



Burroughs L/TC

BASIC ASSEMBLER

REFERENCE MANUAL

Burroughs

L/TC

BASIC ASSEMBLER

REFERENCE MANUAL

INTRODUCTION

CODING
FORM

GP 300
INSTRUCTIONS

SYMBOLIC
PROGRAMING
PROCEDURES

PROGRAMING
EXAMPLE

ASSEMBLERS

APPENDIX

\$5.00

Burroughs Corporation

Detroit, Michigan 48232

**COPYRIGHT © 1970, 1971 BURROUGHS CORPORATION
AA167490**

Burroughs Corporation believes the program described herein to be accurate and reliable, and much care has been taken in its preparation. However, the Corporation cannot accept any responsibility, financial or otherwise, for any consequences arising out of the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

NOTE:

THIS PRINTING INCORPORATES:

**PCN 1045481-001, dated March 29, 1971
PCN 1045481-002, dated July 23, 1971
PCN 1045481-003, dated August 9, 1971
PCN 1045481-004, dated November 8, 1971**

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)

| Ref. No. | Subject |
|-------------|--|
| 2.00 | INTRODUCTION |
| 2.01 | ASSEMBLER PSEUDO INSTRUCTIONS |
| 2.02 | KEYBOARD INSTRUCTIONS |
| 2.02.01 | Enable Numeric Keyboard Instructions |
| 2.02.02 | Operation Control and Program Keys |
| 2.02.03 | Typewriter Keyboard Instructions |
| 2.03 | PRINT INSTRUCTIONS |
| 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 |
| 2.04 | FORMS CONTROL INSTRUCTION |
| 2.04.01 | Forms Handler – Open and Close Instruction |
| 2.04.02 | Platen Control Register Instructions |
| 2.04.03 | Line Advance Instructions |
| 2.05 | ARITHMETIC INSTRUCTIONS |
| 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 |
| 2.06 | DATA MOVEMENT INSTRUCTIONS |
| 2.06.01 | Transfer Instructions |
| 2.06.02 | Shift Accumulator Instructions |
| 2.07 | FLAG INSTRUCTIONS |
| 2.08 | INDEX REGISTER INSTRUCTIONS |
| 2.09 | BRANCH AND DECISION INSTRUCTIONS |
| 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 |
|-------------|---|
| 2.10 | MISCELLANEOUS INSTRUCTIONS |
| 2.11 | CHECK DIGIT INSTRUCTIONS |
| 2.12 | DATA COMMUNICATIONS INSTRUCTIONS |
| 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 |
| 2.13 | POINT-TO-POINT PROGRAMING PROCEDURES |
| 2.13.01 | Basic Point-to-Point Line Discipline |
| 2.13.02 | Control Registers |
| 2.13.03 | Indicator Register Flags |
| 2.14 | CENTRAL TC CONTROLLER PROGRAMING PROCEDURES |
| 2.14.01 | Line Discipline Format Registers |
| 2.14.02 | Data Comm Processor Operations |
| 2.14.03 | Main Memory Processor |
| 2.15 | INPUT WITH PUNCHED PAPER TAPE/EDGE PUNCHED CARD READER |
| 2.15.01 | Paper Tape Reader Instructions |
| 2.15.02 | Paper Tape/Edge Punched Card Input Instructions |
| 2.16 | OUTPUT WITH PAPER TAPE/EDGE PUNCHED CARD PERFORATOR |
| 2.16.01 | Paper Tape/Edge Punched Card Output Instructions |
| 2.16.02 | Reader and Punch Flags |
| 2.17 | 80-COLUMN PUNCHED CARD INPUT INSTRUCTIONS |
| 2.17.01 | 80-Column Card Input Instructions |
| 2.17.02 | Input Indicator Lights and Flags |
| 2.17.03 | Program Keys |
| 2.18 | 80-COLUMN PUNCHED CARD OUTPUT INSTRUCTIONS |
| 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)

| Ref. No. | Subject |
|-----------------|---|
| 2.19 | MAGNETIC UNIT RECORD INSTRUCTIONS |
| 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 |
| 2.20 | MESSAGE UNPACKING ROUTINE |
| 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 |
| 2.21 | TRANSACTION CODE TRANSLATOR |
| 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 |

TABLE OF CONTENTS (continued)

| Ref. No. | Subject |
|----------------|---|
| 5.00.00 | ASSEMBLERS |
| 5.00.00 | FUNCTIONAL DESCRIPTION OF BASIC ASSEMBLERS |
| 5.01.00 | L/TC PAPER TAPE ASSEMBLERS |
| 5.01.01 | Environment |
| 5.01.02 | Phase I |
| 5.01.03 | Assembler I |
| 5.01.04 | Assembler ISL |
| 5.01.05 | Assembler VI and VIII |
| 5.01.06 | Keyboard Mode |
| 5.01.07 | Phase I – Condensed Operating Instructions |
| 5.01.08 | Phase I – Diagnostic Facilities |
| 5.01.09 | Phase II |
| 5.01.10 | Phase II – Operating Instructions |
| 5.01.10 | Phase II – Condensed Operating Instructions and Index |
| 5.01.11 | Phase II – Error Detection and Indication |
| 5.02.00 | 80-COLUMN CARD INPUT ASSEMBLERS |
| 5.02.03 | Environment |
| 5.02.03 | Card I/O Assemblers |
| 5.02.03 | Card In/Paper Tape Out Assemblers |
| 5.02.04 | Input |
| 5.02.05 | Operating Instructions |
| 5.03.00 | L/TC ASSEMBLER III B 3500 VERSION |
| 5.03.02 | Environment |
| 5.03.02 | Library Tape Input |
| 5.03.03 | MCP Control Cards |
| 5.03.04 | Option Control Cards |
| 5.03.05 | Operating Instructions |
| 5.03.06 | Error Detection |
| 5.03.07 | Output |
| 5.03.08 | LIBTAP – Utility Routine |
| 5.04.00 | L/TC ASSEMBLER IV – B 5500 VERSION |
| 5.04.01 | Environment |
| 5.04.02 | MCP Control Cards |
| 5.04.03 | Option Control Cards |
| 5.04.04 | Operating Instructions |
| 5.04.05 | Operation |
| 5.04.06 | Error Detection |
| 5.04.07 | Output |
| 5.05.00 | L/TC ASSEMBLER V – B 300 VERSION |
| 5.05.01 | Environment |
| 5.05.02 | Input |
| 5.05.03 | Output |
| 5.05.04 | Control Cards |
| 5.05.05 | End |
| 5.05.06 | Operating Instructions |
| 5.05.07 | Error Messages |

TABLE OF CONTENTS (continued)

| SECTION | TITLE | PAGE |
|--------------------|---|-------------|
| 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 Keypunching Symbolic Cards | G-1 |
| | Symbolic Card Format | G-1 |
| | A 142/A 150 Keypunching Instructions | G-2 |
| | 024/026/029 Keypunching 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 xii 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.

BURROUGHS ASSEMBLER CODING FORM

PAGE _____ OF _____

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| PROGRAM ID | | | | |
|------------|---|---|---|----|
| 5 | 6 | 7 | 8 | 10 |
| | | | | |

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LEN-GTH | PARAMETER | | | REMARKS |
|------|----------|-------|----------|---------------|-----------|----------------|---|---------|
| | | | | | A | B | C | |
| | | | | | LABEL | + OR - INC/REL | | |
| | 0,1 | | | | | | | |
| | 0,2 | | | | | | | |
| | 0,3 | | | | | | | |
| | 0,4 | | | | | | | |
| | 0,5 | | | | | | | |
| | 0,6 | | | | | | | |
| | 0,7 | | | | | | | |
| | 0,8 | | | | | | | |
| | 0,9 | | | | | | | |
| | 1,0 | | | | | | | |
| | 1,1 | | | | | | | |
| | 1,2 | | | | | | | |
| | 1,3 | | | | | | | |
| | 1,4 | | | | | | | |
| | 1,5 | | | | | | | |
| | 1,6 | | | | | | | |
| | 1,7 | | | | | | | |
| | 1,8 | | | | | | | |
| | 1,9 | | | | | | | |
| | 2,0 | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

SECTION 1

ASSEMBLER CODING FORM

| PROGRAM ID | | | | | |
|------------|---|---|---|---|----|
| 5 | 6 | 7 | 8 | 9 | 10 |
| | | | | | |

PROGRAM IDENTIFICATION

DEFINITION – Identifies a specific program.

FIELD DEFINITION – One through six alphanumeric characters entered in columns 5-10. Right or left justified. Automatically reproduced on succeeding cards with punched card source program.

PAGE _____ OF _____

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

PAGE NUMBER AND HEADING

DEFINITION – Identifies and sequentially locates coded pages of a program.

FIELD DEFINITION – The page number is determined by the sequential order number of the page and the total number of pages. The remaining information is filled in accordingly.

| CODE | SEQUENCE | | | | |
|------|----------|----|----|----|----|
| 1 | 11 | 12 | 13 | 14 | 15 |
| | | | 0 | 1 | |
| | | | 0 | 2 | |
| | | | 0 | 3 | |
| | | | 0 | 4 | |
| | | | 0 | 5 | |

SEQUENCE

DEFINITION – Identifies the sequential order of the operation codes. Applies only to punched card source programs.

FIELD DEFINITION – With a keyboard or paper tape source program the Basic Assembler assigns a sequence number to each line in increasing numerical sequence.

**CODING
FORM**

| LABEL | | | | | |
|-------|----|----|----|----|----|
| 16 | 17 | 18 | 19 | 20 | 21 |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

LABEL

DEFINITION – A symbolic designation utilized by the assembler to describe a parameter for a memory location or other parameter value.

FIELD DEFINITION – A label consists of 1-6 alpha or numeric characters. The first character must be an alpha character. A label may be the same as a mnemonic operation code of any GP 300 instruction or assembler pseudo instruction. The label is entered in columns 16-21 and must be left justified.

| OP. CODE | | | | |
|----------|----|----|----|----|
| 22 | 23 | 24 | 25 | 26 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

OPERATION CODE

DEFINITION – The applicable symbolic instruction is entered in this field.

FIELD DEFINITION – Op. Code is entered in columns 22-26 and left justified.

| |
|----------------------|
| FIELD LEN- GTH |
| 27 28 |
| |
| |
| |
| |
| |
| |

FIELD LENGTH

DEFINITION – Indicates the number of characters or digits in the constants associated with certain instructions. Applies to punch card source programs only.

FIELD DEFINITION – Number of characters contained in required constant entered in columns 27-28 and right justified.

| PARAMETER | | | | | | | | | | | | | | |
|-----------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|
| A | | | | | | B | | | | C | | | | |
| LABEL | | | | | | + OR - INC/REL | | | | | | | | |
| 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |

A PARAMETER – LABEL

DEFINITION – The applicable label or parameter is contained in this field.

FIELD DEFINITION – Label entries consist of 1-6 alpha or numeric characters and the first character must be an alpha character. The parameter or label is entered in columns 29-34 and left justified.

A PARAMETER – ± INCREMENT

DEFINITION – A signed numeric entry may be made in this field to denote a plus or minus value for incrementing or relative addressing with the label in columns 29-34 as a base. If a label is not used when using a branch instruction, the syllable location of the same instruction is used for the base address.

FIELD DEFINITION – If the field has a negative value, the “-” must be entered in column 35. For a positive value the “+” is optional. The increment is entered in columns 36-38 and right justified.

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.

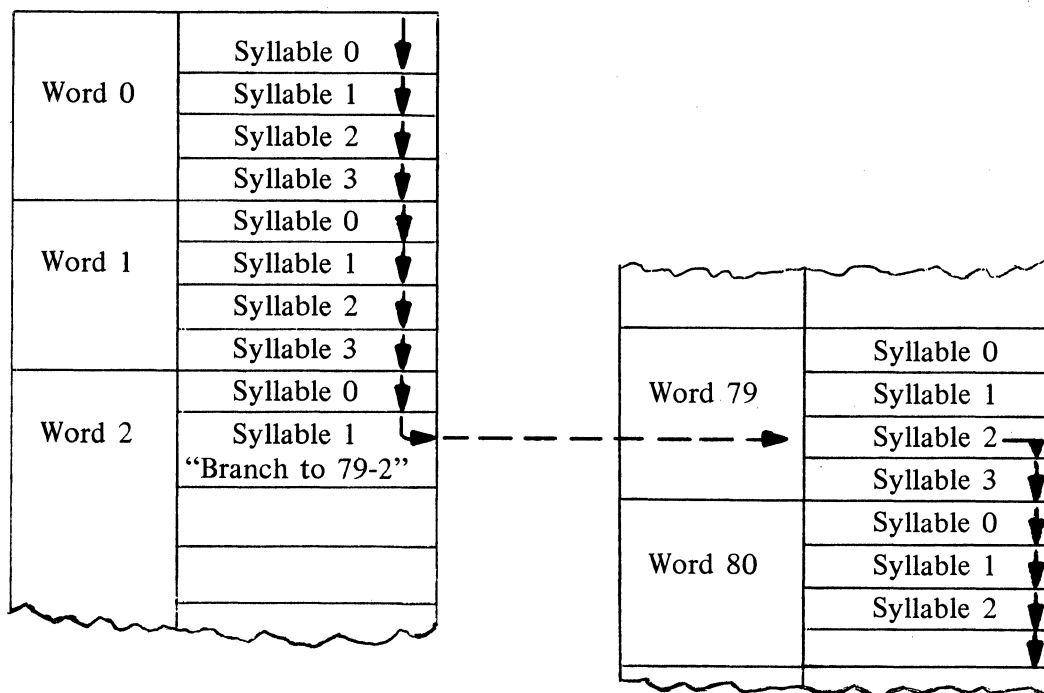
| | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| FLAGS | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|

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.

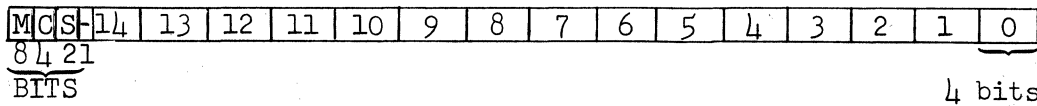
2.00.05 FLAGS

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.

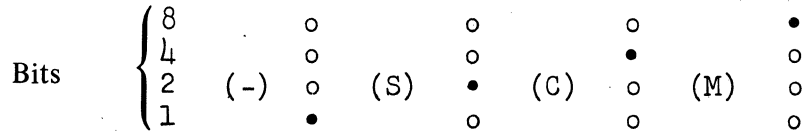
Each flag consists of 1 "bit." When the bit is "ON," the flag is "Set"; when the bit is "off," the flag is "Reset." The program can interrogate a flag to test whether or not it is set or reset, and select a path of program accordingly.

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



| | | |
|-------------|------------------|-----------------------|
| Accumulator | (M) Per Thousand | (S) Special |
| Flags | (C) Per Hundred | (-) 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 |

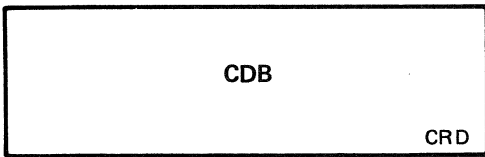
The ALF pseudo instruction permits alphanumeric data, up to 24 characters, to be stored in memory as constant data during program loading. Any character on the keyboard, including space, is a valid character. (Except for Assembler I, a CC in columns 27 and 28 will allow a second line of 24 characters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

| LABEL | | OP. CODE | | FIELD LENGTH | | 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 |
| | | | | | | | | | | | | | NAME | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | 8 | JOHN | DOE | | | | | | | | | | | | | | | | |



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.

The CDB pseudo instruction is necessary only when the L/TC is used with an A 595 card reader.

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 |
| 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | PARAMETER | | | | | | | | | |
|-------|----|----|----|----|----|----------|----|----|----|----|----------------------|-------|---|---|---|---|-------------------|--|--|--|--|----|----|----|----|-----------|----|----|----|----|--|--|--|--|--|
| LABEL | | | | | | OP. CODE | | | | | FIELD LEN- GTH | A | | | | | B | | | | | C | | | | | | | | | | | | | |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | | LABEL | | | | | + OR - INC/REL | | | | | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | | | | | |
| | | | | | | L | P | N | R | | | P | M | A | S | K | | | | | | | | | | | | | | | | | | | |
| | | | | | | L | P | K | R | | | P | K | E | Y | S | | | | | | | | | | | | | | | | | | | |
| | | | | | | L | L | L | R | | | 5 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | B | R | U | | | | B | E | G | I | N | | | | | | | | | | | | | | | | | | | |
| | | | | | | R | E | G | | | | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | |
| B | E | G | I | N | | R | C | D | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| <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 | 1 | | Assembles in word 11, syl- lable 0 |

When using the A 596 Card Reader, the L programmer may utilize the "REG" pseudo instruction to set up multiple buffers anywhere in the Track 0 area (words 1-31) allotted for card input.

A 596 Reader input to Series L

| <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 | | Saves words 1-10 |
| | REG | 10 | | Saves words 11-20 |
| | REG | 10 | | Saves words 21-30 |
| BEGIN | RCD | 1 | | Assembles word 31 Puts data in words 1-10 |
| | RCD | 21 | | Puts data in words 21-30 |
| | RCD | 11 | | Puts data in words 11-20 |

The TC programmer may use any area with Block 0 (words 1-255). However, when processing data for transmission, he should place his buffers on track boundaries (words 32, 64, 96, 128, . . . 224). This allows the data to be accessed with Data Comm instructions. The "ORG" and "REG" pseudo instructions can be used to set up buffers on specific track boundaries. Care must be taken to ensure that buffer area and the memory area required for the macro program do not overlap.

2.01.04 CARD FORMAT INSTRUCTION

| <u>OP CODE</u> | <u>A 595 READER</u> | | <u>A 596 READER</u> | |
|----------------|---------------------|----------|---------------------|----------|
| | <u>A</u> | <u>B</u> | <u>A</u> | <u>B</u> |
| CDF | 1-80 | 1-80 | 1-255 | 1-255 |

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.

Only 16 CDF's may be listed for each table name (LABEL).

| <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 |

The A 596 Card Reader allows the L programmer using card I/O instructions to read and access data into any location in Track 0 except word 0. This provides for a maximum of 248 characters which can be entered on successive cards. If more than 16 CDF's are required, the programmer must use multiple table-names.

The TC programmer must access buffer areas above Track 0 with Data Comm Instructions (Section 2.12.04 – Unpacking Messages Received).

CODE

2.01.05 CODE INSTRUCTION

OP CODE

A

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:

| 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 |
| | | | | | | | | | | | | | C | 9 | 2 | 5 | | | | | | | | | | | | | | |

OP CODE

A

REMARKS

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 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 | |
| | | | | | | POS | | | | | | | SHIPTO | | | | | | | | | | | | | | | | | | |
| | | | | | | } | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | SHIPTO | | | | | | | DEF | | | | | | 35 | | | | | | | | | | | | |

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

| |
|------|
| DOC |
| ESTB |
| DC |

LABEL

OP CODE

A

B

ORDER

NK

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:

| LABEL | | OP. CODE | | | | | FIELD LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | |
|---------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|
| | | | | | | | | A | | | | | | | | B | | | | C | | | | | | | | | | | |
| LABEL | | | | | | | | 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 | | | | | | | | | | | | | | | | | | | |
|-------------|----|----------------------|-----------|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| | | | A | | | | | + OR - INC/REL | | | B | | C | | | | | | | | | |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | |
| NOTE | | | | | | | | | | | | | | | | | | | | | | |

| REMARKS | | | | | | | | | | | | | | | | | | | | |
|-----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 |
| 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 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 |
| | | 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:

| LABEL | | | | | | | | | | | | | | | | | | | | | OP. CODE | | | | | | FIELD LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | |
|-------|--|--|--|--|--|-------|--|--|--|--|--|----|-------|--|--|--|--|--|-------|--|----------|--|--|--|-------|--|----------------------|-----------|--|--|--|--|--|--|--|--|---|--|--|--|--|--|---|--|--|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | A | | | | | | | | | B | | | | | | C | | |
| 16-21 | | | | | | 22-27 | | | | | | 28 | 29-34 | | | | | | 35-40 | | | | | | 41-46 | | | | | | | | | | | | | | | | | | | | |
| 16-21 | | | | | | 22-27 | | | | | | 28 | 29-34 | | | | | | 35-40 | | | | | | 41-46 | | | | | | | | | | | | | | | | | | | | |
| AREA | | | | | | LKBR | | | | | | | AREA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | TKM | | | | | | | 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | ? | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AREA | | | | | | REG | | | | | | | 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:

| 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 | | | |
| | | | | | | L | P | K | R | | | | P | K | E | Y | S | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| <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, 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 "RE," "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 |
| N | K | | | | | | 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 00000000543200 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

| | |
|-----|------|
| 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | | | | | | | | | | | | | | | | |
| | | | | | | PKA | | | | | | | 1 2 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | NK | | | | | | | 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | | | | | | | | | | | | | | | | |
| | | | | | | LKBR | | | | | | | | TYPE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | TKM | | | | | | | | 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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 prepositioned 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.

| | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | | | | | | | | | | | | | | | | |
| | | | | | | TK | | | | | | | | 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 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 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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.

EAM

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 LEN-GTH | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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 | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | 48 | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | P0S | | | 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

| | |
|----------------|----------|
| OP CODE | A |
| 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | PARAMETER | | | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----------|----|----|----|----|----------------------|-------|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|-----------|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| LABEL | | | | | | OP. CODE | | | | | FIELD LEN- GTH | A | | | | | | | | | | B | | | | C | | | | | | | | | | | | | | | | | |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | | 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 | | | | | | | | | | | | | | | | |
| | | | | | | L | P | N | R | | | | | F | 0 | R | M | A | T | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | Z | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | F | 0 | R | M | A | T | M | A | S | K | | | | | | | | | D | D | . | D | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | Z | Z | Z | . | D | D | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | Z | Z | Z | , | Z | Z | Z | . | D | D | | | | | | | | | | | | | | | | | | | | |

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. 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. 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. |

| |
|-------|
| 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 IF MINUS | PNS- | 0-14 | 0-15 |
| PRINT NUMERIC, SHIFT RIBBON 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.

Power to the printer (keyboard) motor will turn off if the printer motor OFF button is being held depressed when the POF instruction is executed.

The POF instruction will not cause power to the printer motor to be turned OFF unless the Printer Off button is depressed when the POF instruction is executed. The POF instruction will execute as a NOP if it is executed when the Printer Motor Off button is not depressed.

When the printer motor is OFF, the execution of the functions for printing, keyboard buffering, keyboard indexing, the forms handler, and alarm are prevented. Attempted execution of any of the preceding instructions will result in a system block. Attempted execution of PON or POF on a machine not equipped with the Printer Motor OFF/ON buttons will also cause a system block. A systems block can be cleared only by depression of the Program Halt and Clear Button.

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u> |
|--------------|----------------|----------|----------|----------|--|
| LOOP | EX | D | 2 | 1 | If message has been received, process it. If keyboard buffer is empty continue in loop. Printer power on button must have been depressed for data to be in keyboard buffer. |
| | BRU | RECEIV | | | |
| | SK | B | 3 | 1 | |
| | BRU | KEYBRD | | | |
| | POF | | | | |
| | BRU | LOOP | | | |
| RECEIV | PON | | | | Printer power must be turned before PAB can be executed. |
| | LRBR | 0 | | | |
| | PAB | 12 | | | |
| PM-OFF | EX | B | 3 | 2 | If the keyboard buffer is empty execute the POF and continue in the PRNTR motor off loop. Before the PM-OFF loop can be exited, the PRNTR power on button must be depressed and data must be entered into the keyboard buffer. |
| | POF | | | | |
| | BRU | PM-OFF | | | |
| | LKBR | TABLE | | | |
| | EAM | 4 | | | |
| | NOTE | | | | |
| | NOTE | | | | |

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.

Handwritten notes at the top left of the page, possibly a title or header.

Handwritten title or section header at the top right of the page.

Handwritten notes in the middle left section, appearing as a list or series of points.

Handwritten notes in the middle right section, continuing the list or series of points.

Handwritten notes in the lower middle section, possibly a paragraph or summary.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten title or section header in the lower middle section.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the lower middle section, continuing the text.

Handwritten notes in the bottom left section, appearing as a list or series of points.

Handwritten notes in the bottom right section, continuing the list or series of points.

Handwritten notes at the bottom of the page, possibly a footer or concluding remarks.

| | | | | |
|------|------|------|------|------|
| 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.

$$\begin{aligned} \text{LLLR} &= 50 \\ \text{LLCR} &= 50 \end{aligned}$$

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

$$3 - 20 = -17$$

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

Value of Limit Register – 17 =

$$255 - 17 = 238$$

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 =

$$25 - 20 = 5$$

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

| | | | |
|------|------|-------|------|
| LLCR | LLLR | LR CR | LRLR |
| AL | AR | ALR | ALTO |
| | | | ARTO |

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

Assume contents of Left Count Register = 20, when ALTO command is executed. This is an example of the type of programing 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.

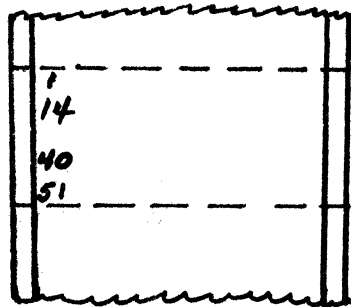
The form advances 10 lines, then the LLLR = LLCR, on the next line advance the Count Register is set to 1. Advancing continues for 4 more lines to line 5 of the new form. In this case, the last line on the form would be line 30.

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

Example:

Suppose we have the following form:

Line 1
 First Print Line
 Last Print Line
 Last Line on Form



The following programing 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> |
|----------------|----------|----------|----------|
| LLLR | 40 | | |
| LLCR | 0 | | |
| AL | 1 | | |
| EX | T | L | 1 |
| ALTO | 17 | | |
| LLLR | 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 | LLLR | LRCR | LRLR | |
| AL | AR | ALR | ALTO | ARTO |

| <u>OP CODE</u> | <u>A</u> |
|----------------|----------|
| LLLR | 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 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 |
| | | | | | ADA | | | | | AREA | | | | | | | | | | | | | | | | | | | | |

| <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 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 |
| | | | | | ADM | | | | | AREA | | | | | | | | | | | | | | | | | | | | |

| <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. |

INK

Example 1:

| 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 |
| | | | | | CLM | | | | | | AREA | | | | | | | | | | | | | | | | | | | | |

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

Example 2:

| 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 |
| | | | | | CLA | | | | | | 0 | | | | | | | | | | | | 0 | | | | | | | | |

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

2.05.04 INSERT CONSTANT IN ACCUMULATOR INSTRUCTION

| <u>INSERT CONSTANT IN ACCUMULATOR</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---------------------------------------|----------------|----------|----------|
| | INK | 0-15 | 0-15 |

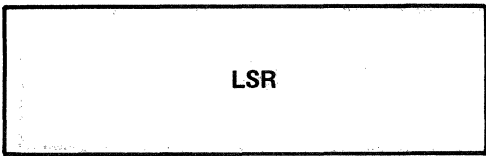
The INK instruction inserts the digit specified by the B field in the digit position of the Accumulator specified by the A field. The remaining digit positions are unaffected.

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 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 |
| | | | | | INK | | | | | | 0 | | | | | | | | | | | | 3 | | | | | | | | |

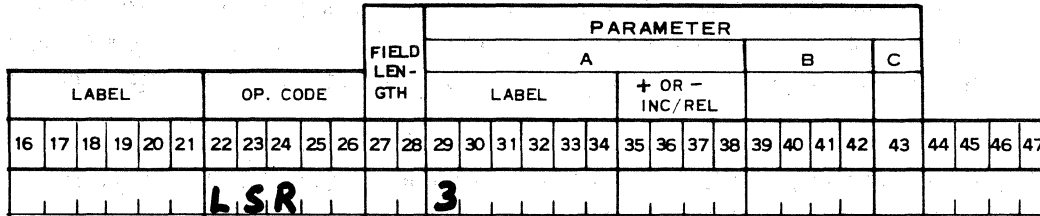


| OP CODE | A | B | REMARKS |
|---------|---|---|---|
| INK | 0 | 3 | The digit 3 will be placed in Accumulator digit position 0 replacing the previous contents of Accumulator digit position 0. |

2.05.05 MULTIPLICATION AND DIVISION INSTRUCTIONS

| | OP CODE | A |
|---------------------|---------|------|
| LOAD SHIFT REGISTER | 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.



| OP CODE | A | REMARKS |
|---------|---|----------------------------|
| 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 in multiplier | + | Number of places in multiplicand | - | Desired Number of places | = | Value of 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 places in divisor | + | Assumed decimal places quotient | = | Assumed decimal places dividends | = | Value of 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:

| | | | | | | | | | | | | | | | | | | | | | | | | | | | 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 | 46 | 47 | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | MUL | | | | | | | PRICE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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

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

MULTIPLY AND ROUND

OP CODE

A

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

OP CODE

A

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 LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | | | A | | | | | | B | C | | | | | | | | | | | | | | | | | | |
| LABEL | | | | | | | | 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 |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

OP CODE

A

REMARKS

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:

OP CODE

A

LSR

5

DIV

200

Accumulator (dividend) = 100
 Memory location 200 (divisor) = 3
 Multiply-Divide Shift Register = 5
 Accumulator (quotient) = 333333
 Remainder = 1

= { printed with decimal = 33.33333
 printing of decimal provided by
 print mask.

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

Example 2:

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

| <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 LEN- GTH | PARAMETER | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|------|----|----------|----|----------------------|-----------|----|----|--|--|--|-------------------|---|--|--|---|--|--|--|
| LABEL | | | | | | | | | | | | | | | | | | | | | | | | | | OP. CODE | | | A | | | | | | | B | | | C | | | |
| LABEL | | | | | | | | | | | | | | | | | | | | | | | | | | OP. CODE | | 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 | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | TRA | | AREA | | | | | | | | | | | | | | | | | | |

| <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 LEN- GTH | PARAMETER | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|----|------|----|----------|----|----------------------|-----------|----|--|--|--|--|-------------------|---|--|--|---|--|--|--|
| LABEL | | | | | | | | | | | | | | | | | | | | | | | | | | OP. CODE | | | A | | | | | | | B | | | C | | | |
| LABEL | | | | | | | | | | | | | | | | | | | | | | | | | | OP. CODE | | 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 | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | TRM | | AREA | | | | | | | | | | | | | | | | | | |

| <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 LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|-------------------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|------------|----|----|----|----|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | A | | | | | B | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | + 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 | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | REM | | | | | | | | | | | | | | | | | | | |

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 LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|-------------------|----|----------------------|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----------|----|----|----|----|--|--|--|--|--|----------|--|--|--|--|
| | | | | | | | | A | | | | | B | C | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | + 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 | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | SLRO | | | | | 5 | | | | | | | | | | 6 | | | | |

| | | |
|----------------|----------|----------|
| <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 | 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.

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

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 |
| | | | | | | L | Ø | D | | | | | X | | | | | | | | | | | | | | | | | | |

| <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 Y R P | 1 2 3 4 - 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 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 |
| | | | | | | R | S | T | | | | | 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 Y R P | 1 2 3 4 - 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 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 | | | | | | | | | | | | | | | | | | | |
| 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 |

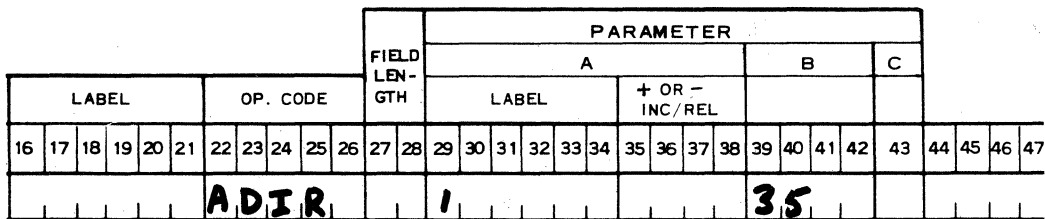
The LIR instruction loads the value contained in the B field into the index register indicated in the A parameter (1, 2, 3 or 4). The B parameter can be any positive value from 0 to 255. The prior contents of the index register are destroyed.

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> |
|--------------------------|----------|----------|
| DECREMENT INDEX REGISTER | 1-4 | 0-255 |

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 | | | | | | | | | | | | | | | | | | | | | LABEL | | | | | | | | | | | | | | | | | | | | | OP. CODE | | | | | | | FIELD LEN- GTH | PARAMETER | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|----------|--|--|--|--|--|--|----------------------|-----------|--|--|--|--|--|--|---|---|--|--|--|--|--|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | A | | | | | | | B | C | | | | | |
| A | | | | | | | + OR - INC/REL | | | | | | | 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 | | | | | | | 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

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 programed 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 programed 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 A & B Fields | Decimal Equivalent | | |
| | PKA PKB A & B field | Flag Instructions | |
| | | 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 Y R P | 1 2 3 4 | |
| SET | A K X Y R P | 1 2 3 4 | |
| RST | A K X Y R P | 1 2 3 4 | |
| CHG | A K X Y R P | 1 2 3 4 | |
| EX | A T K X Y R P | 1 2 3 4 | 1-4 |
| EXE | A T K X | | 1-4 |
| SK | A T K X Y R P | 1 2 3 4 | 1-4 |
| SKE | A T K X 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.

TABLE FOR VALUES

| Number desired in A field | “m” Value to be contained in Index Reg. | Number desired in B field | “n” Value to be contained in 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:

| <u>Parameters required:</u> | | <u>Index Register value required:</u> |
|-----------------------------|---|---------------------------------------|
| A = 8 | = | 128 |
| B = 3 | = | 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 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 |
| | | | | | MOD | | | | | 1 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | PC | | | | | A | | | | | | | | | | | | | | | | | | | | | |

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

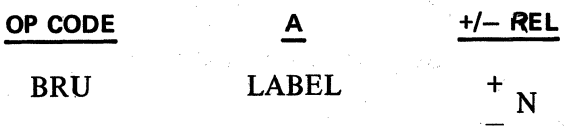
If PC A (A = index value of 65) is to be modified to print M (M = index value of 77), a value of 12 (77-65 = 12) is loaded into the index register #1. Index values are contained in Appendix D. The above remarks also apply to PC+, PC- and PCP.

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



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 LEN- GTH | | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|--------|----------------------|----|-----------|----|----|----|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | | | | A | | | | | | B | | | C | | | | | | | | | | | | | | |
| LABEL | | OP. CODE | | | | | FIELD LEN- GTH | | 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 | 48 |
| | | | | | | SET | | | | | | | X | | | | | | | | | | 2 | | | | | | | | | |
| | | | | | | AL | | | | | | | 2 | | | | | | | | | | | | | | | | | | | |
| | | | | | | BRU | | | | | | | SHIPTO | | | | | | | | | | | | | | | | | | | |
| | | | | | | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | SHIPTO | | | | | | | 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 | + 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:

| <u>LOCATION</u> | <u>ADDRESS</u> |
|-----------------|-----------------|
| 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. |
| | 3 | ? | ? | | |
| 48 | 0 | PRNC | PNS- | 14 | 0 |
| | 1 | | PC- | - | |
| | 2 | | PC+ | + | |
| | 3 | | SRJ | TKMAD | |
| | 3 | | ? | ? | |
| 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.

CPA

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> |
|--------------------------------------|----------------|----------|
| SKIP AND EXECUTE INSTRUCTIONS | 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 LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|----|----------------------|-----------|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | | | A | | | | | | | B | | | | C | | | | | | | | | | | | |
| LABEL | | | | | | | | 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 |
| MAX | | N.K | | | | | | 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

| | | |
|-----------------------------|----------------|----------|
| | <u>OP CODE</u> | <u>A</u> |
| 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.

| | | |
|--------------------------|----------------|----------|
| | <u>OP CODE</u> | <u>A</u> |
| 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 | | | | | | | | | | | | B | | C | | | | | | | | | | | |
|----------|----|----------|----|---------------|----|-----------|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | A | | | | | | + 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 |
| NUMRICNK | | | | | | 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 LEN- GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|----------------------|-----------|----|----|----|----|----|-------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| | | | | | | | A | | | | | | | B | | | | C | | | | | | | | | | | | | | |
| LABEL | | | | | | | 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 | |
| | | | | | | 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 | 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 | |

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 LEN-GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|----|---------------|-----------|----|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | | | A | | | | | | | B | | | | | | C | | | | | | | | | |
| LABEL | | | | | | | | 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 |
| | | | | | | 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 LEN-GTH | PARAMETER | | | | | | | | | | | | | | | | | | | | | | |
|-------|----|----------|----|----|----|----|---------------|-----------|----|----|----|----|----|----|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | | | | | | | | A | | | | | | | B | | | | | | C | | | | | | | | | |
| LABEL | | | | | | | | 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 |
| | | | | | | 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 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 | |
| | | | | | | 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

| LABEL | | OP. CODE | | | | | FIELD LENGTH | | 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 |
| 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 |
| | | 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:

1. ACCUMULATOR FLAGS

A — [— Sign
S Special
C Per Hundred
M Per Thousand

2. KEYBOARD BUFFER FLAGS

B — [2 KB Buffer Filled
3 KB Buffer Empty

3. DATA COMM FLAGS

D — [1 Received TR# Not
Equal Expected TR #
2 Message Received
3 Transmit Ready

4. OCK FLAGS

K — [1 OCK - 1
2 OCK - 2
3 OCK - 3
4 OCK - 4

5. L FLAGS (SHIFT REG)

L — [1
2
3
4

6. PUNCH FLAGS

P — [1 Media Not Present
2 Echo Check
3 Tape Supply
4 Punch Off

7. READER FLAGS

R — [1 Reader Condition
2 Message Received
3 Transmit Ready
4 Invalid Code

8. STRIPE LEDGER FLAG

S — [∅ Not Used
W Write Error
R Read Error
F Filled Sheet

9. TEST FLAGS

T — [∅ Overflow
L Forms Limit
I Index Register
U Unassigned

10. TELLER LOCK FLAGS

V — [1 Teller 1
2 Teller 2
3 Supervisor
4 Not Used

11. PASSBOOK FLAGS

W — [1 Passbook Fold
2 Last Print Line
3 Not Used
4 First Print Line

12. GENERAL PURPOSE FLAGS

X — [1
2
3
4
Y — [1
2
3
4

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 |

STOP

OFF

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 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 | | |
| | | | | | | CDC | | | | | | | 6 | | | | | | | | | | | | | | | | | | | | |

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

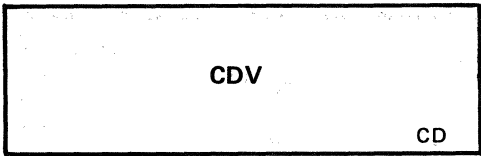
If the Accumulator contains:

| | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|------------------------|
| 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 } | LPNR } | TABLE } | | | LOAD CD & P MASK TABLE } |
| | 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

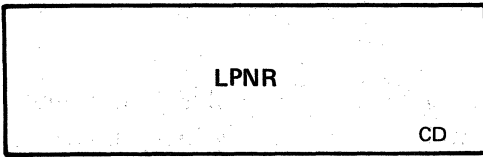
| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| CDV | 1-15 | 0-9 |

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.



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 IN 15 & 14 | | RELATIVE LOCATION OF 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.

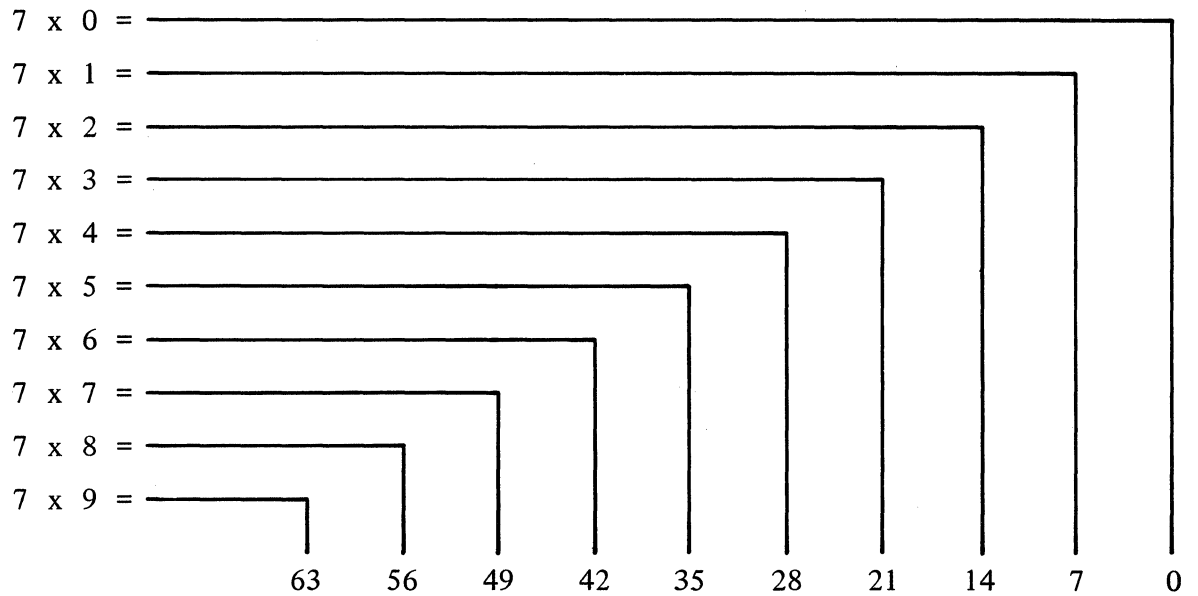
| | | | | | | |
|---------------------------------|------------------------------|---|---|---|---|---|
| Integer | 4 | 3 | 2 | 2 | 7 | 7 |
| Assigned Value From Table | $4 + 9 + 3 + 2 + A + 5 = 33$ | | | | | |
| Remainder | 1 | | | | | |
| Total Sum of Assigned Values | $33 + 1 = 34$ | | | | | |
| Next 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).

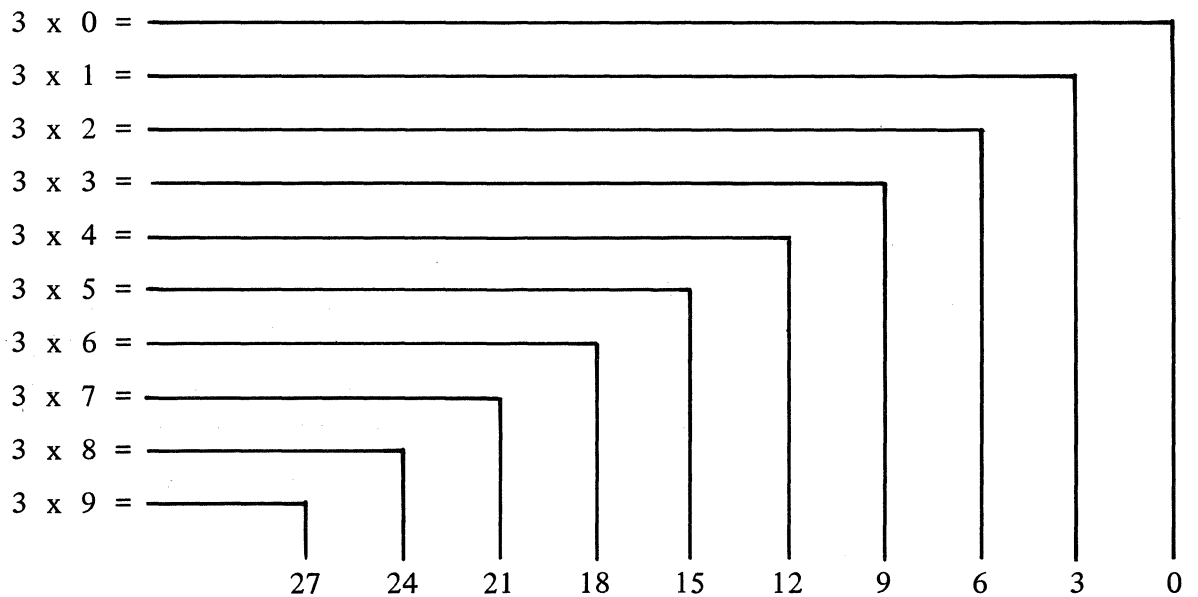


Minus Next Lowest

Multiple of Modulus -55 -55 -44 -33 -33 -22 -11 -11 -0 -0

1st Table Entry = 8 1 5 9 2 6 A 3 7 0

2nd Table Entry (Weighted 3).

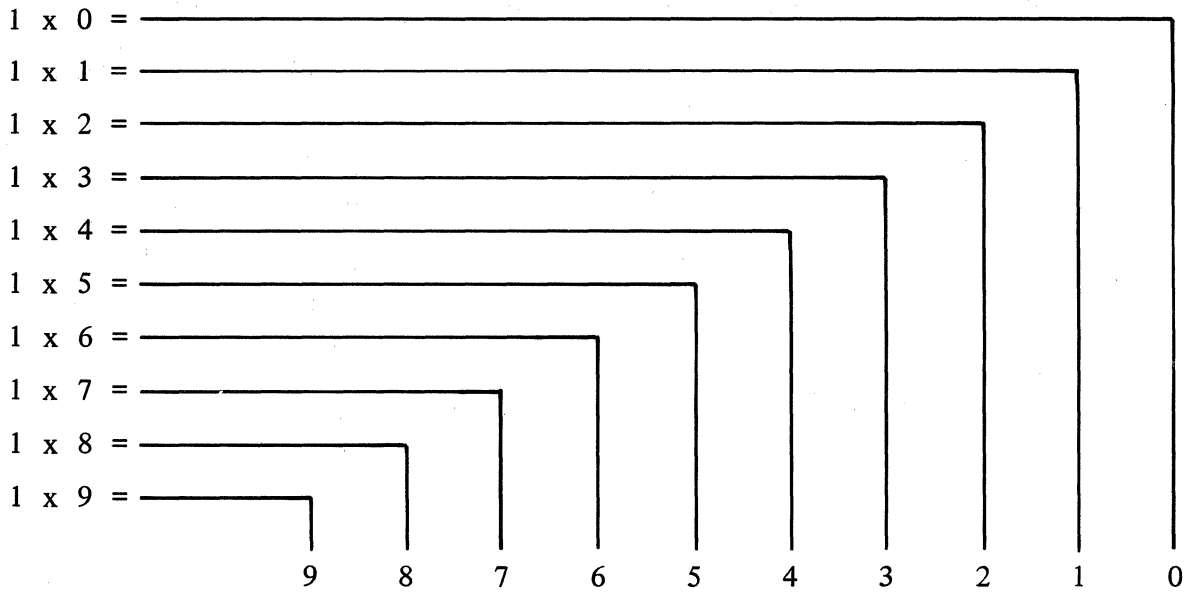


Minus Next Lowest

Multiple of Modulus -22 -22 -11 -11 -11 -11 -0 -0 -0 -0

2nd Table Entry = 5 2 A 7 4 1 9 6 3 0

3rd Table Entry (Weighted 1).



| | | | | | | | | | | |
|---------------------|----|----|----|----|----|----|----|----|----|----|
| Minus Next Lowest | | | | | | | | | | |
| Multiple of Modulus | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 | -0 |
| 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 programming, 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).

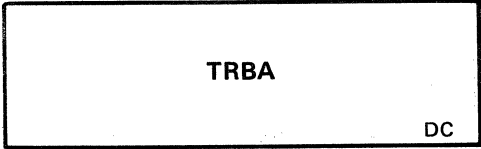
Other characters will be considered as invalid, will cause the "S" flag of the Accumulator to be set, will count as a code transferred, but the instruction will not be terminated.

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 (Sign is lost) |
| TRBA 5 | 1234 + ABC | 20000000000012341 (S flag is set by transfer of 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



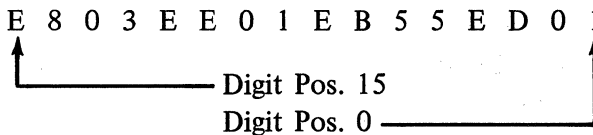
table, column 3 has 16 entries: 0-9 and the hexadecimal digits A through E. This information is useful when, for instance, an "A" is desired in the Accumulator as a result of a TRBA instruction. The central processor would send to the TC an USASCII equivalent of a colon (:). In USASCII code, it is "3,A."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



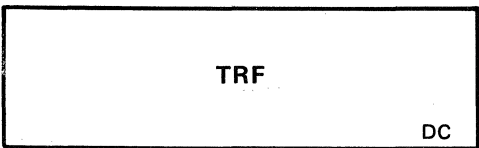
In this instance, the "E's," "B's," and "D's" in the Accumulator resulted from a 3,E, and a 3,B, and a 3,D in memory which are valid codes for the TRBA instruction. The "E" in the Accumulator is, in reality, a hexadecimal 14, the "B" a hexadecimal 11, and the "D" a hexadecimal 13.

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

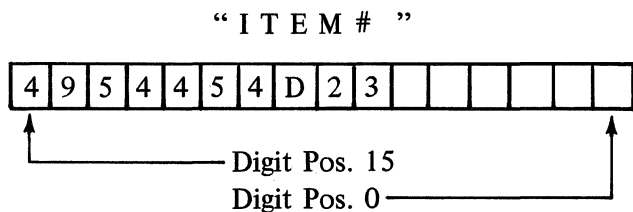
A

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:

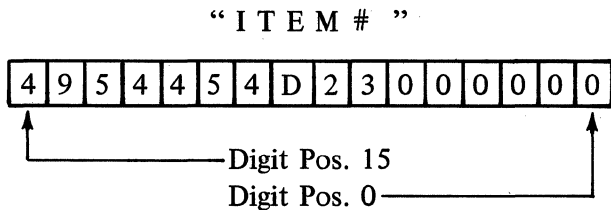
First word of Receive Buffer:



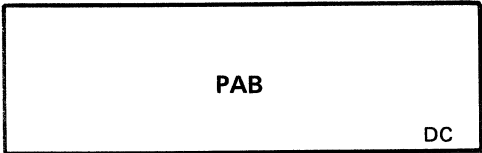
Example:

| | <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:

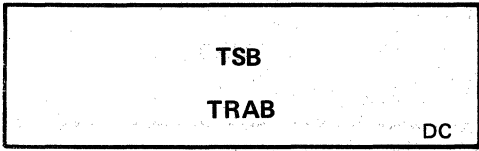
| | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> |
|-------------------------------|--------------|----------------|----------|
| ESTABLISH RECEIVE RECORD AREA | RECEIV | ESTB | |
| LOAD RECEIVE BUFFER REGISTER | | LRBR | RECEIV |
| PRINT ALPHA | | PAB | 15 |

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 |

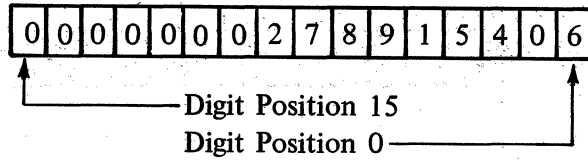
The TRAB instruction will transfer up to 15 numeric digits (4 bits) from the Accumulator into the memory location designated by the last LKBR instruction, placing the digits into memory as 8 bit alpha characters beginning with the current position of the SCP.

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.

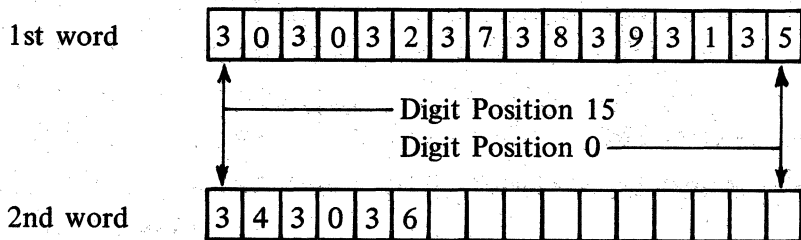
Example 1:

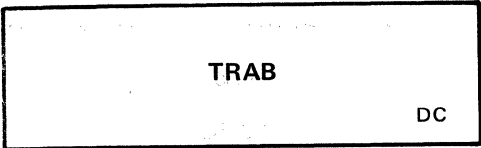
| | | | | |
|-----------------------------|--------------|----------------|----------|----------|
| | <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
| LOAD KEYBOARD BASE REGISTER | | LKBR | SEND | |
| TRANSFER ACCUMULATOR | | TRAB | 10 | 1 |

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:



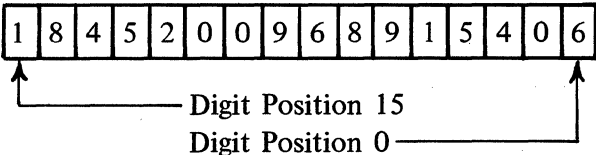


The transfer could also have been directly to the Data Communications Transmit Buffer.

Example 2:

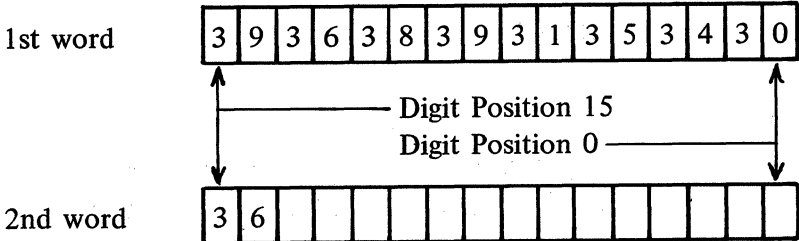
| | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|-------------------------------------|----------------|----------|----------|
| LOAD KEYBOARD BASE REGISTER | LKBR | | |
| TRANSFER ACCUMULATOR TO LAST "LKBR" | TRAB | 10 | 0 |

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 | |
| TRF | |
| TRCB | DC |

If the Accumulator appears like this prior to execution:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | 0 | 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 |
| TKM |
| 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.

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

| | |
|---------|----|
| D FLAGS | |
| RSA | 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

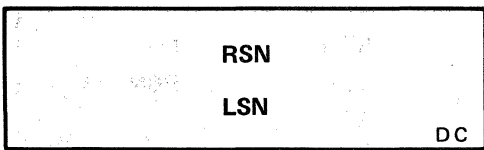
GP 300 has a group of instructions, which allow the programmer to assume some firmware responsibilities. An example is the transmission number that is part of the header portion of a message. This number is usually calculated by firmware and can be an important programming consideration. There are two 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.

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



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

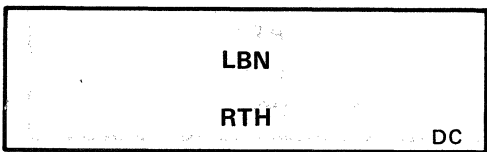
The LGN instruction transfers the contents of the Accumulator into the Expected Group Transmission Number Register. The Expected Group Transmission number may be up to 3 USASCII numeric characters (left justified) in length and although the entire Accumulator is transferred, the remaining locations are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

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 | <u>60 CH-</u> |
| | SCP Limit | 195 CH |

SCP is hexadecimal. Therefore,
195 = C 3 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 | <u>63 CH-</u> |
| | SCP Limit | 192 CH |

Therefore, SCP = C 0

| | | | | |
|-----|----------|----|---|----------------------------|
| 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 |

| | |
|-----------------------|-----------------------|
| LOAD POINTER REGISTER | <u>OP CODE</u> 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 X 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.

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.

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 used in a three (3) TR # system. | TR1 | 88 |
| | TR2 | 89 |
| | TR3 | 8A |
| Must be used in a two (2) TR # system. | TR1 | 89 |
| | 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

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.

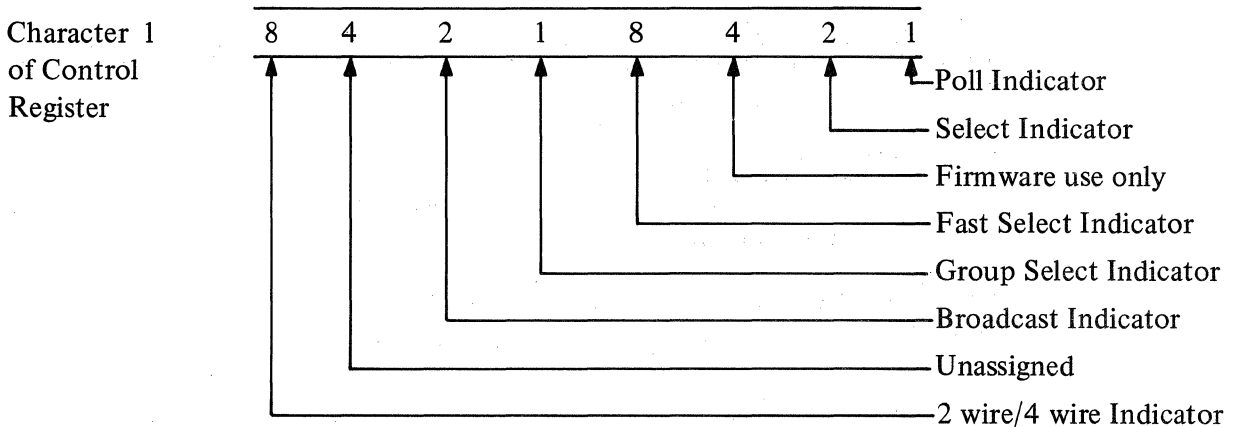
CTCC

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 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.

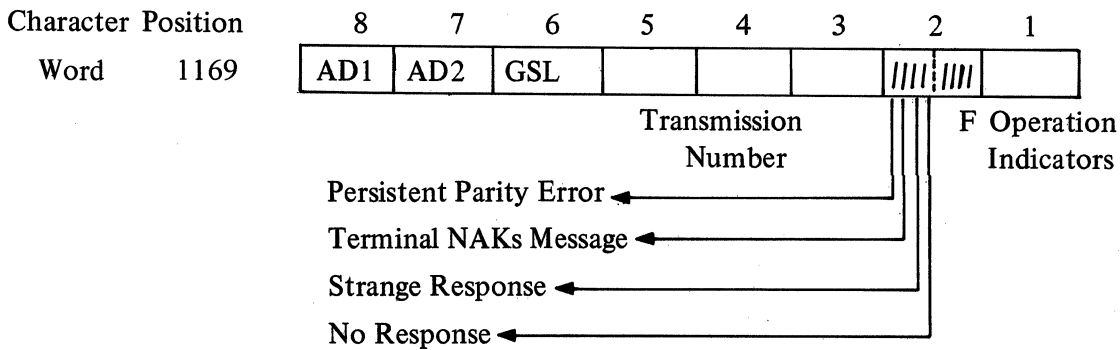
All three flags are available for interrogation through the regular Skip/Execute instructions.

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.

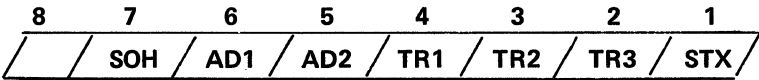
d. Header Register

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

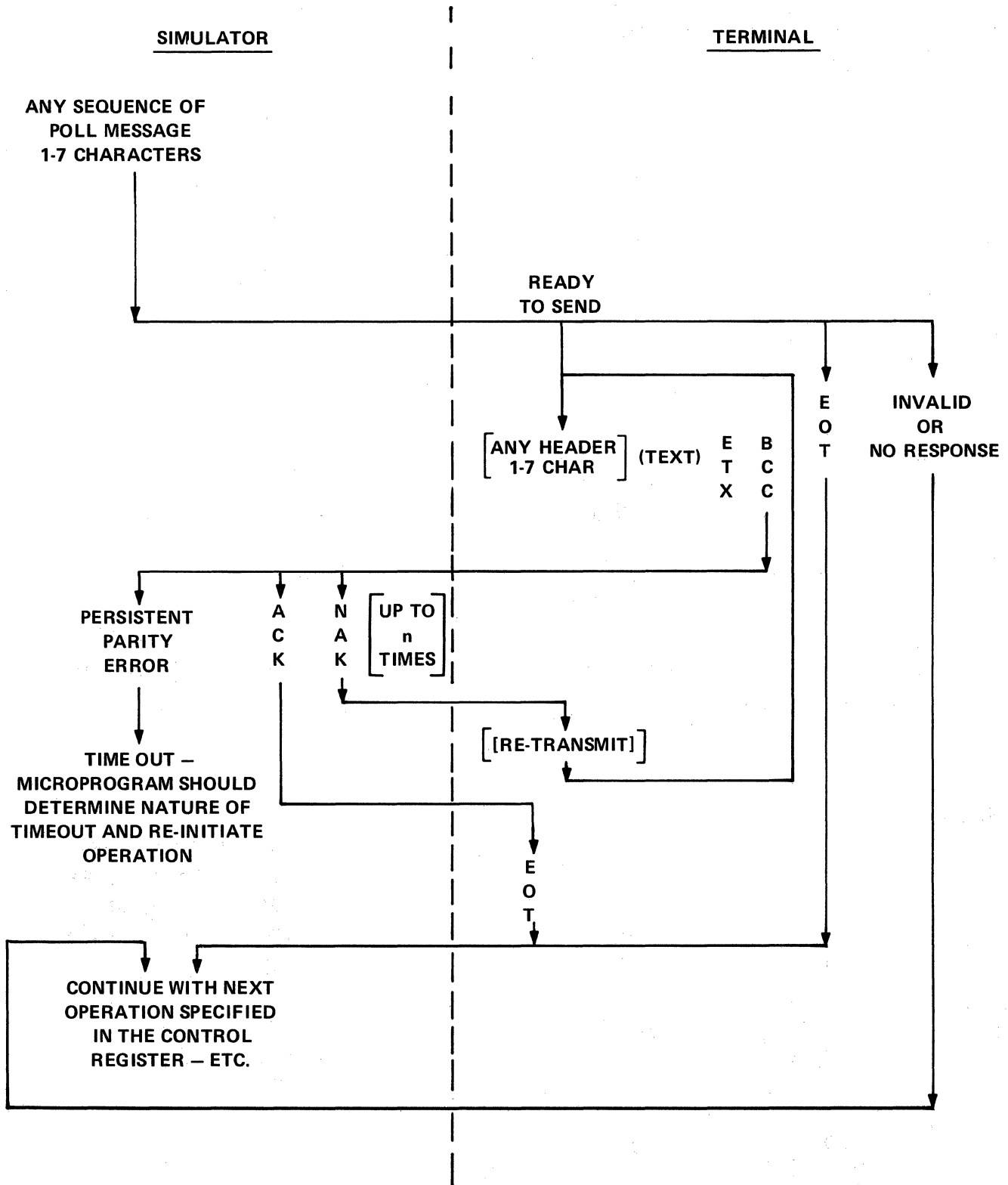


Chart 1. Poll

CTCC

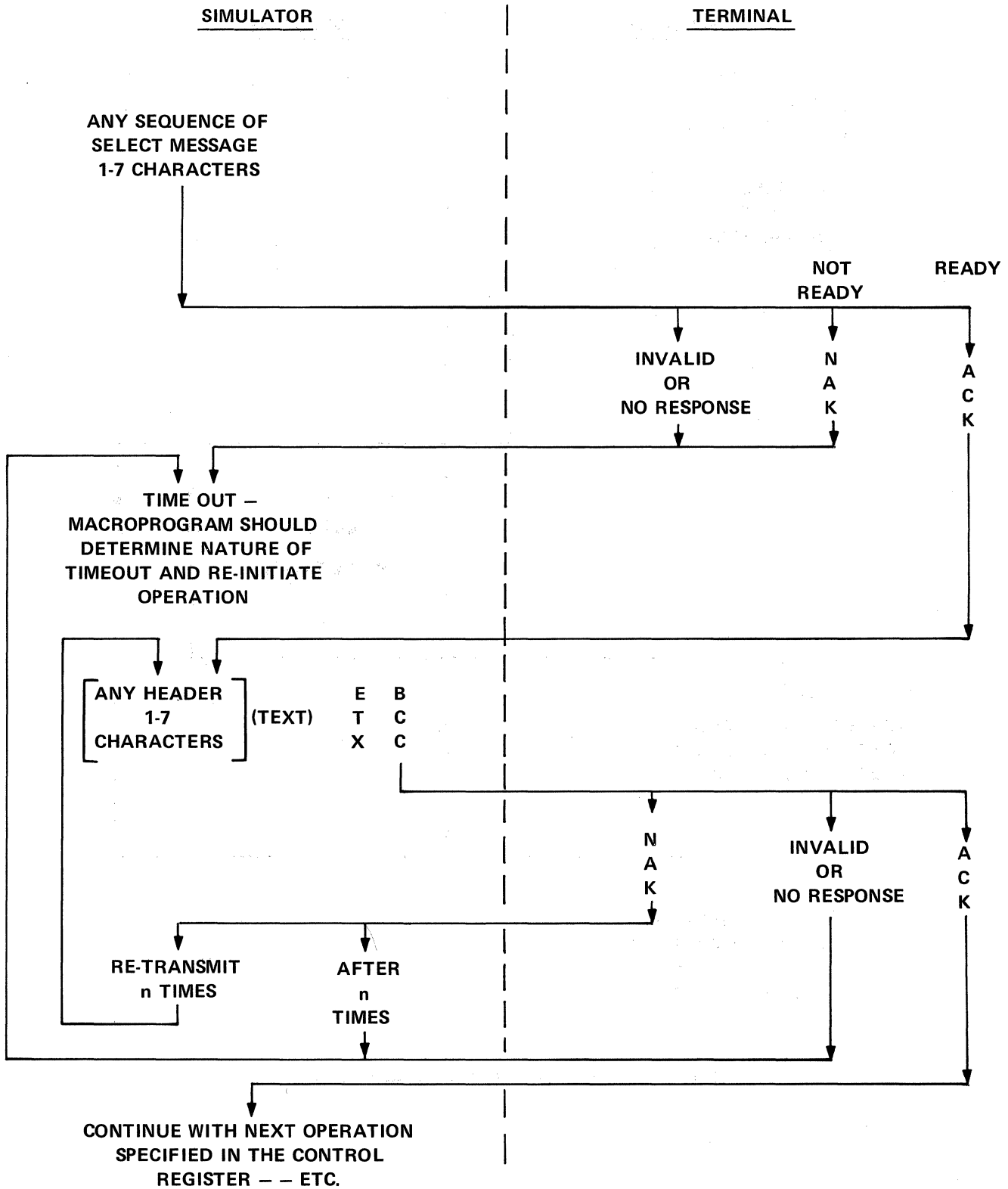


Chart 2. Select

CTCC

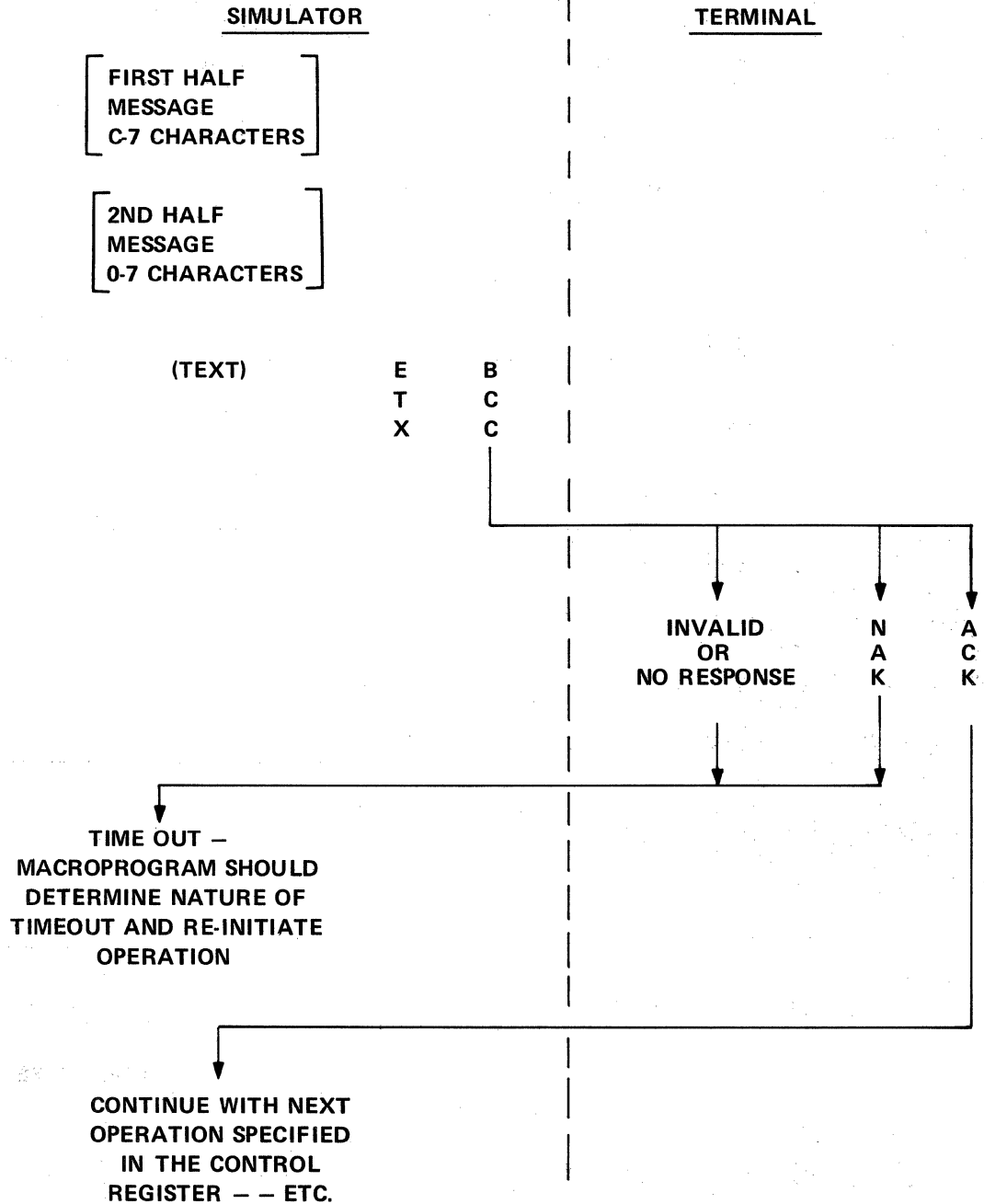


Chart 3. Fast Select, Group Select, and Broadcast Select

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).

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 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

| | |
|---------------------|-----------------------|
| RELEASE MEDIA CLAMP | <u>OP CODE</u> 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 XTKM 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.

| |
|-----|
| XC |
| XPN |
| PT |

PUNCH CODE

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| 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 (b₈, b₇, b₆, b₅, having decimal values of 8, 4, 2, 1 respectively); the B parameter represents the decimal value of the low order 4 bits (b₄, b₃, b₂, b₁, 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 ₈ | b ₇ | b ₆ | b ₅ | b ₄ | b ₃ | b ₂ | b ₁ |
|----------------------------------|-------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 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.

PRINT AND PUNCH NUMERIC

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| 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.

| | |
|-------|-------|
| XPNS- | XPNS+ |
| XN | LXC |
| | PT |

| | <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.

| | | |
|-----|-----|-----|
| LCD | RCD | CRD |
|-----|-----|-----|

2.17 – 80-COLUMN PUNCHED CARD INPUT INSTRUCTIONS

With either the A 595 or A 596 80-column 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>A 595</u> | <u>OP CODE</u> |
| READ CARD | RCD |

When an L/TC is used with an A 595 card reader, the RCD instruction reads a single 80-colum punched card into words 1 through 10 of memory. All 80 columns are read and placed into memory including blank card columns.

| | | |
|--------------|----------------|----------|
| <u>A 596</u> | <u>OP CODE</u> | <u>A</u> |
| READ CARD | RCD | 0-255 |

When an A 596 card reader is used as the input device for an L/TC, the RCD instruction reads a single 80-column punched card into the next 10 words of memory beginning with the word specified by the A parameter. Data can be read into any word in Block 0 except word 0. A parameter of 0 defaults to word 1. The L programmer may use the TRCA and TRCM instruction to access data in Track 0 only

(words 1-31). The TC programmer can use the data comm instructions used in processing messages received to access data at any location in Block 0. If the TSB instruction is to be used, the data to be transmitted must begin on a Track boundary (word 32, 64, 96, . . . 224) and should be contiguous within the track.

In general, whenever a programmer attempts to force data to be read into memory above word 255, the RCD instruction is terminated, the overflow flag is set, and all data following that which was placed in word 255 is lost. The next read instruction automatically resets the overflow flag.

During the execution of each RCD instruction, the original 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, as it will be altered during the execution of the RCD instruction.

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.

| | |
|------|-----|
| LCFR | PBA |
| | CRD |

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

OP CODE A
 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 2.01.04 for explanation of CDF 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 |

OP CODE A
 PRINT ALPHA FROM CARD READ AREA 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> |
|--------------|----------------|----------|-----------------------------|
| | LCFR | CRDTAB | |
| | PBA | 2 | Print second field on card. |
| | NOTE | | Card cols. 3-10 |
| CRDTAB | CDF | 1 | 2 |
| | CDF | 3 | 8 |
| | | | Card cols. 1-2 |
| | | | 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.

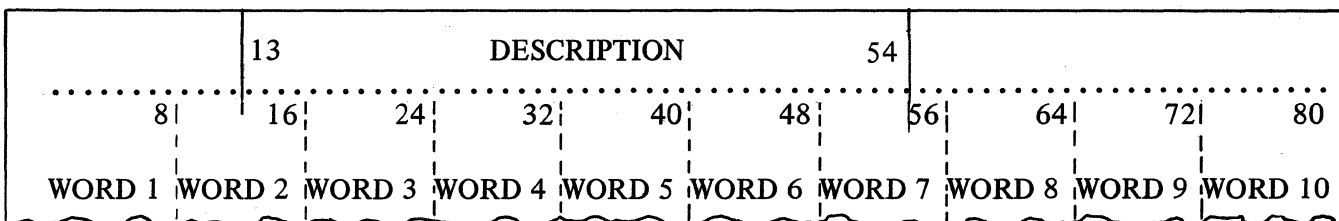
If a "12" or "0" overpunch is present in any of the card columns of the field being transferred, the Invalid Code Flag (R4) is set and the corresponding indicator light is turned on. An unknown digit will be transferred to the Accumulator. The flag is reset and the indicator is turned off at the beginning 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).



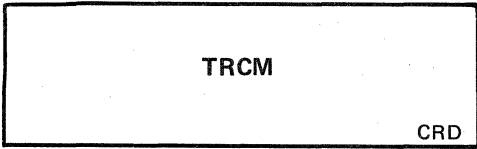
| <u>LABEL</u> | <u>INSTR</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u> |
|--------------|--------------|----------|----------|---------------------|
| FIELDS | CDF | 13 | 20 | SHORT DESCRIPTION |
| | CDF | 13 | 28 | MEDIUM DESCRIPTION |
| | CDF | 13 | 42 | MAXIMUM DESCRIPTION |

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 | | |
| | | LLR | 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 DESCRIP 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 |

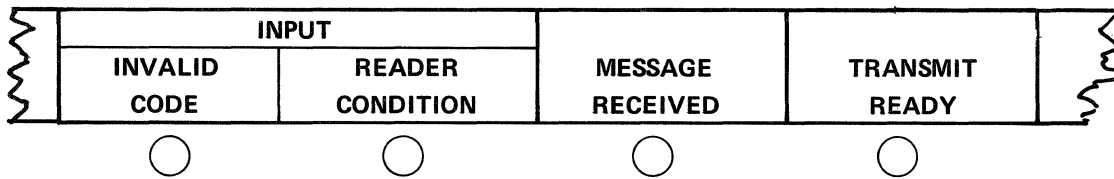
The TRCM instruction transfers the field specified by the format number in the A parameter to a memory location starting with the word designated by the prior use of the LKBR instruction. The instruction is terminated after transferring the number of characters specified by the field length in the format. 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.

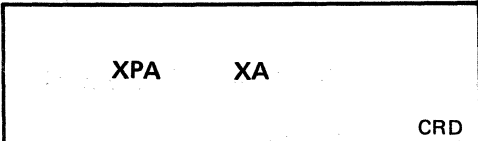
| | <u>OP CODE</u> | <u>A</u> |
|-----------------------------------|----------------|--------------------------|
| TYPE INTO MEMORY, PUNCH AND PRINT | XTKM | 0-150 15½" forms handler |
| | XTKM | 0-255 26" forms handler |

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, 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 D, .D D: | Print Character regardless of significance | Punch Character regardless of significance |
| X .X | Trailing zero suppression | |
| C .C | Leading zero & trailing zero suppression | |
| Z Z, Z: S | Print if: (1) Accum digit not zero (2) A non-zero digit has been printed | Punch if: (1) P is Set (2) Accum digit not zero (3) A non-zero digit has been punched |
| | Print only if Accum digit not zero | |
| 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.

| |
|----------------------------|
| XPN XPNS— |
| 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 | |
| | <u>XC</u> | |
| | | CRD |

120981

PRINT & PUNCH NUMERIC, SHIFT RIBBON IF PLUS

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| 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

PUNCH NUMERIC, NON-PRINT

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| XN | 0-14 | 0-15 |

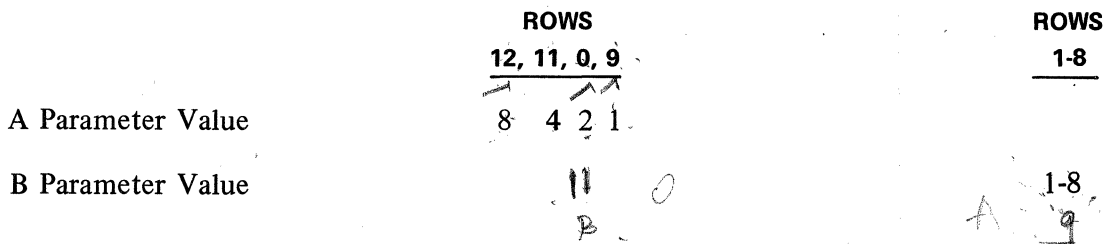
The XN instruction is the same as the XPN instruction except that no printing occurs. A mask word is used with this instruction since it controls punching, and may be the same mask word used with other Print Numeric instructions as there would be no affect on the non-print characteristic of XN. If the punch is off-line, XN is executed as a NOP instruction.

PUNCH CODE

| <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|----------------|----------|----------|
| 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.



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.

| |
|-----|
| LXC |
| SKP |
| CRD |

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.

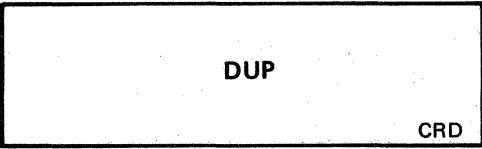
| | <u>OP CODE</u> | <u>A</u> |
|---------------------------|----------------|----------|
| LOAD PUNCH COUNT REGISTER | LXC | 1 |

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. The card containing the incorrect data should be released and duplicated to the beginning column of the reset field. The new entry may then be made without losing synchronization between the L/TC and the card punch. Backspacing will generally result in new data being punched over incorrect data producing incorrect cards.

| | <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.

| | <u>OP CODE</u> | <u>A</u> |
|--------------------------|----------------|----------|
| DUPLICATE THROUGH COLUMN | DUP | 1-80 |

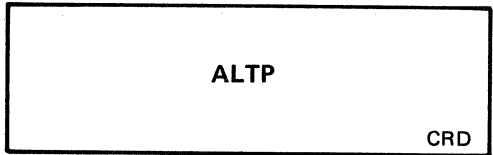
The DUP instruction causes data from the card in the Read Station to be punched (duplicated) into the corresponding columns of the card in the punch station. The duplication function starts at and includes the card column at which it is initiated, and continues through the card column specified in the 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:

| | |
|---------------------------|----------------|
| | <u>OP CODE</u> |
| ALTERNATE STACKING POCKET | ALTP |

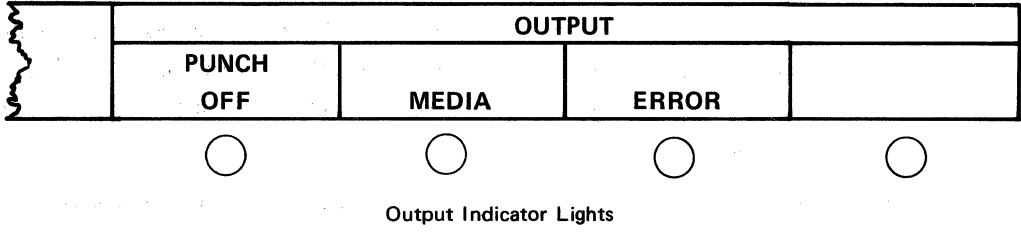
The ALTP instruction causes the card in the Punch Station to be routed to the Alternate Stacking Pocket after it has been released from both the Punch Station and the Read Station. The ALTP instruction must be executed while the card is still in the Punch Station, and prior to any instruction that will 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 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 |
| 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

| | |
|--------------|----------------|
| WRITE RECORD | <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.

The initial phase of execution will open the handler if closed. The data is written while the magnetic record is being ejected.

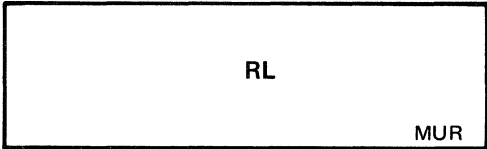
If a write error occurs, the W (write error) flag is set. All error recovery routines are programmatic.

Example:

| 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 |
| | | | | | | SRJ | | | | | | | WRITEL | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | 3 | | | | | | | | | | | | 0 | | | | | | |
| | | | | | | WRITE | RL | | | | | | S | | | | | | | | | | | | R | | | | 2 | | |
| | | | | | | | EX | | | | | | 1 | | | | | | | | | | | | | | | | | | |
| | | | | | | | PKA | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | BRU | | | | | | | | | | | | | | - | 3 | | | | | | | | | |
| | | | | | | WRITEL | WL | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | EX | | | | | | S | | | | | | | | | | | | W | | | | 3 | | |
| | | | | | | | PKA | | | | | | 3 | | | | | | | | | | | | | | | | | | |
| | | | | | | | DEC | | | | | | | PKA | 3 | = | BRU | WRITE | | | | | | | | | | | | | |
| | | | | | | | TK | | | | | | 0 | | | | | | | | | | | | | | | | | | |
| | | | | | | | BRU | | | | | | | | | | | | | | - | 2 | | | | | | | | | |
| | | | | | | | SRR | | | | | | 1 | | | | | | | | | | | | | | | | | | |

| <u>LABEL</u> | <u>OP CODE</u> | <u>A</u> | <u>+1-</u> | <u>B</u> | <u>C</u> | <u>REMARKS</u> |
|--------------|----------------|----------|------------|----------|----------|---------------------------|
| | 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



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 striped or non-striped record and align it to the number contained in the Stripe Count Register. Since this parameter does not attempt to read the stripe, the contents of the Striped Record 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 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 |
| 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.

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:

| L A B E L | | | | | | | | | | | | | | | | | | | | | | | | | | O P . C O D E | | F I E L D L E N - G T H | P A R A M E T E R | | | | | | | | | | | | | |
|-----------|----|----|----|----|----|---------------|----|----|----|----|----|-------------------------------|----|-----------|----|----|----|----|----|----|--------------------------|----|----|----|----|---------------|----|-------------------------------|-------------------|----|----|--|--|--|--|---|--|--|--|---|--|--|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | A | | | | | | | B | | | | C | | |
| L A B E L | | | | | | O P . C O D E | | | | | | F I E L D L E N - G T H | | L A B E L | | | | | | | + O R - I N C / R E L | | | | | | | | | | | | | | | | | | | | | |
| 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 | A | | | | F | I | E | L | D | S | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | P | A | S | | | | | N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | U | B | P | O | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | F | I | E | L | D | S | S | L | A | | | 1 | | | | | | | | | 3 | 1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | S | L | A | | | 6 | 3 | | | | | | | | 4 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | S | L | A | | | 6 | 7 | | | | | | | | 7 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | S | L | A | | | 7 | A | | | | | | | | 1 | 1 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | S | L | A | | | 8 | S | | | | | | | | 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 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 TO ACCUMULATOR | ADB | 1-64 | 0-1 |
| SUBTRACT MAGNETIC RECORD AREA 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 LEN- 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 | | | | | | | | | |

| <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 |
| 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 | | | | | | 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 |
| 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. |

| | | |
|------|------|-----|
| TASB | TMSB | LA |
| | | MUR |

| | <u>OP CODE</u> | <u>A</u> | <u>B</u> |
|---|----------------|----------|----------|
| TRANSFER FROM ACCUMULATOR TO MAGNETIC RECORD AREA | TASB | 1-64 | 0-1 |
| TRANSFER ALPHA FROM MEMORY TO 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 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 |
| T | A | S | B | | | | 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.

Example 1:

| 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 | | | | 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 | EL |

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 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 | | |
| EL | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| <u>OP CODE</u> | <u>A</u> | <u>B</u> | <u>REMARKS</u> |
|----------------|----------|----------|----------------|
| EL | | | EJECT RECORD |

| | |
|-----|------|
| RET | LSCR |
| | MUR |

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 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 |
| 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 |

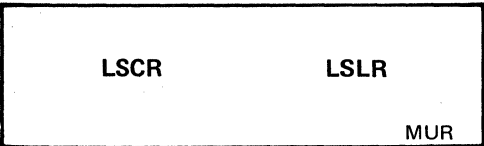
The LSCR instruction loads the Stripe Count Register with the value stored in the "A" parameter. The "A" parameter value may vary from 1 to 46 (46 is maximum number of posting lines on an 11" Magnetic Record).

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.

The Stripe Count Register must be reset to its initial value by the programmer when a filled sheet condition is detected. If the Stripe Count Register is not reset it will continue being bumped by every AR, ALR, and ARTO instruction until it reaches a value of 255. It is then reset to zero. The filled sheet flag is set only when the stripe count register is incremented one beyond the stripe limit register. At all other times the filled sheet flag is reset.

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.



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:

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 LOC.</u> | <u>OP CODE</u> | <u>A PAR</u> | <u>B PAR</u> | <u>C 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 VALUE | ACCUMULATION DIGIT POSITION | ABBREVIATION 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. | | | | | | | | | | | | | | | | |
| (See Above) | | | 2 | | | 1 | 4 | 3 | | | 2 | | | 1 | 4 | 3 |

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 | 4 | 5 | 4 | 8 | 4 | 5 | 1 | 2 | 5 | 3 | 5 | 4 | 5 | 7 | 2 | 3 |
| 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

2.21.03

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 respectively 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 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.

TCT

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 programming 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 programming 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.

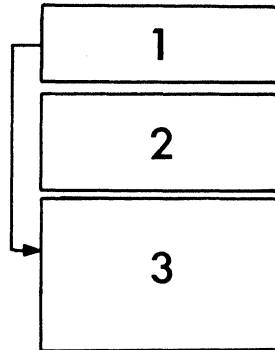
PROGRAMING CONSIDERATIONS

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.

**SYMBOLIC
PROGRAMING PROCEDURES**

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> |
|--------------|----------------|----------|----------|----------|
| | LLR | 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.

**SYMBOLIC
PROGRAMING PROCEDURES**

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 | | | |

**SYMBOLIC
PROGRAMING PROCEDURES**

| <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 PROGRAMING EXAMPLE

PROBLEM

Examine the Program Definition Chart and Worksheet located on pages 4-3 through 4-6.

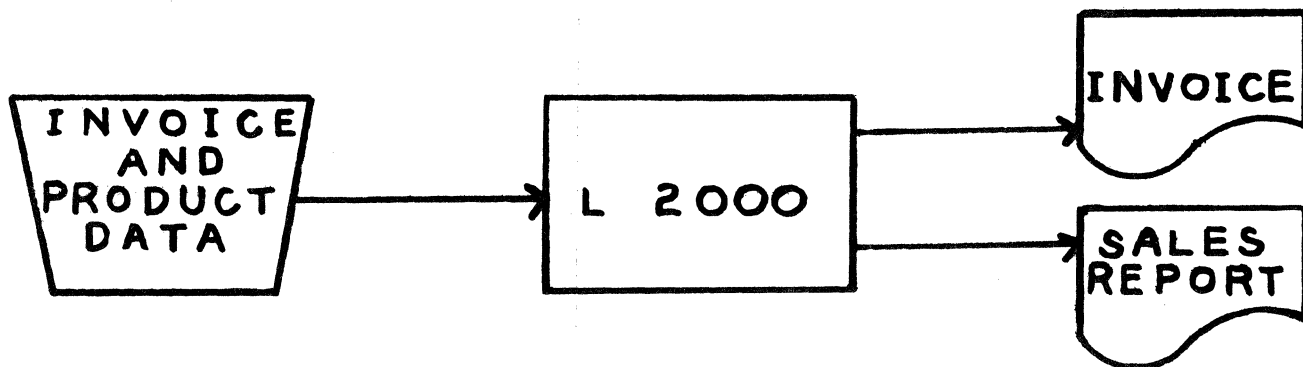
SOLUTION

The proposed solution is located from page 4-7 to page 4-32.

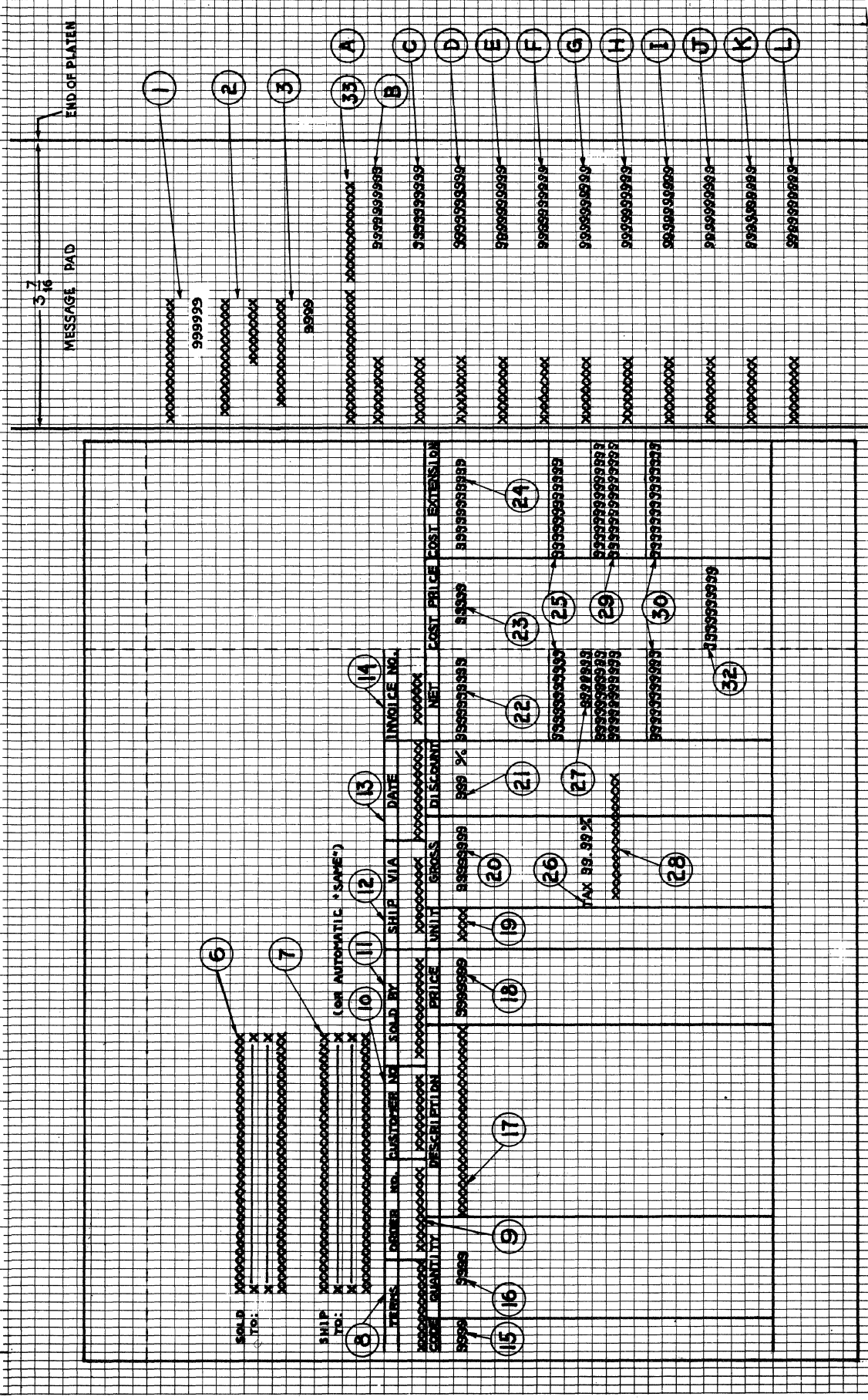
SOLUTION INDEX

| | |
|--|------|
| General Systems Flowchart | 4-2 |
| Program Definition Worksheet..... | 4-3 |
| Program Definition Charts..... | 4-4 |
| Sample Coding Forms..... | 4-7 |
| Assembler III Program Listing | 4-33 |
| Sample Output..... | 4-69 |
| Cross Reference between Assembler III Output and Program Definition Chart | 4-70 |

SAMPLE BILLING PROGRAM
GENERAL SYSTEMS FLOWCHART



0 .5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10 10.5 11 11.5 12 12.5 13 13.5 14 14.5 15 15.5 16 16.5



| | | INPUT | | | | PROCESSING | | | | OUTPUT | | | | | |
|-----|--|------------------------------|------------------------|-----|-----|------------------|---|---|-------------------------|-------------------------|-----|-----|-------------------------------|---|--------------|
| NO. | DATA DESCRIPTION OR OPERATION | ALPHANUMERIC (A) NUMERIC (N) | FIELD DEFINITION INPUT | | | | INPUT SOURCE Keyboard (KB), Specify Media, Specify MB, OCK, PK, etc., only if particular one required; Punched Card (PC); Edge Punched Card (EPC); Punched Tape (PT); Striped Ledger (SL); Specify Reader; Memory (M); Data Com Receive. | PROCESSING REQUIREMENTS Accumulations; Formulas; Extensions; Store for later use; Calculations; Formatting for output. | P (✓) N (✓) R (✓) T (✓) | FIELD DEFINITION OUTPUT | | | | OUTPUT DETAIL Print: Specify Printer and Format Console (CP), Line Printer (LP); Punch: Specify Adjunct, Format, Special codes, etc.; Striped Ledger; Data Com Transmit. | INSTRUCTIONS |
| | | | Fixed Size | Min | Max | Norm | | | | Fixed Size | Min | Max | Norm | | |
| 1 | INVOICE NUMBER | N | 0 | 6 | 5 | KB | STORE FOR AUTOMATIC PRINT INDEX ZERO IF NOT AUTOMATIC | ✓ | | 0 | 6 | 5 | | | |
| 2 | DATE | A | 6 | 12 | 11 | KB | STORE FOR AUTOMATIC PRINT | ✓ | | 6 | 12 | 11 | | | |
| 3 | TAX RATE | N | 0 | 4 | 2 | KB | STORE FOR CALCULATION | ✓ | | 0 | 4 | 2 | | | |
| 4 | CLEAR DAILY TOTALS | | | | | PK SELECTION | | | | | | | | | |
| 5 | CHOICE OF INVOICE A) COST AND SELL B) STANDARD (SELL ONLY) | | | | | SELECT OCK OR PK | COMPUTE AND PRINT COST EXT. SELL EXTENSION ONLY | | | | | | | | |
| 6 | SOLD-TO INFORMATION | A | 0 | 3 | 1 | KB | 1-4 LINES, STORE NAME (LINE 1) FOR POSSIBLE CONTINUATION PAGE | ✓ | | 0 | 3 | 1 | | | |
| 7 | SHIP-TO INFORMATION | A | 0 | 3 | 1 | KB | 1-4 LINES OR AUTOMATIC "SAME" SELECTED BY OCK IN *6 | | | | | | | | |
| 8 | TERMS | A | 0 | 11 | 11 | KB | | ✓ | | 0 | 11 | 11 | | | |
| 9 | ORDER NUMBER | A | 1 | 10 | 10 | KB | | ✓ | | 1 | 10 | 10 | | | |
| 10 | CUSTOMER NUMBER | A | 1 | 9 | | KB | | ✓ | | 1 | 9 | | | | |
| 11 | SOLD BY | A | 0 | 12 | | KB | | ✓ | | 0 | 12 | | | | |
| 12 | SHIP VIA | A | 1 | 9 | | KB | | ✓ | | 1 | 9 | | | | |
| 13 | DATE | | | | | M | | ✓ | | 6 | 12 | 11 | | | |
| 14 | INVOICE NUMBER A) IF STORED B) IF NOT STORED | | | | | M | | ✓ | | 0 | 6 | 5 | | | |
| 15 | PRODUCT NUMBER | N | 0 | 4 | 4 | KB | MUST BE LESS THAN 5,000 (PROVIDE ERROR INDICATION) | ✓ | | 0 | 4 | 4 | | | |
| 16 | QUANTITY | N | 0 | 7 | 3 | KB | ENABLE RE KEY; OPTION OF DISCOUNT OR NO DISCOUNT BY OCK SELECTION | | | | | | RED AND *-- IF MINUS | | |
| 17 | PRODUCT DESCRIPTION | A | 0 | 23 | 15 | KB | PERMIT MULTIPLE LINES | ✓ | | 0 | 23 | 15 | 23 CHARACTER MAXIMUM PER LINE | | |

CUSTOMER _____ APPLICATION _____
 EQUIPMENT _____
 SALESMAN _____ BRANCH _____ DATE _____



PROGRAM DEFINITION CHART

| NO | DESCRIPTION OR OPERATION | ALPHANUMERIC (A) | FIELD DEFINITION INPUT | | | | INPUT SOURCE Keyboard (KB), Specify Media, Specify MB, OCK, PK, etc., only if particular one required; Punched Card (PC); Edge Punched Card (EPC); Punched Tape (PT); Striped Ledger (SL); Specify Reader; Memory (M); Data Com Receive. | PROCESSING REQUIREMENTS Accumulations; Formulas; Extensions; Store for later use; Calculations; Formatting for output. | PRINT (✓) | PUNCH (✓) | FIELD DEFINITION OUTPUT | | | | OUTPUT DETAIL Print: Specify Printer and Format Console (CP), Line Printer (LP); Punch: Specify Adjunct, Format, Special codes, etc.; Striped Ledger; Data Com Transmit. | INSTRUCTIONS |
|----|-----------------------------------|------------------|------------------------|----------|-----|------|---|--|-----------|-----------|-------------------------|----------|------------------------|------|--|--------------|
| | | | Fixed Size | Variable | | | | | | | Fixed Size | Variable | | | | |
| | | | | Min | Max | Norm | | | | | | Min | Max | Norm | | |
| 18 | SELL PRICE | N | 0 | 7 | 3 | KB | ENABLE "C", "M" KEYS DISPLAY EITHER CENTS OR CENTS AND MILLS PROVIDE TEST TO SELECT MASK AND SCALE ARITHMETIC ACCORDINGLY | ✓ | | 2 | 7 | 3 | | | | |
| 19 | UNIT | | | | | | | | | | | | | | | |
| | A) IF "C", "M", "EA" PRICING | | | | | M | AUTOMATIC PRINT | ✓ | | 0 | 2 | 1 | "C", "M" OR "EA" PRINT | | | |
| | B) OTHERWISE | A | 0 | 4 | 2 | KB | | ✓ | | 0 | 4 | 2 | | | | |
| 20 | GROSS | | | | | M | GROSS = PRICE * QUANTITY | ✓ | | 2 | 8 | 5 | RED AND "-" IF MINUS | | | |
| 21 | DISCOUNT (IF SELECTED AT *16) | N | 1 | 4 | 1 | KB | SCALE AT 99.99 | ✓ | | 0 | 4 | 1 | | | | |
| 22 | NET AMOUNT | | | | | M | NET = GROSS - [DISCOUNT * GROSS] | ✓ | | | | | RED AND "-" IF MINUS | | | |
| 23 | COST PRICE (IF SELECTED AT *5) | N | 0 | 7 | 3 | KB | CENTS OR CENTS AND MILLS | ✓ | | 2 | 7 | 3 | | | | |
| 24 | COST AMOUNT (IF APPLICABLE) | | | | | M | COST = COST PRICE * QUANTITY | ✓ | | 2 | 8 | 5 | | | | |
| 25 | INVOICE SUBTOTAL | | | | | | | | | | | | | | | |
| | A) SELL | | | | | M | | ✓ | | 2 | 8 | 5 | RED AND "-" IF MINUS | | | |
| | B) COST (IF APPLIES) | | | | | M | | ✓ | | 2 | 8 | 5 | RED AND "-" IF MINUS | | | |
| 26 | TAX % (OPTIONAL) | | | | | | | | | | | | | | | |
| | A) AUTOMATIC | | | | | | | | | | | | | | | |
| | B) INDEXED | N | 0 | 4 | 1 | KB | | ✓ | | 0 | 4 | 2 | | | | |
| 27 | TAX AMOUNT (IF APPLICABLE) | | | | | M | TAX AMOUNT = SUBTOTAL * TAX % | ✓ | | 2 | 8 | 5 | | | | |
| 28 | ADDONS (OPTIONAL) | | | | | | INDEFINITE NUMBER | | | | | | | | | |
| | A) DESCRIPTION | A | 0 | 18 | 6 | KB | | ✓ | | 0 | 18 | 6 | | | | |
| | B) AMOUNT | N | 0 | 18 | 6 | KB | | ✓ | | 0 | 18 | 6 | RED AND "-" IF MINUS | | | |
| 29 | COST ADDON AMOUNT | N | 0 | 5 | 5 | KB | | ✓ | | 0 | 5 | 5 | | | | |

CUSTOMER _____ APPLICATION _____
EQUIPMENT _____
SALESMAN _____ BRANCH _____ DATE _____



PROGRAM DEFINITION CHART

| SUMCUCZUM | INPUT | | | | | PROCESSING | | | | | OUTPUT | | | | | INSTRUCTIONS | |
|-----------|-------------------------------|------------------|-------------|------------------------|--------------|------------|------|---|--|-----------|-----------|-------------------------|--------------|-----|------|--------------|--|
| | DATA DESCRIPTION OR OPERATION | ALPHANUMERIC (A) | NUMERIC (N) | FIELD DEFINITION INPUT | | | | INPUT SOURCE Keyboard (KB), Specify Media, Specify MB, OCK, PK, etc., only if particular one required; Punched Card (PC); Edge Punched Card (EPC); Punched Tape (PT); Striped Ledger (SL); Specify Reader; Memory (M); Data Com Receive. | PROCESSING REQUIREMENTS Accumulations; Formulas; Extensions; Store for later use; Calculations; Formatting for output. | PRINT (✓) | PUNCH (✓) | FIELD DEFINITION OUTPUT | | | | | OUTPUT DETAIL Print: Specify Printer and Format Console (CP), Line Printer (LP); Punch: Specify Adjunct, Format, Special codes, etc.; Striped Ledger; Data Com Transmit. |
| | | | | Fixed Size | Variable Min | Max | Norm | | | | | Fixed Size | Variable Min | Max | Norm | | |
| 30 | INVOICE TOTAL | | | | | | | | | | | | | | | | |
| | A) SELL | | | | | | | M | | | | | | | | | |
| | B) COST (IF APPLICABLE) | | | | | | | M | | | | | | | | | |
| 31 | AUTOMATIC CONTINUATION PAGES | | | | | | | | | | | | | | | | |
| | A) SOLD TO NAME | | | | | | | 0 3 1 | M | | | | | | | | |
| | B) PAGE NUMBER | | | | | | | | M | | | | | | | | |
| | C) INVOICE NUMBER | | | | | | | | M | | | | | | | | |
| | i) STORED | | | | | | | | M | | | | | | | | |
| | ii) NOT STORED | A | | | | | | 0 7 5 | KB | | | | | | | | |
| 32 | PROFIT | | | | | | | | M | | | | | | | | |
| | | | | | | | | | | | | | | | | | |
| 33 | DAILY TOTALS | | | | | | | | M | | | | | | | | |
| | A) HEADING | | | | | | | | | | | | | | | | |
| | B) TOTAL GROSS | | | | | | | | | | | | | | | | |
| | C) TOTAL NET SALES | | | | | | | | | | | | | | | | |
| | D) TOTAL PLUS ADDONS | | | | | | | | | | | | | | | | |
| | E) TOTAL MINUS ADDONS | | | | | | | | | | | | | | | | |
| | F) TOTAL PLUS COST ADDONS | | | | | | | | | | | | | | | | |
| | G) TOTAL MINUS COST ADDONS | | | | | | | | | | | | | | | | |
| | H) TOTAL TAXES | | | | | | | | | | | | | | | | |
| | I) TOTAL DISCOUNTS | | | | | | | | | | | | | | | | |
| | J) ACC. REC. NET | | | | | | | | | | | | | | | | |
| | K) TOTAL NET COSTS | | | | | | | | | | | | | | | | |

CUSTOMER _____ APPLICATION _____
 EQUIPMENT _____
 SALESMAN _____ BRANCH _____ DATE _____



PROGRAM DEFINITION CHART

BURROUGHS ASSEMBLER CODING FORM

PAGE 1 OF 26

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| PROGRAM ID | | | | | | | | | |
|------------|---|---|---|---|----|--|--|--|--|
| 5 | 6 | 7 | 8 | 9 | 10 | | | | |
| SAMPLE | | | | | | | | | |

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LENGTH | PARAMETER | | | REMARKS |
|------|----------|----------|----------|--------------|-----------|----------------|---|---------------------------|
| | | | | | A | B | C | |
| | | | | | LABEL | + OR - INC/REL | | |
| 1 | 011010 | | NOTE | | | | | BASIC BILLING PROGRAM |
| | 011020 | | LPKR | | PKTABL | | | LOAD PK BASE REGISTER |
| | 011030 | | LPNR | | MASKTB | | | LOAD PRINT NUM. BASE REG. |
| | 011040 | | LLLR | | 51 | | | LOAD LEFT LIMIT REGISTER |
| | 011050 | INITAL | PKA | | 41258 | | | ENABLE PKA1 LOAD DATE |
| | 011060 | | NOTE | | | | | INV. NO. TAX, PKA2 PRINT |
| | 011070 | | NOTE | | | | | DAILY TOTALS, PKA5 COST |
| | 011080 | | NOTE | | | | | INVOICE, PKA8 CLEAR TOTAL |
| | 011090 | | NK | | 0 | | 0 | XANY OK TO START INVOICE |
| | 011100 | | SET | | Y | | 3 | SET STANDARD INVOICE FLAG |
| | 011110 | BEGINVOC | | | SILDITOL | | 7 | ADVANCE TO LINE 10 |
| | 011120 | | RST | | Y | | 1 | SET Y1 CONT. PAGE |
| | 011130 | | CLA | | 0 | | 2 | CLEAR ACC. INSERT2 |
| | 011140 | | TRM | | PAGENO | | | STORE FOR PAGE NO. 2 |
| | 011150 | | PKA | | 17 | | | BRANCH TO INITIALIZE |
| | 011160 | | LKBR | | CUSTNM | | | LOAD KEYBOARD BASE REG. |
| | 011170 | | POS | | NMADIP | | | POSITION TO SOLD TO |
| | 011180 | | TKM | | 31 | | | XTYPE CUSTOMER NAME |
| | 011190 | | AL | | 1 | | | ADVANCE LEFT ONE LINE |
| | 012010 | | LLLR | | SILDITOL | | | LOAD WITH SOLD TO LINE |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

PAGE 3 OF 26

| PROGRAM ID | | | | |
|------------|---|---|---|----|
| 5 | 6 | 7 | 8 | 10 |
| S | A | M | P | L |

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LENGTH | PARAMETER | | | REMARKS |
|------|----------|---------|----------|--------------|-----------|----------------|-----|---------------------------|
| | | | | | A | B | C | |
| | | | | | LABEL | + OR - INC/REL | | |
| | 03,0,1,0 | | PØS | | NMAD-P | | | |
| | 03,0,2,0 | | PAI | | SAME | | | PRINT ALPHA MESSAGE |
| | 03,0,3,0 | RIBBON | ALTO | | RIBBL | | | ADVANCE TO RIBBON LINE |
| | 03,0,4,0 | | PØS | | TERM-P | | | POSITION TO TERMS |
| | 03,0,5,0 | | TK | | 1 | | | XTYPE TERMS |
| | 03,0,6,0 | | PØS | | ORDNØP | | | POSITION TO ORDER NO. |
| | 03,0,7,0 | | TK | | 10 | | | TYPE ORDER NO. |
| | 03,0,8,0 | | PØS | | CUSNØP | | | POSITION TO CUSTOMER NO. |
| | 03,0,9,0 | | TK | | 9 | | | XTYPE CUSTOMER NUMBER |
| | 03,1,0,0 | | PØS | | SLIDBYP | | | POSITION TO SOLD BY |
| | 03,1,1,0 | | TK | | 12 | | | XTYPE SALES NAMES |
| | 03,1,2,0 | | PØS | | SHIPV.IP | | | POSITION TO SHIP VIA |
| | 03,1,3,0 | | TK | | 9 | | | XTYPE SHIP VIA |
| | 03,1,4,0 | | CILM | | INVNET | | | CLEAR INVOICE NET TOTAL |
| | 03,1,5,0 | | CILM | | INVCST | | | CLEAR INVOICE COST TOTAL |
| | 03,1,6,0 | | SIRJ | | DATE-1 | | | PRINT DATE & INVOICE NO |
| | 03,1,7,0 | | BIRU | | | + 2 | | BRU SKIP LINE INCREMENT |
| | 03,1,8,0 | BØDY.IV | SIRJ | | CKLINE | | | INCREMENT & CHECK LINE CT |
| | 03,1,9,0 | | EX | | Y | | 1 2 | TEST FOR CONT. PAGE |
| | 03,2,0,0 | | SIRJ | | SUBTØT | | | PRINT SUB-TOTAL |

CONSTANT DATA (NUMERIC)

ALPHANUMERIC DATA OR PRINT MASK

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

| PROGRAM ID | | | | |
|------------|--------|---|---|---|
| 5 | 6 | 7 | 8 | 9 |
| 10 | SAMPLE | | | |

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LEN-GTH | PARAMETER | | | REMARKS |
|------|----------|-------|----------|---------------|-----------|----------------|------|---------------------------|
| | | | | | A | B | C | |
| | | | | | LABEL | + OR - INC/REL | | |
| I | 11 | 06010 | EX | | A | - | 1 | TEST IF MINUS FLAG SET |
| | 12 | 06020 | BRU | | MPRICE | + | 3 | BRANCH TO INDEX PRICE |
| | 13 | 06030 | SIRJ | | PT-PRC | | | PRINT SELL PRICE |
| | 14 | 06040 | POS | | PERU-P | | | POSITION TO PER UNIT COLN |
| | 15 | 06050 | EX | | A | | C 4 | TEST IF C FLAG SET |
| | 16 | 06060 | PC | | C | | | PRINT C |
| | 17 | 06070 | TIRA | | QTY | | | TRANSFER QUANTITY |
| | 18 | 06080 | SILRO | | O | | 2 | SHIFT FOR PER C PRICE |
| | 19 | 06090 | TRM | | QTY | | | STORE QUANTITY |
| | 20 | 06100 | EX | | A | | M 4 | TEST IF M KEY USED |
| | 21 | 06110 | PC | | M | | | PRINT M |
| | 22 | 06120 | TIRA | | QTY | | | TRANSFER QUANTITY |
| | 23 | 06130 | SILRO | | O | | 3 | SHIFT FOR PER M PRICE |
| | 24 | 06140 | TRM | | QTY | | | STORE QUANTITY |
| | 25 | 06150 | TIRA | | PRICE | | | TRANSFER SELL PRICE |
| | 26 | 06160 | EX | | A | | CM 1 | TEST IF EITHER CM USED |
| | 27 | 06170 | BRU | | GRSAMT | | | BRANCH TO GROSS AMOUNT |
| | 28 | 06180 | EX | | K | | 4 3 | TEST IF CK4 USED |
| | 29 | 06190 | PC | | E | | | PRINT E |
| | 30 | 06200 | PC | | A | | | PRINT A |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| PROGRAM ID | | | | |
|------------|---|---|---|---|
| 5 | 6 | 7 | 8 | 9 |
| SAMPLE | | | | |

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LENGTH | PARAMETER | | | REMARKS | |
|------|----------|----------|------------|--------------|-----------|----------------|-----|---------|---------------------------|
| | | | | | A | + OR - INC/REL | B C | | |
| 1 | 11 | 12,0,1,0 | EX | | A | | - | 1 | TEST IF MINUS |
| | 12 | 12,0,2,0 | ADM | | T,AD,N,CM | | | | ADD TO COST, TOTAL MINUS |
| | 13 | 12,0,3,0 | SK | | A | | - | 1 | TEST IF PLUS |
| | 14 | 12,0,4,0 | ADM | | T,AD,N,CP | | | | ADD TO COST, TOTAL PLUS |
| | 15 | 12,0,5,0 | SRJ | | CK,LINE | | | | INCREMENT, LINE COUNT |
| | 16 | 12,0,6,0 | BRU | | AD,ONS | | | | BRANCH TO NEXT ADDON |
| | 17 | 12,0,7,0 | TOTALI EX | | X | | 4 | 1 | TEST IF ADDONS ON INVOICE |
| | 18 | 12,0,8,0 | SRJ | | SUB,TOT | | | | PRINT TOTALS |
| | 19 | 12,0,9,0 | EX | | Y | | 3 | 1 | TEST IF STANDARD INVOICE |
| | 20 | 12,1,0,0 | BRU | | BEG,INV | | | | BRU TO NEXT INVOICE |
| | 21 | 12,1,1,0 | T,RA | | INV,NET | | | | TRANSFER TOTAL NET INV. |
| | 22 | 12,1,2,0 | SUA | | WRK,REG | | | | SUBT TAX FROM NET |
| | 23 | 12,1,3,0 | SUA | | INV,CST | | | | SUBT COST FROM NET |
| | 24 | 12,1,4,0 | POS | | PR,T%-IP | | | | POSITION TO PRINT PROFIT |
| | 25 | 12,1,5,0 | SRJ | | P,NUM,RC | | | | PRINT PROFIT AMOUNT |
| | 26 | 12,1,6,0 | BRU | | BEG,INV | | | | BRANCH TO NEXT INVOICE |
| | 27 | 12,1,7,0 | INV-NØ SRJ | | AR,PØS | | | | ADVANCE FORM POSITION |
| | 28 | 12,1,8,0 | PA | | FRST,IN | | | | PRINT ALFA FRSTIN |
| | 29 | 12,1,9,0 | AR | | 2 | | | | ADVANCE RIGHT TWO LINES |
| | 30 | 12,2,0,0 | POS | | 1,2,0 | | | | POSITION FOR PRINT |

CONSTANT DATA (NUMERIC)

ALPHANUMERIC DATA OR PRINT MASK

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

PAGE 13 OF 26

| PROGRAM ID | | | | | | | | | |
|------------|---|---|---|---|----|--|--|--|--|
| 5 | 6 | 7 | 8 | 9 | 10 | | | | |
| SAMPLE | | | | | | | | | |

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LEN - GTH | PARAMETER | | | REMARKS |
|------|-----------|--------|----------|-----------------|-----------|---|--------------------------|---------|
| | | | | | A | B | C | |
| | 1,3,0,1,0 | | NK | 6 | | 0 | XINDEX INVOICE NO. | |
| | 1,3,0,2,0 | | PN | 5 | | 0 | PRINT INVOICE NUMBER | |
| | 1,3,0,3,0 | | TIRM | INVNØ | | | STORE INVOICE NO | |
| | 1,3,0,4,0 | | SRJ | AR-POS | | | ADVANCE FORM POSITION | |
| | 1,3,0,5,0 | | PA | DAIT | | | PRINT ALFA DAIT | |
| | 1,3,0,6,0 | | AR | 2 | | | ADVANCE RIGHT TWO LINES | |
| | 1,3,0,7,0 | | POS | 1,2,0 | | | POSITION FOR PRINT | |
| | 1,3,0,8,0 | | LKBR | DATE | | | LOAD DATE BASE M.A. | |
| | 1,3,0,9,0 | | TIRM | 1,2 | | | XTYPE DATE | |
| | 1,3,1,0,0 | | SRJ | AR-POS | | | ADVANCE FORM POSITION | |
| | 1,3,1,1,0 | | PA | TXRT | | | PRINT ALFA TXRT | |
| | 1,3,1,2,0 | | POS | 1,2,0 | | | POSITION FOR PRINT | |
| | 1,3,1,3,0 | | AR | 2 | | | ADVANCE RIGHT TWO LINES | |
| | 1,3,1,4,0 | | NK | 2 | 2 | | XINDEX TAX RATE | |
| | 1,3,1,5,0 | | PN | 7 | 3 | | PRINT TAX RATE | |
| | 1,3,1,6,0 | | TIRM | TAXCISIT | | | STORE FIRST TAX RATE | |
| | 1,3,1,7,0 | | BRU | INITIAL | | | BRU TO INITIALIZE | |
| | 1,3,1,8,0 | DILYIT | POS | 1,2,0 | | | POS SUMMARY HEADING | |
| | 1,3,1,9,0 | | AR | 5 | | | ADVANCE RIGHT FIVE LINES | |
| | 1,3,2,0,0 | | PA | HEADING | | | PRINT ALFA MESSAGE | |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

PAGE 21 OF 26

| PROGRAM ID | | | | |
|------------|---|---|---|---|
| 5 | 6 | 7 | 8 | 9 |
| S | A | M | P | L |

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LEN-GTH | PARAMETER | | | REMARKS | |
|------|----------|--------|----------|---------------|-----------|----------------|-----|---------|---------------------------|
| | | | | | A | B | C | | |
| | | | | | LABEL | + OR - INC/REL | | | |
| I | 21010 | | SK | | T | | I | I | TEST FOR 17TH LINE |
| | 21020 | | SRR | | I | | | | SUBROUTINE RETURN |
| | 21030 | | NOTE | | | | | | SET Y FOR LAST 6 LINES IN |
| | 21040 | | SET | | Y | | 4 | | BODY OF INVOICE REACHED |
| | 21050 | | IIR | | 4 | | 255 | | INCREMENT LINE COUNT |
| | 21060 | | ALARM | | | | | | WARN OPERATOR BOTTOM INV. |
| | 21070 | | EX | | T | | I | I | TEST FOR LAST LINE |
| | 21080 | | SET | | Y | | I | | SET YI FOR CONT. PAGE |
| | 21090 | | SRR | | I | | | | SUBROUTINE RETURN |
| | 21100 | CTMILL | TRM | | WRKREG | | | | TRANSFER PRICE TO M.A. |
| | 21110 | | SILR0 | | 13 | | 0 | | SHIFT OFF DOLLARS & CENTS |
| | 21120 | | EXZ | | 3 | | | | TEST IF ZERO MILLS |
| | 21130 | | TRA | | WRKREG | | | | TRANSFER PRICE |
| | 21140 | | SILR0 | | 0 | | 2 | | REPOSITION FOR CENTS ONLY |
| | 21150 | | SRR | | I | | | | SUBROUTINE RETURN |
| | 21160 | | TRA | | WRKREG | | | | TRANSFER PRICE |
| | 21170 | | SILR0 | | 0 | | 7 | | SHIFT OFF 7 DIGITS |
| | 21180 | | NOTE | | | | | | TEST IF MORE THAN SEVEN |
| | 21190 | | SIKZ | | 3 | | | | DIGITS WERE INDEXED |
| | 21200 | | SET | | A | | - | | YES, SET - FLAG |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

| PROGRAM ID | | | | | | | | | |
|------------|---|---|---|---|----|--|--|--|--|
| 5 | 6 | 7 | 8 | 9 | 10 | | | | |
| SAMPLE | | | | | | | | | |

CUSTOMER _____

BRANCH _____

PROGRAMMER _____

| CODE | SEQUENCE | LABEL | OP. CODE | FIELD LEN-GTH | PARAMETER | | | REMARKS |
|------|----------|-------|----------|---------------|-----------|---------------------------|---|---------------------------|
| | | | | | A | B | C | |
| | | | | | LABEL | + OR - INC/REL | | |
| | 11 | 22010 | ALARM | | | | | WARN OPERATOR ERROR PRICE |
| | 12 | 22020 | SRR | | | | | SUBROUTINE RETURN |
| | 13 | 22030 | TRA | | WRKREIG | | | TRANSFER PRICE |
| | 14 | 22040 | NOTE | | | | | SET FLAG TO INDICATE |
| | 15 | 22050 | SET | | A | S | | CENTS AND MILLS |
| | 16 | 22060 | SRR | | | | | SUBROUTINE RETURN |
| | 17 | 22070 | IVCOST | RST | Y | 3 | | RESET STANDARD INV. FLAG |
| | 18 | 22080 | BRU | | BEGINV | | | BRU TO INVOICE ROUTINE |
| | 19 | 22090 | WORD | | | | | PROGRAM KEY TABLE |
| | 20 | 22100 | PKTABL | BRU | INVTOT | | | BRU TO PRINT DAILY TOTALS |
| | 21 | 22110 | BRU | | DLYTOT | | | BRU TO SUBTOTAL |
| | 22 | 22120 | BRU | | INVTOT | | | BRU TO TAX PERCENT |
| | 23 | 22130 | TRA | | TAXCST | | | BRU TO COST INVOICE |
| | 24 | 22140 | BRU | | IVCOST | | | BRU TO TOTAL INVOICE |
| | 25 | 22150 | BRU | | TOTALI | | | BRU TO INITIALIZE |
| | 26 | 22160 | BRU | | INITAL | | | BRU TO CLEAR MEMORY |
| | 27 | 22170 | BRU | | CLRMEM | | | |
| | 28 | 22180 | PAGE | | | | | |
| | 29 | 22190 | MASKTB | MASK | 15 | ZZZ, ZZZ, ZZZ, ZZZ | | NUMERIC |
| | 30 | 22200 | MASKI | MASK | 16 | ZI, ZIZI, ZIZI, ZIZI. D D | | MONETARY ZERO PRINT |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

BURROUGHS ASSEMBLER CODING FORM

| PROGRAM ID | | | | | |
|------------|---|---|---|---|----|
| 5 | 6 | 7 | 8 | 9 | 10 |
| S | A | M | P | L | E |

CUSTOMER _____
 BRANCH _____
 PROGRAMMER _____

| CODE | SEQUENCE | | | | | LABEL | | | | | OP. CODE | | | | | FIELD LEN-GTH | PARAMETER | | | | | | | | | | | | REMARKS | | | |
|------|----------|----|----|----|----|-------|----|----|----|----|----------|----|----|----|----|---------------|-----------|-------|----------------|---|--|--|--|--|--|---|--|--|---------|---|--|---------------------------|
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | | 26 | 27 | 28 | A | | | | | | B | | | | C | | |
| | | | | | | | | | | | | | | | | | | LABEL | + OR - INC/REL | | | | | | | | | | | | | |
| 1 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 98 | | | | | | | | | | | | | POS COST AMOUNT |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 59 | | | | | | | | | | | | | POS ALPHA ADDONS |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 87 | | | | | | | | | | | | | POS PROFIT/LOSS % |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 64 | | | | | | | | | | | | | POS TO CONT. PAGE |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 120 | | | | | | | | | | | | | POS MISC ENTRIES RIGHT |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 136 | | | | | | | | | | | | | POS SUMMARY COST |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | X3 NO DISCOUNT LINE |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | X4 ADDONS ON INVOICE |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | Y1 START CONT. PAGE |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | Y3 STANDARD INVOICE |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | Y4 LAST 6 INVOICE LINES |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | X1 INDICATES MANUAL INDEX |
| | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | | | | | | | | | | | | | END OF JOB |

| | | | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| CONSTANT DATA (NUMERIC) | | | | | | | | | | | | | | | | | | | | | | | |
| ALPHANUMERIC DATA OR PRINT MASK | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

PROGRAM ID. -

DATE RUN 3/26/70 TIME = 12:48

VERSION 02-01-70

PAGE 001

| WORD SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|----------|-------------|----------|-----------|---------|---------|-----------------------|-------|-------|---------------|---------------------------|
| | | 1 0 | | | | NOTE | | | | BASIC BILLING PROGRAM |
| 0 0 | FC69 | 2 0 | | LPKR | | PKTABL | | | 105 | LOAD PK BASE REGISTER |
| 1 | FA6B | 3 0 | | LPNR | | MASKTB | | | 107 | LOAD PRINT NUM. BASE REG. |
| 2 | E433 | 4 0 | | LLLR | | 51 | | | | LOAD LEFT LIMIT REGISTER |
| 3 | F693 | 5 0 | INITAL | PKA | 4 | 1258 | | | | ENABLE PKA 1 LOAD DATE |
| | | 6 0 | | | | NOTE | | | | INV. NO. TAX, PKA2 PRINT |
| | | 7 0 | | | | NOTE | | | | DAILY TOTALS, PKA5 COST |
| | | 8 0 | | | | NOTE | | | | INVOICE, PKA8 CLEAR TOTL |
| 1 0 | A600 | 9 0 | | NK | | 0 | 0 | | | XANY CKK TO START INVOICE |
| 1 | 6758 | 10 0 | | SET | | Y | 3 | | | SET STANDARD INVOICE FLAG |
| 2 | ER0F | 11 0 | REGINV | OC | | SLDTOL + 7 | | | 15 | ADVANCE TO LINE 10 |
| 3 | 6552 | 12 0 | | RST | | Y | 1 | | | SFT Y 1 CONT. PAGE |
| 2 0 | 8F02 | 13 0 | | CLA | | 0 | 2 | | | CLEAR ACC. INSERT 2 |
| 1 | 3072 | 14 0 | | TRM | | PAGENO | | | 114 | STORE FOR PAGE NO. 2 |
| 2 | F640 | 15 0 | | PKA | 1 | 7 | | | | BRANCH TO INITIALIZE |
| 3 | F075 | 16 0 | | LKBR | | CUSTNM | | | 117 | LOAD KEYBOARD BASE REG. |
| 3 0 | ER0C | 17 0 | | POS | | NMAD-P | | | 13 | POSITION TO SOLD TO |
| 1 | AD1F | 18 0 | | TKM | | 31 | | | | XTYPE CUSTOMER NAME |
| 2 | ED01 | 19 0 | | AL | | 1 | | | | ADVANCE LEFT ONE LINE |
| 3 | E008 | 20 0 | | LLCR | | SLDTOL | | | 8 | LOAD WITH SOLD TO LINE |
| 4 0 | 5000 | 21 0 | | LTR | 4 | | 0 | | | BEGIN TYPING LOOP |
| 1 | 5300 | 22 0 | | LTR | 3 | | 0 | | | LOAD INDEX REGISTER 3 |

MEMORY SIZE NOT ENTERED (512 ASSUMED)

4-34

| PROGRAM ID. - | | DATE RUN | 3/26/70 | TIME - 12:48 | VERSION 02-01-70 | PAGE 002 | | | | | | |
|---------------|-----|-------------|----------|--------------|------------------|----------|-------------------|-----|-------|-------|---------------|--------------------------|
| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
| | 2 | 6544 | 23 0 | | RST | | X | | 2 | | | RESET X FLAG |
| | 3 | EROC | 24 0 | SHIPTO | POS | | NMAD-P | | | | 13 | POSITION TO PRINT |
| 5 | 0 | AC1F | 25 0 | | TK | | 31 | | | | | XTYPE INFO |
| | 1 | ED01 | 26 0 | | AL | | 1 | | | | | ADVANCE LEFT ONE LINE |
| | 2 | 5802 | 27 0 | | IIR | | 4 | | 2 | | | TEST IF THIRD TIME |
| | 3 | 4444 | 28 0 | | EX | | X | | 2 | 4 | | ENTER IF SHIPTO |
| 6 | 0 | 5803 | 29 0 | | IIR | | 3 | | 3 | | | TEST FOR COMPLETION |
| | 1 | 4184 | 30 0 | | SK | | T | | T | 1 | | TERMINATE LOOP |
| | 2 | 7C04 | 31 0 | | BRU | | SHIPTO | | | | 4 3 | REPEAT |
| | 3 | 7809 | 32 0 | | BRU | | RIBBON | | | | 9 2 | JUMP TO RIBBON AREA |
| 7 | 0 | 4184 | 33 0 | | SK | | T | | T | 1 | | TERMINATE LOOP |
| | 1 | 7C04 | 34 0 | | BRU | | SHIPTO | | | | 4 3 | REPEAT LOOP |
| | 2 | 4599 | 35 0 | | EX | | K | | 34 | 1 | | BRANCH TO PRINT SAME |
| | 3 | 7C08 | 36 0 | | BRU | | | + 4 | | | 8 3 | EXIT LOOP |
| 8 | 0 | 6744 | 37 0 | | SET | | X | | 2 | | | SET TO ENTER SHIPTO LOOP |
| | 1 | ED02 | 38 0 | | AL | | 2 | | | | | ADVANCE TWO LINES |
| | 2 | 7C04 | 39 0 | | BRU | | SHIPTO | | | | 4 3 | RETURN TO LOOP |
| | 3 | ED02 | 40 0 | | AL | | 2 | | | | | ADVANCE TWO LINES |
| 9 | 0 | EROC | 41 0 | | POS | | NMAD-P | | | | 13 | |
| | 1 | C888 | 42 0 | | PA | | SAME | | | | 136 | PRINT ALPHA MESSAGE |
| | 2 | E914 | 43 0 | RIBBON | ALTO | | RTRBL | | | | 20 | ADVANCE TO RIBBON LINE |
| | 3 | ER05 | 44 0 | | POS | | TERM-P | | | | 6 | POSITION TO TERMS |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|------------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| 10 | 0 | AC08 | 45 | 0 | TK | | 11 | | | | | XTYPE TERMS |
| | 1 | EB12 | 46 | 0 | POS | | ORDNOP | | | | 19 | POSITION TO ORDER NO. |
| | 2 | AC0A | 47 | 0 | TK | | 10 | | | | | TYPE ORDER NO. |
| | 3 | EB1E | 48 | 0 | POS | | CUSNOP | | | | 31 | POSITION TO CUSTOMER NO. |
| 11 | 0 | AC09 | 49 | 0 | TK | | 9 | | | | | XTYPE CUSTOMER NUMBER |
| | 1 | EB29 | 50 | 0 | POS | | SLDBYP | | | | 42 | POSITION TO SOLD BY |
| | 2 | AC0C | 51 | 0 | TK | | 12 | | | | | XTYPE SALES NAMES |
| | 3 | EB37 | 52 | 0 | POS | | SHPVIP | | | | 56 | POSITION TO SHIP VIA |
| 12 | 0 | AC09 | 53 | 0 | TK | | 9 | | | | | XTYPE SHIPVIA |
| | 1 | D87C | 54 | 0 | CLM | | INVNET | | | | 124 | CLEAR INVOICE NET TOTAL |
| | 2 | D87D | 55 | 0 | CLM | | INVCST | | | | 125 | CLEAR INVOICE COST TOTAL |
| | 3 | 285E | 56 | 0 | SRJ | | DATE-J | | | | 94 2 | PRINT DATE & INVOICE NO |
| 13 | 0 | 780D | 57 | 0 | BRU | | | + | 2 | | 13 2 | BRU SKIP LINE INCREMENT |
| | 1 | 2062 | 58 | 0 | BODYTV SRJ | | CKLINE | | | | 98 0 | INCREMENT & CHECK LINE CT |
| | 2 | 4652 | 59 | 0 | EX | | Y | | 1 | 2 | | TEST FOR CONT. PAGE |
| | 3 | 2856 | 60 | 0 | SRJ | | SURTOT | | | | 86 2 | PRINT SUB-TOTAL |
| 14 | 0 | 2850 | 61 | 0 | SRJ | | CONTPG | | | | 80 2 | BEGIN CONTINUATION PAGE |
| | 1 | 6549 | 62 | 0 | RST | | X | | 34 | | | RESET DISCOUNT FLAG |
| | | | 63 | 0 | NOTE | | | | | | | ADDON FLAG |
| | 2 | F604 | 64 | 0 | PKA | 1 | 3 | | | | | ENABLE SUB-TOTAL PK |
| | 3 | EB04 | 65 | 0 | POS | | PRDCDP | | | | 5 | POSITION TO PRODUCT CODE |
| 15 | 0 | A640 | 66 | 0 | MAX NK | | 4 | | 0 | | | INDEX NO. LESS THAN 5000 |

| PROGRAM ID. - | | DATE RUN | 3/26/70 | TIME - 12:48 | VERSION 02-01-70 | PAGE 004 | | | | | |
|---------------|-----|-------------|----------|--------------|------------------|---------------|-----------------|-------|-------|---------------|---------------------------|
| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. LABEL | A-PARAMETER INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
| | 1 | D487 | 67 | 0 | CPA | TEST | | | | 135 | CHECK IF VALID |
| | 2 | 0800 | 68 | 0 | NOP | | | | | | EQUAL TO, NO OPERATION |
| | 3 | 7410 | 69 | 0 | BRU | | + 2 | | | 16 1 | INVALID CODE |
| 16 | 0 | 7C10 | 70 | 0 | BRU | | + 3 | | | 16 3 | VALID PRINT |
| | 1 | 0980 | 71 | 0 | ALARM | | | | | | INVALID DO NOT PRINT |
| | 2 | 700F | 72 | 0 | BRU | MAX | | | | 15 0 | TO INDEX NEXT CODE |
| | 3 | 4752 | 73 | 0 | EX | Y | | 1 | 3 | | TEST FOR A CONT. PAGE |
| 17 | 0 | 2856 | 74 | 0 | SRJ | SUBTOT | | | | 86 2 | PRINT SUBTOTAL |
| | 1 | 2850 | 75 | 0 | SRJ | CONTPG | | | | 80 2 | BEGIN CONTINUATION PAGE |
| | 2 | 7C0E | 76 | 0 | BRU | BODYIV | + 6 | | | 14 3 | BRANCH TO INDEX PROD-NO. |
| | 3 | E001 | 77 | 0 | AL | 1 | | | | | ADVANCE LEFT ONE LINE |
| 18 | 0 | 0280 | 78 | 0 | SLRD | 11 | | 0 | | | SHIFT FOR PRINTING |
| | 1 | D4E3 | 79 | 0 | PN | 14 | | 3 | | | PRINT PRODUCT NO. |
| | 2 | A440 | 80 | 0 | NKR | 4 | | 0 | | | INDEX QUANTITY |
| | 3 | 4592 | 81 | 0 | EX | K | | 1 | 1 | | TST FOR DISC OCK1 NO DISC |
| 19 | 0 | 6748 | 82 | 0 | SET | X | | 3 | | | SET X3 NO DISCOUNT FLAG |
| | 1 | ER06 | 83 | 0 | PDS | QTY-P | | | | 7 | POSITION TO QUANTITY |
| | 2 | D190 | 84 | 0 | PNS- | 9 | | 0 | | | PRINT WHOLE NO. QUANTITY |
| | 3 | C52D | 85 | 0 | PC- | - | | | | | PRINT - IF MINUS |
| 20 | 0 | 0270 | 86 | 0 | SLRD | 7 | | 0 | | | REPOSITION FOR EXTN |
| | 1 | 307A | 87 | 0 | TRM | QTY | | | | 122 | STORE QUANTITY |
| | 2 | EB15 | 88 | 0 | TKDESC | POS | DESC-P | | | 22 | POSITION TO DESCRIPTION |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | R PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| | 3 | AC17 | 89 | 0 | TK | | 23 | | | | | XTYPE DESCRIPTION |
| 21 | 0 | 4092 | 90 | 0 | SK | | K | | 1 | 4 | | TEST IF OCK1 USED |
| | 1 | 2062 | 91 | 0 | SRJ | | CKLINE | | | | 98 0 | INCREMENT LINE COUNT |
| | 2 | 4252 | 92 | 0 | SK | | Y | | 1 | 2 | | TEST IF LAST INVOICE LINE |
| | 3 | ED01 | 93 | 0 | AL | | 1 | | | | | ADVANCE LEFT ONE LINE |
| 22 | 0 | 7814 | 94 | 0 | BRU | | TKDESC | | | | 20 2 | BRANCH TO TYPE DESC. |
| | 1 | EB2A | 95 | 0 | MPRICE | POS | SPRC=P | | | | 43 | POSITION TO SELL PRICE |
| | 2 | A252 | 96 | 0 | NKCM | | 5 | | 2 | | | XINDEX PRICE |
| | 3 | 4ECC | 97 | 0 | EXE | | A | | CM | 2 | | TEST IF BOTH C M USED |
| 23 | 0 | 0980 | 98 | 0 | ALARM | | | | | | | WARN PRICE ERROR |
| | 1 | 7416 | 99 | 0 | BRU | | MPRICE | | | | 22 1 | BRANCH TO INDEX PRICE |
| | 2 | 2864 | 100 | 0 | SRJ | | CTMILL | | | | 100 2 | DETERMINE IF MILLS INDEX |
| | 3 | 45C1 | 101 | 0 | EX | | A | | - | 1 | | TEST IF MINUS FLAG SET |
| 24 | 0 | 7017 | 102 | 0 | BRU | | MPRICE + 3 | | | | 23 0 | BRANCH TO INDEX PRICE |
| | 1 | 284E | 103 | 0 | SRJ | | PT=PRC | | | | 78 2 | PRINT SELL PRICE |
| | 2 | ER36 | 104 | 0 | POS | | PERU=P | | | | 55 | POSITION TO PER UNIT COLN |
| | 3 | 44C4 | 105 | 0 | EX | | A | | C | 4 | | TEST IF C FLAG SET |
| 25 | 0 | C043 | 106 | 0 | PC | | C | | | | | PRINT C |
| | 1 | 387A | 107 | 0 | TRA | | QTY | | | | 122 | TRANSFER QUANTITY |
| | 2 | 0202 | 108 | 0 | SLRO | | 0 | | 2 | | | SHIFT FOR PER C PRICE |
| | 3 | 307A | 109 | 0 | TRM | | QTY | | | | 122 | STORE QUANTITY |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|----------------------------|
| 26 | 0 | 44C8 | 110 | 0 | EX | | A | | M | 4 | | TEST IF M KEY USED |
| | 1 | C04D | 111 | 0 | PC | | M | | | | | PRINT M |
| | 2 | 387A | 112 | 0 | TRA | | QTY | | | | 122 | TRANSFER QUANTITY |
| | 3 | 0203 | 113 | 0 | SLRD | | 0 | | 3 | | | SHIFT FOR PER M PRICE |
| 27 | 0 | 307A | 114 | 0 | TRM | | QTY | | | | 122 | STORE QUANTITY |
| | 1 | 387B | 115 | 0 | TRA | | PRICE | | | | 123 | TRANSFER SELL PRICE |
| | 2 | 45CC | 116 | 0 | EX | | A | | CM | 1 | | TEST IF EITHER C M USED |
| | 3 | 741D | 117 | 0 | BRU | | GRSAMT | | | | 29 1 | BRANCH TO GROSS AMOUNT |
| 28 | 0 | 4791 | 118 | 0 | EX | | K | | 4 | 3 | | TEST IF DCK4 USED |
| | 1 | C045 | 119 | 0 | PC | | E | | | | | PRINT E |
| | 2 | C041 | 120 | 0 | PC | | A | | | | | PRINT A |
| | 3 | 741D | 121 | 0 | BRU | | GRSAMT | | | | 29 1 | BRANCH TO GROSS AMOUNT |
| 29 | 0 | AC04 | 122 | 0 | TK | | 4 | | | | | XTYPE PER UNIT CHARACTER |
| | 1 | 6429 | 123 | 0 | GRSAMT | LSR | 9 | | | | | LOAD SHIFT REGISTER |
| | 2 | 8C7A | 124 | 0 | MULR | | QTY | | | | 122 | MULT PRICE X QUANTITY |
| | 3 | EB37 | 125 | 0 | POS | | GRS-P | | | | 56 | POSITION TO GROSS AMOUNT |
| 30 | 0 | 2C4D | 126 | 0 | SRJ | | PNUMRC | | | | 77 3 | PRINT GROSS AMOUNT |
| | 1 | 306F | 127 | 0 | TRM | | WRKREG | | | | 111 | STORE GROSS AMOUNT |
| | 2 | 8085 | 128 | 0 | ADM | | TGROSS | | | | 133 | ADD TO GRAND GROSS TOTAL |
| | 3 | 4548 | 129 | 0 | EX | | X | | 3 | 1 | | TEST IF DISCOUNT APPLICABL |
| 31 | 0 | 7C21 | 130 | 0 | BRU | | NETAMT | | | | 33 3 | BRANCH TO NET AMOUNT |
| | 1 | EB43 | 131 | 0 | POS | | DISC-P | | | | 68 | POSITION TO DISCOUNT |

| WORD | OBJECT | SEQ. | SYM. | OP | FD. | A-PARAMETER | B | C | LABEL | R F M A R K S |
|------|--------|------|------|--------|--------|-------------|-----|-----|---------|---------------------------|
| SYL | CODE | NO. | LOC. | CODE | LN. | LABEL INC | PAR | PAR | DEC EQU | |
| 2 | A622 | 132 | 0 | NK | 2 | | 2 | | | XINDEX DISCOUNT |
| 3 | 45D0 | 133 | 0 | EXZ | 1 | | | | | TEST IF ZERO INDEXED |
| 32 | 0 | 7C21 | 134 | 0 | BRU | NETAMT | | | 33 3 | BRANCH TO NET AMOUNT |
| 1 | D473 | 135 | 0 | PN | 7 | | 3 | | | PRINT DISCOUNT AS :A: % |
| 2 | C025 | 136 | 0 | PC | % | | | | | PRINT "%" |
| 3 | 6424 | 137 | 0 | LSR | 4 | | | | | LOAD SHIFT REGISTER |
| 33 | 0 | 8C6F | 138 | 0 | MULR | WRKREG | | | 111 | MULT DISCOUNT X GROSS |
| 1 | 906F | 139 | 0 | SUM | WRKREG | | | | 111 | SUBT DISCOUNT FROM GROSS |
| 2 | 8086 | 140 | 0 | ADM | TDISCT | | | | 134 | ADD TO TOTAL DISCOUNTS |
| 3 | 386F | 141 | 0 | NETAMT | TRA | WRKREG | | | 111 | TRANSFER NET AMOUNT |
| 34 | 0 | ER48 | 142 | 0 | POS | NET-P | | | 76 | POSITION TO NET AMOUNT |
| 1 | 2C4D | 143 | 0 | SRJ | PNUMRC | | | | 77 3 | PRINT NET AMOUNT |
| 2 | 807C | 144 | 0 | ADM | INVNET | | | | 124 | ADD LINE NET TO TOTAL NET |
| 3 | 807E | 145 | 0 | ADM | TNET | | | | 126 | ADD TO TOTAL NET SALES |
| 35 | 0 | 4558 | 146 | 0 | EX | Y | 3 | 1 | | TEST IF COST APPLICABLE |
| 1 | 740D | 147 | 0 | BRU | BODYIV | | | | 13 1 | BRANCH TO NEXT LINE |
| 2 | 6429 | 148 | 0 | LSR | 9 | | | | | LOAD SHIFT REGISTER |
| 3 | ER59 | 149 | 0 | POS | CPRC-P | | | | 90 | POSITION TO COST PRICE |
| 36 | 0 | A632 | 150 | 0 | COSTIV | NK | 2 | | | INDEX COST PRICE |
| 1 | 2864 | 151 | 0 | SRJ | CTMILL | | | | 100 2 | DETERMINE IF MILLS INDEX |
| 2 | 45C1 | 152 | 0 | FX | A | | - | 1 | | TEST IF COST PRICE ERROR |
| 3 | 7024 | 153 | 0 | BRU | COSTIV | | | | 36 0 | BRANCH TO INDEX COST |

| WORD SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|-------------|----------------|-------------|--------------|------------|------------|--------------------------|----------|----------|------------------|---------------------------|
| 37 0 | 2C4E | 154 0 | | SRJ | | PT=PRC + 1 | | | 78 3 | PRINT COST PRICE |
| | | 155 0 | | NOTE | | | | | | SHIFT FOR MILLS |
| 1 | 8C7A | 156 0 | | MULR | | QTY | | | 122 | MULT COST X QUANTITY |
| 2 | EB61 | 157 0 | | POS | | CSTX=P | | | 98 | POSITION TO COST AMOUNT |
| 3 | 2C4D | 158 0 | | SRJ | | PNUMRC | | | 77 3 | PRINT COST AMOUNT |
| 38 0 | 8084 | 159 0 | | ADM | | TCOSTS | | | 132 | ADD LINE COST TO TOT COST |
| 1 | 807D | 160 0 | | ADM | | INVCST | | | 125 | ADD TO INVOICE COST TOTAL |
| 2 | 740D | 161 0 | | BRU | | BODYIV | | | 13 1 | BRANCH TO NEXT LINE |
| 3 | F628 | 162 0 | INVTOT | PKA | 2 | 46 | | | | ENABLE TAX & TOTAL PK |
| 39 0 | D86F | 163 0 | | CLM | | WRKREG | | | 111 | CLEAR WORKING MEMORY |
| 1 | 6424 | 164 0 | | LSR | | 4 | | | | LOAD SHIFT REGISTER |
| 2 | 2856 | 165 0 | | SRJ | | SURTOT | | | 86 2 | PRINT SUBTOTAL |
| 3 | EB3A | 166 0 | | POS | | ADDN=P | | | 59 | POSITION TO ALF TAX |
| 40 0 | 2062 | 167 0 | | SRJ | | CKLINE | | | 98 0 | INCREMENT LINE COUNT |
| 1 | A622 | 168 0 | | NK | | 2 | 2 | | | XINDEX MISC TAX PERCENT |
| 2 | 45D0 | 169 0 | | EXZ | | 1 | | | | TEST IF ZERO INDEXED |
| 3 | 742E | 170 0 | | BRU | | ADDONS | | | 46 1 | BRANCH TO ADDONS |
| 41 0 | 4552 | 171 0 | | EX | | Y | 1 | 1 | | TEST FOR LAST LINE |
| 1 | 2850 | 172 0 | | SRJ | | CONTPG | | | 80 2 | BEGIN CONTINUATION PAGE |
| 2 | 4551 | 173 0 | | EX | | Y | 4 | 1 | | TEST FOR LAST SIX LINES |
| 3 | 285A | 174 0 | | SRJ | | ADLNCK | | | 90 2 | INCREMENT LINE COUNT |
| 42 0 | 6741 | 175 0 | | SET | | X | 4 | | | SET X FOR ADDONS |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| | 1 | E001 | 176 | 0 | AL | | 1 | | | | | ADVANCE LEFT ONE LINE |
| | 2 | EB3A | 177 | 0 | POS | | ADDN=P | | | | 59 | POSITION TO ALF TAX |
| | 3 | C054 | 178 | 0 | PC | | T | | | | | PRINT "TAX" |
| 43 | 0 | C041 | 179 | 0 | PC | | A | | | | | |
| | 1 | C058 | 180 | 0 | PC | | X | | | | | |
| | 2 | D443 | 181 | 0 | PN | | 4 | | 3 | | | PRINT TAX AS A PERCENT |
| | 3 | C025 | 182 | 0 | PC | | % | | | | | PRINT %: |
| 44 | 0 | EB4B | 183 | 0 | POS | | NET-P | | | | 76 | POS TO NET COLUMN |
| | 1 | 8C7C | 184 | 0 | MULR | | INVNET | | | | 124 | MULT TAX % X NET |
| | 2 | 2C4D | 185 | 0 | SRJ | | PNUMRC | | | | 77 3 | PRINT TAX DOLLARS |
| | 3 | 8083 | 186 | 0 | ADM | | ITAXES | | | | 131 | ADD TO TOTAL TAX DOLLARS |
| 45 | 0 | 807C | 187 | 0 | ADM | | INVNET | | | | 124 | ADD TO TOT NET INVOICE \$ |
| | 1 | 306F | 188 | 0 | TRM | | WRKREG | | | | 111 | STORE TAX DOLLARS |
| | 2 | 2062 | 189 | 0 | SRJ | | CKLINE | | | | 98 0 | INCREMENT LINE COUNT |
| | 3 | 4552 | 190 | 0 | EX | | Y | | 1 | 1 | | TEST IF LAST INVOICE LINE |
| 46 | 0 | 2856 | 191 | 0 | SRJ | | SURTOT | | | | 86 2 | PRINT SUB-TOTAL |
| | 1 | F620 | 192 | 0 | ADDONS | PKA | 1 6 | | | | | ENABLE TOTAL INVOICE PK |
| | 2 | EB3A | 193 | 0 | POS | | ADDN=P | | | | 59 | POSITION TO ADDON ALF |
| | 3 | 4241 | 194 | 0 | SK | | X | | 4 | 2 | | TEST IF FIRST ADDON |
| 47 | 0 | 4552 | 195 | 0 | EX | | Y | | 1 | 1 | | TEST IF LAST INVOICE LINE |
| | 1 | 2850 | 196 | 0 | SRJ | | CONTPG | | | | 80 2 | BEGIN CONTINUATION PAGE |
| | 2 | 4241 | 197 | 0 | SK | | X | | 4 | 2 | | TEST IF FIRST ADDON |

4-42

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 010

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| | 3 | 4551 | 198 0 | | EX | | Y | | 4 | 1 | | TEST IF BOTTOM OF INVOICE |
| 48 | 0 | 285A | 199 0 | | SRJ | | ADLNCK | | | | 90 2 | INCREMENT LINE COUNT |
| | 1 | 6741 | 200 0 | | SET | | X | | 4 | | | SET X4 INVOICE HAS ADDONS |
| | 2 | 4652 | 201 0 | | EX | | Y | | 1 | 2 | | TEST IF LAST INVOICE LINE |
| | 3 | 2856 | 202 0 | | SRJ | | SUBTOT | | | | 86 2 | PRINT SUBTOTAL |
| 49 | 0 | 2850 | 203 0 | | SRJ | | CONTPG | | | | 80 2 | BEGIN CONTINUATION PAGE |
| | 1 | ED01 | 204 0 | | AL | | 1 | | | | | ADVANCE LEFT ONE LINE |
| | 2 | EB3A | 205 0 | | POS | | ADDN-P | | | | 59 | POSITION TO ADDON ALF |
| | 3 | AC12 | 206 0 | | TK | | 18 | | | | | XTYPE ADDON DESC. |
| 50 | 0 | EB4B | 207 0 | | POS | | NET-P | | | | 76 | POSITION TO NET COLUMN |
| | 1 | A470 | 208 0 | | NKR | | 7 | | 0 | | | XTYPE ADDON COST AMOUNT |
| | 2 | 2C4D | 209 0 | | SRJ | | PNUMRC | | | | 77 3 | PRINT ADDON AMOUNT |
| | 3 | 45C1 | 210 0 | | EX | | A | | - | 1 | | TEST IF MINUS |
| 51 | 0 | 8080 | 211 0 | | ADM | | TADONM | | | | 128 | ADD TO ADDONS MINUS |
| | 1 | 41C1 | 212 0 | | SK | | A | | - | 1 | | TEST IF PLUS |
| | 2 | 807F | 213 0 | | ADM | | TADONP | | | | 127 | ADD TO ADDONS PLUSSES |
| | 3 | 807C | 214 0 | | ADM | | INVNET | | | | 124 | ADD TO TOT NET INVOICE \$ |
| 52 | 0 | 4558 | 215 0 | | EX | | Y | | 3 | 1 | | TEST IF STANDARD INVOICE |
| | 1 | 7836 | 216 0 | | BRU | | | + 9 | | | 54 2 | BRANCH TO CHECK LINE CT |
| | 2 | EB61 | 217 0 | | POS | | CSTX-P | | | | 98 | POSITION TO COST COLUMN |
| | 3 | A450 | 218 0 | | NKR | | 5 | | 0 | | | XINDEX ADDON COST AMOUNT |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| 53 | 0 | 204D | 219 | 0 | SRJ | | PNUMRC | | | | 77 3 | PRINT ADDON COST AMOUNT |
| | 1 | 807D | 220 | 0 | ADM | | INVCST | | | | 125 | ADD TO TOT INV TOT COST |
| | 2 | 45C1 | 221 | 0 | EX | | A | - | 1 | | | TEST IF MINUS |
| | 3 | 8082 | 222 | 0 | ADM | | TADNCM | | | | 130 | ADD TO COST TOTAL MINUS |
| 54 | 0 | 41C1 | 223 | 0 | SK | | A | - | 1 | | | TEST IF PLUS |
| | 1 | 8081 | 224 | 0 | ADM | | TADNCP | | | | 129 | ADD TO COST TOTAL PLUS |
| | 2 | 2062 | 225 | 0 | SRJ | | CKLINE | | | | 98 0 | INCREMENT LINE COUNT |
| | 3 | 742E | 226 | 0 | BRU | | ADDONS | | | | 46 1 | BRANCH TO NEXT ADDON |
| 55 | 0 | 4541 | 227 | 0 | TOTALI | EX | X | | 4 | 1 | | TEST IF ADDONS ON INVOICE |
| | 1 | 2856 | 228 | 0 | SRJ | | SURTOT | | | | 86 2 | PRINT TOTALS |
| | 2 | 4558 | 229 | 0 | EX | | Y | 3 | 1 | | | TEST IF STANDARD INVOICE |
| | 3 | 7801 | 230 | 0 | BRU | | BEGINV | | | | 1 2 | BRU TO NEXT INVOICE |
| 56 | 0 | 387C | 231 | 0 | TRA | | INVNET | | | | 124 | TRANSFER TOTAL NET INV. |
| | 1 | 986F | 232 | 0 | SUA | | WRKREG | | | | 111 | SUBT TAX FROM NET |
| | 2 | 987D | 233 | 0 | SUA | | INVCST | | | | 125 | SUBT COST FROM NET |
| | 3 | EB56 | 234 | 0 | POS | | PRT%-P | | | | 87 | POSITION TO PRINT PROFIT |
| 57 | 0 | 204D | 235 | 0 | SRJ | | PNUMRC | | | | 77 3 | PRINT PROFIT AMOUNT |
| | 1 | 7801 | 236 | 0 | BRU | | BEGINV | | | | 1 2 | BRANCH TO NEXT INVOICE |
| | 2 | 204D | 237 | 0 | INV-NO | SRJ | AR-POS | | | | 77 0 | ADVANCE FORM POSITION |
| | 3 | C886 | 238 | 0 | PA | | FRSTIN | | | | 182 | PRINT ALFA FRSTIN |
| 58 | 0 | EF02 | 239 | 0 | AR | | 2 | | | | | ADVANCE RIGHT TWO LINES |
| | 1 | ER77 | 240 | 0 | POS | | 120 | | | | | POSITION FOR PRINT |

444

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 012

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|--------------------------|
| | 2 | A660 | 241 | 0 | NK | 6 | | | 0 | | | XINDEX INVOICE NO. |
| | 3 | D450 | 242 | 0 | PN | 5 | | | 0 | | | PRINT INVOICE NUMBER |
| 59 | 0 | 3071 | 243 | 0 | TRM | | INVNO | | | | 113 | STORE INVOICE NO |
| | 1 | 204D | 244 | 0 | SRJ | | AR-POS | | | | 77 0 | ADVANCE FORM POSITION |
| | 2 | C8BE | 245 | 0 | PA | | DAIT | | | | 190 | PRINT ALFA DAIT |
| | 3 | EE02 | 246 | 0 | AR | | 2 | | | | | ADVANCE RIGHT TWO LINES |
| 60 | 0 | ER77 | 247 | 0 | POS | | 120 | | | | | POSITION FOR PRINT |
| | 1 | F073 | 248 | 0 | LKBR | | DATE | | | | 115 | LOAD DATE BASE M.A. |
| | 2 | AD0C | 249 | 0 | TKM | | 12 | | | | | XTYPE DATE |
| | 3 | 204D | 250 | 0 | SRJ | | AR-POS | | | | 77 0 | ADVANCE FORM POSITION |
| 61 | 0 | C8BA | 251 | 0 | PA | | TXRT | | | | 186 | PRINT ALFA TXRT |
| | 1 | ER77 | 252 | 0 | POS | | 120 | | | | | POSITION FOR PRINT |
| | 2 | EF02 | 253 | 0 | AR | | 2 | | | | | ADVANCE RIGHT TWO LINES |
| | 3 | A622 | 254 | 0 | NK | | 2 | | 2 | | | XINDEX TAX RATE |
| 62 | 0 | D473 | 255 | 0 | PN | | 7 | | 3 | | | PRINT TAX RATE |
| | 1 | 3079 | 256 | 0 | TRM | | TAXCST | | | | 121 | STORE FIRST TAX RATE |
| | 2 | 7C00 | 257 | 0 | BRU | | INITAL | | | | 0 3 | BRU TO INITIALIZE |
| | 3 | ER77 | 258 | 0 | POS | DLYTOT | 120 | | | | | POS SUMMARY HEADING |
| 63 | 0 | EF05 | 259 | 0 | AR | | 5 | | | | | ADVANCE RIGHT FIVE LINES |
| | 1 | C8BB | 260 | 0 | PA | | HFADNG | | | | 139 | PRINT ALFA MESSAGE |
| | 2 | ER8C | 261 | 0 | POS | | 141 | | | | | |
| | 3 | 0700 | 262 | 0 | RR | | | | | | | PRINT DATE IN RED |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|------------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| 64 | 0 | C873 | 263 | 0 | PA | | DATE | | | | 115 | PRINT CURRENT DATE |
| | 1 | EE02 | 264 | 0 | AR | | 2 | | | | | ADVANCE RIGHT TWO LINES |
| | 2 | ER77 | 265 | 0 | POS | | 120 | | | | | POSITION FOR PRINT |
| | 3 | C880 | 266 | 0 | PA | | T-GROSS | | | | 176 | PRINT ALFA MESSAGE |
| 65 | 0 | EE01 | 267 | 0 | AR | | 1 | | | | | ADVANCE RIGHT ONE LINE |
| | 1 | ER87 | 268 | 0 | POS | | 136 | | | | | POSITION FOR PRINT |
| | 2 | 3885 | 269 | 0 | TRA | | TGROSS | | | | 133 | TRANSFER TOTAL GROSS |
| | 3 | 2040 | 270 | 0 | SRJ | | PNUMRC | | | | 77 3 | PRINT TOTAL GROSS |
| 66 | 0 | 5100 | 271 | 0 | LIR | | 1 | | 0 | | | LOAD INDEX REGISTER |
| | 1 | 5200 | 272 | 0 | LIR | | 2 | | 0 | | | LOAD INDEX REGISTER |
| | 2 | 5A1C | 273 | 0 | DLYNET IIR | | 2 | | 28 | | | TEST FOR THE 7 LOOP |
| | 3 | 4584 | 274 | 0 | EX | | T | | I | 1 | | EXECUTE IF LOOP 7 |
| 67 | 0 | 7846 | 275 | 0 | BRU | | FINAL | | | | 70 2 | EXIT LOOP |
| | 1 | 5603 | 276 | 0 | ADIR | | 2 | | 3 | | | INCREMENT BY FOUR |
| | 2 | EF04 | 277 | 0 | AR | | 4 | | | | | ADVANCE RIGHT FOUR LINES |
| | 3 | ER77 | 278 | 0 | POS | | 120 | | | | | POSITION FOR PRINT |
| 68 | 0 | 6200 | 279 | 0 | MOD | | 2 | | | | | MODIFY BY REGISTER #2 |
| | 1 | C890 | 280 | 0 | PA | | TNS - 4 | | | | 144 | PRINT ALFA MESSAGE |
| | 2 | 6100 | 281 | 0 | MOD | | 1 | | | | | MODIFY BY REGISTER #1 |
| | 3 | 387F | 282 | 0 | TRA | | TNET | | | | 126 | TRANSFER NET, COST AMOUNT |
| 69 | 0 | EF02 | 283 | 0 | AR | | 2 | | | | | ADVANCE RIGHT TWO LINES |
| | 1 | ER87 | 284 | 0 | POS | | 136 | | | | | POSITION FOR PRINT |

446

| PROGRAM ID. - | | DATE RUN | 3/26/70 | TIME - 12:48 | VERSION 02-01-70 | PAGE 014 | | | | | | |
|---------------|-----|-------------|----------|--------------|------------------|---------------|-----------------|-------|-------|---------------|--------------------------|-------------------------|
| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. LABEL | A-PARAMETER INC | B PAR | C PAR | LABEL DEC EQU | REMARKS | |
| | 2 | D191 | 285 | 0 | PNS- | 9 | | 1 | | | SHIFT IF MINUS | |
| | 3 | C52D | 286 | 0 | PC- | - | | | | | PRINT - IF MINUS | |
| 70 | 0 | 5501 | 287 | 0 | ADIR | 1 | | 1 | | | INCREMENT INDEX REG BY 1 | |
| | 1 | 7842 | 288 | 0 | BRU | DLYNET | | | 66 | 2 | REPEAT LOOP | |
| | 2 | EE04 | 289 | 0 | FINAL AR | 4 | | | | | ADVANCE RIGHT FOUR LINES | |
| | 3 | EB77 | 290 | 0 | POS | 120 | | | | | POSITION FOR PRINT | |
| 71 | 0 | C882 | 291 | 0 | PA | TDISC | | | | 178 | | |
| | 1 | EE01 | 292 | 0 | AR | 1 | | | | | ADVANCE RIGHT ONE LINE | |
| | 2 | EB87 | 293 | 0 | POS | 136 | | | | | POSITION FOR PRINT | |
| | 3 | 3886 | 294 | 0 | TRA | TDISCT | | | | 134 | TOTAL DISCOUNTS | |
| 72 | 0 | 2C4D | 295 | 0 | SRJ | PNUMRC | | | | 77 | 3 | PRINT TOTAL DISCOUNTS |
| | 1 | EE02 | 296 | 0 | AR | 2 | | | | | | ADVANCE RIGHT TWO LINES |
| | 2 | EB77 | 297 | 0 | POS | 120 | | | | | | POSITION FOR PRINT |
| | 3 | C88F | 298 | 0 | PA | ACT | | | | 143 | | ALFA MESSAGE |
| 73 | 0 | EE01 | 299 | 0 | AR | 1 | | | | | | ADVANCE RIGHT ONE LINE |
| | 1 | 8880 | 300 | 0 | ADA | TADDNM | | | | 128 | | ADD MINUS ADDONS |
| | 2 | 887F | 301 | 0 | ADA | TADDNP | | | | 127 | | ADD PLUS ADDONS |
| | 3 | 887E | 302 | 0 | ADA | TNET | | | | 126 | | ADD PRODUCT NET |
| 74 | 0 | 2C4D | 303 | 0 | SRJ | PNUMRC | | | | 77 | 3 | PRINT ACCT REC NET |
| | 1 | EE02 | 304 | 0 | AR | 2 | | | | | | ADVANCE RIGHT TWO LINES |
| | 2 | EB77 | 305 | 0 | POS | 120 | | | | | | POSITION FOR PRINT |
| | 3 | C891 | 306 | 0 | PA | TNC | | | | 145 | | PRINT ALFA MESSAGE |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A=PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|--------------------------|----------|----------|------------------|--------------------------|
| 75 | 0 | EE01 | 307 | 0 | AR | 1 | | | | | ADVANCE RIGHT ONE LINE |
| | 1 | EB87 | 308 | 0 | POS | 136 | | | | | POSITION FOR PRINT |
| | 2 | 3884 | 309 | 0 | TRA | TCOSTS | | | 132 | | TRANSFER TOTAL COSTS |
| | 3 | 8881 | 310 | 0 | ADA | TADNCP | | | 129 | | ADD PLUS COST ADDONS |
| 76 | 0 | 8882 | 311 | 0 | ADA | TADNCM | | | 130 | | ADD MINUS COST ADDONS |
| | 1 | 2040 | 312 | 0 | SRJ | PNUMRC | | | 77 | 3 | PRINT TOTAL NET COSTS |
| | 2 | EE14 | 313 | 0 | AR | 20 | | | | | ADVANCE 20 LINES |
| | 3 | 7000 | 314 | 0 | BRU | INITAL | | | 0 | 3 | BRU TO INITIALIZE |
| 77 | 0 | EF02 | 315 | 0 | AR=POS | AR | 2 | | | | ADVANCE RIGHT TWO LINES |
| | 1 | EB77 | 316 | 0 | POS | SM=HDP | | | 120 | | POS PRINT |
| | 2 | 0400 | 317 | 0 | SRR | 1 | | | | | SUBROUTINE RETURN |
| | 3 | D191 | 318 | 0 | PNUMRC | PNS= | 9 | 1 | | | PRINT MONTERY VALUE |
| 78 | 0 | C520 | 319 | 0 | PC= | - | | | | | PRINT - IF MINUS |
| | 1 | 0400 | 320 | 0 | SRR | 1 | | | | | SUBROUTINE RETURN |
| | 2 | ER2A | 321 | 0 | PT=PRC | POS | SPRC=P | | 43 | | POSITION TO SELL PRICE |
| | 3 | 42C2 | 322 | 0 | SK | A | | S | 2 | | TEST PRICE FOR MILLS |
| 79 | 0 | D471 | 323 | 0 | PN | 7 | | 1 | | | PRINT WITH CENTS MASK |
| | 1 | 0220 | 324 | 0 | SLRD | 2 | | 0 | | | SHIFT LEFT TWO POSITIONS |
| | 2 | 45C2 | 325 | 0 | EX | A | | S | 1 | | TEST PRICE FOR MILLS |
| | 3 | D472 | 326 | 0 | PN | 7 | | 2 | | | PRINT WITH MILLS MASK |
| 80 | 0 | 307B | 327 | 0 | TRM | PRICE | | | 123 | | TRANSFER SELL PRICE |

448

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 016

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-----------------------|-------|-------|---------------|--------------------------|
| | 1 | 0400 | 328 0 | | SRR | 1 | | | | | SUBROUTINE RETURN |
| | 2 | 3070 | 329 0 | CONTPG | TRM | | WRKREG + 1 | | | 112 | TEMPORARY STORE AMOUNT |
| | 3 | E80F | 330 0 | | OC | | SLDTOL + 7 | | | 15 | ADVANCE TO CUSTOMER NAME |
| 81 | 0 | AC00 | 331 0 | | TK | 0 | | | | | CHANGE INVOICE |
| | 1 | ER0C | 332 0 | | POS | | NMAD-P | | | 13 | POSITION TO PRINT |
| | 2 | C875 | 333 0 | | PA | | CUSTNM | | | 117 | PRINT CUSTOMER NAME |
| | 3 | E008 | 334 0 | | LLCR | | SLDTOL | | | 8 | LOAD SOLD TO LINE NO. |
| 82 | 0 | E90D | 335 0 | | ALTD | | SHPTOL | | | 13 | ADVANCE TO SHIP TO LINE |
| | 1 | ER3F | 336 0 | | POS | | CNTP-P | | | 64 | POSITION TO PRINT |
| | 2 | C050 | 337 0 | | PC | | P | | | | PRINT "PAGE" |
| | 3 | C041 | 338 0 | | PC | | A | | | | |
| 83 | 0 | C047 | 339 0 | | PC | | G | | | | |
| | 1 | C045 | 340 0 | | PC | | E | | | | |
| | 2 | 3872 | 341 0 | | TRA | | PAGENO | | | 114 | TRANSFER PAGE NUMBER |
| | 3 | D420 | 342 0 | | PN | 2 | | 0 | | | PRINT PAGE NO. |
| 84 | 0 | 8F01 | 343 0 | | ADK | 0 | | 1 | | | ADD 1 TO PAGE NO. |
| | 1 | 3072 | 344 0 | | TRM | | PAGENO | | | 114 | STORE PAGE NO. |
| | 2 | 3871 | 345 0 | | TRA | | INVNO | | | 113 | TRANSFER INVOICE NO. |
| | 3 | 42D0 | 346 0 | | SKZ | 2 | | | | | TEST IF ZERO |
| 85 | 0 | 9F01 | 347 0 | | SUK | 0 | | 1 | | | SUBT 1 FROM INVOICE NO. |
| | 1 | 3071 | 348 0 | | TRM | | INVNO | | | 113 | STORE INVOICE NO. |
| | 2 | E914 | 349 0 | | ALTD | | RIBBL | | | 20 | ADVANCE TO RIBBON LINE |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | R E M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|--------------------------|----------|----------|------------------|---------------------------|
| | 3 | 285E | 350 0 | | SRJ | | DATE-J | | | 94 2 | PRINT DATE & INVOICE NO |
| 86 | 0 | 3870 | 351 0 | | TRA | | WRKREG + 1 | | | 112 | TRANSFER STORED AMOUNT |
| | 1 | 0400 | 352 0 | | SRR | | 1 | | | | SUBROUTINE RETURN |
| | 2 | 3070 | 353 0 | SUBTOT | TRM | | WRKREG + 1 | | | 112 | TEMPORARILY STORE AMOUNT |
| | 3 | EB4E | 354 0 | | POS | | NET-P + 3 | | | 79 | POSITION FOR UNDERSCORE |
| 87 | 0 | C889 | 355 0 | | PA | | UNDERS | | | 137 | PRINT UNDERSCORE |
| | 1 | 4258 | 356 0 | | SK | | Y | 3 | 2 | | TEST FOR INVOICE COSTING |
| | 2 | EB64 | 357 0 | | POS | | CSTX-P + 3 | | | 101 | POSITION FOR UNDERSCORE |
| | 3 | C889 | 358 0 | | PA | | UNDERS | | | 137 | PRINT UNDERSCORE |
| 88 | 0 | E001 | 359 0 | | AL | | 1 | | | | ADVANCE LEFT ONE LINE |
| | 1 | EB4B | 360 0 | | POS | | NET-P | | | 76 | POSITION TO NET AMOUNT |
| | 2 | 387C | 361 0 | | TRA | | INVNET | | | 124 | TRANSFER TOTAL NET AMOUNT |
| | 3 | 2C4D | 362 0 | | SRJ | | PNUMRC | | | 77 3 | PRINT TOTAL NET AMOUNT |
| 89 | 0 | 4358 | 363 0 | | SK | | Y | 3 | 3 | | TEST FOR INVOICE COSTING |
| | 1 | EB61 | 364 0 | | POS | | CSTX-P | | | 98 | POSITION TO COST AMOUNT |
| | 2 | 387D | 365 0 | | TRA | | INVCST | | | 125 | TRANSFER TOT. COST AMOUNT |
| | 3 | 2C4D | 366 0 | | SRJ | | PNUMRC | | | 77 3 | PRINT TOTAL COST AMOUNT |
| 90 | 0 | 3870 | 367 0 | | TRA | | WRKREG + 1 | | | 112 | TRANSFER STORED AMOUNT |
| | 1 | 0400 | 368 0 | | SRR | | 1 | | | | SUBROUTINE RETURN |
| | 2 | 5402 | 369 0 | ADLNCK | ADIR | | 4 | 2 | | | ADD 2 TO LINE COUNT |
| | 3 | 4684 | 370 0 | | EX | | T | 1 | 2 | | TEST FOR LAST LINE |
| 449 | 91 | 0 | 2850 | 371 0 | SRJ | | CONTPG | | | 80 2 | PRINT TOTAL & CONT. PAGE |

450

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 018

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|-------------------------|
| | 1 | 0400 | 372 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |
| | 2 | 5CE8 | 373 | 0 | DIR | 4 | | | 232 | | | SUBT 1 FROM LINE COUNT |
| | 3 | 0400 | 374 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |
| 92 | 0 | 5300 | 375 | 0 | CLRMEM | LIR | | 3 | 0 | | | LOAD FOR COUNTER |
| | 1 | 6300 | 376 | 0 | | MOD | | 3 | | | | MODIFY WITH COUNTER |
| | 2 | 087C | 377 | 0 | | CLM | INVNET | | | | 124 | CLEAR MEMORY LOCATION |
| | 3 | 580A | 378 | 0 | | IIR | | 3 | 10 | | | CONTROL NUMBER OF LOOPS |
| 93 | 0 | 4484 | 379 | 0 | | EX | T | | T | 4 | | TEST IF LAST TOTAL |
| | 1 | 8E00 | 380 | 0 | | CLA | 0 | | 0 | | | CLEAR ACCUMULATOR |
| | 2 | 204D | 381 | 0 | | SRJ | AR-POS | | | | 77 0 | ADVANCE FORM POSITION |
| | 3 | D411 | 382 | 0 | | PN | 1 | | 1 | | | PRINT ZERO |
| 94 | 0 | 7C00 | 383 | 0 | | BRU | INITAL | | | | 0 3 | BRU TO INITIALIZE |
| | 1 | 745C | 384 | 0 | | BRU | CLRMEM + 1 | | | | 92 1 | BRU TO CLEAR NEXT TOTAL |
| | 2 | ER41 | 385 | 0 | DATE-J | POS | DATE-P | | | | 66 | POSITION TO DATE |
| | 3 | C873 | 386 | 0 | | PA | DATE | | | | 115 | PRINT CURRENT DATE |
| 95 | 0 | ER4F | 387 | 0 | | POS | INVNOP | | | | 80 | POSITION TO INVOICE NO. |
| | 1 | 3871 | 388 | 0 | | TRA | INVNO | | | | 113 | TRANSFER INVOICE NO. |
| | 2 | 43D0 | 389 | 0 | | SKZ | 3 | | | | | TEST FOR ZERO |
| | 3 | D450 | 390 | 0 | | PN | 5 | | 0 | | | PRINT INVOICE NUMBER |
| 96 | 0 | 8F01 | 391 | 0 | | ADK | 0 | | 1 | | | ADD 1 TO INVOICE NO. |
| | 1 | 3071 | 392 | 0 | | TRM | INVNO | | | | 113 | STORE INVOICE NO |
| | 2 | 45D0 | 393 | 0 | | EXZ | 1 | | | | | TEST FOR ZERO |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | TNC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| | 3 | AC07 | 394 | 0 | TK | 7 | | | | | | XTYPE INVOICE NO. |
| 97 | 0 | 50EA | 395 | 0 | LIR | 4 | | | 234 | | | LOAD FOR LINE COUNT |
| | 1 | 6553 | 396 | 0 | RST | Y | | | 41 | | | RESET Y 4 AND 1 |
| | 2 | E916 | 397 | 0 | ALTO | BODYL | | | | 22 | | ADVANCE TO INVOICE BODY |
| | 3 | 0400 | 398 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |
| 98 | 0 | 4051 | 399 | 0 | CKLINE | SK | Y | | 4 | 4 | | TEST IF LAST SIX LINES |
| | 1 | 58F8 | 400 | 0 | IIR | 4 | | | 248 | | | INCREMENT FOR LINE COUNT |
| | 2 | 4184 | 401 | 0 | SK | T | | | 1 | 1 | | TEST FOR 17TH LINE |
| | 3 | 0400 | 402 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |
| | | | 403 | 0 | NOTE | | | | | | | SET Y FOR LAST 6 LINES IN |
| 99 | 0 | 6751 | 404 | 0 | SET | Y | | | 4 | | | BODY OF INVOICE REACHED |
| | 1 | 58FF | 405 | 0 | IIR | 4 | | | 255 | | | INCREMENT LINE COUNT |
| | 2 | 0980 | 406 | 0 | ALARM | | | | | | | WARN OPERATOR BOTTOM INV. |
| | 3 | 4584 | 407 | 0 | FX | T | | | 1 | 1 | | TEST FOR LAST LINE |
| 100 | 0 | 6752 | 408 | 0 | SET | Y | | | 1 | | | SET Y1 FOR CONT. PAGE |
| | 1 | 0400 | 409 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |
| | 2 | 306F | 410 | 0 | CTMILL | TRM | WRKREG | | | 111 | | TRANSFER PRICE TO M.A. |
| | 3 | 0200 | 411 | 0 | SLRO | 13 | | 0 | | | | SHIFT OFF DOLLARS & CENTS |
| 101 | 0 | 4700 | 412 | 0 | EXZ | 3 | | | | | | TEST IF ZERO MILLS |
| | 1 | 386F | 413 | 0 | TRA | WRKREG | | | | 111 | | TRANSFER PRICE |
| | 2 | 0202 | 414 | 0 | SLRO | 0 | | 2 | | | | REPOSITION FOR CENTS ONLY |
| | 3 | 0400 | 415 | 0 | SRR | 1 | | | | | | SUBROUTINE RETURN |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | R PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|-------|-------|---------------|---------------------------|
| 102 | 0 | 386F | 416 | 0 | TRA | | WRKREG | | | | 111 | TRANSFER PRICE |
| | 1 | 0207 | 417 | 0 | SLRN | | 0 | | 7 | | | SHIFT OFF 7 DIGITS |
| | | | 418 | 0 | NOTE | | | | | | | TEST IF MORE THAN SEVEN |
| | 2 | 43D0 | 419 | 0 | SKZ | | 3 | | | | | DIGITS WERE INDEXED |
| | 3 | 67C1 | 420 | 0 | SET | | A | | - | | | YES, SET = FLAG |
| 103 | 0 | 0980 | 421 | 0 | ALARM | | | | | | | WARN OPERATOR ERROR PRICE |
| | 1 | 0400 | 422 | 0 | SRR | | 1 | | | | | SUBROUTINE RETURN |
| | 2 | 386F | 423 | 0 | TRA | | WRKREG | | | | 111 | TRANSFER PRICE |
| | | | 424 | 0 | NOTE | | | | | | | SET FLAG TO INDICATE |
| | 3 | 67C2 | 425 | 0 | SET | | A | | S | | | CENTS AND MILLS |
| 104 | 0 | 0400 | 426 | 0 | SRR | | 1 | | | | | SUBROUTINE RETURN |
| | 1 | 6558 | 427 | 0 | IVCOST | RST | Y | | 3 | | | RESET STANDARD INV. FLAG |
| | 2 | 7801 | 428 | 0 | BRU | | BEGINV | | | | 1 2 | BRU TO INVOICE ROUTINE |
| | 3 | 0000 | | | STOP | | | | | | | ** INSERTED BY ASSEM. ** |
| | | | 429 | 0 | WORD | | | | | | | PROGRAM KEY TABLE |
| 105 | 0 | 7839 | 430 | 0 | PKTABL | BRU | INV-NO | | | | 57 2 | BRU LOAD DATE INV NO TAX |
| | 1 | 7C3E | 431 | 0 | BRU | | DLYTOT | | | | 62 3 | BRU TO PRINT DAILY TOTALS |
| | 2 | 7C26 | 432 | 0 | BRU | | INVTOT | | | | 38 3 | BRU TO SUBTOTAL |
| | 3 | 3879 | 433 | 0 | TRA | | TAXCST | | | | 121 | TRANSFER TAX PERCENT |
| 106 | 0 | 7468 | 434 | 0 | BRU | | IVCOST | | | | 104 1 | BRU TO COST INVOICE |
| | 1 | 7037 | 435 | 0 | BRU | | TOTALI | | | | 55 0 | BRU TO TOTAL INVOICE |
| | 2 | 7C00 | 436 | 0 | BRU | | INITAL | | | | 0 3 | BRU TO INITIALIZE |

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 021

| WORD | OBJECT | SEQ. | SYM. | OP | FD. | A-PARAMETER | B | C | LABEL | REMARKS |
|------|--------|------|------|------|-----|-------------|-----|-----|---------|---------------------|
| SYL | CODE | NO. | LOC. | CODE | LN. | LABEL INC | PAR | PAR | DEC EQU | |
| 3 | 705C | 437 | 0 | BRU | | CLRMEM | | | 92 0 | BRU TO CLEAR MEMORY |
| | | 438 | 0 | PAGE | | | | | | |

454

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 022

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|------------------|-------|-------|---------------|--------------------------|
| 107 | 0 | | 439 | 0 | MASKTB | MASK | 15 | Z77,ZZZ,ZZZ,ZZZ | | | | NUMERIC |
| 107 | 0 | E666 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 6F66 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 66E6 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0333 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 108 | 0 | | 440 | 0 | | MASK | 16 | Z.7ZZ,ZZZ,ZZZ,DD | | | | MONETARY ZERO PRINT |
| 108 | 0 | 6697 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 66E6 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | E66E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0333 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 109 | 0 | | 441 | 0 | | MASK | 15 | Z7,ZZZ,ZZZ,CCCC | | | | MONETARY WITH MILLS |
| 109 | 0 | 8000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | E666 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 6E66 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0333 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 110 | 0 | | 442 | 0 | | MASK | 16 | Z777EZZZZZZZZ,XS | | | | HUNDREDS PERCENT \$ NO. |
| 110 | 0 | 66C2 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 6666 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 6166 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0666 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 111 | 0 | | 443 | 0 | WRKREG | REG | | 2 | | | | GENERAL WORK REGISTER |
| 113 | 0 | | 444 | 0 | INVNO | REG | | 1 | | | | INVOICE NUMBER |
| 114 | 0 | | 445 | 0 | PAGENO | REG | | 1 | | | | PAGE NUMBER |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|----------------|-------------|--------------|------------|------------|--------------------------|----------|----------|------------------|--------------------------|
| 115 | 0 | | 446 | 0 | DATE | REG | 2 | | | | CURRENT DATE |
| 117 | 0 | | 447 | 0 | CUSTNM | REG | 4 | | | | CUSTOMER NAME |
| 121 | 0 | | 448 | 0 | TAXCST | REG | 1 | | | | TAX CONSTANT |
| 122 | 0 | | 449 | 0 | QTY | REG | 1 | | | | INVOICE QUANTITY |
| 123 | 0 | | 450 | 0 | PRICE | REG | 1 | | | | INVOICE PRICE |
| 124 | 0 | | 451 | 0 | INVNET | REG | 1 | | | | INVOICE NET TOTAL |
| 125 | 0 | | 452 | 0 | INVCST | REG | 1 | | | | INVOICE COST TOTAL |
| 126 | 0 | | 453 | 0 | TNET | REG | 1 | | | | TOTAL NET SALES |
| 127 | 0 | | 454 | 0 | TADONP | REG | 1 | | | | TOTAL PLUS ADDONS |
| 128 | 0 | | 455 | 0 | TADONM | REG | 1 | | | | TOTAL MINUS ADDONS |
| 129 | 0 | | 456 | 0 | TADNCP | REG | 1 | | | | TOTAL PLUS COST ADDONS |
| 130 | 0 | | 457 | 0 | TADNCM | REG | 1 | | | | TOTAL MINUS ADDONS |
| 131 | 0 | | 458 | 0 | TTAXES | REG | 1 | | | | TOTAL TAXES |
| 132 | 0 | | 459 | 0 | TCOSTS | REG | 1 | | | | TOTAL COSTS |
| 133 | 0 | | 460 | 0 | TGROSS | REG | 1 | | | | TOTAL GROSS SALES |
| 134 | 0 | | 461 | 0 | TDISCT | REG | 1 | | | | TOTAL DISCOUNTS |
| 135 | 0 | | 462 | 0 | TEST | NUM | 4 | 5000 | | | TEST VALUE 5000 |
| 135 | 0 | 5000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 136 | 0 | | 463 | 0 | SAME | ALF | 4 | SAME | | | |
| 136 | 0 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |

4-56

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 024

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R E M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|----------------------|----------------------|----------|----------|------------------|--------------------------|
| | 2 | 4045 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5341 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 137 | 0 | | 464 | 0 | UNDERS | ALF | 11 | ----- | | | | UNDERSCORE |
| 137 | 0 | 5F5F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 5F5F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5F5F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5F5F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 138 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5F00 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5F5F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 139 | 0 | | 465 | 0 | HEADNG | ALF | 24 | DAILY TOTALS FOR.... | | | | |
| 139 | 0 | 2054 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4059 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4149 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2044 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 140 | 0 | 464F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 414C | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4F54 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 141 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2F20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LAREL | INC | B PAR | C PAR | LABEL DEC EQU | R E M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|----------------------|-------|----------|----------|------------------|--------------------------|
| | 2 | 2F2E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 522E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 142 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 143 | 0 | | 466 | 0 | ACT | ALF | 12 | ACC | REC | NET | | |
| 143 | 0 | 4543 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2052 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4343 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2041 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 144 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4554 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 204E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 145 | 0 | | 467 | 0 | TNC | ALF | 16 | TOTAL | NET | COST | | |
| 145 | 0 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 146 | 0 | 5354 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 434F | | | | | | | | | | ** INSERTED BY ASSEM. ** |

4-58

PROGRAM ID. -

DATE RUN 3/26/70 TIME = 12:48

VERSION 02-01-70

PAGE 026

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R E M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|----------------------|-----------------|----------|----------|------------------|--------------------------|
| | 2 | 5420 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4F45 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 147 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 148 | 0 | | 468 | 0 | TNS | ALF | 24 | TOTAL NET SALES | | | | |
| 148 | 0 | 4F45 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 149 | 0 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C45 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5341 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5420 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 150 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 151 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 027

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|--------------------|-----|-------|-------|---------------|--------------------------|
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 152 | 0 | | 469 | 0 | ALF | 24 | TOTAL PLUS ADDONS | | | | | |
| 152 | 0 | 504C | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 153 | 0 | 4F4E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4444 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2041 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5553 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 154 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 155 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 156 | 0 | | 470 | 0 | ALF | 24 | TOTAL MINUS ADDONS | | | | | |
| 156 | 0 | 4D49 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

4-59

460

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 028

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|------------------------|-----|-------|-------|---------------|--------------------------|
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 157 | 0 | 444F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4144 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4F55 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 158 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4E53 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 159 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 160 | 0 | | 471 | 0 | ALF | 24 | TOTAL PLUS COST ADDONS | | | | | |
| 160 | 0 | 504C | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 161 | 0 | 5420 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4F53 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2043 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5553 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|---------|---------|-------------------------|-----|-------|-------|---------------|--------------------------|
| 162 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4F53 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 444F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4144 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 163 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 164 | 0 | | 472 | 0 | ALF | 24 | TOTAL MINUS COST ADDONS | | | | | |
| 164 | 0 | 4049 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 165 | 0 | 5354 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 434F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4E55 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 166 | 0 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4F4E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4444 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2041 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

462

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 030

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R E M A R K S |
|------|-----|----------------|-------------|--------------|------------|------------|----------------------|-----|----------|----------|------------------|--------------------------|
| 167 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 168 | 0 | | 473 | 0 | ALF | 24 | TOTAL TAXES | | | | | |
| 168 | 0 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 169 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5845 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 170 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 171 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 172 | 0 | | 474 | 0 | ALF | 24 | TOTAL COSTS | | | | | |

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 031

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-------------|-------|-------|---------------|--------------------------|
| 172 | 0 | 434F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 173 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 5354 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 174 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 175 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 176 | 0 | | 475 | 0 | T-GROSS | ALF | 14 | TOTAL GROSS | | | | |
| 176 | 0 | 4C20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 544F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 177 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

4-64

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 032

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|----------------|-------------|--------------|------------|------------|----------------------|----------------------|----------|----------|------------------|--------------------------|
| | 1 | 5320 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4F53 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4752 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 178 | 0 | | 476 | 0 | TDISC | ALF | 24 | TOTAL DISCOUNTS | | | | |
| 178 | 0 | 2044 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 414C | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4F54 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2054 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 179 | 0 | 5453 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 554E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 434F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4953 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 180 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 181 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 182 | 0 | | 477 | 0 | FRSTIN | ALF | 24 | STARTING INVOICE NO. | | | | |
| 182 | 0 | 494E | | | | | | | | | | ** INSERTED BY ASSEM. ** |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | R PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|----------|-------|-------|---------------|--------------------------|
| | 1 | 5254 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5441 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2053 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 183 | 0 | 4943 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 564F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 494E | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4720 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 184 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2E20 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4E4F | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4520 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 185 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 186 | 0 | | 478 | 0 | TXRT | ALF | 24 | TAX RATE | | | | |
| 186 | 0 | 4154 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2052 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4158 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2054 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 187 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

4-66

PROGRAM ID. -

DATE RUN 3/26/70 TIME - 12:48

VERSION 02-01-70

PAGE 034

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B PAR | C PAR | LABEL DEC EQU | R F M A R K S |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|--------------|-------|-------|---------------|--------------------------|
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 4520 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 188 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 189 | 0 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 190 | 0 | | 479 | 0 | DAIT | ALF | 24 | CURRENT DATE | | | | |
| 190 | 0 | 4F54 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 5245 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 5552 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2043 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 191 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 4520 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 4154 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 2044 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 192 | 0 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 2020 | | | | | | | | | | ** INSERTED BY ASSEM. ** |

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | B PAR | C PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-------|-------|---------------|--------------------------|
| | 3 | 2020 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| 193 | 0 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 1 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 2 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | 3 | 0000 | | | | | | | | | ** INSERTED BY ASSEM. ** |
| | | | 480 | 0 | SLDTOL | DEF | 8 | | | | SOLD TO LINE |
| | | | 481 | 0 | SHPTOL | DEF | 13 | | | | SHIP TO LINE |
| | | | 482 | 0 | RIBBL | DEF | 20 | | | | RIBBON LINE |
| | | | 483 | 0 | BODYL | DEF | 22 | | | | BODY OF INVOICE LINE |
| | | | 484 | 0 | NMAD-P | DEF | 13 | | | | POS. NAME ADDRESS |
| | | | 485 | 0 | TERM-P | DEF | 6 | | | | POS TERMS |
| | | | 486 | 0 | ORDNOP | DEF | 19 | | | | POS ORDER NUMBER |
| | | | 487 | 0 | CUSNOP | DEF | 31 | | | | POS CUSTOMER NUMBER |
| | | | 488 | 0 | SLDBYP | DEF | 42 | | | | POS SOLD BY NAME |
| | | | 489 | 0 | SHVIP | DEF | 56 | | | | POS SHIP VIA |
| | | | 490 | 0 | DATE-P | DEF | 56 | | | | POS DATE |
| | | | 491 | 0 | INVNOP | DEF | 80 | | | | POS INVOICE NUMBER |
| | | | 492 | 0 | PRCDP | DEF | 5 | | | | POS PRODUCT NUMBER |
| | | | 493 | 0 | QTY-P | DEF | 7 | | | | POS QUANTITY |
| | | | 494 | 0 | DESC-P | DEF | 22 | | | | POS DESCRIPTION |
| | | | 495 | 0 | SPRC-P | DEF | 43 | | | | POS SELL PRICE |
| | | | 496 | 0 | PERU-P | DEF | 55 | | | | POS PER UNIT COLUMN |
| | | | 497 | 0 | GRS-P | DEF | 56 | | | | POS GROSS AMOUNT |
| | | | 498 | 0 | DISC-P | DEF | 68 | | | | POS DISCOUNT PERCENT |

Burroughs
TOOL DIVISION



INVOICE
Burroughs Ⓟ
DEMONSTRATING FORM

SOLD TO : ACME TOOL AND DIE
4444 MAIN STREET
BAKEWELL, OHIO
43122

SHIP TO : SAME

| TERMS | ORDER NO. | CUSTOMER NO. | SOLD BY | SHIP VIA | DATE | INVOICE NO. |
|-------|-----------|--------------|---------|----------|-------------|-------------|
| COD | 23-A67 | ATD-900 | MLS | RAIL | MAR 15 1970 | 105 |

| CODE | QUANTITY | DESCRIPTION | PRICE | UNIT | GROSS | DISCOUNT | NET | COST PRICE | COST EXTENSION |
|------|----------|----------------------------------|-------|------|-----------|----------|------------------|------------|------------------|
| 6 | 10 | ASST NUTS AND BOLTS WITH WASHERS | 1.00 | EA | 10.00 | | 10.00 | .10 | 1.00 |
| 12 | 5 | REVERSIBLE RATCHETS | 11.50 | EA | 57.50 | | 57.50 | 4.50 | 22.50 |
| 121 | 100 | ASST SIZES SOCKETS | 1.20 | EA | 120.00 | | 120.00 | .67 | 67.00 |
| 205 | 6 | TOOL CHESTS | 4.75 | EA | 28.50 | | 28.50 | 2.18 | 13.08 |
| 99 | 5,000 | NONLEAK RUBBER RUBBER-GASKETS | 2.00 | C | 100.00 | 23 % | 77.00 | .02 | 1.00 |
| 75 | 25 000 | COLD CHISELS | 60.00 | C | 15,000.00 | 12 % | 13,200.00 | 49.00 | 12,250.00 |
| | | | | | TAX 9.12% | | 13,493.00 | | 12,354.58 |
| | | | | | | | 1,230.56 | | |
| | | | | | | | <u>14,723.56</u> | 1,138.42 | <u>12,354.58</u> |

CROSS REFERENCE TABLE BETWEEN
PROGRAM DEFINITION CHART SEQUENCE NUMBER AND
ASSEMBLER III OUTPUT SEQUENCE NUMBER

| <u>FUNCTION</u> | <u>PROGRAM DEFINITION CHART SEQUENCE NUMBERS</u> | <u>ASSEMBLER III OUTPUT SEQUENCE NUMBERS</u> |
|---------------------------------|--|--|
| Invoice | 1. | 2410 - 2430 |
| Date | 2. | 2480 - 2490 |
| Tax Rate | 3. | 2540 - 2550 |
| Clear Daily Totals | 4. | 3750 - 3840 |
| Choice of Invoice | 5. | 50 - 100 |
| Sold-to and Ship-to Information | 6. 7. | 160 - 420 |
| Terms | 8. | 440 - 450 |
| Order Number | 9. | 460 - 470 |
| Customer Number | 10. | 480 - 490 |
| Sold-by-Information | 11. | 500 - 510 |
| Ship-via-Information | 12. | 520 - 530 |
| Print Date | 13. | 560, 3850 - 3860 |
| Print Invoice Number | 14. | 3870 - 3940 |
| Product Number | 15. | 650 - 720 |
| Quantity | 16. | 800 - 820 |
| Product Description | 17. | 880 - 890 |
| Sell Price | 18. | 950 - 1220 |
| Unit | 19. | 1060, 1110, 1220 |
| Gross | 20. | 1230 - 1240 |
| Discount | 21. | 1290 - 1350 |
| Net Amount | 22. | 1410 - 1430 |
| Cost Price | 23. | 1490 - 1500 |
| Cost Amount | 24. | 1560 - 1580 |
| Invoice Subtotal | 25. | 1620 - 1650 |
| Tax Per Cent | 26. | 1666 - 1682 |
| Tax Amount | 27. | 1840 |
| Add Ons | 28. | 1920 - 2060 |
| Cost Add On | 29. | 2070 - 2260 |
| Invoice Total | 30. | 1620 - |
| Automatic Continuation Pages | 31. | 3290 - 3520 |
| Calculate and Print Profit | 32. | 2310 - 2350 |
| Print Totals | 33. | 2580 - 3140 |

SECTION 5

ASSEMBLERS

5.00.00 FUNCTIONAL DESCRIPTION OF BASIC ASSEMBLERS

An assembler is a program or system of programs which prepares a machine language program from a symbolic language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic address.

Versions of the Series L/TC Assembler are available for several of Burroughs Computer systems. The processing time and operation is different in varying degrees from one version to the other although the functions of all versions are basically the same. These functions include error detection, preparation of object program media, symbolic and object program listings, and other operating and debugging aids. Input to each version of the L/TC Assembler consists of GP 300 instructions in the format specified by the Burroughs Assembler Coding Form (MKTG 2296) (See Section 1 of this manual).

5.01.00 L/TC PAPER TAPE ASSEMBLERS

The paper tape assemblers are two phase (two pass) assemblers. Each phase is a separate program and must be loaded prior to its operation. The series L/TC keyboard is used for Phase I input. The output consists of a Phase I listing as well as a symbolic paper tape. This symbolic paper tape is then used as input for Phase II. Phase II output consists of a phase II assembler listing and an object program tape.

The paper tape assemblers which operate exclusively on L/TC equipment are Assembler I, Assembler ISL, Assembler VI, Assembler VIII. The operating instructions for these assemblers are the same except for additional options available with some assemblers as cited below.

Assembler I

Assembler I operates on TC 500/700, L 2000, L 3000, and L 4000 32-track machines, and will generate code for these machines, and also the L 2000, L 3000, and L 4000 40-track machines.

Assembler ISL

Assembler ISL operates on TC 500/700, L 2000, L 3000, and L 4000 32-track machines, and generates code exclusively for the L 5000. Note striped ledger options in Topic 5.02.04.

Assembler VI

Assembler VI operates on the L 2000, L 3000, and L 4000 40-track machines, and generates code for these machines and the L 5000, and also the 32-track machines. Note striped ledger options in Topic 5.02.04.

Assembler VIII

Assembler VIII operates exclusively on the L 5000, and generates code for the TC 500/700, L 2000, L 3000, and L 4000 32-track machines for the L 2000, L 3000, and L 4000 40-track machines and for the L 5000. Note striped ledger options in Topic 5.02.04.

5.01.01 ENVIRONMENT

Equipment required for the assemblers mentioned above is as follows:

Assembler I, and ISL are compatible with any one of the following 32-track full memory Series L/TC's: TC 525, TC 725, TC 1545, TC 2545, L 2101-608, L 2301-608, L 3111-608, L 3311-608, or L 4311-608.

Assembler VI is compatible with any one of the following extended memory Series L's: L 2302-908, L 4312-909, L 3312-908.

Assembler VIII is compatible with the following full memory Series L 5112-609.

These assemblers require the following peripherals and firmware:

A 581 Paper Tape Reader

A 562 Paper Tape Punch

| Assembler | Hardware | Firmware Required* |
|---------------------|-------------------------|--------------------|
| Assembler I and ISL | TC 525, TC 725, TC 1545 | 2-1021-001-XX |
| Assembler I and ISL | L 2101-608, L 2301-608 | 2-1001-001-XX |
| Assembler I and ISL | L 3111-608, L 3311-608 | 2-1001-001-XX |
| Assembler I and ISL | L 4311-608 | 2-1301-001-XX |
| Assembler I and ISL | TC 2545 | 2-1321-001-XX |
| Assembler VI | L 2302-908, L 3312-908 | 2-1101-001-XX |
| Assembler VI | L 4312-909 | 2-1401-001-XX |
| Assembler VIII | L 5112-609 | 2-1201-001-XX |

* XX denotes latest revision

Applicable Program I. D. numbers are as follows:

| Assembler | Program I. D. | |
|----------------|---------------|---------------|
| | Phase I | Phase II |
| Assembler I | 1-1001-008-XX | 1-1001-009-XX |
| Assembler ISL | 1-1001-069-XX | 1-1001-070-XX |
| Assembler VI | 1-1101-001-XX | 1-1101-002-XX |
| Assembler VIII | 1-1201-022-XX | 1-1201-023-XX |

5.01.02 PHASE I

Phase I of the assemblers operates under 3 modes: 1) Keyboard Mode, 2) Correction Mode, and 3) Continuation Mode. Under Keyboard Mode, the symbolic instructions are entered on the Series L/TC keyboard, a label Table is built up in memory, a Phase I listing is prepared on the printer, and a symbolic paper tape is punched. Under Correction Mode, instructions may be changed, added or deleted in the symbolic paper tape. Correction Mode allows the Phase I assembly process to be resumed after an interruption. Phase I also has diagnostic facilities for the detection and indication of errors.

Phase I – Input

The input of Phase I of the assembler program is comprised of the labels, symbolic operation codes, parameters and remarks which are entered sequentially via the keyboard or the tape reader (in the case of continuation mode or correction mode).

Phase I – Operating Instructions

In operating the assemblers pin fed continuous forms, a minimum of 11¼ inches in width, must be used. The left edge of the pin fed form should be at position 5 on the scale and the positioned forms are visible along the bottom form bail. Both Pass I and Pass II object program tapes include their own firmware. Therefore, when loading the assembler programs, all 32 tracks of main memory (words 0-1023) must be unprotected.

1. Via Memory Loader Device: Load in normal manner. (See Section 6 of this manual for specific instructions.)

2. Via Paper Tape Reader:

- a. Load "Memory Load P. T. Reader" into Utility Track with normal load procedure.
- b. From Ready Mode, depress PKA 3 – Utility – and load Pass 1 of Assembler 1 through tape reader.
- c. Execute Assembler Pass 1. See below.
- d. Upon completion of all Pass 1 assemblies, depress PKA 3 from Ready Mode to load Assembler Pass 2. Pass 1 assembly does not destroy the Reader Load Routine in the Utility Track.
- e. Execute Assembler Pass 2. See below.
- f. Upon completion of all Pass 2 assemblies depress PKA 3 from Ready Mode to load in appropriate Main Memory Firmware through Tape Reader, prior to loading and executing any user programs.*

5.01.03 ASSEMBLER I

Start

Depress PKA 1, the "START" Key.

The program will stop at a Numeric Keyboard (NK) instruction with three PK's enabled. The three enabled PK's are:

PKA 2 KEYBOARD MODE OF OPERATION

PKA 3 CORRECTION MODE OF OPERATION

PKA 4 CONTINUATION MODE

OCK's The use of any OCK will allow 15 inches of leader tape (sprocket holes) to be punched and an automatic return to the initial keyboard (NK) instruction.

1. When "MEMORY" prints, enter the number of words of user memory. No entry assumes 512 words.
2. When "EXTMEM" prints, depress OCK 2, 3, 4.
3. When "PAGE 51" prints, depress OCK 1 for 51 lines/page or OCK 2, 3, 4, if 66 line/page are desired.

When the desired OCK's have been depressed, refer to Topic entitled "keyboard mode", 5.01.06 or condensed operating instructions Topics 5.01.07 and 5.01.10.

*Keyboard Modifiers for the Commercial Keyboard may be loaded immediately after loading Pass 1.

5.01.04 ASSEMBLER ISL

Start

Depress PKA 1, the "START" Key.

The program will stop at a Numeric Keyboard (NK) instruction with three PK's enabled.

The three enabled PK's are:

PKA 2 KEYBOARD MODE OF OPERATION

PKA 3 CORRECTION MODE OF OPERATION

PKA 4 CONTINUATION MODE

OCK's The use of any OCK will allow 15 inches of leader tape (sprocket holes) to be punched and an automatic return to the initial keyboard (NK) instruction.

1. When "MEMORY" prints, enter the number of words of user memory. No entry assumes 512 words.
2. When "PAGE 51" prints, depress OCK 1 for 51 lines/page or OCK 2, 3, 4, of 66 lines/page are desired.
3. When "SLT" prints, depress OCK 1 for the striped ledger table option (necessary when using 80-column card input) or OCK 2 if the striped ledger table option is not desired. If this option is chosen, the assembler forces a memory limit of 482 words.

When the desired OCK's have been depressed, refer to Topic entitled "keyboard mode" 5.01.06 or condensed operating instructions Topics 5.01.07 and 5.01.10.

5.01.05 ASSEMBLER VI AND VIII

Start

Depress PKA 1, the Start Key.

The program will stop at a numeric keyboard (NK) instruction with three PK's enabled. The three enabled PK's are:

PKA 2 KEYBOARD MODE OF OPERATION

PKA 3 CORRECTION MODE OF OPERATION

PKA 4 CONTINUATION MODE

OCK's The use of any OCK will allow 15 inches of leader tape (sprocket holes) to be punched and an automatic return to the keyboard (NK) instructions.

1. When "MEMORY" prints, enter the number of words of user memory.
2. When "EXTMEM" prints, depress OCK 2, 3, 4 for 32 Track; OCK 1 for 40 Track. If this option is used, the striped ledger option or the striped ledger table option will not be available.
3. When "PAGE 51" prints, depress OCK 1 for 51 lines/page or OCK 2, 3, 4 if 66 line/page are desired.
4. If the extended memory option, "EXTMEM", was not chosen above, "SL" prints. Depress OCK 1 for the striped ledger option or OCK 2 if the striped ledger option is not desired.
5. If the striped ledger option was chosen above (i.e., if OCK 1 was depressed), then "SLT" prints. Depress OCK 1 for the striped ledger table option (necessary when using 80-column card input) or OCK 2 if the striped ledger table option is not desired. If this option is chosen, the assembler forces a memory limit of 482 words.

When the desired OCK's have been depressed, refer to Topic entitled "keyboard mode", 5.01.06 or condensed operating instructions Topics 5.01.07 and 5.01.10

5.01.06 KEYBOARD MODE

Depress PKA 2 to enter the keyboard mode. The program will stop with the numeric keyboard enabled to allow the operator to enter the total words of memory intended for the object program.

Use any OCK to terminate the instruction. The program will then stop in the INSTRUCTION FIELD with the Alpha Keyboard enabled. Failure to enter the total words of memory will inhibit the operator from continuing at the INSTRUCTION FIELD position.

Instruction Field

Type the Mnemonic Op Code or Pseudo Op Code, listed on the symbolic program form.

The following choices are available to the operator:

OCK 2 – Program will return to and stop in the LABEL FIELD. This may be done before or after the entry of the Mnemonic.

OCK 4 – Program will stop in the parameter field, but a stop in the remarks field will be enforced before the entry of the instruction is completed.

OCK 1-3 – Program will stop in the parameter field without an enforced stop in the remarks field.

PKA 1 – Partial Phase I Halt. The use of PKA 1 will permit Phase I to be halted at any time. A special code is punched in the source tape instead of the pseudo operation code END. The label table is printed and punched in the source tape at each stop or "breakpoint." The label table punched at the conclusion of each segment of assembly is updated and all inclusive to that point. In addition, the last operation sequence number is printed and punched. There are two possibilities of continuing the assembly of the program. The first possibility is that the next section of the program will be assembled before the status of the assembler program in the machine has been disturbed. In this case, it is only necessary to enter the continuation mode and proceed. The second possibility is that the continuation will be at some later date when all current information in the system has certainly been destroyed.

In this case, it is necessary to load the Phase I assembler program and the label table along with the ending sequence number, both of which were printed and punched at the time of the "breakpoint," enter the continuation mode and proceed.

The partial Phase I halt, or breakpoint makes it possible to assemble large programs in sections.

The special code used for the breakpoint makes it possible to use small sections of Phase I source tape for input to Phase II. The breakpoint code will halt Phase II at a keyboard instruction, as described in Phase II operation, and the use of any OCK will permit Phase II to continue.

PKA 8 – Will print "ERROR" in red at left of sequence number column, the form will space and the program will stop in the Instruction Field for re-entry.

The entry of an Invalid Mnemonic will function in the same manner as the depression of PKA 8.

The typing of the Pseudo-Op word END will cause the program to enter a routine where the label table will be printed and punched out. The program will then allow the system to return to the READY MODE.

Label Field

Type in the Label – a maximum of 6 characters is permissible. The first character must be an Alpha character. A maximum of 139 labels may be used.

OCK's – Use of any OCK to terminate the field will cause the program to skip to the PARAMETER FIELD IF THE INSTRUCTION FIELD has been previously entered. If not, the program will stop in the INSTRUCTION FIELD.

Use of OCK 4 will also enforce a stop in the REMARKS FIELD before the line is completed.

PKA 8 – Will print "ERROR" in red at left of sequence number column, the form will space the program and the program will stop in the Instruction Field for line re-entry.

If a duplicate label is entered, the same function will occur as if PKA 8 had been depressed.

Parameter Field

The Parameter Field may actually be a 1, 2 or 3 field entry, depending on the Mnemonic entered in the Instruction Field. Either the Alpha Keyboard or the Numeric Keyboard will be enabled at this time, also depending on the Mnemonic.

OCKS's – Use of OCK 2 before the entry will change the entry mode from NUMERIC to ALPHA; however, the program will allow this switch only if the Mnemonic permits it.

ALPHA ENTRY – Enter the appropriate Alpha Characters. A maximum of 6 characters is permitted on labels.

NUMERIC ENTRY – Enter appropriate numeric digits. Where a zero entry is not permitted or where the numeric entry exceeds the value permitted by the Mnemonics, the program validation routine will re-initiate a numeric keyboard instruction until the operator indexes a valid entry.

OCK 1 – The program will stop in the next PARAMETER FIELD if the Mnemonic calls for another field entry. If the parameter field entry is the ending parameter field entry, the program will print the sequence number, the form will space and the program will stop in the INSTRUCTION FIELD for the next entry. If OCK 4 had been used in any previous entry position, after printing the sequence number, the program will stop in the REMARKS FIELD before ending the line entry.

OCK 2 – Depression of OCK 2 will activate the alternate keyboard when the instruction allows both absolute and symbolic parameter entries. For example, if the numeric keyboard is active, depression of OCK 2 will cause the alphanumeric keyboard to become active if the choice is available. In all other instances the key will function the same as OCK 1.

OCK 3 – The program will stop to permit entry of a +/- increment (numeric).

OCK 4 – On the ending Parameter Field entry, after printing the sequence number, the program will stop in REMARKS FIELD before ending the line entry.

PKA 8 – Use of this PK will print "ERROR" in red at the left of the sequence number column, the form will space and the program will stop in the INSTRUCTION FIELD for re-entry.

If an Invalid Alpha key has been used, the same function will occur as if PKA 8 had been depressed.

+/- Increment

The numeric keyboard is enabled. The numeric entry may be up to 255. Use of the RE key prior to termination, will permit a MINUS value; otherwise, the entry will be positive.

OCK 1, 2, 3 – These OCK's will print the sequence number, space the form and stop in the INSTRUCTION FIELD for the next line entry. If OCK 4 has been used previously, at any time in the symbolic entry, the sequence number will be printed and the program will stop in the REMARKS FIELD.

OCK 4 – Enforces a stop in the REMARKS FIELD.

PKA 8 – This PK will print "ERROR" in red at left of the sequence number column, the form will space and the program will stop in the INSTRUCTION FIELD for re-entry.

Remarks Field

Typing of up to 25 Alpha characters for remarks is permitted. The entry of a 26th character will result in a keyboard Error condition. The program will be halted. The RESET key must be used to correct the error condition and an OCK used to terminate REMARKS FIELD correctly. Remarks are not punched into the output tape until all typing is completed and the instruction terminated by the use of an OCK. The form will space and the program will stop in the INSTRUCTION FIELD for the next entry.

Alf Pseudo-Instruction

The entry of the pseudo-op ALF will permit the entry of up to 24 alpha characters as a constant. The entry of from 1 to 23 characters followed by OCK termination will cause the program to allocate the correct number of words for the message. The program will then stop in the INSTRUCTION FIELD for the next entry. The entry of exactly 24 characters and termination will cause the program to allocate the words and, in addition, the program will automatically print ALF in the INSTRUCTION FIELD on the next line and then stop for an additional alpha constant entry. The entry of a 25th character will result in a keyboard error condition. The RESET key must be used to correct the error condition and permit the proper termination by an OCK. However, the word ALF will still print in the INSTRUCTION FIELD with a stop to allow for an additional alpha-constant entry.

Continuation Mode

In addition to loading the Assembler program tape Phase I, using the standard program load procedures, the label table must also be loaded into memory. The tape perforator must be turned on and sufficient leader tape (sprocket holes only) punched. The RESET KEY will return the machine to the READY MODE.

PKA 4 – This will cause the sequence number printed out with the first line of entry and to be in proper sequence with the last sequence number from the previous section of tape, prior to the breakpoint. The program will then enter the KEYBOARD MODE portion of the Assembler program, Phase I for continuation.

Correction Mode

Depression of PKA 3 will cause the CORRECTION MODE of operation to be entered. Enter the appropriate selection for "MEMORY", "EXTMEM" and "PAGE 51" as previously discussed. The program will

then stop at a NUMERIC KEYBOARD instructions with three (3) PK's enabled. At this point, the source tape must be loaded in the A 581 Tape Reader and the A 562 Tape Perforator must be on. The enabled PK's determine the following functions:

| | |
|-------|--------|
| PKA 5 | ADD TO |
| PKA 6 | CHANGE |
| PKA 7 | DELETE |

Add to Sequence Number – PKA 5

At the numeric keyboard entry which is reached via PKA 3, prior to depressing PKA 5, index the sequence number of the symbolic entry, from Phase I documentation, that precedes the area in which instructions are to be added. "ADD TO" will print followed by the sequence number.

The program will automatically read the source tape, punch out a new tape, and will build a label table in memory. When the sequence number indexed has been read and punched, the program will print the sequence number plus .1 (XX.1). This provides the ability to add one symbolic instruction in the position following the sequence number indexed (XX) with a sequence number of XX.1. Following the entry of this added symbolic operation, the program will return to the Correction Mode. To successively add a group of instructions, PKA 5 must be depressed prior to each added instruction. Re-entry of the sequence number isn't required. The added instruction will be automatically inserted and numbered in .1 increment. The number of "ADD TO" instructions is not limited, but when .9 is exceeded, duplicate sequence numbering will result.

Example:

If instruction sequence number 23 is incremented by ten .1 increments, the result would be a duplicate sequence number 24.0. Further "ADD TO" instructions would cause this sequence number to in turn be incremented (24.1, 24.2, etc.). The original instruction listed as 24.0 will appear in the output tape immediately following the last instruction added.

Re-entry of the original instruction is not required, however, duplicate sequence numbering can lead to difficulty in later correction of the source program; therefore, the original 24.0 sequence number should be "deleted" then a new sequence number "added." See Delete From Source Tape paragraph.

Change Source Tape – PKA 6

At the numeric entry reached by depressing PKA 3, prior to the depression of PKA 6, index the sequence number from the Phase I documentation that is to be changed. When the instruction is terminated by PKA 6, "CHG" will print followed by the number indexed.

The program will automatically read the source tape, punch out a new tape and will build a label table in memory. When the sequence number indexed has been read, it will not be punched out. The program will enter the Keyboard Correction Mode to permit entry of a line of coding following the entry of the changed symbolic operation, the program will return to the Correction Mode described under correction mode.

The assembler program permits instructions to be changed and new instructions immediately added to the program.

Depression of PKA 6 without a sequence number being indexed, will cause the very next instruction on the source tape to be read in but not punched out.

Delete From Source Tape – PKA 7

At the numeric entry reached by the use of PKA 3, prior to the depression of PKA 7, index the sequence number, from the Phase I documentation that is to be deleted. When the instruction is terminated by the use of the PKA 7, "DEL" will print followed by number indexed. The program will automatically read the source tape, punch out a new tape and will build a label table in memory. When the sequence number indexed has been read, it will be ignored and not punched. Once deleted on first correction pass, that sequence number is gone and will never be found on any subsequent correction pass. The program will return to the Correction Mode described under correction mode.

Depression of PKA 7 without a sequence number being indexed, will delete the next sequence in the tape and print out the sequence number deleted.

Source Tape Interrupt Procedures

After indexing a sequence number and depressing any of the PK's described above, if the operator realizes that the wrong sequence number was entered, PKA 8 may be depressed. At the end of the line, the program will print out, "NOW AT" followed by the last sequence number read in and the program will stop at the Keyboard entry with the three (3) PK's enabled.

End Procedure

When the last "ADD TO," "CHANGE" and "DELETE" has been entered, the entry of any sequence number larger than the END sequence number, from Phase I documentation, will read the source tape and punch a new one. When the word END is read, it will cause termination of the Phase I correction routine and will then follow the END pseudo instruction procedure.

5.01.07 PHASE 1 – CONDENSED OPERATING INSTRUCTIONS AND REFERENCE LIST (SEE TOPICS 5.01.02 AND 5.02.04 FOR LOADING INSTRUCTIONS AND OPTIONS.)

1. Depress PKA 2 for Keyboard Mode.
2. Type total words of memory you intend to use.
3. Depress any OCK and it will space correctly and stop for you to type the Op code of the first instruction. (Col. 22-26 on coding forms.)
4. Type OP Code.
5. Use one of the following 3 lists of instructions:
 - a. No remarks (Col. 53)
No label (Col. 16)
 - (1) Depress OCK 1 or 3.
 - (2) If parameters are required, it will stop in each necessary parameter field. Type parameter. Depress OCK 1 or 3 after each parameter entry.
 - (3) The final OCK will space for next Op Code entry.

- b. No label (Col. 16)
Remarks present (Col. 53)
 - (1) Depress OCK 4.
 - (2) If parameters are required, it will stop in each necessary parameter field. Type parameter. Depress OCK 1, 2 or 4 after each parameter entry.
 - (3) Final OCK will space to remarks field. Type remarks.
Depress OCK 1 to space to next OP code entry.
- c. Label present (Col. 16)
 - (1) Depress OCK 2.
 - (2) If parameters are required, it will stop in each necessary parameter field. Type parameter. Depress OCK 2 after each parameter entry.
 - (3) Final OCK will space to label field. Type label.
 - (4) If no remarks, depress OCK 1 to space to next Op code entry.
 - (5) If remarks present, depress OCK 4. Type remarks. Depress OCK 1 to space to next Op code entry.

Typing Error

If mistake made, depress PKA 8 before an OCK. It will print ERROR and will stop in Op code field for re-entry of instruction.

Correction Mode

1. Turn on perforator and put Source Tape in reader.
2. Be in READY MODE.
3. Depress PK 1.
4. Depress PK 3.
5. Type memory size and OCK 1.

Add To

1. Type sequence number before area to be added.
2. Depress PKA 5.
3. Program will read source tape, punch new tape, build label tapes.
4. When indexed sequence number has been reached, .1 will be printed beside it.
5. Add new instruction as other original instruction.
6. If there are additional instructions, depress PKA 5 and type next instruction.

Changing Source Tape

1. Type number to be changed.
2. Depress PKA 6.
3. Sequence number will not be punched.
4. Enter new instructions as original instructions.

Delete from Source Tape

1. Type sequence number.
2. Depress PKA 7.

Sequence Number Typing Error

1. Depress PKA 8.
2. Depress PKA 5, 6 or 7 again.

To Terminate any Correction

1. Enter any sequence number larger than the END sequence number then depress PKA 5, 6 or 7.
2. It will read source tape and punch a new one. This will give new symbolic tape and new label table.

5.01.08 PHASE I – DIAGNOSTIC FACILITIES

Provision is made for the detection and printed indication of errors that may occur in the Assembler program.

Error Detection and Indication

The entry of the symbolic language in Phase I, as previously stated, is via the keyboard or paper tape reader. Errors are detected as the symbolic entry is made.

MNEMONIC ERROR DETECTION – The mnemonic is entered first in sequence. If the mnemonic is invalid, the OCK used to terminate the entry will initiate an error sequence that will space the print head to position 10, print ERROR, align the form 1 space, and re-position the print head so that the invalid mnemonic may be corrected.

LABEL ERROR DETECTION – The next sequential operation, in the entry sequence of the program, is the entry of a label if required. The label error detection consists of determining if the label has been used previously and/or if the total number of labels exceeds the maximum of 139.

Duplicate label validation occurs after the specific symbolic operation has been entered. If the label is invalid, the print head is positioned to 10, ERROR is printed, and the printer document is advanced 1 line and a corrected entry must be made.

If the maximum of 139 labels is exceeded, the printer document advances to the next form (over fold), prints and punches out the contents of the label table up to that point. The assembler program will return the system to the ready mode and the correction routine must be used.

PARAMETER ERROR DETECTION – The parameters are also validated following the termination of their entry. Label-type parameters are not validated in Phase I. Separate error detection is used for numeric parameters and alpha parameters.

NUMERIC PARAMETER ERROR DETECTION – Two numeric parameter conditions are validated in Phase I.

1. Parameter exceeds specified limits (maximum or minimum) value.
2. Parameter is required and has not been entered.

In both cases, the validation routine will, upon recognizing the invalid condition, re-initiate the parameter entry sequence. However, if a zero parameter is acceptable, it is not necessary to index a zero. Leaving the parameter area blank will force the zero entry.

ALPHA PARAMETER ERROR DETECTION – An alpha parameter error is detected upon termination of the entry. In an invalid situation, such as entering an invalid alpha character, or no entry where an alpha entry is required, the print head will be positioned to 10, ERROR is printed, the printer document aligned 1 space and the entire symbolic operation sequence must be re-entered.

MEMORY ERROR DETECTION – Memory error conditions are:

1. The program exceeds the maximum memory available.
2. A specific point in the program is reached where sufficient memory has been occupied such that specifying a REG instruction, with a large parameter, will exceed the maximum memory available.

In both of the above error conditions, the assembler program will return the system to the Ready Mode. This condition can only be corrected by use of the correction routine. (See correction mode.)

Error Detection – Correction Mode

The correction mode features error detection similar to that previously discussed.

MNEMONIC ERROR – If an invalid mnemonic is encountered, the validation routine will print the sequence number, label – if any – and the mnemonic as it appears in the input tape, advance the form 1 line, print INSTR ERROR, punch a NOP instruction in the output tape and continue assembly.

LABEL ERROR – If an invalid label is encountered, the validation routine prints the sequence number, label and instruction as it appears on the input tape, advances the form 1 line, prints LABEL ERROR, removes the invalid label, but punches the rest of the instruction in the output tape and continues assembly.

PARAMETER ERROR – If an invalid ALPHA parameter is encountered, (PK, LOD's, SK's or MASKS) the validation routine prints the sequence number, label, instruction and parameters, advances the form 1 line, prints PARMTR ERROR, punches the instruction in the output tape and continues assembly.

PHASE I – OUTPUT

The output of Phase I of Assembler I consists of a print-out and a punched paper tape (source tape).

The Phase I Print-Out is in the same format as the Phase I output tape. It is in two sections, the first is a listing of the Symbolic operations and the second a listing of the label table with memory addresses, as illustrated in Exhibit I.

The output tape is created in two sections. The first section is the symbolic operation codes, complete with parameters and remarks, with the addition of a sequence number and the decimal equivalent of the actual location of the instruction in memory, punched in USASCII.

The format is:

Sequence Number

Decimal equivalent of the actual memory location of the operation.

Label (if entry is labeled)

Symbolic Operation Code

Parameters

Remarks

The second section of output tape is separated from the first section by approximately ten inches of unpunched tape. This is the label table and is punched in compact format hexadecimal which is the same format punched for the object program.

The format is:

| | Label | Memory Location | | | |
|----|--------|-----------------|---------------|-----|-------------------------|
| 1 | | NOTE | | | ADDING MACHINE EXAMPLE |
| 2 | | BRU | START | | BYPASS PSEUDO INSTRCT |
| 3 | | ORG | 10 | | |
| 4 | TOTAL | NUM | 0 | | INITIALIZE AT ZERO |
| 5 | STAR | ALF | *** | | SYMBOL FOR TOTAL |
| 6 | START | LPNR | PRINT | | LOAD PRINT MASK AREA |
| 7 | | LPKR | KEYS | | LOAD PROGRAM KEYS |
| 8 | ZERO | CLM | TOTAL | | CLEAR MEMORY TOTAL |
| 9 | ENTER | AL | 1 | | ADVANCE FORM A LINE |
| 10 | | POS | 10 | | POSITION TO PRINTER |
| 11 | | PKA | 12 | | ENABLE PROGRAM KEYS |
| 12 | | NKRCM | 15 | 0 | INDEX OPERANDS |
| 13 | | PNS- | 14 | 0 | PRINT OPERAND |
| 14 | | SK | K | 2 2 | USE DCK2 IF ERROR |
| 15 | | ADM | TOTAL | | ADD TO TOTAL |
| 16 | | EX | K | 2 2 | PRINT # FOR ERROR AMT |
| 17 | | POS | 30 | | POSITION TO PRINT |
| 18 | | PC | # | | PRINT CHARACTER |
| 19 | | BRU | ENTER | | RETURN TO INDEX OPERAND |
| 20 | PRINT | MASK | ZZZZZZZZZZ.DD | | PRINT FORMAT |
| 21 | | WORD | | | |
| 22 | KEYS | BRU | PRTSUB | | PKA1 TO SUBTOTAL |
| 23 | | BRU | PRTTOT | | PKA2 TO TOTAL |
| 24 | PRTSUB | SRJ | CUMR | | |
| 25 | | PC | * | | TO SIGNIFY SUBTOTAL |
| 26 | | BRU | ENTER | | |
| 27 | PRTTOT | SRJ | CUMR | | |
| 28 | | PA | STAR | | TO SIGNIFY TOTAL |
| 29 | | BRU | ZERO | | GO TO CLEAR TOTAL |
| 30 | CUMR | TRA | TOTAL | | COMMON TOTAL ROUTINE |
| 31 | | PNS- | 14 | 0 | PRINT TOTAL |
| 32 | | POS | 30 | | |
| 33 | | SRR | 1 | | SUBROUTINE RETURN |
| 34 | | END | | | |

Exhibit 1
Basic Assembler I
Phase I

5.01.09 PHASE II

Phase II of Paper Tape Assemblers uses a symbolic (source) paper tape as input and provides an object program tape which operates on a Series L/TC computer. It also provides a program listing with the

object code as well as the symbolic code for each instruction and the absolute memory assignment. Phase II is also equipped with diagnostic facilities for the detection and indication of Phase II errors.

PHASE II – INPUT

The input to Phase II of the assembler program is the label-table tape, which has been separated from the symbolic tape and loaded via the program loader and the symbolic operation tape (source tape) which is mounted on the optional A 581 paper tape reader and entered under assembler program control.

PHASE II – OPERATING INSTRUCTIONS

The Phase II Assembler program must first be loaded into memory using the standard program load procedure. In addition, the same type of continuous pin feed forms must be used, with the forms positioned with left edge at position 5. If Phase II is being run immediately after running Phase I, it is not necessary to load the LABEL TABLE into memory. If Phase II is being run at any other time, it will be necessary to load the LABEL TABLE. This table is at the end of the source tape which is the output from Phase I. The table is separated from the source tape (sprocket holes only), is punched in compact hexadecimal format and must, therefore, be loaded via the program loader. When this procedure is complete, the RESET KEY must be depressed to return the machine to the READY MODE.

Start

PKA 1 – The program will stop at an Alpha Keyboard entry position to permit the typing of up to 6 alpha characters for any identification purposes desired.

At this point, the Symbolic output (source) tape from Phase I must be loaded on the A 581 tape reader and both the tape reader and tape punch switches turned on.

Depression of any OCK will cause the program to enter the automatic mode. This mode reads the source tape. Complete documentation is printed out, including the hexadecimal object program coding and all remarks. An object program will also be punched out.

The Assembler program Phase II will automatically put the machine into the READY MODE, when the Pseudo-Op END at the end of the source tape is read.

PKA 8 – This PK effects an interrupt. Operation can be resumed by the use of any OCK.

5.01.10 PHASE II – CONDENSED OPERATING INSTRUCTIONS AND INDEX

1. Read Phase II tape in PROGRAM LOADER or through paper tape reader.
2. Return to READY MODE.
3. At this point, if Phase I has not been run immediately prior to this, it will be necessary to load the label-table into memory using the same load procedure as in loading Phase II above. Then return to READY MODE.
4. Depress PKA 1.
5. Type up to 6 alpha characters for program identification and then OCK 1.
6. Insert symbolic tape into reader and depress READ switch.

7. This will give you a complete program listing and object tape.
8. If you have error in program it will be necessary to go to Phase I documentation for changes or to hexadecimally change program.

5.01.11 PHASE II – ERROR DETECTION AND INDICATION

As stated previously, the input to Phase II is the output tape, or source tape, from Phase I.

Error detection in Phase II is supplemental to the error detection in Phase I and is designed to validate the assignment of actual machine language, which is function of Phase II, plus the provision to indicate the possibility of machine-inflicted error, and the possibility of remote combination of programs vs. machine logic which could cause misinterpretation of the symbolic entries.

MNEMONIC ERROR DETECTION – The mnemonic is validated again in Phase II. At this particular point in assembler program progression, the potential error possibility is the misreading of the mnemonic as a result of a mispunch, some internal system failure or reader failure. In any event, an invalid mnemonic in Phase II is an irrecoverable error and will cause the validation sequence to return the system to the ready mode.

PARAMETER ERROR DETECTION – Parameters are again validated in the same manner as described under Phase I diagnostic facilities. In addition, if a label is used to define a parameter, the actual value of the label is validated.

For all parameter errors, except label errors, the word ERROR is printed starting in position 45, the printer escapes one space and PARMTR (parameter) is printed.

In the event of a parameter-label error, the word ERROR is printed starting in position 45, the printer escapes one space and LABEL IS printed.

For all instructions that are in error, a NOP instruction is substituted, printed out in the instruction sequence, and punched in the object program tape. The end result is that the object program tape contains NOP instructions instead of invalid instructions; plus, the program provides documentation which defines the location and type of parameter error.

Correction Routine Phase II

Phase II errors are corrected by using the Source tape (tape output from Phase I) and the Assembler Phase I correction routine, or correcting the Symbolic language and restarting the entire program. Corrections in the object program can be made by using the Memory Modify service routine with a corrected object tape generated with the Punch from Memory service routine. (See Section 3).

PHASE II – OUTPUT

The output from Phase II of Paper Tape Assemblers consists of a print-out and an object program tape.

The Phase II print-out is a complete print-out of the object program along with explanatory information. The format is illustrated by Exhibit 2.

The Assembler program Phase II output tape is the object program tape and is punched in compact hexadecimal format. It contains the complete program, in machine language, ready to be loaded directly into the Series L/TC computer.

| | | | | | | | |
|----|---|------|-----|-------|-----|--------|--------|
| 33 | 0 | C94D | 133 | | PA | LINE | 77 |
| | 1 | ED01 | 134 | | AL | 1 | |
| | 2 | 3949 | 135 | | TRA | TTL | 73 |
| | 3 | EB76 | 136 | | POS | 119 | |
| 34 | 0 | 0700 | 137 | | RR | | |
| | 1 | C94F | 138 | | PA | TOTAL | 79 |
| | 2 | EB7E | 139 | | POS | 127 | |
| | 3 | D496 | 140 | | PN | 8 | 6 |
| 35 | 0 | 904A | 141 | | ADM | DWNTTL | 74 |
| | 1 | D949 | 142 | | CLM | TTL | 73 |
| | 2 | F740 | 143 | LOOPT | PKB | 7 | |
| | 3 | F640 | 144 | | PKA | 7 | OVERFL |

| | | | | | | | |
|----|---|----------------------|-----|--------|------|---------------|--|
| 50 | 0 | 33366797 03333333 | 186 | | MASK | ZZD.DD | |
| 51 | 0 | 66E66797 03333333 | 187 | | MASK | ZZZ.ZZD.DD | |
| 52 | 0 | 33333679 03333333 | 188 | | MASK | ZD.D | |
| 53 | 0 | 66E66797 0333336E | 189 | | MASK | ZZ,ZZZ,ZZD.DD | |
| 54 | 0 | | 190 | ORDRNO | REG | 1 | |
| 55 | 0 | | 191 | WDSAME | ALF | SAME* | |
| 56 | 0 | | 192 | WDPP | ALF | PP* | |
| 57 | 0 | | 193 | WDCOD | ALF | COD* | |
| 58 | 0 | | 194 | WDFGHT | ALF | MTFGHT* | |

Exhibit 2
Basic Assembler I
Phase II

5.02.00 80-COLUMN CARD INPUT ASSEMBLERS

The Card Input assemblers are two pass assemblers with object card or object paper tape output. Unlike the paper tape assemblers, these assemblers consist of only one program and are loaded only once. GP 300 symbolic punched cards are used as input through the card reader. Pass I reads the symbolic deck, validates mnemonics, creates a label table, sequence checks the cards if requested, and prints certain errors. Pass II reads the symbolic deck again, produces a complete listing print-out including error messages and punches an object deck or object tape depending upon which assembler is used.

The card input assemblers which operate exclusively on L/TC equipment are Assembler II, Assembler IIA, Assembler IISL, Assembler IIASL, Assembler VII, Assembler VIIA, Assembler IX, and Assembler IXA.

The operating instructions for these assemblers are the same except for additional options available with some assemblers as cited below.

Assembler II

Assembler II operates on TC 500, L 2000, L 3000, and L 4000 with Card I/O and generates code for these machines and also the L 2000, L 3000, and L 4000 40-track machines.

Assembler IIA

Assembler IIA operates on TC 500/700, L 2000, L 3000, and L 4000, generates code for these machines and also the L 2000, L 3000, and L 4000 40-track machines, and is the same as Assembler II except that output is in object paper tape.

Assembler IISL

Assembler IISL operates on TC 500, L 2000, L 3000, and L 4000 with card I/O and generates code exclusively for the L 5000. Note striped ledger options in topic 5.02.04.

Assembler IIASL

Assembler IIASL operates on TC 500/700, L 2000, L 3000, and L 4000, with object paper tape output, and generates code exclusively for the L 5000. Note striped ledger options in topic 5.02.04.

Assembler VII

Assembler VII operates on the L 2000, L 3000, and L 4000 40-track machines with Card I/O, and generates code for these machines and also the L 5000 and 32-track machines. Note striped ledger options in topic 5.02.04.

Assembler VIIA

Assembler VIIA operates on the L 2000, L 3000, and L 4000 40-track machines, with object paper tape output and, generates code for these machines and also the L 5000 and 32-track machines. Note striped ledger options in topic 5.02.04.

Assembler IX

Assembler IX operates exclusively on the L 5000 with Card I/O and generates code for the TC 500, L 2000, L 3000, and L 4000 32-track machines for the L 2000, L 3000 and L 4000 40-track machines, and for the L 5000. Note striped ledger options in topic 5.02.04.

Assembler IXA

Assembler IXA operates exclusively on the L 5000 with object paper tape output, and generates code for the TC 500/700, L 2000, L 3000 and L 4000 32-track machines, for the L 2000, L 3000 and L 4000 40-track machines, and for the L 5000. Note striped ledger options in topic 5.02.04.

5.02.01 ENVIRONMENT

Equipment required for the assemblers mentioned above is as follows:

5.02.02 CARD I/O ASSEMBLERS

Assemblers II and IISL are compatible with any of the following 32-tracks full memory Series L, TC's: TC 525, TC 1545, TC 2545, L 2101-608, L 2301-608, L 3111-608 or L 4311-608.

Assembler VII is compatible with any of the extended memory L's: L 2302-908, L 3312-908, L 4312-909.

Assembler IX is compatible with the following full memory Series L: L 5112-609.

These assemblers require the following peripherals and firmware:

A 595 Card Reader

A 149 Keypunch

| Assembler | Hardware | Firmware Required* |
|-----------------------|------------------------|--------------------|
| Assembler II and IISL | L 2301-608, L 2101-608 | 2-1004-001-XX |
| Assembler II and IISL | TC 525, TC 1545 | 2-1024-001-XX |
| Assembler II and IISL | L 3311-608, L 3111-608 | 2-1004-001-XX |
| Assembler II and IISL | L 4311-608 | 2-1304-001-XX |
| Assembler II and IISL | TC 2545 | 2-1324-001-XX |
| Assembler VII | L 2302-908, L 3312-908 | 2-1104-001-XX |
| Assembler VII | L 4312-909 | 2-1404-001-XX |
| Assembler IX | L 5112-609 | 2-1204-001-XX |

5.02.03 CARD IN/PAPER TAPE OUT ASSEMBLERS

Assembler IIA and IIASL are compatible with any of the full memory Series L/TC's: TC 525, TC 725, TC 1545, TC 2545, L 2101-608, L 2301-608, L 3111-608, and L 3311-608.

Assembler VIIA is compatible with any one of the following full memory Series L's: L 2302-908, L 3312-908, and L 4312-909.

* XX denotes latest revision

Assembler IXA is compatible with the following full memory Series L: L 5112-609.

These assemblers require the following peripherals and firmware:

A 595 Card Reader

A 562 Paper Tape Punch

| Assembler | Hardware | Firmware Required* |
|----------------|------------------------------------|----------------------------------|
| Assembler IIA | TC 525, TC 725, TC 1545, TC 2545 | 2-1005-002-XX or 2-1024-001-XX** |
| Assembler IIA | L 2101-608, L 2301-608, L 3111-608 | 2-1005-002-XX or 2-1004-001-XX** |
| Assembler IIA | L 3311-608 | 2-1005-002-XX or 2-1004-001-XX** |
| Assembler VIIA | L 2302-908, L 3312-908, L 4312-909 | 2-1104-001-XX** |
| Assembler IXA | L 5112-609 | 2-1204-001-XX** |

Applicable Program I. D. Numbers are as follows:

| Assembler | Program I.D. |
|-----------------|---------------|
| Assembler II | 1-1001-011-XX |
| Assembler IIA | 1-1001-066-XX |
| Assembler IISL | 1-1001-071-XX |
| Assembler IIASL | 1-1001-072-XX |
| Assembler VII | 1-1101-022-XX |
| Assembler VIIA | 1-1101-015-XX |
| Assembler IX | 1-1201-024-XX |
| Assembler IXA | 1-1201-025-XX |

5.02.04 INPUT

Card Input Assemblers use GP 300 symbolic cards as input, as defined in Section 1, and the desired control cards are described below.

** XX denotes latest revision

* Although these firmware sets are used with the card I/O Assemblers, they will also work with Card In/Paper Tape Out Assemblers. The assemblers themselves include the necessary firmware required to provide paper tape output.

CONTROL CARDS – The control cards specify the input and output of the Assembler II program. Each control card must contain a \$ in card column one and the name of that control card starting in card column 16.

The control cards available for the various assemblers are as follows:

- Assemblers II and IIA – \$DATA, \$DATAB, \$SEQ-CK, \$MEMORY, \$PAGE51, \$LABELS, \$EXTMEM
- Assemblers IISL and IIASL – \$DATA, \$DATAB, \$SEQ-CK, \$MEMORY, \$PAGE51, \$LABELS, \$SL-T
- Assemblers VII, VIIA, IX, IXA – \$DATA, \$DATAB, \$SEQ-CK, \$MEMORY, \$PAGE51, \$LABELS, \$SL, \$SL-T, \$EXTMEM

\$DATA – This control card tells the system that the symbolic card deck is punched in EBCDIC code.

\$DATAB – This control card tells the system that the symbolic card deck is punched in BCL code. BCL code does not include some special characters available with EBCDIC.

\$SEQ-CK – This control card enables the checking of sequence numbers in the symbolic card deck. The sequence number of the current card is compared to the sequence number of the preceding card and if the value of the current card is less than or equal to the previous card, the error message SERR is printed and the program continues.

\$MEMORY – Firmware sets vary in memory requirements, which controls the amount of user memory available. If the size of object memory is other than 512 words, the Assembler program may be informed by the control option \$MEMORY followed by the memory size, stated as a 3-digit integer value, starting in card column 29.

\$PAGE51 – This control card tells the system that the Assembler listing is being printed on a form 8½” x 11”. If the above option is not used, the system assumes a form size of 11” x 14”.

\$LABELS – This control card generates a list of labels along with the word and syllable location that each label represents. A maximum of 139 labels are allowed.

\$SL – This control card tells the system that the striped ledger option is desired.

\$SL-T – This control card tells the system that a striped ledger conversion table, contained in words 482-489 are necessary when using 80-column card input, is desired. If this control card is used, the assembler forces a memory limit of 482 words.

NOTE: If the extended Memory option is used, these striped ledger options will not be available.

\$EXTMEM – This control card is required when generating code for a 40-track machine.

5.02.05 OPERATING INSTRUCTIONS

Readying the System

1. Unprotect (jumper) all tracks of main memory – words 0 to 1023 (do not jumper the utility track – Block 2 Track 2 – or any tracks that have been permanently jumpered by a Field Engineer.)
2. From the Ready Mode, load the Utility routine “Memory Load, Card Reader”.
3. Turn on Card Reader and depress Feed key. Place Assembler object program deck in Card Reader and depress PKA 3 from the Ready Mode. After all cards have been loaded, depress any OCK or keyboard character to print the hash total, the system then returns to Ready Mode.
4. Preparation of Card Punch:
 - a. Depress the POWER ON SWITCH and place blank cards in the feed hopper.
 - b. Place a blank card around Program Drum No. 1. Around Program Drum No. 2 (left drum) place a card containing alternate “12” and “11” punches in card columns 1 through 80. These punches may start with either “12” or “11” so long as they alternate through all 80 columns.
 - c. Place drums on the Card Punch and place brush assemblies in contact with drums, the Program Switch on the Card Punch keyboard may be set in any position.

- d. If it is desired to interpret the punch cards, the Print button should be depressed. Miscellaneous characters will be printed if the Card Punch is interpreting object program being punched.
- e. Depress the Auto Feed button to place a card in the punch station, depress the ERR. REL. key to position a blank card in the read station.
- f. Depress the Punch-On-Line button, the card punch is now under control of the Series L/TC. Halts on punch instruction if not on-line.

Pass 1

Place control cards and symbolic deck in the Card Reader, depress PKA 1 from Ready Mode.

When it is necessary to temporarily halt assembly to add or remove cards in the Card Reader, three choices are available.

1. Depress Feed switch to OFF. To restart assembly, depress Feed switch to ON and depress Restart switch.
2. Depress RESET key, the program halts and the Numeric keyboard light is lit. Depression of any OCK will resume the assembly of the program.
3. The system halts on a Reader condition if there are no cards in the Reader and the last card read was not the END card. Placing cards in the Reader and depressing the Restart switch will cause the assembly to continue.

Pass I is completed when the symbolic END card is read. The system will halt on an NK instruction following the print-out of the label table, if a \$ LABELS card was used.

Pass I Errors

During Pass I the only thing printed are the symbolic cards containing errors (card columns 11 through 77) preceded by a description of the error.

The following messages are used on the print-out to indicate detection of errors. These messages are printed in red.

Sequence Error

SERR – If the current card being checked at the request of the control card \$ SEQ-CK has a lower or equal sequence number than the previous card, then that card is out of sequence.

Previously Used Label

LERR – The label in card columns 16 through 21 has been previously used. That instruction is not assigned a label, in Pass I or Pass II. The symbolic deck must be run again through Pass I, with a different label if the instruction is to have one.

Label Limit Exceeded

ELERR – A maximum of 139 labels are allowed, those labels beyond 139 are not put in the label table. The instruction is assembled without a label assigned to it.

Invalid Mnemonic

IERR – The mnemonic in the op code field is invalid. This card or a corrected card must be used in Pass II. It is assigned a syllable in memory.

Word Length Exceeded

No message is used when either the NUM or MASK instruction is too long to be translated into one word. The overflow is ignored and processing continues.

Memory Assignment Error

MERR – This tells the user that the specified memory has been previously assigned. This could occur for example if an ORG instruction is used and specifies a word number already assigned by the Assembler.

Assumed Memory

No message is used. Memory of 512 is assumed by the Assembler unless a \$ MESSAGE specifies memory other than 512.

Memory Capacity Exceeded

MERR – This tells the user that the symbolic program is too large for the specified object memory size. The Assembler does continue to process.

End Card

No message is printed and the Assembler halts on a reader condition. The symbolic deck must have as its last card the mnemonic END card. To correct place END card in Reader and depress Restart key on Reader. Pass I cannot be completed without the END card.

Pass II

Place the control cards and all or part of the symbolic deck in the Card Reader and depress any OCK.

A complete listing print-out including error messages and an object deck is produced.

After Pass II is completed, it is necessary to reload main memory firmware before executing any assembled object program.

Pass II Errors

The description prints next to the field in error.

Undefined A Parameter Label

PERR – The label in the A parameter was never used as a label to a mnemonic instruction. A NOP is generated and Pass II continues.

Invalid Parameter Range

PERR – The value in the parameter is not within the limits allowed by the mnemonic. A NOP is generated and Pass II continues.

Invalid Increment Field

PERR – The value in the increment field is not within 255 or an invalid character is located within the field. A NOP is generated and Pass II is continued.

No Label or Increment Error

PERR – There is no label or the increment field is in error. A NOP is generated and Pass II is continued.

Invalid Mnemonic

IERR – The mnemonic is not in the language specifications. The mnemonic is ignored and a NOP is generated. Pass II is continued.

Word Length Exceeded

No message is used when either the NUM or MASK instruction is too long to be translated into one word. The overflow is ignored and processing continues.

Memory Capacity Exceeded

MERR – The symbolic program is too large for the specified object memory size. Pass II processing continues.

End Card

No message is printed and the Assembler halts on a Reader condition. Place END card in Card Reader and depress Restart switch. The symbolic deck must have as its last card the mnemonic END card. Pass II cannot be completed without the END card.

After the reading of all cards in Pass II, the last object program card is released. This completes Assembler II. The user may now reload main memory firmware, protect memory and run the object program just produced.

5.03.00 L/TC ASSEMBLER III B 3500 VERSION

The Assembler III Program operates on a B 3500 system and prepares an object program for Series L/TC systems. It accepts symbolic input directly from cards or will accept a symbolic punched paper tape if it is loaded on disk via a utility program prior to the execution of the assembler and the appropriate control card is included in the source media.

All references in this document to Assembler I or Basic Assembler I refer to the Series L/TC keyboard version of the Series L/TC Assembler.

5.03.01 ENVIRONMENT

The following system hardware is required for Assembler III:

- B 3500 – 60 KB Bytes Core
- 1 Module Disk (800 segments, 100 bytes each)
- 1 Tape Unit (7 or 9 channel)
- Card Reader
- Paper Tape Punch
- Paper Tape Reader (Optional for Symbolic Paper Tape Input)
- Card Punch (Optional for Symbolic or Object Card Output)
- Line Printer

5.03.02 LIBRARY TAPE INPUT

The input for Assembler III is the Group II Software Library Tape and the source media which is either a card deck that included both the symbolic program and the appropriate control cards, or the symbolic paper tape output from Phase I of the Basic Assembler I Program for Series L/TC systems.

Library Tape

The tapes contain the following programs:

- | | |
|--------|---|
| ASSEMB | This is the assembler program. |
| QCONV | This produces the object program on paper tape. |
| LCNVRS | This converts input symbolic paper tape code to card image and stores it on disk. This must be executed first if symbolic tape input is specified. It creates a disk file which is used as the source file for input to the ASSEMB program. |
| CRDCVR | This produces a symbolic paper tape for input to Assembler I. |
| XRFBTC | This is the cross-reference program. It produces a cross-reference listing of labels. |
| OBJCRD | This produces the object punched card deck. |
| LIBTAP | This is the utility routine used to create and update library tapes used by the \$ library input option of Assembler III. |

The following files are used internally in the Assembler III Program:

- | | |
|-------|--------------------------|
| COND | Used in error detection. |
| OPTBL | The operation code file. |

5.03.03 MCP CONTROL CARDS

The following MCP Control Cards are used in the assembly process; the 1-2-3 indicates a multi-punch in card column 1 for the specified cards:

Execute Card

The execute card initiates program execution. It must be punched in the following format:

```
1
2 EXECUTE ASSEMB
3
```

Data Card

The data card specifies the type source media. It must be punched in the following format:

```
1
2 DATA SOURCE          or          1
3                          2 DATAB SOURCE
3
```

This tells the system that the source media is cards, punched in either EBCDIC or BCL code, respectively. BCL does not include some special characters available with EBCDIC.

Dollar Sign Card

The dollar sign card specifies an option which controls the input and output during the assembly process. It must be punched in the following format:

```
$ (LIST OPTION HERE)
```

The options available on Assembler III are discussed below.

Data Deck

The symbolic deck to be assembled.

End Card

The end card must follow any card deck. It is punched in the following format:

```
1
2 END
3
```

It tells the system that the input from the Card Reader is complete. This card is used in addition to the GP 300 END card.

5.03.04 OPTION CONTROL CARDS

There are several options which control the input and output of the Assembler III Program. The options,

specified in the succeeding sections, must be preceded by a (\$) dollar sign and may either be coded on individual cards, or may be coded free-form serially on one or more cards.

SEQ-CK, Sequence Checking

The format on the control card is SEQ-CK. When a sequence number has been punched into column 11 through 15 of the source cards, sequence checking may be desired. If this control option is used, the sequence number of the current card is compared to the sequence number of the preceding card and, if the value is less or equal to, the error comment "SEQUENCE ERROR" will print. Execution of the program continues. Code generation is not ensured when sequence errors occur.

RESEQ, Resequencing

The format of the control card is RESEQ. If this control option is used, any sequence numbers punched in columns 11 through 15 of the source cards are ignored. Resequencing is initialized at 10 and incremented by 10 for each succeeding card.

MEMORY, Memory Size

Since different firmware sets vary in memory requirements, the size of user memory also varies. If the size of user memory is other than 512 words, the Assembler III Program may be informed by the control option MEMORY followed by the memory size, stated as an integer value. If this option is omitted, memory of 512 words is assumed and appropriate warning message is printed.

SYM-CD, Symbolic Card Output

If output, in symbolic BCL code, is required on punched cards, the symbolic-card control option SYM-CD is used. This will provide a symbolic card source deck, resequenced if the RESEQ option is used. If a program identification is punched in card columns 5-10 of the first symbolic card, it will be punched in every card of the output symbolic deck.

SYM-CN, Symbolic Card Output

This option is the same as SYM-CD with the exception that the output symbolic deck is punched in EBCDIC code.

SYM-PT, Symbolic Paper Tape Output

If symbolic paper tape output is required, the SYM-PT control option is used. This provides an input symbolic paper tape for Phase I of the Basic Assembler I Program.

LABELS, Printed Table of Labels

The entry of the control option LABELS provides a print-out of Assembler III generated table of labels. The list of labels is printed in the order in which they were programmatically defined along with the location that each label represents.

DOC, Documentation Punch and Print

If the pseudo-op DOC is used in a program and it is necessary to reproduce it in a symbolic card, the

control option DOC is used. \$ DOC will retain the psuedo-op and print and punch it in its proper programmatic sequence.

PUNCH, Paper Tape Object Code

If the output of the Assembler III Program must be a Series L/TC object program punched in paper tape, the control option PUNCH is used. (See Section 6, Page 6-6 for a discussion of the object tape format.)

OBJ-CD, Object Card Output

If object card output in hexadecimal object code is required on punched cards, the control option OBJ-CD is used. This will provide an object card deck for input to a Series L/TC card system. (See Section 6, Page 6-2 for a discussion of the object card format.)

DISK-IN, Paper Tape Input

If the symbolic paper tape output from Phase I of the Basic Assembler I Program is to be used as input to the Assembler III Program, it is required that the conversion program, LCNVRS be executed prior to the execution of the Assembler III Program (discussed on Page 5-27. It also requires the control option DISK-IN which must be entered as the last control option in the control option deck. The DISK-IN card must be followed by an end card: ? END.

The Assembler III Program will use the information transferred from the symbolic paper tape to the disk by the conversion program, as source media.

XREF, Cross Reference of Labels

If a cross reference listing of labels is desired, control option XREF is used. The labels are printed in alphabetical order with their sequence number and the sequence numbers of those statements referencing that label.

LINES, Specifying Number of Lines Per Page

This option will permit the programmer to specify the number of lines desired on each page below the heading. It is possible to use any two-digit integer for the number of lines per page. If this option is not used, the maximum of 60 is assumed, which provides 7½ inches of text at 8 lines per inch.

SAVE XXXXXX

This will cause the Assembler to retain the object program upon the disk. Punching of paper tape or 80-column card does not occur. XXXXXX represents a 6 alpha character disk file-name. It must be used in conjunction with \$ PUNCH or \$ OBJECD.

\$ S. L. T.

This Control option card must be used when assembling for an 80-column card striped ledger system. A character translation table is produced in the object program by the assembler. This option also forces a memory size limit of 482 which supersedes any memory option card.

\$ S. L.

This control card tells the system that the striped ledger option is desired.

\$EXTMEM

This control card is required when generating code for a 40-track machine.

\$ LIBRARY

The \$ LIBRARY option provides the capability of assembling from magnetic tape. A program is called in from a library tape labeled LIBL, updated on disk and then assembled. The program on the tape LIBL is not updated.

LIBL is created by the LIBTAP Utility Routine. (See 5.03.08)

Cards can be (1) deleted, (2) added, (3) changed during assembling:

1. Deleting cards.

| | |
|----------|-----------------------|
| CC 1 | D |
| CC 11-15 | Sequence number |
| CC 16-80 | Blank (must be blank) |

2. Adding cards.

| | |
|----------|-------------------------------|
| CC 1 | A |
| CC 11-15 | Sequence number (unique) |
| CC 16-80 | Format of regular source card |

3. Changing cards.

| | |
|----------|---|
| CC 1 | C |
| CC 11-15 | Sequence number - must match a sequence number of a card in the program |
| CC 16-80 | |

The patch cards, contained between the LIB instruction and the pseudo END card, must be in sequential order. These patch cards change the program during assembling but do not alter the tape "LIBL".

The control deck necessary to assemble from magnetic tape is as follows:

? EXECUTE ASSEMB

? DATA SOURCE (or ? DATAB SOURCE if BCL is used)

\$ OPTIONS

*\$ LIBRARY

LIB FILE NAME

PATCH CARDS IF ANY – these must be in sequential order

END (CC 22-24)

? END

*Last \$ OPTION card

5.03.05 OPERATING INSTRUCTIONS

Operation of the Assembler III Program involves the setting up of peripherals as a major function as opposed to Basic Assembler I Program operation which requires manual intervention as a major function.

Equipment Setup

To exercise all the control options of the program, 1 magnetic tape unit, 1 card reader, 1 card punch, 1 paper tape reader, 1 paper tape punch, and a line printer are required.

1. Magnetic Tape Units

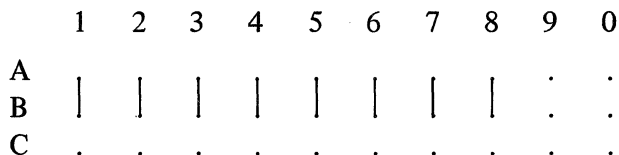
Mount the Library Tape. Choose an MTU with the proper channel (7 or 9) and set the appropriate density. Both of these are marked on the library tape reel label. Load the tape reel.

2. Card Punch (If symbolic or object card-out is required)

Load the hopper on the card punch with sufficient cards and ready the punch.

3. Paper Tape Reader (If symbolic paper tape-in is desired)

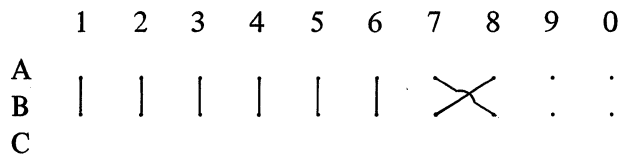
Wire the channel select board, inside the front doors on the upper left of the reader in the following manner.



Turn the three control code switches off. Depress the Parity on-off switch to off. Depress the High-low switch to low. Depress the Strip-Reel switch to strip or reel, depending on input type.

4. Paper Tape Punch

Wire the channel select board, inside the front doors on the upper left of the punch, in the following manner:



Turn the Control Code Switch off. Set the LEVEL Designator Switch to 8-LEVEL. Ready the punch.

5. Line Printer

Equip the printer with an 8 lines per inch control tape and ready the printer.

Operation

Loading the Library Tape: The loading of the magnetic library tape may be initiated by either a control card or a control input message via the supervisory printer (SPO). If a control card is used, the card would read

```
1
2 LOAD FROM GRPII < list of programs and files to be loaded >
3
```

The control card would load the programs specified from the list of available programs and files. For example:

```
1
2 LOAD FROM GRPII ASSEMB QCONV CRDCVR OPTBL COND OBJCD XREF
3
```

In either case the card must be passed through the card reader by itself. When the tape has been loaded, an output message to this effect will be typed on the SPO, after which the Assembler may be executed.

If the library tape load is initiated via the SPO, the input message

```
CC LOAD FROM GRPII < list of programs and files to be loaded >
```

would load only the specified programs and files.

Assembling with Card Input: If symbolic punch card input is used, the following cards must be read in through the card reader in the order specified:

```
1
1. 2 EXECUTE ASSEMB
3

1          1
2. 2 DATA SOURCE (or 2 DATAB SOURCE if BCL is used)
3          3

3. $ Option Cards specifying any or all of the options defined.

4. The symbolic source deck.

1
5. 2 END
3
```

Assembling with Symbolic Paper Tape: If symbolic paper tape input is used, the paper tape must first be loaded via the paper tape reader, converted to card format, and placed on disk using the LCNVRS program (which must have previously been loaded). After mounting the symbolic tape on the paper tape reader, the following card must be read in through the card reader in the order specified:

```
1
2 EXECUTE LCNVRS
3
```

When the symbolic paper tape has been loaded, the procedure discussed above for punch card input would be followed with the exception that the DISK-IN option must be the last option specified and must be followed immediately by the END card (since no symbolic deck is used).

5.03.06 ERROR DETECTION

The Assembler III Program in processing the input, or source, data makes several passes through the information. These passes may be divided into 2 major categories:

1. Pass I – the processing of data prior to the assignment of object code to the symbolic program.
2. Pass II – the processing of the data, the major function of which is the assignment of object code.

Both passes incorporate editing functions and the resultant error detection.

Pass I Error Detection

Pass I errors are printed out prior to the program listing. Each error comment will be followed by the print-out of the symbolic operation which is determined to be in error.

LABEL ERRORS – Several types of label errors are detected in Pass I:

1. Duplicate Label

If a label has been previously used, the error comment printed starts with the label followed by:

... HAS ALREADY BEEN ENTERED AS A SYMBOLIC IDENTIFIER

the duplicate label is not entered, processing continues.

2. Invalid Label

Labels for the Assembler III Program must begin with an alphabetic character. If the first character of the label is not an alphabetic character, the error comment is:

LABEL MUST BEGIN WITH AN ALPHABETIC CHARACTER

Blanks are not allowed in a label. If the label contains a blank, the error message is:

LABEL MUST NOT CONTAIN BLANK CHARACTER

The label is not entered, processing continues.

3. Label Limit Exceeded

The Assembler I Program allows a limit of 139 labels. This limit is not significant to the Assembler III Program, but the possibility exists that a symbolic paper tape, generated by the control option \$ SYM-PT could be used as input to the Assembler I Program for Series L.

The error comment is:

NUMBER OF LABELS EXCEEDS LABEL LIMIT WHEN USING ASSEMBLER I

The error is ignored and processing continues.

4. Card Field Definition Label Error

Labeling of the CDF table is allowed only on the first syllable of a word. If the label is not on the first syllable of a word, the error message is:

LABEL NOT AT START OF WORD

A NOP is inserted and processing continues.

SEQUENCE ERROR – As described in Option Control Cards, if \$ SEQ-CK is used, and the sequence

number in columns 11 through 15 of the current card is less than the sequence number on the previous card, the error comment is:

SEQUENCE ERROR

This is followed by a print-out of the information on the current card and processing continues.

EXCEEDS MEMORY CAPACITY – If the symbolic program is too large for the specified object memory size, or if an erroneous ORG instruction has been entered, the error comment is:

STORAGE EXCEEDED BY INSTRUCTION

followed by a print-out of the instruction that exceeded memory. Processing continues.

REGION ERROR – If a REG instruction is entered with either a 0 parameter or a parameter exceeding 255, the error message is:

REGION MUST HAVE SIZE 1-255

A memory location is not reserved and processing continues.

BACKWARD ORGANIZATION ERROR – If an ORG instruction is entered that attempts to assign memory that has been previously assigned, or at a memory location sequence number that is lower than the current instruction address, the error message is:

BACKWARD ORG NOT ALLOWED ON ASSEMBLER I

Memory is assigned and assembly continues.

CONTINUATION CARD ERROR – If a card with a numeric field length follows either an ALF or MASK instruction with a field length greater than 24, a continuation card was expected. The error message consists of a print-out of the current card followed by:

PREVIOUS CARD HAS INVALID FIELD LENGTH

In an ALF instruction a missing field length will result in the error message:

EMPTY FIELD-LENGTH FIELD

In either case the instruction assigns 1 word which is not filled and processing continues.

INVALID MASK ENTRY – If an invalid mask character has been entered, the character will be printed followed by:

. . . . IS NOT A VALID MASK ENTRY

the character is ignored and processing continues.

MASK LENGTH ERROR – If an erroneous mask length has been specified, the mask will be printed followed by:

MASK LENGTH RECALCULATED TO BE . . .

and the correct length. The mask is corrected and processing continues.

INVALID OPERATION CODE – If an invalid op code is entered, the invalid code will be printed followed by:

. . . IS AN INVALID INSTRUCTION OP CODE

A NOP (no operation) instruction is generated and processing continues.

PARAMETER ERRORS – The parameter errors detected in Phase I are defined in the following sections.

1. No Label

Some of the instructions specified in Series L Assembler Language, required a label in the parameter field. If a label has not been entered, the error message is:

MISSING SYMBOLIC LABEL

The instruction is ignored, a NOP is assembled and processing continues.

2. Increment Exceeds Limit

If the increment exceeds the limit of 255, the error message:

INCREMENT GREATER THAN 255 NOT ALLOWED ON ASSEMB I

is printed.

This does not affect Assembler III or IV, processing continues.

3. Invalid Numeric Entry

If a non-numeric character is entered in the numeric portion of a NUM instruction, the error message is:

INVALID NUM ENTRY

A word of zeros is assembled and processing continues.

4. Parameter Length Exceeded

If the parameter length of either a MASK or NUM instruction is too long to be translated into 1 word, the error message:

PARAMETER TOO LONG

is printed, a word of zeros is assembled and processing continues.

5. Invalid Define Parameters

If the entry in the parameter field of a DEF is invalid, the error message is:

A-ENTRY MUST BE NUMERIC AND LESS THAN 767

If either one or both entries in the parameter fields of a DEFT are invalid, the error message is:

DEFT PARAMETERS MUST BE NUMERIC AND 0-15

In either case memory is not assigned and processing continues.

6. Code Parameter Error

If entries in the CODE instruction parameter field are not 0 through 9 and A through F, the error message is:

ILLEGAL DIGIT ENTERED IN CODE INSTRUCTION

A NOP is inserted and processing continues.

7. Card Field Definition Error

If the total length of the fields defined in the CDF instruction exceeds the maximum number

of characters allowed on a card, the error message is:

SUM OF CDF PARAMETERS MUST NOT BE > 80

A NOP is inserted and processing continues.

CARD BUFFER DECLARATION ERROR – A CDB instruction must be the first card in the symbolic source deck. If it is not the first card, the error message is:

CDB MUST BE FIRST INSTRUCTION IN DECK

The instruction is ignored and processing continues.

Pass II Error Detection

At the beginning of the second pass, the printer will skip to the starting position on the next page.

LABEL ERRORS – Label error detection includes the re-evaluation of the label limit and validation of labels in the parameter field.

1. Label Limit Exceeded

This is identical to Pass I and uses the same error comments.

2. Label Parameter Error

If the label entered as an A parameter has not been recorded as a label identifier, the error comment prints the label followed by:

. . . HAS NOT BEEN ENTERED AS A LABEL

A NOP is generated and processing continues.

3. Label Increment Error

If a label is valid but the increment is invalid, the error message is:

ILLEGAL ENTRY IN INCREMENT FIELD

The increment is ignored and processing continues.

INVALID OPERATION – The validity of the operation code is rechecked in Pass II and if invalid, the entry is printed followed by:

. . . NOT A VALID OP-CODE

A NOP is generated and processing continues.

PARAMETER ERRORS – The parameters are edited for validity and content in Pass II.

1. Label Parameter Error

See above.

2. No Parameter Entry

If a required parameter has not been entered, the error comment will be:

EMPTY

followed by the specific field in which the error occurred, A PARAMETER, B PARAMETER or C PARAMETER. The program assumes a value of zero and processing continues.

3. Illegal Parameter Entry

For a parameter entry in a field that should be empty the error comment is:

ILLEGAL PARAMETER ENTRY IN . . .

followed by; A PARAMETER, B PARAMETER or C PARAMETER, depending upon which parameter contained the error. Processing continues.

4. Invalid Parameter Entry

For an entry that is invalid in either size or type, a listing of the valid parameter entries for that specific instruction is printed including an indication of which parameter(s) is/are in error.

Example:

| | | | |
|------|---|----|---|
| OP | A | B | C |
| ADIR | I | 32 | |

In the above instruction a keypunch error has been made in the A parameter field.

The error message is:

| | | |
|-------------|---|-------------------------------|
| A PARAMETER | — | MUST HAVE NUMERIC VALUE 1-4 |
| B PARAMETER | — | MUST HAVE NUMERIC VALUE 0-255 |

If a symbolic entry is allowed the error message is:

SYMBOLIC ENTRY ALLOWED

5. Flag Instruction Parameter Error

If the flag designated in the parameter is not valid for the flag group specified, the error message is:

INVALID FLAG ENTRY PAIR OF

The entry is ignored and processing continues.

5.03.07 OUTPUT

The output of the Assembler III Program may be any, or all, of the output options described under Control Options. In addition to the paper tape or card media produced as output, a print-out, which lists the symbolic input, object code developed, and any error comments is produced at the completion of the program. An example of a symbolic listing (with or without Control Options) is illustrated on the following page. The word and syllable of the instruction is listed along with sequence number, object code, expanded print-out of the source card, and decimal equivalent for each label used within a source statement.

If the control option PUNCH is used, a separate print-out of the object code will be produced. Similarly, a card listing is produced if the option OBJCD is used. Examples of these as well as the Label Table (LABELS option) and the Cross Reference Listing (XREF option) are provided.

| WORD | SYL | OBJECT CODE | SEQ. NO. | SYM. LOC. | OP CODE | FD. LN. | A-PARAMETER LABEL | INC | B. PAR | C. PAR | LABEL DEC EQU | REMARKS |
|------|-----|-------------|----------|-----------|---------|---------|-------------------|-----|--------|--------|---------------|--------------------------|
| 6 | 0 | 5R03 | 29 | 0 | IIR | 3 | | 3 | | | | TEST FOR COMPLETION |
| | 1 | 4184 | 30 | 0 | SK | T | | I | 1 | | | TERMINATE LOOP |
| | 2 | 7C04 | 31 | 0 | BRU | SHIPTO | | | | 4 | 3 | REPEAT |
| | 3 | 7809 | 32 | 0 | BRU | RIBBON | | | | 9 | 2 | JUMP TO RIBBON AREA |
| 7 | 0 | 4184 | 33 | 0 | SK | T | | I | 1 | | | TERMINATE LOOP |
| | 1 | 7C04 | 34 | 0 | BRU | SHIPTO | | | | 4 | 3 | REPEAT LOOP |
| | 2 | 4599 | 35 | 0 | EX | K | | 34 | 1 | | | BRANCH TO PRINT SAME |
| | 3 | 7C08 | 36 | 0 | BRU | | + 4 | | | 8 | 3 | EXIT LOOP |
| 8 | 0 | 6744 | 37 | 0 | SET | X | | 2 | | | | SET TO ENTER SHIPTO LOOP |
| | 1 | ED02 | 38 | 0 | AL | 2 | | | | | | ADVANCE TWO LINES |
| | 2 | 7C04 | 39 | 0 | BRU | SHIPTO | | | | 4 | 3 | RETURN TO LOOP |
| | 3 | ED02 | 40 | 0 | AL | 2 | | | | | | ADVANCE TWO LINES |
| 9 | 0 | E80C | 41 | 0 | POS | NMAD-P | | | | 13 | | |
| | 1 | C888 | 42 | 0 | PA | SAME | | | | 136 | | PRINT ALPHA MESSAGE |
| | 2 | E914 | 43 | 0 | RIBBON | ALTO | | | | 20 | | ADVANCE TO RIBBON LINE |
| | 3 | E805 | 44 | 0 | POS | TERM-P | | | | 6 | | POSITION TO TERMS |

Notice in sequence number 420 the actual memory location being referenced by label SAME is 136.
 In sequence number 390 the unconditional branch to SHIPTO is actually a branch to word 4 syllable 3.
 In sequence number 410 the actual value associated with label NMAD-P is 13.

Labels Listing:

| | LABEL | VALUE |
|---|--------|--------|
| 1 | INITAL | 000 03 |
| 2 | BEGINV | 001 02 |
| 3 | SHIPTO | 004 03 |
| 4 | RIBBON | 009 02 |
| 5 | BODYIV | 013 01 |
| 6 | MAX | 015 00 |
| 7 | TKDESC | 020 02 |
| 8 | MPRICE | 022 01 |
| 9 | GRSAMT | 029 01 |

Illustration I

The value of a label refers to the word number associated with the label in defined memory (regions, numeric constants, etc.). Value refers to the word number and syllable number associated with the label in program memory. The list is in the order in which the labels were programmatically defined.

Labels Cross Reference Listing:

| | | | |
|--------|--------|-------|-------------------|
| NE | PKA6 | 01620 | 01920 |
| NE | PKA7 | 00150 | |
| NE | PKA8 | 00050 | |
| 0466.0 | ACT | 02980 | |
| 0192.0 | ADDONS | 01700 | 02260 |
| 0369.0 | ADLNCK | 01740 | 01990 |
| 0502.0 | ADDN=P | 01660 | 01770 01930 02050 |

Illustration II

Associated with each label, listed alphabetically, is the sequence number of the instruction where the label was defined as well as the list of sequence numbers of instructions that reference the label.

5.03.08 LIBTAP -- UTILITY ROUTINE

This Utility Routine creates and updates library tapes to be used with the \$ LIBRARY input option of Assembler III.

Options

1. Creating a library tape from punched cards.
2. Updating a program on the library tape.
3. Combining library tapes.
4. Punching card decks of all or specified programs on a library tape.

*Creating a Library Tape

The create option provides the capability of creating a library tape from punched cards. The control deck necessary to do this is as follows:

```
? EXECUTE LIBTAP
? DATA or DATAB SOURCE
$CREAT
  LIB FILE NAME
SOURCE STATEMENTS
  LIB FILE NAME
SOURCE STATEMENTS
  LIB FILE NAME
SOURCE STATEMENTS
  ETC
  :
  LIB FILE NAME
SOURCE STATEMENTS
? END
```

Notes:

LIB is a pseudo instruction to be typed in CC 22-24. FILE NAME is a 6 character or less program name to be typed in the A parameter, CC 29-34.

SOURCE STATEMENTS are the 80-column cards contained in a program deck. This deck must not contain a pseudo END card; that is, no card with END in CC 22-24 is necessary. The MCP end card should be the last card of the deck.

Result of Create Option:

1. A tape labeled "LIBL" containing files (programs) in 80-column card image. Each program is accessed through the 6 character or less program name on the LIB card preceding the source statements comprising that program.
2. A listing of programs on the tape LIBL.

*Updating a Library Tape

The update option provides the following capabilities: (1) deleting programs, (2) adding programs (3) correcting programs by deletion, addition or correction of source statements. This option requires the use of two tapes.

1. Deleting programs.

The format of the option card is:

CC 1 D
CC 22-24 LIB
CC 29-43 6 character or less program name

2. Adding programs.

The format of the option card is:

CC 1 A
CC 22-24 LIB
CC 29-34 6 character or less program name

The LIB card is followed by the cards that make up the program to be added. This program will be added after all the programs have been copied from the old tape.

3. Correcting programs.

The format of the option card is:

CC 1 C
CC 22-24 LIB
CC 29-34 6 character or less program name

This card is followed by the detail (patch) cards to the program.

These detail cards have the following format:

a. To delete a card

CC 1 D D
CC 11-15 Sequence number
CC 16-80 Blank (must be blank)

b. To add a card

CC 1 A
CC 11-15 Sequence number (unique)
CC 16-80 Format of regular source card

c. To correct a card

CC 1 C
CC 11-15 Sequence number – must match a sequence number of a card
 in the program.
CC 16-80 Corrections made to card

The patch cards must be in sequential order.

The control cards would be:

```
? EXECUTE LIBTAP
? DATA or DATAB SOURCE
$UPDAT
C LIB FILE NAME
D 01400
A 01601 POS 10
C 02000 LIR 3 17
D LIB FILE NAME
```

.Etc

```
? END
```

Result of Updating Option:

1. An updated library tape and an original library tape.
2. A listing of the new library tape.

***Combining Library Tapes**

This option provides the ability to merge two or more library tapes onto a new library tape. The original tapes are not altered. The new library tape should be mounted and original tapes should be mounted (one at a time) as called for by the SPO. The control deck necessary to do this is as follows:

```
? EXECUTE LIBTAP
? DATA SOURCE or DATAB SOURCE
$MERGE n n = number of tapes to be combined, must be one digit
? END
```

Results of Combine Option:

1. A newly created library tape containing all combined programs.
2. A listing of all combined programs.
3. The original unaltered library tapes.

***Punching Programs from Library Tapes**

The punch option provides the capability of punching onto 80-column cards one or more programs from a previously created library tape.

```
? EXECUTE LIBTAP
? DATA SOURCE or ? DATAB SOURCE
(A) $PUNCH n n = any non zero number, must be one digit
LIB FILE NAME
? END
```

or

(B) \$PUNCH 0 Punch all files (programs)

? END

Result of Punch Option:

1. Punched program decks as specified.
2. A listing of each program punched.
3. The unaltered library tape.

5.04.00 L/TC ASSEMBLER IV B 5500 VERSION

The Assembler IV Program operates on a B 5500 system and prepares an object program for Series L/TC systems. It accepts symbolic input directly from cards and will accept a symbolic punched paper tape if the appropriate control card is included in the source media.

All references in this document to Assembler I or Basic Assembler I refer to the Series L/TC keyboard version of Assembler I.

5.04.01 ENVIRONMENT

The following system hardware is required for the TC 500, Basic Assembler Program:

- B 5500 – 4 memory modules utilizing MCP
- 1 Module Disk (300 segments, 240 characters each)
- 1 Tape Unit (7 or 9 channel)
- Line Printer
- Card Reader
- Paper Tape Punch
- Paper Tape Reader (Optional for Symbolic Paper Tape Input)
- Card Punch (Optional for Symbolic or Object Card Output)

5.04.02 MCP CONTROL CARDS

The following MCP Control Cards are used in the assembly process:

Execute Card

The execute card initiates program execution. It must be punched in the following format:

- 1
- 2 EXECUTE ASSEMB/TC500
- 3

Data Card

The data card specifies the source media. It must be punched in the following format:

- 1
- 2 DATA SOURCE (Only BCL card code is accepted.)
- 3

Library Tape Input

The input for Assembler IV is the library tape labeled "TC 500," and the source media which is either a card deck that includes both the symbolic program and the appropriate control cards, or the symbolic paper tape output from Phase I of the Basic Assembler I Program for Series L/TC systems.

Library Tape/TC500

The tapes contain the following programs:

ASSEMB/TC500 This is the assembler program.
XREF/B55TC This is the cross-reference program. It produces a cross-reference listing of labels at the end of assembly.

The following files are used internally in the Assembler IV Program.

0000000/COND Used in error detection.
0000000/OPTBL The operation code file.

Dollar Sign Card

The dollar sign card specifies an option which controls the input and output during the assembly process. It must be punched in the following format starting in card column 1.

\$ (LIST OPTION HERE)

Data Deck

The symbolic deck to be assembled.

End Card

The end card must follow any card deck. It is punched in the following format:

1
2 END
3

It tells the system that the input from the Card Reader is complete.

5.04.03 OPTION CONTROL CARDS

The following options available with Assembler III (B 3500 version) are available with Assembler IV and function identically.

SEQ-CK, Sequence checking
RESEQ, Resequencing
MEMORY, EXTMEM, Memory size
SYM-PT, Symbolic paper tape output
LABELS, Printed table of labels
DOC, Documentation – punch and print
XREF, Cross reference listing
LINES, Specifying number of lines per page
SYM-CD, Symbolic card output
\$ S. L. T., 80-Column Card striped ledger system
\$ S. L., Striped Ledger
\$ EXTEM 40-track machine code

The following options available with Assembler III function differently with Assembler IV.

PUNCH, Paper Tape Object Code

This option is different in that the object program is punched in ASCII code and must be converted to internal Series L/TC format.

OBJ-CD, Object Code Card Output

This option is different in that the object program is punched in BCL and must be converted to internal Series L/TC format.

The option PT-IN is used with Assembler IV in a manner identical to that of DISK-IN with Assembler III. That is, PT-IN is used for symbolic paper tape input.

The option SYM-CN, available with Assembler III, is not available on Assembler IV.

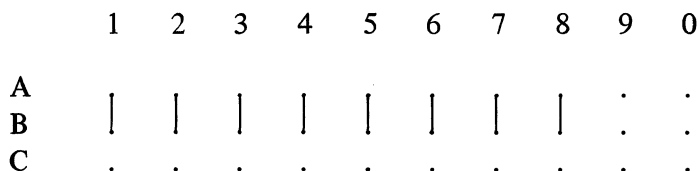
5.04.04 OPERATING INSTRUCTIONS

Operation of the Assembler IV Program is identical to that of Assembler III (B 3500 version) with the following exceptions.

Equipment Setup

The only differences in equipment setup are found with the paper tape reader and paper tape punch. The setup procedure for each is therefore given here.

PAPER TAPE READER – (If symbolic paper tape-in is desired.) The Paper Tape Reader must have an Input Code Translator Board, wired as illustrated in Figure 5-1. (Wire 1 for 1: A-1 to A-1, B-1 to B-1, etc.) Wire the channel select board, inside the front doors on the upper left of the reader in the following manner:



Turn the three control code switches off. Depress the Parity on-off switch to Off. Depress the High-Low switch to Low. Depress the Strip-Reel switch to Reel.

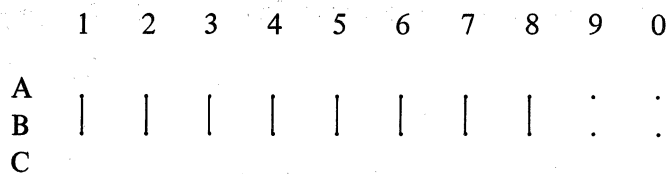
The input paper tape must have an opaque strip approximately 1½" long attached at a minimum of 2½' prior to the first data frame and 2½' after the last frame.

Depress Load Switch. Position tape under the Read Head to the frame preceding the first data frame.

Depress Ready Switch then set Remote/Local Switch to Remote. When the first paper tape read instruction is encountered, the Supervisory Printer will respond with an "unlabeled paper tape file" message. The operator must respond with the appropriate corrective procedure.

PAPER TAPE PUNCH – The Paper Tape Punch must have an Output Code Translator Board, wired as

illustrated in Figure 5-2. (Wire 1 for 1: A-1 to A-1, B-1 to B-1, etc.) Wire the channel select board, inside the front doors on the upper left of the punch, in the following manner:



Turn the Control Code Switch off. Set the LEVEL Designator Switch to 8-LEVEL.

5.04.05 OPERATION

There are a few differences between Assembler IV and Assembler III with respect to the operation of the system. These differences are as follows.

Loading the Library Tape: The SPO message which loads the magnetic library tape is:

CC LOAD FROM TC500 =/=.

If the tape is loaded by card, the necessary control card would be:

1
2 LOAD FROM TC500 =/=.
3

Assembling with Card Input: The first card in the deck must be:

1
2 EXECUTE ASSEMB/TC500.
3

The rest of this procedure is identical to that of Assembler III.

Assembling with Card Input: The option card PT-IN with an Assembler IV replaces the option card DISK-IN of Assembler III. The rest of this operation is the same.

5.04.06 ERROR DETECTION

The editing functions and error messages provided by Assembler IV are identical to Assembler III (B 3500 version).

5.04.07 OUTPUT

The output from Assembler IV is the same as that from Assembler III with a few exceptions.

1. With Assembler IV, no card listing is provided with object card output (OBJCD option).
2. The object paper tape output code with Assembler IV is USASCII rather than the compact hexadecimal provided by Assembler III. Thus, the object code provided by Assembler IV must be converted to compact hexadecimal in order to operate on a Series L/TC System.
3. Object card from Assembler IV is punched in BCL as opposed to compact hexadecimal output from Assembler III. Thus, an output program deck from Assembler IV must be converted prior to attempting to operate on an L/TC System.

\$ PT-IN

This control card is used if the symbolic paper tape output from Phase I of the Assembler I Program is used as input to the Assembler V Program.

\$ SEQ-CK

This control enables sequence checking of the sequence number punched in card columns 11 through 15.

\$ RESEQ

When this control card is used, any sequence numbers punched in columns 11 through 15 of the source cards are ignored. Resequencing is initialized at 10 and incremented by 10 for each succeeding card. If both sequence checking and resequencing are specified, sequence checking will be ignored.

\$ MEMORY

If the size of the object memory is other than 512 words, the desired size may be inserted by the option \$ MEMORY followed by the memory size stated as a 3-digit integer value, starting in card column 29.

\$ SYM-CD

This option provides a symbolic card source deck, resequenced when the \$ RESEQ control option is used.

\$ SYM-PT

This option provides an input symbolic paper tape for Phase I of Assembler I.

\$ OBJCD

This control option is used when the output of the Assembler V Program is to be an object program on punched cards in 80-column card compact hexadecimal format.

\$ O/P OBJECT CARD BCL

This control option is used when the output of the Assembler V Program is to be an object program on punched cards in BCL format.

\$ PUNCH

This control option is used when the output of the Assembler V Program is to be an object program in punched paper tape, USASCII format.

\$ LABELS

The entry of this control option provides a print-out of the label table generated by the Assembler V Program. The list of labels is printed in the order in which they were programmatically defined, along with the location, or value, that each label represents.

5.05.00—L/TC ASSEMBLER V – B 300 VERSION

The Assembler V program operates on a B 300 system and prepares an object program for Series L/TC Systems. The program accepts symbolic input directly from cards and will accept a symbolic punched paper tape if the appropriate control card is included in the source media.

5.05.01 ENVIRONMENT

The following system hardware is used for Assembler V.
B 283 or equivalent having a minimum 9.6K core memory configuration.

- 3 – Tape Storage Units
- Line Printer
- Card Reader
- Card Punch
- Paper Tape Reader (optional)
- Paper Tape Punch (optional)

A Central Processor with punch binary capability is required if object cards in Series L internal code are required.

5.05.02 INPUT

The input for Assembler V is the Assembler V Program object deck and the source media which is either a card deck that includes both the symbolic program and the appropriate control cards, or the symbolic paper tape output from Phase I of Assembler I with appropriate control cards. Symbolic cards are punched as defined in Section 1.

5.05.03 OUTPUT

The output of the Assembler V Program may be any one of the output object options and/or any one of the output symbolics described under control cards.

5.05.04 CONTROL CARDS

Data Cards formulate the option control deck and specify the options which control the input and output of the Assembler V Program. Each card is punched with a (\$) sign in card column 1 and the option information coded starting in column 16.

The control cards allowable are:

\$ Date

This control card is used to express the date. Ten (10) characters are reserved starting in column 29.

\$ I/P Card

This control card is used to tell the system that the source media is cards punched in BCL code.

\$ S. L. T.

This Control option card must be used when assembling for an 80-column card striped ledger system. A character translation table is produced in the object program by the assembler. This option also forces a memory size limit of 482 which supersedes any memory option card.

\$ S. L.

This control card tells the system that the striped ledger option is desired.

\$EXTEM

This control card is required when generating code for a 40-track machine.

5.05.05 END

The system is informed that the input from the Card Reader is complete by use of the mnemonic END card included at the end of the source deck; or by actuation of the END OF FILE button on the Reader.

5.05.06 OPERATING INSTRUCTIONS

Operation of the Assembler V Program involves the setting up of peripherals as a major function as opposed to Assembler I which requires manual intervention as a major function.

Equipment Setup

1. Magnetic tape units

Set the 3 magnetic tape units, designated as stations 1, 2, 3, for remote operation. Make sure that all other magnetic tape units, set for remote operation, are addressed by station numbers other than 1, 2 or 3.

2. Line Printer

Equip the printer with a 6 lines per inch control tape and press the READY button.

3. Card Reader

See Operation.

Paper Tape Reader (if required).

The paper tape reader uses an Input Code Translator Board wired as illustrated in Figure 1. Wire 1 for 1: A-1 to A-1, B-1 to B-1, etc.

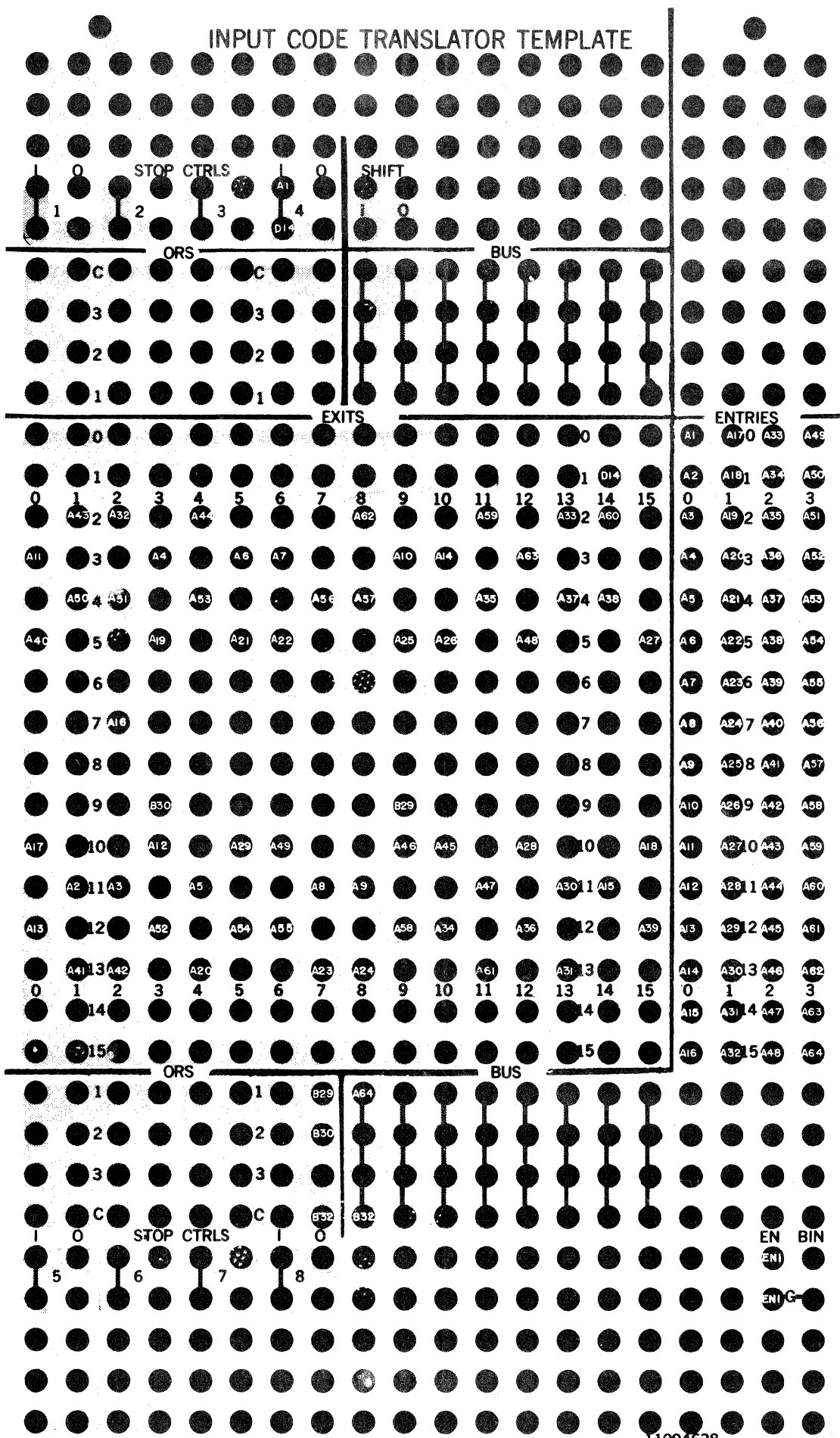
The Channel Select Plugboard is wired one to one:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | | | | | | | | | | |

4. Card Punch (if required).

Load the hopper of the card punch with sufficient cards and press the READY button.

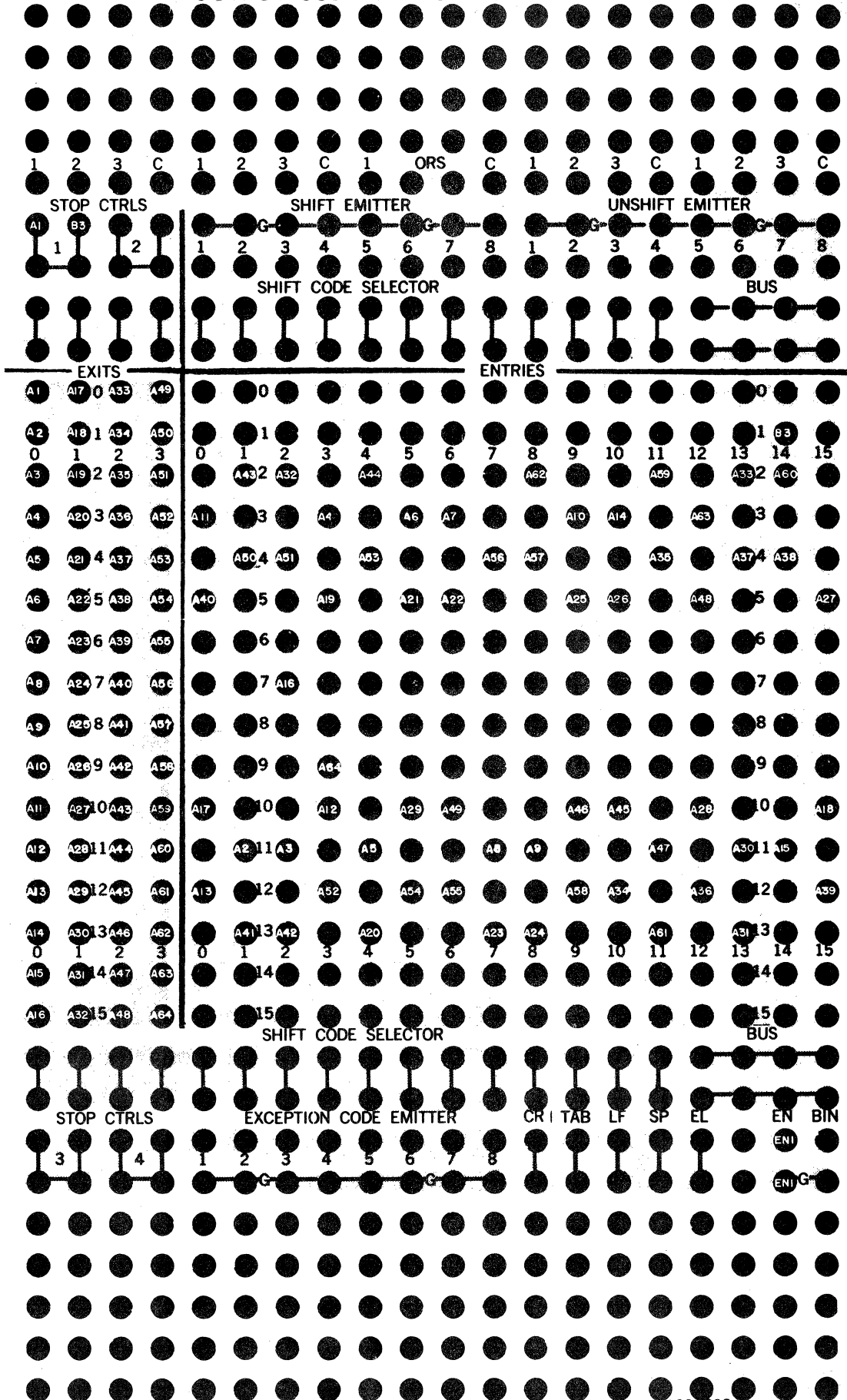
INPUT CODE TRANSLATOR TEMPLATE



Burroughs Corporation

11094638
PRINTED IN U.S.A.

OUTPUT CODE TRANSLATOR TEMPLATE



Burroughs Corporation

11094620
PRINTED IN U.S.A.

FIGURE 2

Revised 8/9/71 by
PCN 1045481-003

5.05.06 (Cont'd-2)

5. Paper Tape Punch (if required).

The paper tape punch uses an Output Code Translator Board wired as illustrated in Figure 2. Wire 1 for 1: A-1 to A-1, B-1 to B-1, etc.

The Channel Select Plugboard, inside the front doors on the upper left of the punch, is wired in the following manner:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | | | | | | | | | | |

Turn the Stop and Delete Switch Off.

Set the Level Switch to 8-Level.

Operation

After the peripherals are set up as described above, proceed with the following operations:

1. Press the CLEAR Switch on the console of the Central Processor.
2. Load the first 160 cards of the Assembler V object deck into the card reader hopper and press the RESET and START switches on the card reader.
3. Press the LOAD switch on the processor and the object cards will begin to read in.
4. After the 160 cards have been read, press the processor CLEAR switch.
5. Place the remainder of the object deck, the control cards and the GP 300 source deck, when used, in the card reader hopper and press the RESET and START switches on the card reader.
6. Press the processor CONTINUE switch and the card reader will read the remaining cards.

NOTE: If the source deck does not end with a GP 300 END pseudo card, press the END OF FILE switch on the card reader after the card reader has read the last card.

7. When the paper tape is used in place of the source deck, the control card deck should be concluded with any card not having a \$ in card column 1. The RESET, START and CONTINUE switches are then pressed to read the remainder of the object deck and control cards.

NOTE: If the control deck of cards is not concluded with a card not having a \$ in card column 1, the END OF FILE and START switches on the card reader must be pressed after the card reader has read the object deck and control cards.

Once the object deck and first control card have been read, the Assembler V Program can be reinitialized automatically by pressing the CLEAR, then CONTINUE switches on the processor console. This means the user can start over at any point of processing without reloading the object card deck.

Programed Halts

The Assembler performs a number of automatic edits of the input data. Programed halts inform the operator of conditions requiring immediate attention. The halt indicator on the central processor is illuminated and the digit 9 is displayed in the O position of the INSTRUCTION register. The M and N positions of the register identify the specific programed halt that is encountered. See table 1.

PROGRAMED HALTS

| INSTRUCTION REGISTER | | | CAUSE | REMEDY |
|----------------------|---|---|--|---|
| O | M | N | | |
| 9 | 2 | 0 | Invalid control card | Correct the last card read, reinsert and press CONTINUE switch on the central processor. |
| 9 | 2 | 2 | End of magnetic tape | Rerun on a larger reel of tape. |
| 9 | 2 | 4 | No input control card entered | Insert card, press CONTINUE. |
| 9 | 2 | 8 | Paper Tape punch out of paper | Reload paper tape punch, press CONTINUE. |
| 9 | 2 | 9 | End of assembly | Press CONTINUE to start next assembly. |
| 9 | 6 | 0 | Paper tape read error | Press CONTINUE. If unable to read, space past the bad record and press CONTINUE again. |
| 9 | 6 | 1 | Paper tape reader out of paper | Reload paper tape reader, press CONTINUE. |
| 9 | 6 | 4 | Paper tape field greater than 26 characters. Tape or reader is in error. | Press CONTINUE to bypass the entry and resume the processing. |
| 9 | 6 | 5 | NUM on paper tape with wrong sign character. Tape or reader in error. | Press CONTINUE to bypass the sign and resume the processing. |
| 9 | 7 | 0 | Paper tape punch out of paper | Reload paper tape punch, press CONTINUE. |
| 9 | 7 | N | Magnetic tape read error. Station number is indicated by N. | Press CONTINUE. If unable to read, restart with new tape and reload the Assembler from cards. |

Table 1

Error Detection

The Assembler V Program, in processing the input, or source data, makes three basic passes through the information.

Pass I accepts input from punched cards or punched paper tape, validates mnemonics and adds control information creating a resultant symbolic output on magnetic tape.

Pass II updates the output or source tape from Pass I, generating a new symbolic tape and creating a magnetic tape label table.

Pass III processes the outputs of Pass II assigning the object code to the symbolic program and supplying a complete listing.

Each error message or group of error messages is followed on the next line of print-out by the symbolic operation which is determined to be in error.

5.05.07 ERROR MESSAGES

No Operation Inserted

NOP INSERTED

This message is used only in conjunction with another error message. The detected error is identified by its specified message followed by NOP INSERTED on the next line. The NOP instruction is inserted in the program by the assembler and processing continues.

No Input Control Card

NO I/P CTL

The user has failed to enter either the \$ I/P CARD or \$ PT IN control card. One or the other has to be inserted in the program and the CONTINUE switch on the central processor activated for continuation of the processing.

Invalid Control Card

INVALID CTL CARD

This programed halt indicates the entered card is not one of the specified control cards. The invalid card is read and printed but must be corrected, reinserted, and the processor CONTINUE switch activated before processing can continue.

Sequence Error

SEQUENCE

The current card being checked at the request of control card \$ SEQ-CK has a lower sequence number than the preceding card and is therefore out of sequence. Processing continues.

No Object

NO OBJECT

This warning message indicates the user failed to specify the type of object output. Processing continues without an object output.

No Symbolic Output

NO O/P SYM

This warning message indicates the user failed to specify the type of symbolic output. Processing continues without a symbolic output.

Invalid Instruction Label

INSTR LABEL INVALID

The instruction label in columns 16 through 21, which must be left justified, begins with an alpha character and contains no blanks, is in error. The label is not entered; processing continues.

Duplicate Instruction Label

INSTR LABEL DUPLICATED

This instruction label has been previously assigned. The label is not entered; processing continues.

Label Limit Exceeded

LABEL LIMIT

More than 139 labels have been assigned and the Assembler V Program therefore cannot be used as input to the Assembler I Program. Processing continues.

Invalid Field Length

INVALID FLD LENGTH

The field length coded in columns 27 and 28, following an ALF instruction, is other than 0-99 or CC. The entry is bypassed and processing continues.

Previous Invalid Field Length

PRE INV FLD LENGTH

Based upon the contents of the field length of the previous ALF instruction, the following entry should have had a field length marked CC. Processing continues.

Invalid A Parameter Label

A-PAR LABEL INVALID

The A parameter label in columns 29 through 34, which must be left justified, begins with an alpha character and contains no blanks, is in error. A NOP is generated and processing continues.

Undefined A Parameter Label

A-PAR LABEL UNDEFINED

This label in the A parameter was never declared as an instruction label. A NOP is generated and processing continues.

Illegal Entry

ILLEGAL

The A parameter, which should be blank, contains an entry. Processing continues.

Invalid Parameter Range

X PARAMETER INVALID RANGE XXX-XXX

The A, B, or C parameter is indicated with its specified permissible range. Comparison of the print-out following the message to the specified range in the error message shows the detected invalidness. A NOP is generated and processing continues.

Invalid Parameter Character

X PARAMETER INVALID X

This message is used with instructions referring to flags. The A or B parameter with the detected invalid flag is specified in the error message. A NOP is generated and processing continues.

Invalid Increment Field

INVALID INCREMENT FLD

Contents of columns 35 through 38, which should be blank or contain the sign in column 35 and 0-255 in columns 36 through 38, are in error. A NOP is generated and processing continues.

No Label or Increment Error

MUST HAVE LBL OR INC

This message is used with BRU mnemonics only. If the label field is empty, the + or - relative address is assumed as the label. A NOP is generated and processing continues.

Invalid Mnemonic

INVALID MNEMONIC

This mnemonic does not appear in the language specification. The instruction is ignored, a NOP is entered and processing continues.

Invalid Sign

INVALID SIGN RESULT

This message is used with a NUM instruction. The user has used an invalid combination of sign characters. The sign portion of the instruction is ignored whereas the rest is printed. Processing continues.

Invalid Character

INVALID CHARACTER(X)

The detected invalid character of the NUM or MASK parameter is indicated in parentheses. Processing continues.

Word Length Exceeded

WORD LENGTH EXCEEDED

This message is used when either a NUM or MASK instruction is too long to be translated into one word. The overflow is ignored and processing continues.

Memory Assignment Error

MEMORY OVERLAYED ERR

This message informs the user that the specified memory has been previously assigned. Processing continues.

Assumed Memory

MEMORY 512

No memory card has been used to specify the memory size. The assembler assumes a memory of 512 and processing continues.

Memory Capacity Exceeded

STORAGE EXCEEDED

The symbolic program is too large for the specified object memory size. Processing continues with the location counter reset to zero.

End Card

LAST LINE NOT "END"

The user failed to use the mnemonic END card to complete the deck file. Processing continues.

GLOSSARY

| | |
|------------------------|---|
| ACCESS TIME | The amount of time required for a computer to locate and transfer a character of data from its storage position and make it available for processing. |
| ACCUMULATOR | A working numeric memory location containing 15 digit positions and a flag position. |
| ALPHA CHARACTER | A character chosen from A-Z. |
| ALPHANUMERIC CHARACTER | A character chosen from A-Z or 0-9 and other specially designated characters. |
| ASSEMBLER | A program written to convert a symbolic program to a corresponding program in machine language. Principally designed to relieve the programmer of the problem of assigning actual storage locations to instructions and data when coding a program and to permit the use of mnemonic operation codes rather than numeric. |
| BASE WORD | The first word in a table. |
| BINARY CODE | A coding system in which successive digits reading from right to left are interpreted as successive powers of two. |
| BLOCK | A memory block consists of 256 words. |
| BRANCH | The point in a program at which the machine will proceed with one or two or more existing possible routines according to existing conditions and instructions. |
| BUFFER | A temporary storage area used to hold data until the data can be accepted for processing. |
| CARD FIELD | A set of card columns fixed in number and position into which the same classification of information appears. |
| CHARACTER | A graphic symbol of any sort. |
| CLEAR MEMORY WORD | A memory word consisting of zero. |
| CODING FORM | A form upon which coding is placed. |
| COMPUTER | A machine for carrying out calculations and performing specified transformations of data. |
| CONSTANT | A magnitude which does not change its value. |
| CONTROL AREA | The area which contains the firmware which determines system control functions. |

APPENDIX A (cont'd)

| | |
|----------------------------------|--|
| DEBUGGING | The process of removing problems from the program so as to meet the program specifications. |
| DIGIT | Any of the figures 0-9. |
| DOCUMENTATION | The explanatory remarks included by a good programmer. |
| ERROR | The amount of precision lost in a quantity. The difference between an accurate quantity and its calculated approximation. Errors occur in numerical methods; mistakes occur in programs, coding, data transcription, and operation; malfunctions occur in computers. |
| FIRMWARE | A control program, stored in the systems memory. The firmware identifies each instruction used by the program and selects the proper "micro string" to perform the functions of the instruction. |
| FLAG | An indicator which is set or reset upon execution of certain instructions, which provides a test factor to determine whether or not the conditions specified by the program exist, so that alternate paths of the program may be selected. |
| FORMAT | An arrangement of information on a form or into storage. |
| HARDWARE | Physical equipment. |
| INPUT | The information fed into a computer system, in the form of numbers or letters, from punched paper tapes, punched cards, keyboard, etc. |
| INSTRUCTION | The information which tells a machine where to obtain the operands, what operations to perform, what to do with the result, and sometimes, where to obtain the next instruction. |
| INTEGER | A whole number. |
| KEYBOARD BASE REGISTER | Specifies the starting memory location in which succeeding information will be stored. |
| LABEL | A set of characters identifying an absolute machine address. |
| LEAST SIGNIFICANT DIGIT POSITION | The 0 position of an Accumulator word or a memory word, if the word is numeric. In an <u>alpha</u> word, the 0 position is the <u>most significant character position</u> . |
| LEFT JUSTIFY | To position a field to begin at the left-most margin. |
| LOGIC | A reasonable analysis of the procedures followed in solving a problem. |

| | |
|---------------------------------|--|
| LOGICAL COMPARISON | The consideration of two things with regard to some characteristic, to obtain a yes if they are the same, or a no if they are different. |
| LOGICAL OPERATION | A computer operation of comparing, selecting, or taking alternative action. |
| LOOP | A number of instructions which occur sequentially, a given number of times. |
| MACHINE LANGUAGE | A code that the computer can recognize and execute. |
| MACRO INSTRUCTION | A main memory instruction which serves to activate a series of micro instructions contained in firmware. |
| MAIN MEMORY | 1,024 words subdivided into the control area and the normal area. |
| MASK | A print format for numeric values. |
| MASK WORD | A word of memory containing mask codes which define print format. |
| MEMORY LOCATION | A component of the computer in which memory is stored. |
| MICROSECOND | A millionth of a second (.000001 = ms.) |
| MILLISECOND | A thousandth of a second (.001 seconds). |
| MNEMONIC OPERATION CODE | See operation code. |
| MOST SIGNIFICANT DIGIT POSITION | The highest digit position of an Accumulator word or a memory word, if the word is numeric. In an <u>alpha</u> word, the <u>highest</u> position or the > position is the <u>least significant character</u> position. |
| MULTIPLICAND | The quantity which is multiplied by each digit of the multiplier in the operation of multiplication. |
| MULTIPLIER | The operand which controls the repetitive addition of the multiplicand in the operation of multiplication. |
| NANOSECOND | A billionth of a second. |
| NEGATIVE VALUE | A value less than zero. |
| NORMAL AREA | An area in memory used to store the users programs which are written with the macro instructions. It is also used to store constant data, messages, and for accumulating totals. |
| OBJECT PROGRAM | A program in machine language resulting from the translation of a source program by an assembler. |

APPENDIX A (cont'd)

| | |
|---------------------------|--|
| OPERATION CODE | A symbolic abbreviation for a machine code which controls a computer function. |
| OUTPUT | The results of computer operations in the form of punched cards, punched paper tape, or printing, etc. |
| PAPER TAPE | A specially treated strip of paper in which a pattern of holes is punched, which in combination with blank spaces represent numbers and letters. |
| PAPER TAPE SOURCE PROGRAM | A source program punched on paper tape. |
| PARAMETER | A quantity which may be assigned different values. |
| PARITY CHECK | A summation check in which the bits, in a character or word are added and the sum checked against a single previously computed parity digit. |
| PRINT HEAD | The print ball. |
| PROGRAM COUNTER | A special register which contains the memory address being executed. |
| PROGRAM KEY | A key which allows insertion of an arbitrary instruction in the program. |
| PROGRAM KEY BASE REGISTER | A register containing the first word of the word program key table. |
| PROGRAM KEY TABLE | An area of memory containing instructions assigned to Program Keys. |
| POSITIVE VALUE | A value greater than zero. |
| PSEUDO INSTRUCTION | An instruction designed to provide information to an assembler program. |
| PUNCH CARD SOURCE PROGRAM | A source program punched on cards. |
| READ IN | To place data in storage at a specified address. |
| RIGHT JUSTIFY | To position a field to terminate at the right most margin. |
| ROUTINE | A set of instructions arranged in proper sequence to cause a machine to perform a desired operation. |
| SAFEGUARD SYMBOL | Commonly referred to as the dollar sign (\$). |
| SEQUENCE NUMBER | A number identifying the desired order of operations. |
| SHIFT | To move an ordered set of characters one or more places to the right or left. |

| | |
|-----------------------|--|
| SIGNIFICANT DIGIT | Any digit (except a zero in places higher than the highest order non-zero digit) in the expression of a quantity. |
| SOFTWARE | The material supplied by a computer manufacturer along with the actual equipment (Hardware) i.e., programs, service routines, and operating manuals. |
| SOURCE PROGRAM | A program written in other than machine language, intended for automatic translation into machine language. |
| STORE | To transfer information to a place from which the unaltered information can be obtained at a later time. |
| SYMBOLIC PROGRAM | The use of arbitrary symbols to represent addresses in order to facilitate programing. |
| TRACK | A track consists of 32 words. |
| TRANSFER | To convey information from one location to another. |
| TRANSLATE | To convert information from one language to another without significantly affecting the meaning. |
| UNPACK | To separate a machine word into parts according to fields of information. |
| UTILITY PROGRAM | A program which has a use in common to a number of different and unrelated programs, yet is not integral to any program, i.e., trace. |
| VARIABLE LENGTH FIELD | A field in which the number of characters within the record are not restricted to a given number of positions. |
| WORD LENGTH | The number of symbols that constitute a word. |
| WORD NUMBER | A memory address or memory location. |
| WORD OF MEMORY | 16 digits (64 bits) used to store information. |
| WORKING STORAGE AREA | A portion of storage in which a data item may be processed or temporarily stored. The term often refers to a place in storage used to retain intermediate results of calculations which will not appear as direct output from the program. |
| ZERO | The integer represented by the character 0, having the property that 0 multiplied by any number and 0 divided by any number is 0. |
| ZERO SUPPRESSION | The elimination of non-significant zeros to the left of the integral part of a quantity before printing is begun. |

GP 300 INSTRUCTIONS TO MACHINE LANGUAGE

Appendix B provides an alphabetical listing for the GP 300 instructions and the hexadecimal machine language code for each GP 300 instruction.

Appendix B also 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, X,Y | 1,2,3,4,-, 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 } } D,R,X,Y,W,S } | -SCMWRF 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, X,Y | 1,2,3,4,-, 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, 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, Previous Ribbon | PC+ | CHARACTER | | |
| Print Character if Accumulator minus, 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, X, Y, L, D | 1, 2, 3, 4, -, S, C, M | |
| Read Alpha and Print | RTK | 0-255 | | |
| Read Alpha into Memory and Punch, 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, X, Y, L, D | 1, 2, 3, 4, -, S, C, M | |
| Skip if any Flag | SK | A, K, R, P S, X, Y, B, L T, V, D, W | 1, 2, 3, 4, -, W S, C, M, R 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, X,Y,L,B D,V,W,S | 1,2,3,4,-,W S,C,M,R 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 as Numeric | TRCA | 1-16 | | |
| Transfer Card Column to Memory as 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, Non-Print | XBA | 1-16 | | |
| Punch Code | XC | 0-15 | 0-15 | |
| Enter Alpha into Memory and Punch, 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 Read Area | XPBA | 1-16 | | |
| Print and Punch Numeric | XPN | 0-14 | 0-15 | |
| Print and Punch Numeric Shift Ribbon if Plus | XPNS+ | 0-14 | 0-15 | |
| Print and Punch Numeric Shift Ribbon 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 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 | | |

OPERATION CODE AND PARAMETER TO MACHINE CODE

| INSTRUCTION | OP CODE | | PARAMETER | | SEE FOOTNOTE |
|-------------|----------|----------|-----------|----------|---------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| ADA | 8 | 8 | 0:15 | 0:15 | 1* |
| ADB | A | | 0 | 0 | 17,18 |
| ADIR | 5 | 4 | 0:15 | 0:15 | 2 |
| ADK | 8 | F | 0:14 | 0:9 | |
| ADM | 8 | 0 | 0:15 | 0:15 | 1 |
| AL | E | D | 0:15 | 0:15 | |
| ALARM | 0 | 9 | 8 | 0 | |
| ALR | E | F | 0:15 | 0:15 | |
| ALTO | E | 9 | 0:15 | 0:15 | |
| ALTP | E | 5 | 0 | 0 | |
| AR | E | E | 0:15 | 0:15 | |
| ARTO | E | A | 0:15 | 0:15 | |
| BRU | 7 | 0 | 0:15 | 0:15 | 3 |
| CC | E | C | 0 | 0 | |
| CDC | B | 2 | 2:15 | 0:9 | |
| CDV | B | 3 | 2:15 | 0:9 | |
| CHG | 6 | 6 | | | 5 |
| CLA | 8 | E | 0:15 | 0:15 | |
| CLM | D | 8 | 0:15 | 0:15 | 1* |
| CPA | D | A | 0:15 | 0:15 | 1* |
| DIR | 5 | C | 0:15 | 0:15 | 2 |
| DIV | 9 | A | 0:15 | 0:15 | 1* |
| DUP | E | 1 | 0:5 | 0:15 | |
| EAM | A | 9 | 0:5 | 0:15 | 8 |
| EL | A | 7 | B | 0 | |
| EX | 4 | | | | 4,5 |
| EXE | 4 | | | | 4,5 |
| EXL | 6 | | 0:15 | 0:15 | 4 |
| EXZ | 4 | | D | 0 | 4 |

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| IIR | 5 | 8 | 0:15 | 0:15 | 2 |
| INK | 9 | E | 0:15 | 0:15 | |
| IRCP | 1 | A | 0:15 | 0:15 | |
| LA | A | 7 | 8 | 0 | |
| LBN | 3 | 4 | A | 9 | |
| LCD | C | D | 0:15 | 0:15 | |
| LCFR | D | C | 0:15 | 0:15 | 1 |
| LGN | 3 | 4 | A | 8 | |
| LIR | 5 | 0 | 0:15 | 0:15 | 2 |
| LKBR | F | 0 | 0:15 | 0:15 | 1 |
| LLCR | E | 0 | 0:15 | 0:15 | |
| LLLR | E | 4 | 0:15 | 0:15 | |
| LOD | 6 | 4 | | | 5 |
| LPF | 3 | 4 | A | D | |
| LPKR | F | C | 0:15 | 0:15 | 1 |
| LPNR | F | 8 | 0:15 | 0:15 | 1 |
| LPR | 3 | 2 | 4 | A | |
| LRA | 3 | 4 | B | 1 | |
| LRBR | 1 | 8 | 0:15 | 0:15 | 1 |
| LRCR | E | 2 | 0:15 | 0:15 | |
| LRLR | E | 6 | 0:15 | 0:15 | |
| LSA | 3 | 4 | B | 2 | |
| LSCR | B | 7 | 0 | 0 | 19 |
| LSFR | B | 4 | 0:15 | 0:15 | 1 |
| LSLR | B | 7 | 4 | 0 | 19 |
| LSN | 3 | 4 | A | 7 | |
| LSR | 6 | 4 | 2 | 0:15 | |
| LTF | 3 | 4 | A | 4 | |
| LTN | 3 | 4 | A | 6 | |
| LXC | 0 | 6 | 0:15 | 0:15 | |

APPENDIX B (Cont'd.)

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| MOD | 6 | 0 | 0 | 0 | 2 |
| MUL | 8 | A | 0:15 | 0:15 | 1* |
| MULR | 8 | C | 0:15 | 0:15 | 1* |
| NK | A | 6 | 0:15 | 0:15 | |
| NKCM | A | 2 | 0:15 | 0:15 | |
| NKR | A | 4 | 0:15 | 0:15 | |
| NKRCM | A | 0 | 0:15 | 0:15 | |
| NOP | 0 | 8 | 0 | 0 | |
| OC | E | 8 | 0:15 | 0:15 | |
| OFF | 0 | 9 | 1 | 0 | |
| PA | C | 8 | 0:15 | 0:15 | 1*, 20 |
| PA | 1 | 8 | 0:15 | 0:15 | 1*, 21 |
| PAB | 1 | D | 0:15 | 0:15 | 8 |
| PAS | A | 5 | 0 | 0 | 18 |
| PBA | B | C | 0 | 0:15 | 18 |
| PC | C | 0 | | | 7 |
| PC+ | C | 4 | | | 7 |
| PC- | C | 5 | | | 7 |
| PCP | C | 1 | | | 7 |
| PKA | F | 6 | 0:15 | 0:15 | 6 |
| PKB | F | 7 | 0:15 | 0:15 | 6 |
| PKC | F | 5 | 0:15 | 0:15 | 6 |
| PN | D | 4 | 0:14 | 0:15 | |
| PNS+ | D | 0 | 0:14 | 0:15 | |
| PNS- | D | 1 | 0:14 | 0:15 | |
| POF | 0 | 1 | 1 | 0 | |
| PON | 0 | 1 | 2 | 0 | |
| POS | E | B | 0:15 | 0:15 | 8 |

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| RBN | 3 | C | A | 9 | |
| RCD | C | C | 0 | 0 | |
| RCP | 1 | C | 0:15 | 0:15 | |
| REAM | B | 9 | 0:15 | 0:15 | 8 |
| REL | 0 | 1 | 0 | 1 | |
| REM | 3 | A | 4 | 1 | 20 |
| REM | 3 | B | 4 | 1 | 21 |
| RET | A | 7 | C | 0 | |
| RGN | 3 | C | A | 8 | |
| RL | A | 7 | 0:5 | 0:15 | 23 |
| RNK | B | 0 | 0:15 | 0:15 | |
| RPF | 3 | C | A | D | |
| RPR | 3 | A | 4 | A | |
| RR | 0 | 7 | 0 | 0 | |
| RRA | 3 | C | B | 1 | |
| RSA | 3 | C | B | 2 | |
| RSN | 3 | C | A | 7 | |
| RST | 6 | 5 | | | 5 |
| RTF | 3 | C | A | 4 | |
| RTH | 3 | C | A | 0 | |
| RTK | B | C | 0:15 | 0:15 | 8 |
| RTKM | B | D | 0:15 | 0:15 | 8 |
| RTN | 3 | C | A | 6 | |
| RXEAM | B | B | 0:15 | 0:15 | 8 |
| RXTK | B | E | 0:15 | 0:15 | 8 |
| RXTKM | B | F | 0:15 | 0:15 | 8 |
| SCP | 1 | 4 | 0:15 | 0:15 | |
| SET | 6 | 7 | | | 5 |
| SK | 4 | | | | 4,5 |
| SKE | 4 | | | | 4,5 |
| SKL | 6 | | 0:15 | 0:15 | 4 |
| SKP | E | 3 | 0:5 | 0:15 | |
| SKZ | 4 | | D | 0 | 4 |

APPENDIX B (Cont'd.)

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| SLRO | 0 | 2 | 0:14 | 0:14 | |
| SLROS | 0 | 3 | 0:15 | 0:15 | |
| SRJ | 2 | 0 | 0:15 | 0:15 | 3 |
| SRR | 0 | 4 | 0 | | 9 |
| STOP | 0 | 0 | 0 | 0 | |
| SUA | 9 | 8 | 0:15 | 0:15 | 1* |
| SUB | A | | 8 | 0 | 17, 18 |
| SUK | 9 | F | 0:14 | 0:9 | |
| SUM | 9 | 0 | 0:15 | 0:15 | 1 |
| TAIR | 9 | C | 0 | 0:3 | 22 |
| TASB | A | | C | 0 | 17, 18 |
| TK | A | C | 0:15 | 0:15 | 8 |
| TKM | A | D | 0:15 | 0:15 | 8 |
| TMSB | A | 5 | 8 | 0 | 18 |
| TRA | 3 | 8 | 0:15 | 0:15 | 1 |
| TRAB | 1 | 5 | 0:1 | 0:15 | |
| TRB | 1 | E | 0 | 1:15 | |
| TRBA | 1 | B | 0 | 1:15 | |
| TRCA | B | 8 | 0 | 0:15 | 18 |
| TRCB | 1 | 6 | 0:15 | 0:15 | 16 |
| TRCM | B | 9 | 0 | 0:15 | 18 |
| TRF | 1 | 7 | 0:15 | 0:15 | |
| TRM | 3 | 0 | 0:15 | 0:15 | 1 |
| TSB | 1 | F | 0 | 1:15 | |
| TSBA | A | | 4 | 0 | 17, 18 |
| TSBM | A | 5 | 4 | 0 | 18 |

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| WL | A | 7 | A | 0 | |
| XA | C | 6 | 0:15 | 0:15 | 1* |
| XB | 0 | C | 0:15 | 0:15 | |
| XBA | B | A | 0 | 0:15 | |
| XC | C | 2 | 0:15 | 0:15 | 16 |
| XEAM | A | B | 0:15 | 0:15 | 8 |
| XMOD | 0 | A | 0 | 0 | |
| XN | D | 7 | 0:14 | 0:15 | |
| XPA | C | A | 0:15 | 0:15 | 1*, 20 |
| XPA | 1 | C | 0:15 | 0:15 | 1*, 21 |
| XPBA | B | E | 0 | 0:15 | |
| XPN | D | 6 | 0:14 | 0:15 | |
| XPNS+ | D | 2 | 0:14 | 0:15 | |
| XPNS- | D | 3 | 0:14 | 0:15 | |
| XTK | A | E | 0:15 | 0:15 | 8 |
| XTKM | A | F | 0:15 | 0:15 | 8 |

**OPERATION CODE AND PARAMETER TO MACHINE CODE
(DATA COMMUNICATIONS ADDITION)**

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| IRCP | 1 | A | 0:15 | 0:15 | 11 |
| LKBR | F | 0 | 0:15 | 0:15 | 1, 10 |
| LPF | 3 | 4 | A | D | |
| LPR | 3 | 2 | 4 | A | |
| LRA | 3 | 4 | B | 1 | |
| LRBR | 1 | 8 | 0:15 | 0:15 | 1, 10 |
| LSA | 3 | 4 | B | 2 | |
| LSN | 3 | 4 | A | 7 | |
| LSR | 6 | 4 | 2 | 0:15 | |
| LTF | 3 | 4 | A | 4 | |
| LTN | 3 | 4 | A | 6 | |
| OFF | | | | | |
| PAB | 1 | D | 0:9 | 0:15 | 8 |
| RCP | 1 | C | 0:15 | 0:15 | 11 |
| RPF | 3 | C | A | D | |
| RPR | 3 | A | 4 | A | |
| RRA | 3 | C | B | 1 | |
| RSA | 3 | C | B | 2 | |
| RSN | 3 | C | A | 7 | |
| RTH | 3 | C | A | 0 | |
| RTK | 3 | C | A | 4 | |
| RTN | 3 | C | A | 6 | |
| SCP | 1 | 4 | 0:15 | 0:15 | 11 |

| <u>INSTRUCTION</u> | <u>OP CODE</u> | | <u>PARAMETER</u> | | <u>SEE FOOTNOTE</u> |
|--------------------|----------------|----------|------------------|----------|---------------------|
| | <u>U</u> | <u>L</u> | <u>U</u> | <u>L</u> | <u>NUMBER</u> |
| TKM | | | | | |
| TRAB | 1 | 5 | | 0:15 | 15 |
| TRB | 1 | E | | 1:15 | 12 |
| TRBA | 1 | B | | 1:16 | 13 |
| TRCB | 1 | 6 | 0:15 | 0:15 | 16 |
| TRF | 1 | 7 | 0:15 | 0:15 | 14 |
| TSB | 1 | F | | 1:15 | 12 |

FOOTNOTES

FOOTNOTE (Statements preceded by an asterisk apply to only extended memory machines.)

1. For word number 256:511 add 1 to the OP code lower.

For word number 512:767 add 2 to the OP code lower except for the following instructions which are restricted to referencing words 0 to 511 of user memory:

ADA
SUA
MUL
MULR
DIV
CLM
CPA
XA

The PA instruction for the 40-track system has a new OP code of 18. For word number 256:511 add 1 to the OP code lower (18 + 1=19). For word number 512:767 add 2 to the OP code lower (18 + 2=1A).

*These instructions are restricted to referencing words of 0 to 511 of user memory.

2. Modify OP code lower as follows:

| | | | | |
|-----------------|---|---|---|---|
| INEX REG. NO. | 4 | 1 | 2 | 3 |
| ADD TO OP LOWER | 0 | 1 | 2 | 3 |

3. Modify OP code lower as follows:

| <u>Syllable</u> | <u>Word Number</u> | <u>Add to OP Lower</u> |
|-----------------|--------------------|------------------------|
| 0 | 0:255 | 0 |
| 1 | 0:255 | 4 |
| 2 | 0:255 | 8 |
| 3 | 0:255 | C |
| 0 | 256:511 | 1 |
| 1 | 256:511 | 5 |
| 2 | 256:511 | 9 |
| 3 | 256:511 | D |
| *0 | 512:767 | 2 |
| *1 | 512:767 | 6 |
| *2 | 512:767 | A |
| *3 | 512:767 | E |

4. OP code lower is derived from table below.

| <u>INSTRUCTION</u> | <u># OF INSTRUCTIONS TO BE EXECUTED OR SKIPPED</u> | <u>OP LOWER</u> |
|--------------------|--|-----------------|
| SK | 4 | 0 |
| or | 1 | 1 |
| SKZ | 2 | 2 |
| | 3 | 3 |
| EX | 4 | 4 |
| or | 1 | 5 |
| EXZ | 2 | 6 |
| | 3 | 7 |
| SKE | 4 | 8 |
| | 1 | 9 |
| | 2 | A |
| | 3 | B |
| EXE | 4 | C |
| | 1 | D |
| | 2 | E |
| | 3 | F |
| SKL | 4 | 8 |
| | 1 | 9 |
| | 2 | A |
| | 3 | B |
| EXL | 4 | C |
| | 1 | D |
| | 2 | E |
| | 3 | F |

5. Parameter Upper Digit is determined as follows:

| <u>FLAG TYPE</u> | | <u>PARAMETER UPPER</u> |
|------------------|------------|------------------------|
| PUNCH | P | 1 |
| READ | R | 0 |
| | X | 4 |
| | Y | 5 |
| TEST | T | 8 |
| LOCK'S | K | 9 |
| ACCUM | A | C |
| KEYBOARD | B (Buffer) | 3 |
| DATA COMM | D (Buffer) | A |
| SKZ | | D |
| EKZ | | D |

Parameter Lower Digit is determined as follows:

| <u>FLAG GROUP</u> | <u>MACHINE LANGUAGE</u> |
|-------------------------|-------------------------|
| A T P, R, X, Y, K, B, D | <u>CODE VALUE</u> |
| - 0 4 | 1 |
| S L 1 | 2 |
| C I 2 | 4 |
| M U 3 | 8 |

EXAMPLE: If test flag L is being tested, the lower digit parameter is 2. If test flag L and U are being tested, the lower digit parameter is the sum of the representative code values 2 and 8 = 10. Since the hexadecimal representative of 10 is A, the Parameter lower digit is A.

6. Program Key Parameter Designation is determined as follows for Upper and Lower Digit Parameters.

| | UPPER | | | | LOWER | | | |
|-------------|-------|---|---|---|-------|---|---|---|
| PROGRAM KEY | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| WEIGHT | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 |

EXAMPLE: To determine upper and lower parameter values for keys 7 and 4. Key 7 value is 4, key 4 value is 8. The upper and lower parameter digits are then 4, 8.

EXAMPLE: To determine Upper and Lower parameter values for keys 8 and 5. Key 8 value is 8 and 5 value is 1. Both are within the Upper parameter giving an Upper parameter digit of 9 (8 + 1) and Lower parameter digit of 0.

7. PC character codes are determined from the following chart:

TC 500 CHARACTER SETS

The USASCII and Commerical character sets for the TC 500 are listed below in their collating sequence in ascending order. Each character set consists of 64 graphic characters, the Space code, and the End of Alpha code. The USASCII character set consists of the USASCII characters in columns 2, 3, 4, and 5 of the USASCII table, plus End of Alpha (NUL) and Overline. Those Commercial characters that differ from the USASCII characters are shown in parentheses.

The internal or machine language code for each character is given; this code consists of two hexadecimal digits which correspond to the column and row number of the character in the USASCII table (A = row 10, B = 11, C = 12, D = 13, E = 14, F = 15). In addition, the decimal value of each character is given as required when using Index Registers for modification.

| Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value |
|--------------------|---------------|----------------|-----------|---------------|----------------|-----------|---------------|----------------|-----------|---------------|----------------|
| End of Alpha (NUL) | 0 0 | 0 | | | | | | | | | |
| Space | 2 0 | 32 | 0 | 3 0 | 48 | @ | 4 0 | 64 | P | 5 0 | 80 |
| ! | 2 1 | 33 | 1 | 3 1 | 49 | A | 4 1 | 65 | Q | 5 1 | 81 |
| ” | 2 2 | 34 | 2 | 3 2 | 50 | B | 4 2 | 66 | R | 5 2 | 82 |
| # | 2 3 | 35 | 3 | 3 3 | 51 | C | 4 3 | 67 | S | 5 3 | 83 |
| \$ | 2 4 | 36 | 4 | 3 4 | 52 | D | 4 4 | 68 | T | 5 4 | 84 |
| % | 2 5 | 37 | 5 | 3 5 | 53 | E | 4 5 | 69 | U | 5 5 | 85 |
| & | 2 6 | 38 | 6 | 3 6 | 54 | F | 4 6 | 70 | V | 5 6 | 86 |
| , | 2 7 | 39 | 7 | 3 7 | 55 | G | 4 7 | 71 | W | 5 7 | 87 |
| (| 2 8 | 40 | 8 | 3 8 | 56 | H | 4 8 | 72 | X | 5 8 | 88 |
|) | 2 9 | 41 | 9 | 3 9 | 57 | I | 4 9 | 73 | Y | 5 9 | 89 |
| * | 2 A | 42 | : | 3 A | 58 | J | 4 A | 74 | Z | 5 A | 90 |
| + | 2 B | 43 | ; | 3 B | 59 | K | 4 B | 75 | [(%) | 5 B | 91 |
| , | 2 C | 44 | <(½) | 3 C | 60 | L | 4 C | 76 | \(φ) | 5 C | 92 |
| - | 2 D | 45 | = | 3 D | 61 | M | 4 D | 77 |] (CR) | 5 D | 93 |
| . | 2 E | 46 | >(¼) | 3 E | 62 | N | 4 E | 78 | ^ (°) | 5 E | 94 |
| / | 2 F | 47 | ? | 3 F | 63 | O | 4 F | 79 | - | 5 F | 95 |
| | | | | | | | | | ~(◊) | 7 E | 126 |
| | | | | | | | | | DEL | 7 F | 127 |

8. PARAMETER LIMIT

| | <u>LIMIT</u> |
|------------------------|--------------|
| 15½ INCH FORMS HANDLER | 0:150 |
| 26 INCH FORMS HANDLER | 0:255 |

9. Lower Parameter determined as follows:

| | | | | |
|-------------------------|---|---|---|---|
| SUBROUTINE RETURN LEVEL | 1 | 2 | 3 | 4 |
| MACHINE CODE | 0 | 1 | 2 | 3 |

10. Use upper and lower parameter of 0 (zero) to indicate data communication processor send or receive buffer.

11. Parameter limit 1:255

12. Parameter limit 1:15

13. Parameter limit 0:16

14. Parameter limit 0:255

15. Parameter limit 0:15

16. Column number (stick number) from USASCII table is upper parameter. Row number (level number) from USASCII table is the lower parameter.

17. Lower OP code is determined as follows:

| | |
|----------|---|
| Unsigned | 1 |
| Signed | 3 |

18. Add hexadecimal value of format number, minus 1, to the upper and lower parameter.

19. Add hexadecimal value of line number, minus 1, to the upper and lower parameter.

20. For 32-track systems.

21. For 40-track systems.

22. Modify lower parameter as follows:

| <u>INDEX REG. NO.</u> | <u>LOWER PARAMETER</u> |
|-----------------------|------------------------|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 0 |

23. Upper parameter is determined as follows:

- 0 Read and align to the line number on the magnetic unit record.
- 1 Read and align to the line number contained in the magnetic unit record count register.
- 2 Read and align to the first posting line.
- 3 Non-read and align to the line number contained in the magnetic unit record count register.
- 4 Read and eject magnetic unit record.
- 5 Read from magnetic unit record reader.

DECIMAL TO HEXADECIMAL CONVERSION TABLE

| DEC. EQUIV. | ADD TO OP | | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | | PARAMETER FIELD | | | | | |
|----------------|--------------|---|--------------------|--|----------------|--------------|---|--------------------|--|----------------|--------------|---|--------------------|--|-----|---|---|---|
| | L | U | L | | | L | U | L | | | L | U | L | | L | U | L | |
| 256 | 1 | 0 | 0 | | 288 | 1 | 2 | 0 | | 320 | 1 | 4 | 0 | | 352 | 1 | 6 | 0 |
| 257 | 1 | 0 | 1 | | 289 | 1 | 2 | 1 | | 321 | 1 | 4 | 1 | | 353 | 1 | 6 | 1 |
| 258 | 1 | 0 | 2 | | 290 | 1 | 2 | 2 | | 322 | 1 | 4 | 2 | | 354 | 1 | 6 | 2 |
| 259 | 1 | 0 | 3 | | 291 | 1 | 2 | 3 | | 323 | 1 | 4 | 3 | | 355 | 1 | 6 | 3 |
| 260 | 1 | 0 | 4 | | 292 | 1 | 2 | 4 | | 324 | 1 | 4 | 4 | | 356 | 1 | 6 | 4 |
| 261 | 1 | 0 | 5 | | 293 | 1 | 2 | 5 | | 325 | 1 | 4 | 5 | | 357 | 1 | 6 | 5 |
| 262 | 1 | 0 | 6 | | 294 | 1 | 2 | 6 | | 326 | 1 | 4 | 6 | | 358 | 1 | 6 | 6 |
| 263 | 1 | 0 | 7 | | 295 | 1 | 2 | 7 | | 327 | 1 | 4 | 7 | | 359 | 1 | 6 | 7 |
| 264 | 1 | 0 | 8 | | 296 | 1 | 2 | 8 | | 328 | 1 | 4 | 8 | | 360 | 1 | 6 | 8 |
| 265 | 1 | 0 | 9 | | 297 | 1 | 2 | 9 | | 329 | 1 | 4 | 9 | | 361 | 1 | 6 | 9 |
| 266 | 1 | 0 | A | | 298 | 1 | 2 | A | | 330 | 1 | 4 | A | | 362 | 1 | 6 | A |
| 267 | 1 | 0 | B | | 299 | 1 | 2 | B | | 331 | 1 | 4 | B | | 363 | 1 | 6 | B |
| 268 | 1 | 0 | C | | 300 | 1 | 2 | C | | 332 | 1 | 4 | C | | 364 | 1 | 6 | C |
| 269 | 1 | 0 | D | | 301 | 1 | 2 | D | | 333 | 1 | 4 | D | | 365 | 1 | 6 | D |
| 270 | 1 | 0 | E | | 302 | 1 | 2 | E | | 334 | 1 | 4 | E | | 366 | 1 | 6 | E |
| 271 | 1 | 0 | F | | 303 | 1 | 2 | F | | 335 | 1 | 4 | F | | 367 | 1 | 6 | F |
| 272 | 1 | 1 | 0 | | 304 | 1 | 3 | 0 | | 336 | 1 | 5 | 0 | | 368 | 1 | 7 | 0 |
| 273 | 1 | 1 | 1 | | 305 | 1 | 3 | 1 | | 337 | 1 | 5 | 1 | | 369 | 1 | 7 | 1 |
| 274 | 1 | 1 | 2 | | 306 | 1 | 3 | 2 | | 338 | 1 | 5 | 2 | | 370 | 1 | 7 | 2 |
| 275 | 1 | 1 | 3 | | 307 | 1 | 3 | 3 | | 339 | 1 | 5 | 3 | | 371 | 1 | 7 | 3 |
| 276 | 1 | 1 | 4 | | 308 | 1 | 3 | 4 | | 340 | 1 | 5 | 4 | | 372 | 1 | 7 | 4 |
| 277 | 1 | 1 | 5 | | 309 | 1 | 3 | 5 | | 341 | 1 | 5 | 5 | | 373 | 1 | 7 | 5 |
| 278 | 1 | 1 | 6 | | 310 | 1 | 3 | 6 | | 342 | 1 | 5 | 6 | | 374 | 1 | 7 | 6 |
| 279 | 1 | 1 | 7 | | 311 | 1 | 3 | 7 | | 343 | 1 | 5 | 7 | | 375 | 1 | 7 | 7 |
| 280 | 1 | 1 | 8 | | 312 | 1 | 3 | 8 | | 344 | 1 | 5 | 8 | | 376 | 1 | 7 | 8 |
| 281 | 1 | 1 | 9 | | 313 | 1 | 3 | 9 | | 345 | 1 | 5 | 9 | | 377 | 1 | 7 | 9 |
| 282 | 1 | 1 | A | | 314 | 1 | 3 | A | | 346 | 1 | 5 | A | | 378 | 1 | 7 | A |
| 283 | 1 | 1 | B | | 315 | 1 | 3 | B | | 347 | 1 | 5 | B | | 379 | 1 | 7 | B |
| 284 | 1 | 1 | C | | 316 | 1 | 3 | C | | 348 | 1 | 5 | C | | 380 | 1 | 7 | C |
| 285 | 1 | 1 | D | | 317 | 1 | 3 | D | | 349 | 1 | 5 | D | | 381 | 1 | 7 | D |
| 286 | 1 | 1 | E | | 318 | 1 | 3 | E | | 350 | 1 | 5 | E | | 382 | 1 | 7 | E |
| 287 | 1 | 1 | F | | 319 | 1 | 3 | F | | 351 | 1 | 5 | F | | 383 | 1 | 7 | F |

DECIMAL TO HEXADECIMAL CONVERSION TABLE

| DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | |
|-------------|-----------|-----------------|---|-------------|-----------|-----------------|---|-------------|-----------|-----------------|---|-------------|-----------|-----------------|---|
| | | L | U | | | L | L | | | U | L | | | L | U |
| 384 | 1 | 8 | 0 | 416 | 1 | A | 0 | 448 | 1 | C | 0 | 480 | 1 | E | 0 |
| 385 | 1 | 8 | 1 | 417 | 1 | A | 1 | 449 | 1 | C | 1 | 481 | 1 | E | 1 |
| 386 | 1 | 8 | 2 | 418 | 1 | A | 2 | 450 | 1 | C | 2 | 482 | 1 | E | 2 |
| 387 | 1 | 8 | 3 | 419 | 1 | A | 3 | 451 | 1 | C | 3 | 483 | 1 | E | 3 |
| 388 | 1 | 8 | 4 | 420 | 1 | A | 4 | 452 | 1 | C | 4 | 484 | 1 | E | 4 |
| 389 | 1 | 8 | 5 | 421 | 1 | A | 5 | 453 | 1 | C | 5 | 485 | 1 | E | 5 |
| 390 | 1 | 8 | 6 | 422 | 1 | A | 6 | 454 | 1 | C | 6 | 486 | 1 | E | 6 |
| 391 | 1 | 8 | 7 | 423 | 1 | A | 7 | 455 | 1 | C | 7 | 487 | 1 | E | 7 |
| 392 | 1 | 8 | 8 | 424 | 1 | A | 8 | 456 | 1 | C | 8 | 488 | 1 | E | 8 |
| 393 | 1 | 8 | 9 | 425 | 1 | A | 9 | 457 | 1 | C | 9 | 489 | 1 | E | 9 |
| 394 | 1 | 8 | A | 426 | 1 | A | A | 458 | 1 | C | A | 490 | 1 | E | A |
| 395 | 1 | 8 | B | 427 | 1 | A | B | 459 | 1 | C | B | 491 | 1 | E | B |
| 396 | 1 | 8 | C | 428 | 1 | A | C | 460 | 1 | C | C | 492 | 1 | E | C |
| 397 | 1 | 8 | D | 429 | 1 | A | D | 461 | 1 | C | D | 493 | 1 | E | D |
| 398 | 1 | 8 | E | 430 | 1 | A | E | 462 | 1 | C | E | 494 | 1 | E | E |
| 399 | 1 | 8 | F | 431 | 1 | A | F | 463 | 1 | C | F | 495 | 1 | E | F |
| 400 | 1 | 9 | 0 | 432 | 1 | B | 0 | 464 | 1 | D | 0 | 496 | 1 | F | 0 |
| 401 | 1 | 9 | 1 | 433 | 1 | B | 1 | 465 | 1 | D | 1 | 497 | 1 | F | 1 |
| 402 | 1 | 9 | 2 | 434 | 1 | B | 2 | 466 | 1 | D | 2 | 498 | 1 | F | 2 |
| 403 | 1 | 9 | 3 | 435 | 1 | B | 3 | 467 | 1 | D | 3 | 499 | 1 | F | 3 |
| 404 | 1 | 9 | 4 | 436 | 1 | B | 4 | 468 | 1 | D | 4 | 500 | 1 | F | 4 |
| 405 | 1 | 9 | 5 | 437 | 1 | B | 5 | 469 | 1 | D | 5 | 501 | 1 | F | 5 |
| 406 | 1 | 9 | 6 | 438 | 1 | B | 6 | 470 | 1 | D | 6 | 502 | 1 | F | 6 |
| 407 | 1 | 9 | 7 | 439 | 1 | B | 7 | 471 | 1 | D | 7 | 503 | 1 | F | 7 |
| 408 | 1 | 9 | 8 | 440 | 1 | B | 8 | 472 | 1 | D | 8 | 504 | 1 | F | 8 |
| 409 | 1 | 9 | 9 | 441 | 1 | B | 9 | 473 | 1 | D | 9 | 505 | 1 | F | 9 |
| 410 | 1 | 9 | A | 442 | 1 | B | A | 474 | 1 | D | A | 506 | 1 | F | A |
| 411 | 1 | 9 | B | 443 | 1 | B | B | 475 | 1 | D | B | 507 | 1 | F | B |
| 412 | 1 | 9 | C | 444 | 1 | B | C | 476 | 1 | D | C | 508 | 1 | F | C |
| 413 | 1 | 9 | D | 445 | 1 | B | D | 477 | 1 | D | D | 509 | 1 | F | D |
| 414 | 1 | 9 | E | 446 | 1 | B | E | 478 | 1 | D | E | 510 | 1 | F | E |
| 415 | 1 | 9 | F | 447 | 1 | B | F | 479 | 1 | D | F | 511 | 1 | F | F |

DECIMAL TO HEXADECIMAL CONVERSION TABLE

| DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | | DEC. EQUIV. | ADD TO OP | PARAMETER FIELD | |
|----------------|--------------|--------------------|---|----------------|--------------|--------------------|---|----------------|--------------|--------------------|---|----------------|--------------|--------------------|---|
| | | L | U | | | L | U | | | L | L | | | U | L |
| 512 | 2 | 0 | 0 | 544 | 2 | 2 | 0 | 576 | 2 | 4 | 0 | 608 | 2 | 6 | 0 |
| 513 | 2 | 0 | 1 | 545 | 2 | 2 | 1 | 577 | 2 | 4 | 1 | 609 | 2 | 6 | 1 |
| 514 | 2 | 0 | 2 | 546 | 2 | 2 | 2 | 578 | 2 | 4 | 2 | 610 | 2 | 6 | 2 |
| 515 | 2 | 0 | 3 | 547 | 2 | 2 | 3 | 579 | 2 | 4 | 3 | 611 | 2 | 6 | 3 |
| 516 | 2 | 0 | 4 | 548 | 2 | 2 | 4 | 580 | 2 | 4 | 4 | 612 | 2 | 6 | 4 |
| 517 | 2 | 0 | 5 | 549 | 2 | 2 | 5 | 581 | 2 | 4 | 5 | 613 | 2 | 6 | 5 |
| 518 | 2 | 0 | 6 | 550 | 2 | 2 | 6 | 582 | 2 | 4 | 6 | 614 | 2 | 6 | 6 |
| 519 | 2 | 0 | 7 | 551 | 2 | 2 | 7 | 583 | 2 | 4 | 7 | 615 | 2 | 6 | 7 |
| 520 | 2 | 0 | 8 | 552 | 2 | 2 | 8 | 584 | 2 | 4 | 8 | 616 | 2 | 6 | 8 |
| 521 | 2 | 0 | 9 | 553 | 2 | 2 | 9 | 585 | 2 | 4 | 9 | 617 | 2 | 6 | 9 |
| 522 | 2 | 0 | A | 554 | 2 | 2 | A | 586 | 2 | 4 | A | 618 | 2 | 6 | A |
| 523 | 2 | 0 | B | 555 | 2 | 2 | B | 587 | 2 | 4 | B | 619 | 2 | 6 | B |
| 524 | 2 | 0 | C | 556 | 2 | 2 | C | 588 | 2 | 4 | C | 620 | 2 | 6 | C |
| 525 | 2 | 0 | D | 557 | 2 | 2 | D | 589 | 2 | 4 | D | 621 | 2 | 6 | D |
| 526 | 2 | 0 | E | 558 | 2 | 2 | E | 590 | 2 | 4 | E | 622 | 2 | 6 | E |
| 527 | 2 | 0 | F | 559 | 2 | 2 | F | 591 | 2 | 4 | F | 623 | 2 | 6 | F |
| 528 | 2 | 1 | 0 | 560 | 2 | 3 | 0 | 592 | 2 | 5 | 0 | 624 | 2 | 7 | 0 |
| 529 | 2 | 1 | 1 | 561 | 2 | 3 | 1 | 593 | 2 | 5 | 1 | 625 | 2 | 7 | 1 |
| 530 | 2 | 1 | 2 | 562 | 2 | 3 | 2 | 594 | 2 | 5 | 2 | 626 | 2 | 7 | 2 |
| 531 | 2 | 1 | 3 | 563 | 2 | 3 | 3 | 595 | 2 | 5 | 3 | 627 | 2 | 7 | 3 |
| 532 | 2 | 1 | 4 | 564 | 2 | 3 | 4 | 596 | 2 | 5 | 4 | 628 | 2 | 7 | 4 |
| 533 | 2 | 1 | 5 | 565 | 2 | 3 | 5 | 597 | 2 | 5 | 5 | 629 | 2 | 7 | 5 |
| 534 | 2 | 1 | 6 | 566 | 2 | 3 | 6 | 598 | 2 | 5 | 6 | 630 | 2 | 7 | 6 |
| 535 | 2 | 1 | 7 | 567 | 2 | 3 | 7 | 599 | 2 | 5 | 7 | 631 | 2 | 7 | 7 |
| 536 | 2 | 1 | 8 | 568 | 2 | 3 | 8 | 600 | 2 | 5 | 8 | 632 | 2 | 7 | 8 |
| 537 | 2 | 1 | 9 | 569 | 2 | 3 | 9 | 601 | 2 | 5 | 9 | 633 | 2 | 7 | 9 |
| 538 | 2 | 1 | A | 570 | 2 | 3 | A | 602 | 2 | 5 | A | 634 | 2 | 7 | A |
| 539 | 2 | 1 | B | 571 | 2 | 3 | B | 603 | 2 | 5 | B | 635 | 2 | 7 | B |
| 540 | 2 | 1 | C | 572 | 2 | 3 | C | 604 | 2 | 5 | C | 636 | 2 | 7 | C |
| 541 | 2 | 1 | D | 573 | 2 | 3 | D | 605 | 2 | 5 | D | 637 | 2 | 7 | D |
| 542 | 2 | 1 | E | 574 | 2 | 3 | E | 606 | 2 | 5 | E | 638 | 2 | 7 | E |
| 543 | 2 | 1 | F | 575 | 2 | 3 | F | 607 | 2 | 5 | F | 639 | 2 | 7 | F |

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)

| OP CODE | A | B | FUNCTION | SUBJECT REFERENCE |
|----------------|----------|----------|---|--------------------------|
| 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.**

SERIES L/TC CHARACTER SETS

The USASCII and Commercial character sets for the Series L/TC Systems are listed below in their collating sequence in ascending order. Each character set consists of 64 graphic characters, the Space code, and the End of Alpha code. The USASCII character set consists of the USASCII characters in columns 2, 3, 4, and 5 of the USASCII table, plus End of Alpha (NUL) and Overline. Those Commercial characters that differ from the USASCII characters are shown in parentheses.

The internal or machine language code for each character is given; this code consists of two hexadecimal digits which correspond to the column and row number of the character in the USASCII table (A=row 10, B=11, C=12, D=13, E=14, F=15). In addition, the decimal value of each character is given as required when using Index Registers for modification.

| Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value | Character | Internal Code | Indexing Value |
|--------------------|---------------|----------------|-----------|---------------|----------------|-----------|---------------|----------------|-----------|---------------|----------------|
| End of Alpha (NUL) | 0 0 | 0 | | | | | | | | | |
| Space | 2 0 | 32 | 0 | 3 0 | 48 | @ | 4 0 | 64 | P | 5 0 | 80 |
| ! | 2 1 | 33 | 1 | 3 1 | 49 | A | 4 1 | 65 | Q | 5 1 | 81 |
| ” | 2 2 | 34 | 2 | 3 2 | 50 | B | 4 2 | 66 | R | 5 2 | 82 |
| # | 2 3 | 35 | 3 | 3 3 | 51 | C | 4 3 | 67 | S | 5 3 | 83 |
| \$ | 2 4 | 36 | 4 | 3 4 | 52 | D | 4 4 | 68 | T | 5 4 | 84 |
| % | 2 5 | 37 | 5 | 3 5 | 53 | E | 4 5 | 69 | U | 5 5 | 85 |
| & | 2 6 | 38 | 6 | 3 6 | 54 | F | 4 6 | 70 | V | 5 6 | 86 |
| ' | 2 7 | 39 | 7 | 3 7 | 55 | G | 4 7 | 71 | W | 5 7 | 87 |
| (| 2 8 | 40 | 8 | 3 8 | 56 | H | 4 8 | 72 | X | 5 8 | 88 |
|) | 2 9 | 41 | 9 | 3 9 | 57 | I | 4 9 | 73 | Y | 5 9 | 89 |
| * | 2 A | 42 | : | 3 A | 58 | J | 4 A | 74 | Z | 5 A | 90 |
| + | 2 B | 43 | ; | 3 B | 59 | K | 4 B | 75 | [(¾) | 5 B | 91 |
| , | 2 C | 44 | < (½) | 3 C | 60 | L | 4 C | 76 | \ (¢) | 5 C | 92 |
| - | 2 D | 45 | = | 3 D | 61 | M | 4 D | 77 |] (CR) | 5 D | 93 |
| . | 2 E | 46 | > (¼) | 3 E | 62 | N | 4 E | 78 | ^ (°) | 5 E | 94 |
| / | 2 F | 47 | ? | 3 F | 63 | O | 4 F | 79 | ~ (◊) | 5 F | 95 |
| | | | | | | | | | DEL | 7 F | 127 |

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: 1. P is set. 2. Accumulator digit not zero. 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 |

ERROR MESSAGES

| <u>MESSAGE</u> | <u>TOPIC REFERENCE</u> |
|--|----------------------------|
| Assembler III – B 3500 Error Messages and Warnings | 5.03.06 |
| A PARAMETER – MUST HAVE NUMERIC VALUE 1-4 | ” |
| B PARAMETER – MUST HAVE NUMERIC VALUE 0-255 | ” |
| A – ENTRY MUST BE NUMERIC AND LESS THAN 767 | ” |
| BACKWARD ORG NOT ALLOWED ON ASSEMBLER I | ” |
| CDB MUST BE FIRST INSTRUCTION IN DECK | ” |
| DEFT PARAMETERS MUST BE NUMERIC AND 0-15 | ” |
| EMPTY | ” |
| EMPTY FIELD – LENGTH FIELD | ” |
| HAS ALREADY BEEN ENTERED AS A SYMBOLIC IDENTIFIER | ” |
| HAS NOT BEEN ENTERED AS A LABEL | ” |
| ILLEGAL DIGIT ENTERED IN CODE INSTRUCTION | ” |
| ILLEGAL ENTRY IN INCREMENT FIELD | ” |
| ILLEGAL PARAMETER ENTRY IN . . . | ” |
| INCREMENT GREATER THAN 255 NOT ALLOWED ON ASSEMB I | ” |
| INVALID FLAG ENTRY PAIR OF - - | ” |
| INVALID NUM ENTRY | ” |
| . . . IS AN INVALID INSTRUCTION OP CODE | ” |
| . . . IS NOT A VALID MASK ENTRY | ” |
| LABEL MUST BEGIN WITH AN ALPHABETIC CHARACTER | ” |
| LABEL MUST NOT CONTAIN BLANK CHARACTER | ” |
| LABEL NOT AT START OF WORD | ” |
| MASK LENGTH RECALCULATED TO BE . . . | ” |
| MISSING SYMBOLIC LABEL | ” |
| . . . NOT A VALID OP CODE | ” |
| NUMBER OF LABELS EXCEEDS LABEL LIMIT WHEN USING ASSEMBLER I | ” |
| PARAMETER TOO LONG | ” |
| PREVIOUS CARD HAS INVALID FIELD LENGTH | ” |
| REGION MUST HAVE SIZE 1-255 | ” |
| SEQUENCE ERROR | ” |
| STORAGE EXCEEDED BY INSTRUCTION | ” |
| SUM OF CDF PARAMETERS MUST NOT BE > 80 | ” |
| SYMBOLIC ENTRY ALLOWED | ” |
| Assembler IV – B 5500 Error Messages and Warnings | 5.04.06 |
| The messages are identical to those printed by Assembler III. See above. | ” |
| Assembler V – B 300 Error Messages and Warnings | 5.05.07 |
| A-PAR LABEL INVALID | ” |
| A-PAR LABEL UNDEFINED | ” |
| ILLEGAL | ” |

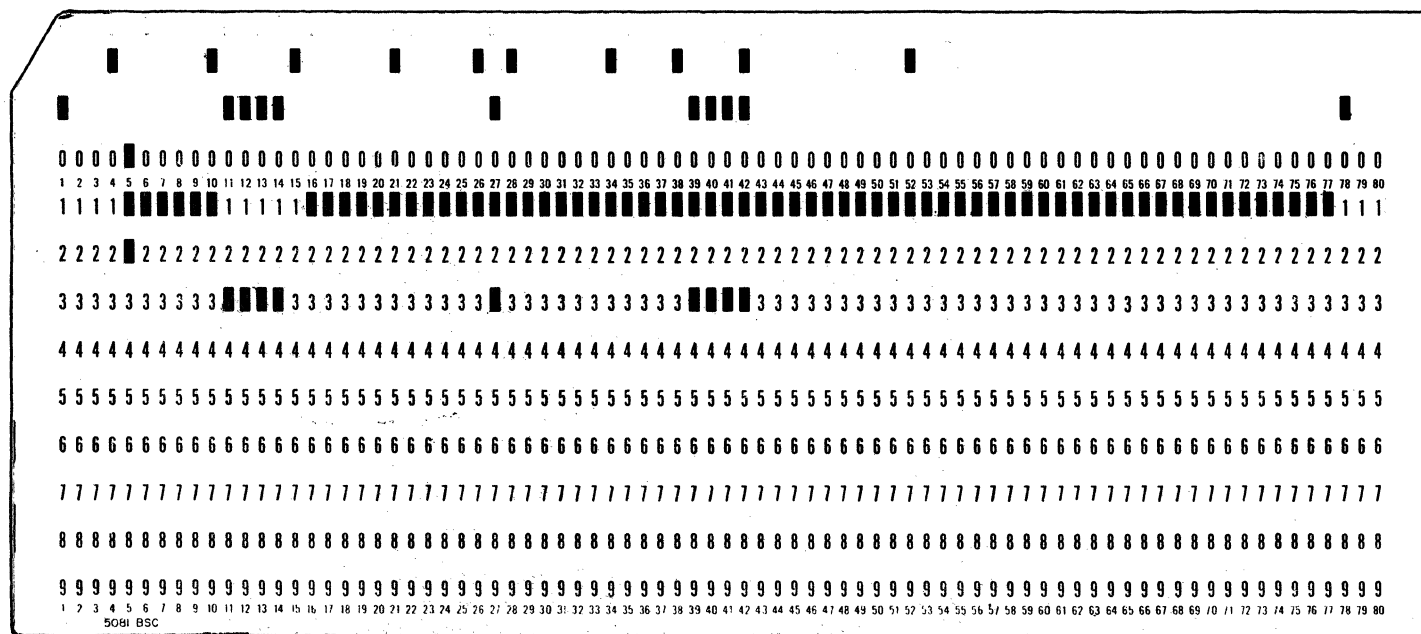
APPENDIX F (cont'd)

| <u>MESSAGE</u> | <u>TOPIC REFERENCE</u> |
|---------------------------------------|-----------------------------------|
| INSTR LABEL DUPLICATED | 5.05.07 |
| INSTR LABEL INVALID | ” |
| INVALID CHARACTER | ” |
| INVALID CTL CARD | ” |
| INVALID FLD LENGTH | ” |
| INVALID INCREMENT FLD | ” |
| INVALID MNEMONIC | ” |
| INVALID SIGN RESULT | ” |
| LABEL LIMIT | ” |
| LAST LINE NOT “END” | ” |
| MEMORY 512 | ” |
| MEMORY OVERLAYED ERR | ” |
| MUST HAVE LBL OR INC | ” |
| NO I/P CTL | ” |
| NO O/P SYM | ” |
| NO OBJECT | ” |
| NOP INSERTED | ” |
| . . . PARAMETER INVALID X | ” |
| . . . PARAMETER INVALID RANGE XXX-XXX | ” |
| PRE INV FLD LENGTH | ” |
| SEQUENCE | ” |
| STORAGE EXCEEDED | ” |
| WORD LENGTH EXCEEDED | ” |

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 Keypunch

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.

USASCII CHART AND CHARACTER SETS

| Bits | | | | | Column | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----|-----|----|---|---|---|-----|
| b ₇ | b ₆ | b ₅ | b ₄ | b ₃ | b ₂ | b ₁ | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| | | | | | Row | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | NUL | DLE | SP | @ | P | \ | p |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | SOH | DC1 | ! | A | Q | a | q |
| 0 | 0 | 1 | 0 | 0 | 2 | 2 | STX | DC2 | " | B | R | b | r |
| 0 | 0 | 1 | 1 | 0 | 3 | 3 | ETX | DC3 | # | C | S | c | s |
| 0 | 1 | 0 | 0 | 0 | 4 | 4 | EOT | DC4 | \$ | D | T | d | t |
| 0 | 1 | 0 | 1 | 0 | 5 | 5 | ENQ | NAK | % | E | U | e | u |
| 0 | 1 | 1 | 0 | 0 | 6 | 6 | ACK | SYN | & | F | V | f | v |
| 0 | 1 | 1 | 1 | 0 | 7 | 7 | BEL | ETB | ' | G | W | g | w |
| 1 | 0 | 0 | 0 | 0 | 8 | 8 | BS | CAN | (| H | X | h | x |
| 1 | 0 | 0 | 1 | 0 | 9 | 9 | HT | EM |) | I | Y | i | y |
| 1 | 0 | 1 | 0 | 0 | 10 | 10 | LF | SUB | * | J | Z | j | z |
| 1 | 0 | 1 | 1 | 0 | 11 | 11 | VT | ESC | + | K | [| k | { |
| 1 | 1 | 0 | 0 | 0 | 12 | 12 | FF | FS | , | L | \ | l | |
| 1 | 1 | 0 | 1 | 0 | 13 | 13 | CR | GS | - | M |] | m | } |
| 1 | 1 | 1 | 0 | 0 | 14 | 14 | SO | RS | . | N | ^ | n | ~ |
| 1 | 1 | 1 | 1 | 0 | 15 | 15 | SI | US | / | O | _ | o | DEL |

USA Standard Code for Information Interchange

| USASCII | | | | | EBCDIC | | | BCL | | | 80 COL CARD | | PAPER TAPE | | |
|--------------|-------------|-----|--------------|-----------------|-----------------|--------------|--------------------|-----|--------------|-----------|-------------|---------------|------------|-----------|-----|
| USASCII CHAR | INTRNL CODE | | L/TC GRAPHIC | PAPER TAPE CODE | INDEX REG VALUE | GRAPHIC CHAR | CARD CODE | KEY | GRAPHIC CHAR | CARD CODE | KEY | INTRNL BUFFER | | TAPE CODE | |
| | UP | LOW | | | | | | | | | | UP | LOW | UP | LOW |
| NUL | 0 | 0 | | | 0 | NUL | 12 - 0 - 9 - 8 - 1 | M | | | | | | 0 | 0 |
| SOH | 0 | 1 | | • | 1 | SOH | 12 - - 9 - - 1 | M | | | | | | 8 | 1 |
| STX | 0 | 2 | | • | 2 | STX | 12 - - 9 - - 2 | M | | | | | | 8 | 2 |
| ETX | 0 | 3 | | • | 3 | ETX | 12 - - 9 - - 3 | M | | | | | | 0 | 3 |
| EOT | 0 | 4 | | • | 4 | EOT | 9 - 7 | M | | | | | | 8 | 4 |
| ENQ | 0 | 5 | | • | 5 | ENQ | 0 - 9 - 8-5 | M | | | | | | 0 | 5 |
| ACK | 0 | 6 | | • | 6 | ACK | 0 - 9 - 8-6 | M | | | | | | 0 | 6 |
| BEL | 0 | 7 | | • | 7 | BEL | 0 - 9 - 8-7 | M | | | | | | 8 | 7 |
| BS | 0 | 8 | | • | 8 | BS | 11 - 9 - 6 | M | | | | | | 8 | 8 |
| HT | 0 | 9 | | • | 9 | HT | 12 - 9 - 5 | M | | | | | | 0 | 9 |
| IF | 0 | A | | • | 10 | IF | 0 - 9 - 5 | M | | | | | | 0 | A |
| VT | 0 | B | | • | 11 | VT | 12 - 9 - 8-3 | M | | | | | | 8 | B |
| FF | 0 | C | | • | 12 | FF | 12 - 9 - 8-4 | M | | | | | | 0 | C |
| CR | 0 | D | | • | 13 | CR | 12 - 9 - 8-5 | M | | | | | | 8 | D |
| SO | 0 | E | | • | 14 | SO | 12 - 9 - 8-6 | M | | | | | | 8 | E |
| SI | 0 | F | | • | 15 | SI | 12 - 9 - 8-7 | M | | | | | | 0 | F |
| DLE | 1 | 0 | | • | 16 | DLE | 12-11-9-8-1 | M | | | | | | 9 | 0 |
| DC1 | 1 | 1 | | • | 17 | DC1 | 11 - - 9-1 | M | | | | | | 1 | 1 |
| DC2 | 1 | 2 | | • | 18 | DC2 | 11 - - 9-2 | M | | | | | | 1 | 2 |
| DC3 | 1 | 3 | | • | 19 | DC3 | 11 - 9-3 | M | | | | | | 9 | 3 |
| DC4 | 1 | 4 | | • | 20 | DC4 | 9-8-4 | M | | | | | | 1 | 4 |
| NAK | 1 | 5 | | • | 21 | NAK | 9-8-5 | M | | | | | | 9 | 5 |
| SYN | 1 | 6 | | • | 22 | SYN | 9-2 | M | | | | | | 9 | 6 |
| ETB | 1 | 7 | | • | 23 | ETB | 0 - 9-6 | M | | | | | | 1 | 7 |
| CAN | 1 | 8 | | • | 24 | CAN | 11 - 9-8 | M | | | | | | 1 | 8 |
| EM | 1 | 9 | | • | 25 | EM | 11 - 9-8-1 | M | | | | | | 9 | 9 |
| SUB | 1 | A | | • | 26 | SUB | 9-8-7 | M | | | | | | 9 | A |
| ESC | 1 | B | | • | 27 | ESC | 0 - 9-7 | M | | | | | | 1 | B |
| FS | 1 | C | | • | 28 | FS | 11 - 9-8-4 | M | | | | | | 9 | C |
| GS | 1 | D | | • | 29 | GS | 11 - 9-8-5 | M | | | | | | 1 | D |
| RS | 1 | E | | • | 30 | RS | 11 - 9-8-6 | M | | | | | | 1 | E |
| US | 1 | F | | • | 31 | US | 11 - 9-8-7 | M | | | | | | 9 | F |

APPENDIX H (cont'd)

CHARACTER SETS

| USASCII | | | | | EBCDIC | | | BCL | | | BUFFER | | TAPE | | |
|-----------------|----------------|-----|-----------------|--------------------|-----------------------|-----------------|--------------|-------------|-----------------|-----------|-------------|----|------|----|-----|
| USASCII CHAR | INTRNL CODE | | L/TC GRAPHIC | PAPER TAPE CODE | INDEX REG VALUE | GRAPHIC CHAR | CARD CODE | K E Y | GRAPHIC CHAR | CARD CODE | K E Y | UP | LOW | UP | LOW |
| | UP | LOW | | | | | | | | | | | | | |
| SP | 2 | 0 | SP | • • • | 32 | SP | BLANK | * | SP | BLANK | * | 0 | 0 | A | 0 |
| ! | 2 | 1 | ! | • • • • • | 33 | ! | 11-0 | I | X | 11-0 | M | 6 | 0 | 2 | 1 |
| ” | 2 | 2 | ” | • • • • • | 34 | ” | 8-7 | # | ” | 0-8-7 | M | 2 | F | 2 | 2 |
| # | 2 | 3 | # | • • • • • | 35 | # | 8-3 | * | # | 8-3 | * | 0 | B | A | 3 |
| \$ | 2 | 4 | \$ | • • • • • | 36 | \$ | 11-8-3 | * | \$ | 11-8-3 | * | 4 | B | 2 | 4 |
| % | 2 | 5 | % | • • • • • | 37 | % | 0-8-4 | * | % | 0-8-4 | * | 2 | C | A | 5 |
| & | 2 | 6 | & | • • • • • | 38 | & | 12 | * | & | 12 | * | 8 | 0 | A | 6 |
| , | 2 | 7 | , | • • • • • | 39 | , | 8-5 | # | ≥ | 8-7 | M | 0 | F | 2 | 7 |
| (| 2 | 8 | (| • • • • • | 40 | (| 12-8-5 | # | (| 12-8-5 | # | 8 | D | 2 | 8 |
|) | 2 | 9 |) | • • • • • | 41 |) | 11-8-5 | # |) | 11-8-5 | # | 4 | D | A | 9 |
| * | 2 | A | * | • • • • • | 42 | * | 11-8-4 | * | * | 11-8-4 | * | 4 | C | A | A |
| + | 2 | B | + | • • • • • | 43 | + | 12-0 | I | + | 12-0 | I | A | 0 | 2 | B |
| , | 2 | C | , | • • • • • | 44 | , | 0-8-3 | * | , | 0-8-3 | * | 2 | B | A | C |
| - | 2 | D | - | • • • • • | 45 | - | 11 | * | - | 11 | * | 4 | 0 | 2 | D |
| . | 2 | E | . | • • • • • | 46 | . | 12-8-3 | * | . | 12-8-3 | * | 8 | B | 2 | E |
| / | 2 | F | / | • • • • • | 47 | / | 0-1 | * | / | 0-1 | * | 2 | 1 | A | F |
| 0 | 3 | 0 | 0 | • • • • • | 48 | 0 | 0 | * | 0 | 0 | * | 2 | 0 | 3 | 0 |
| 1 | 3 | 1 | 1 | • • • • • | 49 | 1 | 1 | * | 1 | 1 | * | 0 | 1 | B | 1 |
| 2 | 3 | 2 | 2 | • • • • • | 50 | 2 | 2 | * | 2 | 2 | * | 0 | 2 | B | 2 |
| 3 | 3 | 3 | 3 | • • • • • | 51 | 3 | 3 | * | 3 | 3 | * | 0 | 3 | 3 | 3 |
| 4 | 3 | 4 | 4 | • • • • • | 52 | 4 | 4 | * | 4 | 4 | * | 0 | 4 | B | 4 |
| 5 | 3 | 5 | 5 | • • • • • | 53 | 5 | 5 | * | 5 | 5 | * | 0 | 5 | 3 | 5 |
| 6 | 3 | 6 | 6 | • • • • • | 54 | 6 | 6 | * | 6 | 6 | * | 0 | 6 | 3 | 6 |
| 7 | 3 | 7 | 7 | • • • • • | 55 | 7 | 7 | * | 7 | 7 | * | 0 | 7 | B | 7 |
| 8 | 3 | 8 | 8 | • • • • • | 56 | 8 | 8 | * | 8 | 8 | * | 0 | 8 | B | 8 |
| 9 | 3 | 9 | 9 | • • • • • | 57 | 9 | 9 | * | 9 | 9 | * | 1 | 0 | 3 | 9 |
| : | 3 | A | : | • • • • • | 58 | : | 8-2 | # | : | 8-5 | M | 0 | D | 3 | A |
| ; | 3 | B | ; | • • • • • | 59 | ; | 11-8-6 | # | ; | 11-8-6 | # | 4 | E | B | B |
| < | 3 | C | < or ¼ | • • • • • | 60 | < | 12-8-4 | # | < | 12-8-6 | M | 8 | E | 3 | C |
| = | 3 | D | = | • • • • • | 61 | = | 8-6 | # | = | 0-8-5 | M | 2 | D | B | D |
| > | 3 | E | > or ¼ | • • • • • | 62 | > | 0-8-6 | # | > | 8-6 | M | 0 | E | B | E |
| ? | 3 | F | ? | • • • • • | 63 | ? | 0-8-7 | # | ? | ALL OTHER | M | | | 3 | F |

Key: * Keys on 026, 029 A 149 and A 150 Punch Correct Code.
M Multipunch on 026, 029, A 149 and A 150.

Keys on 029, and A 150 punch correct code; multipunch on 026 and A 149.
I Keys on 029 and A 150 punch invalid code; multipunch on 026, 029, A 149 and A 150.

| USASCII | | | | | EBCDIC | | | BCL | | | BUFFER | | TAPE | | |
|-----------------|----------------|-----|--------------------|--------------------|-----------------------|-----------------|-----------|-------------|-----------------|-----------|-------------|----|------|----|-----|
| USASCII CHAR | INTRNL CODE | | L/TC GRAPHIC | PAPER TAPE CODE | INDEX REG VALUE | GRAPHIC CHAR | CARD CODE | K E Y | GRAPHIC CHAR | CARD CODE | K E Y | UP | LOW | UP | LOW |
| | UP | LOW | | | | | | | | | | UP | LOW | UP | LOW |
| @ | 4 | 0 | @ | ●● . . . | 64 | @ | 8-4 | * | @ | 8-4 | * | 0 | C | C | 0 |
| A | 4 | 1 | A | ● | 65 | A | 12-1 | * | A | 12-1 | * | 8 | 1 | 4 | 1 |
| B | 4 | 2 | B | ● | 66 | B | 12-2 | * | B | 12-2 | * | 8 | 2 | 4 | 2 |
| C | 4 | 3 | C | ●● . . . | 67 | C | 12-3 | * | C | 12-3 | * | 8 | 3 | C | 3 |
| D | 4 | 4 | D | ● | 68 | D | 12-4 | * | D | 12-4 | * | 8 | 4 | 4 | 4 |
| E | 4 | 5 | E | ●● . . . | 69 | E | 12-5 | * | E | 12-5 | * | 8 | 5 | C | 5 |
| F | 4 | 6 | F | ●● . . . | 70 | F | 12-6 | * | F | 12-6 | * | 8 | 6 | C | 6 |
| G | 4 | 7 | G | ● | 71 | G | 12-7 | * | G | 12-7 | * | 8 | 7 | 4 | 7 |
| H | 4 | 8 | H | ● | 72 | H | 12-8 | * | H | 12-8 | * | 8 | 8 | 4 | 8 |
| I | 4 | 9 | I | ●● . . . | 73 | I | 12-9 | * | I | 12-9 | * | 9 | 0 | C | 9 |
| J | 4 | A | J | ●● . . . | 74 | J | 11-1 | * | J | 11-1 | * | 4 | 1 | C | A |
| K | 4 | B | K | ● | 75 | K | 11-2 | * | K | 11-2 | * | 4 | 2 | 4 | B |
| L | 4 | C | L | ●● . . . | 76 | L | 11-3 | * | L | 11-3 | * | 4 | 3 | C | C |
| M | 4 | D | M | ● | 77 | M | 11-4 | * | M | 11-4 | * | 4 | 4 | 4 | D |
| N | 4 | E | N | ● | 78 | N | 11-5 | * | N | 11-5 | * | 4 | 5 | 4 | E |
| O | 4 | F | O | ●● . . . | 79 | O | 11-6 | * | O | 11-6 | * | 4 | 6 | C | F |
| P | 5 | 0 | P | ● | 80 | P | 11-7 | * | P | 11-7 | * | 4 | 7 | 5 | 0 |
| Q | 5 | 1 | Q | ●● . . . | 81 | Q | 11-8 | * | Q | 11-8 | * | 4 | 8 | D | 1 |
| R | 5 | 2 | R | ●● . . . | 82 | R | 11-9 | * | R | 11-9 | * | 5 | 0 | D | 2 |
| S | 5 | 3 | S | ● | 83 | S | 0-2 | * | S | 0-2 | * | 2 | 2 | 5 | 3 |
| T | 5 | 4 | T | ●● . . . | 84 | T | 0-3 | * | T | 0-3 | * | 2 | 3 | D | 4 |
| U | 5 | 5 | U | ● | 85 | U | 0-4 | * | U | 0-4 | * | 2 | 4 | 5 | 5 |
| V | 5 | 6 | V | ● | 86 | V | 0-5 | * | V | 0-5 | * | 2 | 5 | 5 | 6 |
| W | 5 | 7 | W | ●● . . . | 87 | W | 0-6 | * | W | 0-6 | * | 2 | 6 | D | 7 |
| X | 5 | 8 | X | ●● . . . | 88 | X | 0-7 | * | X | 0-7 | * | 2 | 7 | D | 8 |
| Y | 5 | 9 | Y | ● | 89 | Y | 0-8 | * | Y | 0-8 | * | 2 | 8 | 5 | 9 |
| Z | 5 | A | Z | ● | 90 | Z | 0-9 | * | Z | 0-9 | * | 3 | 0 | 5 | A |
| [| 5 | B | [or $\frac{3}{4}$ | ●● . . . | 91 | [| 12-8-2 | M | | 12-8-4 | M | 8 | C | D | B |
| \ | 5 | C | \ or ϕ | ● | 92 | \ | 11-8-7 | # | \ | 11-8-7 | M | | | 5 | C |
|] | 5 | D |] or cr | ●● . . . | 93 |] | 11-8-2 | M | | 0-8-6 | M | 2 | E | D | D |
| ^ | 5 | E | ^ or ° | ●● . . . | 94 | + | 12-8-6 | # | | | | | | D | E |
| - | 5 | F | - | ● | 95 | - | 0-8-5 | # | - | 0-8-2 | M | 2 | A | 5 | F |

TABLE OF INPUT CODE ASSIGNMENTS

A Table of Input Code Assignments provides the means by which any type of paper tape code (BCL, etc.) may be read and interpreted into the Series L/TC internal code (USASCII). The table not only permits any type of code (from any 5, 6, 7, or 8 channel tape), but also enables assigning any desired character or certain functions to be interpreted from a particular code. Tables are available for such common code sets as BCL, IBM 046, Friden and 5 Channel Teletype (Baudot); however, any other code set (up to 8 channels or bits) may be incorporated.

Input tape that contains USASCII code does not require a table for conversion, but may use a table if special functions are desired from certain codes.

The conversion table, where required, is stored in the Normal (or user) area of memory and occupies up to 16 words. Each code (character) in the tape is represented by a pattern of punches in one position (or frame) which constitutes a unique configuration of "bits." As codes are read from tape, each code references its own character position in the conversion table based on its "bit" configuration. In other words, the bit configuration of the code serves as an "address" to a specific position in the table. The way in which that code is interpreted is determined by the internal code value that the programmer has placed in that position of the table. The tables available represent "standard" interpretations of characters and functional codes. The internal code representing an input code may be changed in the table to suit a user's particular need and give any desired interpretation as outlined in the following paragraphs.

INPUT FUNCTIONS FOR 6, 7, or 8 CHANNEL TAPE BASED ON THE TABLE OF CODE ASSIGNMENTS

An input code is interpreted in any one of the following ways depending on the internal code placed in its position in the table (does not apply to 5 channel code):

1. Interprets the incoming code as one of the Series L/TC printable (graphic) characters when the internal code for that character is contained in that position of the table.
2. Ignores the incoming code when the Series L/TC internal code for Ignore is contained in that position of the table.
3. Interprets the incoming code as an invalid character when a forced parity error is contained in that position of the table. This turns on the Invalid Code Indicator Light and sets Reader Invalid Code flag.
4. Causes the incoming code to set any or all of the flags of one flag group (the Y or K flag groups). The flags can then be tested as part of the user program, to provide alternate results. Codes that set the Y or K flags also terminate the read instruction. Incoming codes interpreted in this manner serve as Field Identifier codes and do not provide a printable Series L/TC character.
5. Causes the incoming code to set any or all of the Accumulator (A) Flags during a Read Numeric instruction. This permits numeric data to be read as minus and/or identified uniquely (as per hundred, etc.). The flags can then be tested to cause alternate results as part of the user program. Codes that set the Accumulator Flags do not terminate the Read instruction; therefore, they can be located in any character position in the data field on the tape. They do not provide a printable character during read-in, but as in the case of the Sign flag, subsequent Print Numeric instructions can be affected.

The codes described in paragraphs 4 and 5 above may or may not correspond to those codes that are normally considered "control" or "functional" in a given code set, depending on the interpretation value given to them in the table by the programmer.

APPENDIX I (cont'd)

FIRMWARE SUBSETS FOR THE TABLE OF CODE ASSIGNMENTS

Specific GP 300 Firmware subsets are provided with paper tape input/output capability. However, the Table of Code Assignments is usually loaded into memory as part of the user program load procedure. This permits using various code sets at different times with the same user program, or permits use of a different code set with each separate user program without changing the firmware, with certain exceptions:

1. Input with any code set requiring conversion to the internal code (USASCII), with a table of code assignments, requires a firmware subset that provides "table look-up." Input with USASCII does not require "table look-up" firmware since no conversion is necessary. However, various code sets can be used as input to the same system, along with USASCII, so long as "table look-up" firmware is used and a table of code assignments is provided for each code set including USASCII.
2. Firmware for 5 channel code includes "table look-up" capability; however, it is different than firmware for 8, 7, or 6 channel code, or for USASCII (no table look-up).

USASCII PAPER TAPE CODE WITHOUT TABLE LOOK-UP FIRMWARE

When USASCII is the paper tape input code, a table of code assignments is not required, and a separate Firmware subset is provided.

The following chart shows the code that represents each of the USASCII characters on tape (even parity). Each character is represented by two hexadecimal digits: the left for the upper four bits, the right for the lower four bits. As the tape is read and after parity checking, the parity bit (b8) is set to zero before the character is stored in memory; therefore, if a tape code's upper four bits are A, B, C, or D, they would become 2, 3, 4, or 5 memory respectively. (See Appendix D.)

| | | | | | | | |
|----|------|-------|------|-----|------|---------|------|
| Sp | A, 0 | 0 | 3, 0 | @ | C, 0 | P | 5, 0 |
| ! | 2, 1 | 1 | B, 1 | A | 4, 1 | Q | D, 1 |
| " | 2, 2 | 2 | B, 2 | B | 4, 2 | R | D, 2 |
| # | A, 3 | 3 | 3, 3 | C | C, 3 | S | 5, 3 |
| \$ | 2, 4 | 4 | B, 4 | D | 4, 4 | T | D, 4 |
| % | A, 5 | 5 | 3, 5 | E | C, 5 | U | 5, 5 |
| & | A, 6 | 6 | 3, 6 | F | C, 6 | V | 5, 6 |
| ' | 2, 7 | 7 | B, 7 | G | 4, 7 | W | D, 7 |
| (| 2, 8 | 8 | B, 8 | H | 4, 8 | X | D, 8 |
|) | A, 9 | 9 | 3, 9 | I | C, 9 | Y | 5, 9 |
| * | A, A | : | 3, A | J | C, A | Z | 5, A |
| + | 2, B | ; | B, B | K | 4, B | 3/4 (I) | D, B |
| , | A, C | ½ (<) | 3, C | L | C, C | ¢ (\) | 5, C |
| - | 2, D | = | B, D | M | 4, D | CR (J) | D, D |
| . | 2, E | ¼ (>) | B, E | N | 4, E | ° (^) | D, E |
| / | A, F | ? | 3, F | O | C, F | — | 5, F |
| | | | | DEL | F, F | <> (~) | 7, E |

FIELD IDENTIFIER (TERMINATION) CODES: The following chart shows the paper tape USASCII control codes which cause tape read instructions to be terminated, and some of which set a specified flag pattern. Each code is represented by two hexadecimal digits. Codes in column 1 of the table set the "K" flags. These codes do not enter into memory.

| USASCII COLUMN 0 FIELD IDENTIFIER CODES** | | | | | |
|---|------------------|---------------------------|---|---|---|
| CODE | PAPER TAPE VALUE | FLAG PATTERN SET BY CODE* | | | |
| | | Y FLAG NUMBER | | | |
| | | 3 | 2 | 1 | 4 |
| NUL | 0,0 | 0 | 0 | 0 | 0 |
| SOH | 8,1 | 0 | 0 | 0 | 1 |
| STX | 8,2 | 0 | 0 | 1 | 0 |
| ETX | 0,3 | 0 | 0 | 1 | 1 |
| EOT | 8,4 | 0 | 1 | 0 | 0 |
| ENQ | 0,5 | 0 | 1 | 0 | 1 |
| ACK | 0,6 | 0 | 1 | 1 | 0 |
| BEL | 8,7 | 0 | 1 | 1 | 1 |
| BS | 8,8 | 1 | 0 | 0 | 0 |
| HT | 0,9 | 1 | 0 | 0 | 1 |
| IF | 0,A | 1 | 0 | 1 | 0 |
| VT | 8,B | 1 | 0 | 1 | 1 |
| FF | 0,C | 1 | 1 | 0 | 0 |
| CR | 8,D | 1 | 1 | 0 | 1 |
| SO | 8,E | 1 | 1 | 1 | 0 |
| SI | 0,F | 1 | 1 | 1 | 1 |

| USASCII COLUMN 1 FIELD IDENTIFIER CODES | | | | | |
|---|--------------------------|---------------------------|---|---|---|
| CODE | PAPER TAPE VALUE a, b | FLAG PATTERN SET BY CODE* | | | |
| | | OCK FLAG NUMBER | | | |
| | | 3 | 2 | 1 | 4 |
| DLE | 9,0 | 0 | 0 | 0 | 0 |
| DC1 | 1,1 | 0 | 0 | 0 | 1 |
| DC2 | 1,2 | 0 | 0 | 1 | 0 |
| DC3 | 9,3 | 0 | 0 | 1 | 1 |
| DC4 | 1,4 | 0 | 1 | 0 | 0 |
| NAK | 9,5 | 0 | 1 | 0 | 1 |
| SYN | 9,6 | 0 | 1 | 1 | 0 |
| ETB | 1,7 | 0 | 1 | 1 | 1 |
| CAN | 1,8 | 1 | 0 | 0 | 0 |
| EM | 9,9 | 1 | 0 | 0 | 1 |
| SUB | 9,A | 1 | 0 | 1 | 0 |
| ESC | 1,B | 1 | 0 | 1 | 1 |
| FS | 9,C | 1 | 1 | 0 | 0 |
| GS | 1,D | 1 | 1 | 0 | 1 |
| RS | 1,E | 1 | 1 | 1 | 0 |
| US | 9,F | 1 | 1 | 1 | 1 |

*0 = flag is reset; 1 = flag is set

**Firmware sets are available which prevent the setting of the Y flags when these codes are read.

The NUL code is the same as a sprocket feed code in that no channels are punched in a frame, and thus, it functions differently than the other field identifier codes. During a read tape instruction, it is ignored (treated like a delete code – DEL) until the first significant character of data is read. If encountered after the first significant character, it will then be treated as a field identifier code and will terminate the read instruction. It should not be used for a field identifier code if a variable field of data would ever contain no significant data but only a field identifier code. This would cause the NUL code to be ignored, since a significant character was not read, and it would not serve its intended function to terminate the instruction. This would result in the paper tape getting out of step with the program.

The END OF ALPHA (code 0,0) is the same as the NUL code.

The DEL (Delete) code is completely ignored by all paper tape read instructions, and does not count as a character read. It consists of a punch in all 8 channels in a frame of tape.

APPENDIX I (cont'd)

| TAPE CODES | | | | ACCUMULATOR FLAGS* | | | |
|------------|-----|-----|---------|-----------------------|---|---|---|
| | | | | M | C | S | - |
| A,0 | | C,0 | 5,0 | 0 | 0 | 0 | 0 |
| 2,1 | | 4,1 | D,1 | 0 | 0 | 0 | 1 |
| 2,2 | | 4,2 | D,2 | 0 | 0 | 1 | 0 |
| A,3 | | C,3 | 5,3 | 0 | 0 | 1 | 1 |
| 2,4 | | 4,4 | D,4 | 0 | 1 | 0 | 0 |
| A,5 | | C,5 | 5,5 | 0 | 1 | 0 | 1 |
| A,6 | | C,6 | 5,6 | 0 | 1 | 1 | 0 |
| 2,7 | | 4,7 | D,7 | 0 | 1 | 1 | 1 |
| 2,8 | | 4,8 | D,8 | 1 | 0 | 0 | 0 |
| A,9 | | C,9 | 5,9 | 1 | 0 | 0 | 1 |
| A,A | 3,A | C,A | 5,A | 1 | 0 | 1 | 0 |
| 2,B | B,B | 4,B | D,B | 1 | 0 | 1 | 1 |
| A,C | 3,C | C,C | 5,C | 1 | 1 | 0 | 0 |
| 2,D | B,D | 4,D | D,D | 1 | 1 | 0 | 1 |
| 2,E | B,E | 4,E | D,E 7,E | 1 | 1 | 1 | 0 |
| A,F | 3,F | C,F | 5,F | 1 | 1 | 1 | 1 |

TABLE OF OUTPUT CODE ASSIGNMENTS

When a code set other than USASCII is desired in the output tape, or when certain variations may be desired in the USASCII set, a Table of Output Code Assignments may be used. This permits output into any 5, 6, 7, or 8 channel code without modification to the Perforator. Output in USASCII code does not require a table.

The table is loaded into a Normal memory area and occupies up to 16 words. The loading may accompany regular loading of user programs. This table is a separate table from the Table of Input Code Assignments described above. Each Series L/TC internal character selects a particular character position in the output table. The 8-bit code that is put in each character position of the table is the code that will be punched into the output tape.

Normally, the Punch Code (XC) instruction will be used to punch field identifier (functional) codes. However, since any of the Series L/TC internal characters, through the table, can cause any 8-bit code to be punched, field identifier codes may be punched in this manner also.

The programmer may construct an output table to achieve any desired output code. However, tables are available that contain "standard" values for the following code sets:

| | |
|----------|--------------------|
| BCL/IBM | 8 channel |
| Friden | 8 channel |
| USASCII | 8 channel |
| Teletype | 5 channel (Baudot) |

The bit configuration of most Friden tape codes is the same as BCL. However, many of the functional code names given to the various codes are different, and for that reason a table is provided for ease in interpretation.

FIRMWARE SUBSETS FOR THE TABLE OF CODE ASSIGNMENTS

The firmware which includes "table look-up" for conversion of the internal code to the output code is different than firmware which does not use "table look-up" (output in USASCII). Thus, a USASCII table is available for use in systems that require "table look-up" firmware due to varying output code requirements.

NOTE: Output in 5-channel tape code requires firmware that is different from either 8-channel "table look-up" firmware or for output in USASCII without "table look-up."

GP 300 TIMINGS

This section contains the timings for GP 300 instructions as recorded by tests on Firmware Set 2-1002-001-02. The timings are averages depending upon two factors:

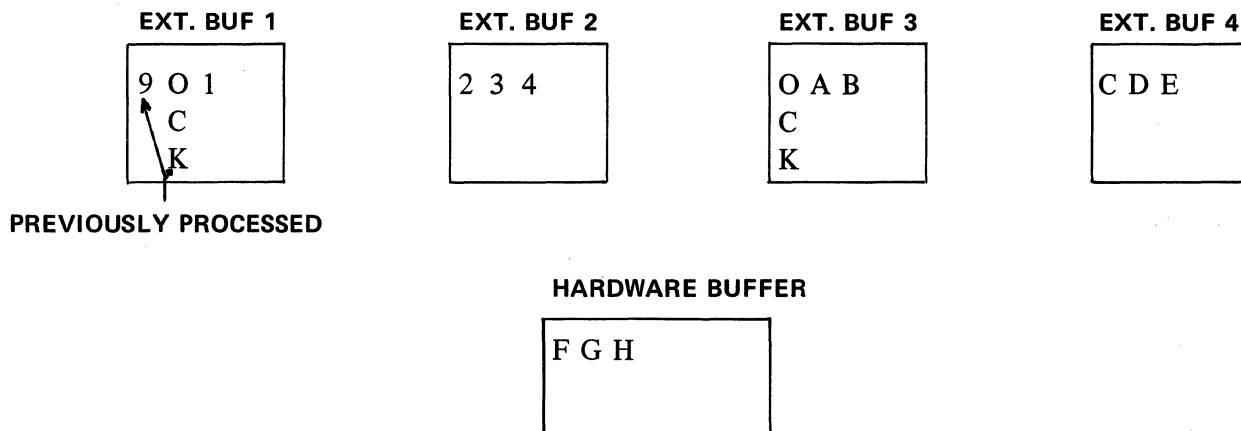
1. The Firmware Set being used.
2. The context in which the particular macro instruction appears, in particular fetch time and other disk position considerations.

TIMINGS

The following instruction timings were measured on firmware set 2-1002-001-02, with the instruction placed in syllable zero. When an instruction is placed in syllable three (3) or the execution of the instruction causes a word boundary to be crossed, an additional 10 to 20 milliseconds will be required for a new instruction word access.

When the keyboard extension buffers are full and the hardware buffer (A3) contains at least three (3) characters an additional 10 ms must be added to every macro instruction.

This condition could occur with 13 entries in the buffers. The following is a typical example of this condition.



The 9 and one OCK were processed on the last keyboard instruction. Extension buffer 1 now contains 1 entry to be processed. Extension buffers 2, 3, and 4 each contain 3 entries.

When the hardware buffer receives 3 or more entries an attempt will be made to unload the hardware buffer. Since the extension buffers are full the unloading cannot take place. Thus, an additional 10 ms cycle will occur with each new instruction fetched, until at least one buffer is emptied and the hardware buffer is unloaded.

| <u>MACRO INSTRUCTION</u> | <u>EXECUTION TIME IN MS</u> |
|--------------------------|-----------------------------|
| ADA | 50 |
| ADIR | 20 |
| ADK | 30 |
| ADM | 70 |
| BRU | 20 |
| CHG | 20 |

APPENDIX J (cont'd)

MACRO INSTRUCTION

EXECUTION TIME IN MS

| | | |
|-------|---------------|---------------|
| CLA | | 20 |
| CLM | | 40 |
| CPA | | 60 |
| DIR | | 20 |
| DIV | | * SEE NOTE 2 |
| EX | | 20 |
| EXE | | 20 |
| EXL | | 20 |
| EXZ | | 20 |
| IIR | | 20 |
| INK | | 20 |
| LIR | | 20 |
| LKBR | | 20 |
| LLCR | | 30 |
| LLLR | | 30 |
| LOD | | 30 |
| LPKR | | 30 |
| LPNR | | 30 |
| LRCR | | 30 |
| LRLR | | 30 |
| MOD | | 30 |
| MUL | | * SEE NOTE 3 |
| MULR | | * SEE NOTE 3 |
| NOP | | 10 |
| PKA | | 20 |
| PKB | | 20 |
| POS | | 10 SEE NOTE 4 |
| REM | | 30 |
| RR | | 20 |
| RST | | 20 |
| SET | | 20 |
| SK | | 20 |
| SKE | | 20 |
| SKL | | 20 |
| SKZ | | 20 |
| SLRO | | * SEE NOTE 1 |
| SLROS | | * SEE NOTE 1 |
| SRJ | | 40 |
| SRR | Stack address | 0 = 50 |
| | Stack address | 1 = 60 |
| | Stack address | 2 = 70 |
| | Stack address | 3 = 80 |
| SUA | | 50 |
| SUK | | 30 |
| SUM | | 70 |
| TAIR | | 50 |
| TRA | | 30 |
| TRM | | 30 |

The following instructions are variable, since mechanical synchronization is required. These may vary from machine to machine. The minimum and maximum execution times shown were measured on machine serial #Q1001P.

| <u>MACRO INSTRUCTION</u> | <u>FORMS COMMANDS</u> | <u>EXECUTION TIME IN MS</u> |
|--------------------------|-----------------------|-----------------------------|
| ALARM | | 20-50 |
| AL | 1 LINE | 60-80 |
| | 10 LINES | 480-520 |
| AR | 1 LINE | 60-80 |
| | 10 LINES | 480-520 |
| ALR | 1 LINE | 60-80 |
| ALTO | 10 LINES | 550-590 |
| ARTO | 10 LINES | 550-590 |
| CC | | 210-280 |
| OC | | 210-280 |

PRINT COMMANDS *SEE NOTE 4

| | | |
|-----|--|-------|
| PA | Per character | 50 ms |
| PC | | 30-70 |
| PC+ | a) If no print occurs | 10 |
| PC- | b) If print occurs | 30-70 |
| PN | a) First digit or punctuation printed is 90-120 ms; all succeeding places are 50 ms each | |
| | b) 10 ms per digit or punctuation suppressed. | |

An additional 250 ms will be required for the first printing instruction encountered after exiting the READY MODE unless the carriage is closed prior to executing one of the following instructions.

| | |
|-----|------|
| PA | PN |
| PC | PNS+ |
| PC+ | PNS- |
| PC- | TK |
| PCP | TKM |

KEYBOARD COMMANDS

NK

| | |
|-----|--------------|
| 40 | ms per digit |
| 80 | ms per OCK |
| 110 | ms per PSK |

*SEE NOTE 5

*SEE NOTE 6

APPENDIX J (cont'd)

TK

| | | |
|-----------|---|-------------------------------------|
| BASE | = | 20-50 ms (due to mechanical Timing) |
| CHARACTER | = | 50 ms |
| OCK | = | 100 ms – from hardware buffer |
| OCK | = | 110 ms – from extension buffer |
| PSK | = | 130 ms – from hardware buffer |
| PSK | = | 140 ms – from extension buffer |

*SEE NOTE 5

*SEE NOTE 6

TKM

| | | |
|-----------|---|--------------------------------|
| BASE | = | 20-50 ms |
| CHARACTER | = | 50 ms |
| OCK | = | 100 ms – from hardware buffer |
| OCK | = | 110 ms – from extension buffer |
| PSK | = | 130 ms – from hardware buffer |
| PSK | = | 140 ms – from extension buffer |

*SEE NOTE 5

*SEE NOTE 6

30 ms required to load each word to memory. This will occur every 8 characters or upon receiving a termination code.

EAM

| | | |
|-----------|---|--------------------------------|
| CHARACTER | = | 40 ms |
| OCK | = | 100 ms – from hardware buffer |
| OCK | = | 110 ms – from extension buffer |
| PSK | = | 130 ms – from hardware buffer |
| PSK | = | 140 ms – from extension buffer |

30 ms required to load each word to memory. This will occur every 8 characters or upon receiving a termination code.

*SEE NOTE 5

*SEE NOTE 6

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 +20 ms/word boundary | RTH | | |
| | | | LTN | | |
| TRCB | | 140 ms to 180 ms (avg. 150 ms) | RSN | | |
| TRF | | 120 ms + 10 ms/char transferred +20 ms/word boundary | LSN | | |
| | | | RPR | | |
| | | | LPR | | |
| TRBA | | 50 ms +10 ms/char | RPF | | |
| IRCP | | +20 ms/word boundary | LPF | | |
| | | | RTF | | |
| PAB | | 20 char/sec + 30 ms base time | 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

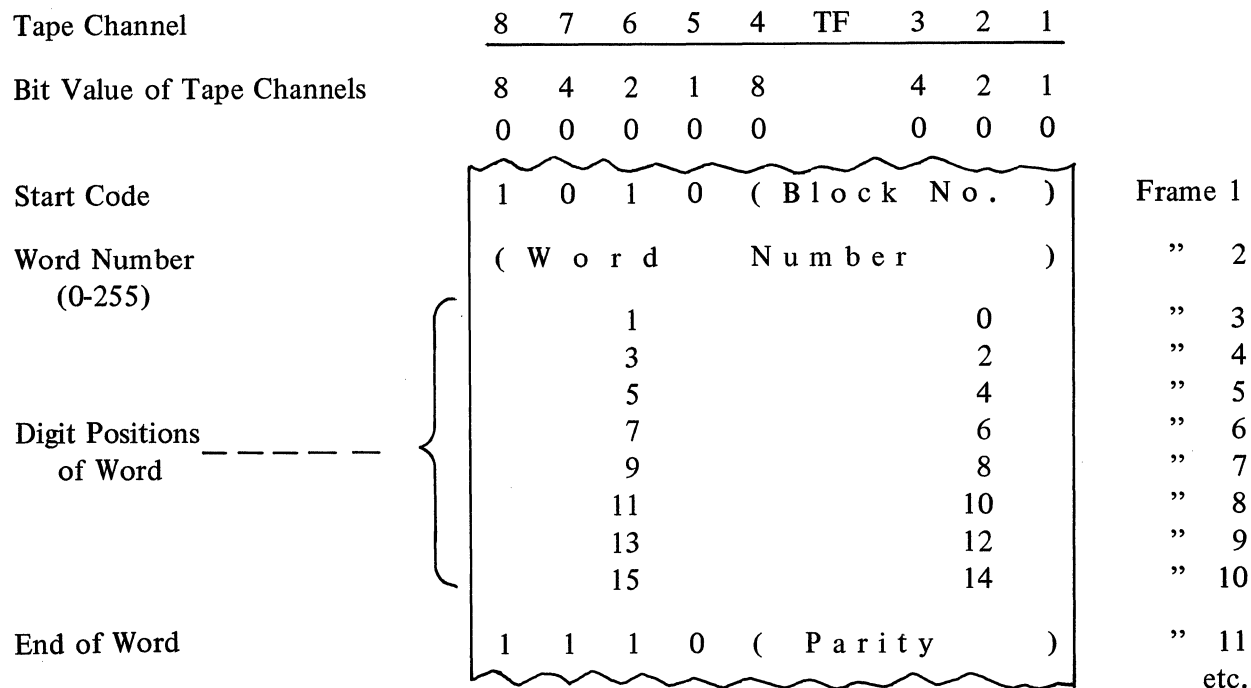
1. Set down the contents of the shift register.
2. When shift register is not equal to zero:
 - a. Examine the accumulator contents for timing purposes.
 - b. For each accumulator digit starting least significant digit.
For digit = 0 to 6 set down 10 ms
For digit = 6 to 9 set down 20 ms
 - c. Subtract 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

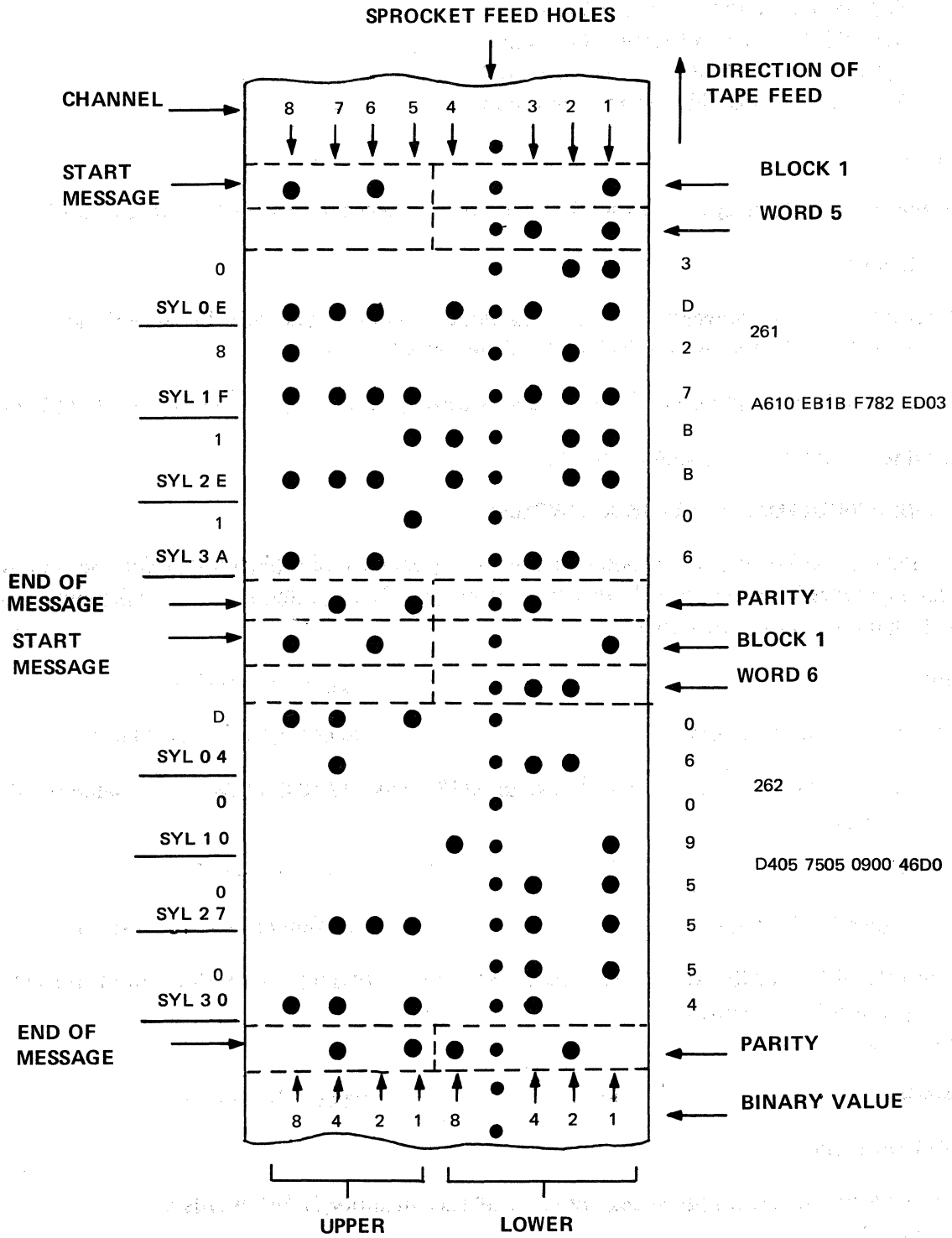


Figure 1.

REFERENCE:

1. OP code lower digit requires modification according to word syllable.

Modify thusly:

| | | |
|------------|--------|-------------------|
| Syllable 0 | add 0 | to previous value |
| Syllable 1 | add 4 | to previous value |
| Syllable 2 | add 8 | to previous value |
| Syllable 3 | add 12 | to previous value |

Example:

What is the machine language code for the instruction "Branch to word 625 syllable 2"?

Examine table K-2.

The OP code upper and lower for a branch to word numbers between 511 and 676 are 72. Reference 1 indicates add 8 to OP code lower (2+8=A) resulting in 7A.

The hexadecimal value for the difference between word 625 and word 511 (625-511=114) is 72.

The machine language code becomes 7A72.

ASSEMBLING PROGRAMS FOR 40 TRACK SYSTEMS

The information below is the only material which is needed in addition to Section 5 when attempting to assemble a symbolic program whose object program will function in an environment which allows extended memory macro programs.

Assembler I pages 5-1 to 5-16

L/TC (32 track) Environment Paper Tape I/O version

When EXTMEM prints on the journal, depress OCK 1 to indicate this is an assembly for a 40 track Style L. See page 5-1.

Assembler II pages 5-17 to 5-21

L/TC (32 track) Environment 80 Column Card I/O version

The control card \$ EXTMEM indicates that the object program will utilize firmware which allows a macro program of 767 words.

See page 5-17.

Assembler III pages 5-22 to 5-34

B 3500 Environment

The option EXTMEM will indicate that the size of user memory is 767 words.

See page 5-22.

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 USASCII table:

USASCII

Hexadecimal

| | |
|----|---|
| 30 | 0 |
| 31 | 1 |
| 32 | 2 |
| 33 | 3 |
| 34 | 4 |
| 35 | 5 |
| 36 | 6 |
| 37 | 7 |
| 38 | 8 |
| 39 | 9 |
| 41 | A |
| 42 | B |
| 43 | C |
| 44 | D |
| 45 | E |
| 46 | F |

21 Termination Code (1,E)

| | Sprocket Feed Holes | | | | | | | | | |
|------------------|---------------------|---|---|---|---|---|---|---|---|---|
| Channel | P | 7 | 6 | 5 | 4 | o | 3 | 2 | 1 | |
| Block Number | ● | | ● | ● | | o | | | ● | 1 |
| Word Number | ● | | ● | ● | | o | | | ● | 1 |
| | | | ● | ● | | o | ● | | ● | 5 |
| | | | ● | ● | | o | | ● | ● | 3 |
| | | | ● | ● | | o | | | | 0 |
| | | | ● | ● | | o | | ● | ● | 3 |
| | ● | ● | | | | o | ● | | ● | E |
| | | ● | | | | o | ● | | | D |
| | ● | | ● | ● | ● | o | | | | 8 |
| | ● | | ● | ● | | o | | ● | | 2 |
| Program Word | ● | ● | | | | o | ● | ● | ● | F |
| | ● | | ● | ● | | 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 | | | | |

153

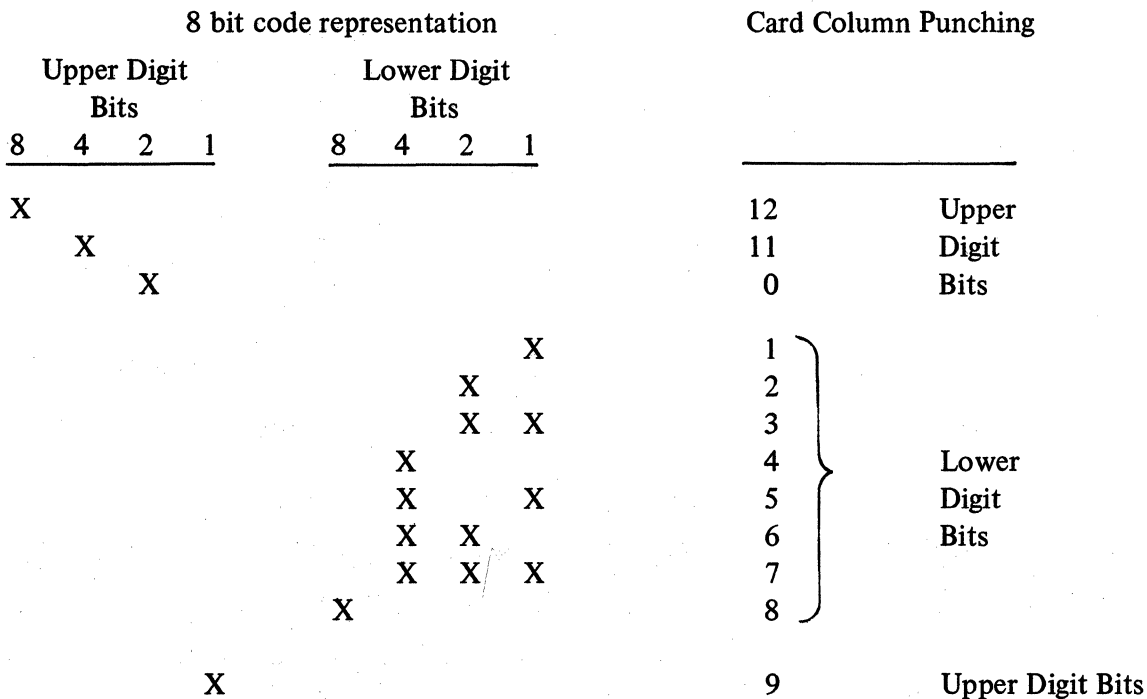
A610 EB1B F782 ED03

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 Block 0 = blank card column 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 DIGIT BITS | 8 | ■ ■ | | ■ | ■ | ■ ■ ■ ■ | ■ ■ | ■ ■ ■ ■ | ■ ■ | ■ ■ | ■ ■ | ■ ■ |
| | 4 | ■ | ■ ■ | ■ ■ | ■ ■ ■ ■ | ■ ■ | ■ ■ ■ ■ ■ ■ | ■ ■ ■ ■ ■ ■ | ■ ■ ■ ■ | ■ ■ ■ ■ | ■ ■ ■ ■ | ■ ■ ■ ■ |
| LOWER DIGIT BITS | 2 | 0 0 0 | 0 0 0 | 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 | 0 0 0 0 0 0 |
| | 1 | 1 1 1 | 1 1 1 | 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 | 1 1 1 1 1 1 |
| P U N C H | 2 | 2 2 2 2 | 2 2 2 2 | 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 | 2 2 2 2 2 2 2 2 |
| | 3 | 3 3 3 3 3 3 | 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 | 3 3 3 3 3 3 3 3 |
| V A L U E | 4 | 4 4 4 4 4 4 | 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 | 4 4 4 4 4 4 4 4 |
| | 5 | 5 5 5 5 5 5 | 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 | 5 5 5 5 5 5 5 5 |
| 1 C A R D C O L U M N | 6 | 6 6 6 6 6 6 | 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 | 6 6 6 6 6 6 6 6 |
| | 7 | 7 7 7 7 7 7 | 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 | 7 7 7 7 7 7 7 7 |
| P U N C H V A L U E | 8 | 8 8 8 8 8 8 | 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 | 8 8 8 8 8 8 8 8 |
| | 9 | 9 9 9 9 9 9 | 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 | 9 9 9 9 9 9 9 9 |
| PUNCH VALUE | | LBIVRP | 88 | 0 BAC71490 | B4C16000 | 10090DC2 | 70215197 | BA5D12BA | 182858C7 | E289CD22 | 43E38042 | |
| FOOTNOTE | | 1 | 23 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | |

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) |

ALPHABETICAL INDEX

A

Accumulator – 2.00.04
Accumulator Skip and Execute Instructions – 2.09.04
Add Constant to Accumulator – 2.05.02
Add from Magnetic Record Area to Accumulator – 2.19.07
Add to Accumulator – 2.05.01
Add to Index Register – 2.08.01
Add to Memory – 2.05.01
Advance Left and Right Platen – 2.04.03
Advance Left to – 2.04.03
Advance Left Platen – 2.04.03
Advance Line Instruction – 2.01.01
Advance Right Platen – 2.04.03
Advance Right to – 2.04.03
Alarm Instruction – 2.10.02
Alpha Constant Instruction – 2.01.02
Alphanumeric Data – 1-4
Alphanumeric Printing from Memory – 2.03.01
Alpha Word – 2.00.03, 2.02.03
Alternate Stacking Pocket – 2.18.04
A Parameter Label – 1-3
A Parameter – +/- Increment – 1-3
Arithmetic Instructions – 2.05.01
Assemblers – 5.00.00
Assembler Pseudo Instruction 2.01.02
Assembler VI and VIII – 5.01.05

B

B Parameter – 1-4
Branch – Decision Instructions – 2.09.01
Branch Unconditional – 2.09.01

C

Card Format Instruction – 2.01.04
Card In/Paper Tape Out Assemblers – 5.02.03
Card Instructions (80 Column) – 2.17.01
Card I/O Assemblers – 5.02.03
Change Flags Instructions – 2.07.02
Check Digit Instructions – 2.11.01
Clear Accumulator and Insert Constant – 2.05.04
Clear Memory Word – 2.05.04
Close Forms Handler Instruction – 2.04.01
Code Instruction – 2.01.05
Coding Form Heading – 1-1
Coding Form Page Number – 1-1
Compare Alphanumeric Instruction – 2.09.03
Computing Shift Factor – 2.05.05
Constant Data – 1-4
Control Registers – 2.13.01
C Parameter – 1-4
Cross Reference Table Between Program Definition Chart and Assembler III Output Sequence Number – 4-70

D

Data Comm Debugging – 3-3
Data Communications Instructions – 2.12.01
Data Comm Processor Operations – 2.14.02
Data Element Codes – 2.20.03
Decrement Index Register – 2.08.01
Define Instructions – 2.01.06
Delimiter – 2.20.06
D Flag Group – 2.12.07
Division Instruction – 2.05.05-2

Documentation Instruction – 2.01.07
Duplicate Through Column – 2.18.03-1

E

Eject Record – 2.19.09-1
Enable Numeric Keyboard Instruction – 2.02.01
End Instruction – 2.01.11
Enter Alpha Into Memory and Punch, Non-print – 2.18.01
Enter Alpha into Memory Instruction – 2.02.03
Enter Into Memory and Punch – 2.16.01-1
Equate Instruction – 2.01.11
Error Conditions – 2.20.05
Establish Record Areas – 2.01.07
Execute if Accumulator Zero – 2.09.04
Execute if Any Flag – 2.09.05
Execute if Digit Less Than Constant – 2.09.05
Execute if Every Flag – 2.09.05-2
Execute Instructions – 2.09.04

F

Field Identifier Codes – 2.12.06
Field Length – 1-3
Flags – 2.00.05
Flag Execute and Skip Instructions – 2.09.05
Flag Instruction – 2.07.02
Forms Control Instructions – 2.04.01
Forty Track Style Series L – K-1

G

GP300 Operation Code Modifications – 3-4

H

I

Increment Index Register – 2.08.05
Increment Receive Character Pointer – 2.12.03
Indicator Register Flags – 2.13.03
Insert Constant in Accumulator – 2.05.04

J

K

Keyboard Error Indicator – 2.02.01

L

Label – 1-2
Line Advance Instructions – 2.04.03
Line Discipline Format Registers – 2.14.01
Load Card Format Register – 2.17.01-2
Load Expected Broadcast Transmission Number – 2.12.09-2
Load Expected Group Transmission Number – 2.12.09-1
Load Expected Transmission Number – 2.12.09-1
Load Flags Instruction – 2.07.02
Load Index Register – 2.08.01
Load Keyboard Base Register – 2.12.04
Load Left Count Register – 2.04.03
Load Left Limit Register – 2.04.03
Load Memory from Card – 2.17.01
Load Position Register Instruction – 2.03.01
Load Print Numeric Base Register – 2.03.04
Load Program Key Base Register – 2.02.02

ALPHABETICAL INDEX (Continued)

Load Punch Count Register – 2.16.01-3, 2.18.03
Load Receive Address – 2.12.08
Load Receive Buffer Register – 2.12.03
Load Right Count Register – 2.04.03
Load Right Limit Register – 2.04.03
Load Send Address – 2.12.08
Load Send Transmission Number – 2.12.09
Load Shift Register – 2.05.05
Load Stripe Count Register – 2.19.09-2
Load Stripe Format Registers – 2.19.01
L/TC Assembler III B 3500 Version – 5.03.02

Environment – 5.03.02
Error Detection – 5.03.06
Library Tape Input – 5.03.02
MCP Control Cards – 5.03.04
Operating Instructions – 5.03.05
Option Control Cards – 5.03.04
Output – 5.03.07
Utility Routine – 5.03.08

L/TC Assembler IV B 5500 Version – 5.04.02

Environment – 5.04.02
Error Detection – 5.04.07
MCP Control Cards – 5.04.02
Operating Instructions – 5.04.04
Operation – 5.04.07
Option Control Cards – 5.04.03
Output – 5.04.07

L/TC Assembler V – B 300 Version – 5.05.00

Control Cards – 5.05.00
End – 5.05.06
Environment – 5.05.00
Error Messages – 5.05.07
Input – 5.05.00
Operating Instructions – 5.05.06
Output – 5.05.00

L/TC Character Set – D-1

L/TC 80-Column Card Input Assemblers – 5.02.00

L/TC Paper Tape Assemblers – 5.01.00

Environment – 5.01.00
Keyboard Mode – 5.01.06
Phase I – 5.01.02
Phase I – Condensed Operating Instructions – 5.01.07
Phase I – Diagnostic Facilities – 5.01.08
Phase II – 5.01.09
Phase II – Condensed Operating Instructions and Index – 5.01.10
Phase II – Error Detection and Indication – 5.01.11

M

Machine Code for Transaction Code Translation Instruction – 2.21.06
Magnetic Record Format Instruction – 2.19.02
Magnetic Unit Record Flag – 2.19.04
Main Memory Processor – 2.14.03
Mask Control Code – 2.03.05-1
Mask Flags – 2.03.05-2
Mask Instruction – 2.01.11
Mask Word – 2.03.05
Memory Organization – 2.00.01
Memory Word Organization – 2.00.02
Miscellaneous Instructions – 2.10.02
Modes for Printing – 2.03.01
Modify by Index Register – 2.08.06
Modify by Punch Count Register – 2.16.02
Multiplication Instruction – 2.05.05-1
Multiply with Rounding – 2.05.05-2

N

NOP Instruction – 2.10.02
Note Instruction – 2.01.12
Number Instruction – 2.01.12, 2.01.13
Numeric Keyboard Instructions – 2.02.01
Numeric Printing Instructions – 2.03.06
Numeric Word – 2.00.02

O

Open and Close Instruction – 2.04.01
Open Forms Handler Instruction – 2.04.01
Operation Code – 1-2
Operation Control Keys – 2.02.02
Origin Instruction – 2.01.14, 2.01.16

P

Page Instruction – 2.01.15
Paper Tape Instructions – 2.15.02
Platen Control Register Instructions – 2.04.03
Point to Point Line Discipline – 2.13.01
Position Table – 2.20.03
Power On and Off Instruction – 2.03.09
Print Alpha and Punch – 2.16.01-1, 2.18.02
Print Alpha from Card Read Area – 2.17.01-2
Print Alpha from Magnetic Record Area – 2.19.06
Print Alpha from Memory (D.C.) – 2.12.05
Print Alpha from Memory Instruction – 2.03.01
Print Alpha from Memory, Non-Print – 2.16.01-1
Print and Punch Alpha from Card Read Area – 2.17.01-3
Print and Punch Numeric – 2.16.01-2, 2.18.02-2
Print and Punch Numeric, Shift Ribbon if Minus – 2.16.01.3, 2.18.02
Print and Punch Numeric, Shift Ribbon if Plus – 2.16.01.3, 2.18.02
Print Format (Mask) Word – 2.03.05
Print Format Word – 2.00.03
Print Instructions – 2.03.01
Program Debugging – 3-3
Program Execution – 2.00.03
Program Identification – 1-1
Programing Example – 4-1
Program Keys – 2.02.02
Program Word – 2.00.03
Program Writing – 3-1
Punch Alpha from Card Read Area, Non-print – 2.17.01-3
Punch Code Instruction – 2.16.01-2, 2.18.02-3
Punch Feed Codes – 2.16.02
Punch Flags – 2.16.02
Punch Numeric, Non-print – 2.16.01-3, 2.18.02-3

Q

R

Read Alpha and Print – 2.15.02
Read Alpha Into Memory and Print – 2.15.02
Read Alpha Into Memory and Punch, Non-print – 2.15.02-1
Read Alpha Into Memory, Non-print – 2.15.02
Read Alpha Into Memory, Print and Punch – 2.15.02-1
Read Alpha, Print and Punch – 2.15.02-1
Read Card Instruction – 2.17.01
Reader Flags – 2.16.02
Reading Punched Cards – 2.17.01
Read Numeric Into Accumulator – 2.15.02-1
Read Record Instruction – 2.19.05
Receive Ready State – 2.12.01

ALPHABETICAL INDEX (Continued)

Record Align – 2.19.09
Record Alignment Errors and Flag Indications – 2.19.10
Region Instruction – 2.01.16
Remarks – 1-4
Reserve Card Buffer Instruction – 2.01.03
Reset Flags – 2.07.03
Retract Record – 2.19.09-2
Retrieve Expected Broadcast Transmission Number – 2.12.09-1
Retrieve Expected Group Transmission Number – 2.12.09-1
Retrieve Expected Transmission Number – 2.12.09-1
Retrieve Pointer Register – 2.12.09
Retrieve Receive Address – 2.12.08
Retrieve Send Address – 2.12.08
Retrieve Send Transmission Number – 2.12.09
Retrieve Transmission Header – 2.12.09-2
Retrieve Transmission Number – 2.12.09-2
Retrieve Two/Four Wire Register – 2.12.09-3
Ribbon Shifts – 2.03.08

S

Sequence Number – 1-1
Set Flags – 2.07.03
Set Receive Character Pointer – 2.12.03
Set Send Character Pointer – 2.12.04
Shift Off Instruction – 2.06.02
Shift Off with Sign – 2.06.02-1
Single Character Print Instruction – 2.03.07
Skip and Execute – TC 700 – 2.09.06
Skip if Accumulator Zero – 2.09.04
Skip if Any Flag – 2.09.05-2
Skip if Digit Less Than Constant – 2.09.05
Skip if Every Flag – 2.09.05-3
Skip Instructions – 2.09.04
Skip to Column – 2.18.03
Stop Instruction – 2.10.03
Storage Area – 2.20.04
Subroutine Jump – 2.09.02
Subroutine Return – 2.09.02
Subtract Constant from Accumulator – 2.05.06-1
Subtract from Accumulator – 2.05.06-1
Subtract from Memory – 2.05.06-1
Subtract Magnetic Record Area from Accumulator – 2.19.07
Symbolic Programming Procedures – 3-1

T

TC 700 – L-1
Transaction Code Translator – 2.21.02
 Automatic Codes – 2.21.03
 Code Modification – 2.21.04
 Error Condition – 2.21.05
 Programming Example – 2.21.09
 User Program Requirements – 2.21.08
 Word 576 – 2.21.08
Transfer Accumulator to Index Register – 2.08.05
Transfer Accumulator to Memory Instruction – 2.12.05
Transfer Alpha from Magnetic Area to Memory – 2.19.08
Transfer Alpha from Memory to Magnetic Record Area – 2.19.09
Transfer Alpha Instruction – 2.12.04
Transfer Card Columns to Memory as Alpha – 2.17.02
Transfer Card Field to Accumulator As Numeric – 2.17.01-3
Transfer Character to Memory Instruction – 2.12.05.3
Transfer from Accumulator to Magnetic Record Area – 2.19.09
Transfer Numeric from Magnetic Area to Memory – 2.19.08
Transfer Receive Buffer to Record Area – 2.12.04
Transfer Remainder to Accumulator – 2.06.02
Transfer Send Record Area to Memory Instruction – 2.12.05-1

Transfer to Accumulator as Numeric – 2.06.01
Transfer to the Accumulator – 2.06.01
Transfer to Memory – 2.06.01
Translation Table Format – 2.21.02
Transmit Ready State – 2.12.01
Type and Punch Instruction – 2.16.01-1, 2.18.01
Type Instruction – 2.02.03
Type Into Memory Instruction – 2.02.03
Type Into Memory, Punch and Print – 2.16.01-1, 2.18.01
Typewriter Keyboard Instructions – 2.02.03

U

Unpacking Routine Programming Requirements – 2.20.04

V

Variable Length Field – 2.12.06

W

Word Instruction – 2.01.17
Write Record Instruction – 2.19.04

X

Y

Z



*Wherever There's
Business There's*



Burroughs