to Subject 2.01 for explanation of Pseudo Instructions.)

Example:

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--------------|----------------|----------|----------|
|              | LCFR           | CRDTAB   |          |
|              | WORD           |          |          |
| CRDTAB       | CDF            | 1        | 2        |
|              | CDF            | 3        | 5        |

**PRINT ALPHA FROM CARD READ AREA**

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| PBA            | 1-16     |

The PBA instruction prints from the card read area, the field, specified by the format number, as alphanumeric data.

The format number, references the format table last identified by the LCFR instruction.

Example:

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>              |
|--------------|----------------|----------|----------|-----------------------------|
|              | LCFR           | CRDTAB   |          |                             |
|              | PBA            | 2        |          | Print second field on card. |
|              | NOTE           |          |          | Card cols. 3-10             |
| CRDTAB       | CDF            | 1        | 2        | Card cols. 1-2              |
|              | CDF            | 3        | 8        | Card cols. 3-10             |

|      |     |      |     |
|------|-----|------|-----|
| XPBA | XBA | TRCA |     |
|      |     |      | CRD |

PRINT & PUNCH ALPHA FROM CARD READ AREA

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| XPBA           | 1-16     |

The XPBA instruction prints from the card read area, the field specified by the format number, as alphanumeric data, and punches the data into an output card in the A 149 Card Punch. The instruction is terminated after printing and punching the number of characters specified by the field length in the format. The status of OCK flags is not affected.

If the Punch is off, XPBA is executed as a PBA instruction.

If there are no cards in the card hopper and the Punch is on and on-line, the XPBA instruction will be held up until cards are placed in the card hopper and the auto feed button depressed on the Punch.

PUNCH ALPHA FROM CARD READ AREA, NON-PRINT

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| XBA            | 1-16     |

The XBA instruction punches into an output card, from the card read area, the field specified by the format number, as alphanumeric data. The data is not printed. The instruction is terminated after punching the number of characters specified by the field length in the format.

If the Punch is off, XBA is executed as a NOP instruction.

If no cards are in the card hopper and the Punch is on line, the XBA instruction will be held up until cards are placed in the card hopper and the auto feed button depressed on the Punch.

TRANSFER CARD FIELD TO ACCUMULATOR AS NUMERIC

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| TRCA           | 1-16     |

The TRCA instruction transfers the field of data, specified by the format number in the A parameter, from the Card Read Area into the Accumulator. The digits in that field are right justified when transferred into the Accumulator. The instruction is terminated by transferring the number of card columns specified in the format. The status of the OCK flags is not changed by this instruction.

If an "11" overpunch is present in any of the card columns of the field being transferred (denoting a negative field), the Minus Flag in the Accumulator is set.

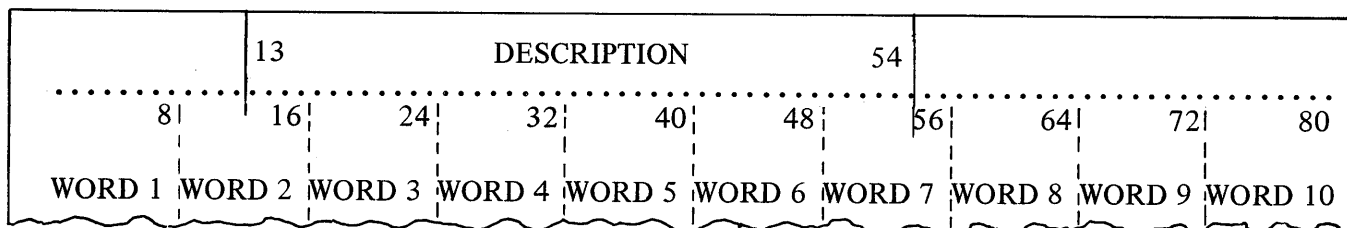
If a "12" or "0" overpunch is present in any of the card columns of the field being transferred, the Invalid Code Flag (R4) is set and the corresponding indicator light is turned on. An unknown digit will be transferred to the Accumulator. The flag is reset and the indicator is turned off at the beginning of the next Card Input Transfer instruction; therefore, this flag must be examined immediately in the program (with the SK or EX instructions) when it is necessary to detect illegal codes in a given field. The characters "+" (card codes 12,0) and "&" (card code 12) will not affect the Minus flag nor set the Invalid Code flag, but will transfer as the digit "0" in accordance with their position in the field. The hyphen character (minus sign) "-" (card code 11) and "X" (minus zero - card code 11,0) set the

Minus flag, do not set the Invalid Code flag, and are transferred as the digit "0" in accordance with their position in the field. The letters A through I and S through Z, as well as all other special characters, will set the Invalid Code flag and a digit will be transferred. The letters J through R are the same as numerals with an "11" overpunch. The space code (blank card column) is treated as the numeral "0".

An invalid code can be used to advantage to indicate special conditions, such as the last card in an input file. For example, a "12" overpunch with a transaction type number would permit the program to determine when to stop reading cards. This would not require a separate card column for this purpose, and would not affect the usability of the transaction number.

The programing below is an example of minimizing the length of alpha print time by examining certain positions of a description field in the card read area to determine the amount of significant data, and selecting a field format length accordingly; thereby eliminating some of the trailing space codes in the unused portion of the field when printing or transferring to memory.

The diagram below illustrates a card with a description field of 42 characters (col's. 13 to 54). On the premise that most descriptions are less than 21 characters, some are less than 29, only a few use the maximum field capacity, and that no more than 6 consecutive space codes are permitted within the description, then three formats are defined for the description field to permit the program to select the shortest length; thus, considerably reducing print time and/or transfer time (42 characters require approximately 2100 ms print time vs. approximately 1000 ms using a 20 character length format).



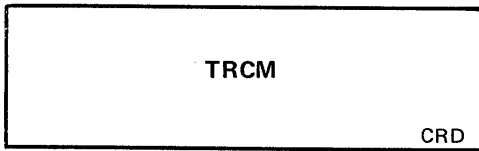
| <u>LABEL</u> | <u>INSTR</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>      |
|--------------|--------------|----------|----------|---------------------|
| FIELDS       | CDF          | 13       | 20       | SHORT DESCRIPTION   |
|              | CDF          | 13       | 28       | MEDIUM DESCRIPTION  |
|              | CDF          | 13       | 42       | MAXIMUM DESCRIPTION |

For simplest programing, the positions in the field to be examined for space codes must be defined taking into account the word boundaries of the card read area. The 21st through 28th positions in the description field are card columns 32 to 40 and are in word 5 (base word +4). If word 5 contains all zeros (8 space codes), then significant data is presumed to not extend beyond col. 32 (20th field position). If word 5 contains any significance, then word 6 is examined. If word 6 has all zeros, then data does not extend beyond col. 40 (28th position). If word 6 contains data, the infrequency of occurrence suggests that no further tests should be made and a maximum field size is used. The card read area is reserved with REG instead of CDB to permit a label for referencing specific words. (Refer to Subject 2.01 for explanation of Pseudo Instructions.)

Program Segments:

| <u>LABEL</u> | <u>INSTR</u> | <u>A</u>   | <u>B</u> | <u>REMARKS</u>  |
|--------------|--------------|------------|----------|---|
| START        | LPNR         | PMASKS     |          |   |
|              | LPKR         | PKEYS      |          |   |
|              | LLLR         | 51         |          |   |
|              | BRU          | BEGIN      |          |   |
| CARDIN       | REG          | 10         |          | RESERVE CARD READ AREA  |
| BEGIN        | ---          |            |          | Note that Card Read area is reserved with REG to permit labeling; but must be sequenced to assure assembly in words 1-10. |
|              | ---          |            |          |   |
|              | ---          |            |          |   |
|              | RCD          |            |          | READ A CARD   |
|              | LCFR         | FIELDS     |          | SELECT FORMAT TABLE   |
|              | LKBR         | DESCRP     |          | SELECT DESCRP TANK  |
|              | TRA          | CARDIN + 4 |          | READ COLS 33 TO 40  |
|              | SLROS        | 0          | 2        | MOVE FLAG POSITION  |
|              | EXZ          | 3          |          | EXAMINE FOR SPACES  |
| a            | PBA          | 1          |          | PRINT SHORT FIELD   |
| a            | TRCM         | 1          |          | TRANSFER SHORT FLD  |
| a            | BRU          | + 9        |          |   |
| b c          | TRA          | CARDIN + 5 |          | READ COLS 41 TO 48  |
| b c          | SLROS        | 0          | 2        | MOVE FLAG POSITION  |
| b c          | EXZ          | 2          |          | EXAMINE FOR SPACES  |
| b            | PBA          | 2          |          | PRINT MEDIUM FIELD  |
| b            | TRCM         | 2          |          | TRANSFER MED FLD  |
| b c          | SKZ          | 2          |          | EXAMINE FOR DATA  |
| c            | PBA          | 3          |          | PRINT LONG FIELD  |
| c            | TRCM         | 3          |          | TRANSFER LONG FLD   |
|              | DESCRP       | REG        | 6        | DESCRIPTION WORK AREA   |

Note: The key along the left margin indicates the program path selected depending on field size; "a" = short field, "b" = medium field, "c" = long field. Statements without a key are executed by all three paths.



TRANSFER CARD COLUMNS TO MEMORY  
AS ALPHA

|                |          |
|----------------|----------|
| <u>OP CODE</u> | <u>A</u> |
| TRCM           | 1-16     |

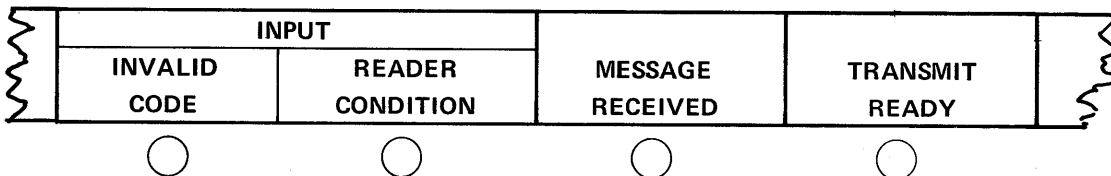
The TRCM instruction transfers the field specified by the format number in the A parameter to a memory location starting with the word designated by the prior use of the LKBR instruction. The instruction is terminated after transferring the number of characters specified by the field length in the format. An "End of Alpha" code is placed in memory following the last code transferred. The status of OCK flags is not affected.

Space codes (blank columns) are transferred and translated as Space Codes; in subsequent printing of this data from memory (not the card read area) with the PA instruction, the space characters will cause the printer to escape rather than increment the position register. This condition would be common in the unused portion of a description field such as name or address, when the card input data has to be retained for further processing while additional cards are being read. Escaping through space codes can be reduced, by programmatically examining certain points in the card read field and using a smaller field format when transferring the field to memory. This may be desirable when the field must be designed with a large capacity to accommodate all transactions, but which may have many transactions with small entries of data (see example, above).

An indication of Invalid Code is not provided if an incorrect combination of punches has been read into the Card Read Area. Invalid Code indication is only included with the TRCA instruction.

**2.17.02 INPUT INDICATOR LIGHTS AND FLAGS**

The two Series L keyboard input indicator lights advise the operator as to whether the Card Reader is operable, and, under certain conditions, whether invalid codes have been read. Also, the associated Reader flags enable the program to provide alternate procedures in the event of a Reader Condition or invalid code.



Input Indicator Lights

**INVALID CODE INDICATOR** – The Invalid Code Indicator is turned on and its associated flag (R4) is set, when, during the execution of the TRCA (Transfer to Accumulator) instruction, a code is sensed that represents an invalid combination as described in the TRCA instruction. This flag is reset and the Indicator turned off at the beginning of the next transfer instruction.

**READER CONDITION INDICATOR** – The Reader Condition Indicator is illuminated and flag R1 set when a card read instruction (RCD) is being executed and any of the following conditions exist:

1. The reader is not on
2. The reader is out of cards
3. Burned out bulb in reader

The read instruction is held up pending operator action as follows:

1. If the Reader is out of cards, the placing of cards in the feed hopper and depression of the Restart Switch on the reader will then cause the card read instruction to be executed.
2. If the Reader is not on, the Reader power on switch must first be turned on and then the Restart switch depressed.
3. The use of the Ready push button, at this point will return the program to the READY mode.

The R1 flag is set only while waiting to read a card, and is reset when the instruction is executed. Therefore, only the Indicator light can be used to notify the operator of this condition.

The R2 and R3 flags are set or reset by Data Comm instructions and are not controlled by card instructions.

**FLAG INSTRUCTIONS (LOAD, SET, RESET, CHANGE)** – The execution of a LOD, SET, RST, or CHG Flag instruction involving the Reader Flags will also cause their associated indicator lights to either be turned on or off depending on the instruction used.

### **2.17.03 PROGRAM KEYS**

Program Keys that have been enabled prior to a Card Read instruction or any of the Card Transfer instructions will be ignored during those instructions. If a Reader Condition occurs and the Card Read instruction is held up, use of a PK will have no immediate affect except to place the PK code in the keyboard buffer pending the next keyboard instruction where it will be recognized.

## 2.18 – 80-COLUMN CARD OUTPUT INSTRUCTIONS

### 2.18.01 PUNCHING ALPHANUMERIC DATA

The following instructions provide for punching alphanumeric data during keyboard entry or directly from storage in memory. Each use of one of these instructions punches one field, or a portion thereof, depending on the number of characters and the field size. Therefore the SKP (See Subject 2.18.03) instruction should normally be used following each of these instructions to by-pass unused trailing positions in the field and to position the card to the first column in the next field.

|                | <u>OP CODE</u> | <u>A</u>                 |
|----------------|----------------|--------------------------|
| TYPE AND PUNCH | XTK            | 0-150 15½" forms handler |
|                | XTK            | 0-255 26" forms handler  |

The XTK instruction combines typing, printing and punching up to the maximum number of characters specified in the A parameter. This instruction functions like a TK instruction in most respects with the additional function of punching the data into an 80-column card. However, the use of the Backspace Key is disabled, since a code would already have punched. The termination of this instruction with an OCK or PK does not cause a code to punch.

If the punch is off-line, XTK will be executed only as a TK instruction.

The use of the Backspace Key has been prohibited; therefore, if it is depressed, an error state occurs which requires depression of the Reset Key. Caution must be exercised with use of the Reset Key since, if in the middle of a keyboard entry but not in an error state, use of the Reset Key re-initiates the instruction and sets the LXC Register back to the start of the field. This puts the card out of step since part of the field has already punched. These considerations also apply to XTKM and XEAM following.

|                                   | <u>OP CODE</u> | <u>A</u>                 |
|-----------------------------------|----------------|--------------------------|
| TYPE INTO MEMORY, PUNCH AND PRINT | XTKM           | 0-150 15½" forms handler |
|                                   | XTKM           | 0-255 26" forms handler  |

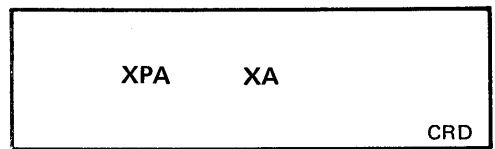
The XTKM instruction combines typing, printing, entering the data into memory and punching up to the maximum number of characters specified in the A parameter. The prior use of LKBR designates the starting word for storing the data. The XTKM instruction functions like the TKM instruction in every respect with the additional function of punching into an 80-column card. However, the use of the Backspace Key is disabled (see XTK) since a code would already have punched. The termination of this instruction with an OCK or PK does not cause a code to punch, but does place an End of Alpha code in memory.

If the Punch is off-line, XTKM is executed only as a TKM instruction.

|   | <u>OP CODE</u> | <u>A</u>                 |
|---|----------------|--------------------------|
| ENTER ALPHA INTO MEMORY AND PUNCH,<br>NON-PRINT | XEAM           | 0-150 15½" forms handler |
|   | XEAM           | 0-255 26" forms handler  |

The XEAM instruction functions exactly like the XTKM instruction except that printing does not occur. If the Punch is off-line, XEAM is executed only as an EAM instruction.





OP CODE      A

PRINT ALPHA AND PUNCH

XPA      LABEL

The XPA instruction prints and punches the alphanumeric data stored in the memory location designated by the A parameter. The instruction is terminated upon reaching an End of Alpha code in the data; the End of Alpha code does not punch. This instruction functions like a PA instruction in every respect with the additional function of punching into an 80-column card. If the Punch is off-line, the XPA instruction is executed only as a PA instruction.

OP CODE      A

PUNCH ALPHA FROM MEMORY, NON-PRINT

XA      LABEL

The XA instruction functions exactly like the XPA instruction except that printing does not occur. If the Punch is off-line, XA is executed as a NOP instruction.

**2.18.02 PUNCHING NUMERIC DATA FROM THE ACCUMULATOR**

The following instructions provide for printing and punching, or just punching, numeric data from the Accumulator. The Pointer designates the high order digit position of the Accumulator at which printing and punching begin; the printing format and punching are controlled by the Mask word selected. The instruction is terminated after punching and printing through digit position zero or when an "E" (End) Mask code is encountered in the Mask word. A Mask word is used for all punch numeric instructions even though printing may not be a function of a given instruction. It serves to right justify the numeric data in the card field, filling in preceding zeros or blank columns. Therefore, a fixed field length results and the use of SKP subsequently is not needed.

The Punch Flag (P) in the Mask word, when set, causes leading zeros to punch even though leading zero suppression Mask codes (Z,Z) prevent their printing. If the Punch Flag is not set, a blank card column results for each leading zero suppressed by a Z (or Z,) Mask code; however, if the Punch Flag is not set and if an Unconditional Print Mask code is used (D D, etc.), all leading zeros will punch into the card (refer to the following table). The Punch Flag has no effect on the print characteristics of the Mask codes.

| MASK CODE           | PRINTING   | PUNCHING   |
|---------------------|--|--|
| F                   | Print \$   | No Effect  |
| +                   | Suppress Punctuation   | No Effect  |
| P                   | No Effect  | Leading zeros punch if P flag set, blank card column if reset                                  |
| D<br>D,<br>.D<br>D: | Print Character regardless of significance   | Punch Character regardless of significance   |
| X<br>.X             | Trailing zero suppression  |  |
| C<br>.C             | Leading zero & trailing zero suppression   |  |
| Z<br>Z,<br>Z:<br>S  | Print if:<br>(1) Accum digit not zero<br>(2) A non-zero digit has been printed<br><br>Print only if Accum digit not zero | Punch if:<br>(1) P is Set<br>(2) Accum digit not zero<br>(3) A non-zero digit has been punched |
| I                   | Ignore   | Ignore   |
| E                   | Terminate, Non-print   | Terminate, Non-punch   |

TABLE

If an Ignore (I) Mask code is used, the corresponding digit in the Accumulator does not print or punch. If the End (E) Mask code is used, the corresponding digit neither prints nor punches and the instruction is terminated. All other Mask codes cause the corresponding digit to punch.

The punctuation provided by some of the Mask codes during printing does not punch.

|                            |
|----------------------------|
| <b>XPN</b><br><b>XPNS—</b> |
| CRD                        |

In a numeric field on the output card, if only significant digits are to be interpreted along the top of the card, then leading zeros of the numeric word in the Accumulator must be represented by blank card columns in the output card (P Flag must be reset and "Z" mask codes used in order for this to occur).

|                       | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-----------------------|----------------|----------|----------|
| PRINT & PUNCH NUMERIC | XPN            | 0-14     | 0-15     |

The XPN instruction prints and punches the contents of the Accumulator, starting at the high order digit position designated by the A parameter, in accordance with the print mask designated by the B parameter. The print mask value is relative to the mask table base word established by the last LPNR instruction. This instruction functions like a PN instruction in every respect with the additional function of punching.

If the Accumulator Minus Flag is set, an "11" overpunch is punched with the least significant digit of the Accumulator (digit 0); if minus, and if the mask word terminates printing/punching prior to digit 0 (with an "E") or ignores digit 0 (with an "I"), an "11" overpunch does not punch. If the "11" overpunch is not desired in the field, the Minus flag must first be reset.

All Accumulator digits of a higher order position than the A parameter are ignored.

When it is necessary to punch a plus "+" or minus "-" sign into a separate card column, or when the value of the other Accumulator flags (S, C, M) must be punched, this can be accomplished by testing the individual flag settings (SK or EX) and punching an appropriate code in the card column(s) with the XC (Punch Code) instruction prior to or after punching the numeric field with the XPN instruction. If the sign column must follow the numeric field, a set Minus flag must first be reset before punching the data; this usually requires separate program paths, after testing for a minus condition, to both punch the data and punch the correct sign code.

If the Punch is off-line, XPN is executed only as a PN instruction.

|  | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--|----------------|----------|----------|
| PRINT & PUNCH NUMERIC, SHIFT RIBBON IF MINUS | XPNS—          | 0-14     | 0-15     |

The XPNS— instruction is the same as the XPN instruction except that the ribbon color is changed if the Accumulator Sign Flag is set (Minus). If the Punch is off-line, XPNS— is executed only as a PNS— instruction.

|       |    |     |
|-------|----|-----|
| XPNS+ | XN |     |
|       | XC |     |
|       |    | CRD |

|   | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---|----------------|----------|----------|
| PRINT & PUNCH NUMERIC, SHIFT RIBBON IF PLUS | XPNS+          | 0-14     | 0-15     |

The XPNS+ instruction is the same as the XPN instruction except that the ribbon color is changed if the Accumulator Sign Flag is reset (Plus). If the punch is off-line, XPNS+ is executed only as a PNS+ instruction.

**Punch Numeric Data, Non-print**

|                          | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--------------------------|----------------|----------|----------|
| PUNCH NUMERIC, NON-PRINT | XN             | 0-14     | 0-15     |

The XN instruction is the same as the XPN instruction except that no printing occurs. A mask word is used with this instruction since it controls punching, and may be the same mask word used with other Print Numeric instructions as there would be no affect on the non-print characteristic of XN. If the punch is off-line, XN is executed as a NOP instruction.

|            | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|------------|----------------|----------|----------|
| PUNCH CODE | XC             | 0-15     | 0-15     |

The XC instruction permits outputting any desired single card code (without it being resident in memory) or any special punch pattern in a card column (except only one punch can be created in rows 1 to 7 in a card column although any punch combination in the other rows can be obtained). The A parameter controls punching in card rows 12, 11, 0, and 9; the B parameter controls punching in card rows 1 through 8.

Printing does not occur with this instruction. If the Punch is off, XC is executed as a NOP instruction.

|                   | <u>ROWS</u><br><u>12, 11, 0, 9</u> | <u>ROWS</u><br><u>1-8</u> |
|-------------------|------------------------------------|---------------------------|
| A Parameter Value | 8 4 2 1                            |                           |
| B Parameter Value |                                    | 1-8                       |

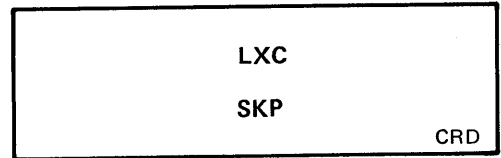
To punch an "A" (Row 12, 1) the XC instruction would be

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| XC             | 8        | 1        |

To punch Rows 12, 11, 0, 8, 6 the XC instruction would be

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| XC             | 14       | 14       |

Refer to Appendix H to find A and B parameter values of various characters to be punched.



### 2.18.03 CARD COLUMN SYNCHRONIZATION WITH THE PUNCH COUNT REGISTER

A Punch Count Register is used by firmware to count the card columns either punched or escaped in order to control the location of the card and maintain synchronization. When the system is turned on, the value in this register is indeterminable, and therefore it must be loaded with the value "1" at the start of a program.

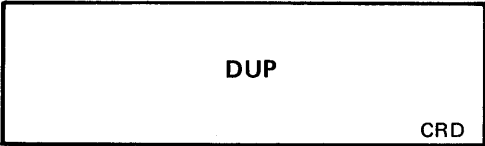
|                           | <u>OP CODE</u> | <u>A</u> |
|---------------------------|----------------|----------|
| LOAD PUNCH COUNT REGISTER | LXC            | 1        |

The LXC instruction loads the value specified in the A parameter into the Punch Count Register. The parameter value must be "1" to synchronize the register with the card in the punch station (card must be registered in the punch station at card column one).

The LXC instruction is normally used only once in a program, during the initialization routine. Once into the program, firmware resets the Punch Count Register to 1 whenever a card is released in the punch and another card registered at column 1. However, it is recommended that a provision be included in the program for the operator to reset the register to 1 in the event a card becomes out of step. This condition could occur from the improper use of the keyboard Reset Key during a keyboard entry, or from inadvertent manipulation of the control keys on the card punch (which should not be necessary once a program is in operation). Note that if the keyboard Reset Key is used during a keyboard entry and the system is not in an error state, the keyboard instruction is re-initiated (repositioning the printer and permitting a complete new entry) and the Punch Count Register is set back to the beginning column of that field; thus, the card must be backspaced to the same card column, using the Backspace control on the card punch, to regain synchronization.

|                | <u>OP CODE</u> | <u>A</u> |
|----------------|----------------|----------|
| SKIP TO COLUMN | SKP            | 1-80     |

The SKP instruction causes the card to skip to the card column specified in the A parameter. A skip to card column 1 causes the card to be released and a new card registered at column 1. This is the prescribed manner in which the Series L program releases a card. If the card is presently on the card column specified by the SKP instruction, no skipping occurs. An exception to this is a skip to 1 when the card is already on column 1; this results in the card being released and another card registered.



Once the skip function has been initiated, the program resumes execution while the skipping is being completed, except for skips of up to 3 columns. If the program reaches another punch instruction while skipping is occurring, the program is held up until skipping has been completed. Skips of 3 columns or less are actually treated as Punch Blanks (XC 0 0, blank card columns), and in this situation, program execution is held up until the skip is completed.

A skip to a lesser numbered column than the present card location will cause the release of the card and the registration of a new card; however, the count register will be in error for the newly registered card.

If the punch is off-line, the SKP instruction is executed as a NOP instruction.

The SKP instruction should normally be used after each punch instruction where unused card columns could remain, such as with XTK, XTKM, XPA, etc. It is normal for these instructions to be terminated before punching the total number of characters specified in the parameter; therefore, a SKP instruction must be used to ensure that the card is properly positioned to the start of the next field.

DUPLICATE THROUGH COLUMN

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| DUP            | 1-80     |

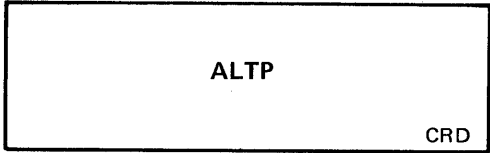
The DUP instruction causes data from the card in the Read Station to be punched (duplicated) into the corresponding columns of the card in the punch station. The duplication function starts at and includes the card column at which it is initiated, and continues through the card column specified in the A parameter. A DUP through 80 will cause the card to be duplicated through column 80, released, and a new card registered at column 1. A DUP through the same card column number as the present location of the card results in no duplication.

Once the duplication function has been initiated, the program resumes execution while the duplication is being completed. If the program reaches another punch instruction while duplication is occurring, the program is held up until the duplication has been completed.

A DUP through a lesser numbered card column than the present location of the card will cause a duplication through column 80, release of the card and registration of a new card; however, the count register will be in error for the newly registered card.

If the punch is off-line, the DUP instruction is executed as a NOP instruction.

Cards are released from the punch station by the Series L program with the use of a Skip to Column 1 instruction (SKP 1) or a Duplicate Through Column 80 instruction (DUP 80). Use of the card punch manual controls, during program operation, or any other type of program release will in most cases cause the newly registered card to be out of synchronization with the Punch Count Register.



The Regular Card Stacker is selected automatically if the program has not specified otherwise for the card being released. The Alternate Stacker is selected by executing the following instruction:

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <p>ALTERNATE STACKING POCKET</p> | <p><u>OP CODE</u></p> <p>ALTP</p> |
|----------------------------------|-----------------------------------|

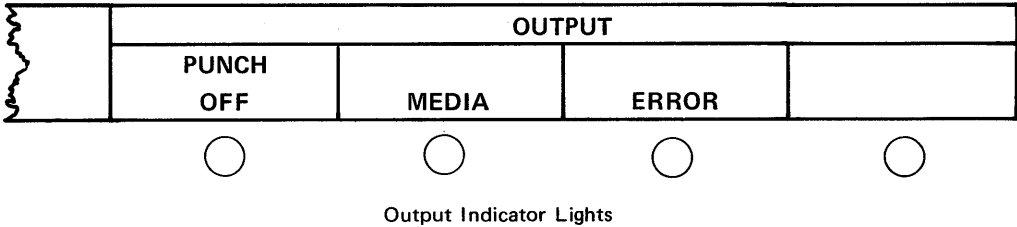
The ALTP instruction causes the card in the Punch Station to be routed to the Alternate Stacking Pocket after it has been released from both the Punch Station and the Read Station. The ALTP instruction must be executed while the card is still in the Punch Station, and prior to any instruction that will cause the card to be released from the Punch Station, in order to affect that card when it is finally released from the Read Station.

This instruction can be used to advantage in many ways, such as to segregate two groups of transactions, or to out-sort special information cards from standard transaction cards (such as low quantity alerts, etc.) or to collect reject cards from error entries.

If the punch is off-line, the ALTP instruction is executed as a NOP instruction.

**2.18.04 OUTPUT INDICATOR LIGHTS AND FLAGS**

Three of the Output Indicator Lights on the Series L keyboard are used to advise the operator of the operating status of the card punch.



The Punch Off Indicator Light is turned on and Punch Flag P4 is set if the card punch "On-Line" switch is not on, or if the On/Off switch is not on while a card punching instruction is attempted. The punch portion of the instruction is inhibited and the instruction is executed in the manner of its counterpart keyboard or print instruction. The program does not halt. An instruction involving no other functions but punching is executed as a NOP instruction. The correction of the condition by turning on the punch and placing it in the On-Line mode will cause the indicator to be turned off and Punch Flag P4 to be reset on the next punch instruction.

To avoid the possibility of the operator failing to turn on the punch when beginning an operation, it is recommended that during the program initialization a card be released (SKP 1) and the Punch Off Flag P4 be examined. If P4 is set, the program can warn the operator (with the Alarm or by printing a warning message) and in addition may prohibit further processing or halt to allow an operator decision as to whether the following group of transactions requires card output.

If the program attempts to execute a punch instruction and a card is not registered in the punch station, the instruction is held up, the Media Indicator light is turned on, and Punch Flag P1 is set. Correction of the condition by registering a card in the punch station permits the instruction to be executed, at which time the Indicator light is turned off and Punch Flag P1 is reset. Only the Indicator light can be used to notify the operator that a card is not present in the punch station since the P1 flag is set only while the punching instruction is held up and is reset after the punching instruction is executed.

The Error Indicator Light is turned on and Punch Flag P2 is set if a card punch malfunction or misoperation occurs. If this condition occurs, the card punch is not operative, the RESET key (switch-light) on the card punch is turned on, and the program is held up on the punch instruction. A depression of the RESET key removes the error condition and permits execution of that instruction to be completed and the program to continue; Punch Flag P2 and the Indicator light are turned off.

Depression of the RESET key does not change the fact that mis-punching may have occurred, or that a newly registered card may be out of synchronization with the punch count register.

The execution of a LOD, SET, RST, or CHG Flag instruction involving the Punch Flags will also cause their associated indicator lights to either be turned on or off depending on the instruction used.

Program keys that have been enabled prior to a card punch instruction involving a keyboard entry (XTK, XTKM, XEAM) may be used to terminate that instruction. If the instruction is terminated with an OCK, such PK's as were enabled will be disabled.



**SUBJECT 2.19 – MAGNETIC UNIT RECORD INSTRUCTIONS**

The Magnetic Unit Record (MUR) Instructions provide the ability to read data from or write data on, a single magnetic record on a magnetic record card. These instructions apply to a unit record handling mechanism integrated into the console of the system with the magnetic unit record option, or an option magnetic record handling Auto Reader. All reading and writing is from a 22-word section of main memory used as an input/output buffer. Input Instructions provide the ability to read data from the magnetic record, to transfer the variable length data fields from the buffer into either memory or the accumulator, and to process data directly from the buffer. Output Instructions provide the ability to transfer both numeric and alpha data to the buffer and to write the contents of the buffer on the magnetic record. The location of the buffer is dependent upon, and specified by the type of firmware used.

A maximum of 349 digits of data, plus 2 line-find digits, and a block check digit, may be stored on the magnetic record of a standard 11" magnetic unit record. The data is read from, or written on, the magnetic record in one continuous motion of the record mechanism past the read/write heads. There are no separation digits or characters written on, or read from, the magnetic record. All data field formatting is accomplished after the data has been read from the magnetic record into the buffer, following the read or input mode, and upon entry of data into the buffer prior to the write or output mode. Formatting of data is accomplished by values stored in a stripe format table.

**2.19.01 MAGNETIC UNIT RECORD FORMATS**

The Magnetic Record Format specifies the starting digit location and the length of a data field within the magnetic record input/output area. This allows variable length data fields to be moved from, or inserted into, the input/output buffer. The values that describe these fields are contained in a Stripe Format Table. A Stripe Format Register is used to contain the memory location of the first word of the Stripe Format Table, and it must be loaded in the program before any fields are accessed.

|                             |                |          |          |
|-----------------------------|----------------|----------|----------|
|                             | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| LOAD STRIPE FORMAT REGISTER | LSFR           | LABEL    |          |

The LSFR instruction provides the ability to establish the location of a Stripe Format Table in memory. The format instruction loads the Stripe Format Register with the memory location of the label contained in the A parameter. The Stripe Format Register establishes the base address of the Stripe Format Table. A format table for the magnetic record is 16 words in length, and may contain up to 64 formats. More than 1 table may be used; however, when replacing a table currently in use, the base address of the replacement table must be initialized by an LSFR (Load Stripe Format Register) instruction.

Example:

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |       |  |  |  |  |  |                   |  |    |    |    |    |    |    |    |    |    |  |  |
|----------|----|----------------------|-----------|----|----|----|-------|--|--|--|--|--|-------------------|--|----|----|----|----|----|----|----|----|----|--|--|
|          |    |                      | A         |    |    |    |       |  |  |  |  |  |                   |  | B  |    | C  |    |    |    |    |    |    |  |  |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | LABEL |  |  |  |  |  | + OR -<br>INC/REL |  | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |
| LSFR     |    |                      | FIELDS    |    |    |    |       |  |  |  |  |  |                   |  |    |    |    |    |    |    |    |    |    |  |  |
|          |    |                      |           |    |    |    |       |  |  |  |  |  |                   |  |    |    |    |    |    |    |    |    |    |  |  |

The initial phase of execution will open the handler if closed. The data is written while the magnetic record is being ejected.

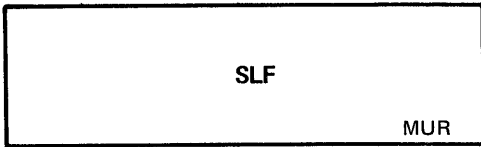
If a write error occurs, the W (write error) flag is set. All error recovery routines are programmatic.

Example:

|       |    |    |    |    |    |        |     |    |    |    |    |    |                 |    |    |    |    |    |    |    |    |    |    |    |    | PARAMETER    |    |    |    |    |    |  |  |  |  |  |  |
|-------|----|----|----|----|----|--------|-----|----|----|----|----|----|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|--|--|--|--|--|--|
|       |    |    |    |    |    |        |     |    |    |    |    |    |                 |    |    |    |    |    |    |    |    |    |    |    |    | FIELD LENGTH |    |    |    |    |    |  |  |  |  |  |  |
| LABEL |    |    |    |    |    |        |     |    |    |    |    |    | OP. CODE        |    |    |    |    |    |    |    |    |    |    |    |    | PARAMETER    |    |    |    |    |    |  |  |  |  |  |  |
| A     |    |    |    |    |    |        |     |    |    |    |    |    | B               |    |    |    |    |    |    |    |    |    |    |    |    | C            |    |    |    |    |    |  |  |  |  |  |  |
| LABEL |    |    |    |    |    |        |     |    |    |    |    |    | + OR - INC. REL |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
| 16    | 17 | 18 | 19 | 20 | 21 | 22     | 23  | 24 | 25 | 26 | 27 | 28 | 29              | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42           | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |  |
|       |    |    |    |    |    | SRJ    |     |    |    |    |    |    | WRITEL          |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        |     |    |    |    |    |    | W               |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    | WRITE  | RL  |    |    |    |    |    | 3               |    |    |    |    |    |    |    |    |    | 0  |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | EX  |    |    |    |    |    | S               |    |    |    |    |    |    |    |    |    | R  | 2  |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | PKA |    |    |    |    |    | 1               |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | BRU |    |    |    |    |    |                 |    |    |    |    |    |    |    |    |    | -3 |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    | WRITEL | WL  |    |    |    |    |    |                 |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | EX  |    |    |    |    |    | S               |    |    |    |    |    |    |    |    |    | W  | 3  |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | PKA |    |    |    |    |    | 3*              |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | TK  |    |    |    |    |    | 0               |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | BRU |    |    |    |    |    |                 |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |
|       |    |    |    |    |    |        | SRJ |    |    |    |    |    | 1               |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |  |  |  |  |  |  |

| LABEL  | OP CODE | A      | +1- | B | C | REMARKS                   |
|--------|---------|--------|-----|---|---|---------------------------|
|        | SRJ     | WRITEL |     |   |   | GO TO WRITE RECORD        |
| WRITE  | RL      | 3      |     | 0 |   | NON-READ AND ALIGN RECORD |
|        | EX      | S      |     | R | 2 | EX IF JAM/READ ERROR      |
|        | PKA     | 1      |     |   |   | ENABLE RECONSTRUCT PK     |
|        | BRU     |        | -3  |   |   | GO RETRY                  |
| WRITEL | WL      |        |     |   |   | WRITE MAGNETIC RECORD     |
|        | EX      | S      |     | W | 3 | EXECUTE IF WRITE ERROR    |
|        | PKA     | 3*     |     |   |   | PKA 3—WRITE ERROR ROUTINE |
|        | TK      | 0      |     |   |   | HALT FOR PK SELECTION     |
|        | BRU     |        | -2  |   |   | GO TO SELECT PK.          |

\*PKA 3 – BRU WRITE



|                |          |          |  |
|----------------|----------|----------|--|
| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>   |
| LSFR           | FIELDS   |          | LOAD THE STRIPE FORMAT REGISTER WITH FIELDS,<br>THE BASE ADDRESS OF THE STRIPE FORMAT TABLE. |

**2.19.02 MAGNETIC UNIT RECORD PSEUDO INSTRUCTIONS**

The Pseudo instructions allow the programmer to communicate both with the assembler program and the system. These Pseudo instructions do not directly produce machine language instructions for the object program. They do, however, control the manner of assembly, determine the interpretation of data input to the assembler and exert control over the system such as forms control and word-syllable counter control.

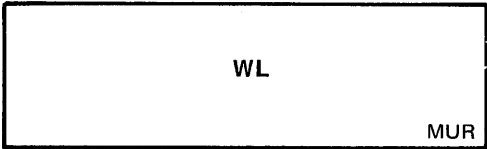
|                                 |                |          |                                       |
|---------------------------------|----------------|----------|---------------------------------------|
|                                 | <u>OP CODE</u> | <u>A</u> | <u>B</u>                              |
| MAGNETIC RECORD FORMAT (PSEUDO) | SLF            | 1-349    | 1-15 (numeric)<br>1-63 (alphanumeric) |

The SLF instruction is used to format the magnetic record data (read from the unit record) during a transfer from the input area into either memory or the accumulator, or is used to format data transfer to the output area prior to a magnetic record write instruction.

The A parameter specifies the starting digit location of a data field; the B parameter specifies the length of that data field within the magnetic record input/output area. Signs for signed numeric data require a digit. Alpha characters require two digits. The values entered are assembled into one syllable as part of the Stripe Format Table which begins at the location designated by the use of the LSFR instruction (Load Stripe Format Register). The table may contain up to 64 field formats if more than 64 are required, another table must be designated with LSFR. The table must begin with syllable 0 of the designated word; therefore, it should be preceded with the "WORD" pseudo instruction to assure proper assembly.

Example:

| LABEL |    | OP. CODE | FIELD<br>LEN-<br>GTH | PARAMETER |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|-------|----|----------|----------------------|-----------|----|------|----|----|----|----|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|
|       |    |          |                      | A         |    | B    | C  |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
| 16    | 17 | 18       | 19                   | 20        | 21 | 22   | 23 | 24 | 25 | 26 | 27     | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |
|       |    |          |                      |           |    | LSFR |    |    |    |    | FIELDS |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |                      |           |    |      |    |    |    |    |        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |



| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>              |
|--------------|----------------|----------|----------|-----------------------------|
|              | LSFR           | FIELDS   |          | LOAD STRIPE FORMAT REGISTER |
|              | WORD           |          |          |                             |
| FIELDS       | SLF            | 1        | 31       | 1—ACCOUNT NAME              |
|              | SLF            | 63       | 4        | 2—CHECK COUNT               |
|              | SLF            | 67       | 7        | 3—ACCOUNT NUMBER            |
|              | SLF            | 74       | 11       | 4—BALANCE + SIGN            |
|              | SLF            | 85       | 11       | 5—LOW MONTHLY BAL. + SIGN   |

**2.19.03 MAGNETIC UNIT RECORD FLAG**

Three flags (the “S” group) are included in the system with the Magnetic Record option: the Read Error Flag (R), the Filled Sheet Flag (F), and the Write Error Flag (W).

READ ERROR FLAG (R) – The Read Error Flag is set if a read error occurs during the record-read process. Read errors occur because of the following conditions:

1. The data encoded on the magnetic unit record has become corrupted.
2. There is a blank magnetic record in the magnetic unit record mechanism.
3. The magnetic record is prematurely removed from the mechanism.

The “R” flag may be interrogated by the Skip and Execute instructions, but is reset by the initiation of the next read or write instruction.

FILLED SHEET FLAG (F) – The Filled Sheet Flag is set when the Stripe Count Register is incremented to a value of 1 greater than the contents of the Stripe Limit Register. The “F” flag may be interrogated by the Skip and Execute instructions, but it is reset by the initiation of the next read or write instruction.

WRITE ERROR FLAG (W) – The Write Error Flag is set if a write error occurs during the record-write process. Write errors occur because of the following conditions:

1. The magnetic record in the mechanism is improperly coded.
2. There is no unit record in the mechanism.
3. The magnetic unit record is prematurely removed from the mechanism.

The “W” flag may be interrogated by the Skip and Execute instructions, but is reset by the initiation of the next read or write instruction.

**2.19.04 WRITE INSTRUCTIONS**

| <u>WRITE RECORD</u> | <u>OP CODE</u> |
|---------------------|----------------|
|                     | WL             |

The WL instruction writes the data from the Magnetic Record Buffer onto the magnetic record on the unit record. The line number contained in the Stripe Count Register is written in the line-find-digits area of the magnetic record.



**2.19.05 READ INSTRUCTION**

|             | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-------------|----------------|----------|----------|
| READ RECORD | RL             | 0-5      | 0-15     |

The RL instruction provides the ability to read the magnetic record on a unit record either from the console mechanism, or from the auto reader. This instruction is comprised of two operational phases. Phase one is a numeric keyboard operation and phase two is a read and/or align operation. (There is not a numeric phase on a read from auto reader instruction.)

The A parameter specifies the type of read and/or alignment. It also specifies the input device. The possible entries for the A parameter are:

- 0 – Read and align to the line number on the magnetic record.
- 1 – Read and align to the line number contained in the Stripe Count Register.
- 2 – Read and align to posting line 1 (the first posting line).
- 3 – Non-read and align to the line number contained in the Stripe Count Register.
- 4 – Read and eject record.
- 5 – Read from auto reader.

Parameter 0 – Reads the magnetic record and loads the line number contained on the magnetic record, automatically, incremented by one by firmware because it is the last posting line number, into the Stripe Count Register, and aligns the ledger to the contents of the Stripe Count Register.

Parameter 1 – Reads the magnetic record, ignores the line-find digits, and aligns the unit record to the number contained in the Stripe Count Register.

Parameter 2 – Reads the magnetic unit record and aligns the record to posting line 1, the first posting line. The line-find number read from the magnetic record is incremented by one, since it is the number of the last posting line, and loaded into the Stripe Count Register. The unit record may be posted in its current position, or may be aligned to the contents of the Stripe Count Register, or may be aligned after reloading the register.

Parameter 3 – This parameter provides the ability to insert either a magnetic or Register. Since this parameter does not attempt to read the magnetic record, the contents of the Striped Ledger Buffer are not affected.

Parameter 4 – Reads the contents of the magnetic record into the buffer and ejects the unit record.

Parameter 5 – This parameter specifies an auto reader read. the contents of the magnetic unit record are read into the buffer. If the auto reader is turned off, or is not connected to the system, the instruction will change control to the console mechanism and perform a Read and Eject as described for parameter 4.

The B parameter specifies the number of numeric digits which may be entered into the Accumulator during the numeric keyboard phase of the RL instruction. This is a standard keyboard operation, except that the 00,000, Decimal Fraction, RE, C and M keys are not valid.

RL

MUR

When the instruction is initiated, the numeric keyboard indicator is turned on and the number of digits specified by the "B" parameter may be entered. If this number is exceeded, a keyboard error results. The use of the Reset Key will clear the Accumulator and reinitiate the instruction. PK's may be enabled prior to the RL instruction, so the entry may be terminated by depressing an activated PK. The entry may also be terminated by any OCK which will also set the appropriate OCK flag. If the keyboard entry is in error and has been terminated by an OCK, the depressing of the Ready Button will return the system to the Ready Mode. When the system is in the Ready Mode, the use of the Reset Key will reinitiate the RL instruction; however, any PK's that were enabled, when the RL was originally initiated, have been eliminated by the OCK termination.

If the numeric keyboard phase of the RL instruction is terminated by a PK, a jump to some specific subroutine takes precedence.

If the numeric keyboard phase of the RL instruction is terminated by an OCK, the read phase is initiated, and the system idles waiting for the insertion of a unit record. The insertion of a magnetic unit record will execute the read phase of the instruction.

The numeric keyboard phase may also be terminated by the insertion of a magnetic record, without depressing any OCK. It is possible to initiate the RL instruction, enter numeric digits not exceeding the number specified by the B parameter, and insert a unit record which terminates the numeric keyboard phase and initiates execution of the read phase. This type of termination of the keyboard phase resets all OCK flags.

The forms handler is automatically opened during the initiation of the instruction. It is closed by the first print instruction or a close instruction.

In the read phase, data is transferred to the unit record buffer, destroying the prior contents.

If a magnetic record from a previous operation remains in the mechanism when an RL instruction is initiated, the "presence" sensor logic requires that it be removed and reinserted, even if it is intended as the media for the current read operation.

If a read error occurs, either in the console mechanism or the auto reader, the R (read error) flag is set and magnetic record is ejected. All error recovery routines are programmatic for either reader; however, provision is made in the presence sensor logic, for the console mechanism, to allow the unit record to be pushed from the eject position for a programmatic retry of the RL instruction. If a read error occurs when the auto reader has been selected, the unit record must be moved from the stacking hopper back to the feed hopper for the programmatic retry of the RL instruction.

If a Filled Sheet is detected during the execution of an RL instruction, the unit record is automatically ejected and the Filled Sheet Flag is set. Detection of the filled sheet condition, and error recovery, must be programmatic.

RL

MUR

Example 1:

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    | B                 |    |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30                | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| RL       |    |                      |           |    |    | 0  |    |                   |    |    |    |    |    |    |    |    | 4  |    |    |    |    |    |    |    |    |
| EX       |    |                      |           |    |    | S  |    |                   |    |    |    |    |    |    |    |    | R  |    |    |    | 2  |    |    |    |    |
| PKA      |    |                      |           |    |    | 1  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| BRU      |    |                      |           |    |    |    |    |                   |    |    |    |    |    | -  |    | 3  |    |    |    |    |    |    |    |    |    |
| EX       |    |                      |           |    |    | S  |    |                   |    |    |    |    |    |    |    |    | F  |    |    |    | 3  |    |    |    |    |
| PKA      |    |                      |           |    |    | 2  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| TK       |    |                      |           |    |    | 0  |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| BRU      |    |                      |           |    |    |    |    |                   |    |    |    |    |    | -  |    | 2  |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>+/-</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>   |
|----------------|----------|------------|----------|----------|--|
| RL             | 0        |            | 4        |          | INDEX 4-DIGIT NUMBER AND INSERT RECORD   |
| EX             | S        |            | R        | 2        | IF READ ERROR OR JAM   |
| PKA            | 1        |            |          |          | PKA 1—RECONSTRUCT ROUTINE  |
| BRU            |          | -3         |          |          | BRANCH BACK IF "R" FLAG IS SET AND ATTEMPT TO READ AGAIN, UNTIL PKA 1 IS SELECTED. |
| EX             | S        |            | F        | 3        | IF FILLED SHEET  |
| PKA            | 2        |            |          |          | PKA 2—FILLED SHEET ROUTINE   |
| TK             | 0        |            |          |          | HALT FOR ENFORCED PK SELECTION   |
| BRU            |          | -2         |          |          | BRANCH BACK IF "F" FLAG IS SET TO ENSURE DEPRESSION OF PKA 2.                      |

If a Filled Sheet is detected during the posting procedure, that is, if during the posting procedure an OCK or a PK was selected which would advance the magnetic unit record to a line below the last available posting line (or if the Stripe Count Register is incremented to a value of 1 greater than the contents of the Stripe Limit Register), the Filled Sheet Flag is set. (The automatic ejection of the record can be suppressed if desired.) Detection of the filled sheet condition and error recovery, must be programmatic.

RL  
 MUR

Example 2:

| LABEL  |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER  |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|--------|----|----------|----|----|----|----|----------------------|------------|----|----|----|----|----|----|----|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        |    |          |    |    |    |    |                      | A          |    |    |    |    |    |    |    | B                  |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    |    |    |                      | LABEL      |    |    |    |    |    |    |    | + OR -<br>INC. REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16     | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24         | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| INITAL |    | LSLR     |    |    |    |    |                      | 45         |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    |          |    |    |    |    |                      | }          |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | AR       |    |    |    |    |                      | 1          |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | SK       |    |    |    |    |                      | S          |    |    |    |    |    |    |    | F                  |    |    |    | 1  |    |    |    |    |    |    |    |    |    |    |    |
|        |    | BRU      |    |    |    |    |                      | POST       |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | DOC      |    |    |    |    |                      | LINE 46 IS |    |    |    |    |    |    |    | BALANCE            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | DOC      |    |    |    |    |                      | FORWARD    |    |    |    |    |    |    |    | LINE               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | POS      |    |    |    |    |                      | BAL        |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | PA       |    |    |    |    |                      | BALFWD     |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | POS      |    |    |    |    |                      | BALCOL     |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | TSBA     |    |    |    |    |                      | 6          |    |    |    |    |    |    |    | 1                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | PNS-     |    |    |    |    |                      | 4          |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | SRJ      |    |    |    |    |                      | FILLIN     |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |    | BRU      |    |    |    |    |                      | POST       |    |    |    |    |    |    |    |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u>   | <u>+/-</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>              |
|--------------|----------------|------------|------------|----------|----------|-----------------------------|
| INITAL       | LSLR           | 45         |            |          |          | LOAD MAG REC LIMIT REGISTER |
|              |                | }          |            | }        |          | }                           |
|              | AR             | 1          |            |          |          | ALIGN TO NEXT LINE          |
|              | SK             | S          |            | F        | 1        | TEST LAST LINE              |
|              | BRU            | POST       |            |          |          | GO TO POST NEXT LINE        |
|              | DOC            | LINE 46 IS |            |          |          | BALANCE                     |
|              | DOC            | FORWARD    |            |          |          | LINE                        |
|              | POS            | BAL        |            |          |          | POSITION TO BALANCE         |
|              | PA             | BALFWD     |            |          |          | TYPE BALANCE FWD MSG        |
|              | POS            | BALCOL     |            |          |          | POSITION TO BALANCE COLUMN  |
|              | TSBA           | 6          |            | 1        |          | READ BALANCE                |
|              | PNS-           | 4          |            |          |          | PRINT BALANCE               |
|              | SRJ            | FILLIN     |            |          |          | GO TO FILLED SHEET          |
|              | BRU            | POST       |            |          |          | GO POST NEXT ENTRY          |



In this example, when the AR instruction advances the record to line 46 (Limit Register value +1) the Filled Sheet Flag (F) is set and the system displays a notification message and the Balance. The program then jumps to a filled sheet routine for heading up the next record and returns to the correct posting routine.

**2.19.06 PRINT ALPHA FROM MAGNETIC RECORD AREA INSTRUCTION**

|                                       |                |          |
|---------------------------------------|----------------|----------|
|                                       | <u>OP CODE</u> | <u>A</u> |
| PRINT ALPHA FROM MAGNETIC RECORD AREA | PAS            | 1-64     |

The PAS instruction prints alpha characters from the Magnetic Record Read-In Area. The number of characters printed, and their location in the Magnetic Record Buffer, is determined by the format selected by the A parameter.

The PAS instruction is terminated by the printing of the number of characters specified by the selected format, or by the presence of NUL (0,0) codes in the data field.

Example:

| LABEL |    | OP. CODE |    |    |    |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|
|       |    |          |    |    |    |    |                      | A         |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    |    |                      | LABEL     |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
| 16    | 17 | 18       | 19 | 20 | 21 | 22 | 23                   | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |
|       |    |          |    |    |    | L  | S                    | F         | R  |    |    |    | F  | I  | E  | L                 | D  | S  |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    |    |                      |           |    |    |    |    | 2  |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | P  | A                    | S         |    |    |    |    | 1  |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    |    |                      |           |    |    |    |    | 2  |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | W  | R                    | D         |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | F  | I                    | E         | L  | D  | S  |    |    | 1  |    |                   |    |    |    |    |    |    |    | 3  | 1  |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | S  | L                    | F         |    |    |    |    | 6  | 3  |    |                   |    |    |    |    |    |    | 4  |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | S  | L                    | F         |    |    |    |    | 6  | 7  |    |                   |    |    |    |    |    |    | 7  |    |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | S  | L                    | F         |    |    |    |    | 7  | 4  |    |                   |    |    |    |    |    |    | 1  | 1  |    |    |    |    |    |    |    |  |  |
|       |    |          |    |    |    | S  | L                    | F         |    |    |    |    | 8  | 5  |    |                   |    |    |    |    |    |    | 1  | 1  |    |    |    |    |    |    |    |  |  |

|     |     |
|-----|-----|
| ADB | SUB |
|     | MUR |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|--------------|----------------|----------|----------|---|
|              | LSFR           | FIELDS   |          | LOAD STRIPE FORMAT REGISTER   |
|              | PAS            | 1        |          | PRINT 31 ALPHA CHARACTERS FROM THE<br>MAGNETIC RECORD READ-IN AREA. |
|              | WORD           |          |          |   |
| FIELDS       | SLF            | 1        | 31       | 1—ACCOUNT NAME  |
|              | SLF            | 63       | 4        | 2—CHECK COUNT   |
|              | SLF            | 67       | 7        | 3—ACCOUNT NUMBER  |
|              | SLF            | 74       | 11       | 4—BALANCE AND SIGN  |
|              | SLF            | 85       | 11       | 5—LOW MONTHLY BALANCE AND SIGN                                      |

**2.19.07 ARITHMETIC INSTRUCTIONS**

|   | <u>OP .CODE</u> | <u>A</u> | <u>B</u> |
|---|-----------------|----------|----------|
| ADD FROM MAGNETIC RECORD AREA<br>TO ACCUMULATOR   | ADB             | 1-64     | 0-1      |
| SUBTRACT MAGNETIC RECORD AREA<br>FROM ACCUMULATOR | SUB             | 1-64     | 0-1      |

The ADB instruction adds the number of digits specified by the format, which is selected by the A parameter, to the Accumulator.

The B parameter of the ADB instruction, if 0, specifies an unsigned data field; if 1, a signed data field. If the field is signed, the least significant digit contains the sign (all Accumulator flags).

The SUB instruction subtracts the number of digits specified by the format, which is selected by the A parameter, from the Accumulator. The B parameter specifications are identical to those described for the ADB instruction above.

Example: This example utilizes the Stripe Format Table as defined in the PAS instruction example.

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    | B  |    |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| A        | D  | B                    |           |    |    |    | 2  |    |    |    |    |    |    |    |    |    | 0  |    |    |    |    |    |    |    |    |

|      |      |
|------|------|
| TSBA | TSBM |
|      | MUR  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                              |
|----------------|----------|----------|---|
| ADB            | 2        | 0        | ADD 4 DIGITS (UNSIGNED) TO THE ACCUMULATOR. |

**2.19.08 TRANSFER INSTRUCTIONS**

|  | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|--|----------------|----------|----------|
| TRANSFER NUMERIC FROM MAGNETIC AREA TO ACCUMULATOR | TSBA           | 1-64     | 0-1      |
| TRANSFER ALPHA FROM MAGNETIC AREA TO MEMORY        | TSBM           | 1-64     |          |

The TSBA instruction transfers the number of digits specified by the format, which is selected by the A parameter, into the Accumulator.

The B parameter of the TSBA instruction, if 0, specifies that the field is unsigned. If the B parameter is 1, the field is signed. The sign digit is contained in the least significant digit position of the data field defined by the format. It is inserted into the sign position of the Accumulator during the transfer process. The sign digit is considered to occupy a digit position in the field defined by the format. All Accumulator flags (– S C M) will be transferred.

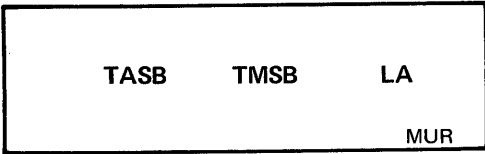
The TSBM instruction transfers the number of alpha characters, specified by the format selected by the A parameter, into memory. An LKBR instruction must precede this instruction, since the value contained is the memory location of the first word of the transfer.

The TSBM instruction is terminated by the transfer of the number of characters specified by the selected format. NUL (0,0) codes will be inserted into memory following the last character of the transfer. If the data does not completely occupy the last word of memory addressed in the transfer process, the balance of the word is filled with NUL (0,0) codes. If data completely fills the last word of memory addressed in the data transfer process, the next sequential memory word is filled with NUL (0,0) codes.

Example 1: This example utilizes the Stripe Format Table as defined in the PAS instruction example.

| OP. CODE | FIELD LEN - GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----------|-----------------|-----------|----|----|----|----|----|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|--|--|--|--|---|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|          |                 | A         |    |    |    |    |    |    |    |    |    |    | + OR - INC/REL |    |    |    |    | B  |    |    |    |    | C  |    |    |  |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|          |                 | LABEL     |    |    |    |    |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 22       | 23              | 24        | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35             | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |  |  |  |  |  |  |  |  |  |   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TSBA     |                 |           |    | 4  |    |    |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|----------------|----------|----------|---|
| TSBA           | 4        | 1        | TRANSFER 11 DIGITS (INCLUDING THE SIGN) TO THE ACCUMULATOR. |



|   | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---|----------------|----------|----------|
| TRANSFER FROM ACCUMULATOR TO<br>MAGNETIC RECORD AREA  | TASB           | 1-64     | 0-1      |
| TRANSFER ALPHA FROM MEMORY TO<br>MAGNETIC RECORD AREA | TMSB           | 1-64     |          |

The TASB instruction transfers the number of digits specified by the format, which is selected by the A parameter, from the Accumulator into a data field in the Magnetic Record Buffer. The location of the data field within the buffer is also specified by the format.

If the B parameter of the TASB instruction is 0, the sign of the Accumulator is ignored. If the B parameter is 1, the sign of the Accumulator is transferred into the least significant digit position of the data field. If the sign is included, it is considered a digit transfer. (All Accumulator flags are transferred.)

The TMSB instruction transfers the number of alpha characters, specified by the format, which is selected by the A parameter, from memory to a data field in the Magnetic Record Buffer. The location of the data field within the buffer is also specified by the format. The memory location of the starting word of the transfer is contained in the Keyboard Base Register. To specify an intended memory location the TMSB instruction must be preceded by an LKBR instruction. The instruction is terminated by transferring the number of characters specified by the selected format or upon recognizing an end of alpha code.

Example 2: This example utilizes the Stripe Format Table as described in the PAS instruction example.

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    | B                 |    |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |                      | LABEL     |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30                | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| TASB     |    |                      | 5         |    |    |    |    |                   |    |    |    |    | 1  |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |                      |           |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>  |
|----------------|----------|----------|---|
| TASB           | 5        | 1        | TRANSFER 11 DIGITS (INCLUDING THE SIGN) FROM THE ACCUMULATOR TO THE MAGNETIC RECORD BUFFER. |

### 2.19.09 UNIT RECORD ALIGNMENT INSTRUCTIONS

The Unit Record Alignment instructions provide the ability to control record movement and alignment in the console mechanism.

|              | <u>OP CODE</u> |
|--------------|----------------|
| RECORD ALIGN | LA             |

The record align instruction provides the ability to move the handling mechanism from its current position to the line number contained in the Stripe Count Register. The Record Alignment Errors, "jam" indications, and error recovery procedures are discussed under subject 2.19.10.

|     |
|-----|
| EL  |
| MUR |

Example 1:

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    |    |    |    |    |    |    |    | B                 |    |    |    |    |    |    | C  |    |    |    |
|          |    |                      | LABEL     |    |    |    |    |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37                | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| LSCR     |    |                      | 45        |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |
| LA       |    |                      |           |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |
|          |    |                      |           |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>             |
|----------------|----------|----------|----------------------------|
| LSCR           | 45       |          | LOAD STRIPE COUNT REGISTER |
| LA             |          |          | ALIGN RECORD TO LINE 45    |

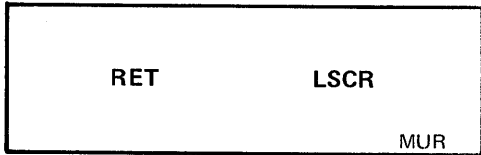
| <u>OP CODE</u> | <u>EL</u> |
|----------------|-----------|
| EJECT RECORD   |           |

The EL instruction ejects the unit record that is in the handling mechanism. This is the only operation performed. The Magnetic Record Buffer is not affected. See subject 2.19.10 for error conditions and recovery procedures. If the handler is closed, it is open for the execution of the EL instruction.

Example 2:

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    |    |    |    |    |    |    |    | B                 |    |    |    |    |    |    | C  |    |    |    |
|          |    |                      | LABEL     |    |    |    |    |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37                | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| EL       |    |                      |           |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |
|          |    |                      |           |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u> |
|----------------|----------|----------|----------------|
| EL             |          |          | EJECT RECORD   |



OP CODE

RETRACT RECORD

RET

The magnetic unit record handler travels down and to the rear of the console until a fixed limit is reached. The RET instruction moves the handler to this fixed limit, with the handler open, to permit the insertion and manual alignment of a record or form. See subject 2.19.10 for error conditions and recovery procedures. The handler will remain retracted until an EL, LA, or RL instruction moves it back to its forward limit. If the handler is in the retracted position when the power is turned on, the power-on routine will move it to the forward position.

Example 3:

| OP. CODE |    | FIELD<br>LEN-<br>GTH |    | PARAMETER |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|----|-----------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |                      |    | A         |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
|          |    |                      |    | LABEL     |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25 | 26        | 27 | 28 | 29 | 30 | 31 | 32                | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| RET      |    |                      |    |           |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |                      |    |           |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>           |
|----------------|----------|----------|--------------------------|
| RET            |          |          | RETRACT RECORD MECHANISM |

| <u>OP CODE</u>             | <u>A</u>  |
|----------------------------|-----------|
| LOAD STRIPE COUNT REGISTER | LSCR 1-46 |

The LSCR instruction loads the Stripe Count Register with the value stored in the "A" parameter. The "A" parameter value may vary from 1 to 46 (46 is maximum number of posting lines on an 11" Magnetic Record).

The Stripe Count Register and the appropriate Forms Count Register are incremented by the AR, ALR, and ARTO instruction if a Magnetic Record is in handler. If a Magnetic Record is not present, only the appropriate Forms Count Register is incremented.

When the Stripe Count Register is incremented one beyond the Stripe Limit Register, the Filled Sheet Flag (F) is set. The Filled Sheet Flag is reset at all other times.

When a Write Magnetic Record (WL) instruction is executed, the contents of the Stripe Count Register are written on the magnetic record in the area reserved for the line-find number.

In a Read and Align operation the contents of the Line-Find number on the magnetic record are incremented by one and stored in the Stripe Count Register.

|      |      |
|------|------|
| LSCR | LSLR |
|      | MUR  |

Example 4:

| OP. CODE |    | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|---------------|-----------|----|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |               | A         |    |    |    |    |    |    | B              |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
|          |    |               | LABEL     |    |    |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24            | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32             | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| LSCR     |    |               | 1         |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |               |           |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                                    |
|----------------|----------|----------|---|
| LSCR           | 1        |          | LOAD THE STRIPE COUNT REGISTER WITH A VALUE OF 1. |

|                            | <u>OP CODE</u> | <u>A</u> |
|----------------------------|----------------|----------|
| LOAD STRIPE COUNT REGISTER | LSCR           | 1-46     |

The LSLR instruction loads the Stripe Limit Register with the value contained in the A parameter.

Example 5:

| OP. CODE |    | FIELD LEN-GTH | PARAMETER |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|---------------|-----------|----|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |               | A         |    |    |    |    |    |    | B              |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |    |
|          |    |               | LABEL     |    |    |    |    |    |    | + OR - INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24            | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32             | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| LSLR     |    |               | 45        |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |    |               |           |    |    |    |    |    |    |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

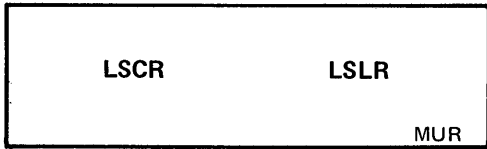
| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                                     |
|----------------|----------|----------|--|
| LSLR           | 45       |          | LOAD THE STRIPE LIMIT REGISTER WITH A VALUE OF 45. |

**2.19.10 RECORD ALIGNMENT ERRORS AND FLAG INDICATIONS**

Record Alignment Errors occur because of the following conditions:

1. The "gripper" jaws in the handler mechanism are not moving or are not at proper speed when the handler has been activated.
2. When the total number of lines, from the line-find operation, plus the number of programmatic line advances, does not equal the number of lines the form moves when it travels back to the limit to prepare for an eject or write operation.

If either of the above conditions occur, a "jam" condition is probable. A jam can also be caused by a torn or accorded form. The jam condition will result in the following indications:



1. The execution of the instruction in process when the alignment error occurred will not be terminated.
2. PKA 1 is enabled. Its indicator and the Error indicator are turned on. All other PK's are disabled. All keyboard indicators, other than PKA 1 and the Error indicator are turned off. The alarm is sounded.

Error recovery consists of clearing the alignment condition or record jam by pressing PKA 1 and by removing the unit record from the handler. The depression of PKA 1 clears the error condition, terminates the execution of the instruction in process when the error occurred, turns off the PKA 1 indicator and the Error indicator, sets both the R and W flags, and returns to sequential execution of the program.

It is essential since the instruction in process when the error occurred was terminated, that the Record Align (AL), Eject Record (EL), Write Record (WL), Read Record (RL), and the line advance instructions (AR, ALR, ARTO) each are followed by flag interrogation instructions to allow programmatic recovery from an error condition.

Example:

| OP. CODE |    | FIELD<br>LEN-<br>GTH | PARAMETER |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----------|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          |    |                      | A         |    |    |    |    |    |    |    | B                 |    |    |    | C  |    |    |    |    |    |    |    |    |    |    |
|          |    |                      | LABEL     |    |    |    |    |    |    |    | + OR -<br>INC/REL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22       | 23 | 24                   | 25        | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33                | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| L        | S  | C                    | R         |    |    |    | 1  |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| R        | L  |                      |           |    |    |    | 0  |    |    |    |                   |    |    |    |    |    | 0  |    |    |    |    |    |    |    |    |
| E        | X  |                      |           |    |    |    | S  |    |    |    |                   |    |    |    |    |    | R  |    |    |    | 2  |    |    |    |    |
| P        | K  | A                    |           |    |    |    | 1  |    |    |    |                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| B        | R  | U                    |           |    |    |    |    |    |    |    |                   |    | 1  |    | 0  |    |    |    |    |    |    |    |    |    |    |

| <u>OP CODE</u> | <u>A</u> | <u>+/-</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>   |
|----------------|----------|------------|----------|----------|--|
| LSCR           | 1        |            |          |          | LOAD STRIPE COUNT REGISTER   |
| RL             | 3        |            | O        |          | NON-READ AND ALIGN RECORD  |
| EX             | S        |            | R        | 2        | IF READ ERROR OR JAM   |
| PKA            | 1        |            |          |          | PKA 1-RECONSTRUCT ROUTINE  |
| BRU            | -3       |            |          |          | BRANCH BACK IF "R" FLAG IS SET AND ATTEMPT TO READ AGAIN, UNTIL PKA 1 IS SELECTED. |



## 2.20 – MESSAGE UNPACKING ROUTINE

### 2.20.01 GENERAL DESCRIPTION

The message unpacking microstring is used for unpacking numeric information after it has been transferred from the Data Communications buffer to the accumulator. The use of this macro, as opposed to a user written routine to accomplish the same results, will on some applications result in a considerable reduction in the time it takes for the TC to process the data. The message unpack macro should be used when the following conditions exist: The number and types of data elements in the message are variable; and like elements in the message are to be grouped for printing, totaling or storing. Up to 32 different numeric elements may be stored.

To use this Macro the Programmer must set up a Position Table and a Storage Area. The element to be unpacked is programmatically transferred to the accumulator from the receive buffer. The last two digits in the accumulator make up the data element code that directs the microstring to a position table which in turn determines the particular word of the storage area to transfer the item to.

### 2.20.02 POSITION TABLE

The position table must occupy words 11-14. Its function is to determine whether or not the contents of the accumulator will be transferred to the storage area and if so, into which word. Each word of the table contains 8 hexadecimal indicator codes, ending in digit positions 0, 2, 4, 6, 8, 10, 12, 14 respectively.

| Digit Position | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Word II        | E  | E  | F  | F  | 0  | 6  | 0 | A | 0 | 4 | 0 | 3 | 0 | 7 | 0 | 2 |

In the position table, each indicator code is referenced by its least significant digit position. For instance, the code in digit position 4 and 5 is referenced by a 4; the code in digits 2 and 3 is referenced by the 2; etc. There are three different types of codes:

1. A code that indicates which word of the storage area the data is to be transferred to. In the diagram, indicator code 6 (the two digit values in digit position 6 and 7) would cause the data to be transferred to word 4 of the storage area. Indicator code 8 would transfer the data to word 10 of the storage area.
2. FF is a code that indicates the numeric data transmitted to the TC is invalid. The Special (S) flag of the accumulator is set by this code. FF would be most useful when first developing and debugging the on-line system.
3. A code of EE indicates that the data in the accumulator is to be ignored. If the central processor sends a fixed format message to all remotes, some of the fields in that message may pertain to only certain remotes and should be ignored by all others. In this type situation, the EE code proves to be most helpful.

### 2.20.03 DATA ELEMENT CODES

After the data element is transferred to the accumulator from the buffer, the last two digits, which make up the data element code supplied by the data center, are in accumulator digit positions 1 and 0. The hexadecimal value in digit position 1 refers to a particular position table word i.e., actual position

table words 11, 12, 13 and 14 are referenced by numbers 0, 1, 2 and 3 respectively. The hexadecimal value in digit position 0 of the accumulator indicates which code of the position table word is to be accessed. For example, to reference indicator code 2 in word 11 of the position table, a data element code of 02 is used.

#### 2.20.04 STORAGE AREA

The storage area starts in word 15. Word 1 of the storage area would be word 15; 2, word 16, etc. The number of areas used in the program is determined by the programmer, up to a maximum of 32 areas or words.

#### 2.20.05 ERROR CONDITIONS

If the microstring detects an error the data will not be stored and the accumulator S flag will be set. The following will result in an error condition:

1. The word designation given in accumulator digit position 1 for the position table is other than 0, 1, 2 or 3.
2. The digit designation given in accumulator digit position 0 of the word in the position table is other than 0, 2, 4, 6, 8, A, C, or E.
3. An illegal indicator code in the position table. The only valid entries are FF, EE and hexadecimal values 1 to 20.

#### 2.20.06 DELIMITER

The delimiter is a character transmitted to the TC which is used to determine the status of the message being transmitted. For example, DC1 may indicate the end of a print line; DC2 may indicate the end of a buffer but not the end of a message; ETX is used to indicate the end of a message. Each delimiter will set its appropriate K or Y flags.

#### 2.20.07 PROGRAMING REQUIREMENTS

The instruction B40B accesses the unpacking routine. The data element is transferred to the accumulator from the buffer by the application program.

Also, the K and Y flags must be reset by the programmer for each data element, since delimiters which set the flags are used to indicate when to stop unpacking and begin to print the message. The following group of instructions demonstrate how the message unpacking routine is used.

| <u>SYM<br/>LOC.</u> | <u>OP<br/>CODE</u> | <u>A<br/>PAR</u> | <u>B<br/>PAR</u> | <u>C<br/>PAR</u> | <u>REMARKS</u>                                       |
|---------------------|--------------------|------------------|------------------|------------------|--|
| NEWFLD              | RST                | K                | 1                |                  | RESET K FLAGS BEFORE MOVING DATA TO THE ACCUMULATOR. |
|                     | TRBA               | 15               |                  |                  |  |
|                     | CODE               | B40B             |                  |                  | ACCESS UNPACKING ROUTINE.                            |
|                     | EX                 | A                | S                | 1                | CHECK FOR INVALID DATA ELEMENT CODE.                 |
|                     | BRU                | ERROR            |                  |                  |  |
|                     | SK                 | K                | 1                | 1                | IF K1 IS SET UNPACKING IS FINISHED.                  |
|                     | BRU                | NEWFLD           |                  |                  | NOT SET SO CONTINUE UNPACKING.                       |
|                     |                    | (PRINT ROUTINES) |                  |                  |  |

When printing from the storage area, it is necessary to examine each word to determine whether or not it contains a numeric value. If an area does contain numeric information, it should be cleared by the programmer after printing.

## 2.21 – TRANSACTION CODE TRANSLATOR

### 2.21.01 GENERAL DESCRIPTION

The Transaction Code Translator is a Firmware Add-On Micro string used for interpreting typewriter keyboard depressions. As a result, a 2-character abbreviation is stored in memory for printing and a transaction code is stored in a designated location of the Accumulator for transmission to the Data Center. The 2-character abbreviation and the transaction code and its location in the accumulator are determined by a table which is stored in main memory. The Translation table can be of any length; however, it must be located entirely within block 0 (words 0-255).

The Transaction Code Translator also provides for the automatic insertion of a predetermined (modular) 2-character abbreviation and transaction code when a key is not depressed in Row 2 of the typewriter keyboard.

The Transaction Code Translator is primarily designed for use with the TC 700 in a financial application environment. However, versions of the Transaction Code Translator are available which are compatible with most GP 300 firmware sets.

### 2.21.02 TRANSLATION TABLE FORMAT:

a. Work Area

The First five words of the table are reserved as a work area and must be located entirely within a Track in Block 0 (words 0-31 of a track). The Five word work area is used in the following manner:

Word 0: The First word of the table must contain the keyboard codes before executing the Translation instruction. (Maximum of 4 codes, one for each typewriter keyboard row.)

Words 1-4: After executing the instruction the 2-character abbreviation for the key depressed is stored in the 4 high order positions of words 1-4 of the 5-word work area. The exact location of the abbreviation is determined by a code stored in the Translation Table entry for the key depressed. (See factor 4 below).

One key from each row can be translated each time the instruction is executed; Multiple key depressions in the same row will cause an error condition.

b. Translation Area:

The Translation area of the table must immediately follow the work area and it can be of any length depending on the number of key codes being translated. Each word in the translation area contains the factors necessary for translating 2 keyboard characters. These factors are as follows:

1. 2-digit hexadecimal USASCII value for key indexed.
2. 4-digit hexadecimal USASCII value of the 2-character abbreviation to be printed.
3. One-digit hexadecimal value (0-F) of transaction code to be stored in the accumulator for later transmission to the data center.
4. One-digit decimal value (0-3) representing the Accumulator digit position where the transaction code is to be stored and the location in the work area where the

TCT

abbreviation codes are stored. The location of the abbreviation code in the work area can be determined by the following chart.

| FACTOR 4<br>VALUE | ACCUMULATION<br>DIGIT POSITION | ABBREVIATION<br>TABLE LOCATION |
|-------------------|--------------------------------|--------------------------------|
| 0                 | 0                              | 1                              |
| 1                 | 1                              | 2                              |
| 2                 | 2                              | 3                              |
| 3                 | 3                              | 4                              |

The above factors are located within a table word in the following manner:

|                | Character 1 |    |    |    |    |    |   |   | Character 2 |   |   |   |   |   |   |   |
|----------------|-------------|----|----|----|----|----|---|---|-------------|---|---|---|---|---|---|---|
| Digit Position | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7           | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Factor No.     |             |    | 2  |    |    | 1  | 4 | 3 |             |   | 2 |   |   | 1 | 4 | 3 |
| (See Above)    |             |    |    |    |    |    |   |   |             |   |   |   |   |   |   |   |

Example:

|                 | Character 1 |          |          |          |          |          |          |          | Character 2 |          |          |          |          |          |          |          |
|-----------------|-------------|----------|----------|----------|----------|----------|----------|----------|-------------|----------|----------|----------|----------|----------|----------|----------|
| Digit Positions | 15          | 14       | 13       | 12       | 11       | 10       | 9        | 8        | 7           | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
| Table Entry     | <u>4</u>    | <u>5</u> | <u>4</u> | <u>8</u> | <u>4</u> | <u>5</u> | <u>1</u> | <u>2</u> | <u>5</u>    | <u>3</u> | <u>5</u> | <u>4</u> | <u>5</u> | <u>7</u> | <u>2</u> | <u>3</u> |
| Value           |             | E        |          | H        |          | E        | 1        | 2        | S           |          | T        |          | W        |          | 2        | 3        |
| Factor No.      |             |          | 2        |          |          | 1        | 4        | 3        |             |          | 2        |          |          | 1        | 4        | 3        |

|                          | Character 1 |  |    |  | Character 2 |  |    |  |
|--------------------------|-------------|--|----|--|-------------|--|----|--|
| Key Depressed            |             |  | E  |  |             |  | W  |  |
| Stored For Print         |             |  | EH |  |             |  | ST |  |
| Transaction Code         |             |  | 2  |  |             |  | 3  |  |
| Digit Position in Accum. |             |  | 1  |  |             |  | 2  |  |

The last entry in the translation table must be 0000FF00. Upon recognizing an FF code, the search routine halts and the results can be processed by the user program.

The microstring searches the table sequentially beginning with the character stored in digit position 15-8 of the first table entry following the work area. The table entries can be in any order within the table. However, since the microstring searches sequentially, the most frequently used entries should be at the beginning of the table.

### 2.21.03 AUTOMATIC CODES

The Transaction Code Translator instruction will automatically insert an SV abbreviation into word 4 of the work area and a transaction code of 1 into Accumulator digit position 3, when a typewriter key in row 2 is not depressed. Also, when a typewriter key in row 4 is not depressed a transaction code of 9 is

stored in Accumulator digit position 1. An automatic abbreviation is not provided. These automatic abbreviations and the transaction codes can be modified as required by the application.

The values which determine what abbreviation and/or transaction codes are to be generated in the absence of a key depression in Row 2 and/or Row 4 (Transaction Code only), are stored within the instruction microstring. Hence, care must be taken to ensure that only the desired values in the microstring are modified.

Since the instruction microstring will be located in various tracks depending on which main memory firmware set is implemented, all memory locations are relative to the base word of the track (first word of the track) in which the microstring is stored. The microstring is stored in the highest available track provided by the main memory firmware set being utilized.

#### **2.21.04 CODE MODIFICATION**

Modification of the various "automatic" codes is accomplished by changing the desired codes using the Memory Modify utility and then punching out the modified firmware using one of the Memory Punch utility routines.

The bit configuration of the desired abbreviation characters to be printed is determined by each character's row (upper bits) and column (lower bits) location in the USASCII chart. The abbreviation characters are stored in memory in the following manner. (Addresses are relative to the base address of the microstring).

##### 1st Print Character

1. The lower 4 bits of the first print character are stored in digit position 6 of word 3.
2. The upper 4 bits of the first print characters are stored in digit position 14 of word 3.

##### 2nd Print Character

1. The lower 4 bits of the second print character are stored in digit position 10 of word 5.
2. The upper 4 bits of the second print character are stored in digit position 2 of word 6.

The transaction code which is stored in the Accumulator when a key in row 2 is not depressed is located in digit position 6 of word 24.

The transaction code which is stored in the accumulator when a key in row 4 is not depressed is located in digit position 10 of word 31.

Example: The Firmware configuration used is:

Main Memory 2-1021-001 (384 words of user memory), CDC-CDV Firmware Add-On. Using this configuration, the transaction code translator microstring would be in words 320-351 (Block 1, Track 2). The base word of the microstring is word 320. (CDC-CDV would occupy words 352 to 383).

The automatic abbreviation to be printed is DR and a transaction code of 4 is to be inserted in the Accumulator. The keyboard row 4 automatic transaction code is to remain the same (9).

TCT

USASCII Column and Row locations are:

D = 4,4      R = 5,4.

The upper and lower digits of the first print character (D) are stored in word 3 digit position 14 & 6 respectfully of the microstring. The actual memory location is word 323. (Base word is 320 + 3 = 323).

The printout of the TC using Memory Modify would be as follows:

|     |   |   |    |      |   |   |    |      |    |           |
|-----|---|---|----|------|---|---|----|------|----|-----------|
| 323 | 2 | 5 | A2 | 4FE2 | F | 3 | E2 | CC38 | 14 | 4A24FE2F4 |
|     | 2 | 4 | A2 | 4FE2 | F | 4 | E2 | CC38 |    |           |

The lower digit (4) of the second print character (R) is located in word 5 digit position 10 (word 325)

Printout:

|     |      |   |   |    |      |      |    |   |
|-----|------|---|---|----|------|------|----|---|
| 325 | 40E2 | 8 | 6 | E2 | AA16 | 9F5B | 10 | 4 |
|     | 40E2 | 8 | 4 | E2 | AA16 | 9F5B |    |   |

The upper digit (5) of the second print character (R) is located in word 6 digit position 2. (word 326)

Printout:

|     |      |      |      |   |   |    |   |   |
|-----|------|------|------|---|---|----|---|---|
| 326 | 7F14 | A140 | B1E1 | D | 5 | A2 | 2 | 5 |
|     | 7F14 | A140 | B1E1 | D | 5 | A2 |   |   |

The Row 2 transaction code is located in word 24 digit position 6. (word 344)

Printout:

|     |      |      |   |   |    |      |   |   |
|-----|------|------|---|---|----|------|---|---|
| 344 | 235B | 5042 | 3 | 1 | E2 | 9751 | 6 | 4 |
|     | 235B | 5042 | 3 | 4 | E2 | 9751 |   |   |

When the modification of the microstring is complete, the new microstring is punched out using one of the Memory Punch utility routines. It is recommended that all firmware extensions which are used in an installation be incorporated on one tape with the main memory firmware set. In the above example this would be accomplished by punching words 320-575 and words 608-1023.

## 2.21.05 ERROR CONDITIONS

The Transaction Code Translator instruction will detect the following two types of errors.

1. No Table entry for the keyboard character depressed.
2. Multiple depressions on the same typewriter keyboard row.

When one of the above errors is detected, the instruction will set all of the accumulator flags.

### 2.21.06 MACHINE CODE FOR TRANSACTION CODE TRANSLATION INSTRUCTION

This instruction is executed by using a machine language code of 104A. This machine language code is incorporated into the object program by use of a CODE psuedo instruction with an A parameter value of 104A.

### 2.21.07 WORD 576

Word 576 of the utility track is used as a link address between the MACRO instruction (104A) and the microstring. Since the location of the microstring is variable, the content of word 576 will also vary depending on the location of the microstring. The content of word 576 for the various possible locations of the microstring can be determined by the following chart.

| MICROSTRING LOCATION |       | CONTENTS OF WORD<br>576 |
|----------------------|-------|-------------------------|
| BLOCK                | TRACK |                         |
| 1                    | 0     | F244 0000 31F1 0000     |
| 1                    | 1     | F344 0000 31F1 0000     |
| 1                    | 2     | F254 0000 31F1 0000     |
| 1                    | 3     | F354 0000 31F1 0000     |
| 1                    | 4     | F264 0000 31F1 0000     |
| 1                    | 5     | F364 0000 31F1 0000     |
| 1                    | 6     | F274 0000 31F1 0000     |
| 1                    | 7     | F374 0000 31F1 0000     |

### 2.21.08 USER PROGRAM REQUIREMENTS

1. Set word 576 during the initialize portion of the user program.

In addition to one track in user memory, the Transaction Code Translator also uses Syllables 1 and 3 of Word 576. Since various Utility Routines also use word 576, the User Program should set word 576 during the initialize phase of the program.

2. Clear words 1 and 3 of the work area. These words must be cleared prior to executing the microstring to ensure that the abbreviation codes from the previous entry are not printed twice. Words 2 and 4 of the table are cleared automatically.
3. Set the keyboard base register (LKBR) to the first word of the work area and enter the keyboard codes to be translated into the first word of the work area using the EAM macro instruction.
4. Execute the Translation Instructions: (Code 104A).

**NOTE: The Code 104A instruction MUST be executed immediately after the EAM instruction.**

5. Test for an error condition (all Accumulator flags set) immediately after executing the instruction.
6. Set the LKBR to the Send Buffer or Work area.

7. Transfer the transaction codes stored in Accumulator into the send buffer or send record area.
8. Print the abbreviation codes stored in words 1-4 of the work area.

**2.21.09 PROGRAMING EXAMPLE**

Transaction Code Translator could be incorporated into the user Program and utilized in the following manner.

| <u>LABEL</u> | <u>INST</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>                     |
|--------------|-------------|----------|----------|------------------------------------|
| INITIL       | CLM         | MCHTOT   |          | CLR DAILY TOTAL                    |
|              | CLM         | OFLNTT   |          | CLR OFF-LINE TOTAL                 |
|              | LPNR        | PMASKS   |          | LD PRT MASK REGISTER               |
|              | LPKR        | PKEYS    |          | LD PK REGISTER                     |
|              | BRU         | START    |          |                                    |
|              | NOTE        |          |          | THE FOLLOWING TABLE IS USED BY THE |
|              | NOTE        |          |          | MICRO-STRING TO TRANSLATE KEYBOARD |
|              | NOTE        |          |          | ENTRIES.                           |
| TABLE        | REG         | 5        |          | 5 WORD WORK AREA                   |
|              | CODE        | 4134     |          | KB=A, TRANS CODE = 4 COL 3         |
|              | CODE        | 4D4F     |          | ABBV = MO                          |
|              | CODE        | 3101     |          | KB = 1, TRANS CODE = 1 COL 0       |
|              | CODE        | 2031     |          | ABBV = 1                           |
|              | CODE        | 5527     |          | KB = U, TRANS CODE = 7 COL 2       |
|              | CODE        | 5452     |          | ABBV = TR                          |
|              | CODE        | 4637     |          | KB = F, TRANS CODE = 7 COL 3       |
|              | CODE        | 5353     |          | ABBV = SS                          |
|              | CODE        | 0000     |          |                                    |
|              | CODE        | 0000     |          |                                    |
|              | CODE        | FF00     |          | END OF                             |
|              | CODE        | 0000     |          | TABLE                              |
| START        | PKA         | 13       |          | ENABLE PK KEYS                     |
| TRANS        | LKBR        | TABLE    |          | SET BASE REG POINTER TO TABLE      |
|              | CLM         | TABLE    | +1       | CLEAR WORK WORD 1                  |
|              | CLM         | TABLE    | +3       | CLEAR WORK WORD 3                  |
|              | EAM         | 4        |          | ENTER KEYBOARD CODES               |
|              | CODE        | 104A     |          | TRANSLATE CODES                    |
|              | EXE         | A        | -SCM2    | TEST FOR INVALID ENTRY             |
|              | ALARM       |          |          | SIGNAL OPERATOR                    |
|              | RST         | A        | -SCM2    |                                    |



TCT

| <u>LABEL</u> | <u>INST</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u>         |
|--------------|-------------|----------|----------|------------------------|
|              | BRU         | TRANS    |          | BRANCH TO RE, ENTER    |
|              | LKBR        | SENBUF   |          | SET SCP TO SENBUF      |
|              | TRAB        | 3        | 0        | STORE TRANSACTION CODE |
|              | AL          | 1        |          | ALIGN FOR M            |
|              | POS         | 10       |          | POSITION PRINTER       |
|              | PA          | TABLE    | +1       | PRINT ABBREVIATION 1   |
|              | POS         | 14       |          | POSITION PRINTER       |
|              | PA          | TABLE    | +2       | PRINT ABBV 2           |
|              | POS         | 18       |          | POSITION PRINTER       |
|              | PA          | TABLE    | +3       | PRINT ABBV 3           |
|              | POS         | 22       |          | POSITION PRINTER       |
|              | PA          | TABLE    | +4       | PRINT ABBV 4           |
| MCHTOT       | REG         | 1        |          | DAILY TOTAL            |
| OFLNTT       | REG         | 1        |          | OFF LINE TOTAL         |
| PKEYS        | BRU         | RECV     |          | PK1-TO PROCESS MSG     |
|              | NOP         |          |          |                        |
|              | BRU         | SEND     |          | TO TRANSMIT MSG        |

# SYMBOLIC PROGRAMING PROCEDURES

## PROGRAM DEFINITION

A program definition is a set of specifications used for the efficient development of the application software needed for a machine-oriented data processing system. The program definition procedure is:

1. Systems Analysis.
2. Defining the output.
3. Defining the processing.
4. Defining the input.
5. Evaluating the system and,
6. Defining for programing – or – reanalyzing and repeating the procedure.

When the program definition procedure is used to design an acceptable system, the system specifications are recorded in the form of:

1. A general systems flow chart of the complete data processing system.
2. Completed Program Definition Worksheets, MKTG 2366, illustrating the required output from each program in the system.
3. Complete Program Definition Charts, MKTG 2402, explaining the input, processing, and output requirements of each program in the system.

The necessary applicational software will then be developed from this information.

## PROGRAM WRITING

After the program definition specifications are completed and given to the programmer, the process of writing the program begins.

The first step the programmer should take, is to thoroughly analyze the program definition specifications. This will serve two basic purposes. First, it will enable the programmer to ask questions about any area or steps in the definition, that are unclear. This can save later reprograming on steps the programmer incorrectly understood. Second, it will give the programmer an opportunity to develop a general idea of what the program will contain when completed, how much memory it is going to take (this evaluation becomes more accurate with experience) and to look for possible use of any routines, already written, which can be used in the program.

After the definition is thoroughly analyzed and all questions answered, the writing of symbolic instruction begins.

Every program generally has three separate sections, initialize, main body, and definition section. Coding forms should be set aside for each section. This enables the programmer to add pages to any section without interrupting the order.

An explanation of each section using the programing example in Section 4 follows.

The initialize portion of a program is generally the shortest portion of a program (in terms of numbers of instructions). In its narrowest sense, this portion will be executed before an NK or TK instruction, halts the internal program execution for the first operator action. In the example Seq. No.'s 20, 30, 40 loads the base register for the PK table, the print mask table, and the line limit register for the form being used in the machine. Even though its instructions are few in number, without them the programmer could not control the program. For example program execution stops at Seq. 90, if the operator selected PKA 5 without having the LPKR instruction at Seq. No. 20, the base register for the PK Table would contain the word number for the LPKR instruction of the previous program in the machine. Therefore selecting PKA 5 would not have caused the execution of the BRU INCOST instruction.

A broader description of the initialize section would be to include routines in the program which are not part of the main program. Seq. steps 1 through 5 on the Program Definition Chart in Section 4 could be included under this broader definition. These sequence steps are not concerned with the mainline function, i.e., creating the invoice, but rather prepare the system for invoice writing.

The second section of a program, the main body, is the area of the program which accomplishes the task assigned to the program. In the programing example, sequence steps 6 through 32, are concerned with creating an invoice. Each sequence step should be completely programmed before going to the next. In the example, sequence steps 8 through 14, are accomplished by Sequence Numbers 430 to 570. Since these sequence steps are concerned with the ribbon line on the invoice, the programmer has labeled Sequence No. 430 RIBBON. The use of descriptive labels gives the program added readability. This enables others who read the program documentation to follow the logic with a better understanding. Using the REMARKS field on each instruction to explain the purpose of the instruction also increases the readability of a program. These comments in the REMARKS field also help the programmer when debugging the program.

While programing the sequence steps from the Program Definition Chart, the programmer will generally make use of three techniques, straight line, loops, and subroutines. The straight line method is exactly as its name implies, it is a series of instructions, without any branches which solves the given problem. Sequence numbers 110 through 230 are an example of this method. This sequence accomplishes the task of storing the page number, positioning the printer, printing the customer name, storing it, advancing the form, etc., without the use of loops or subroutines. The looping technique uses a counter to execute the same series of instructions a desired number of times. The routine which clears 11 words of memory labeled CLRMEN uses the looping technique. An index register value is incremented each time the loop is executed, up to a maximum number of times, when this limit is reached the program branches out of the loop. The subroutine technique is like the straight line method except that in the series of instructions we branch out to execute another series of instructions and when finished with these the program returns to the instruction following where we left the series. This allows writing a routine, which is to be executed a number of times during a program, only once; and going to it any time and returning to where it branched from. An example is sequence number 560 where we leave the straight line to print the date and invoice number and when finished, return to sequence number 570.

The last section of the program, the define section, is actually written along with the initialize and main body. This area contains all PK Tables, Print Masks, storage regions, numeric constants, alpha constants, etc. An example of how this section is completed would be to look at Sequence Number 30. The LPNR instruction has in its A parameter the label MASKTB. Right after this instruction is written, the programmer codes the first MASK instruction with the label MASKTB in the definition section. This process is repeated for all storage locations, numeric constants, alpha descriptions, etc., as the program is written.

After the program is written, the last step is to assemble it and debug the program when it is loaded in the machine.

## **PROGRAM DEBUGGING**

Generally, program debugging is completed in two steps. The first step is to correct Assembler errors, these are invalid conditions which the Assembler finds in the symbolic instructions, these errors are corrected by removing the invalid conditions in the symbolic instructions. The second step is to find the logic errors, i.e., areas of the program which are not giving the desired results.

When the Assembler detects an error in the source program, the invalid instruction is replaced by a NO-OP instruction. Thus the object program contains the correct instructions and the Assembler inserted NO-OP's. It is possible to load the object program and replace the NO-OP's with the correct machine language code for the desired instruction, through the use of the Memory Modify service routine.

Logic errors can be found by analyzing the sequence of instructions or by using one of three available Trace service routines. When a logic error is found, its proposed solution should be tested before re-assembly. This is accomplished by inserting the appropriate machine language codes for the symbolics in place of the incorrect codes. If the new solution cannot be placed within the area of the incorrect codes, a branching out of that area to an area not used by the program (usually starting at the word location following the last word of the program) placing the rest of codes and then branching back into the program at the appropriate place. If the new solution is correct, then it can be written in symbolics and inserted in the program before re-assembly. Once debugging is completed, the corrected program can be obtained, by the Punch from Memory service routine.

As mentioned before, during debugging the Trace routines will sometimes be used. In general they are useful for (1) reading the program execution sequence (especially for conditional branches), (2) to check when the flags are being set or reset, (3) to read the values of the index registers (especially when used as counters in loops), (4) to read the value in the Accumulator (to debug shift and arithmetic instructions).

## **DATA COMM DEBUGGING**

Debugging a TC 500 on-line program can be expensive if a central processor remains on demand while the TC 500 operator is detecting and correcting errors on the TC. It is possible to debug off-line by using the memory modify utility routine, especially the selective start feature.

The first word of the receive buffer in Data Comm Memory is located in word 1247. The second word is 1216 and the remaining words follow serially to word 1246. Knowing this, it is possible to access these words using memory modify and index from the keyboard the USASCII code representation of the characters of any message the operator is anticipating, thus doing the work of data comm memory by placing the message in the receive buffer. Then, using the selective start feature of memory modify, access the word and syllable of the instruction immediately after the receive flag (R2) has been interrogated and determined to be set. The object program will begin executing from that word and syllable. This routine allows the operator to proceed as if a message had been received from the central processor and allows testing of those parts of the object program that unpack messages.

Likewise, the transmission of messages can be tested off-line. The first word of the transmit buffer is located in word 1249 and the next 30 words proceed serially to word 1279. The last word is 1248. After programmatically packing a message into the transmit buffer, the operator should depress the

program halt button after the transmit ready flag (R3) is set (evidenced by the transmit ready light being on) and then use memory modify to read these words and determine if the message was assembled in the buffer correctly.

The word locations of transmit or receive record work areas are determined by the Assembler and would be accessed accordingly.

### **MODIFICATIONS NECESSARY TO THIS MANUAL FOR PROGRAMING THE 40 TRACK STYLE SERIES L**

Previously presented information in this manual applies only to 32 Track Styles of the Series L except for Assembler VI which utilizes the 40 track styles of the Series L. This section details all the additional information needed to utilize this assembler manual when programing the Extended Memory Styles.

An object program which was assembled for a 32 track system will operate on a 40 track system using 40 track firmware, except for the REM instruction. An object program which was assembled for a 40 track system will operate only on a 40 track system.

#### **GP 300 OPERATION CODE MODIFICATIONS**

Forty track systems allow the use of any GP 300 instruction explained in this manual except for the Data Communications Message Handling instructions. All user memory may contain program data or any other desired data. However, certain instructions do not permit referencing memory locations above word 511. These instructions are listed in Table 1 below:

| INSTRUCTIONS |
|--------------|
| ADA          |
| CLM          |
| CPA          |
| DIV          |
| MUL          |
| MULR         |
| SUA          |
| XA           |

Table 1

Instructions which only can reference words 0 to 511 of user memory

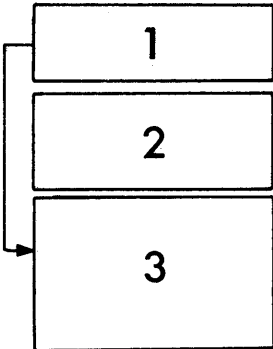
It is essential that the instructions contained in Table 1 be borne in mind when moving or accumulating data in memory. Generally, the machine language codes are the same for either 32 track or 40 track systems. Examine Appendix B, for the machine language codes of both 32 track and 40 track systems.

#### **PROGRAMING CONSIDERATIONS**

Due to the fact that some instructions cannot reference user memory locations above word 511, it is necessary that all constant data and working data be assembled in memory locations below word 511. The remaining memory is then used for program instructions.

The following example illustrates a generally used programing principle

Example:



The three rectangles above illustrate a technique to have the working and storage area of the program assembled below memory word 511.

Rectangle 1 represents word 0. The first three syllables (0, 1, 2) contain programing. Syllable 4 contains a branch around rectangle 2 to rectangle 3.

Rectangle 2 contains the working-storage area.

Rectangle 3 contains further programing as required for data manipulation.

The following sample program illustrates the technique described above.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|--------------|----------------|----------|----------|----------|
|              | LLLR           | 35       |          |          |
|              | LRLR           | 15       |          |          |
|              | LPKR           | PKEYS    |          |          |
|              | BRU            | BEGIN    |          |          |
|              | }              | }        |          |          |
| TOTALS       | REG            | 200      |          |          |
| ZERO         | NUM            | 0        |          |          |
| STORE        | REG            | 150      |          |          |
| BEGIN        | NK             | 5        | 1        |          |

With the expanded memory size it may become necessary to clear a memory area larger than 255 words. This cannot be accomplished, easily, in a single loop since Index Registers have a maximum value of 255.

|   |
|---|
| <b>SYMBOLIC<br/>PROGRAMING PROCEDURES</b> |
|---|

The following technique is recommended:

| <u>OP CODE</u> | <u>A</u> | <u>+/- INCREMENT</u> | <u>B</u> | <u>C</u> |
|----------------|----------|----------------------|----------|----------|
| LIR            | 1        |                      | 0        |          |
| MOD            | 1        |                      |          |          |
| CLM            | TOTAL    |                      |          |          |
| MOD            | 1        |                      |          |          |
| CLM            | TOTAL    | + 200                |          |          |
| IIR            | 1        |                      | 199      |          |
| SK             | T        |                      | I        | 1        |
| BRU            |          | - 6                  |          |          |

The above programing clears 400 words of memory beginning with the word number referenced by TOTAL.

Example:

This example illustrates a method to reference an array of memory larger than 255 words. Controlling such an array of memory must be accomplished by examining the indexing value and changing the base address for values over 255.

Problem: Accumulate sales by 500 product codes (in words 1 to 500).

The programing segment below utilizes the fact that Index Registers have a capacity of 255. When a value transferred to an Index Register exceeds 255, only the difference between that value and 256 remains in the Index Register.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>     |
|--------------|----------------|----------|----------|----------|--------------------|
|              | SRJ            | CLEAR    |          |          |                    |
|              | LPNR           | PMASKS   |          |          |                    |
|              | BRU            | BEGIN    |          |          |                    |
| TOTAL 1      | REG            | 255      |          |          |                    |
| TOTAL 2      | REG            | 245      |          |          |                    |
| BEGIN        | AL             | 1        |          |          |                    |
|              | POS            | 10       |          |          |                    |
|              | NK             | 3        | 0        |          | Enter Product Code |
|              | SKL            | 2        | 5        | 2        | Valid Code 0-499   |
|              | ALARM          |          |          |          |                    |
|              | BRU            |          | -3       |          |                    |
|              | TRM            | CODE     |          |          | Store Valid Code   |
|              | PN             | 2        | 0        |          | Print Code         |
|              | POS            | 16       |          |          |                    |
|              | NKR            | 8        | 0        |          | Enter Amount       |
|              | PNS-           | 7        | 1        |          | Print Amount       |
|              | TRM            | AMOUNT   |          |          | Store Amount       |
|              | TRA            | CODE     |          |          |                    |
|              | TAIR           | 1        |          |          |                    |

|   |
|---|
| <b>SYMBOLIC<br/>PROGRAMING PROCEDURES</b> |
|---|

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u>      |
|--------------|----------------|----------|----------|----------|---------------------|
|              | SUA            | LIMIT    |          |          | Compare Code to 256 |
|              | EX             | A        | -        | 4        |                     |
|              | TRA            | AMOUNT   |          |          | Under 256           |
|              | MOD            | 1        |          |          |                     |
|              | ADM            | TOTAL 1  |          |          |                     |
|              | BRU            | BEGIN    |          |          |                     |
|              | TAIR           | 1        |          |          | Reset I.R.          |
|              | TRA            | AMOUNT   |          |          |                     |
|              | MOD            | 1        |          |          |                     |
|              | ADM            | TOTAL 2  |          |          | Use Base of 257     |
|              | BRU            | BEGIN    |          |          |                     |



GP 300 INSTRUCTIONS TO MACHINE LANGUAGE

Appendix B provides the hexadecimal machine language code for each GP 300 instruction and an alphabetical listing for the GP 300 instruction.

Table C of Appendix B provides hexadecimal values for decimal numbers between 0 and 767.

If a table is not available, the desired value (for values between 0 and 255) may be calculated with the following chart and two simple procedures.

If the value is between 0-15, convert by this chart:

|   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |             |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|-------------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | decimal     |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A  | B  | C  | D  | E  | F  | hexadecimal |

1. Decimal to hexadecimal conversion:

Divide the decimal value by 16.

Insert the hexadecimal equivalent from above chart as the left most digit of the two digit number.

Insert the hexadecimal equivalent of the remainder as the right most digit of the two digit number.

Example: What is the hexadecimal equivalent of the decimal value 255?

$$255 \div 16 = 15 \text{ with } 15 \text{ remainder}$$

So 255 must be represented as FF in hexadecimal notation.

2. Hexadecimal to decimal conversion:

Multiply the decimal equivalent of the left most digit of the two digit hexadecimal number by 16.

Add the decimal equivalent of the hexadecimal value in the right most position to the previous sum.

Example: What is the hexadecimal value 2A equivalent to in decimal notation?

$$2 \times 16 = 32$$

A = 10 (see above chart)

$$32 + 10 = 42$$

The machine language code for a GP 300 instruction consists of 4 hexadecimal digits. These digits are identified as Op Code Upper, Op Code Lower, Parameter Upper, and Parameter Lower.

Example:

The machine language code for AL 5 is ED05.

E is the Op Code Upper.

D is the Op Code Lower.

0 is the Parameter Upper.

5 is the Parameter Lower.

In some cases the Op Code Lower is incremented by 1 for word locations above 255 (See Table A).

APPENDIX B (cont'd)

| INSTRUCTION                           | OP CODE | A                                  | B                   | C   |
|---------------------------------------|---------|------------------------------------|---------------------|-----|
| Add to Accumulator                    | ADA     | LABEL                              |                     |     |
| Add to Index Register                 | ADIR    | 1-4                                | 0-255               |     |
| Add Constant to Accumulator           | ADK     | 0-14                               | 0-9                 |     |
| Add to Memory                         | ADM     | LABEL                              |                     |     |
| Advance Left Platen                   | AL      | 0-255                              |                     |     |
| Alarm                                 | ALARM   |                                    |                     |     |
| Advance both platens                  | ALR     | 0-255                              |                     |     |
| Advance left platen to                | ALTO    | 1-255                              |                     |     |
| Select Alternate Stacker              | ALTP    |                                    |                     |     |
| Advance right platen                  | AR      | 0-255                              |                     |     |
| Advance right platen to               | ARTO    | 1-255                              |                     |     |
| Branch unconditionally                | BRU     | LABEL                              |                     |     |
| Close forms handler                   | CC      |                                    |                     |     |
| Check Digit Compute                   | CDC     | 1-15                               | 0-9                 |     |
| Check Digit Verify                    | CDV     | 1-15                               | 0-9                 |     |
| Change Flags                          | CHG     | A,K,R,P,<br>X,Y                    | 1,2,3,4,-,<br>S,C,M |     |
| Clear Accumulator and insert constant | CLA     | 0-15                               | 0-15                |     |
| Clear Memory Word                     | CLM     | LABEL                              |                     |     |
| Compare alphanumeric                  | CPA     | LABEL                              |                     |     |
| Decrement Index Register              | DIR     | 1-4                                | 0-255               |     |
| Divide                                | DIV     | LABEL                              |                     |     |
| Duplicate thru column                 | DUP     | 1-80                               |                     |     |
| Enter Alpha into Memory               | EAM     | LABEL                              |                     |     |
| Execute if any Flag                   | EX      | }A,T,K,P,V,B,L }<br>}D,R,X,Y,W,S } | -SCMWRF<br>1234OLIU | 1-4 |
| Execute if every Flag                 | EXE     |                                    |                     | 1-4 |
| Execute if digit less than constant   | EXL     | 0-15                               | 0-15                | 1-4 |
| Execute if Accumulator zero           | EXZ     | 1-4                                |                     |     |
| Increment Index Register              | IIR     | 1-4                                | 0-255               |     |
| Insert Constant in Accumulator        | INK     | 0-15                               | 0-15                |     |
| Load Index Register                   | LIR     | 1-4                                | 0-255               |     |
| Load Memory from Card                 | LCD     | 0-255                              |                     |     |
| Load Card Format Register             | LCFR    | LABEL                              |                     |     |
| Load Keyboard Base Register           | LKBR    | LABEL                              |                     |     |
| Load Left Count Register              | LLCR    | 0-255                              |                     |     |
| Load Left Limit Register              | LLLR    | 0-255                              |                     |     |
| Load Flags                            | LOD     | A,K,R,P,<br>X,Y                    | 1,2,3,4,-,<br>S,C,M |     |
| Load Program Key Base Register        | LPKR    | LABEL                              |                     |     |
| Load Print Numeric Base Register      | LPNR    | LABEL                              |                     |     |
| Load Right Count Register             | LRCR    | 0-255                              |                     |     |
| Load Right Limit Register             | LRLR    | 0-255                              |                     |     |
| Load Shift Register                   | LSR     | 0-15                               |                     |     |

| INSTRUCTION  | OP CODE | A   | B   | C   |
|--|---------|---|---|-----|
| Load Punch Count Register                                | LXC     | 1   |   |     |
| Load Punch Count Register                                | LXC     | 0-255                                     |   |     |
| Modify by Index Register                                 | MOD     | 1-4                                       |   |     |
| Multiply   | MUL     | LABEL                                     |   |     |
| Multiply and Round                                       | MULR    | LABEL                                     |   |     |
| Numeric Keyboard   | NK      | 0-15                                      | 0-15  |     |
| Numeric Keyboard Permit C, M Keys                        | NKCM    | 0-15                                      | 0-15  |     |
| Numeric Keyboard Permit Reverse Entry                    | NKR     | 0-15                                      | 0-15  |     |
| Numeric Keyboard Permit Reverse Entry,<br>C, M Keys      | NKRCM   | 0-15                                      | 0-15  |     |
| No-Operation   | NOP     |   |   |     |
| Open forms transport                                     | OC      | 0-255                                     |   |     |
| Print Alphanumeric                                       | PA      | LABEL                                     |   |     |
| Print Character  | PC      | CHARACTER                                 |   |     |
| Print Character if Accumulator plus,<br>Previous Ribbon  | PC+     | CHARACTER                                 |   |     |
| Print Character if Accumulator minus,<br>Previous Ribbon | PC-     | CHARACTER                                 |   |     |
| Print Character Previous Ribbon                          | PCP     | CHARACTER                                 |   |     |
| Enable Program Key Group A                               | PKA     | 1-8                                       |   |     |
| Enable Program Key Group B                               | PKB     | 1-8                                       |   |     |
| Enable Program Key Group C                               | PKC     | 1-8                                       |   |     |
| Print Numeric  | PN      | 0-14                                      | 0-15  |     |
| Print Numeric Shift Ribbon if plus                       | PNS+    | 0-14                                      | 0-15  |     |
| Print Numeric Shift Ribbon if minus                      | PNS-    | 0-14                                      | 0-15  |     |
| Load Position Register                                   | POS     | 1-255                                     |   |     |
| Read Card  | RCD     |   |   |     |
| Enter Alpha into memory, punch non-print                 | REAM    | 0-150                                     |   |     |
| Release Media Clamp                                      | REL     |   |   |     |
| Transfer Remainder to Accumulator                        | REM     |   |   |     |
| Read Numeric into Accumulator                            | RNK     | 0-15                                      | 0-15  |     |
| Red Ribbon   | RR      |   |   |     |
| Reset Flags  | RST     | A, K, R, P,<br>X, Y, L, D                 | 1, 2, 3, 4, -,<br>S, C, M                       |     |
| Read Alpha and Print                                     | RTK     | 0-255                                     |   |     |
| Read Alpha into Memory and Punch,<br>non-print           | RXEAM   | 0-255                                     |   |     |
| Read Alpha, print and punch                              | RXTK    | 0-255                                     |   |     |
| Read Alpha into memory, print and punch                  | RXTKM   | 0-255                                     |   |     |
| Set Flags  | SET     | A, K, R, P,<br>X, Y, L, D                 | 1, 2, 3, 4, -,<br>S, C, M                       |     |
| Skip if any Flag   | SK      | A, K, R, P<br>S, X, Y, B, L<br>T, V, D, W | 1, 2, 3, 4, -, W<br>S, C, M, R<br>O, C, I, U, F | 1-4 |

APPENDIX B (cont'd)

| INSTRUCTION                                      | OP CODE | A                                | B                                   | C   |
|--|---------|----------------------------------|-------------------------------------|-----|
| Skip if every Flag                               | SKE     | A,K,R,P,T,<br>X,Y,L,B<br>D,V,W,S | 1,2,3,4,-,W<br>S,C,M,R<br>O,C,I,U,F | 1-4 |
| Skip if digit less than Constant                 | SKL     | 0-15                             | 0-15                                | 1-4 |
| Skip to card column                              | SKP     | 1-80                             |                                     |     |
| Skip if Accumulator Zero                         | SKZ     | 1-4                              |                                     |     |
| Shift Off  | SLRO    | 0-14                             | 0-14                                |     |
| Shift Off with Sign                              | SLROS   | 0-15                             | 0-15                                |     |
| Subroutine Jump                                  | SRJ     | LABEL                            |                                     |     |
| Subroutine Return                                | SRR     | 1-4                              |                                     |     |
| Stop   | STOP    |                                  |                                     |     |
| Subtract from Accumulator                        | SUA     | LABEL                            |                                     |     |
| Subtract Constant from Accumulator               | SUK     | 0-14                             | 0-9                                 |     |
| Subtract from Memory                             | SUM     | LABEL                            |                                     |     |
| Transfer Accumulator to Index Register           | TAIR    | 1-4                              |                                     |     |
| Type   | TK      | 0-255                            |                                     |     |
| Type into Memory                                 | TKM     | 0-255                            |                                     |     |
| Transfer to Accumulator                          | TRA     | LABEL                            |                                     |     |
| Transfer Card Field to Accumulator<br>as Numeric | TRCA    | 1-16                             |                                     |     |
| Transfer Card Column to Memory as<br>Alpha       | TRCM    | 1-16                             |                                     |     |
| Transfer to Memory                               | TRM     | LABEL                            |                                     |     |
| Punch Alpha from Memory, Non-Print               | XA      | LABEL                            |                                     |     |
| Punch Feed Codes                                 | XB      | 0-255                            |                                     |     |
| Punch Alpha from Card Read Area,<br>Non-Print    | XBA     | 1-16                             |                                     |     |
| Punch Code                                       | XC      | 0-15                             | 0-15                                |     |
| Enter Alpha into Memory and Punch,<br>Non-Print  | XEAM    | LABEL                            |                                     |     |
| Modify by Punch Count Register                   | XMOD    |                                  |                                     |     |
| Punch Numeric, Non-Print                         | XN      | 0-14                             | 0-15                                |     |
| Print Alpha and Punch                            | XPA     | LABEL                            |                                     |     |
| Print and Punch Alpha from Card<br>Read Area     | XPBA    | 1-16                             |                                     |     |
| Print and Punch Numeric                          | XPN     | 0-14                             | 0-15                                |     |
| Print and Punch Numeric Shift Ribbon<br>if Plus  | XPNS+   | 0-14                             | 0-15                                |     |
| Print and Punch Numeric Shift Ribbon<br>if Minus | XPNS-   | 0-14                             | 0-15                                |     |
| Type Punch and Print                             | XTK     | 0-255                            |                                     |     |
| Type to Memory Punch and Print                   | XTKM    | 0-255                            |                                     |     |

## DATA COMMUNICATION INSTRUCTIONS

| INSTRUCTIONS                                    | OP CODE | A                 | B      | C   |
|---|---------|-------------------|--------|-----|
| Change Flags                                    | CHG     | R                 | 23     |     |
| Execute if any Flag                             | EX      | RBD               | 1234   | 1-4 |
| Execute if every Flag                           | EXE     | RBD               | 1234   | 1-4 |
| Increment Receive Character Pointer             | IRCP    | 0-255             |        |     |
| Load Flags                                      | LOD     | R                 | 23     |     |
| Load Polled Flags Register                      | LPF     | -                 |        |     |
| Load Receive Address Register                   | LRA     | -                 |        |     |
| Load Receive Buffer Register                    | LRBR    | LABEL or<br>BLANK |        |     |
| Load Expected Group Transmission Number         | LGN     |                   |        |     |
| Load Expected Broadcast Transmission Number     | LBN     |                   |        |     |
| Load Send Address Register                      | LSA     | -                 |        |     |
| Load Send Transmission Number                   | LSN     | -                 |        |     |
| Load Expected Transmission Number Register      | LTN     | -                 |        |     |
| Power Off                                       | OFF     |                   |        |     |
| Print Alpha from Receive Buffer                 | PAB     | 0-150             |        |     |
| Set Receive Character Pointer                   | RCP     | 1-255             |        |     |
| Retrieve Expected Broadcast Transmission Number | RBN     |                   |        |     |
| Retrieve Polled Flags                           | RPF     | -                 |        |     |
| Retrieve Expected Group Transmission Number     | RGN     |                   |        |     |
| Retrieve Character Pointer Register             | RPR     | -                 |        |     |
| Retrieve Receive Address                        | RRA     | -                 |        |     |
| Retrieve Send Address                           | RSA     | -                 |        |     |
| Retrieve Send Transmission Number               | RSN     | -                 |        |     |
| Reset Flags                                     | RST     |                   |        |     |
| Retrieve Header Transmission Number             | RTH     | -                 |        |     |
| Retrieve Expected Transmission Number           | RTN     | -                 |        |     |
| Set Send Character Pointer                      | SCP     | 1-255             |        |     |
| Set Flags                                       | SET     | R                 | 23     |     |
| Skip if Flag                                    | SK      | RBD               | 1234   | 1-4 |
| Skip if every Flag                              | SKE     | RBD               | 1234   | 1-4 |
| Transfer Accumulator to Send Record Area        | TRAB    | 0-15              | 0 or 1 |     |
| Transfer Receive Buffer                         | TRB     | LABEL             |        |     |
| Transfer to Accumulator as Numeric              | TRBA    | 0-16              |        |     |
| Transfer Character                              | TRCB    | 0-15              | 0-15   |     |
| Transfer Alpha                                  | TRF     | 0-255             |        |     |
| Transfer Send Record Area                       | TSB     | LABEL             |        |     |

## ASSEMBLER PSEUDO INSTRUCTIONS

| OP CODE | A                    | B    | FUNCTION   | SUBJECT REFERENCE |
|---------|----------------------|------|--|-------------------|
| ADVL    | 1-4                  |      | To space the Assembler output form the number of lines specified in the A parameter.                                     | 2.01.01           |
| ALF     |                      |      | To store alphanumeric constants.   | 2.01.02           |
| CDB     |                      |      | To reserve words 1-10 as card read-in buffer, automatic branch to word 11, syllable 0.                                   | 2.01.03           |
| CDF     | 1-80                 | 1-80 | A parameter indicates the beginning card column of a field, B parameter defines the number of card columns in the field. | 2.01.04           |
| CODE    | 4 hexadecimal digits |      | To allow insertion of 4 hexadecimal digits into a syllable of memory.  | 2.01.05           |
| DEF     | O-N *                |      | To assign a value to a label.  | 2.01.06           |
| DEFT    | 0-15                 | 0-15 | To assign a value to a label in both A and B fields.   | 2.01.06           |
| DOC     |                      |      | For documentation when assembling on B 2500, B 3500, B 5500 49 characters beginning in column 29.                        | 2.01.07           |
| END     |                      |      | To terminate the Assembler program, the last line of code in the program.  | 2.01.09,          |
| EQU     |                      |      | To equate the label in label field to the label in the A parameter.  | 2.01.10           |
| ESTB    |                      |      | To reserve 32 words in high order memory for receive, send buffer.   | 2.01.08           |
| MASK    |                      |      | Allow entry of mask word 24 print format characters are accepted.  | 2.01.11           |
| NOTE    |                      |      | For documentation will allow entry of 25 characters, beginning in column 53.   | 2.01.12           |
| NUM     |                      |      | To store numeric constants.  | 2.01.13           |
| ORG     | O-N**                |      | To assemble the instruction following ORG in the word location specified in the A parameter.                             | 2.01.14           |

**APPENDIX C (cont'd)**

| <b>OP CODE</b> | <b>A</b> | <b>B</b> | <b>FUNCTION</b>   | <b>SUBJECT REFERENCE</b> |
|----------------|----------|----------|---|--------------------------|
| PAGE           |          |          | To space Assembler output to the first line of the next page.                         | 2.01.15                  |
| REG            | 1-255    |          | To reserve the number of words specified in the A parameter for working-storage.      | 2.01.16                  |
| WORD           |          |          | To cause the Assembler to assign the next instruction in syllable O of the next word. | 2.01.17                  |

\*The upper limit is variable depending upon which Operation Code the label will be used.

\*\*The upper limit is variable depending upon the amount of user memory.

## TABLES OF MASK CODES

TABLE E-1 MASK CONTROL CODES

| CONTROL CODES | PRINT FUNCTION          | PUNCH FUNCTION   |
|---------------|-------------------------|--|
| F             | Print \$                | No effect  |
| +             | Suppress Punctuation    | No effect  |
| P             | No effect               | Leading zeros punch in P flag set blank card column in 80 column card if P flag reset. |
| —             | Print Condensed Numeric | Monetary punctuation prints without causing printer escapement. Requires PIP hardware. |

TABLE E-2 MASK FLAG CODES

| FLAG CODES | PRINT FUNCTION  | PUNCH FUNCTION   |
|------------|---|--|
| D          | Print digit regardless of significance.                                   | Punch character regardless of significance.  |
| D,         | Print digit and comma regardless of significance.                         |  |
| .D         | Print decimal point and digit regardless of significance.                 |  |
| D:         | Print digit and decimal point regardless of significance.                 |  |
| X          | Suppress Terminal Zeros   |  |
| .X         | Decimal point and terminal zero suppression.                              |  |
| C          | Units of cents leading and terminal zero suppress.                        |  |
| .C         | Tenths of cents decimal point with leading and terminal zero suppression. |  |
| Z          | Leading zero suppression.   | Punch if:<br>1. P is set.<br>2. Accumulator digit not zero.<br>3. A non-zero digit has been punched. |
| Z,         | Leading zero suppression and comma.                                       |  |
| Z:         | Leading zero suppression and decimal point.                               |  |



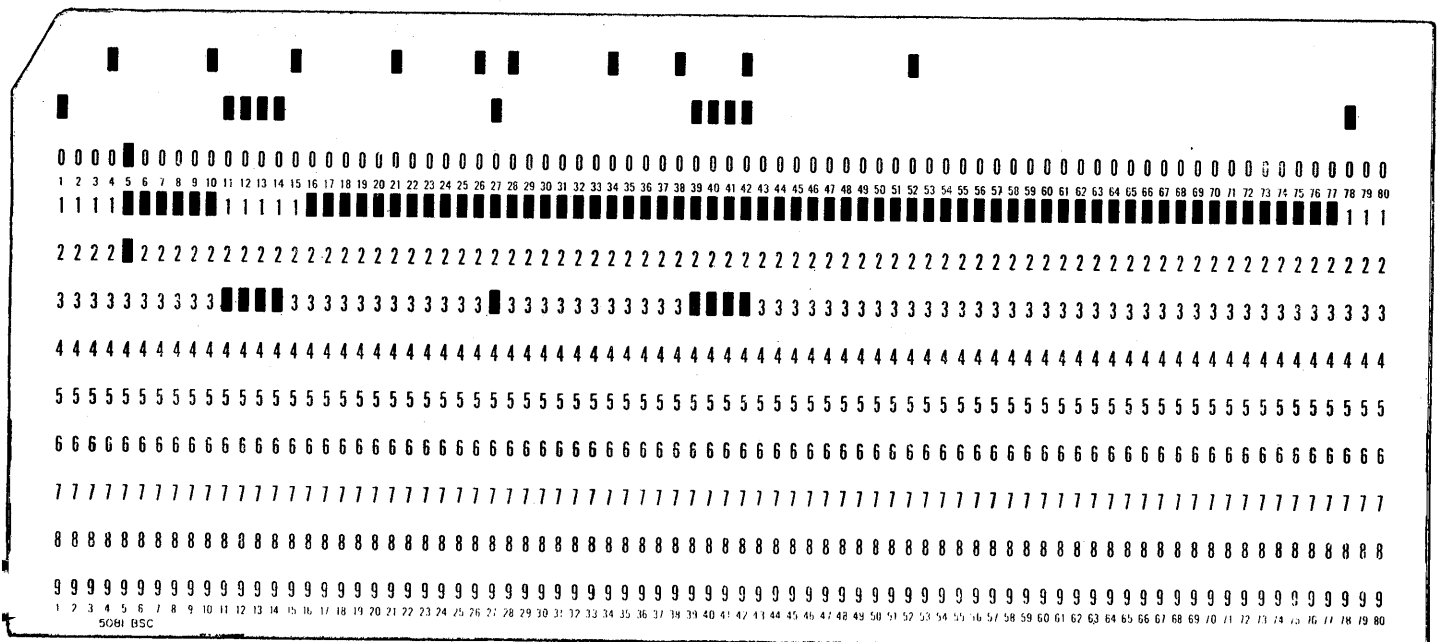
APPENDIX E (cont'd)

| FLAG CODES | PRINT FUNCTION                            | PUNCH FUNCTION       |
|------------|---|----------------------|
| S          | Print only if Accumulator digit non-zero. |                      |
| I          | Ignore digit                              | Ignore               |
| E          | Terminate, non-print                      | Terminate, non-print |

INSTRUCTIONS FOR KEYPUNCHING  
SYMBOLIC CARDS

SYMBOLIC CARD FORMAT

| <u>CARD COLUMNS</u> | <u>DESCRIPTION</u>              |
|---------------------|---------------------------------|
| 5 - 10              | Program ID                      |
| 11 - 15             | Sequence                        |
| 16 - 21             | Label                           |
| 22 - 26             | Op Code                         |
| 27 - 28             | Field Length                    |
| 29 - 34             | Label "A" Parameter             |
| 29 - 47             | Constant Data (Numeric)         |
| 29 - 52             | Alphanumeric Data or Print Mask |
| 35 - 38             | + or - inc/rel                  |
| 39 - 42             | "B" Parameter                   |
| 43                  | "C" Parameter                   |
| 55 - 77             | Remarks                         |



Drum Card For Burroughs A 149/A 150 Key punch

## APPENDIX G (cont'd)

### A 149/A 150 KEYPUNCHING INSTRUCTIONS

1. Insert drum card – position 1.
2. Lower drum card brushes.
3. Turn Power switch ON.
4. Turn PRINTER switch ON.
5. Turn AUTO FEED switch ON.
6. Turn Program switch 1 (P1) ON.
7. First card stops in CC 5. ERR REL light turns on. Depress ERR REL switch.
8. Must punch Program I.D. CC 5-10 in 1st card\*. Thereafter, CC 5-10 will automatically duplicate.
9. CC 11-15. Sequence Number – numeric (right justified).
10. CC 16-21. Label. If no Label, depress SKIP key.
11. CC 22-26. Op Code. If OP CODE less than 5 characters, depress SKIP key.
12. CC 27-28. Field Length (right justified). If no field length, depress SKIP key.
13. CC 29-34. “A” Parameter. If less than 5 characters, depress SKIP key. If numeric, hold NUMERIC key down while punching numeric character.
14. CC 35-38. + or – Increment field. If –, enter – in CC 35 (if CC 35 is blank, + is assumed). Enter numeric in CC 36-38. If no + or – Increment, depress SKIP key.
15. CC 39-42. “B” Parameter. If numeric, hold NUMERIC key down while punching numeric character. If no “B” parameter, depress SKIP key.
16. CC 43. “C” Parameter. Numeric only. If no “C” parameter, depress SKIP key.
17. CC 53-77. Remarks columns – alphanumeric. If no Remarks, depress SKIP or REL key.
18. When numeric is to be punched, other than sequence field, hold numeric key down while punching that field.

**\*If Program I.D. is not required, the user may modify the existing drum card thusly:**

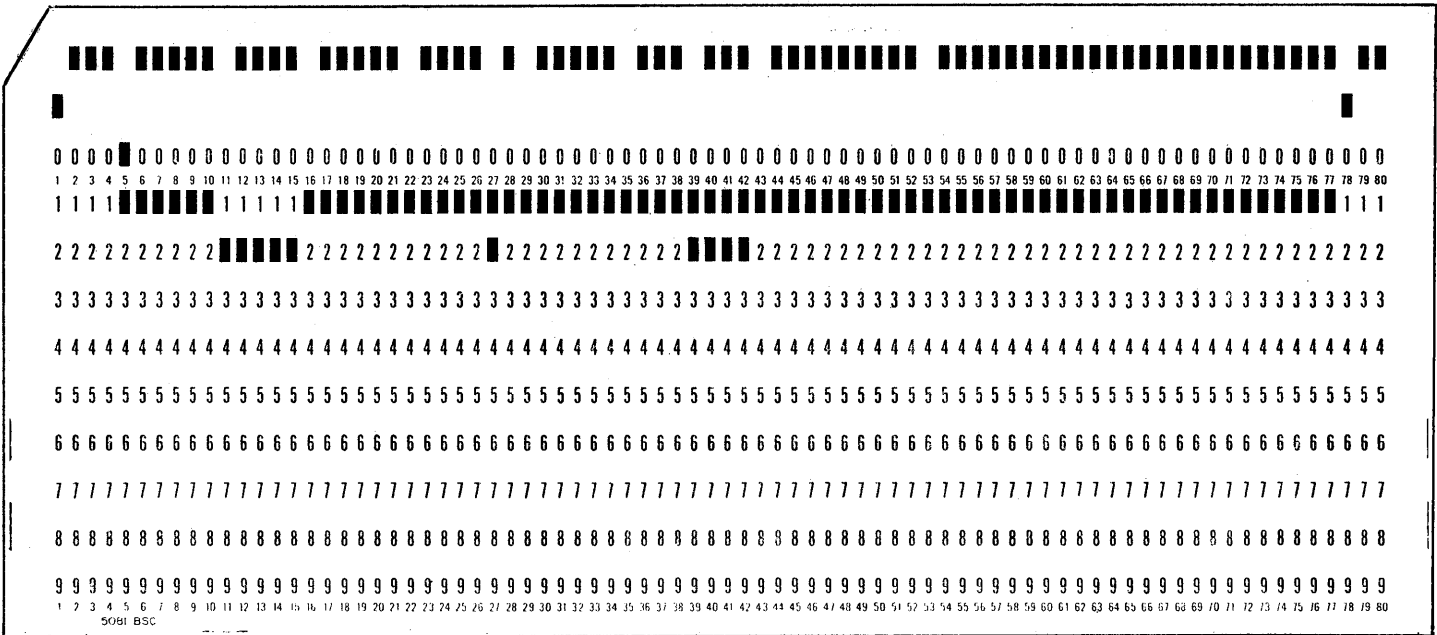
1. **Eliminate the 2 punch in card column 5. This will allow the detail card to duplicate blank columns 5 through 10.**
- or**
2. **Eliminate the 12 punch in card column 4. This will allow a skip over columns 5-10.**

## 024/026/029 KEYPUNCHING INSTRUCTIONS

1. Insert front drum card – star wheels down.
2. Turn Power switch ON.
3. Turn PRINT switch ON.
4. Turn AUTO DUP-AUTO SKIP switch OFF - first card only.
5. Must punch Program I.D. CC 5-10\*. Turn on AUTO DUP-AUTO SKIP after punch of sequence field, so that CC 5-10 will automatically duplicate on all subsequent cards.
6. CC 11-15. Sequence Number – numeric (right justified).
7. CC 16-21. Label. If no Label, depress SKIP key.
8. CC 22-26. Op Code. If OP CODE less than 5 characters, depress SKIP key.
9. CC 27-28. Field Length (right justified). If no field length, depress SKIP key.
10. CC 29-34. "A" Parameter. If less than 5 characters, depress SKIP key. If numeric, hold NUMERIC key down while punching numeric character.
11. CC 35-38. + or – Increment field. If –, enter – in CC 35 (if CC 35 is blank, + is assumed). Enter numeric in CC 36-38. If no + or – Increment, depress SKIP key.
12. CC 39-42. "B" Parameter. If numeric, hold NUMERIC key down while punching numeric character. If no "B" parameter, depress SKIP key.
13. CC 43. "C" Parameter. Numeric only. If no "C" parameter, depress SKIP key.
14. CC 53-77. Remarks columns – alphanumeric. If no Remarks, depress SKIP or REL key.
15. Whenever numeric punching is required, other than sequence field, numeric key must be held down while punching that field.

\* If Program I.D. is not required in detail card, insert a "12" punch in CC 5 of program card. This will allow skip CC 5 through 10 in detail card. AUTO DUP/AUTO SKIP key can be turned on from the very first card.

APPENDIX G (Cont'd)



A drum card with 80 columns and 9 rows of data. The data is organized into rows of characters (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and vertical bars. The first row contains a series of vertical bars of varying heights. The second row contains a series of vertical bars, with some columns containing the number 1. The third row contains a series of vertical bars, with some columns containing the number 2. The fourth row contains a series of vertical bars, with some columns containing the number 3. The fifth row contains a series of vertical bars, with some columns containing the number 4. The sixth row contains a series of vertical bars, with some columns containing the number 5. The seventh row contains a series of vertical bars, with some columns containing the number 6. The eighth row contains a series of vertical bars, with some columns containing the number 7. The ninth row contains a series of vertical bars, with some columns containing the number 8. The tenth row contains a series of vertical bars, with some columns containing the number 9. The bottom left corner of the card contains the text "5081 BSC".

Front Drum Card for 024/026/029

APPENDIX J (cont'd)

DATA COMM COMMANDS

|      |   |  |     |   |       |
|------|---|--|-----|---|-------|
| SCP  | } | 50 ms  | RSA | } | 30 ms |
| RCP  |   |  | RRA |   |       |
| LRBR |   |  | LSA |   |       |
| LKBR |   |  | LRA |   |       |
|      |   |  | RTN |   |       |
| TRAB |   | 100 ms + 10 ms/digit transferred<br>+20 ms/word boundary | RTH |   |       |
| TRCB |   | 140 ms to 180 ms (avg. 150 ms)                           | LTN |   |       |
| TRF  |   | 120 ms + 10 ms/char transferred<br>+20 ms/word boundary  | RSN |   |       |
|      |   |  | LSN |   |       |
| TRBA |   | 50 ms +10 ms/char  | RPR |   |       |
| IRCP |   | +20 ms/word boundary                                     | LPR |   |       |
|      |   |  | RPF |   |       |
| PAB  |   | 20 char/sec + 30 ms base time                            | LPF |   |       |
|      |   |  | RTF |   |       |
|      |   |  | LTF |   |       |
| TRB  | } | 40 ms  | OFF |   | 10 ms |
| TSB  |   |  |     |   |       |
| TKM  |   | can process at keyboard speed                            |     |   |       |

**NOTE:** The above times do not include the Fetch II word boundary time of 20 ms. "Word Boundary" mentioned above pertains to the actual transferring of data during the execution of a single instruction.

## APPENDIX J (cont'd)

### NOTE 1

#### Shift Timing

SLRO

SLROS

Base = 30 ms

0-3 shifts = 10 ms

4-6 shifts = 20 ms

7-9 shifts = 30 ms

10-12 shifts = 40 ms

13-15 shifts = 50 ms

Compute number of shifts left  
and number of shifts right.

### NOTE 2

#### Divide

1. a. Set down dividend (15 digits) followed by 15 zeros.
- b. Subtract divisor from dividend and repeat until dividend is smaller than divisor.
- c. Using the number of successful subtractions:
  - For no. = 0 to 3 set down 10 ms
  - For no. = 4 to 8 set down 20 ms
  - For no. =9,10,11 set down 30 ms
- d. Shift divisor one place to the right and repeat steps a, b, c, d for 15 times.
- e. Add base timing of 70 ms to total obtained above.
- f. Multiply contents of the shift register by 10 ms and add to total obtained in e.

### NOTE 3

#### Multiply

1. Set down the contents of the shift register.
2. When shift register is not equal to zero:
  - a. Examine the accumulator contents for timing purposes.
  - b. For each accumulator digit starting least significant digit.
    - For digit = 0 to 6 set down 10 ms
    - For digit = 6 to 9 set down 20 ms
  - c. Subtract 1 from shift register and repeat steps 2 a, b, c until register becomes zero.



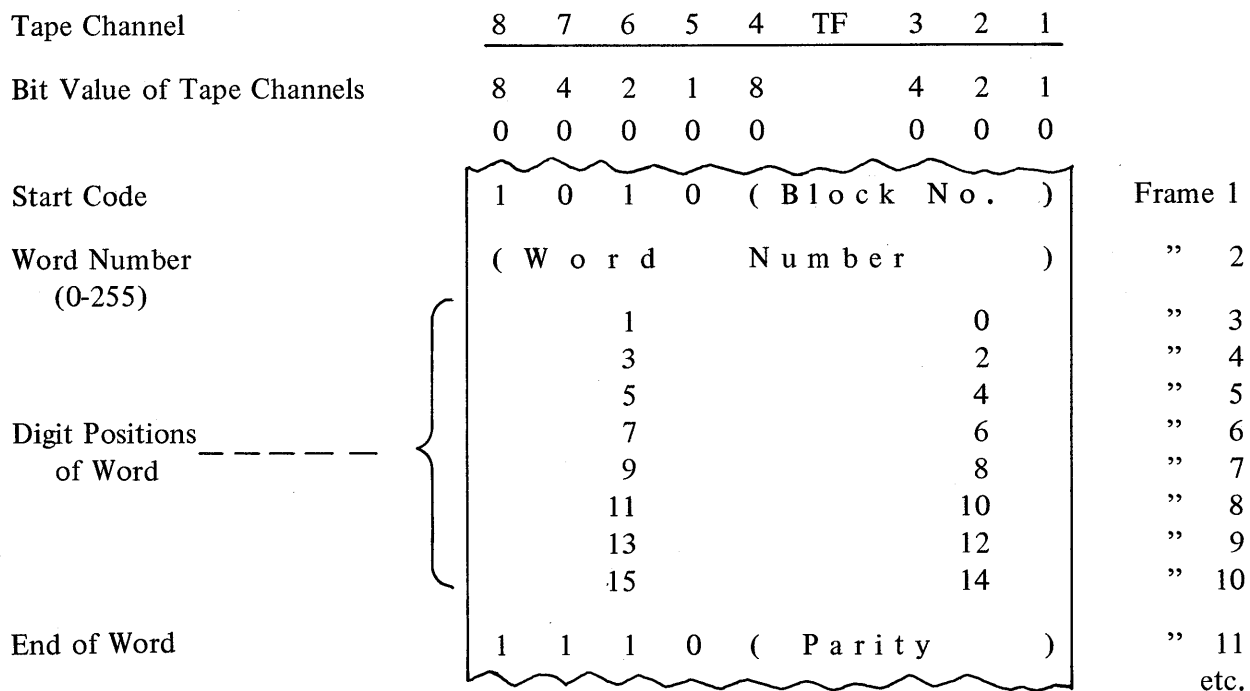


SERIES L/TC OBJECT CODE

PUNCHED PAPER TAPE OBJECT TAPE CODE

The paper tape output is in the format illustrated below. Each 16-digit word is compressed into 8 frames of tape. Each frame contains a "lower digit" (channels 1-4) and an "upper digit" (channels 5-8). This type of punch format is referred to as "compact code" or "compact Hexadecimal code". Most object program tapes will be punched in this format.

In the diagram below, "1" represents a punch (bit on) and "0" represents no punch (bit off) in the tape channel indicated.



The diagram shown represents a word of program as punched into paper tape. The first word contains the Start Code-Block Number frame and the Word Number frame. The End of Word Parity frame will be punched with every word. The End of Word code with the very last word in a sequence is "0 1 0 1". Parity (4 bits, as indicated above) is arrived at by exclusively OR'ing each four bit grouping as shown above with the exception of the Start Code and the End of Word Code. Figures 1 and 2 illustrate the two types of compact object tape.

PUNCH PAPER TAPE COMPACT OBJECT CODE

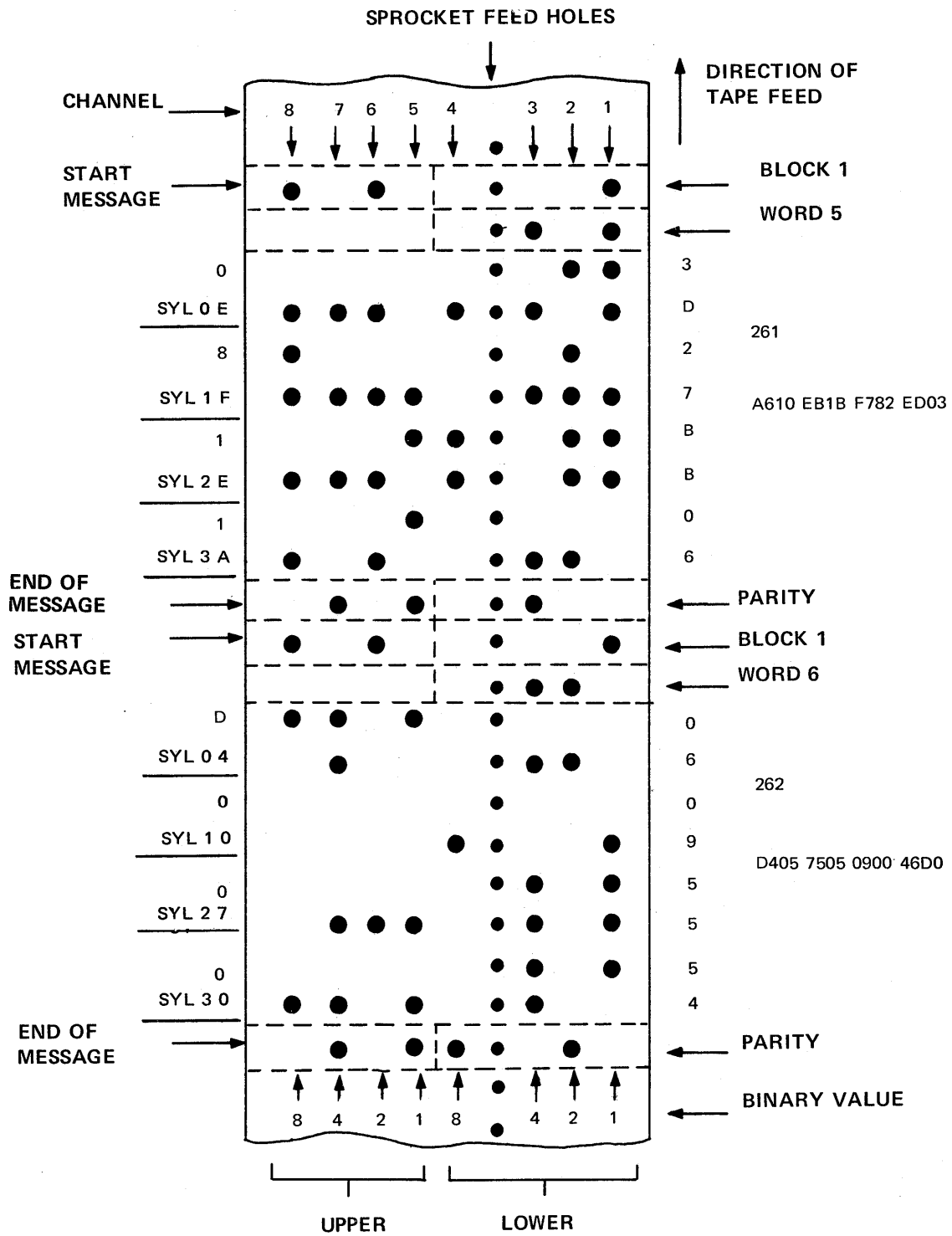


Figure 1.

PUNCH PAPER TAPE COMPACT OBJECT CODE

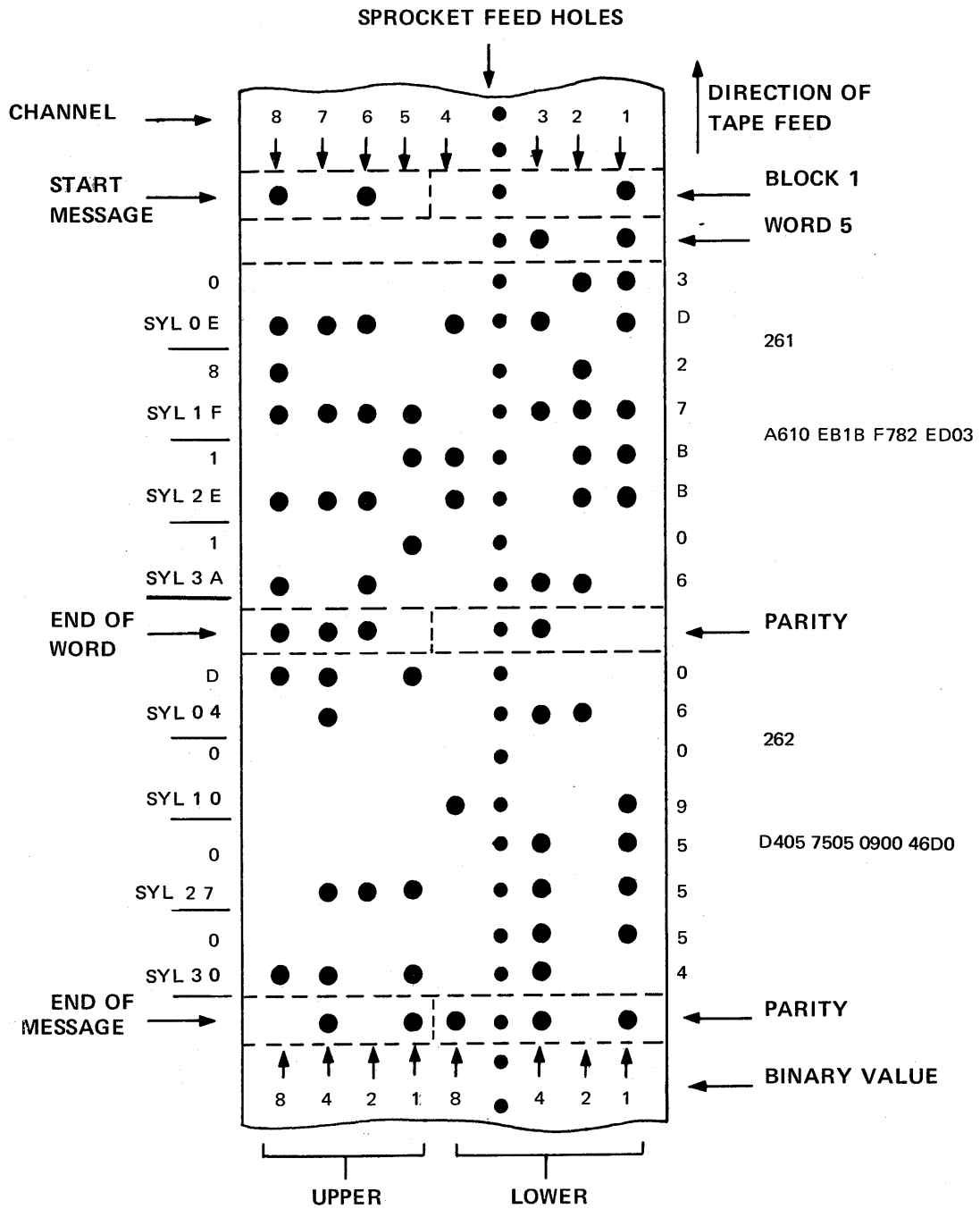


Figure 2

**APPENDIX K (Cont'd)**

**USASCII OBJECT PROGRAM TAPE FORMAT**

**B 5500/B 300 OUTPUT**

Each program word in tape consists of 21 frames punched in USASCII code.

**Tape Frame**

- 1 Block No. (0 to 3)
- 2-4 Word No. (000 to 255)
- 5-20 16 Digits:

**Frame Digit Position**

- 5 1
- 6 0
- 7 3
- 8 2
- 9 5
- 10 4
- 11 7
- 12 6
- 13 9
- 14 8
- 15 11
- 16 10
- 17 13
- 18 12
- 19 15
- 20 14

Hexadecimal digit Value expressed as code from Column 3 of USASCII table:

**Hexadecimal**

**USASCII**

|   |   |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| A | : |
| B | ; |
| C | < |
| D | = |
| E | > |
| F | ? |

- 21 Termination Code (1,E)

Sprocket Feed Holes

|                  | P | 7 | 6 | 5 | 4 | o | 3 | 2 | 1 |                       |
|------------------|---|---|---|---|---|---|---|---|---|-----------------------|
| Channel          |   |   |   |   |   | o |   |   |   |                       |
| Block Number     | • |   | • | • |   | o |   |   | • | 1                     |
| Word Number      | • |   | • | • |   | o |   | • | • | 1                     |
|                  |   |   | • | • |   | o | • |   | • | 5 153                 |
|                  |   |   | • | • |   | o |   | • | • | 3                     |
|                  |   |   | • | • |   | o |   |   |   | 0                     |
|                  |   |   | • | • |   | o |   | • | • | 3                     |
|                  | • |   | • | • | • | o | • | • |   | E                     |
|                  | • |   | • | • | • | o | • |   | • | D                     |
|                  | • |   | • | • | • | o |   |   |   | 8                     |
|                  | • |   | • | • |   | o |   | • |   | 2                     |
| Program          |   |   | • | • | • | o | • | • | • | F A610 EB1B F782 ED03 |
| Word             | • |   | • | • |   | o | • | • | • | 7                     |
|                  | • |   | • | • |   | o |   |   | • | 1                     |
|                  | • |   | • | • | • | o |   | • | • | B                     |
|                  | • |   | • | • | • | o | • |   | • | E                     |
|                  | • |   | • | • | • | o |   | • | • | B                     |
|                  | • |   | • | • |   | o |   |   | • | 1                     |
|                  |   |   | • | • |   | o |   |   |   | 0                     |
|                  |   |   | • | • | • | o |   | • |   | A                     |
|                  |   |   | • | • |   | o | • | • |   | 6                     |
| Termination Code |   |   | • | • |   | o | • | • |   |                       |
|                  |   |   |   |   |   | o |   |   |   |                       |
|                  |   |   |   |   |   | o |   |   |   |                       |
|                  |   |   |   |   |   | o |   |   |   |                       |
|                  |   |   |   |   |   | o |   |   |   |                       |

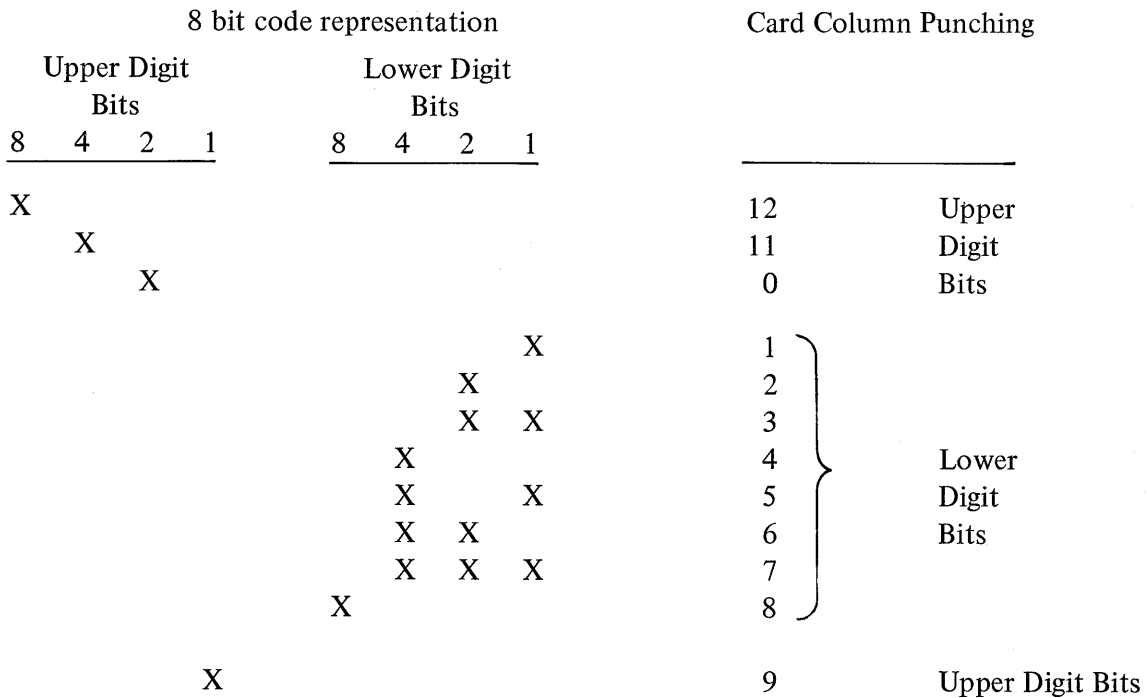
APPENDIX K (Cont'd)

THE SERIES L PROGRAM PUNCH CARD FORMAT (COMPACT HEXADECIMAL)

Card-Column

|         |  |
|---------|--|
| 1 - 6   | Program Identification (Alpha-in BCL Code)   |
| 9       | Beginning Word Number (hexadecimal value for word 0 to 255)                          |
| 10      | Number of Program Words on Card (Decimal 1 to 8)                                     |
| 11 - 15 | No significance  |
| 16      | Block Number<br>Block 0 = blank card column<br>Block 1 = decimal 1                   |
| 17 - 24 |  |
| 25 - 32 |  |
| 33 - 40 | Up to 8 program words  |
| 41 - 48 | Each word occupies 8 card columns.   |
| 49 - 56 | Each card column contains binary value for 2 of the 16 hexadecimal digits in a word. |
| 57 - 64 |  |
| 65 - 72 |  |
| 73 - 80 |  |

Hardware will cause the 12 bit representation on an 80 Column Card to be compressed into 8 bits in the Card Read Area during input, and conversely will cause the 8 bit representation in memory to be expanded into 12 bits on the output punch card in the following manner.



A Sample Output Card would appear as follows:

|                        |   |  |  |  |  |  |  |  |  |  |  |
|------------------------|---|--|--|--|--|--|--|--|--|--|--|
| UPPER<br>DIGIT<br>BITS | 8 |  |  |  |  |  |  |  |  |  |  |
|                        | 4 |  |  |  |  |  |  |  |  |  |  |
| LOWER<br>DIGIT<br>BITS | 2 |  |  |  |  |  |  |  |  |  |  |
|                        | 1 |  |  |  |  |  |  |  |  |  |  |
| P<br>U<br>N<br>C<br>H  |   |  |  |  |  |  |  |  |  |  |  |
|                        |   |  |  |  |  |  |  |  |  |  |  |
| V<br>A<br>L<br>U<br>E  |   |  |  |  |  |  |  |  |  |  |  |
|                        |   |  |  |  |  |  |  |  |  |  |  |
| CARD COLUMN            | 1 |  |  |  |  |  |  |  |  |  |  |
| PUNCH VALUE            |   |  |  |  |  |  |  |  |  |  |  |
| FOOTNOTE               |   |  |  |  |  |  |  |  |  |  |  |

## APPENDIX K (Cont'd)

### FOOTNOTES

1. Program I. D. (Alpha BCL Code)
2. Beginning Word Number (Hexadecimal 0-255)
3. Number of Program Words on card (Decimal 1-8)
4. Block Number (Decimal 0-4)
5. 12 Program Words 1-8 (Compact Hexadecimal-2 Hexadecimal digits per card column)

| Word No. | Syllable | 3    | 2    | 1    | 0    |
|----------|----------|------|------|------|------|
| 1        |          | EB0A | 7C07 | 4184 | 5910 |
| 2        |          | EB14 | 9C01 | D620 | B030 |
| 3        |          | 6100 | 3039 | B02D | BC32 |
| 4        |          | D7E0 | C241 | 45C1 | 3927 |
| 5        |          | EB5A | C52D | D1E2 | EB4A |
| 6        |          | 4198 | C218 | 4598 | BC07 |
| 7        |          | 6E02 | 3839 | 0CFD | C212 |
| 8        |          | 6423 | 6E03 | 7810 | 6422 |

### BCL OBJECT PROGRAM CARD FORMAT

#### Card Column

|       |  |
|-------|--|
| 1-6   | PROGRAM I. D.                                    |
| 12    | Number of words in card (1 to 4)                 |
| 13-16 | Beginning Word Number (0000 to 1023)             |
| 17-32 | 1st word (digit 15 in cc 17 to digit 0 in cc 32) |
| 33-48 | 2nd word (digit 15 in cc 33 to digit 0 in cc 48) |
| 49-64 | 3rd word (digit 15 in cc 49 to digit 0 in cc 64) |
| 65-80 | 4th word (digit 15 in cc 65 to digit 0 in cc 80) |